Continual class-incremental learning for autonomous object recognition in image sequences

Ruiqi Dai

N°d'ordre NNT : 2022ISAL0072

THESE de DOCTORAT DE L'INSA LYON,
membre de l'Université de Lyon

Ecole Doctorale N° 512
(InfoMaths)

Spécialité/ discipline de doctorat :
Informatique

Soutenue publiquement le 14/09/2022, par :
(Ruiqi DAI)

Continual class-incremental learning for autonomous object recognition in image sequences

Devant le jury composé de :

Nom, prénom	grade/qualité	établissement/entreprise	
Vincent Nicole	PR	Université de Paris Cité	Examinatrice
Hudelot Céline	PR	CentraleSupélec Paris	Examinatrice
Reignier Patrick	PR	Univ. Grenoble Alpes	Rapporteur
Château Thierry	PR	Université Clermont-Auvergne	Rapporteur
Duffner Stefan	MCF-HDR	INSA LYON	Directeur de thèse
Armetta Frederic	MCF	Université Claude Bernard Lyon 1	Co-encadrant
Lefort Mathieu	MCF	Université Claude Bernard Lyon 1	Co-encadrant
Guillermin Mathieu	MCF	Université catholique de Lyon	Co-encadrant

Département FEDORA – INSA Lyon - Ecoles Doctorales

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** **https://www.edchimie-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage secretariat@edchimie-lyon.fr	**M. Stéphane DANIELE** C2P2-CPE LYON-UMR 5265 Bâtiment F308, BP 2077 43 Boulevard du 11 novembre 1918 69616 Villeurbanne **directeur@edchimie-lyon.fr**
E.E.A.	**ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE** **https://edeea.universite-lyon.fr** Sec. : Stéphanie CAUVIN Bâtiment Direction INSA Lyon Tél : 04.72.43.71.70 secretariat.edeea@insa-lyon.fr	**M. Philippe DELACHARTRE** INSA LYON Laboratoire CREATIS Bâtiment Blaise Pascal, 7 avenue Jean Capelle 69621 Villeurbanne CEDEX Tél : 04.72.43.88.63 **philippe.delachartre@insa-lyon.fr**
E2M2	**ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE, MODÉLISATION** **http://e2m2.universite-lyon.fr** Sec. : Bénédicte LANZA Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 secretariat.e2m2@univ-lyon1.fr	**Mme Sandrine CHARLES** Université Claude Bernard Lyon 1 UFR Biosciences Bâtiment Mendel 43, boulevard du 11 Novembre 1918 69622 Villeurbanne CEDEX **sandrine.charles@univ-lyon1.fr**
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTÉ** **http://ediss.universite-lyon.fr** Sec. : Bénédicte LANZA Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 secretariat.ediss@univ-lyon1.fr	**Mme Sylvie RICARD-BLUM** Institut de Chimie et Biochimie Moléculaires et Supramoléculaires (ICBMS) - UMR 5246 CNRS - Université Lyon 1 Bâtiment Raulin - 2ème étage Nord 43 Boulevard du 11 novembre 1918 69622 Villeurbanne Cedex Tél : +33(0)4 72 44 82 32 **sylvie.ricard-blum@univ-lyon1.fr**
INFOMATHS	**INFORMATIQUE ET MATHÉMATIQUES** **http://edinfomaths.universite-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage Tél : 04.72.43.80.46 infomaths@univ-lyon1.fr	**M. Hamamache KHEDDOUCI** Université Claude Bernard Lyon 1 Bât. Nautibus 43, Boulevard du 11 novembre 1918 69 622 Villeurbanne Cedex France Tél : 04.72.44.83.69 **hamamache.kheddouci@univ-lyon1.fr**
Matériaux	**MATÉRIAUX DE LYON** **http://ed34.universite-lyon.fr** Sec. : Yann DE ORDENANA Tél : 04.72.18.62.44 yann.de-ordenana@ec-lyon.fr	**M. Stéphane BENAYOUN** Ecole Centrale de Lyon Laboratoire LTDS 36 avenue Guy de Collongue 69134 Ecully CEDEX Tél : 04.72.18.64.37 **stephane.benayoun@ec-lyon.fr**
MEGA	**MÉCANIQUE, ÉNERGÉTIQUE, GÉNIE CIVIL, ACOUSTIQUE** **http://edmega.universite-lyon.fr** Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bâtiment Direction INSA Lyon mega@insa-lyon.fr	**M. Jocelyn BONJOUR** INSA Lyon Laboratoire CETHIL Bâtiment Sadi-Carnot 9, rue de la Physique 69621 Villeurbanne CEDEX **jocelyn.bonjour@insa-lyon.fr**
ScSo	**ScSo*** **https://edsciencessociales.universite-lyon.fr** Sec. : Mélina FAVETON INSA : J.Y. TOUSSAINT Tél : 04.78.69.77.79 melina.faveton@univ-lyon2.fr	**M. Christian MONTES** Université Lumière Lyon 2 86 Rue Pasteur 69365 Lyon CEDEX 07 **christian.montes@univ-lyon2.fr**

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Table of contents

List of Figures

List of Tables

Abstract

For an agent, it is very challenging to autonomously elaborate and make use of a visual representation of its open environment. It is necessary to introduce new categories for unseen objects, and recognize already seen objects to refine and make evolve the representation, which is even more difficult when undergoing uncertainty and variability under uncontrolled operation conditions. The autonomy of the agent in classification, its adaptation to a changing environment as well as continual representation building are important properties of such a dynamic machine learning system. However, in some of the existing systems, especially with neural networks, the acquired representation tends to gradually drift away from initial observations which causes catastrophic forgetting. Another challenge comes from the decision whether incoming observations are new or belong to already seen objects. One has to detect novelty and introduce a new concept or class if necessary, and maintain the already acquired representation, i.e. recognize the object and make use of new observations appropriately, which is part of the more general problem of finding a meaningful and robust representation under the stability-plasticity dilemma.

In this thesis, we address these challenges by adopting a state-of-the-art continual unsupervised representation learning model based on a specific Variational Auto-Encoder (VAE) architecture that we extended and applied on sequences of images showing different objects. Assuming that, in a realistic environment, there is some continuity in the sequence of object observations, the first contribution is an algorithm that robustly detects when object class changes occur in the image stream based on the dynamic evolution of the observation likelihood. It is shown that the thus obtained clusters in the representation space are more consistent with real objects and therefore facilitate the classification of known objects. The second contribution replaces the first approach and further increases the autonomy of the model by introducing an algorithm based on the Hotelling t-squared statistical hypothesis test that is able to continuously detect class changes and simultaneously decides if the currently observed object is novel or already present in the learnt representation. Extensive experimental results on several image benchmarks show that this self supervised continual learning approach is able to build representations that favor the autonomy of the agent by minimising supervision while maintaining a high object recognition accuracy compared to the state of the art.

Résumé

Pour un agent, apprendre une représentation visuelle de façon autonome en environnement ouvert est une tâche complexe. Elle nécessite pour un agent de reconnaître les catégories d'objets déjà perçues et apprises ou d'introduire de nouvelles catégories d'objets à sa représentation. Ce problème s'avère difficile de par la variabilité des objets perçus au sein d'une même catégorie, mais également à cause du caractère imprévisible des séquences d'objets auxquelles est soumis l'agent. Pour le système ainsi formé, les capacités à faire évoluer une représentation de façon dynamique et continue représente un défi pour les méthodes du domaine de l'apprentissage automatique (machine learning). Pour certains modèles, particulièrement à base de réseaux de neurones, la représentation apprise à tendance à progressivement s'éloigner des observations initiales pour finalement mener à l'oubli de l'expérience la plus ancienne (catastrophique forgetting) résultant du dilemme de stabilité-plasticité. Une autre difficulté provient du couplage étroit entre la dynamique de reconnaissance du statut des objets (connus/inconnus) qui exploite la représentation, et la représentation elle-même alimentée et mise à jour par ce processus de reconnaissance.

Dans cette thèse, nous adressons ces défis en adoptant une méthode de l'état de l'art dédiée à l'apprentissage continu et non supervisé basé sur un autoencodeur variationnel spécifique (VAE), pour l'étendre et l'appliquer à l'apprentissage d'objets à travers des séquences d'images de différents objets. Dans ce travail, l'accent est mis sur le fait qu'en environnement réaliste, on peut supposer une certaine continuité dans les objets perçus. Une première contribution introduit un algorithme qui détecte de façon robuste les changements de catégories d'objet dans le flux d'images suivant l'évolution des probabilités d'observation. Nous montrons que les groupes d'objets appris au sein de la représentation permettent une amélioration de la consistance avec les objets perçus ce qui facilite la reconnaissance des objets issus de catégories connues. Pour la deuxième contribution de ce travail, nous améliorons l'autonomie du modèle en introduisant un algorithme exploitant le test statistique du T carré de Hotelling qui est capable de détecter de façon continue les changements de catégories d'objets, tout en permettant d'identifier un objet encore inconnu pour le modèle. Des expérimentations étendues sur différents jeux d'apprentissage d'images montrent que cette approche d'apprentissage continu auto-supervisée permet d'accroître l'autonomie de l'agent, en minimisant les besoins de supervision extérieure tout en conservant un niveau de reconnaissance élevé par rapport à ce qui est présenté dans la littérature.

Chapter 1

Introduction

Artificial intelligence takes its earliest origins at the Dartmouth conference in 1956. It has evolved throughout time while incorporating other research domains, such as cognitive theories, neuroscience, psychology, and philosophy. It has been applied in many disciplines: i.e. automatic diagnostics of cancer in the medical research domain, fraud detection in the financial field, etc.. The fields of application might vary, but building an autonomous agent seems to be one of the essential problems in AI research. In AI, one of the agent's most essential properties is autonomy, which is defined as the ability to learn or make decisions in an unknown and unconstrained environment. While learning with autonomy can be challenging, since the agent may have to adapt to changes in a dynamic environment.

AI also faces challenges related to the explainability and interpretability of models. The historical artificial intelligence models derived a part of their origins from biology, but many of them are still black box-like even if AI makes progress. Consequently, they have a limited explanation for the decisions they make. Besides the explainability problem, the model has another limitation: its generalization.

The development of a model is usually specific to a problem or use case, and it is dependent on both data and context. It is crucial for the model to have the ability to generalize so that it can be effective on unseen but similar data points.

Although many issues remain open, machine learning has advanced with satisfying performance in various tasks, in particular classification or clustering approaches using statistics or similarities to distinguish different objects. But they are often offline. The problem of unsupervised continual learning is particularly interesting and fundamental. It raises questions both theoretical (what is the purpose for such a learning approach? What are the cognitive properties supporting it?) and applicative (How to provide concrete implementations for AI?).

Continual learning, or life-long learning as a synonym, is to learn continually and incrementally from the input data stream. In contrast to common offline deep learning scenarios, the unsupervised continual learning problem targets a higher level of autonomy for the agent, which can be seen as the first step towards AI applied to more general cases.

In this thesis, we will target this problem while considering the notion of continuity presented by the notion of class-incremental, which indicates that an object will be visible for at least a certain duration of time. This continuity hypothesis can be found in many fields, with slightly adapted or modified definitions depending on the context to which it is applied: in robotics, for example, it is called spatio-temporal continuity; or from a more general standpoint, works in the literature taking videos as input. In this thesis, this continuity takes place in the way we present the objects to the system. Objects of the same class category are shown consecutively, while increasing the class introduced to the system in different ways. As a first step, this hypothesis of continuity can simplify the problem to which our proposal contributes and will be further studied in more complicated scenarios.

In this chapter, we take the initiative to introduce some notions in cognition theory that are tightly related to our targeted problem. Although some works in AI (those that are bio plausible, for example) draw inspiration from cognitive theories, proposals in the field of machine learning do not often emphasize the implications for AI. This can guide our work through the combination of principles from a different discipline, cognitive development. From an AI positioning, it helps justify choices for concrete implemen-

tations. In this way, we can identify the required properties for an AI and match them with the proposals in the machine learning field, which allows us to define and motivate our contribution in the dedicated areas. Our objective is to construct an autonomous agent that is capable of learning the representation of the unknown environment through the sources of information that are made available to him.

In this chapter, we first conceptualize our work in section 1.1 from the perspective of cognitive development. This section introduces the notion of autonomy for representation acquiring, and discusses the role of perception in this process. The role of continuity is studied, and the continuity-based scenarios we simulate are presented. Then, we present an overview of machine learning methods advances related to cognitive development. From this positioning, we describe our contributions in section 1.2, and we will give a brief outline of this thesis in section 1.3.

1.1 Learning and cognitive development

For an autonomous agent, autonomy is essential and requires the agent to learn to construct the representation of the environment, as we will show in section 1.1.1. In this section, we will start with the notion of ontology, which is part of the cognitive development. For an autonomous agent, we highlight several frequently targeted properties. In section 1.1.2, we will start by showing different sources of information that are accessible to agents. The agent can learn to construct a representation of the environment by exploiting these sources. Specifically, the actions and interactions are supported by cognition theories, thereby demonstrating the originality of this work. Among all the possible sources, perception is a key point for machine learning. Throughout this study we choose to focus exclusively on passive perception as a crucial component of the cognition theories and as a means to acquire knowledge. Meanwhile, on the contrary, sensorimotor theories treat perception and action as inseparable.

Nevertheless, the perception remains an important process to integrate sensory inputs, in the continual learning context, just as in many other machine learning or computer vision problems, as it is the first step towards a more complicated cognitive model. The chosen approach to categorize objects for object recognition is a purely statistical one. We will discuss the advantages of using such an approach, meanwhile its limitation regarding the separation of easily confused objects by shape in the absence of possible interactions with the objects that can provide additional information about them, such as object utility. We will elaborate on our objective of continual learning and the potential intrinsic limitation of purely statistical criteria in this first phase of learning from perceptual information in section 1.1.3, which will be shown with a concrete example for illustration.

1.1.1 Autonomy and representation acquiring

When an agent learns to gradually acquire an ontology, the cognitive theory of learning provides theoretical support for how the representation is built and emphasizes the role that action and interaction play in learning. We will start by presenting the notion of ontology and some cognitive theories that explain learning from a cognitive development aspect. Then we show how it is related to different properties of AI and the sources of information and interaction available to autonomous agents.

1.1.1.1 Ontology : creation, recognition

The word ontology take its origin from philosophy. It refers to the notion of what objects are and their associated relationship. Ontology is a common concept that people use for an object. The notion of ontology is essential in cognitive development. According to the similarity of an external source of information and the agent's experience or acquired knowledge, the ontology needs to be created in the case of a novel concept, or on the contrary, what the agent perceived can be recognized and associated to an existing one. This comes to the notion of classification, but its difference with the notion of ontology is that the ontology presents concepts by following certain hierarchies. According to certain criteria, the ontology defines subdivisions between concepts and things. When one develops cognitive skills, one can then perform categorization that tells an object is distinct from another object, which is an important

cognitive activity. One must group the same object and separate different objects in order to categorize them accordingly. This is a cognitive process that needs ontology and at the same time classification.

1.1.1.2 AI and classification

Construction of the ontology and representation of the environment remained a challenging problem, which was partly adjacent to the definition of classes. One needs a general understanding of what is the same and what is different for the construction of ontology.

The notion of objects and classes or categories can therefore be defined. As we illustrate in figure 1.1, data points can be partitioned according to their ontology definitions. A *class* or a *category* gather all the different objects that are similar to each other. If two objects belong to distinct classes, then they are different objects. On the contrary, objects can belong to the same category. In this case, they can be visually distinguished from each other, for example through size, color, or shape. For example, a "red cup" and a "white cup" both belong to the class/category "cup" but are nevertheless different. Besides, the ontology often presents hierarchical properties. For example, "cups" and "bowls" both belong to "containers". This notion of hierarchies models more refined relationships between classes in the learning theory and cognition. In humans' ontology, similar objects will be categorized and affected the same class. For example, "cup" will give similar responses in our eyes. Therefore, a cup can be distinguished from a pencil (which has a thoroughly different appearance). This raises the notion of classification. Ontology construction itself remains challenging and contains other issues in addition to classification. But in this thesis, we will mainly focus on the creation and recognition aspects of ontology, or the ability of a model to distinguish between different classes. Therefore, it is possible to assimilate it to a classification problem.

In AI and in machine learning, classification is a common task. In classification, the model needs to predict the category of input as close as possible to its ground truth label. We expect the agent to correctly affect the images it receives with a cluster. The predictions should not be arbitrary, but rather be based on underlying similarities or statistics.

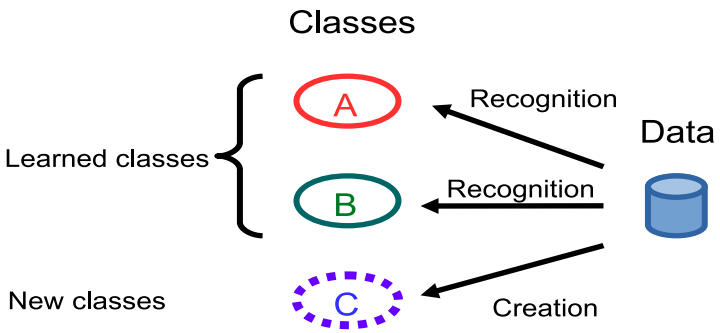


Figure 1.1: Ontology creation and cognitive development: for the construction of ontology, a class is either recognized (learned classes A and B encercled in red and green) or rejected to create a new class (the class C encercled in blue). The notion of ontology is tightly related to the notion of categorization or classification in the field of AI.

Therefore, we will define several terminologies in AI concerning learning. Approaches that learn with supervision or need the ground truth label to train the models are referred to as *supervised learning*. On the contrary, when annotated examples are inaccessible or learning is without supervision or ground truth labels, *unsupervised learning* is used to refer to these approaches without external information of the ground truth label. Many of these approaches in the machine learning or deep learning field have supposed i.i.d. (independent and identically distributed) data when the entire dataset is accessible to the model and the input data is stationary. Images inside the dataset are shuffled and presented randomly to the model, and this learning scenario is also referred to as offline. In the following sections, we give

examples of supervised and unsupervised approaches in artificial intelligence.

1.1.1.3 Autonomy and underlying required properties

In artificial intelligence, the targeted autonomy of the agent often requires several properties, for example, a memory to enhance performances of continual learning. In this section, we will show some of the most important properties of autonomous agents.

Autonomy, for an artificial agent, refers to the ability to learn in an open and unconstrained environment, where the changes in the environment and its dynamic is unrevealed to the agent and requires the agent to learn and adapt by itself. For example, an autonomous vehicle has to recognize and avoid obstacles on roads in the city or muddy roads in the countryside with hollows or with stones. Although it may not be trained to avoid all kinds of obstacles in every context, we will still expect it to adapt to possible new contexts never seen before and avoid obstacles to drive safely. This is related to another notion of the capacity to generalize of an artificial intelligence system. That is, during learning, the model has access to a certain amount of data, and it is important that the model extracts information that can be extrapolated onto unseen data, or synthesized. Another common property required for autonomy is plasticity, which refers to the ability to adapt to changes or to integrate new concepts. In addition to plasticity, there is also stability, which is the ability to maintain previously acquired information. Both plasticity and stability can find their origin in neuroscience from the human brain, which forms the well-known stability-plasticity dilemma [113], that the system should be stable enough not to lose previously acquired knowledge meanwhile being plastic to acquire new ones. A common difficulty that many systems face is catastrophic forgetting, which is closely related to the stability-plasticity dilemma. The catastrophic forgetting itself is a barrier for common offline approaches due to parameter sharing between new/old classes or tasks. For this reason, they have usually a quite weak performance in the online and continual learning scenario. The "optimal" agent for continual learning would be a memory system, as [82] mentioned.

Memory is an important notion, both in cognitive models and in artificial intelligence. A simple way to define memory is a component used to store useful information that are crucial or necessary parts to construct representation of the environment. The stored information can be further integrated into training data to enhance the learning on certain tasks, which data are no more accessible to the agent. The human brain is accompanied by dual memory systems [88]: short-term memory (hippocampus) and long-term memory (episodic memory, neocortex, consolidates short-term memories). The memory allows the capacity of human learning in a one-shot manner [108] while associating with another related mechanism such as replaying certain sequences or episodes of memory. The implication of memory is often studied, as it is an important aspect of both understanding human learning and its possible implications within deep learning. Besides, humans can manipulate information stored in the memory, referred to as the mechanism of working memory, which has inspired various artificial intelligence works. For example, the model of Differentiable Neural Computer (DNC) [59] from which information is read, written, retrieved, and copied. This can be seen as a way to battle the model forgetting previously acquired knowledge.

The capacity to generalize is indispensable in common approaches in machine learning and deep learning that use images as input, should the model be properly trained to avoid overfitting. But also they often operate under the i.i.d. data assumption for which the dataset is stationary, and images in the dataset are shuffled and presented randomly to the model. The autonomy can not be fully guaranteed in an open environment since the dynamic of such machine learning models is not ensured to adapt to these environments. Moreover, when the input is not i.i.d., offline models will have poor performance, being plastic enough to acquire new classes or tasks, but their optimization will also degrade those of the old ones. Therefore, offline approaches are not suited for an online or incremental learning scenario. Later, in section 1.1.4, we will briefly introduce existing machine learning and deep learning approaches to show their advantages and drawbacks concerning these required properties.

1.1.2 Perception as a keypoint of cognition

In this section, we will introduce the main source of information in the context of artificial intelligence. We will present the first and the major process of learning, which is the perception, that agent perceives

images of objects and the environment. Other sources include the action, which indicates that the agent can take action inside the environment and will get feedback. Besides, the agent can interact with the environment or interact with other agents.

We will equally introduce cognition theories that are tightly related to artificial intelligence. We will highlight theories of cognition development and their relationship to different artificial intelligence works.

1.1.2.1 Available interactions for an agent

In artificial intelligence, autonomous agents can have several sources of information to learn from. We illustrate these sources in figure 1.2, with a dedicated illustration for each one of the possible sources.

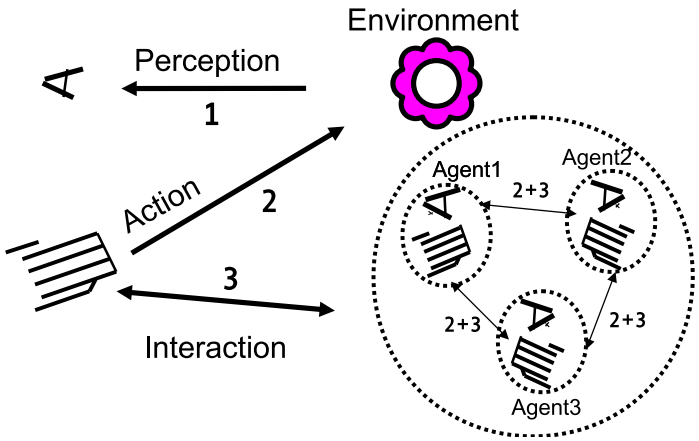


Figure 1.2: Sources of information accessible for an agent: the passive perception as source 1 (for humans, through eyes and sensors, for example, for autonomous agents) ; source 2 action illustrated by a hand) and source 3 interactions that can take place between several agents that communicate with each other or transfer knowledge through other forms of social activities.

The most basic source of information is perception which refers to the agent acquiring information about the environment, for example, objects in 3D. On the right hand of figure 1.2, we present the perception through the data that are received by the agent, source 1, through its sensors (eyes in humans' cases, but in the context of an autonomous agent, it will be, for example, a camera or other types of sensors or modalities). In the domain of computer vision and machine learning, usually, the agent receives images from sensors like the camera in 2D or in video streams or directly uses a public dataset of images as input of the agent. From the perception, the agent builds a visual representation of the environment.

Apart from the perception, the agent can take action and learn from the feedback of the environment, thanks to a reward for example. In figure 1.2, this is represented by source 2, marked as a hand that can take actions. A simple example is that an agent can learn the representation of an object by interacting with the object. Additionally, in sensorimotor activities, perceptual activities and other interactions are tightly coupled or cannot be clearly separated.

Besides the interaction with the environment, there are also other possible interactions, for example, social interaction. In figure 1.2, the interaction is represented by the source 3. The agent can further have social interactions to transfer acquired knowledge from one agent to another, illustrated in figure 1.2 (with encircled agent 1, agent 2, agent 3). Possible interaction includes interaction with human [35] (or social interactions) to complete perception and to guide learning. To learn the identity of an object, the agent can learn from social activities or a human teacher. Or social interaction, for example, if an agent does not know what a "cup" is, it can learn from another agent who has already constructed the ontology enriched enough to include the notion of a "cup". Through social interaction, the ontology of one agent

can be transferred to another agent so that they share the same notion of ontology.

Recent advances in reinforcement learning show more integration of the cognitive theory and different mechanisms inspired by neuroscience. In reinforcement learning, agents learn to perform tasks and take actions to maximize the reward. Compared to the common computer vision field, the action and interaction can enrich its constructed representation

1.1.2.2 Some cognition theories encourage interaction

As we have mentioned, some cognitive theories from psychology and neuroscience have inspired artificial intelligence, leading to its success today, as reviewed in [64, 21]. For this reason, we give a detailed focus on the cognition theory related to the learning of an agent.

We hereby show the essential problems and definition of cognitive development, its construction from a psychological or neuronal point of view, and its applications. We also include the constraints that we consider in this study.

Cognition can be seen as among the first steps toward the definition of intelligence. Cognition theories explain how the mind works, including different mechanisms such as perception, action, reasoning, or in problems solving and, decision-making, and planning. It has thus inspired the advances of AI. The cognition theories show how the agent understands and adapts to a dynamic environment [90]; or how the agent constructs representation and reasons based on them. This can also be extended to the understanding of the phenomena of conscience, that will not be addressed in this thesis.

An essential problem in cognition is to build symbols that are not arbitrary but meaningful. In the literature, this is also referred to as the symbol grounding problem. A famous example is in the "Chinese Room Argument" [148]. In the "Chinese Room Argument", a man is isolated in a room, without truly knowing how to speak Chinese, but a dictionary is given to him with answers to different questions. With the dictionary, the isolated man can somehow answer all the questions that were given to him in Chinese from outside the room. But he does not truly know how to speak Chinese. Similarly, an autonomous agent can build a representation of the environment from its perception or other sources of information, but will it be able to understand the representation it builds, or are they merely symbols that the program manipulates? Notably, it is difficult for the agent to understand the meaning of these symbols.

In the literature, for example, the sensorimotor theory of cognition development considers interaction as the core of cognition. The agent learns based on information received from the sensors, by interacting with the environment, it will learn the existence of an object and the associated perceptual invariance of the object. It can thus construct the representation of the environment and use this invariance to actively explore and compose its perception.

Different points of view in cognition theory can be divided into several categories: for example, the *relativism* believes that the truth is relative, the reasoning is relative to some standpoint which can be the observer, the place, or the cultural context. For *constructivism*, learning is considered a dynamic process of adaptation and construction in order to understand the meaning of different situations. Learning is a gradual and developmentally constructed process and can be without prior knowledge. Learning can be active when an agent gets the feed-back of its actions from the environment. Piaget [128], considers that learning occurs in several stages: the first stage is the sensorimotor stage that occurs before 2 years old. In this stage, infants discover and explore the environment via actions and senses. As infants begin to construct their mental representation of the environment, they are developing a notion of ontology. The concept of symbols and languages is first learned and understood by infants between the ages of 2 and 7, but they do not have the capacity to abstract. As they reach the age of 6 and 12, they begin understanding mathematical operations, developing logic and reasoning skills. Their capacity for abstraction is developed between the ages of 11 and 15. In addition, in the theory of Piaget, intelligence is considered as the autonomy to adapt to the environment and consists of both *accommodation and assimilation*. Assimilation refers to expanding previously acquired knowledge with new concepts, whereas accommodation refers to the adjustment of old knowledge to accommodate new ones.

From a more general point of view, the notion of cognition is related to actions and interactions (possibly with humans or objects). From a sensorimotor point of view, perception (source 1) and action (source 2) are coupled, as shown in figure 1.2. The motor can guide the sensor, as in the case of young babies, they explore what an object is by taking actions and exploring. Interaction with the object

enriches the notion of ontology by object affordance, that is, the possible actions related to an object: the notions of affordance related to the object, for example, a "cup" is related to "drink", as to a "chair" is related to "sit". This notion of affordance can be integrated into the learned representation.

1.1.2.3 Passive perception to support cognition

In this thesis, we choose to focus on the perception of the autonomous agent and take it as the only input information as a first and necessary step toward a more inclusive interaction. Indeed, even if a more active strategy seems advantageous, this one has to be supported by perception, the subject on which we focus our study. Public datasets are frequently used as inputs by machine learning approaches because they contain perceptual information of images. Meanwhile, in some more complicated artificial intelligence models based on the cognition theory, interactions are often applied to obtain extra feedbacks. For this reason, perception, even if passive, is a central way to acquire information and the first step towards a complete cognition model enabling the agent with all the possible sources of information. For young infants, they are often attracted by visually salient objects, perceive and eventually take actions such as touching to interact with the objects. For a task like object recognition, the agent needs to separate different objects and recognize those that are similar. Through perception, perceptual invariance, such as the invariance in the perception when it is originated from different views for the same object can be built, which is crucial for the separation between different objects. For the same object, although the perception might vary from time to time (for example, due to a change of illumination), there is invariance for the same object. Recognizing different objects is nevertheless a difficult task, requiring the agent to learn perceptual invariance. Furthermore, this variation in the same object has to present a certain temporal consistency [37], as an object will exist for a certain period, without any acute drift into another object. Therefore, the perception is crucial for the agent to learn different objects and the associated invariances.

1.1.3 Continuity hypothesis for discrimination

As previously introduced, the source of information for the agent to construct its representation of the world is based on perception. Thus in this section, we introduce the dedicated criteria for the separation of different classes, which is a purely statistical one in section 1.1.3.1. In addition, another notion, temporal consistency, will be introduced as a part of our hypothesis. Temporal consistency is an important aspect of perception from which humans form object hypotheses and tell if the objects presented to them are the same. With this regard, we will partially integrate the temporal consistency in this work as a first step to guide continual learning.

1.1.3.1 Pure statistics complemented by a temporal continuity

In this thesis, in our proposed approach, we consider that the agent builds representations only from passive perception. While the agent builds its clusters, it relies only on the internal representation learned automatically to separate an object from another when learning is unsupervised. As with ontologies that we have previously introduced in the section 1.1.1.1, the agent will need to either create the notion of a class or recognize an existing one. Therefore, it should be able to distinguish and separate dissimilar classes from those that are similar. But in our case, the separation criteria will be purely based on statistics of the representation it builds. Ideally, it will work well in many cases, but some classes are statistically close and difficult for the agent to separate without any extra sources of information that will come from action or interaction in other more complex models based on the cognition theories. Without action or interaction, in some cases the separation of these classes is hard if it is only based on representation built purely from passive perception since the learning capacity of agents is limited in these cases. Indeed, separation between classes of objects is somehow arbitrary when uncorrelated with the meaning of objects for the activity of the agent. To alleviate this problem, one possible solution is to apply the bootstrapping strategy, which is to use the learned knowledge (for example, the constructed representation) to guide learning in the future. But still, it will depend entirely on the self-constructed internal representation

by the agent. Furthermore, humans are very adept at learning based upon experiences or transferring knowledge, but this is usually not the case for agents that require much training.

As a purely statistical approach may suffer from intrinsic limitations, to alleviate this problem, we propose to make use of the spatio-temporal consistency. It refers to the fact that in a certain space or time, an object will not make a swift shift into a different place (or time) but instead stay visible for a certain time.

We will target the problem of continual learning, and this hypothesis of temporal continuity makes sense in this context, because our "sampling rate" of observations is high enough compared to the physical changes of the observed objects. This is a common assumption for many approaches modelling the dynamics of physical systems, e.g. Markov models, Bayesian filters or more recent methods for (visual) tracking. In addition, this hypothesis is reasonable in particular in the perception of an autonomous agent, which perceives sequences not in a chaotic order but with certain temporal continuity. For example, the agent will first learn the category "cup", followed by other categories, instead of seeing all the images of all the categories at a time. To simplify the conception, in this thesis, we will not take into account the classification of different objects within a category. This relationship is not always clearly defined and may depend on the context. Thus, the notion of object, object category and class are used interchangeably here. The definition of temporal continuity is implemented in the class-by-class sequential order in our scenario, which is refined to a class-incremental one, in order to evaluate and to study thoroughly the implications for our proposed model (Chapter 3). This simplified scenario will further be achieved by integrating the case where the same object can reappear multiple times (Chapter 4).

1.1.3.2 A dedicated testbed

Our objective is to build an autonomous agent that can learn in an unsupervised and continuous way. Presenting instances of objects in a random order to the system in machine learning is common to most machine learning algorithms, but is not realistic for an agent evolving sequentially in its environment. Thus, we suppose that objects will not appear in a completely chaotic order as opposed to i.i.d. and randomly shuffled image sequences. But an autonomous agent will perceive objects with some consistency in space and time. Continuity in perceived objects is consistent with the fact that the position of an object in a space will not change drastically over a short period of time [19], and that their motion, if there exists any, should be continuous and relatively slow. In practice, babies learn from fewer images of cats and dogs than all that are presented in datasets used for machine learning algorithms to distinguish cats and dogs.

Presented to the model are therefore class-by-class image sequences of different categories. We therefore first show the agent images of the same class before moving on to images of the next class based on a public dataset. The identities of these classes are, however, not revealed to the agent and are learned automatically through recognition or detection of a novel class. In figure 1.3, we give a simple illustration of how the testbed is created and deployed in our learning scenarios. We took the example of Fashion-MNIST, a common public dataset that contains images of different categories of clothes. The input images sequence is constructed with temporal continuity by showing the images from class to class. That is, the agent will first perceive images of T-shirts, of trousers, of shoes etc.. Each of the categories be visible for at least a certain duration of time, i.e. a certain number of images. By introducing objects this way, we aim to get close to a realistic video stream. We have to admit that the so defined scenarios are far simpler than taking a video stream as input (this will add difficulties such as noise, background environment, object following etc..), but the problem of unsupervised cognitive development still remains very challenging which motivates us to develop our approach in a simplified and controlled environment, in order to bring the results of the study to more immersive environments as a second step. One has to note that in a more immersive environment, a robot can decide to change the object it observes, which can in some way provide additional information for cognitive development (see next section).

The change in input and the arrival of new categories needs to be detected automatically and on-line if we want a system that continuously learns a representation that is pertinent at each point in time during the training. Our proposed system exploits this information to learn in an autonomous and self-supervised manner using self-determined internal labels. The creation of internal labels depends on how the clusters were created and the task itself (i.e., the order it sees objects), it will indicate what are same

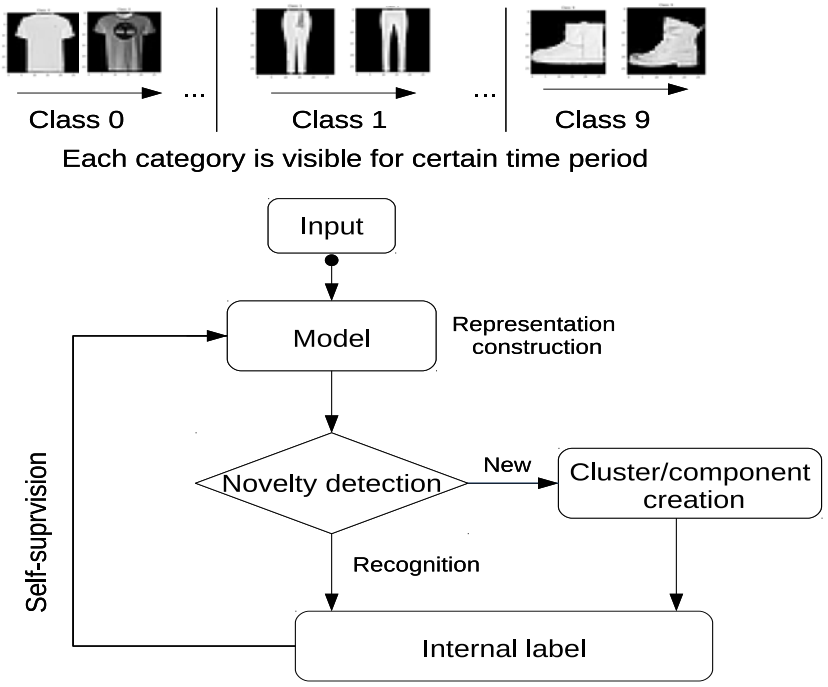


Figure 1.3: Model simulating the temporal continuity of perception: the agent perceives in a class-by-class order instead of completely randomized sequences. This continuity is inspired by the spatial-temporal consistency that at a given point in the space in a very short period of time, there is no shift between objects. This helps the model determine its object hypothesis (that babies can understand starting from a very young age, an object will exist for a certain time), from which the model determines object identities by distinguishing unknown objects from learned ones.

or different entities, but not be unified from a common definition of ontology and classes since ground truth supervision is not used. For this reason, an internal label is a form of "pseudo label". Consequently, it is necessary to assign each cluster created by the agent to a corresponding category via a post-labeling process. In our approach, this is not part of the training process and is only applied during the evaluation of a model's prediction of clusters.

1.1.3.3 The integration of cognitive development

While we have taken direct perception as the only input, other sources can be considered based on a more general view of cognitive development. The action and interaction can be used to generate pseudo-labels that can guide learning. In other words, the agent can utilize object affordance to differentiate between a "cup" and a "light bulb", for example, even though they share some similarities in shape since they may be both cylindrical, the action associated with them is different. Other individuals or agents in the environment may have also constructed pseudo-labels of object identity. As a result, the agent can complete or adjust its ontology assumptions.

A more general example of the integration of action and interaction can be a model that interacts with an object pursuing a goal at the same time (goal-directed) or motivated by curiosity (intrinsic motivation) [56, 125], this allows to further incorporate specific mechanisms such as attention. The agent can therefore actively choose what to learn [144] or what to attend or formulate its own curriculum of learning. In this way, extra feedback is allowed, as well as the integration of action and interaction to form a model that incorporates all three sources of information mentioned earlier.

Previously, we have introduced, from a general point of view, the implications of cognitive theory

and different related targeted properties in different AI approaches. Among all the properties, autonomy is an essential part and is ensured by different sources of information; perception is a key point that is accompanied by action and interaction that allows a full cognitive architecture.

In the next section, we will briefly review some of the approaches and advances in the field of machine learning and artificial intelligence. We give an overview that is centered on briefly introducing the general ideas of these approaches and a particular focus on how they allow the fulfillment of the targeted properties of artificial intelligence.

1.1.4 Machine learning advances: achievements and drawbacks

In this section, we will outline some of the achievements in computer vision and machine learning. Moreover, we will briefly resume the achievements and drawbacks of these approaches regarding the targeted properties in AI.

We have previously reviewed several properties commonly required in artificial intelligence systems in section 1.1.1.3: the autonomy is the capacity for an agent to learn, take actions, and make the decision in an open and unconstrained environment; the capacity to generalize, which is to extrapolate the acquired knowledge onto unseen data; plasticity and stability refer to the capacity to acquire new knowledge and not to lose completely what was acquired in the past.

Computer vision focuses mainly on the perceptual and, notably, visual information as input, from which the model often needs to extract features to solve different tasks. SIFT and SURF are examples of classical feature extraction approaches in computer vision. However, these features remain handcrafted. From a cognitive theory point of view, this feature extraction is similar to humans' abstraction ability. Compared to handcrafted features extracted by classical computer vision approaches, deep learning learns more effective representation by applying deep neural networks. Deep learning has shown state-of-the-art performance but requires a large amount of data for training using deep structures to guarantee the model can generalize well and to avoid over-fitting.

In general, learning can be supervised or unsupervised. As introduced previously, *supervised learning* is to learn with some type of supervision, or ground truth labels to train, thus is of less autonomy in an unknown environment when access to annotated data is limited. *Unsupervised learning* is independent from the ground truth labels. Taking clustering as an example, the underlying criterion for grouping examples is mostly a purely statistical one taking extracted features as input. However, disentangling statistically fine-grained examples without other sources of feedback like actions and interactions may be difficult in these cases. Although there have been some recent progress with deep learning-based approaches, other challenges can come from the difficulty of probability and uncertainty estimation of these models.

In addition, deep learning models often require an i.i.d. dataset with randomly shuffled images. On the contrary, in a dynamic environment, agents receive non-stationary data streams. Thus these models often cannot guarantee the autonomy of an agent in an open and unknown environment. In such a scenario, the dynamics of its perception often engage the agent to self-adapt to changes inside the environment, and offline approaches are little applicable in an online learning scenario. Approaches in the few-shot learning and continual learning domain partially respond to these limitations. In few-shot learning, the models target the capacity to be able to generalize even if only a few data points are accessible. But it is challenging due to the fact that common deep learning approaches usually demand large amount of data to be able to generalise and to avoid over-fitting. In continual learning, the agent continuously learns from a non-stationary data stream and constructs knowledge of new tasks or objects based on the previously learned ones without infinitely memorizing past observations in its memory. In particular, when it comes to learning the best strategies for different tasks, the continual reinforcement learning approaches are often applied in this scenario. Like in common reinforcement learning approaches, agents are required to learn to take actions in a certain state to optimize the expected overall reward they can get overtime, but learning is continuous and based on the knowledge accumulated during previous tasks. The autonomy required in this context is to learn in a continuous manner while adapting to the dynamics of the environment the agent is exposed.

Meanwhile, when the agent needs to learn continuously, it can also suffer from other problems like catastrophic forgetting, related to the stability-plasticity dilemma as introduced in section 1.1.1.3. In

offline deep learning approaches, this problem is not a constraint on its performance. The entire dataset is at the agent's disposal with all the classes or tasks it needs to learn from. However, in continual learning, while learning incrementally new tasks or classes, it will potentially change the learned representation of the past due to the optimization of new tasks or categories. The learned representation can be evolving all the time. Meanwhile, the agent should not forget or destroy the performance of past tasks and categories completely. Although the problem of catastrophic forgetting is not studied explicitly in this thesis, it is still a crucial aspect that impacts the performance of different continual learning approaches. But in the targeted continual learning scenario, the catastrophic forgetting problem needs to be alleviated for the reason that the model can not retain each one of its past observations in the memory, while the data points on new classes or tasks arrive in an online manner. The catastrophic forgetting will principally influence the performance of old tasks while optimizing new ones. In the literature, to alleviate catastrophic forgetting, the model is often paired with a memory system that (selectively) stores some of the real training examples or uses generated examples for the rehearsal of learned categories. While using replay memory for real examples might be harming the capacity of generalization of model [170], for the capacity of storage is limited, but the model relies completely on them to recover the performance on past categories.

But among unsupervised continual learning approaches, the agent will have no access to the ground truth labels. However, as the environment is dynamic, changes can occur during learning while the agent will have to detect them on its own. Therefore, drift detection is important for both the autonomy and the adaptability of the agent to decide if the object is known/unknown when a new category appears. This is also part of the construction of the ontology, as we have previously explained in section 1.1.1.1. It is a challenging problem, as, in some cases, the model can erroneously affect arbitrary high confidence with irrelevant images or an unknown category. And the drift detection needs to be performed online, which implies that the model should estimate statistics in an online manner without having the entire dataset at a time. In this context, the online learning scenario itself can be causing additional problems, such as evolving representation of learned categories while optimizing new tasks or categories. Although usually not applied in machine learning or computer vision approaches, drift detection can come from the feedback of an action that is related, for example, to object affordance. Another possibility is to get additional feedback from the interaction with other agents but these topics go beyond the scope of this thesis.

1.2 Contributions

Previously in this chapter, we have briefly introduced some notions in cognitive theory that are in favor of learning through perception and through other sources involving action and interactions that provide additional feedback to the agent. Several properties are targeted to elaborate such an agent, such as autonomy towards the adaptation in an environment of uncertainty or being able to explore and generalize to unseen but similar data or task structures. From a more general point of view, we have shown how the integration of cognitive theory and models may be a pertinent approach of satisfying these required properties.

We will hereby address the contribution of this thesis with regard to these required properties. We target the context of unsupervised continual learning for object recognition. As we have mentioned, continual learning itself remains a challenging issue. It raises problems that common offline deep learning or machine learning approaches do not need to target due to the non-stationary and non-i.i.d. nature of the input data stream that is contradictory to that of offline learning. In this work, we apply a hypothesis on temporal consistency through the class-by-class order in the objects' presence. Although this class-incremental learning seems to be a simplification of the problem definition, in practice, it is plausible from a cognition point of view regarding how humans can make object hypotheses and how humans distinguish an object from another.

We propose an approach of self-supervision to guide continual representation learning. The core of our contribution lies in the determination of self-supervision through the internal label determined autonomously through novelty detection and recognition, integrated into an existing model in the literature originally targeting unsupervised continual object representation learning. In this regard, our

model is able to detect and adapt to the category changes exposed to it and optimize the model under self-supervision training on both real batches of images corresponding to the current category and the generated ones for generative replay to alleviate catastrophic forgetting.

We have considered two different scenarios. Each is accompanied by a dedicated model. The first is to consider the problem without object review and the second with the review of objects. Although, at first sight, they may seem to differ little, the fact that objects may reappear or not, changes the way how the agent is able to learn without supervision, as we will see later. In both cases, we introduce a self-defined label to guide learning, i.e. the training of our proposed models is self-supervised. To build a proper internal label, the review of objects requires a learned object to be recognized and to distinguish the unknown objects from learned ones, but in the other case (without review), the agent needs only to detect the unknown objects. The model should neither assign too high confidence to an unknown object which leads to false recognition and results in the failure in learning new objects, nor should the agent constantly treat learned objects as new ones. Technically, it may seem a simplified solution for the model to always expand for every object it encounters. But it will tend to overfit and create separate representations for a single object (what we call "over-segmentation") and ultimately fail to generalise over possible appearance variations. Besides, this is actually against what happens in humans' construction of cognition.

In the first scenario, we have proposed a mechanism allowing the detection of unknown objects using the Page-Hinckley test and use novelty detection as a process to create an internal label to guide the model in learning new objects. In this way, the model can automatically detect the number of clusters and discriminate different objects without over-segmentation of clusters. In the second scenario, we have proposed an approach to both detect unknown objects and recognize and distinguish learned ones. It is based on the statistical hypothesis test called Hotelling t-squared test that we have modified and adapted for online operation by estimating first and second-order statistics of learned representation for different categories during learning.

Our model responses to the previously mentioned requirements, typically. For the *autonomy*, learning is self-supervised by the novelty detection in our model and learning remains independent of label information. This also ensures *plasticity* in the way that the model adapts to changes in the distribution of input objects accordingly. Overall, the clustering produced by our model allows to more easily separate the real (ground-truth) classes in the dataset, i.e. corresponds to a better representation of real object categories. The Hotelling t-squared test adapted online allows us to both recognize categories and introduce new ones. This demonstrates the capacity of *generalization* of our model.

1.3 Outline

Here is the outline of different chapters in the manuscript. In Chapter 2, we will first give an insight into the general introduction of different approaches and frameworks in the field of machine learning and deep learning. We will first start by reviewing the most frequently used neural network architectures that will include discriminative models and generative models. Afterwards, we will show some advances in deep learning and the application of these deep neural networks in supervised tasks like classification and unsupervised learning tasks that are tightly related to our context of object recognition. We will expose their advantages and limitations as offline learning approaches and how they were or were not able to respond to the requirements of learning with autonomy in a continual and unsupervised scenario. We will introduce the literature on continual learning, which is contrary to the common offline approaches and corresponds the most to our context of an autonomous agent that continuously learns for object recognition. A special focus will be given to the model of CURL, as it fits most of our objectives of unsupervised continual learning. We will introduce its architecture and how it is able to learn object representations in such a non-stationary context. We will also outline the state of the art in novelty detection that is adapted for offline learning and concept drift detection for data streams and positioning our work for comparison since we consider that without supervision, novelty detection or change detection is an important point to ensure the agents' autonomy in a dynamic environment, it is the key for it to detect changes and first step towards better adaptability.

In Chapter 3, we first place ourselves in a simple scenario in which we suppose a class-by-class

sequential learning order without review of the objects. We will introduce a statistical test, the Page-Hinckley test, with its common use in novelty detection. We will show why the Page-Hinckley test is specifically adapted in our targeted context. Later, we will explain how the Page-Hinckley test was adapted and applied to detect the category change during learning. In our model, we use it as an essential process to create an internal label and to guide learning without using external supervision. Through different experiments, we will be able to compare our method based on the Page-Hinckley test, with other state-of-the-art models, in a simple and harder scenario to examine their robustness facing categories that are statistically close and hard to separate. In the meantime, we will also study the factors that may impact the performance of different models, such as the number of components that helps validate and justify our results.

In Chapter 4, we further suppose a more complicated scenario in which the object can be reviewed. We give a special focus on this scenario since, compared to the first one, it introduces more difficulty and complexity in self-guided learning. Not only does the model need to detect unknown categories, but at the same time, it will need to give a proper estimation of the past categories that eventually allows to recognize learned categories. The model will need to target both the difficulties in the detection of unknown categories and that of recognition while the learned representation of past categories has evolved. Our second proposal is to use the Hotelling t-squared test, for which we will first give a general introduction on how it is applied to common offline scenarios. Afterwards, we will show the difference between an offline and an online scenario, for which we need to adapt the test online. We will introduce the approaches that we have applied for online estimation of the parameters and their integration in the Hotelling t-squared test, to recognize objects and detect new categories in our model. Finally, our model can use this process as a self-supervision that replaces the ground-truth label to guide learning.

Finally, in Chapter 5, we will draw our conclusions and present some perspectives of this work.

Chapter 2

State of the art

Computer vision and deep learning have seen tremendous advances throughout the past decaces. Promising results have been obtained in various tasks like image classification, object detection, image segmentation and 3D reconstruction just to name a few. Classical feature detectors and descriptors like Harris corner [63], Scale-invariant feature transform (SIFT) [104, 105], Speeded Up Robust Features(SURF) [13], were commonly used in different computer vision tasks. The arrival of machine learning technics in the 1990's and 2000's, led to more effectiveness and better performance in visual tasks like classification and regression for many applications. But these approaches based on "hand-crafted" features are outperformed nowadays by deep learning, towards which more and more attention has been drawn.

Deep learning could be seen as one of the most popular recent approaches. In this chapter, we will briefly introduce basic notions in deep learning and deep neural network models concentrating on the computer vision domain. Typically, as mentioned in the previous chapter, in this thesis we focus on the context of continual learning, its definition and its main categories of approaches in the literature. We will also review the state of the art concerning novelty detection, which is important for an agent to automatically detect the arrival of new categories and adapt to changes in the input data distribution.

2.1 Deep learning

The idea of artificial neural networks has its origin from the human brain where billions of biological neurons are connected between each other. As Hebbian theory[69] stated, "cells that fire together wire together". Neurons spark to transfer information. Due to plasticity, human brains learn and acquire new knowledge. Inspired from the biological neurons, the "artificial neuron" was first proposed in [111] by McCulloch and Pitts in 1943. Till today, artificial neurons have still been the most basic unit of some neural networks, yet deep learning has witnessed various neural network structures prosper.

As mentioned in Chapter 1, depending on whether learning involves ground-truth labels, machine learning could be divided into supervised or unsupervised approaches.

Definition 2.1.1 (Supervised learning) *Supervised learning approaches require accessible ground truth label information on training instances. In supervised learning, a model is trained with these annotated examples to make its prediction from the input data to be as close as possible to the ground truth labels.*

Definition 2.1.2 (Unsupervised learning) *Unsupervised learning approaches are independent of ground truth labels, and models are trained on unannotated data.*

Besides, in some cases, only a few annotated data or label information is available over the entire dataset, in this case, we use the term "semi-supervised" ; Another terminology is "weakly supervised" which refers to the case if the algorithm is under weak supervision or uses self-generated pseudo labels to learn.

Definition 2.1.3 (Weakly supervised learning) *Weakly supervised learning refers to learning with weak supervision, which refers to supervision that is either incomplete or noisy (inaccurate).*

Among numerous tasks, classification or regression are the earliest tasks researchers have tried to solve with deep learning.

Definition 2.1.4 (Classification and regression) *In classification, the model needs to predict the class ID of a given instance, i.e. the model prediction is discrete and corresponds to the category identification. Whereas for regression, the target is real-valued and, in most cases, continuous.*

Models for classification or regression can further be divided into discriminative and generative models. In this section, we will briefly review different neural network models, their architectures, and recent applications in deep learning.

2.1.1 Discriminative models

Discriminative models refer to models that explicitly discriminate the input in the target space, predicting their category in a classification task or their target values in regression.

2.1.1.1 Multilayer perceptrons

Structure of multilayer perceptrons

The multilayer perceptrons (MLP) is a discriminative neural network model used for classification or regression tasks[119]. As its name suggests, "multilayer" signifies multiple fully-connected layers that are composed of neurons (perceptrons): the input layer, hidden layers, and the output layer. An illustration of the structure of the MLP can be found in figure 2.1. The input layer refers to the images that the model perceives, and the output layer refers to model output, for example, in classification tasks, the model output is the predicted class. Each hidden layer and the output layer are fully connected to their respective preceding layer, such that the activation of the preceding layer forms the input of the following layer. Neurons have weights W associated to their incoming connections and biases b as parameters of the model. The model parameters need to be learned through an optimization algorithm during training, which will be discussed more in detail later.

More specifically, for an MLP with n layers, let x be the input vector, $h_0 \dots h_{n-1}$ be the activations of the input and hidden layers, and W_i the weight matrix of layer i. Then the output y of the model is obtained by computing the activations of each neuron, layer by layer with the following equations:

$$h_0 = \sigma_1(W_1^T x + b_0) \tag{2.1}$$

$$h_i = \sigma_i(W_i^T h_{i-1} + b_i) \tag{2.2}$$

$$y = \sigma_n(W_n^T h_{n-1} + b_n) \tag{2.3}$$

In MLP for each layer, there is also an activation function σ that may add non-linearity to the model. Common choices for activation functions are, *tanh*, *sigmoid* or *ReLu*. with *tanh* as

$$\sigma_n = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.4}$$

with *sigmoid* as

$$\sigma_n = \frac{1}{1 + e^{-x}} \tag{2.5}$$

and *ReLu* as

$$\sigma_n = max(0, x) \tag{2.6}$$

When the MLP network is used as a classifier, the softmax function is often used after the output layer of the MLP, with the softmax function defined a:

$$softmax = \frac{e^{y_i}}{\sum_1^K e^{y_i}} \tag{2.7}$$

Training of multilayer perceptrons

Training of an MLP is composed of two phases that are alternated and repeated several times for each example: the first one is the feedforward phase, during which the input is fed to the MLP to compute the model output. This output is then compared to a target value (in supervised learning), and an objective function (or loss) measures the error. Depending on the task, the objective function could be for example the mean squared error (MSE) between the model output prediction and a target value, which can be used for regression. Assuming that y as target, and $\hat{y}$ as model prediction, the MSE loss is defined as

$$L(y, \hat{y}) = \frac{1}{N}\sum_i^N (y - \hat{y_i})^2 \ . \tag{2.8}$$

During the second phase, neural network parameters are then updated through backpropagation. The loss function is computed considering the difference between model output and the target, and the model iteratively optimizes this function, e.g. by using gradient descent In the literature, there are many optimization algorithms, for example Stochastic Gradient Descent (SGD), Adam, Conjugate Gradient (CG), BFGS. Formally, the equation of gradient descent could be written as:

$$W = W - \alpha * \frac{\partial}{\partial W} L(W, b) \tag{2.9}$$

$$b = b - \alpha * \frac{\partial}{\partial b} L(W, b) \tag{2.10}$$

with α being the learning rate that determines the amount to update the parameters at each iteration. The learning rate is one of the hyperparameters of the neural network model, that, unlike weight W and bias b, is not learned by the model but instead tuned manually, and it is crucial to choose an appropriate value. A too large value will induce more changes in the network parameters, which might cause learning to diverge. Another important hyperparameter is the choice of network structure, especially the number of hidden layers and the number of neurons present in each layer, which is a factor that impacts the learning performance.

During training one needs to be cautious to avoid overfitting– the neural network model becomes an expert on training data and generalizes poorly to others, test data for example. To prevent the effect of overfitting, the validation set is used for the application of strategies like early stopping– once the errors on the validation start to increase, training would be stopped.

2.1.1.2 Convolutional Neural Networks

Convolutional Neural Networks (CNN) [92] are another common type of discriminative models. They are known for their capacity to cope with structured data with a certain 2D topology and locally correlated information such as images in computer vision tasks as mentioned in [94], and they also automatically extract effective features, as is reviewed in [93].

A definition of common CNNs can be found in [184]: suppose that the input 2D images are noted as x, through CNN they are mapped into y, the output layer predicting to which class x belongs to in the classification scenario. CNNs are composed of stacks of layers: alternating convolutional layers of kernels (filters) and pooling layers followed by one or several fully connected layers that map feature maps into an output vector or matrix with the desired dimensions. For classification, the output dimension would be the number of classes with a softmax function as activation function) as shown in figure 2.2.

We will give a more detailed explanation for different layers: in CNN, "convolutional" takes its name from convolutional layers, composed of convolutional filters. Compared to MLPs, in CNN, neurons are

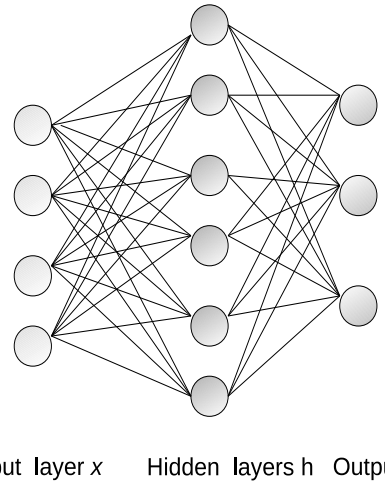


Figure 2.1: an illustration of MLP architecture: the input layer x, the hidden layer h and output layer y, and all the layers are composed of neurons that are fully connected between each other.

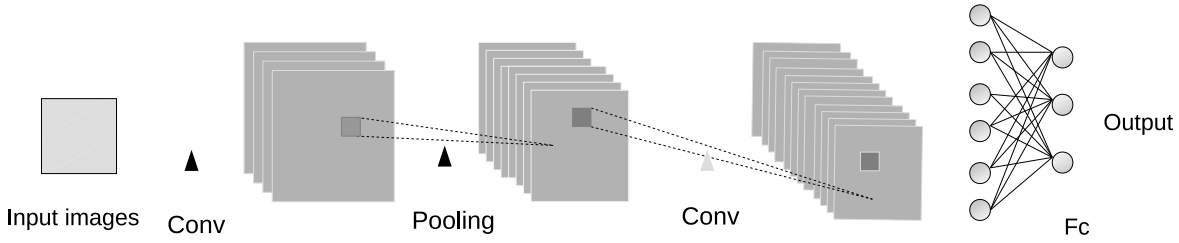


Figure 2.2: an illustration of CNN architecture: CNN are composed of convolutional layers, pooling layers and fully connected layers.

with one more dimension, the depth, and neurons are of 3D. For a convolution layer, its depth is controlled by the channel number, referring to the number of filters implied. The use of multiple kernels enables the model to extract enriched information. Each convolutional layer will take the feature maps resulting from the previous layer as input, and perform dot product through each kernel, resulting in a stack of feature maps with the number of feature maps equal to the channel size. To note that the convolution operation remains 2D locally within the width and height dimension, but fully throughout the depth dimension. For example, for a given convolutional layer of channel size N, if there are 3 channels in the previous layer as input, they will be convolved with N filters of kxkx3 resulting in N feature maps as output. Convolutional filters could be with different sizes in different convolution layers, and the kernel size k is among the hyperparameters while defining the architecture of CNN. For convolutional layers, another hyperparameter is the stride, referring to the size skipped while sweeping kernels over the input. Like in the MLP neural networks, activation functions, like ReLU or sigmoid, are also used in CNNs.

Pooling layers are also used in CNNs, to sample for the maximum value (max pooling) or the average value (average pooling) inside a certain local region (inside a kernel). For an illustration of how the pooling layer works, see figure 2.3. Through the pooling layer, the dimension of the input is reduced by retaining only the maximum or average value inside a kernel. The use of the pooling layer in the CNN structure helps strengthen the translation invariance–in case there are locally some inaccurate parts, the pooling layer will only take into account the largest or the average value inside a window.

In a CNN, more generic features of images could be obtained from the first/second layer of the CNN,

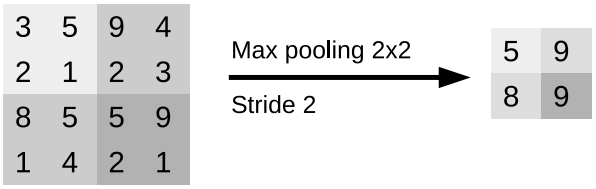


Figure 2.3: An illustration of the max pooling layer of kernel size 2X2 and stride 2. In each sliding window, to compute the output of the pooling layer, only the maximum value is preserved. Taking an example of the first sliding window, among "3,5,2,1" the maximum value is 5.

and as the structure goes deeper, the features become more and more specific[150, 181]. As mentioned in [181], the features obtained by the first layers are general, and could be applied in other tasks, thus giving the possibility on the widely applied fine-tuning [53] technique in which the model is first pre-trained on a large dataset and then with a small learning rate fine-tuned on another dataset. It is beneficial for certain cases where the amount of data are limited, and the model is more robust compared to training from scratch.

In recent advances, CNNs have an increasing number of layers, for example, VGG[152] proposed by Karen Simonyan and Andrew Zisserman. In the VGG model targeting classification on the ILSVRC 2014 dataset, a deep CNN is constructed by stacking convolutional and pooling layers. VGG takes as input images of size $224 * 224$. The receptive field size for convolutional layers are small ($3 * 3$), and max pooling is of size with $2 * 2$ with stride 2. In GoogLenet [161], there are 22 layers applying "inception" modules that are sparse blocks composed of filters (in parallel) of different sizes (1x1,3x3,5x5), and are stacked together.

Though better performance is usually obtained with a deeper network structure, adding layers might sometimes lead to the vanishing gradient problem, namely the problem that the gradient arriving through backpropagation at the first layers becomes very small. To target these problems in [68], ResNet is proposed which introduces the concept of shortcuts or skip connections.

2.1.2 Generative models

Generative models learn the statistical distribution of input data, from which the model is capable of sampling and generating new examples. Models like VAE [78] allow learning with variational inference with deep neural networks and extracting higher-level information, called embedding. This notion of learning semantic and meaningful representations of objects is also referred to as *representation learning* or *manifold learning*.

Definition 2.1.5 (representation learning and manifold learning) *In representation learning, the model extracts semantic features by mapping images into an embedding space of lower dimension yet preserving information related to their similarities.*

Definition 2.1.6 (disentangled representation learning) *the terminology "disentangled representation learning" is also mentioned, referring to the capacity of breaking the representation(disentangle) into disjoint parts, for which a formal definition could be found in [70].*

By extracting deep features that encourage better disentanglement of different objects, the model could construct a more robust estimation of object distribution.

2.1.2.1 Autoencoders

Autoencoders are neural networks that can learn a compact representation of input data by an encoding and decoding process that reconstructs the input (see [165] for a review). The learned representation is often of lower dimension than the input data. An autoencoder minimizes the reconstruction loss between the output and the input and uses backpropagation to update the model parameters. Formally,

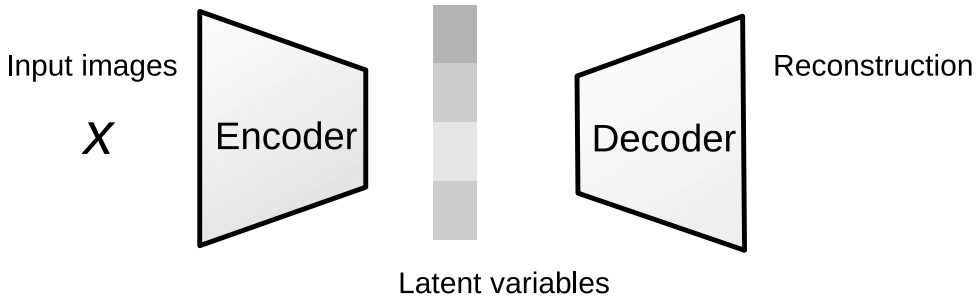


Figure 2.4: An illustration of VAE architecture: the encoder maps the input into a latent space, and the decoder reconstructs the input. Latent variables are supposed to follow a standard normal distribution.

suppose that the input images are denoted as x, the encoder E and the decoder D and the embedding vector z. Then the reconstruction of the decoder $\hat{x}$ can be obtained as:

$$z = E(x) \tag{2.11}$$

$$\hat{x} = D(z) \tag{2.12}$$

During training, the autoencoder optimizes the reconstruction, i.e. the l2 norm between the output and the input images.

$$arg\min_{E,D} = ||x - \hat{x}||_2 \tag{2.13}$$

In the literature, there are also several variants of autoencoders – for example, denoising autoencoders [173] (DAE). A DAE takes as input corrupted data and is trained to predict the original (uncorrupted) image in a way that removes the noise, also referred to as "denoising".

2.1.2.2 Variational Autoencoders

VAE [78] is a generative model. Concerning its structure, similar to autoencoders, variational autoencoders are composed of an encoder and a decoder. In practice, for encoder and decoder, deep neural network structures are used for better performances. The encoder maps the input images into a latent space and the decoder reconstructs the input data from a representation in the latent space, i.e. the output of the encoder, as demonstrated in figure 2.4. Yet its difference from common autoencoders is that the latent variables are supposed to follow a standard normal distribution $N(0, I)$. In this way, the VAE learns the statistical latent distribution of input data by estimating the mean and the covariance of in the latent space which allows the easy generation of (realistic) new images, a specificity that distinguishes generative models from discriminative models.

Mathematically, as described in [77], let x be the input image, z the latent variable and θ the decoder parameters, the distribution of the marginal likelihood p_θ can be written as follows:

$$p_\theta(x) = \int p(x|z)p(z)dz \tag{2.14}$$

The true posterior $p_\theta(z|x)$ is considered intractable. Thus, it is approximated by $q_\phi(z|x)$ the distribution of the latent variable z learned by the encoder (with parameters ϕ), and by the distribution learned by the decoder denoted $p_\theta(x|z)$.

A VAE is trained to optimize the variational lower bound of the marginal likelihood or evidence lower bound (ELBO). That is, VAE models optimize the reconstruction loss and a regularisation term based on the Kullback-Leibler divergence D_{KL} between the variational approximation inferred by the model $q(z|x)$ and the prior distribution, the standard normal distribution $p(z) \sim \mathcal{N}(0, 1)$.

$$E(x) = \mathbb{E}_{q_\phi(z|x)}[p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z)) \tag{2.15}$$

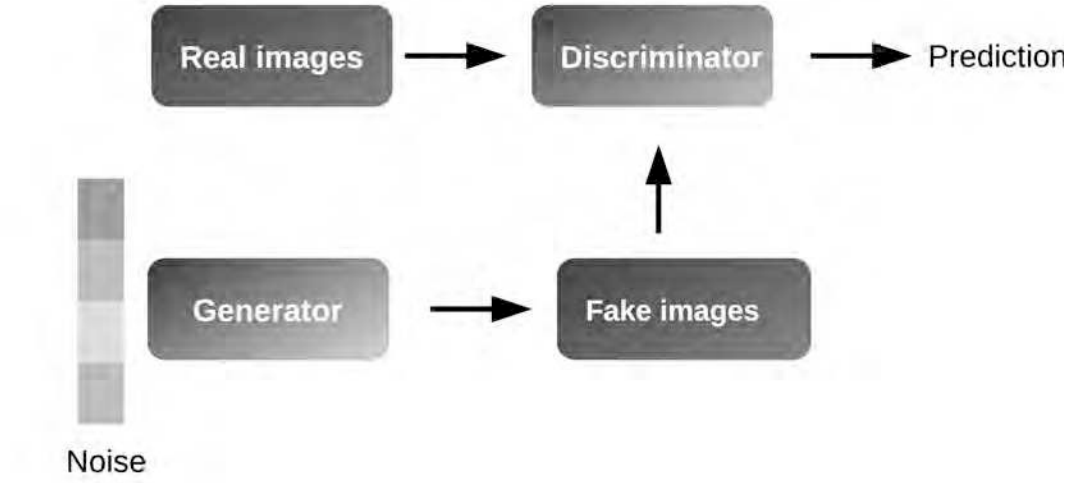


Figure 2.5: An illustration of the classical GAN architecture: the generator takes a noise vector as input and is trained to generate an image as realistic as possible, while the discriminator is trained to discriminate these fakes images from real images. In that way, the generator and the discriminator compete against each other.

The encoder $q_\phi(z|x)$ and decoder $p_\theta(x|z)$ are implemented as two neural networks similar to a standard auto-encoder (cf. Fig. 2.4). By regularising the latent space to be close to a standard normal distribution, the VAE can generate images that have never been seen by the model, In practice, for a given image x, z is sampled from $q_{phi}(z|x)$ and the so-called re-parametrization trick allows to optimize the loss function and backpropagate the gradient to update model parameters.

In the literature, several variants of VAEs have been proposed. For instance, the β-VAE [71] is a common approach in representation learning, that uses an additional regularization factor β in the original objective of VAE.

$$E(x) = \mathbb{E}_{q_\phi(z|x)}[p_\theta(x|z)] - \beta D_{KL}(q_\phi(z|x)||p(z)) \tag{2.16}$$

This factor allows weighting the KL divergence regularization term such that by increasing β the resulting embedding is more forced to look like a standard normal distribution with diagonal covariance matrix, and thus the dimensions of the latent space are more independant When $\beta = 1$, one can find the formulation of a common VAE. As argued in [71], the use of factor β could improve the "disentanglement" between representation, which refers to a better separation in representation compared to a common VAE.

2.1.2.3 Generative Adversarial Networks

Compared to VAE, another model, *Generative Adversarial Networks (GAN)* [55], presents stronger image generation capacity and can generate images of better quality. For the structure of *GAN*, see figure 2.5. There is a generator and a discriminator "competing" against each other; the objective of the generator is to generate images to be as realistic as possible. The objective of the discriminator, on the other hand, is to distinguish the generated images from real ones. The generators take as input a noise vectors from a certain distribution and transforms them into new images, and the discriminator will estimate the probability of data being generated rather than being real data. Let x be a (real) input image, and z some noise vector, D the discriminator G the generator. A GAN tries to maximize $D(x)$ (to obtain a value close to 1) and minimize $D(G(z))$ (to be close to 0), resulting in a min-max game as stated in [55]:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1 - D(G(z)))] \tag{2.17}$$

Apart from the traditional GAN, in recent advances in state of the art, several variants of GAN have been proposed. For example, Conditional Generative Adversarial Networks (CGAN) [114] conditions the

training of the discriminator and the generator on auxiliary information y, guiding data generation with respect to the introduced information. In CGAN, assuming some extra information as y to be conditioned on, the loss function has become:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x|y)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z|y)))] \tag{2.18}$$

Deep Convolutional Generative Adversarial Networks (DCGAN) [132] are another extension of GANs. Compared to the original GAN, several improvements in the deep neural network structure have been made, notably in the use of convolutional layers to replace pooling layers. Batch Normalization is applied in both the generator and the discriminator. In that way, DCGAN improved the stability of common GAN structures and can learn robust representations in different tasks. Another variant is the CycleGAN [190] that targets learning for image-to-image translation between two domains. In CycleGAN the input is mapped from source to target domain, and from the target into source domain (forming a cycle), and during this process, CycleGan would try to generate images as real as possible with respect to the discriminator in both the source and the target domain.

In InfoGan [30], compared to the conventional VAE, apart from the "incompressible noise" noted as z, InfoGAN has an additional input corresponding to the "latent code" denoted as c. The author proposed a modification that concerns an information theory-related objective to be optimized. Concretely, the model learns representations by constructing latent variables that maximize the mutual information between the latent c and the observations. With $G(z,c)$ being the output of the generator, $I(c;G(z,c))$ the mutual information between the generator output and the latent code c, the loss to be optimized is:

$$\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c;G(z,c)) \tag{2.19}$$

with λ being a regularization factor controlling the amount of mutual information. Again, similar to β-VAE, the proposal of InfoGan aims to target better disentanglement of learned representations.

In this section, we have briefly seen different deep neural network structures in deep learning, including discriminative models like MLP, CNN and generative models like VAE and GAN. In the following section, we will focus on applications of these models in the domain of object recognition.

2.2 Object recognition

As mentioned previously in section 2.1.1, in object recognition, the model needs to classify a given (image) input, i.e. determine its category. In the next few sections, we will briefly review different techniques that are used for supervised learning and unsupervised or weakly supervised learning for object recognition.

2.2.1 Supervised approaches

Classical object recognition approaches are based on specific visual feature extractors, e.g. local histograms of colour, shape or texture, or dictionaries of local patches like in Bag-of-Words representations, followed by a classification method. In the simplest case, this is a k-Nearest Neighbor (kNN) [44] or a naive Bayes classifier. But these techniques are limited by the expressiveness of the hand-crafted features and their low dimensionality. Therefore, more advanced machine learning techniques based on Support Vector Machines (SVM) [34] have become very popular and led to state-of-the-art results in the 2000's. SVM-based methods learn a hyperplane (possibly in a non-linear projection space) that corresponds to the optimum separation of different classes and maximizes the margin. Although these methods allow the learning of effective models for classification and regression with larger amounts of training data, there are still mostly based on hand-crafted features. Thus, with the growth of the complexity of tasks, the amount of available training data especially image data and computational power, deep learning techniques based on CNN have superseded these methods and led to state-of-the-art performance in object recognition.

Deep learning techniques have first demonstrated their robustness in offline supervised learning problems by using deeper neural network structures trained on a large amount of data. Starting with AlexNet

(ILSVRC 2012) [87], deeper network structures of CNN could be found, followed by VGG [152] and GoogleLeNet [161] with some later extensions and improvements, Inception-v3 [162] and Inception-v4 [160], and then ResNet (ILSVRC 2015) [68]. For a more detailed description on the architecture of these models, please refer to 2.1.1.2.

Most of these approaches rely on the assumption that the data are independently and identically distributed, and need to be trained on large annotated datasets, and the more there are annotated data, the better performance can usually be expected provided that the model has enough parameters.

2.2.1.1 Detection and segmentation

Classically, object detection approaches detect and localise objects of a given category (e.g. faces) in an image. With the advent of deep learning, object detection [192] could be implicitly related to object classification and object recognition, since, in the multi-object cases, the model indicates to which object category it belongs to. Therefore, we would also like to introduce several object detection models and give a brief review on the use of deep neural network structures in these models.

Definition 2.2.1 (object detection) *In object detection, the model needs to detect object candidates and corresponding regions. There are mainly two groups of object detection approaches: "two-stage" approaches treat this issue in a "coarse-to-fine" manner starting with region proposals in form of bounding boxes, and use them for training, which implies they only use part of the input images. On the other hand, "one-stage" approaches use the full image to train the model end-to-end.*

Two-stage "coarse-to-fine" approaches demonstrate higher accuracy in general, but one-stage approaches often provide faster solutions for object detection. An example of a *two-stage "coarse-to-fine"* model is the method based on "Regions with Convolutional Neural Network" (R-CNN) [53]. In R-CNN, the authors first use selective search to generate region proposals, that are regions containing possible candidates of objects, and then train a CNN from region proposals for the feature extraction of fixed length, and finally fit (category-specific) SVM models for the classification for each category. The use of a CNN in R-CNN allows to extract features from the region proposals, but the selective search in R-CNN is relatively time-consuming, which is one of its limitations. Compared to the two-stage R-CNN [53] approach, the Fast R-CNN[52] is of higher speed. The entire images are mapped through convolutional layers. Then for regions of interest (object proposals), the resulting feature map from the convolutional layer is passed through a region of interest (RoI) pooling layer, which is a pooling layer that maps regions of interest on feature maps through max pooling. This allows the extraction of features of fixed size regardless of the RoI size. Finally, the Fast R-CNN uses fully-connected layers for softmax probability estimation and the regression of bounding box coordinators to refine the initial region proposals.

The related approach Faster R-CNN [52] can be seen as a one-stage approach, as it uses an integrated region proposal network for region proposals. Other recent *one-stage* object detection advances include "You only look once" (YOLO) [135]. estimating the probability of an object appearing in an image region. YOLO builds a regression model by taking the entire image as input and dividing it into a grid, and using a CNN learning generalizable features for both the prediction of bounding boxes and class probabilities. Yet as mentioned in [135], the limitation of YOLO lies in the detection of groups of small objects. Later other approaches that are related to YOLO have been proposed, for example, YOLO9000 [136] and YOLOv3 [137]. SSD[101] is another one-stage object detection approach. SSD uses a CNN to map the input images into feature maps of different scales and predict the bounding box and confidence for different object classes with convolutional filters. DEtection TRansformer(DETR)[23] adopts transformer-based encoder-decoder architecture for object detection, combined with a bipartite matching loss for prediction. Like other transformer models, the use of transformer based architectures has allowed a self-attention mechanism to aggregate information from a sequence, could help in tasks like removing duplicate predictions.

Extensions of existing object detection approaches could be found in other similar tasks, for example, object segmentation. Here, apart from detecting the object detection, the model also needs to segment the instances. Mask-R-CNN [67], is an extension of Faster R-CNN [52], that for each region of interest (RoI) additionally predicts a binary mask. The Mask-R-CNN also allows pixel-to-pixel alignment of features

with RoIAlign, a pooling layer improving RoI pooling layer which was originally applied in Fast R-CNN as a layer for feature map extraction. By avoiding harsh quantization applied by RoI pooling, RoIAlign improves the alignment of features and avoids misalignment that could be caused by the quantization of RoI boundaries.

Object detection addresses the problem of detecting and localizing objects within images, feature extraction is mainly performed with different CNN-based network structures. Two-stage object detection approaches are generally of higher accuracy, while one-stage object detection approaches are often faster. Most of the approaches that we have reviewed until now are supervised. However, this supervision is costly, and is not given in our studied setting. On the contrary, unsupervised object recognition together with representation learning approaches may allow to extract features that are more relevant to an unknown context and thus provide more autonomy in cases with unannotated data. Therefore, in the following section, we will describe the main unsupervised and weakly supervised approaches in the literature.

2.2.2 Unsupervised and weakly supervised approaches

Unsupervised and weakly supervised approaches are defined as approaches independent or, respectively, little dependent on data annotation, a detailed definition can be found in section 2.1. This signifies that learning is more autonomous, referring to the fact that little or no external information, such as class labels, are given or when the annotation is noisy. From this point of view, unsupervised or weakly supervised approaches are appliable in more general use cases since they have more flexibility and adaptability even in unknown environments. In the following sections, we will first take an insight into semantic SLAM, an approach that is often used in robotics, and we will show the use of object detection and recognition in semantic SLAM and their role in semantic information extraction. We will then outline some unsupervised and weakly supervised approaches using machine learning and deep learning. We also consider that for unsupervised object recognition, their performance is related to the model's capacity in effective representation learning; the relationship between object recognition and representation learning will be discussed more in detail later.

2.2.2.1 Semantic SLAM

Definition 2.2.2 (Simultaneous Localization and Mapping) *Simultaneous Localization and Mapping (SLAM) [39] solves a mapping of the environment and simultaneous localization of a robot or different objects on the map based on its observations.*

SLAM is widely used in the robotic field for finding a semantic mapping for its environment and to find its location inside the environment, its application includes the navigation and path planning for robots [120]. In SLAM, as [39] mentioned, probabilistic filtering algorithms are applied by traditional SLAM approaches, such as the Extended Kalman Filter (EFK) in EKF-SLAM, a common filtering approach based on Kalman Filter and estimation theory that is usually used to estimate states in a dynamic system. The extended Kalman filter in SLAM is often applied as an estimator for localization on the map [39]. A concept related to SLAM, is *semantic* SLAM, which uses high-level semantic features to attribute different categories of room or object identities to information perceived from sensors, which works even without supplementary kinetic or angle position information which are required in traditional SLAM. A review of this field can be found in [62].

Semantic information extraction is also called anchoring in the literature [62]. Some existing methods use traditional computer vision approaches with feature extraction based on SIFT [105] and SURF [13] descriptors, for example. In [27], a SIFT detector and descriptor are used for object detection. Concretely, SIFT [104, 105] extracts scale-invariant features, it first detects the location of keypoints by finding minima or maxima of the difference of Gaussian that approximates Laplacian of Gaussian (LoG), and uses orientation histogram for gradients in the neighborhood of detected keypoint to compute descriptors. Different from SIFT, SURF [13] detectors using determinant of Hessian matrix for the detection of keypoints, box filters are used in SURF approximating LoG. In SURF descriptors, the sum of Haar wavelet responses is used to extract descriptors around the keypoint. The use of SURF in semantic

SLAM can be found in [41]. An object detection module was integrated for semantic segmentation and object recognition, which allow the model to perform a semantic mapping of the environment. RGBD (red, green, blue, depth) images are perceived from the sensor (a Microsoft Kinect camera) and converted to a 3D point cloud. The authors first remove floor-related information, and segment objects by creating object clusters from 3D point clouds regarding the color information. For object recognition, 3 types of features are combined: local color histograms, SURF, and 3D surflets, with a bag of visual words approach, to compute histograms for compares the similarity with all learned objects in each feature space. Then, the authors train a feedforward neural network with the similarity score for an overall similarity regression. SIFT and SURF extract useful information from the input images that are more effective and robust than raw pixels, however, they remain *handcrafted*. Other examples of the use of feature extraction in semantic SLAM in the literature can be found in [24, 25]. Here, instance recognition and scene segmentation were considered for semantic information extraction, in the context of a robot that learns different objects to construct its representation of the environment. The authors consider instance recognition and scene segmentation problems by using color or depth information from an RGBD camera in the context of mobile robots. For scene segmentation, the approach relies only on depth information, and creates clusters of objects with regard to a distance metric.For object recognition, similar to [41], the authors use 3 features of SIFT, color histograms (RGB) and point feature histograms (distances between randomly selected points), to train a 3-layer feed-forward network that calculates a confidence metric of similarity to training instances.

Recent advances in object detection and segmentation using deep neural networks have led to new semantic SLAM approaches with increased robustness. Using effective object detection models allows to detect "object" candidates. Thus, compared to other feature detection and extraction approaches that detect "generic" keypoints and require noise-sensitive matching, object detection gives more object-level information that can be further refined, for example using a Mask-R-CNN [67], an object segmentation model as explained in section 2.2.1.1. Its application can be found in [141], where the author proposes MaskFusion that tracks multiple objects, building 3D models for each object and background consisting of surfels. For multiple objects, to associate each object with its model in the correspondence, the algorithm applies Mask-RCNN for semantic object segmentation in combination with geometric segmentation. In this way, MaskFusion enables real-time object-level RGBD-SLAM. [156] apply AlexNet to extract features for place recognition. In [155] a CNN is used as a classifier and the model further uses a Bayesian filter to take into account the temporal consistency, for the semantic mapping of a robot.

The use of object detection and object recognition in semantic SLAM allows the model to incorporate high-level semantic features. However, most approaches are not completely online or in real-time– they need to first extract features on the entire database for post-processing. In the state of the art, other approaches often use machine learning or deep learning for object recognition.

2.2.2.2 Approaches using machine learning and deep learning

Deep learning has demonstrated its capacity to extract semantic representations. As mentioned in [30], unsupervised learning is an ill-posed problem, for this reason, the capacity of the model to learn effective representations is crucial for an unknown task since the "disentanglement" of the representation allows "explicit separation of different attributes". Although "representation learning", as defined in section 2.1.2, is not necessarily "designed" for object recognition, we argue that a good representation learning approach implicitly leads to better performance in object recognition. As remarked in [30], generative models are among the solutions to learn such effective representation by creating or generating synthetic data, a review can be found in [131]. In the next sections, we will outline several state-of-the-art models of unsupervised representation learning with generative and weakly supervised neural network approaches.

Unsupervised generative approaches

Generative models like VAE are often used for representation learning, and are able to learn semantic higher-level representations by means of a multi-layer encoder-decoder structure with a "bottleneck" in the middle. In section 2.1.2.2, we described the architecture of Variational Autoencoder and several of its variants. These models are capable of learning the distribution of objects of the training set and to generate new examples. Here we would like to focus on the application of VAE in representation learning.

In the literature, while using VAE to learn representations, researchers often combined common VAE with other statistical approaches or statistical-related loss functions to improve the disentanglement of learned representation. For example, in [57], for representation learning, a VAE is combined with a stochastic process – the Chinese Restaurant Process (nCRP) to learn a hierarchical representation model. Total Correlation VAE [48] aims to learn "disentangled and interpretable representations" by applying an information-theoretic representation learning approach. The proposal of [48] optimizes an objective based on the Total Correlation [177] (a variant of the multivariate mutual information), as an alternative to the traditional VAE ELBO objective function (refer to section 2.1.2.2) encouraging disentangled representations.

Apart from its application in representation learning, [169] use a VAE to generate synthetic exemplars during learning for object recognition, and synthetic exemplars motivate a feedback mechanism through self-supervised learning using network prediction to further improve the learning of the model.

Though often used for representation learning, compared to other generative models such as GAN, the capacity of generating synthetic data of VAE is limited. In addition, during training of the variational autoencoder, it often needs to explore the trade-off between reconstruction accuracy of input data and the disentanglement of learned representation. Apart from unsupervised approaches that are independent of ground-truth labels, other approaches use weak supervision. In the next section, we are going to introduce some of these weakly supervised approaches.

Weakly supervised approaches

In weakly supervised approaches, the models need to learn under weak supervision –supervision is incomplete and could be noisy, but compared to unsupervised approaches that use unannotated data, the weakly supervised approaches are often with some supplementary information in supervision. In the literature, the Siamese Neural Network [22] is one of the solutions in weakly supervised learning, because the ground-truth labels are not directly exploited in the training, but instead *similarities* from pairs of input examples are learnt by indirectly using the labels. Concretely for its structure, siamese neural networks project the input data into a lower-dimensional output space (embedding). They are composed of two input and two branches of neural networks that share network parameters between them, during training the model needs to learn from positive and negative pairs and is optimized with respect to the similarities of their embeddings, i.e. attracting those that are similar and pushing far away from each other those that are dissimilar, a principle that has been effectivly adopted later on by contrastive learning.

An example could be found in [83], where a siamese convolutional neural network is trained on image pairs to learn their similarities of handwritten digits and letters from datatsets like omniglot. Another example of visual representation learning of objects is [112], where a convolutional Siamese Neural Network has been used for unsupervised representation learning in videos for an autonomous agent. To this end, the proposed approach detects proto-object proposals based on object saliency, and form similar and dissimilar pairs to train the Siamese neural networks. The Siamese neural network models similarities between objects, via its optimization – similar objects become closer, and dissimilar objects become far away from each other. A recent example of using the siamese neural network for representation learning is SimSiam [31], but compared to the conventional training of the siamese network, SimSiam uses no negative pairs. For a pair of inputs, their method applies a siamese neural network with a shared-weight encoder, followed by an MLP projection layer. Then, a symmetrized loss is optimized based on the negative cosine similarity between two model branch outputs.

2.3 Continual and incremental learning

As mentioned previously, in common offline deep learning approaches, during training the model sees the entire dataset with numerous examples, and each example is seen multiple times. In scenarios where the entire dataset is not available all the time during learning, common offline approaches are not appropriate anymore and not functioning well.

In that case, *"continual learning"*, sometimes also referred to as "lifelong learning" or "online learning" in the literature, is a more suitable approach, since it studies the case of learning with non-stationary distributions and sequentially arriving data, i.e. the model does not have the entire dataset at hand.

There are also incremental learning approaches, and in the literature, these different terms do not always have the same interpretation and are sometimes confused. Therefore, we will here give our definitions of them which will be used in this thesis.

Definition 2.3.1 (Continual learning) *Continual learning is defined as learning and adapting a model such that training examples are presented by following a certain continuity in the data, e.g. different learning tasks following one after the other. Previous examples cannot be presented again or only to a limited extent, and the input data stream can be non-stationary.*

Definition 2.3.2 (Incremental learning) *In incremental learning, data samples are presented in a sequential order to the model but are usually i.i.d., and the number of classes may be known in advance. The learning algorithm needs to continuously update the model without forgetting the knowledge acquired in the past.*

And another notion related to continual learning is *few-shot learning*. In both learning scenarios, the model should not retain training examples infinitely.

Definition 2.3.3 (Few shot learning) *Few-shot learning refers to learning in the case where there are only a few examples in training classes, which implies that the model needs to generalize on a new data distribution using a very small data sample.*

In this section, we would like to focus on continual and incremented learning approaches, where the model needs to be built continuously or incrementally. Starting with few-shot learning, a first step approaching continual learning, we will show different few-shot learning models, followed by use cases and approaches in continual and incremental learning.

2.3.1 Non-neural network incremental and continual learning approaches

Previously, we have seen some continual learning approaches using neural networks. In state of the art, some approaches use other technics such as reinforcement learning and incremental learning. In the following section, we will present reinforcement learning and its use in continual learning, followed by other incremental approaches targeting incremental learning.

2.3.1.1 Incremental approaches

In the following sections, we will focus on the last scenario mentioned in section 2.3, the incremental learning scenario in general. Facing the limitations of offline learning approaches based on incremental learning [103] partially improves the application in the case where learning needs to be dynamic and adapted to the arrival of new data. Refer to section 2.3 for a definition of incremental learning. Their application could be found in many fields. One common application is the online object tracking for which the model needs to classify online the object and separate it from the background, please refer to [1] for a review.

Here we present several approaches based on incremental learning of several traditional machine learning techniques. We will show different incremental or online learning approaches and their limitation. Incremental SVM[159] extends common SVM approaches as incremental by learning incrementally new data and recursively constructing previous solutions for previous observations. The algorithm dynamically updates support vectors for training. *LWPR* [172, 171] incrementally learn a nonlinear regression model as a sum of the projection of local linear univariate regression models in several directions, and the model uses Gaussian kernels [81] for the region of validity for each local linear model (receptive field). But like many other approaches in this domain, LWPR remains a regression method and provides no explicit feature extraction.

Apart from incremental SVM and regression, other incremental learning approaches use clustering. Clustering refers to the creation of a data partition that separates different categories and regroups the same categories.Classical clustering approaches[180] include: K-means[130], DBSCAN [151]. Similar to clustering, *Self-organizing maps* (SOM)[84, 153] allow learning the topology of the data structure based on their similarity. SOM is trained using a competitive learning rule that creates a map on which grid of

neighborhood nodes are similar and close to each other. From this point of view, it is different from other deep learning approaches, which optimize a cost function to minimize the model prediction or estimation errors. Input data are mapped into nodes that are associated with the weights vector. During training, the model selects the most representative node with its weight vector for input and adjusts similar nodes (usually its neighborhood) in correspondence[9]. However, there are several limitations of traditional clustering approaches. Clustering remains limited in terms of feature extraction. In general, no explicit feature extraction is included in clustering models, so it might be necessary to train a supplementary network or use pre-trained models to extract features. In addition, as pointed out in [180], classical clustering approaches usually face several limitations. They suffer from problems of scalability and often lack explicit feature extractions. They are constrained with large-scale or high-dimensional data and sensitive to the increase in scale, and some clustering approaches might classify outliers within a cluster of known classes. But to some extent, they are less sensitive to noise present in the data.

Facing the limitations of traditional clustering and SOM, which tend to target static datasets, advances in incremental clustering allow creating subdivisions without seeing the entire dataset at a time. This allows the processing the input data in an online manner. For a given input sample, the model needs to either categorize it into an existing cluster or initiate a new cluster[29]. Mini-batch Kmeans [147] The traditional K-means clustering approach needs to pre-define the K, the number of clusters before clustering. During clustering, the algorithm assigns each data point to the nearest cluster with respect to the distance metric, euclidean distance, to optimize the within-cluster sum-of-squares criterion until convergence. BIRCH[189] allows online and incremental clustering on a large amount of data, and each data point can be seen once. BIRCH builds a Clustering Feature tree (CF tree) that groups sets of (CF) subclusters and form different levels of CF nodes. During clustering, BIRCH could incrementally incorporate new arriving data points into the CF tree. BIRCH is memory efficient and doesn't need the entire dataset to be available at a time. Mini-batch Kmeans improved the traditional Kmeans approach by applying the mini-batch optimization, allowing fitting Kmeans incrementally with randomly chosen mini-batches. In Incremental K-means[127], unlike traditional K-means that need to predetermine the number of clusters (the value of K), in incremental clustering, a maximum K is fixed, and starting from $k = 1$ and the algorithm cluster data points and incrementally increase the value of k, to insert new clusters. StreamKM++[2] targets clustering of data streams. The algorithm constructs coreset trees and incrementally sample coresets that are composed of a small set of weighted examples. The Kmeans algorithm is then applied to the coreset. Recent advances in clustering combine clustering with techniques of deep learning. An example is in DeepCluster[26], an unsupervised approach using deep CNN to learn to cluster.

Other online approaches concern boosting[45] is often used to train an ensemble of classifiers. It enforces the learning capacity of weak learners through aggregation. Staring with weak classifiers or base learners, boosting groups gradually multiple classifiers and weighting them with regard to their learning capacity or performance, and dynamically adding them into a strong learner. Researchers have integrated online learning into boosting approaches, referred to as online boosting in the literature, targeting problems like binary decision [143, 6] for example, online AdaBoost is a common approach and widely applied in problems like object detectionand online tracking[10]. Online AdaBoost[123] is an extension of the AdaBoost algorithm and permits incrementally dealing with new examples through a weighting strategy on misclassified examples that is adapted in the online scenario.

We have previously mentioned some incremental learning approaches, such as incremental SVM, online boosting, and incremental clustering. In general, the learning scenario of "incremental learning" could not be considered identical to continual learning since, for continual learning, the classes should be seen sequentially (there is no Spatio-temporal consistency in data presence).

2.3.1.2 CL with reinforcement learning

Definition 2.3.4 (Reinforcement learning) *Reinforcement learning [158] refers to the learning process that which the agent needs to take actions in the environment according to its state. During learning, the agent needs to learn the optimum polity π of taking actions from a certain state of the environment that gives the maximum overall reward, as illustrated in figure 2.6.*

As a review in [8], deep reinforcement learning has incorporated deep learning models into reinforcement learning. In reinforcement learning, the problem could be modeled as a Markov decision process(MDP), as the Markov decision process is memoryless. This implies that the current state and transition are independent of historical states before the previous one.

Although not directly related to continual learning, in approaches like Neural Turing machine [58] (NTM) or differentiable neural computer (DNC)[59], the use of memory coupled with neural networks could be seen as the first step towards the solution of continual learning. In NTM, the memory is accessible through selectively reading and writing using an attention mechanism. The use of NTM or DNC would enable the model to select and encode the information and retrieve and modify the contents of the memory through a module called a controller. This avoids storing infinitely or entirely past information. The use of NTM is further found in Evolving Neural Turing Machine(ENTM) [106], a variant of NTM targeting one-shot learning. Similar to NTM, ENTM also implies an external memory system, but its difference from common NTM is that ENTM is trained by using a neuroevolutionary approach, the Neuroevolution of Augmenting Topologies (NEAT) approach, instead of gradient descent. This allows learning the evolving topology of the neural network in the controller. Further mentioned in [60], ENTM permits to read, write copy and shift within the memory while considering the evolution of the model to adapt to changes online.

Other use of memory or buffer mechanism could be found in approaches with *"rehearsal" or "experience replay"*[99]. Namely, this implies that the model store some data that it receives, repeatedly showing the model with these examples. An example is shown in [3] where the author combine ER with different RL techniques like Q learning and SARSA and shows its efficiency. In the model, ER is applied by keeping a sample set storing transitions, and the model uses data from the stored sample set for further training and parameter updates. Other examples could be found in HER[7], which stores transitions into a buffer and considers multiple goals(additional goals) for replay. However, the nature of tasks that reinforcement learning tries to solve is usually different from that of computer vision.

Another most widely recognized use of experience replay is probably in Deep Q-Network(DQN)[115]. As pointed out in [116], from a sliding window involving the last experience, the model stores the experience that it receives, including states, actions, and transitions. During training, the stored data is selected at random and replayed to the model, and the model uses q learning or mini-batches based on a $\epsilon-$greedy policy to train the model. Apart from the random selection strategy, some other sampling strategies for replay exist in state of the art.

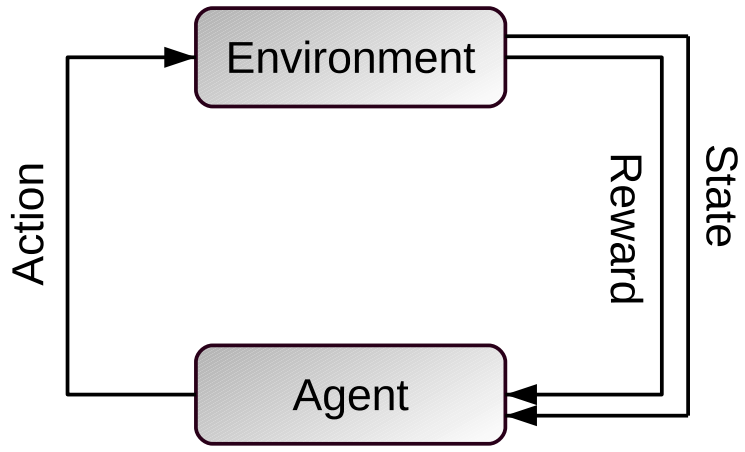


Figure 2.6: RL – the basic principle of reinforcement learning is that in the environment agent perceives the state, by taking actions, it receives the award from the environment. The goal is to choose proper actions that an agent could get maximum reward.

Furthermore, in recent advances in reinforcement learning, reinforcement learning is further combined with meta-learning as a proposal to target model adaptivity in unknown situations or awareness of the context changes in the environments due to the capacity of generalization in meta-learning approaches. A typical example is in Context Adaptation via Meta-Learning(CAVIA)[191] which extends the meta-learning MAML[42] in the reinforcement learning to learn different tasks. In CAVIA, the model is augmented with context Parameters as additional input and learning is divided into two steps– the learning of context parameters and meta-update of other network parameters. Further, in [15], CAVIA is combined with the model structural modification that introduces a supplementary neuron component called "neuromodulators" that help to adapt the plasticity or changing neuron activation of the model through the gating mechanism. [15] thus shows its adaptivity and the capacity to dynamically learn representation in different tasks.

Note that approaches with reinforcement learning often target tasks that involve taking actions to maximize the reward. Though, in recent advances in deep learning, for example, in tracking, some models have coupled RL with visual attention mechanism [157, 186]. The problem that RL targets is usually different from object recognition since in object recognition and more generally in deep learning, instead of the reward, the model needs to minimize the cost function, which was defined as the error in the prediction of the model, for example, loss in reconstructing the input image. However, the experience replay strategy originally applied in RL has inspired many advances in continual learning. A review that investigates the replay strategy in both domains, the reinforcement learning, and the continual learning could be found in [66], where the authors give a detailed survey on the application of the replay strategy and its advances in recent years, and in addition an insight from a biological aspect.

2.3.2 Continual Learning with Neural Networks

2.3.2.1 Few-shot learning approaches

Although few-shot learning is not specifically applied to learn continuously it is somehow related because it performs an adaptation of a trained model to new data. Thus, few-shot learning could be considered as a starting point towards continual learning. Here we list several categories of approaches that are often used in few-shot learning problems [96, 12].

Meta-learning based approaches are sometimes also called learning to learn in the literature, referring to learning while based on previously acquired experience. A simple way to define meta-learning, is to make an analogy with humans – knowledge isn't learned from scratch. From previously learned tasks, the model should adapt or learn faster on new tasks that it has never seen, the model could be generalized to new tasks using only a few data points.

There are two phases in meta-learning, meta-training and meta-testing. In each phase, there is a support set that contains data sampled from different tasks, on which the model will be trained composed of n classes and k examples, called "n way k shot", and a query set which contains unseen data from the same tasks where the model need to be evaluated on. In general, the model first endures a phase called meta-training during which the model is trained on several tasks with a few batches of examples. Afterward, during the second phase, meta-testing, the model will see new tasks that it has never learned during meta-training. The model needs to find parameters that could optimize loss on new tasks[43]. Its difference with transfer learning, however, is that in transfer learning the model usually needs to be trained on a task in the target domain that is different from the source domain, and meta-learning targets fast adaptation on new tasks. The optimization of meta-learning is divided into inner loop and outer loops: in the inner loop, the model is trained on the support set while in the outer loop, the model needs to be optimized with respect to a meta-objective.

In approaches of meta-learning for few-shot learning, some suggest learning to learn a better initialization on new tasks, combined with techniques such as fine-tuning. For example, in Model-Agnostic Meta-Learning (MAML) [42], meta-learning for few-shot learning, the model targets fast adaption to a new task, while only using a few data points or iterations. The model should learn parameters that could be easily generalized as initialization for new tasks, and sensitive to changes on new tasks while learning parameters by using gradient descent rules. Concretely, In MAML, iteratively, a few batches of examples are sampled (also called episodes) over different tasks, and the model is trained on these episodes from a

certain network initialization. By optimizing a meta-objective (computed on data from the same tasks but containing data that the model has never seen), the model finally fine-tunes its parameters by minimizing the meta-objective that sums the loss over different tasks. During meta-testing, the performance of MAML is evaluated on new tasks excluded from meta-training. Reptile [122] is another similar work in few-shot meta-learning. Like MAML, Reptile uses meta-learning to learn an initialization for new tasks on only a few data points, that optimize the model's performances and the capacity of generalization. The difference with MAML is that Reptile has changed the gradient update rules of the meta-objective compared to MAML. Reptile performs the final updates of model parameters as a function of the difference between initial model parameters and the model parameters after episodes of meta-training on different tasks (the difference itself could be considered as a gradient in Reptile). In [54], the authors further shed light on features extracted during meta-learning by studying the variance of inter-class and intra-class features and by defining a regularizer showing that it is important to minimize the within-class variation for the robustness of meta-learning extracted features.

In few-shot learning, apart from the meta-learning approaches mentioned above, some methods use metric learning techniques, to learn a metric that projects data to an embedding space such that different classes are separated and instances of the same class are close. In Repmet [76], the model allows few-shot detection by solving an open set recognition problem. Specifically, the model performs distance metric learning in an embedding space, with a few fully-connected layers, jointly with the learning of a mixture model for the class posterior distribution. For a given embedding vector, the model computes a distance matrix between class representatives it holds. By assuming a Gaussian distribution of different classes, the model computes the probability of given images belonging to a certain class based on the computed distances. Repmet combines a cross-entropy loss for the prediction of the right class, with respect to the ground truth label, and a distance metric modeling the inter-class and intra-class margin. In Matching Networks [174], the model adapted a specific training strategy: training is performed over the support set sampled from the task distribution where there are multiple (episodes of) batches of examples. The model classifies a given example through probability estimation by using an attention mechanism (a softmax on cosine similarities of embedding vectors) to point to specific entries stored in the external memory system.

Another approach for adapting a neural network model is transfer learning [18]. Transfer learning implies data from two different domains: the source domain where a basic model is trained on, and a target domain where the model needs to transfer its knowledge to by using information extracted from the source domain. For a detailed literature review see [178]. Transfer learning enables solving tasks using information from other domains or fields, this benefits the case where not enough data is accessible for certain tasks since on the source domain models can learn general and robust deep features. A common approach is to first train the model on the source domain, and "freeze" the parameters for certain layers, and fine-tune the model by adding some new layers while training in the target domain. The use of transfer learning can also be found in approaches targetting few-shot learning. For example, in TransMatch [183] the pretraining-finetuning mechanism is used for transfer learning to target few-shot learning. The model first pretrains a feature extractor for initialization, and further use a semi-supervised approach for model updates on "N way K shot" examples from the new classes. Compared to meta-learning approaches, in transfer learning, there is no concept as inner/outer loop or meta-training/meta-testing, but as in other applications, the use of transfer learning enables to extract robust features and initializes better the model.

In general, few-shot learning targets the generalization capacity when there are only a few training examples available. As mentioned in [166], few-shot learning methods improve the capacity of the model to generalize to unseen situations, which is crucial for its robustness and safety in domains like autonomous driving where the model needs to be aware of the changes and safely adapt to them. However, the spatio-temporal consistency is not considered in the setting of few-shot learning, although it uses only few training examples, there is no constraint on the class order of these examples. The limitation of few-shot learning approaches has been partially addressed by different continual learning approaches.

In the next few sections, we will present the common use cases and learning scenarios of continual learning, and then categorize the different approaches in the literature.

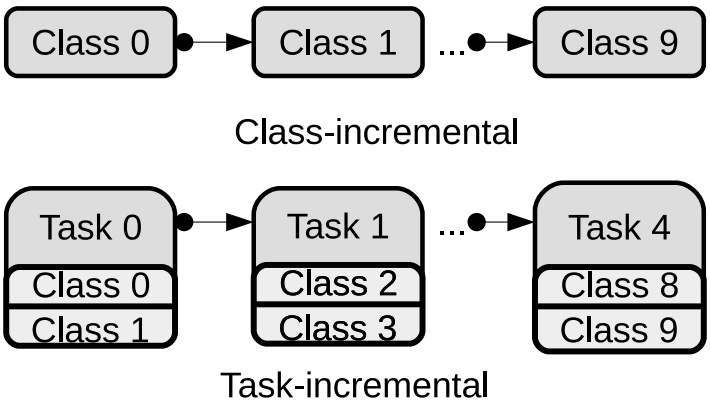


Figure 2.7: Use cases of CL: illustration of how the data sequence is presented in the class-incremental scenario and the task-incremental scenario. As an example in [167] with the MNIST dataset, in the class-incremental scenario, the model sees one digit class after the other: 0, 1 ... 9. In the task-incremental scenario, a typical example is the splitMNIST dataset where, for each task, the model needs to learn the blend of two classes, i.e. for task 0: 0 and 1, for task 1: 2 and 3, until task 5: 8 and 9.

2.3.2.2 Use cases of continual learning

In continual learning, several use cases can be considered where the input data is not i.i.d. and non-stationary, The works in the literature can be divided into 4 scenarios explained in the following (a review could be found in [167]).

Single-task class-incremental scenario

In the single-task class-incremental scenario, the model learns incrementally the representation of different classes, the input data are from the same domain but concerning different classes of objects. The evaluation would be at the end of training and the performance metric is calculated on all the categories of objects (since there is only one single task). During the evaluation, the model should predict which classes the test data belong to.

Task-incremental scenario

Here the model needs to learn multiple tasks consecutively, thus multiple tasks are presented during training. An example is to separate the dataset into several groups of classes, and one group corresponds to one task during training. Further, in a task, there could be multiple tasks. Taking the splitMNIST as an example, if each task contains two classes, the task-incremental learning scenario would be presenting classes, 0/1 as task 0, 2/3 as task 1 ... , and 8/9 as task 5. Thus its main difference with the first scenario is that in the class-incremental scenario, the model perceives the classes one by one. During the evaluation phase, the performance metric is computed task-wise, and the task identity is provided. An illustration of the difference between the class-incremental scenario and the task-incremental scenario is illustrated in figure 2.7.

Domain-incremental scenario

In the domain-incremental scenario, the input distribution changes from task to task. The model takes as input different domains, but the structures of the tasks remain similar, and the model needs to solve the task with regard to the change of input distribution. A definition can be found in [167, 73]. The task identity is not provided at test time, but the model needs to solve the task without knowing which

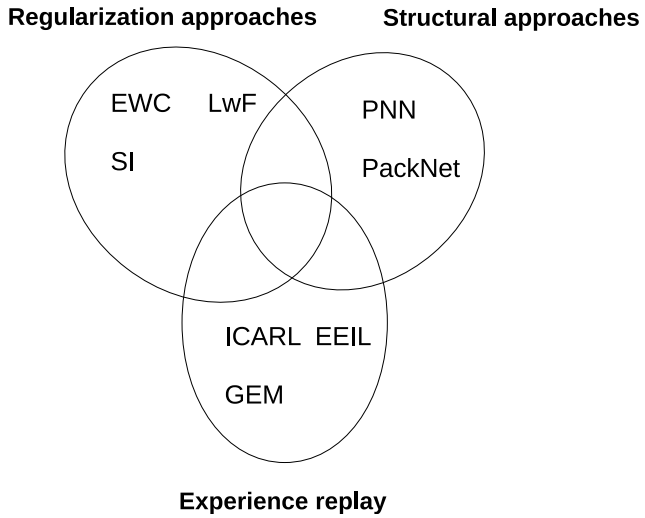


Figure 2.8: A categorization of common continual learning approaches with neural networks: regularization approaches, structural approaches, and experience replay. A detailed description of the acronyms and the corresponding models can be found in the text.

task it is facing. This implies no change in the distribution of targets, and the output space is shared between different tasks. An example, as mentioned in [73], is the Permuted MNIST dataset, where the input is 0/1, or 2/3 and the model needs to output its prediction as 0 or 1, learning new tasks while preserving its performances on the previous task. This is different from the transfer learning models that need to transfer knowledge from previous tasks onto new ones for domain adaptation [73].

Incremental learning scenario

In this setting, new data or concepts are joined gradually, and the model needs to be built incrementally without losing information acquired in the past. But unlike other scenarios, there are no constraints on the sequential arriving order in classes or tasks. The model is updated incrementally, receiving new data, potentially from all classes which are usually known in advance. Although the distinction to the other continual learning scenarios is not always made in the literature, incremental learning usually supposes that the incoming data stream is stationary.

Among state-of-the-art works, many target the task-incremental scenario and the incremental scenario. In the next section, we will take a deeper look at different continual learning approaches which are mostly supervised.

2.3.2.3 Supervised Continual Learning Approaches

One of the biggest challenges of continual learning is catastrophic forgetting, implying the tendency that during the learning of new classes or new tasks, the model risks destroying the performance on old classes or old tasks. The plasticity of artificial neural networks makes them capable of acquiring new knowledge, while not completely losing previously acquired information. Nevertheless, the more the model adapts to new incoming data the more it risks replacing the previously stored information, which is commonly referred to as the plasticity-stability dilemma. There are three main categories of continual learning approaches, adopting different strategies against catastrophic forgetting (see figure 2.8).

Regularization approaches: These approaches regularize the loss function, with respect to the relevance of past tasks or categories. This is similar to the principle of knowledge distillation used in other works since the regularization term "distills" knowledge from previous experiences. In elastic weight consolidation (EWC) [79], a regularization term is added to the optimization integrating the importance of parameters for previous tasks. This ensures that learning will be performed in a way that maintains the learned network parameters for both the previous tasks and the current one so that the perfor-

mance on the former will not be destroyed completely by incrementally learning the latter. The authors propose to use Fisher information as the regularization term. However, the limitation is that it might constrain too much the learning of new tasks when Fisher Information accumulates, leading to problems with increasing number of classes; also EWC methods often have poor experimental performances in the class-incremental scenario. In Synaptic Intelligence (SI) [185], a measure of importance is computed for each synapse so that the catastrophic forgetting could be decreased by regularizing weight updates while preserving task-relevant synapses. The importance metric denotes the sensibility of the loss function to each synapse change, and the importance metric is calculated online considering its evolution during training. A variant of the SI approach can be found in [85], where the regularization approach is further combined with an attention mechanism using Hebbian learning. This mechanism however is mainly involved during the update of the parameters modeling synaptic importance, while using a different update rule based on Hebbian learning. The author apply contrastive Excitation Back-Propagation that was first proposed in [188], modelling a top-down attention winner takes all (WTA) selective Turing model [36]. The Learning without forgetting (LwF) approach [97] trains and uses (only) data points on new tasks to learn to add task-specific parameters to the network in such a way that it allows solving both the previous tasks and the current task. A regularization term based on knowledge distillation is introduced in the loss function for previous tasks.

Structural approaches: Approaches of this category use different strategies to manage neuron resources, where training on new tasks updates weights in the network in the direction that optimizes these tasks but without overwriting weights related to the previous tasks. From a bio-plausible point of view, these approaches are able to change the modularity of the neural network facing new experiences and improve the plasticity of the model– in the literature, some models uses the terminology of "neurogenesis" [109] to describe the generation of new neurons, or "neuromodulatory" [40, 14] to describe the change in connection or neural network modularity or to make gating decisions [5]. To some extent, structural approaches reduce the redundancy of the use of neuron resources when new tasks arrive, by partially activating the neural network with respect to the corresponding task or category or reallocating new neuron resources.

Progressive neural networks (PNN) [142] create and initialize a new branch of neural network resources for a new task, as well as lateral connections with previously learned branches for the transfer of features acquired from previous tasks. This allows to alleviate catastrophic forgetting, also compared to the common pretrain-finetuning mechanism, as the authors point out, the structures of PNN does not assume task relationship, thus could handle cases even if there were few overlaps between tasks. [142] is designed for the multitasking scenarios, the limitation lies in its scalability. More precisely, as the number of tasks increases, parameters within the model will also grow. Another structural approach is PackNet [107]. Through pruning, the model selectively changes network connectivity, to obtain sparse filters that could be retrained to maintain its performance when new tasks are added. In this way, the model creates new neuron resources for new tasks, that can be combined with neurons of the old tasks.

Experience replay: In continual learning, to keep information on previous tasks, some approaches store a part of the real training examples or generate synthetic examples to alleviate catastrophic forgetting. These examples are presented regularly to the model during training so that the previously learnt knowledge would not be completely forgotten. The experience replay strategy, especially integrated with a memory system could be viewed as bio-plausible since, in human brains, there are also certain zones for the storage of memory (hippocampus). The replay does not review the entire past experiences–this thus raises the question of how to select samples and the choice of what to replay. A review of experience replay could be found in [66], where the authors summarized several selection criteria, such as the similarity between old and new tasks, or the relevance of different experiences. The Incremental Classifier and Representation Learning (ICARL) method [134] selects exemplars of past observations to construct the memory that will be replayed to the model to alleviate catastrophic forgetting. It uses the nearest mean of exemplars (NCM) to classify different objects which performs clustering regarding a distance with the mean of each class. The representation and feature extraction, however, is decoupled [28] from the classification. A knowledge distillation term is applied for feature extraction for previous classes of objects. Unlike ICARL, which decouples learning of a classifier and representation learning, in the "End-to-End Incremental Learning" (EEIL) approach [28], the classifier and representations of data are trained jointly and end-to-end with a neural network structure that is composed of a feature extractor

and a classification layer. EEIL also keeps old class examples, a memory that retains the most representative subset of a new category (representative memory). The model can be fine-tuned with these examples and also dynamically update the memory to integrate representatives of new arriving categories. In addition, a distillation term is added in the loss function, to alleviate the catastrophic forgetting of old tasks. Gradient Episodic Memory (GEM) [102] is another approach using experience replay. GEM constitute episodes of memory to prevent catastrophic forgetting, and applies gradient regularization on previous tasks. But as mentioned in [32], the applied restrictions of gradients are also limiting backward transfer (learning a new task while updating the old ones, but in the direction that might be benefiting the performance of past tasks). Also, in the case of concept conflicts where the new tasks contradict the old ones, GEM might not be applicable.

Many of the mentioned approaches alleviating catastrophic forgetting are supervised and require information of ground truth labels for data or the identification of tasks. Also, the scalability in the number of tasks/classes and the imbalance of training instances can be an issue. In the next section, we will focus on unsupervised continual learning approaches providing more autonomy because they do not require annotated data or tasks which are difficult to obtain.

2.3.3 Unsupervised Continual learning

Among existing continual learning approaches, only very few of them are unsupervised and independent from the ground truth labels. In the unsupervised setting, the model should not only prevent catastrophic forgetting like all the supervised approaches, but also adapt to task or class change independently and accordingly modify the model structure to acquire new classes. The Self-Organizing Incremental Neural Network (SOINN) [47], for example, is an unsupervised approach. The model incrementally learns the topology of input data by dynamically adding new nodes to the model. But it suffers from the limitation that it does not provide an explicit feature extraction, as many other clustering approaches and SOM. The Self-Taught Associative Memory (STAM) [154] is another, more recent, example of unsupervised continual learning. STAM does not use neural networks, instead, it is an approach based on hierarchies of clustered image features that are continually learned by selecting centroids based on distance metrics. However, as opposed to neural network-based models, it is not clear to what extent the learned representation (for example, hierarchical sets of image patches) can generalize to unseen object appearances and can be "re-used" for new object categories. Another unsupervised approach with neural networks is MAS [4] which can be trained in an unsupervised manner or using unlabeled data during learning. Similar to SI and EWC, MAS computes an importance weight as the sensitivity to the changes in parameters of model prediction (output) function. This metric is computed online and is updated when new tasks arrive.

Among unsupervised continual learning approaches, CURL [133] fits our objective of unsupervised continual object recognition. The models that we use in our proposed approaches are all based on CURL. Therefore, in the following section, we will describe this model in more detail and refer to it in the following chapters.

2.4 Model of CURL

CURL[133] targets the problem of continual unsupervised representation learning. By default, it considers a class-incremental scenario, where new objects arrive sequentially during training, one after the other. Although in this standard setting, hard object boundaries are assumed, the authors also demonstrated the effectiveness of their model with continuous boundaries ("continuous drift"), i.e. new object classes are introduced by blending examples from the currently trained class with the new class and by gradually increasing the proportion of the latter.

CURL [133] is based on a VAE and Mixtures of Gaussians for modelling different objects. The encoder is composed of a shared encoder (i.e. the first layers), and component-specific latent encoder heads (i.e. one fully-connected layer for each head). The output of these heads represent the multivariate Gaussian means and variances, $\mu_1, \cdots, \mu_K \sigma_1, \cdots, \sigma_K$, of the distributions of the K components that constitute the latent variables $z^{(k)}$ and that are used to model the approximate posterior latent variable distribution

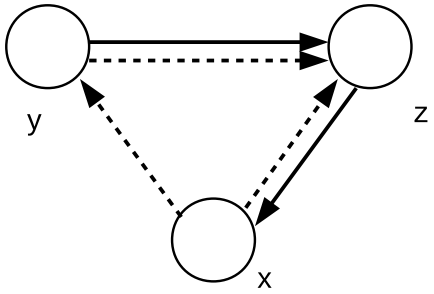


Figure 2.9: The probabilistic graphical model of CURL [133]: x is the input image, y the object category and z latent variables. The inference of the joint posterior $p(y, z|x)$ is not tractable, therefore it is approximated by the variational posterior: $q(y, z|x) = q(y|x)q(z|x, y)$ (marked with dashed arrows)

$q(z|x, y)$. The variable x represents the input image, and y the category variable. The encoder maps the input images to a shared representation for all the categories. Its output is used as the input of a fully-connected softmax layer to estimate the object category $q(y|x)$; and to update the parameters μ_k, σ_k of the corresponding component(s) k during training. Also, the prior $p(z|y)$ of latent variable z follows a Gaussian distribution. As with conventional VAEs, the image $\hat{x}$ is reconstructed from the resampled $\hat{z}$ using the decoder. Due to the intractability of $p(y, z|x)$, it is approximated by the variational posterior $q(y, z|x)$, with $q(y, z|x) = q(y|x)q(z|x, y)$. See [133] for more details.

Since learning occurs dynamically, and the model needs to adapt to different objects that arrive from time to time, the model creates a new component each time when a new category is detected. CURL considers that possible new category candidates are poorly modeled examples (i.e. outliers). The ELBO objective (see Eq. 2.20 below) reflects the estimation of the likelihood of ankexample being an inlier. By comparing the ELBO to a threshold, CURL stores those examples for which it is inferior to a threshold, and allocates a buffer to maintain these possible outlier candidates. Once the buffer is full, CURL creates a new latent encoder layer representing a new component.

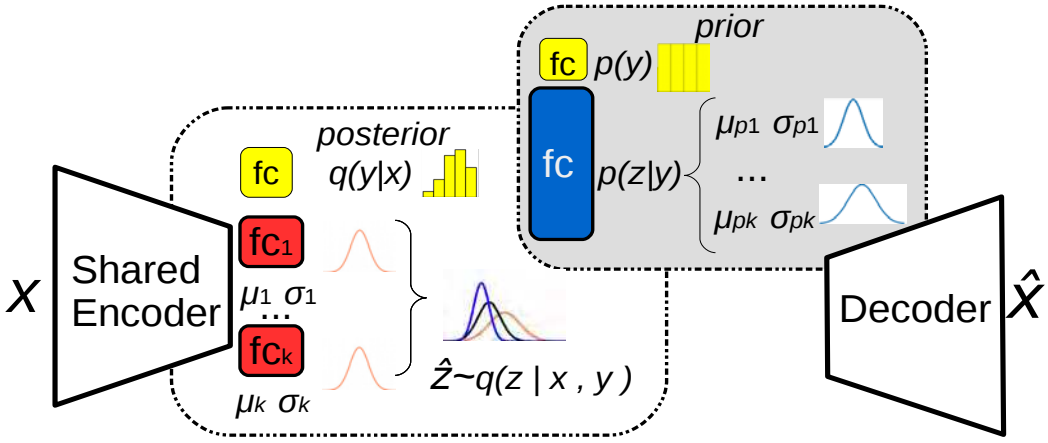


Figure 2.10: an illustration of the architecture of CURL

To alleviate catastrophic forgetting, CURL uses a mechanism called mixture generative replay in order to generate new instances instead of storing real training examples. Concretely, using mixture generative replay, CURL generates images for all the past categories that the model has already learned, mixes them with real training images of the current object. The generated images are "replayed" to the model to preserve knowledge learned in the past.

2.4.1 Loss function of CURL

Unsupervised ELBO loss of CURL:

The model optimizes a modified ELBO (Evidence Lower Bound) objective (maximizing the likelihood of the data), with input images x, categorical variable y (the index of the Gaussian component), latent variable z corresponding to the internal representation (formed by the GMM): $E(x) =$

$$\sum_{k=1}^{K} q(y=k|x)\Big[\underbrace{\log p(x|\widetilde{z}^{(k)})}_{\substack{\text{component-wise}\\ \text{reconstruction}}} - \underbrace{\mathrm{KL}(q(z|x,y=k)||p(z|y=k))}_{\substack{\text{Kullback-Leibler divergence}\\ \text{regularization on z}}}\Big] - \underbrace{\mathrm{KL}(q(y|x)||p(y))}_{\substack{\text{categorical}\\ \text{regularization}}} \tag{2.20}$$

where $q(y=k|x)$ represents the component posterior, computed by a dense layer with softmax, marked as yellow nodes in Fig. 2.10, $\widetilde{z}^{(k)} \sim q(z|x,y=k)$ is the latent code sampled from the k-th Gaussian component, modelled by a dense layer (latent encoder head), $log\ p(x|\widetilde{z}^{(k)})$ corresponds to the component-wise reconstruction loss of input images, with reconstructed image $\hat{x}$ at the output, $KL((q(z|x,y=k)||p(z|y=k))$ is a Kullback-Leibler divergence acting as the component-wise regularizer and encouraging z for each category to follow a standard normal distribution with diagonal covariance matrix $p(z|y)$, and $KL(q(y|x)||p(y))$ is the categorical regularizer that ensures that classes are well-balanced and imposes a uniform prior distribution $p(y)$ over all categories.

By maximizing Eq. 2.20, the model learns to reconstruct the input images and at the same time, due to the two regularization terms, to cluster objects into different classes in the latent space z by dynamically assigning them to different components.

Supervised ELBO loss of CURL:

CURL operates in an unsupervised manner, but the authors also consider a *supervised setting* that allows CURL to learn with ground truth labels. To this end, a slight modification of the loss function is made. In the supervised setting, CURL uses the ground-truth label y_t for supervised training for specific components. The supervised ELBO loss optimizes the prediction of the category (with a cross-entropy loss) and the reconstruction of input images:

$$E_{sup}(x) = \log q(y = y_t|x) + \log p(x|\widetilde{z}^{y_t}, y = y_t) - KL(q(z|x,y=y_t)||p(z|y=y_t)) \ . \tag{2.21}$$

2.4.2 Limitation of CURL

In the sequential setting, CURL allows learning the representation of different categories in an unsupervised and continual way. However, as CURL is an unsupervised approach, it is difficult to automatically determine the number of categories. As a result, it tends to separate objects of the same category into different sub-clusters. In addition, as with many other VAE models, the capacity to learn representations of more complex images is limited compared to GAN models or contrastive learning approaches, which are however not designed for continual learning. We will expose more of the advances and disadvantages of CURL in the next section, by comparing experimentally the performance of different models.

2.5 Novelty detection

Previously, we have briefly reviewed several continual learning approaches. As we have stated, most of the state-of-the-art works have addressed the problem in a supervised manner, and only a few works are unsupervised. One of the requirements of continual unsupervised learning is that the model should not use ground truth labels. Instead, the algorithm decides by itself if and how the incoming data is learned and integrated into the model. More specifically, in the class-incremental scenario, for example, the model needs to determine if the current observation belongs to the same object (i.e. the object that is currently learnt) or if a new object has arrived. In fact, this can be seen as a novelty detection problem. For instance, in [95], different types of data drift in the continual learning context are reviewed, which clarifies the different types of changes in data distribution the model could be facing. This also sheds

light on the relationship between continual learning and novelty detection and, more specifically, the data shift problem that the model could encounter during continual learning.

Definition 2.5.1 (Novelty detection) *Considering a stream of data, novelty detection refers to the process of distinguishing examples coming from a novel or unknown distribution from inliers with respect to a model that is built using already seen data.*

It allows the model to detect changes in the input so that it can be adjusted, which is essential in a dynamic environment. Besides, this can also be a means to tackle the stability-plasticity problem, by discerning the way or the amount of updating of the model, e.g. creating a new sub-model vs. adjusting the existing model. There are various application domains, for example, in economics to detect frauds or money laundering [49], where the models need to find suspicious transactions; Or in the medical domain, where the model needs to detect abnormalities in EEG signals [50], just to name a few. Novelty detection is a topic that researchers have been focused on for decades, and different approaches have been proposed in the literature: from historical statistical hypothesis testing approaches or information-theoretic techniques based on entropy to approaches using machine learning. We will detail the most common ones in the following.

In the literature, several concepts are close to novelty detection, but with some subtle differences.

Definition 2.5.2 (Anomaly detection) *: In anomaly detection, the model needs to find examples that are far from normal examples (abnormalities); the abnormality detection is a binary problem, "normality" or "abnormality".*

Note that this does not imply that data is arriving sequentially in a stream. Thus the term "anomaly detection" is mostly applied "offline" on a static dataset.

Definition 2.5.3 (OOD detection) *: In out-of-distribution detection (OOD), the model needs to detect and reject outliers or examples that do not fit the background inlier distribution.*

Again, this term is commonly used in the context of static datasets. Sometimes, the terms "anomaly" and "outlier" are used interchangebly depending on the application context. However, in theory, outliers can be considered as normal but are usually rejected in the model construction because they may harm its convergence or its general performance, whereas anomalies are considered data coming from a different source or corresponding to some abnormal behavior of the observed phenomenon.

Definition 2.5.4 (Concept drift detection) *: Unlike other notions mentioned previously, concept drift detection is more applied in the scenario with a non-stationary input. Concept drift detection refers to detecting gradual changes in the data distribution in the input stream; the approach needs to be incremental.*

In the following, we will give a brief and general overview of the state of the art in novelty detection and related problems mentioned above.

2.5.1 One-class Novelty detection

In the literature, some considered novelty detection as a *one-class classification problem* [129]: the problem of novelty detection, in this context, could be reformulated as the process of distinguishing test examples that are novel from common examples (inliers) used for training. Thus, the model is constructed as a binary classifier that models the distribution of inliers, and it thus needs to learn a decision boundary for novel examples.

One class SVM (OCSVM) [146] is among one of the common approaches for one-class novelty detection, an illustration is made in figure 2.11. It detects new objects with regard to all the training data (this may include all "known" objects) [163] by finding a hyperplane with a maximum margin from the origin. Support vector data description (SVDD) [163] is another algorithm for learning the decision boundary. SVDD solves the minimum hypersphere that encompasses all the inliers and separates outliers.

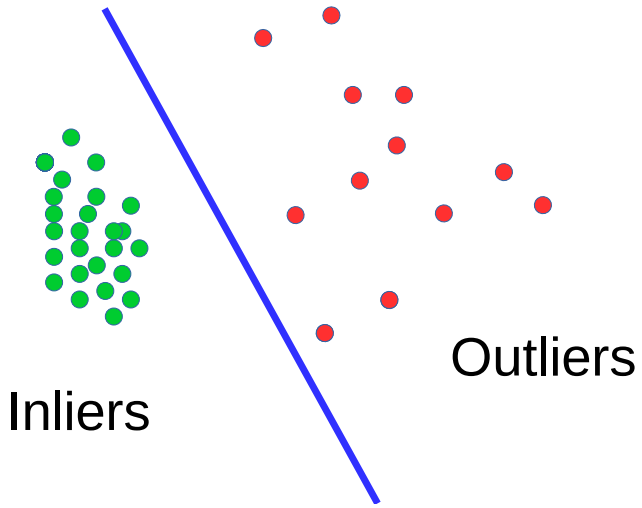


Figure 2.11: One-class classification to detect outliers: typically, the model needs to separate outliers (illustrated as red dots) from inliers (illustrated as green dots). An example in this problem or category of approach is the one-class SVM – the model only has training examples from the inlier class and needs to find the optimum hyperplane to make a binary decision.

A classifier with a rejection mechanism is another solution facing unknown objects. Compared to other classifiers, these classifiers discard ambiguous examples, i.e. examples that are very close to the decision boundary and hard to separate [11].

However, as one-class novelty detection needs to separate unknown from known objects, in the case of multiple classes, one-class SVM raises the problem of scale with respect to the increasing number of categories; That is, the model needs to consider examples of different categories as inliers. In addition, the training of SVM is often offline, which implies that the model needs to see the entire dataset. Therefore, one-class classification approaches for novelty detection are limited with respect to large-scale datasets.

2.5.2 Approaches for multi-class novelty detection/open set recognition

Multi-class novelty detection is also referred to as the open set recognition problem [145] in the literature. Some applied probabilistic approaches, where the core problem lies in estimating the density of inliers. A review of these approaches can be found in [129]. Among them, parametric approaches require prior knowledge of the data distribution to model the probability density distribution. Examples at low density are considered as outliers. One common approach is the Grubbs' test [61] (z-test), which assumes a Gaussian distribution of the data and computes the z-statistics by comparing the example with the estimated mean and covariance for accept/rejection. For more sophisticated data distributions, Gaussian Mixture Models (GMM) can be employed, and they are based on the Maximum Likelihood to estimate the parameters. Others apply Extreme Value Theory (EVT), which models the statistics of extreme values in the tails of the data distribution, to deal with extreme deviations, an example that applies EVT is in [17], which will be discussed later. Another statistical hypothesis test is the Kolmogorov-Smirnov test [110], with the Kolmogorov-Smirnov statistic measuring the upper bound of the distance between the empirical distribution and the cumulative distribution function. The two-sample Kolmogorov-Smirnov test can be used to indicate if two samples are from the same distribution, by applying the test on two empirical distribution functions.

Among various hypothesis tests, we highlight that the *Hotelling t squared test* allows determining if two samples of data belong to the same distribution, by calculating the t statistics based on the estimation of their mean and covariance. The Hotelling t squared test is described in more detail in Section IV.

Non-parametric approaches require no prior knowledge for density estimation. In a histogram analysis, for example, the continuous support of the data distribution is discretized and the proportion of the data

falling into the different bins is computed. By using histogram analysis, outliers that occur with low probability can be easily separated from inliers. However, the application of histogram analysis is rather limited in the case of multivariate and especially high-dimensional data. A more sophisticated non-parametric approach is the Dirichlet process mixture model [164].It is also referred to as an infinite mixture model in the literature, modeling a distribution over distributions [164]. One of its applications can be found, for example, in [187] where a Dirichlet process is used to build an online clustering model.

However, approaches based on the density estimation of the inliers suffer from overlaps between inlier and outlier distributions as shown in [100, 138], and may produce high scores for outliers. In addition, in an continual learning scenario, in which we do not have infinite access to all the past learning data, the estimation of the probability density distribution might be biased (since it usually needs to take into account the entire dataset). With the above mentioned approaches, the probability of an example belonging to an unknown category is still difficult to model, as the model is often built on the density distribution estimation of known classes or categories. This is also known as the "closed world assumption".

Recent advances in machine learning have allowed handling more complex and high-dimensional data and building models allowing to better cope with these challenges. In [139], a review of different state-of-the-art OOD detection approaches can be found. In common classifiers, the softmax layer is often used for scoring the chance of an example belonging to different categories. Nevertheless, the softmax function should not be considered as a real probability estimation. It also suffers from the limitation of the "closed world assumption", deteriorating its performance under the presence of unknown categories. Therefore, some recent works propose a recalibration of the softmax output enhancing the estimation of probability estimation when there are unknown categories. Openmax [17], for example, first recognizes outliers by using the Nearest Class Mean (NCM) concept, which classifies instances with respect to their distance to the "mean" of each class. The model represents the centroid of each class by the mean activation vector, averaged on the correct examples, where activation vectors are the outputs of the layer before softmax. Then the OpenMAX function is computed by fitting a Weibull distribution based on Extreme Value Theory to rescale the activation vectors, and thus the OpenMAX function allows a final probability estimation that takes unknown classes into account. A method called "Outlier Detection In Neural networks" (ODIN) [98] detects OOD by applying small perturbations to the input images as pre-processing and using the technique of temperature scaling which corresponds to using an additional hyperparameter in the softmax function, to better recalibrate the softmax function. In this way, outliers are easier to be distinguished from inliers. However, the model has additional hyperparameters related to perturbation and thus adds some complexity. Another approach called Energy-based model (EBM) [100] builds a model of energy score for each input instance to replace the softmax function for the OOD improving the discrimination from inliers.

To better detect OOD examples, likelihood ratio-based approaches have been proposed in the literature. With a likelihood ratio, the model can discard the influence of a background distribution [138], or the complexity of the input [149]. More specifically, the method proposed by [138] considers each input as the composition of a background model and a semantic model representing inliers. To detect OOD, and to alleviate the effect of the background distribution, [138] computes the ratio between inliers and the background that results in a background contrastive score and highlights the semantic model (inliers) in comparison with the background. Background statistics are obtained by training the model while adding perturbations to the inputs. [149] shows the influence of input complexity in OOD detection problems and introduced a non-parametric OOD score based on likelihood-ratio test statistics between the log-likelihood and the complexity estimation.

In the literature, other approaches exist to separate outliers from inliers based on distance metrics, where outliers are considered as examples that are far away from the distribution of inliers. In Open-Set Nearest Neighbor (OSNN) [75], for example, the Nearest Neighbor classifier is adapted to open-set problems. The Nearest Neighbor Distance Ratio, i.e. the ratio of the distance between its two nearest neighbors originating from two different classes, is used to determine whether an instance is unknown. [16] proposed Nearest Non-Outlier (NNO) that extends the NCM classifier for open set recognition problems. NNO computes the probability of an example belonging to a class, by defining a metric with respect to class mean vectors, which further allows the rejection of an example as an outlier, or marking it as unknown or novel. Other approaches are based on learning an ensemble of deep neural network

models. In [89], a deep neural network ensemble is used for the uncertainty estimation that optimizes the calibration of uncertainty prediction. Neural networks in the deep ensemble are trained in parallel. The model predicts a probability that could be either averaged for classification or applied directly in Gaussian mixture regression.

Finally, approaches based on deep learning have been proposed for novelty or outlier detection. For example, some methods use an auto-encoder trained to reconstruct input images and consider novelty as examples with high reconstruction error, i.e. the error between the target and network reconstruction. Other generative models allow generating examples and can be used to target a specific category of OOD, adversarial examples generated by the model that resemble a real image example; This is opposite to the common use of generative models which learn the input data (i.e. inlier) distribution and not outliers. They are often used to predict the most probable classes in other learning scenarios. This concept is also applied in adversarial learning, typically with GANs, where the model learns to generate examples in order to learn to detect novelties. For example, OSRCI [121, 51] generates counterfactual images that are close to true image class boundaries by using a GAN with encoder-decoder structure to augment the training set and learn to categorize known classes from the novel and unknown instances. ASG [182] generates positive and negative instances and trains an open set classifier. For negative instances, ASG considers instances that are close to the boundary of a seen category but separable by the discriminator as an instance of the unseen class. Positive examples of a seen class (not separable by the discriminator) are also generated in the case of imbalanced classes, i.e. when a class has only a few training examples.

Although the previously described common novelty detection approaches provide good results in static and stationary settings, their application to more complicated scenarios is not possible or very limited. For example, in a dynamic environment, the model receives a stream of data with considerable variations. To separate inliers from outliers, the model often needs to estimate the statistics of inliers. And this is mostly done offline on a static dataset, which does not fit our targeted continual learning scenario. Nevertheless, in the literature, some algorithms have been proposed that detect novelty in continuous data streams. We will outline the principal approaches in the following section.

2.5.3 Novelty detection in data streams

Apart from novelty detection in static datasets, other works in the literature considered data streams and an online detection. For a more detailed review see [176].Changes in the data stream's underlying distribution are also called concept drift in some works. An effective concept drift detection largely increases the autonomy of an agent, as the model improves its adaptability w.r.t. a dynamic data stream and stays consistent with the overall input data distribution. As is mentioned in [95], depending on whether there is a change in the target distribution, concept drift could be divided into real drift (with target distribution change) and virtual drift (without target distribution change).Here, we briefly introduce several categories of approaches to detect concept drift.

Statistical approaches are based on statistics of the input data stream [175]. The cumulative sum (CUSUM) [124], for example, detects changes by accumulating incrementally the difference with the estimation of the parameter of the probability distribution (the mean or z-statistic under Gaussian distribution assumption), which is compared to a threshold with regard to a tolerance value. Another example is the Page-Hinckley test [124] that is commonly used in the literature for detecting abrupt changes in the data stream. The detail of this test will be introduced in the next section.

The application of sliding windows to the input data stream could also be found in the literature, in these approaches, different structures of sliding windows are used, to replace the instance-by-instance decision paradigm. A method called "Very Fast Decision Tree" (VFDT) [38], or Hoeffding tree, is an incremental decision tree algorithm based on Hoeffding bounds. The "Concept-adapting Very Fast Decision Tree" (CVFDT)[74] is an extension of VFDT for time-changing concepts. The model dynamically maintains a sliding window of examples depending on their relevance and incrementally updates the statistics of input data to construct a decision tree to detect concept drifts. In ADaptive WINdowing (ADWIN) [20], the model maintains a sliding window of variable size that is determined automatically by the algorithm and determines if the distribution of input data has changed by comparing the means of subdivisions of windows to a threshold.

Other approaches use ensemble-based classifiers. For instance, Dynamic Weighted Majority (DWM) [86]

tackles the problem of concept drift by training an ensemble of online learners and dynamically adding, weighting and removing them based on their performance.

Concept drift detection approaches allow detecting the distribution change in the input data stream, but do not explicitly propose solutions to the recognition/categorization of inliers; For example, when an object reappears, it might simply be marked as known but not necessarily be explicitly attributed to a specific previously seen object.

2.6 Conclusion

In this chapter, we have reviewed different approaches in the literature in domains that are closely related to continual learning and novelty detection.

State-of-the-art methods for incremental and online learning have combined traditional machine learning approaches like SVM or boosting with online learning techniques, which in the scenario of dynamic data input, outperform other offline approaches by allowing to incrementally deal with new arriving data without completely losing information on the past data. In few-shot learning approaches, the generalization capacities of the model have been strengthened by using techniques like meta-learning. However, neither in incremental learning nor in few-shot learning, the sequential order or spatial-temporal consistency of the arrival of objects is considered.

And among all, the continual learning, especially in a class-incremental scenario with different objects, remains a challenging and delicate issue. In state-of-the-art approaches, the difficulties in continual learning lie in the evolution of the learned representation, for the reason that different objects/tasks may appear subsequently. Learning a new class could be deforming representations constructed from past observations. And many existing state-of-the-art approaches remain supervised and dependent on class or task identities. Concerning the more specific topic of OOD and novelty detection with neural networks, the approaches often need to operate offline for the estimation of statistics that do not fit in the context of continual learning. And those algorithms that are designed for data streams are usually not suitable for neural network models.

Facing these difficulties in the state of the art, as mentioned in the introduction, we would like to target the continual and unsupervised object recognition problem for autonomous agents while maintaining certain desired properties. Starting with novelty detection in this context as the first step towards our goal, we will show how to improve novelty detection while staying independent of ground truth labels. And at the same time, we show the role of unsupervised novelty detection in the continual learning problem that we focused on.

In the following chapter, we will introduce our contribution in novelty detection, which allows the model to better detect the arrival of new classes in unsupervised continual representation learning. Our model is an extension of CURL, previously mentioned in section 2.4, and we show how our proposed novelty detection improves the learnt representations and overall performance of the model.

Chapter 3

Novelty detection in continual learning of image sequences

3.1 Introduction

3.1.1 Novelty detection in continual learning

In chapter 1 and chapter 2, we defined the targeted context as unsupervised continual learning, which requires the agent to continuously learn from the input data that evolve in a non-predictive manner. From this standpoint, the capacity of an agent to learn, make decisions independently without using external information, and detect and adapt to the changes in its environment is crucial. The terminology "autonomy" describes the agent that learns autonomously in an unknown and unconstrained environment. Autonomy is an essential property of the agent.

New classes, by definition, are those the model is unfamiliar with. For this reason, they should be considered as outliers compared to the already seen classes (inliers), of the data stream. When the agent encounters different classes in this data stream, the identity of the classes remains unknown to the agent if learning is unsupervised. The agent has to learn to detect novel classes by itself, adapt to the change in the input data and build an effective semantic representation. The term novelty detection has been introduced in section 2.5 in chapter 2 and refers here to the detection of new classes that the model has never seen. In the literature, as mentioned in chapter 2, other terminologies such as anomaly detection or outlier detection can be found. The difference is that new classes are to be detected as outliers; but outliers are not always new classes. They can also represent examples from known classes that are poorly modeled. In addition, the term "anomaly detection" is generally used in the context of a binary decision: "inlier" vs. "outlier", but novelty detection can be employed in a multi-class problem. As resumed in chapter 2, novelty detection is a fundamental yet challenging problem. The continual learning scenario promotes the model's adaptation to dynamic variations in the input sequence each time a new class is presented and can accommodate the representation of new classes into those learned ones. However, most existing approaches are constrained in this online learning due to limited capacity to deal with streaming data. They often rather target the offline scenario and need to estimate statistics over the entire dataset. Therefore, they are not directly applicable in the continual learning scenario. In addition, in many state-of-the-art novelty detection approaches, perception is considered the only source of information provided to the agent. In the ideal case, the model can properly detect the unknown classes, but in practice, there can be similar classes in the input data stream for which the model has learned representation that are statistically close. Thus, they remain difficult to separate for the model. As a result, the model can give false negative detections by recognizing the new class as a learned one.

In unsupervised continual learning for object recognition, it is important that the agent is able to detect novelty (e.g. new objects) or, on the contrary, recognize objects that are similar. Object recognition, or from a more general perspective, object categorization, is tightly related to the notion of ontology. As mentioned in chapter 1, important steps to building the ontology include its creation and recognition.

But we will clarify the definition of several similar terms, as mentioned in chapter 1 in section 1.1.1.1: "object" and "category/class". A category can have several objects. For example, the category or class "coat" can contain a "white coat" and a "black coat" that are different objects.

In the literature, CURL [133] has given an insight into object recognition in unsupervised representation learning, which we have described in more detail in section 2.4 in chapter 2. With CURL, the model learns the representation of different classes in an unsupervised way through a VAE that models Gaussian mixtures for the distribution of different categories. The model detects novelty in the input data stream by temporarily storing and grouping poorly modeled examples (outliers) and then dynamically expanding the VAE with a new mixture component (i.e. a new dense layer). It can thus incrementally adapt to the possible category changes. In theory, this adaptability can further ensure that the agent learns the representation of new classes and builds new representations upon old ones without destroying knowledge gained so far by effectively reusing the existing components. But the representation learned by CURL regularly evolves, both due to the incremental integration of new classes and the shared generic low-level representation (i.e. the first layers) common to all the categories in CURL. Thus, like many other continual learning approaches, the internal representations of previously seen objects may gradually drift if the object has not been observed for a long time, which can influence its capacity for the online estimation of statistics, which in turn may lead to a reduced capacity in novelty detection.

Facing the difficulties of unsupervised continual learning that include both evolving representation and catastrophic forgetting, as a first step, we suppose that when an autonomous agent learns different categories, it perceives objects with temporal correlation in its perception. This temporal correlation is in fact similar to the perception of a human in a natural environment where objects are visible and focused for a certain period of time and undergo some continuous appearance variations. For example, one can have a strong or weak light exposure that can change in time, or a varying view point. This change is continuous and in most situations relatively slow compared to acquisition rate of new observations. Although, in theory, there may be numerous (physical) constraints in a real-world scenario that lead to some type of spatio-temporal consistency in the sequence of observations, in our work, we define the above mentioned temporal correlation as the fact that object classes are presented in an ordered way, i.e. one after the other and being visible for a certain period of time. Such a partially correlated and continuous perception clearly accentuates the non-i.i.d (independent and identically distributed) nature of the data stream which is known to be unsuitable for many machine learning algorithms, especially those based on neural networks. This reinforces the need for compensation mechanisms to be included in the training algorithm and the model such that the convergence and performance is not too much affected. Note that, in this thesis, we study the problem of unsupervised continual object recognition from a purely visual aspect (direct perception). Although this may seem limited with regard to more advanced sensorimotor theories that cannot conceive of cognition as a separate model of motion and perception, this can be viewed as a first step to developing a more complex model involving for example a robot interacting with actual objects and its environment. We do not make use of actions that can for example be found in approaches with an active perception of robots [118, 125]. In terms of continual learning, such mechanisms may give additional information or labels, e.g. the motion to shift the focus of attention to a different object, which facilitate the integration of new knowledge in the representation. Here, we concentrate on a more general unsupervised context without any action-perception feedback loops in order to be able to study and evaluate more easily and rigorously the different models and algorithms for continual representation learning. Moreover, a purely statistical approach based on passive perception is the first step to making effective use of object consistency. It is a problem that needs to be addressed on its own, as introduced in chapter 1. Once a model and a clear understanding of the benefits and drawbacks of a purely statistical approach are understood, one can integrate with more confidence the proposal into a larger active learning strategy, although certain adaptations will definitely need to be made.

In this chapter, we will focus on the problem of novelty detection which will help us to effectively exploit the temporal consistency mentioned above. Our model, which is based on CURL, includes the hypothesis of a class-by-class sequential order as a first step. Our objective is similar to the one of CURL, targeting unsupervised continual learning for object recognition. We have already explained the overall model in chapter 2 and will further introduce our method and the adaptations in section 3.2. By introducing a specific novelty detection algorithm based on the Page-Hinckley test – a classical statistical test to detect abrupt changes in time series – we are able to guide the learning to either update existing

class representations or create new ones. This new self-supervised way of learning based on novelty detection represents an original approach in continual learning that has not been studied before. We will introduce the Page-Hinckley test in section 3.1.2 as an effective approach to detect novelty in a noisy input data stream. Its adaptation to the continual learning scenario and our representation learning model will be presented in section 3.2.

3.1.2 Page-Hinckley test

3.1.2.1 General introduction

In chapter 2, we have introduced novelty detection approaches in the literature, some of which can be applied in change detection in data streams (see section 2.5.3). In these approaches, the model continuously receives observations that are potentially non-stationary. Other OOD and novelty detection models are often performed offline, but novelty detection in non-stationary data requires the model to estimate statistics in an online manner, which is more difficult than application scenarios where data are assumed to be i.i.d. Novelty detection in data streams is of particular interest in our continual learning scenario since the entire dataset of all objects is not presented randomly, but rather in a sequential order, class by class, and we want to detect automatically when a new object is observed. Naturally, this should correspond to abrupt changes in the input signal since, on the contrary, the appearance variations of the same object are slower and smoother. The Page-Hinckley test [124] has these properties and is well suited for our non-stationary online scenario. It is a classical approach to change detection in a 1-dimensional signal or data stream. The test assumes that abrupt variations in the input stream statistics imply changes that can be attributed for example to outliers, abnormalities or noise. In our case, the abnormalities are supposed to correspond to the moments when a new object class is observed.

The Page-Hinckley test is a variant of the CUSUM test that we have introduced in section 2.5.3. CUSUM is also a change detection approach based on the cumulated variations. We will illustrate how these two tests are related. Both tests are proposed by E. S. Page for detection of change or concept drift in data streams [124]. The Page-Hinckley test can be used to both detect high abnormal values (i.e. an increase) or low abnormal values (i.e. a decrease) in the data stream. Similar to CUSUM, Page-Hinckley maintains an accumulation of variation of values in the data stream that are greater than a tolerated value υ and compares the change in accumulation to a threshold. Specifically, the Page-Hinckley test is particularly suitable in our learning scenario since it is able to detect abrupt changes or sudden increases or decreases compared to the mean of the input data stream which is updated online.

3.1.2.2 Equations of the Page-Hinckley test

With the Page-Hinckley test, two kinds of variation can be detected: either an abnormal low point that corresponds to a decrease; or an abnormal high point that corresponds to an increase, which is similar and symmetrical to each other.

Formally, let $s(t) \in \mathcal{S} = \{s(0), \ldots, s(T)\}$ the input data stream. Let N number of samples since the last change detection at time t, $\mu_s(t)$ the mean of the input data perceived by the model computed online, and $g(t)$ the decision function. The Page-Hinckley test is initialized as Eq. 3.1 and is performed following equation Eq. 3.2, To detect a low abnormal point (decrease).

$$\begin{aligned} g(0) &= 0 \\ G(0) &= 0 \end{aligned} \tag{3.1}$$

$$\begin{aligned} \mu_s(t) &= \frac{(N-1)}{N}\mu_s(t-1) + \frac{1}{N}s(t) \\ g(t) &= g(t-1) + s(t) - \mu_s(t) + \upsilon \\ G(t) &= max\{G(t-1), g(t-1)\} \end{aligned} \tag{3.2}$$

The decision function $g(t)$ compares the input data to its mean and cumulates the difference that is greater than a tolerance υ, $G(t)$ corresponds to the minimum or the maximum value of all the historical

values of g from instant 0 until instant t. An abrupt decrease is detected if $G(t) - g(t)$ is superior to a threshold.

To give an intuition of how the test works, for decreasing signal $s(t)$, $s(t)-\mu_s(t)$ will be negative as the change occurs and the $s(t)$ falls inferior to its mean μ_s. This makes $g(t)$ decrease. Then the Page-Hinckley test compares the difference between the decreasing $g(t)$ and its maximum historical value $G(t)$, which corresponds to a normal point. Finally, the difference between $g(t)$ and $G(t)$ is compared to a threshold to decide if the test has detected a change in the input data stream. When the Page-Hinckley test detects a change, it is reinitialized and $g(t)$ and $G(t)$ are set to zero. In practice, to apply the Page-Hinckley test, one needs to choose two hyperparameters, the tolerance υ and a threshold for decision. To choose a proper tolerance value for υ, one can measure the oscillation of the input signal $s(t)$ around its mean$\mu_s(t)$, as $s(t)$ is not smooth, properly fixing υ helps ignore the little variations that do not correspond to a real class change.

Similarly, the way to detect a high abnormal point (increase) is shown in equation Eq. 3.3. Reversely, for an increasing abnormal point, the difference between $s(t) - \mu_s$ will be positive, making $g(t)$ increase. A historical minimum point $G(t)$ of $g(t)$ will correspond to a normal point. The Page-Hinckley test then compares the difference between $g(t)$ and $G(t)$ to a threshold to detect the change point.

$$\begin{aligned} g(t) &= g(t-1) + s(t) - \mu_s(t) - \upsilon \\ G(t) &= min\{G(t-1), g(t-1)\}, \end{aligned} \tag{3.3}$$

Similarly, a high abnormal is detected if $g(t) - G(t)$ is greater than a threshold.

Though the Page-Hinckley test is presented as a two-sided test for detection of both high abnormal and low abnormal values, in practice, while applying the test to detect a change in the opposite direction (high/low changes), one can always stick with the one-sided equation by reversing the sign of the input signal.

Once the change is detected at time t_{ch}, we note the change point as t_{ch} and the variables g, G and mu_s are reinitialized:

$$\begin{aligned} g(t_{ch}) &= 0 \\ G(t_{ch}) &= 0 \\ N &= 1 \\ \mu_s(t_{ch}) &= 0\ . \end{aligned} \tag{3.4}$$

3.1.3 The relationship with the CUSUM test

The Page-Hinckley test can be seen as a variant of the CUSUM test. We demonstrate here how this is derived from a one-sided Page-Hinckley test, which illustrates the relationship between both tests. The CUSUM test detects high or low changes. Low abnormal values are detected using the following equation:

$$P(t) = max(P(t-1) - s(t) + \mu_s(t) - \upsilon, 0) \tag{3.5}$$

Analogously, to detect high abnormal changes:

$$P(t) = max(P(t-1) + s(t) - \mu_s(t) - \upsilon, 0) \tag{3.6}$$

We demonstrate how Eq. 3.5 and Eq. 3.6 can be deduced from previous equations of the Page-Hinckley test by a change of variable. As one can remark, the test in Eq. 3.5 is one-sided, so we equally choose the one-sided Page-Hinckley test for demonstration (Eq. 3.2) as mathematically their expressions are close.

Eq. 3.2 is used for an abnormal low value detection in the Page-Hinckley test through $G(t) - g(t)$. Let us define a similar test $P(t) = G'(t) - g(t)$ with:

$$G'(t) = max\{G'(t-1), g(t)\}\ . \tag{3.7}$$

Then we have:

$$\begin{aligned} P(t) &= G'(t) - g(t) \\ &= max\{G'(t-1), g(t)\} - g(t) \\ &= max(G'(t-1) - g(t), 0) \\ &= max(G'(t-1) - g(t-1) + g(t-1) - g(t), 0) \\ &= max(G'(t-1) - g(t-1) - (s(t) - \mu_s(t) + \upsilon), 0) \\ &= max(P(t-1) - s(t) + \mu_s(t) - \upsilon), 0) \ , \end{aligned} \tag{3.8}$$

which corresponds to the CUMSUM test in Eq. 3.5. Thus, the only difference lies in Eq. 3.7 using $g(t)$ instead of $g(t-1)$.

For an increasing signal $s(t)$, a signal with reversed sign $-s(t)$ will be a decreasing one. Thus, detecting an abnormal high value (increase) of $s(t)$, is equivalent to detecting an abnormal low value of $-s(t)$ with its mean $-\mu_s$. In this way, one can find the equation of a CUSUM test as in Eq. 3.6.

$$\begin{aligned} P'(t) &= max(G'(t-1) - g(t-1) + g(t-1) - g(t), 0) \\ &= max(G'(t-1) - g(t-1) - (-s(t) - \mu_{-s}(t) + \upsilon), 0) \\ &= max(P(t-1) + s(t) - \mu_s(t) - \upsilon), 0) \end{aligned} \tag{3.9}$$

This shows that the Page-Hinckley test can be seen as a variant of the CUSUM test.

3.1.3.1 Discussion

The Page-Hinckley test detects abrupt changes in data streams. As previously explained, the unsupervised ELBO loss can be used as an indicator of novelty in CURL. When the model has been trained over a certain time, abrupt changes in the loss indicate that the input example is poorly modeled and implies the possible arrival of a new object class. The unsupervised ELBO loss is composed of both the image reconstruction loss of the VAE and the regularization terms based on KL divergence. The dynamics of training with category change, when an unknown category arrives, can be captured by the variation of the ELBO objective (Eq. 2.20) as shown previously in section 2.4.1 in chapter 2.

The complete Page-Hinckley test is two-sided, and both Eq. 3.2-Eq. 3.3 and Eq. 3.5-Eq. 3.6 detect changes on one side. This one-sided test is more suitable in our context of novelty detection in unsupervised continual learning as the indicator of novelty (i.e. new classes) usually only varies in one single direction, decreasing abruptly when the input is poorly modeled. We will further illustrate the indicator and will give a detailed explanation in section 3.2.

Here we give an example to show the dynamics of the chosen indicator ($E(x)$) in figure 3.1 while training with self-supervision for continual learning. One can remark that with self-supervision, when the VAE is confronted with examples that it is unfamiliar with, unsupervised ELBO loss varies abruptly. Details of the loss function and the self-supervision will be explained later in section 3.2.4. Although eventually, the model can employ some variants of how the actual self-supervision is performed relating to the details of the application of novelty detection, these dynamics of the model during training suggest that the indicator is effective in our learning scenario. Thus, as we will show experimentally, the deployment of the Page-Hinckley test enables the detection of new categories via abrupt changes of the ELBO loss.

In the following section, we will introduce our model that extends CURL by integrating the Page-Hinckley test for novelty detection, starting with an overall explanation in section 3.2. We show in section 3.2.2 that novelty detection can guide the model to identify new classes, and the model can thus predict the category automatically and use this information to guide training. In section 3.2.4, we show the loss function applied in our model and how self-supervision guides training.

3.2 A self-supervised continual learning approach

As mentioned in section 3.1 and chapter 2, CURL fits our objective of unsupervised continual learning. Our model uses CURL as a basis, with the same VAE to learn representations of different classes and the

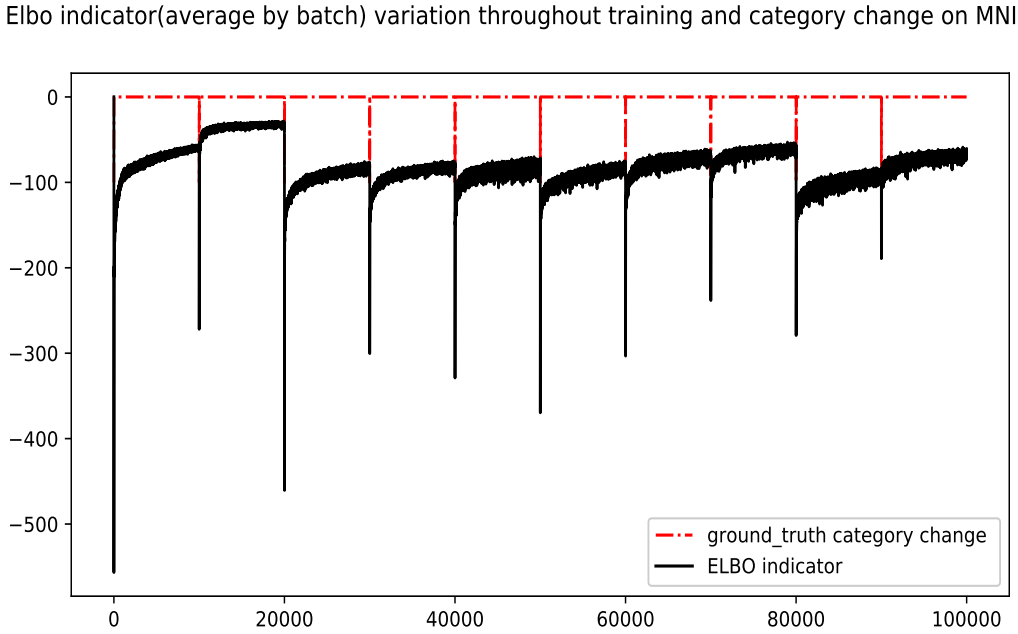


Figure 3.1: An example to illustrate the dynamics of the ELBO objective ($E(x)$) that our proposed novelty indicator is based on. During training, the unsupervised ELBO loss exhibits abrupt changes regarding its mean value when new categories arrive. For our approach, we will give a detailed explanation of how the self-supervision is determined from this signal.

same generative replay strategy. This corresponds to creating a mixture between batches of generated images of historical learned classes and the real training images of the current object for training. Our proposal is to introduce a mechanism for novelty detection to guide self-supervised learning. The main difference between our model and CURL lies in the automatic detection of new classes, i.e. the arrival of new categories is detected autonomously. As previously reviewed in section 2.4, CURL uses the ELBO loss (shown in Eq. 2.20) as an indicator and compares it to a threshold to detect poorly modeled examples that are new class candidates. But the threshold mechanism applied to the indicator has no control over the number of components created during learning. In addition, the model does not accurately detect the number of categories. Therefore, the number of components varies depending on the chosen threshold, and the clustering performance is highly correlated with the chosen value. This results in over-segmentation of clusters and requires additional effort for classification (i.e. more annotated examples or more supervision) during post-labelling. Since the number of clusters in model clustering is incoherent with the number of categories, we will assign each cluster to its corresponding category by majority vote during evaluation. But when the prediction is separated over more clusters with over-segmentation, the model will also need more annotated examples to correctly assign each cluster to a class. In section 3.3.6, we will show experimentally the influence of the number of components, both on the clustering performance and the need for annotated examples during post-labelling with varying number of components. For this reason, we expect to improve the novelty detection process. In the following sections, we will describe our novelty detection approach based on the Page-Hinckley test. We will first give a general introduction to our model and the hypothesis that is applied in section 3.2.1, followed by the presentation of the new-class detection process in section 3.2.2 and its use in self-supervision in section 3.2.4.

3.2.1 Our Model

In our learning scenario, we suppose that the model perceives the input class by class, and a class will maintain its visibility during a certain duration. The class-incremental continual learning as we

have defined it in section 2.3.2.2 remains the closest to our learning scenario. Likewise, as a first step, we assume that objects do not reappear for a second time during learning. Such a hypothesis simplifies unsupervised continual learning since known objects that reappear do not explicitly need to be recognized by the agent, and class changes necessarily correspond to new objects.

Our model uses the same neural network architecture as CURL, i.e. an extended VAE that learns a latent representation in an unsupervised manner. Previously, in section 3.1 and mostly in chapter 2, section 2.4, we have presented the model of CURL, its architecture and the loss function that CURL applies in section 2.4.1. The latent variable learned by VAE follows a single Gaussian distribution for each component or cluster. Note, however, that in our model for each category, a *single* Gaussian component is used to model the distribution of the latent variables of the given category. This is contrary to CURL, which assumes a Gaussian *mixture* learned by multiple components. Although, in theory, a Gaussian Mixture Model has a higher expressiveness and can represent any arbitrary distribution, there are several advantages for a single-component model. First, the latent representations of each object are encouraged to form a unique cluster, and thus the clusters corresponding to different objects should be more compact and semantically more consistent at the object-level. Second, a model with fewer components requires less supervision during post-labelling where each component must be assigned a class label. Hence, the overall autonomy of the system is increased.

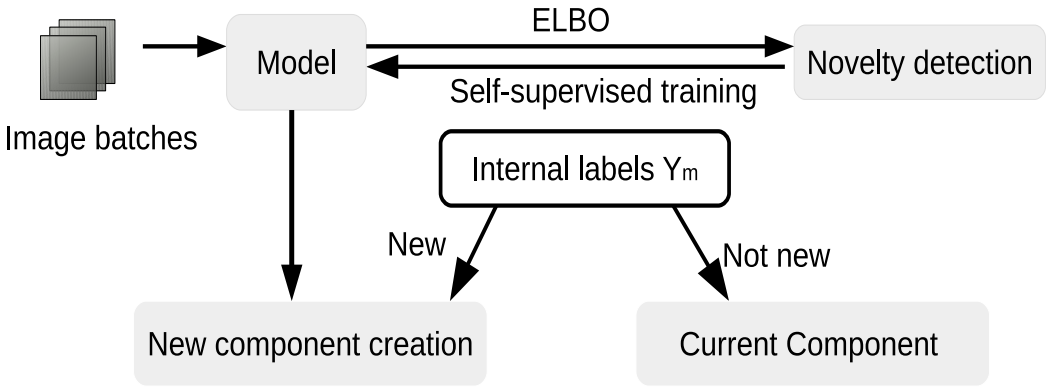


Figure 3.2: The proposed self-supervised approach using novelty detection: through the application of the Page-Hinckley test, the model can detect new object categories and define an internal label for self-supervision to guide the learning. Under the hypothesis that objects are presented one after the other (i.e. temporal continuity), our approach determines whether the current observation belongs to the currently learned object or to a new, unknown one. In the latter case, a new component is added to the latent representation of the VAE model which becomes the current component.

In the next section, we will provide more details on the new-class detection process in which we have integrated the Page-Hinckley test as a major contribution to guide the training of our model.

3.2.2 New-class detection

Approach

In section 3.1, we have briefly introduced the Page-Hinckley test as a common test to detect changes in the data stream. In our use cases, formally, let $x_t \in \mathcal{X} = \{x_0, \ldots, x_T\}$ be the input training examples

presented in a sequence. As with CURL, in our model, poorly modeled examples are considered as new class candidates, with the *unsupervised* ELBO objective $E(x)$ (Eq. 2.20) below a threshold θ. The *unsupervised* ELBO objective $E(x)$ marginalizes over all the existing categories y through a weighted sum over the $q(y|x)$. Therefore, those learned in the past are also taken into account. In cases where the introduced category is different from the current one, it can refer to a past observation. This marginalization helps partially address this special case, which reduces false-positive detection of new classes that correspond to past categories instead of new ones.

In our model, we use the Heaviside step function H to compare the ELBO objective $E(x)$ to a threshold θ to determine if the example is poorly modeled. Its equation is shown in Eq. 3.10 $H(\theta - E(x_t))$ will be equal to 1 if $E(x)$ is smaller than a threshold θ (implying an outlier); or be equal to 0 otherwise for an inlier.

$$H(\theta - E(x_t)) = \begin{cases} 1, & \text{if } \theta - E(x_t) > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{3.10}$$

We adopted the version of Page-Hinckley test as defined in [117] (in Eq. 3.12). In fact, in Eq. 3.8 in section 3.1.2.2, we have explained why Eq. 3.5 and Eq. 3.6 can be considered as a variant of the Page-Hinckley test. As previously explained, the Page-Hinckley test is normally two-sided. In contrast, Eq. 3.12 is one-sided and detects the moments of change of $H(\theta - E(x_t))$ from 0 to 1 (i.e. an abrupt increase in the loss), since many of the examples will have ELBO values that are inferior to θ, indicating that they are poorly modeled (ELBO is a log likelihood). For our problem of category change detection, only one direction of the ELBO objective is of interest, thus the one-sided Page-Hinckley test is more suitable here. In fact, the meaning of the ELBO objective varying in the opposite direction (i.e. an abrupt increase), is more difficult to assess. These are well-modeled examples, but it is difficult to determine how "good" they are by simply looking at an indicator value that is above the threshold.

Concretely, we apply the Page-Hinckley test on the $H(\theta - E(x_t))$ function in the following way:

$$g_H(t) = \max(0, g(t-1) + H(\theta - E(x_t)) - \mu_H(t) - \upsilon) \tag{3.11}$$

$$\mu_H(t) = \frac{(N-1)}{N}\mu_H(t-1) + \frac{1}{N}H(t)\ , \tag{3.12}$$

with N being the number of samples the agent has seen since the previous category change and υ being the tolerated change for each step. If $g(t)$ is greater than a threshold θ_n, then a new class is detected, i.e. a Gaussian is added to the GMM in the VAE, and we reinitialize $g(t)$ to 0, and the μ_H will also be reinitialized using Eq. 3.4. In the following, we denote this model as *ours w/o* p_n, as we also proposed a variant using a variable p_n described below.

One can note that the $H(\theta - E(x_t))$ function reverses the direction of an ELBO loss change. Its value will tend to 0 when learning tends to stabilize on some categories; otherwise will change to 1 when there are lots of outliers. The change of $H(\theta - E(x_t))$ function corresponds to an abrupt "high" value when changes occur, which is the opposite to that of the direction of the ELBO objective due to the *log* function in Eq. 2.20. In addition, compared to the ELBO loss, by definition, the $H(\theta - E(x_t))$ is bounded. This facilitates the setting of hyperparameters in the Page-Hinckley test. However, the ELBO objective itself is not bounded. For this reason, if one wants to apply Eq. 3.12 directly on the ELBO loss function, its sign needs to be reversed to apply the Page-Hinckley test on the *negative* ELBO objective.

Although, this approach gives satisfying results in most cases, in practice during training, there can be considerable ELBO fluctuations due to inter-class variability. When a high variance is present in the dataset the model might detect false positive object changes when the observations of the instances of the same object are different from previously perceived instances. In figure 3.3, we illustrate this problem with an example of the dynamics of the ELBO indicator, logged instances by instance, before and after the category change. Although the Page-Hinckley test partially smooths these fluctuations when it is applied to the $H(\theta - E(x_t))$ function, because $H(\cdot)$ takes a value between 0 and 1, there is still a large within-class variance. Thus, we will apply other statistical smoothing strategies like a running average to see if this provides a better and clearer internal supervision signal.

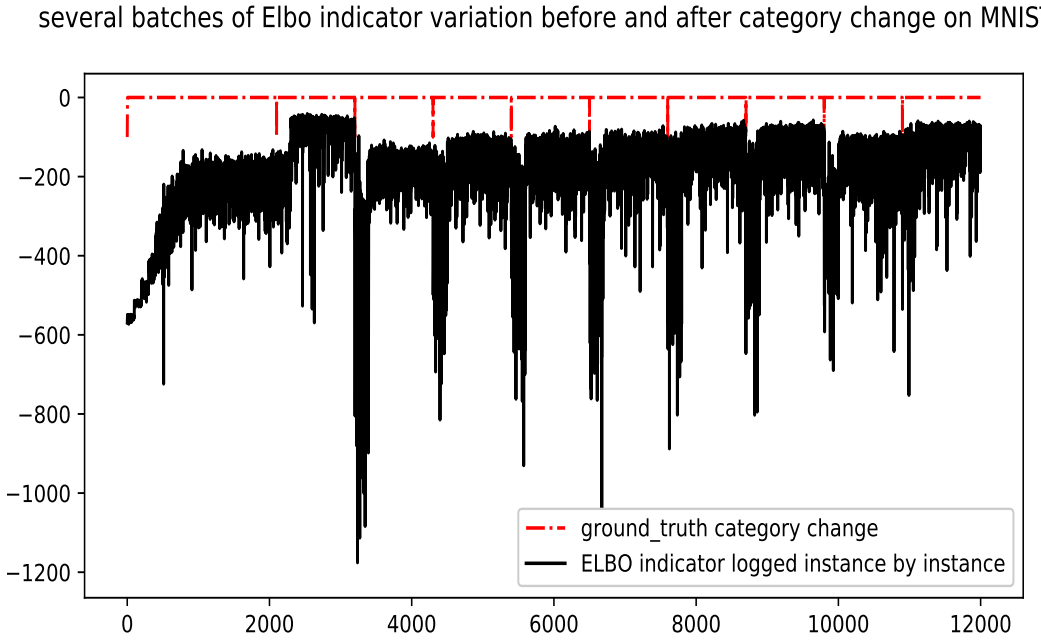


Figure 3.3: An example to illustrate the dynamics of the ELBO objective, logged instance by instance. (For better readability, we only illustrate around 10 batches before and after the category changes, each batch containing 100 images.) Since it is pertinent to detect the precise moment of category change, it makes sense to analyse these dynamics instance by instance.

We further propose to use a running average denoted as $p_n(t)$ to smooth the $H(\theta - E(x_t))$. This model is called "ours with p_n" in the following, Eqs. 3.13 to 3.15 show this variant of our approach to detect new categories. This helps to smooth potential fluctuations in the ELBO loss and to obtain a cleaner supervision signal in the presence of outliers.

$$p_n(t) = \alpha * p_n(t-1) + (1-\alpha) * H(\theta - E(x_t)) \ . \tag{3.13}$$

$$g_p(t) = \max(0, g(t-1) + p_n(t) - \mu_{p_n}(t) - \upsilon) \tag{3.14}$$

$$\mu_{p_n}(t) = \frac{(N-1)}{N}\mu_{p_n}(t-1) + \frac{1}{N}p_n(t) \tag{3.15}$$

$$\tag{3.16}$$

Model variants for comparison

To illustrate and justify different adaptations in our model, we propose several model variants to compare with. The first model variant is to directly apply the Page-Hinckley test on the ELBO objective $E(x)$ denoted as "ours w/o H" in order to study the impact of the function $H(\theta - E(x_t))$. Note that, as ELBO gives low abrupt changes with novelty, in the reversed direction as in $H(\theta - E(x_t))$ (which is 1 for poorly modelled examples, thus gives large, abrupt changes), we reverse the sign of ELBO objective to apply the one-sided test on the *negative* ELBO objective:

$$\begin{aligned} g_E(t) &= \max(0, g(t-1) + (-E(x_t)) - \mu_{-E(x_t)} - \upsilon) \\ &= \max(0, g(t-1) - E(x_t) + \mu_{E(x_t)} - \upsilon) \end{aligned} \tag{3.17}$$

$$\mu_E(x_t) = \frac{(N-1)}{N}\mu_E(x_{t-1}) + \frac{1}{N}E(x_t) \tag{3.18}$$

While applying Eq. 3.18 to calculate the mean of the ELBO objective, by reversing the sign of $E(x_t)$, the sign of $\mu_{-E(x_t)}$ is also reversed.

To justify the use of the Page-Hinckley test, we further compare our model without the Page-Hinckley test, which uses p_n as an indicator to detect new classes by comparing p_n to a threshold. We denote this model as "ours w/o P-H test".

3.2.3 Component creation and buffer

Another modification of the CURL model that we propose in our model concerns the usage of the buffer storing recent examples in the incoming data stream. In the original model, the buffer stores outlier candidates, instance by instance, until it fills its pre-defined maximum size n. In our model, once a category change is captured, the buffer is filled with all the following instances in the sequence until reaching its maximum size n. The proposed Page-Hinckley test detects abrupt changes and is capable to detect the instant (i.e. the training instance) at which the change occurs.

As with CURL, the examples in the (unfilled) buffer are not used for training immediately to prevent over-fitting resulting from too few training instances and to ensure having enough observations for each class. Once the buffer is full, the training of the new class is initiated on the newly created component, then the buffer is emptied. Overall, an important role of novelty detection in this context is to ensure the model's autonomy w.r.t. its dynamic input through the creation of a self-supervision mechanism. Thus, the model creates new components parallel to existing ones to avoid mixing new classes into the past observations. This would eventually lead to forgetting or erroneous clustering in the representation learning. Moreover, our temporal continuity constraint together with the buffer mechanism allows us to make a "hard" decision at a given instant when a new object class is detected and to favor a more consistent initialisation of new components with a continuous stream of observations that are supposed to belong to the same object.

We illustrate the overall integration of the novelty detection process in the model in figure 3.2. The novelty detection represents the core of the self-supervision that assigns an internal label (identifier) to each new component supposedly belonging to a unique object. In this way, the model can dynamically adapt to changes in the input distribution and, at the same time, incrementally learn new categories.

3.2.4 Loss function

We use internal labels deduced from our new-category detection algorithm to self-supervise the loss function that is used for training the model. As we mentioned previously in the section 3.2, we suppose that each category can be modeled by a single component. This motivates us to optimize a supervised version of the ELBO objective function $E_{sup}(x)$ that CURL [133] originally used for a supervised baseline comparison of their algorithm. However, we apply self-supervision instead of ground truth labels. The training of the ELBO objective function $E_{sup}(x)$ is self-supervised by an internal supervision signal $y_m \in \mathbb{N}$ based on the detection of new classes for training, which is determined autonomously by the model. More specifically, $y_m \in \mathbb{N}$ corresponds to the class of the instance determined by our model. It is incremented if the presence of an unseen object class is detected; otherwise, it is maintained constant as shown in equation 3.19. Note that our proposed approach is still completely unsupervised, as no ground truth labels are used.

$$y_m = \begin{cases} y_m + 1, & \text{if } g_t \geq \theta_n \\ y_m, & \text{otherwise.} \end{cases} \tag{3.19}$$

The objective is defined as:

$$E_{sup}(x) = \log q(y = y_m|x) + \log p(x|\tilde{z}^{y_m}, y = y_m) - KL(q(z|x, y = y_m)||p(z|y = y_m)) \ , \tag{3.20}$$

and we continue to use the same variable definition as in Eq. 2.20, where the first term trains a fully connected layer with softmax to predict the label, the second term minimizes the auto-encoded reconstruction error and the last term again represents the Kullback-Leibler divergence between the variational posterior of z and its corresponding Gaussian prior distribution.

3.3 Experiments

3.3.1 Datasets

We evaluated our model experimentally on the MNIST [91], Fashion-MNIST [179] (shown in figure 3.4) and EMNIST datasets [33]. The MNIST dataset is composed of gray-scale images of handwritten digits,

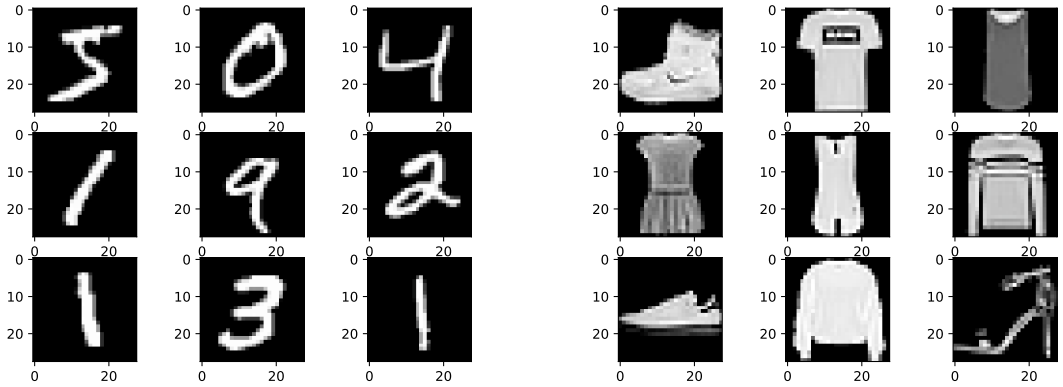


Figure 3.4: Illustration of examples in the MNIST dataset (left) and in the Fashion-MNIST (right) dataset.

with 10 classes of digits from 0 to 9. It contains 60000 images of size 28×28 for training and 10000 images for testing. The MNIST dataset is used in the experimental evaluation of CURL. Hence, for comparison, we include it as a baseline.

The Fashion-MNIST dataset contains gray-scale images of clothes (Zalando articles), from 10 categories: T-shirt, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, ankle boot. As MNIST, it contains 60000 images of size 28×28 for training and 10000 images for testing. Note that the images of Fashion-MNIST, as opposed to MNIST and EMNIST, correspond to real-world objects. They are more difficult to model and recognize since there is more variability of shape and appearance between different examples of clothes of the same class, which not the case of MNIST where digits of the same class are comparatively homogeneous. In addition, in Fashion-MNIST, some classes share inherent similarities, although labeled as different by the ground truth, for example, sneakers and ankle boots.

EMNIST (Extended MNIST) is composed of handwritten digits and letters (upper and lower case). There are several available settings, and we used the "EMNIST/balances" setting with 47 categories, where the number of examples is balanced by category, to stay comparable to the MNIST and Fashion-MNIST datasets. In this setting, the training set is composed of 112800 28×28 gray-scale images (2400 images for each class), and for the test set 18800 28×28 gray-scale images (400 images for each class). We conducted more extensive experiments on MNIST and Fashion-MNIST, with ablation studies and varying different hyper-parameters, training protocols and neural network architectures, whereas the aim of using EMNIST is mainly to evaluate our proposed methods with respect to an increase in the number of classes.

3.3.2 Compared methods

As mentioned in section 2.3.3, there are only a few methods in the literature that do unsupervised continual learning and that are comparable to our approach. The first method that we compare to is CURL [133] (see section 2.4), which also forms the base model for our approach.

Another model that we included in our evaluation is SOINN, previously introduced in the section 2.3.3. In the original SOINN model [47], there is no explicit extraction of features from the images, which implies that the original algorithm works on raw pixel intensities. However, as we have seen in the literature in chapter 2, in traditional visual tasks, feature extraction is often applied as a first step to improve

the robustness and performance of the model. For this reason, we first use the Scale-Invariant Feature Transform (SIFT) [104] for feature extraction and apply SOINN on the SIFT descriptors. However, one can note that, in the traditional SIFT algorithm, for a single input image, the number of keypoints detected by the SIFT detector can be varying. Thus, to obtain fixed-size feature vectors, we extract SIFT descriptors from a fixed grid with cell size 7×7 pixels resulting in $3 \times 3 = 9$ grid points for the 28×28 images from MNIST and Fashion-MNIST. For each keypoint, a 128-d feature will be extracted. The extracted features are concatenated and used as the input of the SOINN model.

3.3.3 Model hyperparameters

We used the same VAE architecture as CURL: a 4-layer MLP as encoder, with the number of neurons in the hidden layers being $\{1200, 600, 300, 150\}$ and a 2-layer MLP as decoder $\{500, 500\}$. In the following chapter where we will present the final and more advanced model, we will also present results for a CNN architecture replacing the MLP layers in the VAE model, which seems more suitable for an image classification problem. For the experiments in this chapter, we kept the original model architecture to allow for a fair comparison with the baseline CURL method, and to concentrate on other hyperparameter choices. The learning rate is 10^{-3} for all experiments applying Adam optimizer. For generative replay mechanism, during training, batches of real and generated images are alternated, where the generated batches contain examples distributed over all past learning categories according to the label prior. As we have mentioned in section 3.2, for this first proposal on novelty detection, we suppose that images are presented class by class, i.e. each object representing a different class, and new classes are thus introduced one after the other. Each object remains visible for at least n iterations, each iteration corresponding to a batch of N examples, since the perception is continual in this case. That is, there are no frequent and chaotic shifts or oscillations between different objects. In the experiments, we consider $n = 100$ and $N = 100$. Thus, for MNIST and Fashion-MNIST, for example, each category is trained during 10000 iterations, which results in a total number of 10×10000 iterations for the 10 categories. This is the same experimental setting as in [133]. However, for the EMNIST dataset, since there are 47 classes, 2400 examples per class, we reduce the number of iterations per class by more than 3 times, i.e. the period of training is limited to 3000 iterations for each class; Maintaining a batch of size 100 for each iteration, this results in a total number of 141000 $(47 * 3000)$ training iterations.

For our approach applying the Page-Hinckley test on $E(t)$, ours w/o p_n (Eq. 3.12), $\upsilon = 0.45$ is chosen empirically regarding the dynamics of variation in $H(\theta - E(x_t))$. For the variant "ours with p_n" (Eqs. 3.13-3.15), we set $\alpha = 0.85$, the hyperparameter for the running average, for both datasets. α controls the speed of updates and the weight on more recent data points. And $\upsilon = 0.3$ for both datasets. Since υ is related to the dynamics of variation of p_n on which we apply the Page-Hinckley test, it should be large enough to ignore fluctuations that can potentially correspond to a false positive detection, but meanwhile permits the detection of abrupt changes when new classes arrive. Since the test is applied on p_n and for the running average the current example $H(\theta - E(x_t))$ is weighted $1 - \alpha$, in our proposal with Page-Hinckley test applied on p_n, we choose υ to be $2 * (1 - \alpha)$. For other hyperparameters, we will present a more detailed study in section 3.3.5 and in section 3.3.6.

3.3.4 Evaluation metrics

We use several metrics to evaluate the performance of the different compared models. The Adjusted Mutual Information (AMI) score, Adjusted Rand Index (ARI) score, and classification accuracy are three metrics that are commonly used for the evaluation of clusters. Adjusted mutual information (AMI) is the mutual information adjusted by chance. It measures the dependency or overlap between two partitions of data. In our context, it measures thus the coherence between the ground truth label and the clustering result of the model prediction, with model prediction P_r and ground truth labels L . Concretely, we use the implemented AMI metric in the scikit-learn package [126], shown in Eq. 3.23. With the entropy being:

$$U(P_r) = -\sum_{i=1}^{|P_r|}[p_i log(p_i)] \,, \tag{3.21}$$

and the mutual information:

$$MI(P_r, L) = -\sum_{i=1}^{|P_r|}\sum_{j=1}^{|L|}[p_{ij} log(\frac{p_{ij}}{p_i p_j})] \ , \tag{3.22}$$

the AMI score is defined as:

$$AMI(P_r, L) = \frac{MI(P_r, L) - E(MI(P_r, L))}{\text{mean}(U(P_r), U(L)) - E(MI(P_r, L))} \ , \tag{3.23}$$

with $E(MI(P_r, L)$ being the expected value of the mutual information MI, and $mean(\cdot)$ the arithmetic mean of the two arguments.

The second metric is the Adjusted Rand Index (ARI), which counts among the two partitions of a dataset: the ground truth and the model prediction, the number of correctly affected pairs among all the possible pairs. The ARI, similarly to the AMI, measures the similarity between ground truth and prediction. We show the equation of ARI metric in Eq. 3.25. We denote p_s the number of pairs belonging to the same category correctly assigned to the same cluster and p_d as the number of pairs belonging to different categories and correctly assigned to different clusters, and $C_2^{n_s}$ the binomial coefficient for the total number of unordered pairs with n_s number of samples in the test set. The Rand Index RI is then defined as:

$$RI = \frac{p_s + p_d}{C_2^n} \ , \tag{3.24}$$

and the Adjusted Rand Index as:

$$ARI(P_r, L) = \frac{RI(P_r, L) - E(RI(P_r, L))}{\max(RI(P_r, L)) - E(RI(P_r, L))} \ , \tag{3.25}$$

where E is again the expected value, and $max(RI(\cdot))$ is the maximum possible Rand Index.

Both AMI and ARI are adjusted by chance to target the bias that can exist in the case where the number of clusters in the partitions is unbalanced, and that the model with a higher number of clusters scores higher while using metrics like mutual information and rand index. The values of both metrics are in the range (0,1). When the score approaches 0, this indicates the model prediction is nearly random, while a higher score signifies a better clustering performance that is close to the ground truth distribution. Besides, we also consider the metric of homogeneity [140] that evaluates if a cluster includes only points from a single class: this is to show if the prediction is homogeneous within different clusters and defined as:

$$h(P_r, L) = 1 - \frac{U(L|P_r)}{U(L)} \ , \tag{3.26}$$

where U is again the entropy, L, the ground truth label and P_r the predicted label. However, one can note that the more the model creates sub-clusters for a given ground-truth class, the more probable it is to be homogeneous.

For the classification accuracy, as there can be more clusters in the model prediction than the real number of classes during evaluation (i.e. falsely detected new classes) and since it can be given an arbitrary cluster id according to the order and how they were created in the model, the cluster id is not aligned with the category. For example, in the unsupervised learning scenario, the model can attribute cluster '0' to the digit '9', since it was the first category it perceived during learning; or in other cases, the model creates more clusters than the effective numbers of categories. Therefore, to evaluate the accuracy, an additional labelling step is necessary to assign to each cluster its most represented (ground truth) class. In order to associate a label to each cluster, a majority vote among the training set is used to determine the most frequently predicted class from the ground truth labels. This label assignment map is then used for the test and allows to compute the classification accuracy. In the following, we refer to this step as post-labelling.

In the following sections, we will present different protocols to evaluate our model and compare it with the state-of-the-art methods. We also study different variants of the model, and the order of classes in the training sequence. For instance, we consider a simple baseline case in section 3.3.5 and a case where

we introduce hard class transitions in section 3.3.11, defined as transitions where similar categories are presenting one after the other. This shows if our model is pertinent in these scenarios, at the same time if the Page-Hinckley can detect the category changes properly to guide training. We will equally show the influence of other factors that can impact learning performance, for example, the number of components created during learning.

3.3.5 Simple transition protocol

In our first scenario, classes are introduced in random order without repetition. The classes will not reappear later in the sequence if the agent has already learned them. To facilitate reproducibility, have tested 3 different fixed random orders listed in table 3.1. In the following, we will call this protocol *simple transition protocol*.

An important factor of performance is the number of components created during learning, especially when it is not predetermined but detected automatically by the model. We may give a simple and intuitive example as shown in figure 3.5. To separate data points presented by different colors, even if

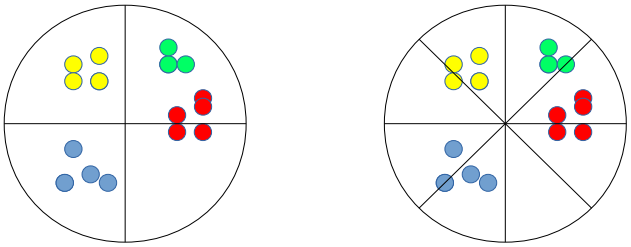


Figure 3.5: A simple illustration: the influence of the number of clusters. Colors represent the ground truth label and black lines the cluster separation. *Left:* with 4 clusters and majority vote, some of the points in the top right cluster are mis-classified. *Right:* increasing the number of clusters to 8 results in 100% classification accuracy here. This illustrates that optimising classification accuracy may favor over-segmentation in the clustering at the cost of more post-labelling, which is not a desired result for achieving a fully autonomous system.

with an arbitrary and uniform separation, when the model separates the space into 4 parts, it tends to confuse red with green. However, by simply increasing the number of separations from 4 to 8, even if the separation still seems uniform and arbitrary, it is possible to separate "red" dots from "green" dots, if later this separation is combined with a post-labelling by majority vote, for example. A higher level of division or over-segmentation actually increases the probability of high purity of each cluster and impacts the "calibration" of clustering. And when combined with a majority vote, this has greater chances to be assigned to the true category. For this reason, we will study the impact of the number of clusters in CURL and our model.

seq	MNIST										Fashion-MNIST									
1	3	6	4	2	9	5	1	8	7	0	3	6	9	4	8	2	5	1	7	0
2	9	8	5	7	6	4	3	2	1	0	9	8	7	6	3	4	1	2	5	0
3	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9

Table 3.1: Tested sequences.

3.3.6 Influence of the number of components

The process to detect novelty is completely autonomous, as previously introduced in chapter 3 and in chapter 2. CURL supposes a GMM model learned by a VAE for different categories, contrary to our

approach which aims to associate a single Gaussian (corresponding to a cluster) with each category. Our method thus aims to reduce the over-segmentation in the clustering and tries to leverage the novelty detection algorithm to form clusters that best fit to the real classes or objects (i.e. ground truth labels). During training, CURL detects novelty by comparing the ELBO loss to a threshold, as we have previously shown in section 3.2. As shown in [133], the choice of threshold impacts the moment to detect and create a new component, resulting in different numbers of components during learning. This obviously affects the GMM distribution that the model learns, which in turn results in a different clustering performance. For this reason, we will qualitatively show the influence of the number of components created during training and measure the clustering performance while varying the number of components by varying the threshold for the ELBO loss. The tested ELBO loss thresholds were chosen by dichotomy to have a sufficient number of points that are more or less homogeneously distributed. We show the tested threshold values in the table 3.2. Using the approach described above of varying the threshold, we were able to test different numbers of components, as illustrated in table 3.3.

model	MNIST	Fashion-MNIST
CURL(ELBO θ)	-250 -225 -200 -187 -175 -125	-400 -375 -362 -350 -325 -275 -250
Ours w/o p_n θ_n	3.0 2.0 1.75 1.5 1.25	3.0 2.5 2.25 1.875 1.75 1.625 1.5
Ours with p_n θ_n	1.5 1.0 0.5 0.2 0.1 0.05	1.5 1.0 0.7 0.68 0.58 0.5

Table 3.2: Tested thresholds used to vary the number of created components in CURL and our proposed method.

model	MNIST	Fashion-MNIST
CURL	10 17 29 33 37 131	20 29 37 49 69 135 157
Ours w/o p_n	10 17 33 71 95	8 9 17 64 90 124 168
Ours with p_n	10 11 19 58 85 119	13 33 53 82 105 124

Table 3.3: Tested number of components in CURL and our proposed method (resulting from thresholds in Table 3.2).

Similarly for our proposed method, we vary the number of components created during training. As introduced in section 3.2.2, for our model, once the threshold θ of ELBO loss is chosen between the least evident ELBO loss change and the most significant one, the hyperparameters υ and θ_n of the Page-Hinckley test need to be determined, playing a crucial role in the novelty detection process. We fix the value of θ and υ, while varying the θ_n for decision function $g(t)$ in Eq. 3.12 or in Eq. 3.14 of the Page-Hinckley test to create the different number of components, which is also a more logical choice in common application cases of the Page-Hinckley test. The value of θ (the ELBO loss threshold) for "ours w/o p_n" and "ours with p_n" were chosen empirically regarding the dynamics of the decision function of the applied Page-Hinckley test. We fix the threshold of the ELBO loss to $\theta = -150$ and $\theta = -190$ respectively for the MNIST and Fashion-MNIST dataset. For all the models, the maximum number of components that the model can reach is limited so that the model does not expand infinitely.

In this preliminary test and only here, we have divided the number of iterations by half (5000 iterations instead of 10000) to train the model for each category, compared to a complete experimental setting. We argue that although the model sees fewer examples, the number of iterations and examples that it sees for each category is still sufficient to show the impact of the number of components. This does not change the tendency of variation in the number of components and the relative performance between the compared methods.

We show the influence of the number of components on the clustering performances on the MNIST and Fashion-MNIST in the figures 3.6 and 3.7 respectively, using the metrics AMI, ARI, and accuracy as previously presented in the section 3.3.4.

With more components, the AMI and ARI scores decrease for CURL due to the separation of examples into excessive clusters. The AMI and the ARI metrics are sensitive to cluster numbers during clustering.

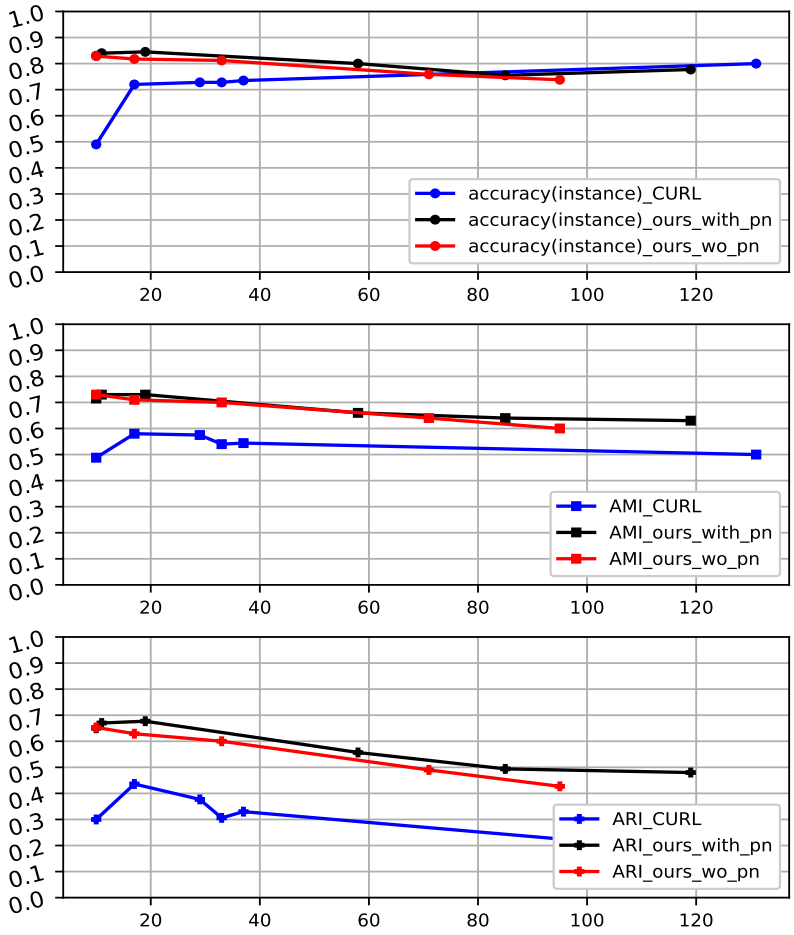


Figure 3.6: Illustration on MNIST of the influence of the number of components on clustering performance evaluated with accuracy, AMI and ARI, deduced from 1 run for each point, on CURL, ours w/o p_n and ours with p_n

But at the same time, as CURL uses a GMM to model different categories, having more clusters also helps to learn a more complex class distributions and potentially increase the overall performance. This also reduces the chance to mix different categories, due to the marginalization in the unsupervised ELBO loss in CURL and the KL category regularizer. But the over-segmentation of CURL also induces a decrease in its ARI scores. We use the Pearson correlation to measure the dependency between the ARI score and the number of components, for which the value is between -1 and +1. For our model and that of CURL, the correlation coefficients are close to -1, eventually with a small difference due to the presence of outliers, which means that the ARI score decreases almost linearly as the number of clusters increases. Globally our model outperforms CURL on ARI score and for our model, its performance is at the optimum when the model creates a number of components close to the real number of categories in the dataset (i.e. 10). For Fashion-MNIST, CURL is able to achieve a higher accuracy, but with a decreased ARI score and a larger number of components (>49), hence increasing the post-labelling cost. We have also noticed that with p_n smoothing, our model achieves slightly higher ARI scores globally when compared with the model applying the Page Hinckley test directly on the H function

Concerning SOINN, the model targets topology learning, for this reason the nodes that SOINN creates are slightly different from the clusters in the clustering approaches. Moreover, the threshold to create new nodes is determined automatically and not tuned manually. To vary the number of nodes that the

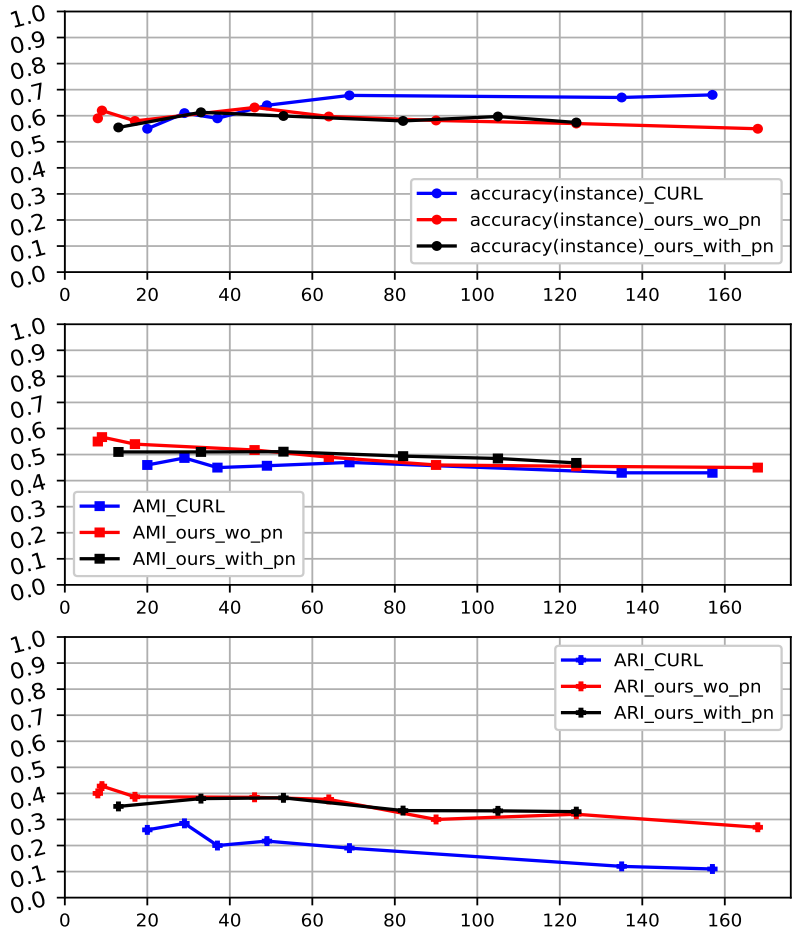


Figure 3.7: Illustration on Fashion-MNIST of the influence of the number of components on clustering performance evaluated with accuracy, AMI and ARI, deduced from 1 run for each point, on CURL, ours w/o p_n and ours with p_n

model creates during learning, we mainly vary the max edge age and the period of insertion and deletion of nodes – we change the period of node insertion/deletion in such a way that it detects and deletes nodes frequently from the input signal, by varying the period (with values taken from $2, 3, 6, 10$). Compared to a normal setting of SOINN, the model's frequency on the deletion of noisy nodes is raised considerably, so that the model will create fewer nodes and we can study the impact of number of nodes on the learning performance. The max edge age in this case is shortened to 5. Compared to a normal SOINN setting, this will slightly change the algorithm which originally aims for topology learning, but makes it thus comparable with our studied clustering approaches.

3.3.7 Optimum hyperparameter choice

The previous study illustrates the influence of number of components we aim for. More components in CURL, lead to higher accuracy and a decrease in the ARI score, we then optimize all the models with respect to an indicator that takes into account both the classification accuracy and the ARI score. We choose the mean of accuracy and ARI score as an indicator of this trade-off, while choosing hyperparameter settings with respect to the optimum indicator values. We show the indicator curves as a function of the number of components in figure 3.8 for various hyperparameters (i.e. thresholds). For CURL we have

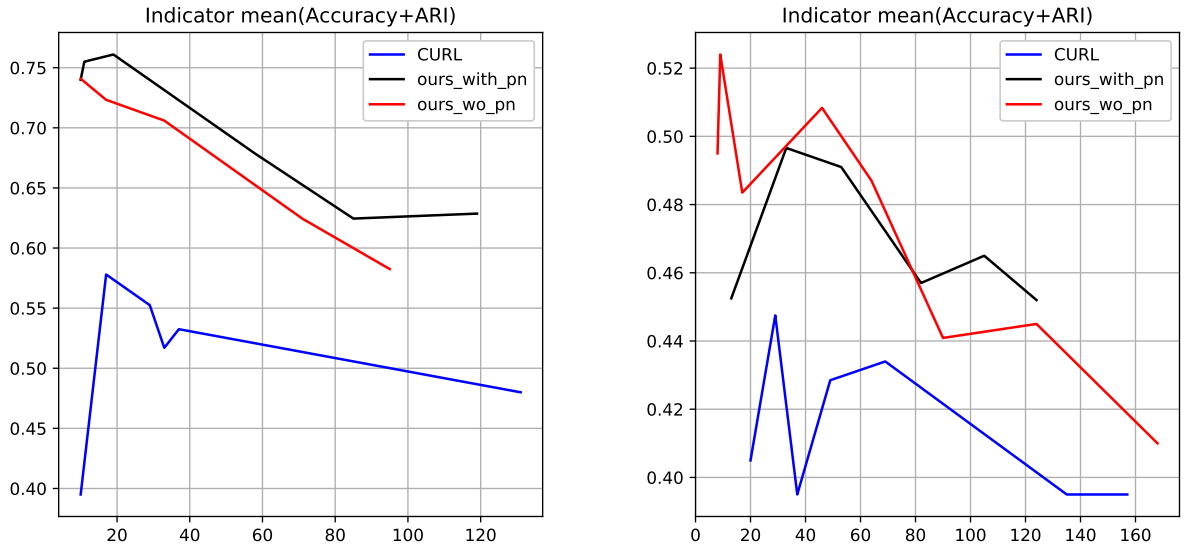


Figure 3.8: Performance criteria (mean of accuracy and ARI score) allowing to choose the optimal hyperparamters for each method.

chosen $\theta = -225$ that creates 17 components in the preliminary study for all the experiments on MNIST. For Fashion-MNIST, we set $\theta = -375$ for CURL that creates 29 components in the preliminary study. For our model applying Page-Hinckley test, for hyperparameters related to the Page-Hinckley of ours w/o p_n, we choose $\theta_n = 3.0$ on the MNIST dataset, and $\theta_n = 2.5$ on the Fashion-MNIST dataset. For ours with p_n that applies Page-Hinckley test on running average p_n smoothed $H(\theta - E(x_t))$, we choose $\theta_n = 1.5$ on that gives 11 components on MNIST and 13 components on FashionMNIST, which correspond to the minimum number of cluster at comparable indicator value. For our model's variant, "ours w/o H", in which the Page-Hinckley test is applied directly on the *negative* ELBO objective without running average p_n smoothing, υ is chosen with regard to the dynamics of ELBO Loss, when learning tends to stabilize to ignore oscillations. We set $\upsilon = 55.0$ on MNIST and $\upsilon = 70.0$ on Fashion-MNIST and $\theta_n = 1500.0$ for the parameters of the Page-Hinckley test that is applied directly on unsupervised ELBO loss without p_n smoothing. We list all the optimal hyperparameters in the table 3.4.

dataset	model	θ (ELBO)	υ (PH)	θ_n (PH)
MNIST	ours w/o p_n	-150	0.45	3.0
	CURL	-225	-	-
	Ours with p_n	-150	0.3	1.5
	Ours wo H	-150	55.0	1500.0
Fashion-MNIST	ours w/o p_n	-190	0.45	2.5
	CURL	-375	-	-
	Ours with p_n	-190	0.3	1.5
	Ours wo H	-190	70.0	1500.0
EMNIST	ours w/o p_n	-150	0.45	3.0
	CURL	-150	-	-
	Ours with p_n	-150	0.3	1.5

Table 3.4: Optimal hyperparameters for the detection of new categories for our model and CURL.

We illustrate the impact of number of nodes in SOINN separately in figure 3.9 on the MNIST dataset and in figure 3.10 on the Fashion-MNIST dataset, since the range that the number of nodes vary is different from other clustering approaches in comparison. Finally, we choose the insert/delete node period as 3 on both datasets (giving 101 nodes on MNIST and 50 nodes on Fashion-MNIST) corresponding to the best trade-off we found between classification accuracy and the ARI score.

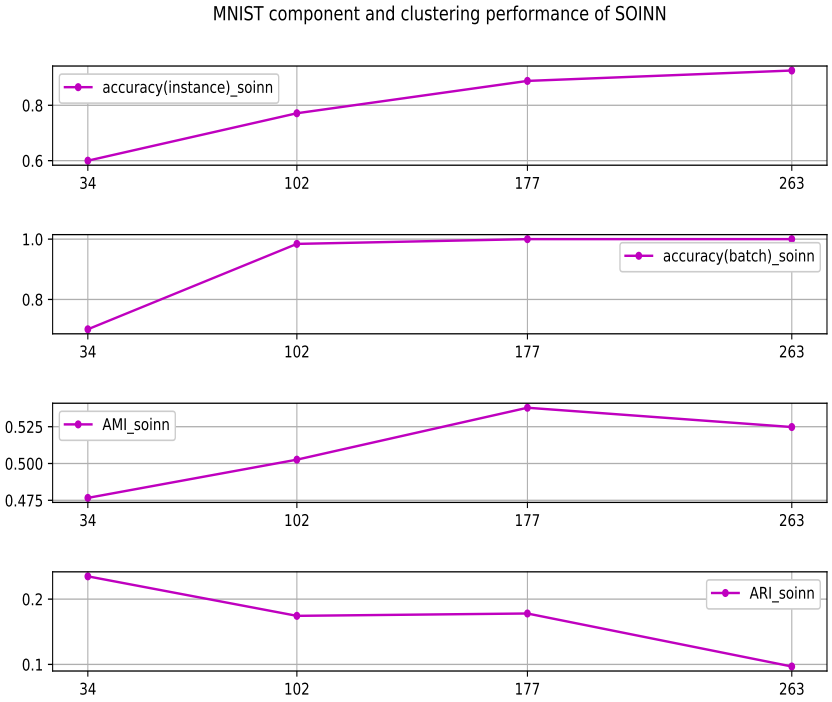


Figure 3.9: Impact of the number of nodes in SOINN on the MNIST dataset.

3.3.8 Model comparison under the simple transition protocol

We show results evaluated using the metrics AMI, ARI, the classification accuracy, and the homogeneity score, respectively in table 3.5 for the MNIST and in table 3.6 for the Fashion-MNIST dataset.

Model	AMI	ARI	# components	accuracy	Homogeneity
CURL [133]	0.6 ± 0.03	0.43 ± 0.05	17.33 ± 1.25	0.69 ± 0.04	0.65 ± 0.03
CURL supervised [133]	0.773 ± 0.009	0.753 ± 0.03	10 ± 0	0.88 ± 0.015	0.77 ± 0.01
SOINN with SIFT [47]	0.505 ± 0.003	0.174 ± 0.006	101 ± 0.816	0.77 ± 0.004	$\mathbf{0.73 \pm 0.002}$
Ours w/o p_n	$\mathbf{0.72 \pm 0.016}$	0.66 ± 0.004	9.33 ± 0.47	0.809 ± 0.017	0.705 ± 0.02
Ours with p_n	0.704 ± 0.017	0.65 ± 0.036	9.33 ± 0.47	0.8055 ± 0.03	0.69 ± 0.024
Ours w/o H	$\mathbf{0.72 \pm 0.025}$	0.665 ± 0.025	$\mathbf{9.67 \pm 0.94}$	0.818 ± 0.03	0.71 ± 0.035
Ours w/o P-H test	$\mathbf{0.732 \pm 0.03}$	$\mathbf{0.69 \pm 0.04}$	19.67 ± 3.29	$\mathbf{0.838 \pm 0.004}$	0.723 ± 0.037

Table 3.5: Comparison of our method with the state of the art on MNIST (averaged over 3 runs) for each metric (mean±SD).

Overall, our models with Page-Hinckley test "ours w/o p_n" and "ours with p_n" get a higher AMI and ARI score compared to CURL since the number of clusters of our model is close to the number of real

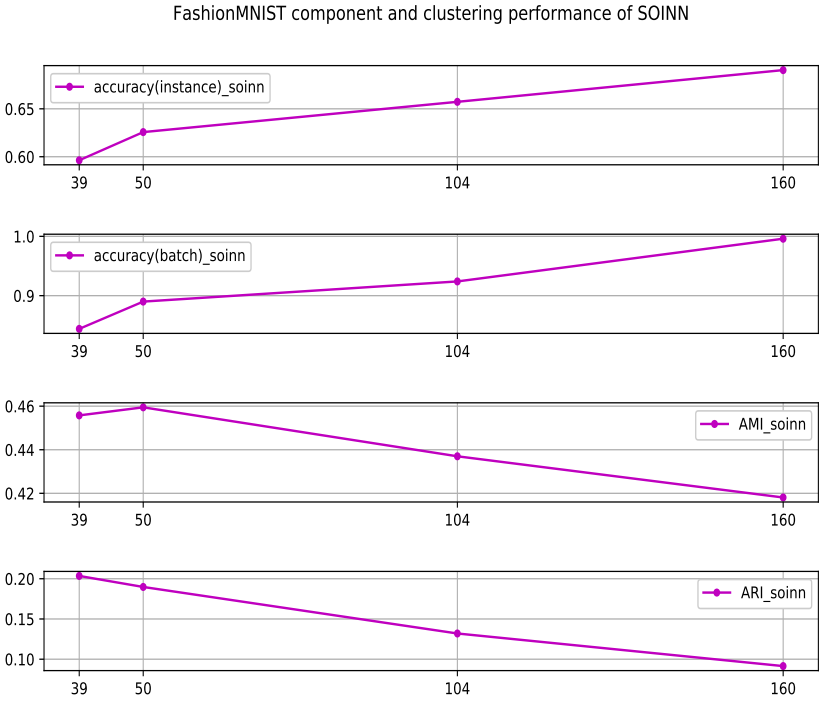


Figure 3.10: Impact of the number of nodes in SOINN on the Fashion-MNIST dataset.

Model	AMI	ARI	# components	accuracy	Homogeneity
CURL [133]	0.48 ± 0.003	0.25 ± 0.032	40 ± 5.72	0.63 ± 0.0124	$\mathbf{0.6 \pm 0.0136}$
CURL supervised [133]	0.599 ± 0.007	0.44 ± 0.02	10 ± 0	0.67 ± 0.005	0.58 ± 0.004
SOINN with SIFT [47]	0.452 ± 0.009	0.178 ± 0.012	60.6 ± 4.11	0.637 ± 0.01	$\mathbf{0.593 \pm 0.003}$
Ours w/o p_n	$\mathbf{0.573 \pm 0.013}$	$\mathbf{0.413 \pm 0.01}$	$\mathbf{16.67 \pm 1.89}$	$\mathbf{0.642 \pm 0.01}$	0.555 ± 0.01
Ours with p_n	$\mathbf{0.5697 \pm 0.016}$	$\mathbf{0.42 \pm 0.015}$	20.33 ± 2.86	$\mathbf{0.645 \pm 0.007}$	0.55 ± 0.009
Ours w/o H	0.553 ± 0.024	0.398 ± 0.05	26.3 ± 13.89	0.627 ± 0.034	0.537 ± 0.026
Ours w/o P-H test	0.55 ± 0.012	0.39 ± 0.035	74.0 ± 17.2	0.635 ± 0.015	0.538 ± 0.016

Table 3.6: Comparison of our method with the state of the art on Fashion-MNIST (averaged over 3 runs) for each metric (mean±SD).

categories, at the same time achieves better classification accuracy. This demonstrates that our method is capable of creating a clustering close to the distribution and at the same time detecting the number of categories with autonomy. On the MNIST dataset, our model variants' performance are statistically close, and "ours w/o H" slightly outperformed the other model variants applying the Page Hinckley test in terms of classification accuracy, with the number of clusters more properly detected and more homogeneous clusters. Without applying the Page Hinckley test, the model variant "ours w/o P-H test" scores the highest on model classification accuracy but fails to detect the number of components and creates subdivisions of clusters. In fact, without the Page Hinckley test, the model uses only p_n to detect changes, therefore can be exposed to local variations in losses. But on the Fashion-MNIST dataset, for the variant "ours w/o H", applying P-H test on the negative ELBO objective, one can remark that without the H function, the model sees more deviations in the number of components on the Fashion-MNIST dataset. This is because the ELBO loss is not bounded, and the choice of υ and the threshold becomes more difficult. The model is therefore more vulnerable to fluctuations in the ELBO loss function, as shown in figure 3.3. SOINN, on the other hand, is a model for learning the topology of the input data,

which is a little different from other models. As mentioned previously, we modified the hyperparameter setting while optimizing a trade-off between classification accuracy and ARI. Therefore, this modification makes the model slightly different from what the algorithm was originally aimed for. And for SOINN with SIFT, the model detects the moment to create a node to construct the topology, but the model fails to determine automatically the number of clusters since the nodes are different from clusters. It would be possible for the model to get better clustering performances at a higher number of nodes to construct the topology; but in this setting, at a number of nodes comparable to or slightly higher than that of CURL, it did not outperform CURL in clustering performances, while getting lower AMI and ARI scores than other models.

As an upper bound, we also show the performance of supervised CURL on MNIST and Fashion-MNIST. But the supervision in supervised CURL by definition is from the ground truth label. But our model and its variants use the novelty detection process as self-supervision to guide learning, which depends essentially on the model's self-constructed internal representation and the novelty detection process.

We illustrate the composition of clusters on the MNIST and Fashion-MNIST dataset of CURL with variants "ours w/o p_n" and "ours with p_n" in figures 3.11-3.13 and in figures 3.14-3.17. The composition of clusters is only from one training run as an illustration, since the number of components and the association between components and categories can vary from run to run. The clusters in the composition of clusters were illustrated by following a component creation order. Horizontally, one can observe the partition of different classes in clusters, and following columns the purity of clusters can be deduced. On both datasets, the composition of clusters is not condensed on diagonal entries, but rather divided into multiple clusters when CURL optimizes an unsupervised ELBO loss, which marginalizes across categories. Due to marginalization, the model prediction for a certain category can vary from time to time while optimizing the unsupervised ELBO loss. Our model with the Page-Hinckley test reduces false positive detection, and while optimizing the supervised ELBO loss, the model predictions globally concentrated on one cluster and correspond to our hypotheses of 1 cluster per category. Vertically from the figure, one observes different categories that are misclassified into the same component: "3" and "5" mixed by component 9 for CURL; and categories 7 and 9 mixed by component 8 and 11. For our model, the confusion between "9", "7" and "4" occurs in cluster 9 on MNIST for both "ours w/o p_n" and "ours with p_n". The confusion between categories "2", "4" and "6" on the Fashion-MNIST dataset can be observed for both models.

But since we suppose that one cluster corresponds to one class, with our model "ours w/o p_n" and "ours with p_n", when excessive components are created and the number of components exceeds 10, these components are actually "empty" and correspond to components of poor quality. This is due to the application of self-supervision in the supervised ELBO loss, for our model when excessive components are created for the same category, the model concentrates its prediction for this category on the component that is created afterwards while optimizing the loss function. In addition, as the model maintains a component count to generate images, "noisy" components are seldom predicted as the most probable and are more and more discarded by the model. On the contrary, in CURL the unsupervised loss marginalizes different categories, making "empty" components rare.

This can be illustrated by a component count that was originally proposed in the CURL paper and that accumulates the number of times a component is predicted as being the best during training, respective for MNIST in figure 3.18 for "ours w/o p_n" and "ours with p_n" in figure 3.19 and for Fashion-MNIST 3.20 for "ours w/o p_n" and 3.21 for "ours with p_n". The component count accumulates the number of times that a component is predicted as the most probable component, and assimilates the prior distribution. This count is further used to generate the distribution from which images will be generated from generative replay. The count did not include the count for the last component, as it sees no generated images but only real images. Ideally one would expect the component count to be uniformly distributed. This is almost the case in the MNIST dataset apart from the first component. This is because during training on the first category, the model sees only real images instead of using the mixture generative replay strategy. In the first half, components have a slightly higher count than those in the second half, since the model incrementally learns more and more categories. But on the Fashion-MNIST dataset, one could observe that some components score low as they are rarely predicted as the most probable. However, a side effect is that while the cumulative count permits discarding poorly modelled examples, when the model

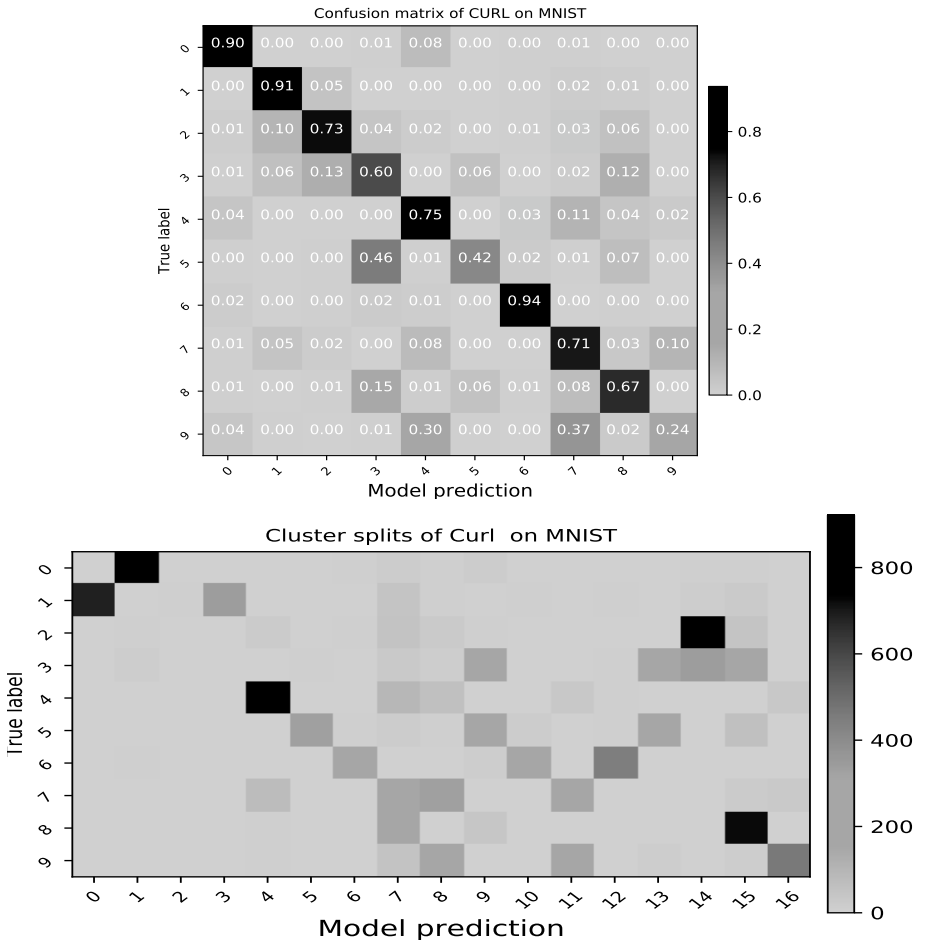


Figure 3.11: Confusion matrix averaged over 3 runs and composition of clusters between ground truth and predicted cluster components for CURL on MNIST.

sees some categories with similarity, its prediction is imprecise. This can be a source of bias in the prior estimation, for example for the Fashion-MNIST dataset. For example for "ours w/o p_n", apart from the first component, some categories also have their component count higher than the others. This can also be a source of confusion.

3.3.9 Influence of class number increase

In this section, we will study the influence the number of classes on the clustering performance and compare different models. To this end, we take the simplest order, from 0 to 47 on the EMNIST dataset. We evaluate the three clustering performance metrics, the classification accuracy, the AMI, and the ARI each time a new category is introduced. For each point of the curve, the model has seen n classes and a new class is added compared to the previous point. We illustrate the results of our model with Page Hinckley test with and without p_n, the results of SOINN SIFT and the performances of CURL in figure 3.23. As one could expect, the performance declines as new categories are incrementally introduced. A possible reason is that, as the number of categories increases, the number of images generated for each component decreases since the batch size and the total number of iterations the model will see generated batches is fixed. This leads to fewer generated images. To this end, we illustrate the component count for our model on the EMNIST dataset in figure 3.22, which refers to the normalized count of the number of times each component is predicted as the best. From this distribution, the model will generate images for generative mixing. The count indicates that the distribution is almost uniform, but fewer and fewer

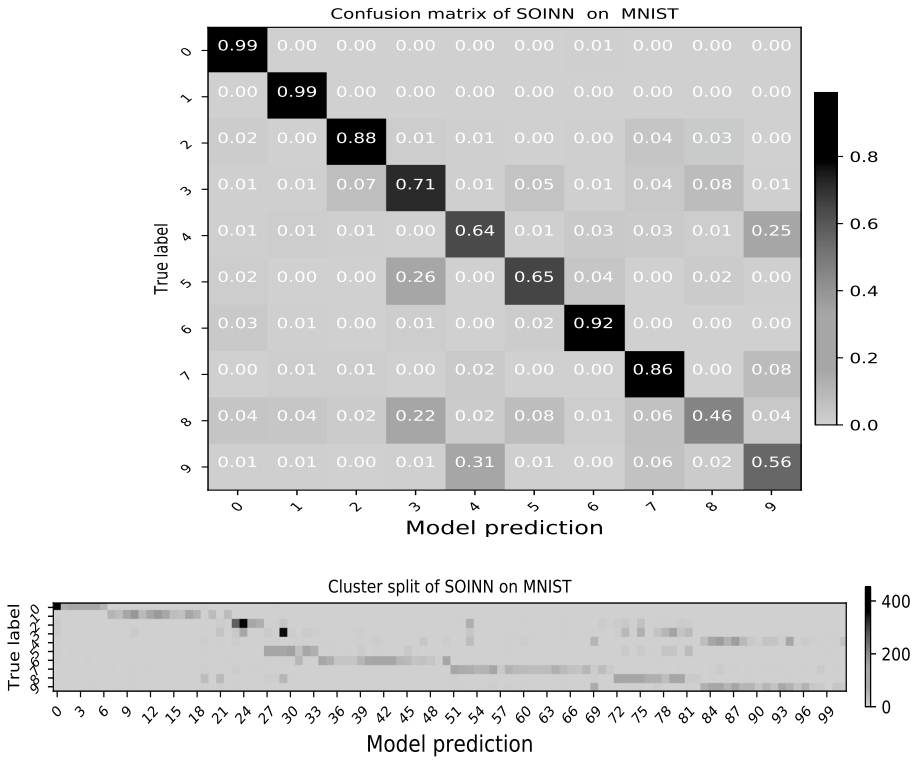


Figure 3.12: Confusion matrix averaged over 3 runs and composition of clusters between ground truth and predicted cluster components for SOINN on MNIST.

examples are generated for each component, which is a possible source of change in performance when the number of categories to be learned increases.

In table 3.7, we show the results of "ours w/o p_n", "ours with p_n" (application of the Page-Hinckley test on the smoothed indicator), and the performances of CURL on the EMNIST dataset. Other models are not illustrated as they are variants of our proposal and we only evaluate those optimum, for example, "ours w/o H" since they were outperformed by "ours w/o p_n" and "ours with p_n" in terms of stability. Our model was capable to detect almost perfectly the number of categories in the dataset (47), and outperforms CURL in terms of AMI and accuracy although having a worse homogeneity and slightly worse ARI. In fact, the general drop in accuracy can be explained by the increase in the number of categories, 47 instead of 10, and although we were applying the generative replay strategy just the same as on MNIST and Fashion-MNIST, having more categories also leads to less generated examples and may bias the model prediction towards the current examples while optimizing the supervised ELBO loss as in Eq. 3.20.

model	AMI	ARI	# components	accuracy	homogeneity
CURL	0.387 ± 0.009	0.11 ± 0.008	150 ± 0.0	0.322 ± 0.007	0.453 ± 0.011
SOINN SIFT	$\mathbf{0.454 \pm 0.0133}$	$\mathbf{0.137 \pm 0.01}$	187 ± 21.9	$\mathbf{0.43 \pm 0.036}$	$\mathbf{0.55 \pm 0.025}$
Ours w/o p_n	0.365 ± 0.025	0.062 ± 0.015	$\mathbf{43.4 \pm 0.94}$	0.292 ± 0.02	0.332 ± 0.026
Ours with p_n	0.395 ± 0.02	0.08 ± 0.028	$\mathbf{45.6 \pm 1.24}$	0.329 ± 0.038	0.363 ± 0.028

Table 3.7: Comparison with the state of the art on EMNIST (averaged over 3 runs) in the simple transition scenario (mean±SD).

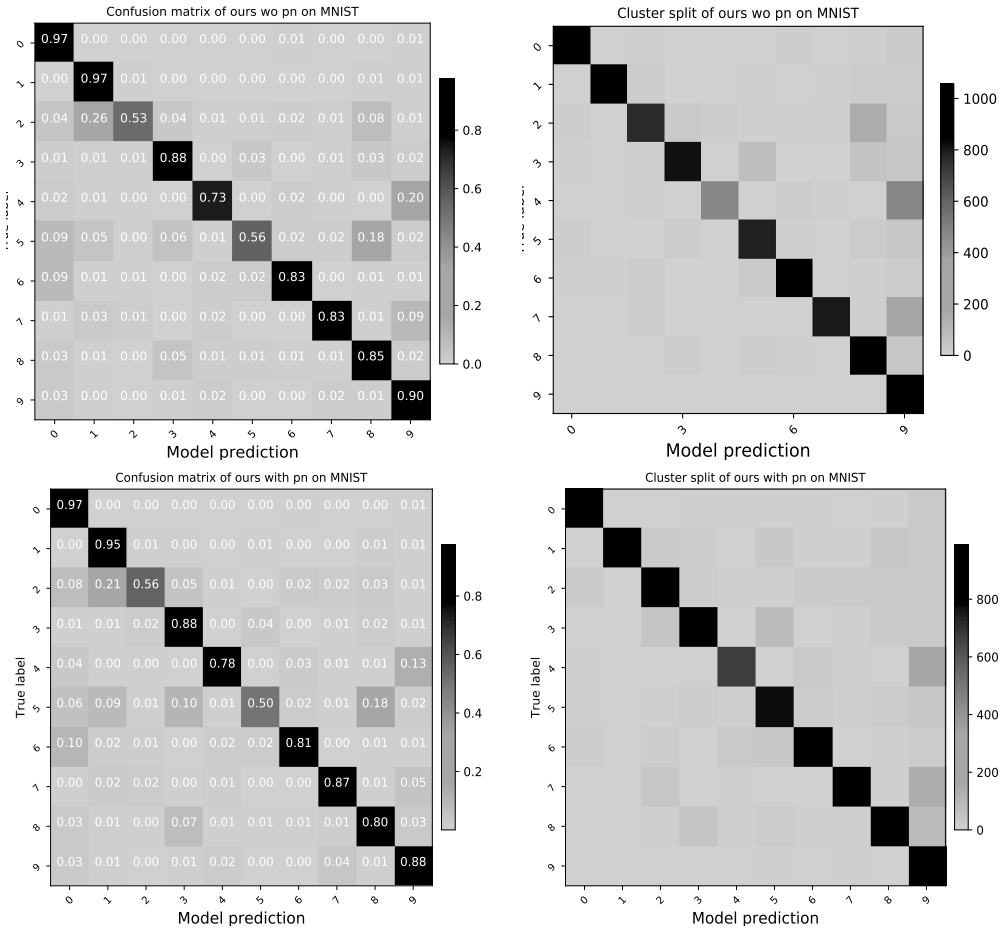


Figure 3.13: Confusion matrix averaged over 3 runs and composition of clusters between ground truth and predicted cluster components for our approach on MNIST: "ours w/o p_n" (top) and "ours with p_n" (bottom) (the darker the cells the more instances they represent).

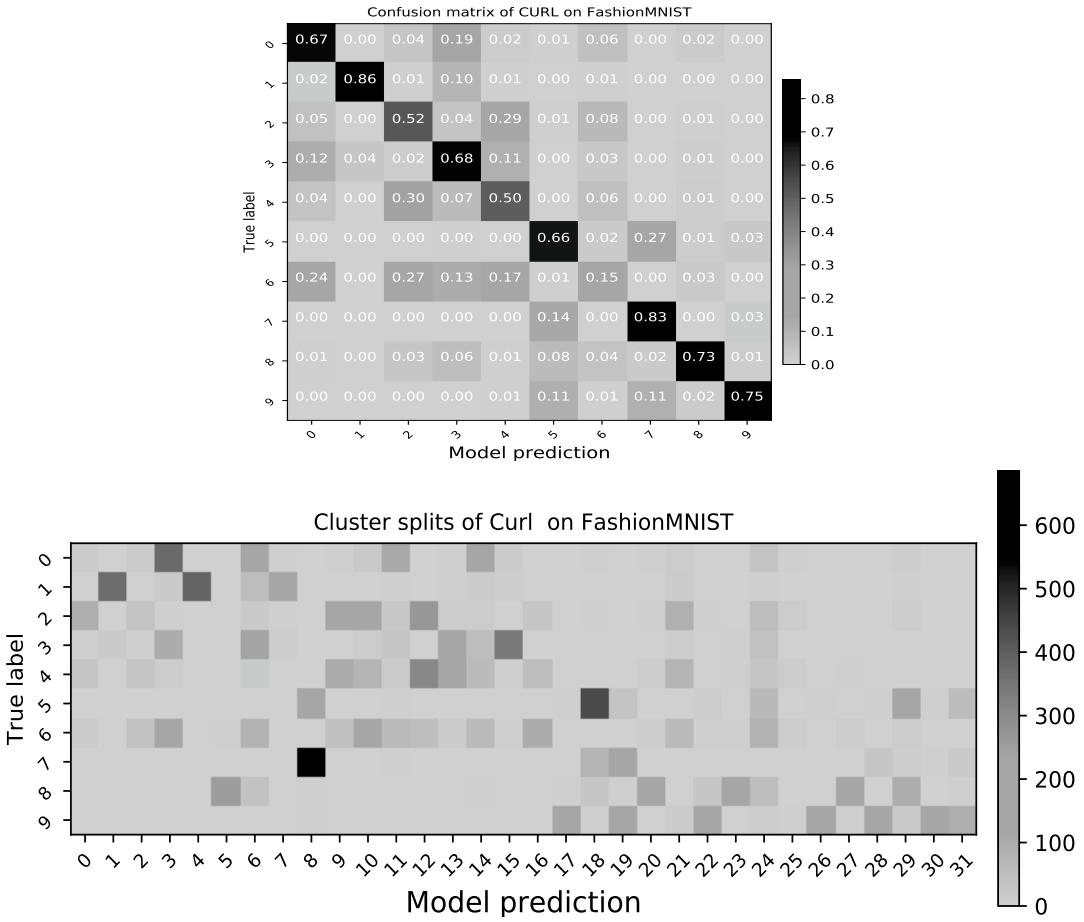


Figure 3.14: Confusion matrix averaged over 3 runs and composition of clusters between ground truth and predicted cluster components on Fashion-MNIST of CURL (the darker the cells the more instances they represent). The cluster split of CURL on FashionMNIST dataset.

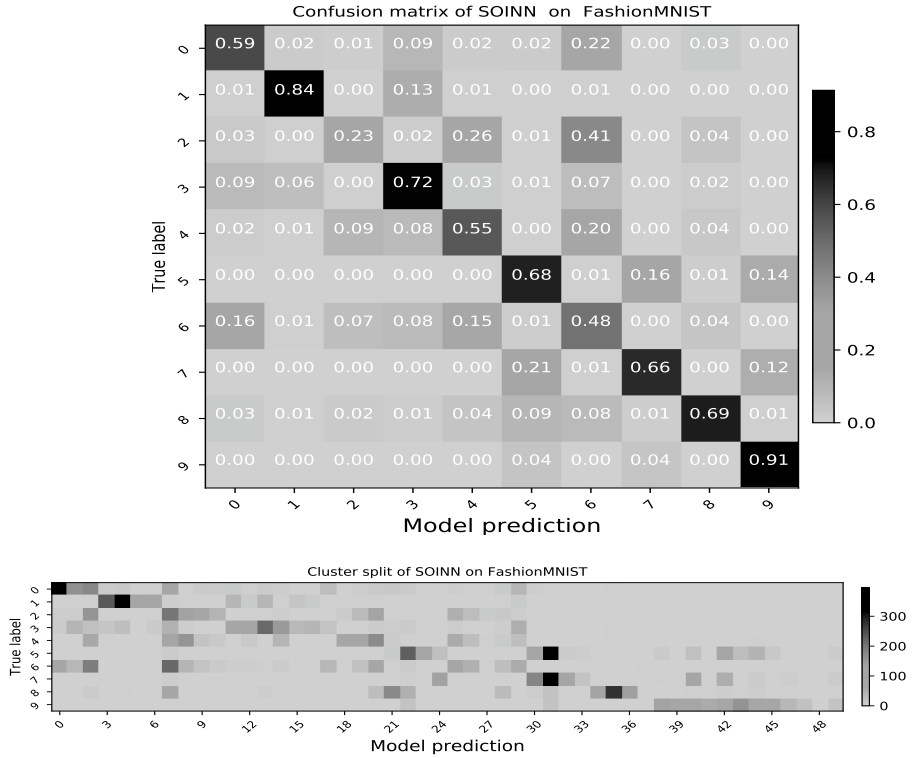


Figure 3.15: Confusion matrix averaged over 3 runs and composition of clusters between ground truth and predicted cluster components for SOINN on Fashion-MNIST (the darker the cells the more instances they represent). The cluster split of SOINN on FashionMNIST dataset.

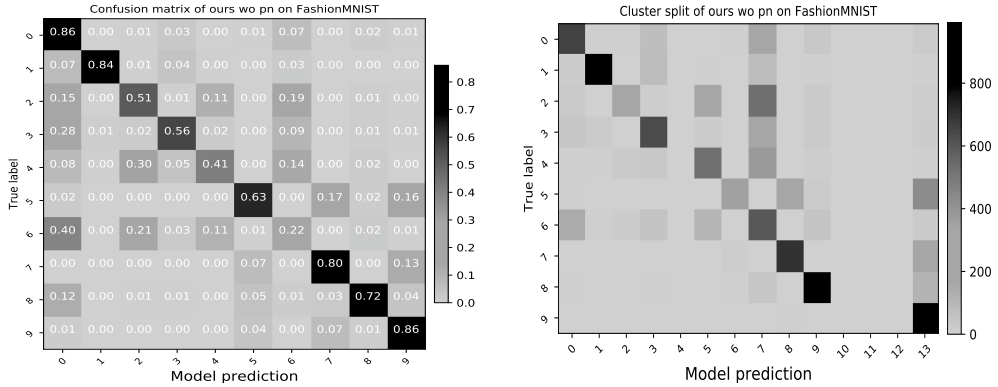


Figure 3.16: Confusion matrix averaged over 3 runs and composition of clusters between ground truth and predicted cluster components on Fashion-MNIST for "ours w/o p_n" (the darker the cells the more instances they represent).

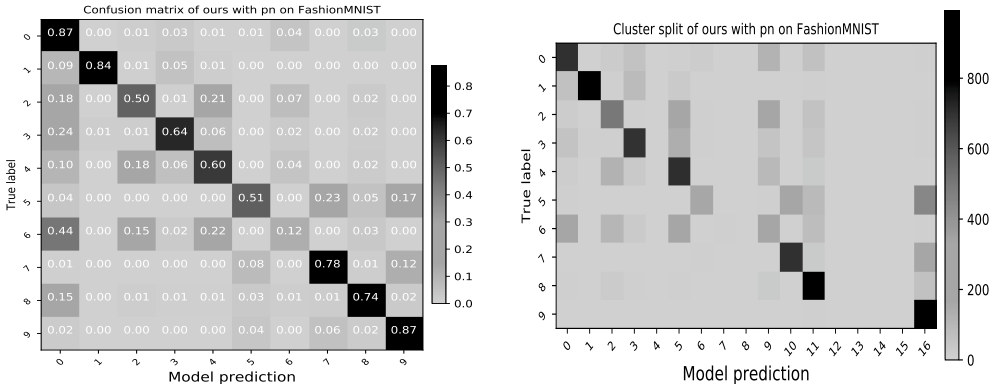


Figure 3.17: Confusion matrix averaged over 3 runs and composition of clusters between ground truth and predicted cluster components on Fashion-MNIST for "ours with p_n" (the darker the cells the more instances they represent).

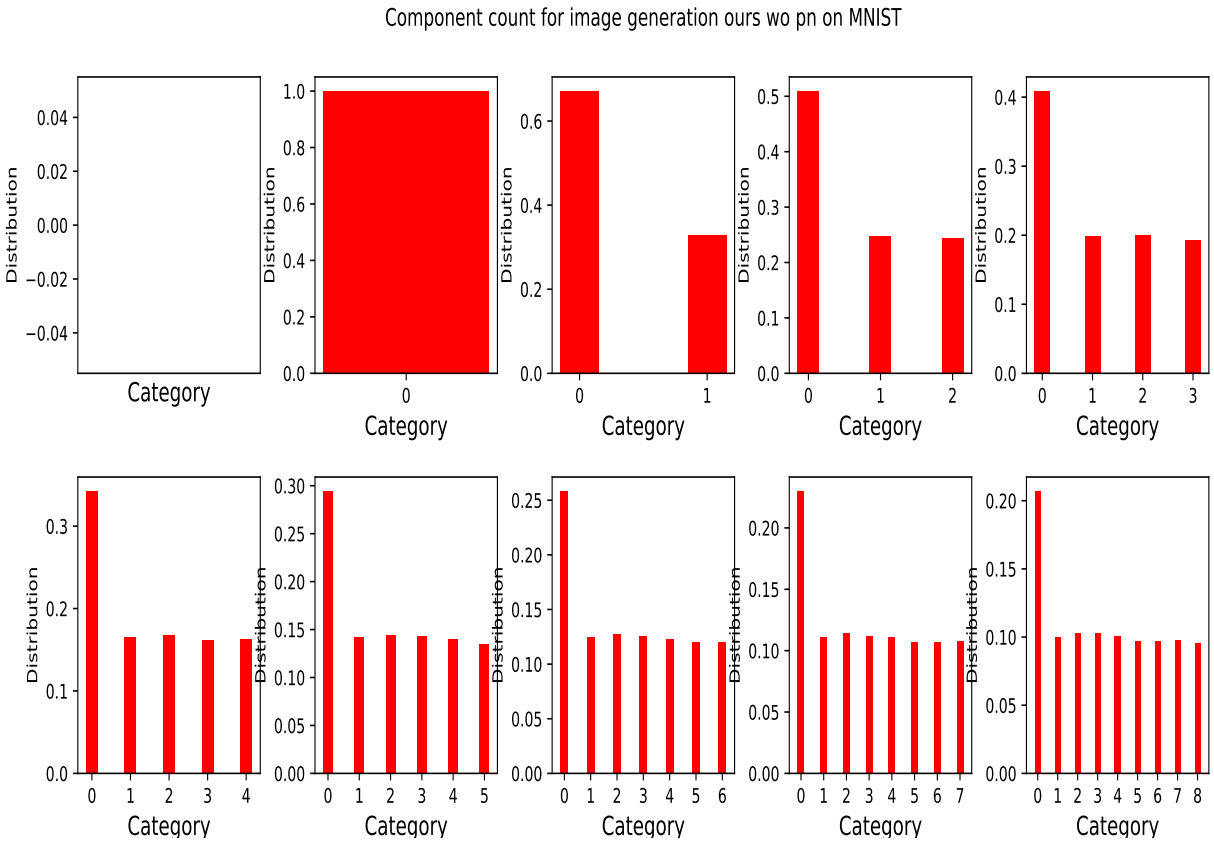


Figure 3.18: The distribution from which images are generated: each bar subplot correspond to the count of times a component is predicted as the best normalized by sum, and they are plotted in temporal order from left to right. Evolution of cumulative component count during training on the MNIST dataset for seq 1, which models the distribution of generated images (from which labels and images are generated) for our model "ours w/o p_n".

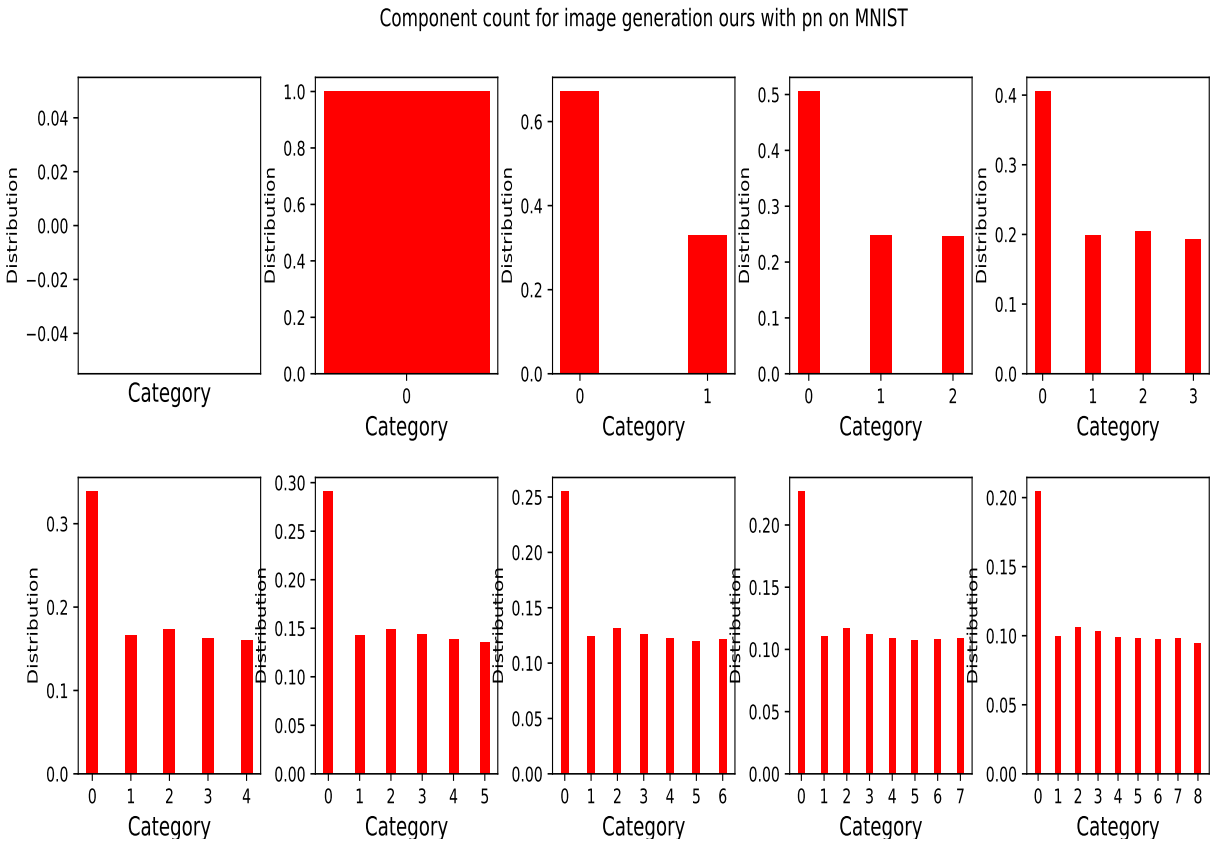


Figure 3.19: The distribution from which images are generated: each bar subplot correspond to the count of times a component is predicted as the best normalized by sum, and they are plotted in temporal order from left to right. Evolution of cumulative component count during training on the MNIST dataset for seq 1, which models the distribution of generated images (from which labels and images are generated) for our model "ours with p_n".

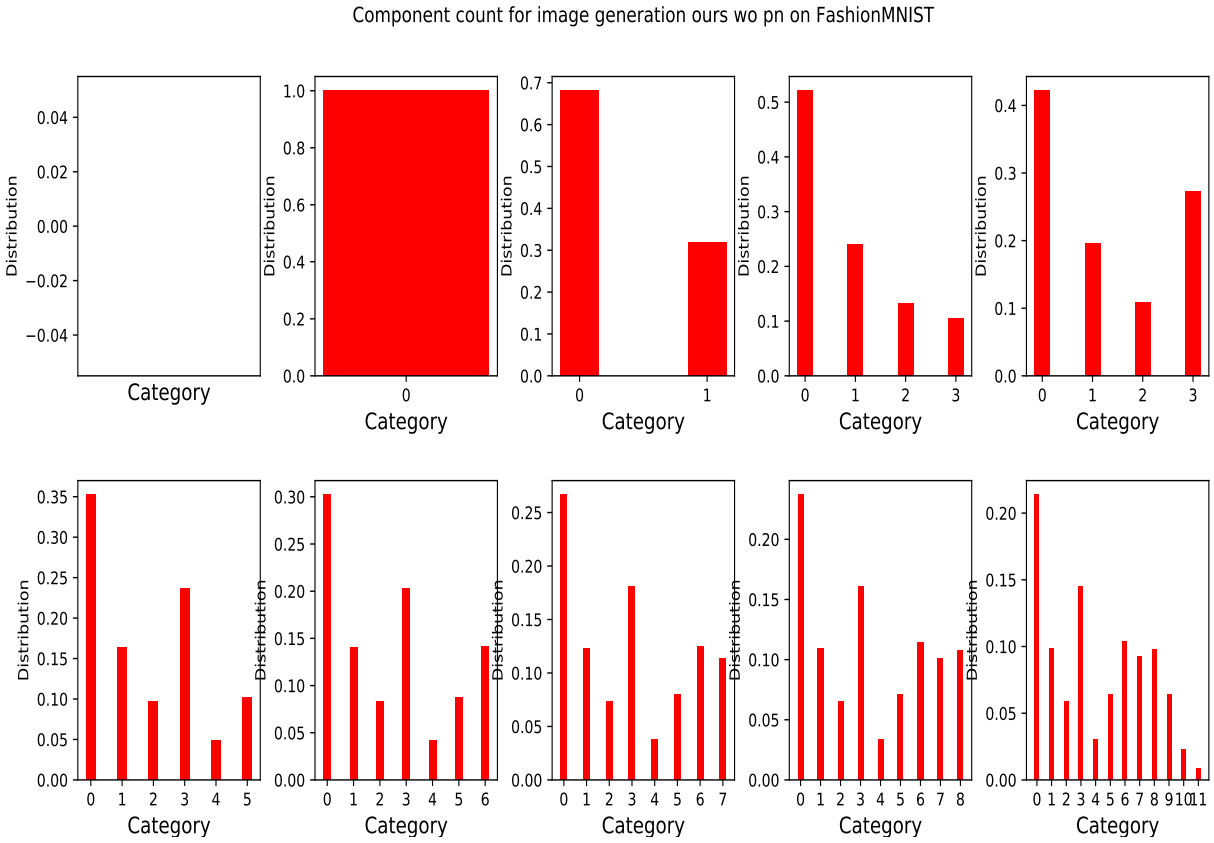


Figure 3.20: The distribution from which images are generated: each bar subplot correspond to the count of times a component is predicted as the best normalized by sum, and they are plotted in temporal order from left to right. Evolution of component count during training on the Fashion-MNIST dataset for seq 1, which models the distribution of generated images (from which labels and images are generated) for our model "ours w/o p_n"

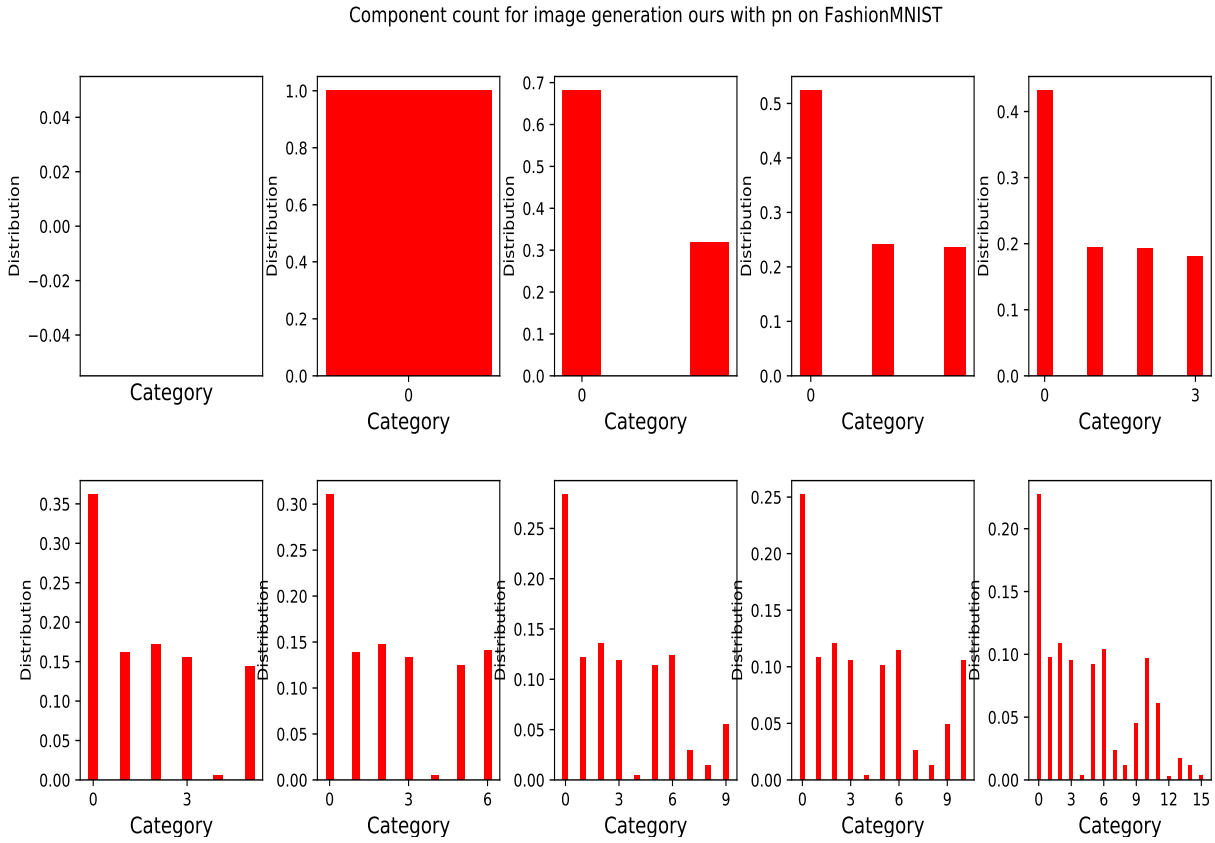


Figure 3.21: The distribution from which images are generated: each bar subplot correspond to the count of times a component is predicted as the best normalized by sum, and they are plotted in temporal order from left to right. Evolution of component count during training on the Fashion-MNIST dataset for seq 1, which models the distribution of generated images (from which labels and images are generated) for our model "ours with p_n".

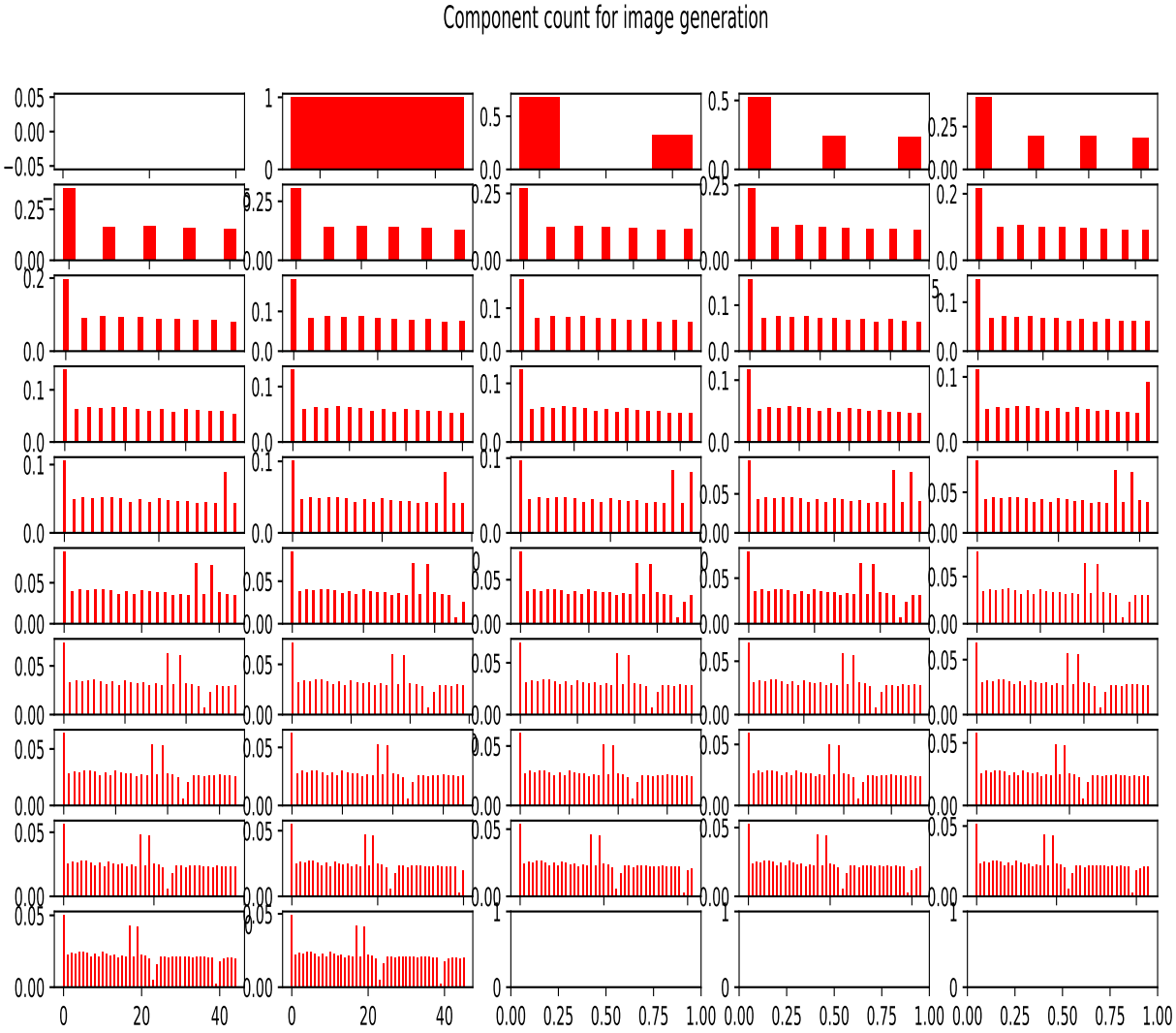


Figure 3.22: The distribution from which images are generated: each bar subplot correspond to the component count normalized by its sum, and they are plotted in a timely order from left to right. Evolution during training for component count on the EMNIST dataset that counts the number of times a component is predicted as the best for seq 1.

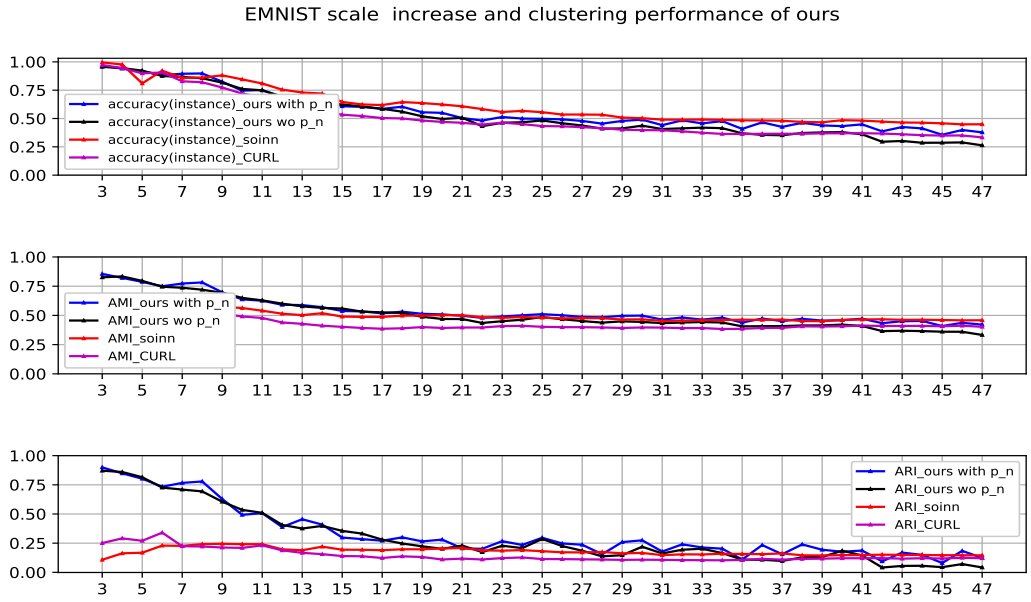


Figure 3.23: An illustration of the influence of the number of categories. For one training run and at each newly introduced class, the model is evaluated (on the test set) on all the categories it has learned.

3.3.10 Influence of the number of annotated examples during post-labelling

To calculate the classification accuracy, a post-labelling process is necessary during the evaluation to associate each cluster to the corresponding class. For each cluster, a majority vote using annotated examples is done. But as previously illustrated, CURL, our model and SOINN create different numbers of clusters for model prediction and thus may require different efforts in post-labelling. A model creating more components also requires more supervision to label these clusters to maintain comparable clustering performances.

The protocol to study the influence of the number of data, in this context, is to vary the number of annotated data n_l for post-labelling. To sample n_l examples, we randomly shuffle and choose n_l annotated examples (from the training set), and compute the majority vote among these n_l examples, to find the most represented classes for each cluster. We relabel different clusters amongst the model prediction over the test set during evaluation in this regard. This post-labelling process serves only to associate each cluster created by the model to a true category, while the evaluation itself by definition is always performed on the test set. Finally, we recalculate the classification accuracy.

One can observe that if we only use part of the train set to label components by their majority class, a drop in the clustering performance can be remarked from Fig. 3.24 on MNIST and from Fig. 3.25 on Fashion-MNIST. If more clusters were created and the model prediction is separated into these clusters, to associate each cluster with its most represented class implies at least one annotated example for these clusters. But if a cluster is associated with no categories, this impacts the clustering performance. The tendency can be seen as exponential $c_1 - c_2 exp^{(-c_3 x)}$ (with a time constant that is close to the number of clusters). The SOINN model converges the slowest compared to CURL and our model on MNIST since SOINN creates many clusters. The more there are clusters that over-segments the category distribution, the more annotated examples are required to reach the optimum performance with post-labelling, as one can remark from the Fig. 3.24 and Fig. 3.25. This shows that our model reduces the effort required for post-labelling, and thus improves the autonomy of the model, since it only needs a few annotated examples.

In previous experiments, we proposed a simple protocol for comparison between our model, CURL,

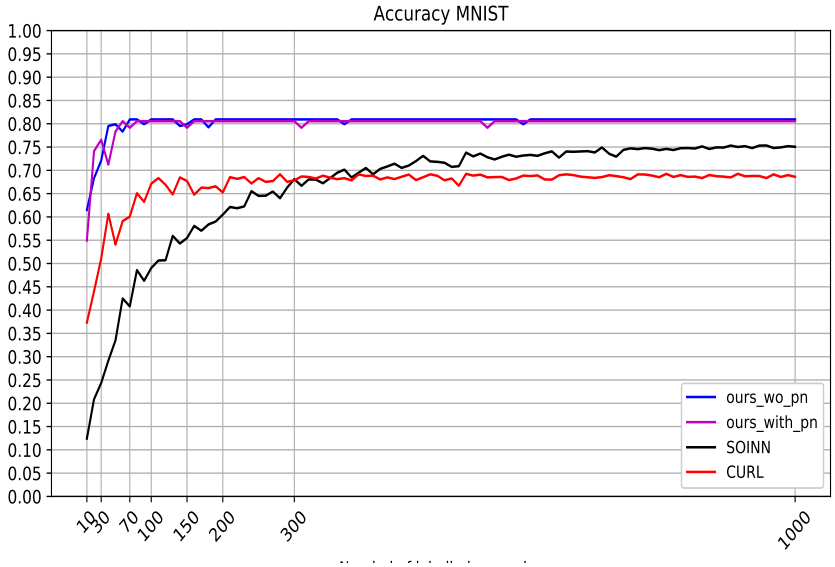


Figure 3.24: Influence of the number of annotated examples used for labelling the test set on the accuracy for MNIST. The x-axis is not uniform, since different models show different speed of convergence depending on the number of clusters.

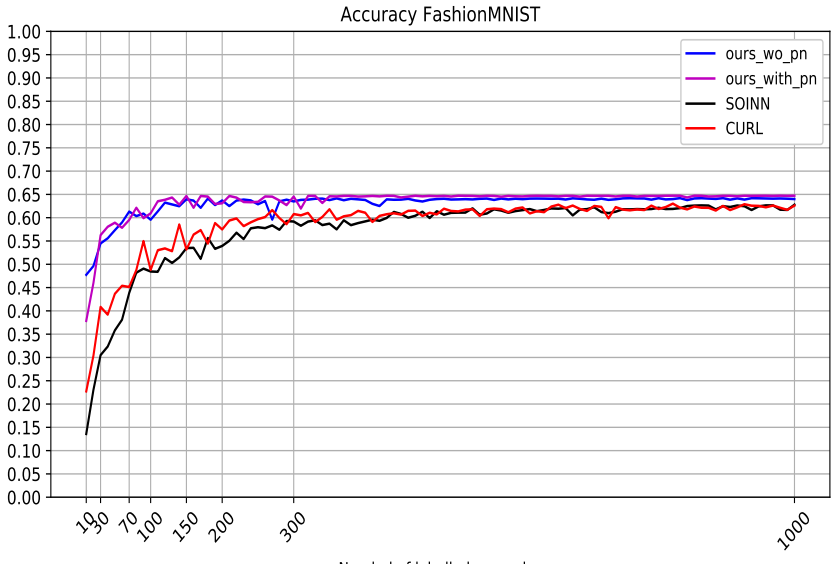


Figure 3.25: Influence of the number of annotated examples used for labelling the test set on the accuracy for Fashion-MNIST. The x-axis is not uniform, since different models show different speed of convergence depending on the number of clusters.

and the SOINN model. In our study, we investigated the relationship between the number of components created during learning. We found that with more components, the model tends to reach better classification accuracy, but that it leads to a reduction in AMI and ARI scores. Moreover, our model avoids over-segmentation of different clusters, while CURL creates several components that exceed the actual number of categories in the dataset. However, from previous test results, shown in section 3.3.6,

in figure 3.17, one can observe that certain similar but different categories tend to get confused by the model. For example, the model tends to mix "9", "4" for the MNIST dataset, and the category "2, 4, 6" for the Fashion-MNIST dataset. Therefore, in the next section, we propose a second experimental protocol that includes hard transitions created by these easily confused categories.

3.3.11 Hard transition sequences

In this section, we will investigate the influence of hard transitions, referring to successively presented categories of close appearance. Compared to the previous experimental protocol, this protocol can lead to more difficulties with novelty detection. We will first provide a detailed explanation of the experimental protocol and then compare our model with CURL on this protocol.

Protocol

Our objective is to demonstrate the novelty detection capacity and the robustness of the Page-Hinckley test applied in detecting transitions that are more difficult to detect, some of which can be statistically close to each other. To this end, we create hard transitions mixing similar categories and incorporate them into the sequence to form a protocol with more difficulty than the sequences introduced previously in the section 3.3.5. We determine the similarity between categories from similar categories in the composition of clusters we obtained in the previous experiment in figures 3.11-3.13 and in figures 3.14-3.17. The categories that are confused are considered similar. The most confusing categories on MNIST are '4', '9', and '7'. For the Fashion-MNIST dataset, categories '2', '4' and '6'; similarly for the categories "5" , "7" and "9". Those are presented one after the other to create hard transitions for novelty detection. Besides for other transitions, we also consider the similarity defined by human visual perception, for the rest of the categories among the entire dataset. We demonstrate whether the agent can maintain its autonomy when faced with hard transitions by using only internal labels deduced from the novelty detection and the internal representation of the agent.

We have tested 3 sequences listed in the table 3.8. The hard transition protocol affects both detections

seq	MNIST										Fashion-MNIST									
1	7	1	9	4	3	5	2	0	8	6	2	4	6	7	5	9	8	1	0	3
2	5	3	2	9	7	4	8	0	6	1	5	7	9	6	4	2	1	0	3	8
3	8	0	6	5	3	2	9	7	4	1	1	0	3	2	4	6	7	5	9	8

Table 3.8: Tested "hard" sequences.

from the point of view that similar categories are presented one after another, but also learning dynamics. Figure 3.26 shows an illustration of the variation of the ELBO loss during learning, for 1 run of each protocol, above for the simple transition protocol and below the hard transition protocol. Apart from the peaks formed by a change in the categories, one could also observe several other things concerning the dynamics of the evolution of ELBO loss. For the learning of each category, the slope of the curve while learning on a certain category shows the speed of models' optimization on the ELBO loss; while at the end of training of each category, if one compares the ELBO loss value, to the previous ELBO loss value, one could see if the optimization of the ELBO loss has recovered to its previous value or has it still needs to be optimized.

In particular, if we compare the dynamics of these two protocols, another interesting thing could be remarked: for the hard transition protocol, when we represent similar categories one after another, the level of ELBO loss at the end of training on the previous category is rapidly joined or surpassed by the following one. In fact, the ELBO loss shows both if the example is poorly modeled or if learning converges; and if the model has seen similar categories in adjacent positions and properly detected them. Within the hard transition protocol, the ELBO objective is optimized to the previous value shortly after the change of the category, and the optimization of the following category could further continue with learning. This could also be explained by the initialization of each component, after its creation, the wight

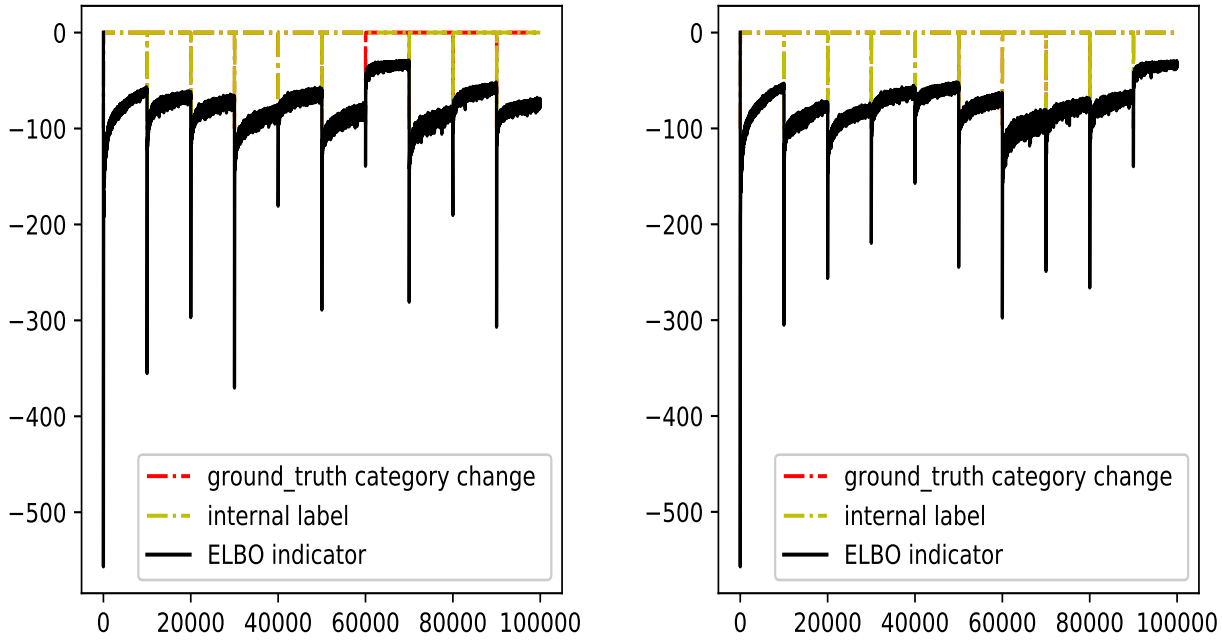


Figure 3.26: ELBO loss dynamics evolution in the simple transition protocol (left) for 1 run (seq 2) and in the hard transition protocol (right) for 1 run (seq 2).

of the component is copied from an existing one that shares the most similarity with it; and potentially this initialization could be part of the factors that impact convergence.

In figure 3.26, in the simple protocol, there is one category change that is not properly detected ("marked in red"), meanwhile, before the error in detection occurs, the learning of the previous category at its endpoint falls equally lower than the ELBO loss at the end of training of other past categories. Meanwhile, in this example of a hard transition protocol, all the changes in categories are properly detected. And in this example, the transition between "5, 3, 2, 9, 7, 4" and "8, 0, 6, 1" show a dynamic of the ELBO loss that learning is harder to recover to the previous level at the end of training of "4", since among the classes that are presented at the end of the training, they share little similarities with the previously learned ones.

Results

We compare our models "ours w/o p_n", "ours with p_n" and CURL on these sequences with hard transitions, to show their clustering and classification performance. The results on the MNIST and Fashion-MNIST datasets are shown respectively in tables 3.9 and 3.10.

model	AMI	ARI	# components	accuracy	homogeneity
CURL	0.58 ± 0.02	0.39 ± 0.03	20.33 ± 3.68	0.68 ± 0.06	0.64 ± 0.05
SOINN with SIFT	0.55 ± 0.02	0.21 ± 0.02	91.67 ± 6.8	$\mathbf{0.84 \pm 0.04}$	$\mathbf{0.78 \pm 0.04}$
Ours w/o p_n	$\mathbf{0.746 \pm 0.02}$	$\mathbf{0.716 \pm 0.05}$	$\mathbf{9.67 \pm 0.47}$	$\mathbf{0.848 \pm 0.04}$	0.737 ± 0.03
Ours with p_n	$\mathbf{0.752 \pm 0.018}$	$\mathbf{0.722 \pm 0.048}$	$\mathbf{9.67 \pm 0.47}$	$\mathbf{0.85 \pm 0.04}$	0.74 ± 0.03

Table 3.9: Comparison with the state of the art on MNIST (averaged over 3 runs) with hard transitions (mean±SD).

In the hard scenario, one can remark that our model is capable of maintaining clustering model that

model	AMI	ARI	# components	accuracy	Homogeneity
CURL	0.49 ± 0.01	0.32 ± 0.02	34 ± 3.74	0.62 ± 0.01	$\mathbf{0.58 \pm 0.01}$
SOINN with SIFT	0.45 ± 0.0017	0.18 ± 0.009	64.67 ± 9.98	$\mathbf{0.64 \pm 0.022}$	$\mathbf{0.6 \pm 0.02}$
Ours w/o p_n	$\mathbf{0.553 \pm 0.03}$	$\mathbf{0.397 \pm 0.035}$	$\mathbf{14.6 \pm 3.4}$	0.617 ± 0.02	0.527 ± 0.014
Ours with p_n	$\mathbf{0.556 \pm 0.012}$	$\mathbf{0.399 \pm 0.039}$	15.3 ± 3.68	$\mathbf{0.635 \pm 0.025}$	0.54 ± 0.009

Table 3.10: Comparison with the state of the art on Fashion-MNIST (averaged over 3 runs) with hard transitions (mean±SD).

detects the number of clusters close to the number of categories and outperforms the other methods on most other metrics (AMI, ARI and accuracy). Hard transitions do not drop drastically the cluster performance for our model, except for a slight decrease in the Fashion-MNIST dataset. In both scenarios, the performance of "ours w/o p_n" and "ours with p_n" were close in terms of AMI and ARI score, but "ours with p_n achieves a slightly higher accuracy. When the order of presence changes in sequences, the number of generated images differs depending on whether the class is presented at the beginning or at the end. Since the model alternates between real batches and generated batches which are distributed over all past learned categories. At the beginning of the learning, there are only few learned categories. For this reason, "9" and "7" can be difficult to separate in a sequence (i.e. {0,1...9}) introducing them at the end of the sequence, but gets better clustering performances when "7", "1" and "9" are presented at the beginning. At the end of the learning sequence, while training real "9" for example, the generated examples are separated onto "0" to "8", making fewer generated images for "7"; but on the contrary, at the beginning of the learning sequence, when the model sees "9", the generated images for the past categories consist only of "7" and "1", making more generated examples than in the previous case.

We illustrate the composition of clusters for CURL and our model for MNIST in figure 3.27- 3.28 and for Fashion-MNIST in figure 3.29- 3.30, taking the example of the first sequence. In this illustration, the components are arranged in their creation order. It is not averaged over different runs, because the total number of components tends to vary. Similar to the previous section, it demonstrates the partition of categories into different categories if one observes horizontally; and the composition of a cluster if one observes vertically. On the MNIST dataset, CURL tends to confuse the categories '4', '7' and '9' that fall simultaneously into component 0, but our model allows to separate them. But on the Fashion-MNIST dataset, the confusion of '2', '4' and '6' persists for all the models. Additionally, compared to previous experiment, the last component is less homogeneous since it mixes categories '0', '3' and '6'. This confusion is due to a decrease in the number of generated examples for components created at the end of the training sequence compared to those created at the beginning. For training of 3, the model is presented with batches of real examples and generated examples, distributed over all the past learning categories.

3.4 Conclusion and perspectives

In this chapter, we have targeted the problem of unsupervised continual learning for object recognition. Continual learning is a difficult problem due to challenges such as catastrophic forgetting and evolving learned representations while incrementally learning new concepts. Many existing continual learning approaches partially alleviate the problem of catastrophic forgetting, but remain supervised. Among unsupervised continual learning approaches, the automatic detection of the number of clusters questions the autonomy of the model, as unsupervised continual learning approaches tend to over-segment clusters when they detect and create clustering that exceed the true number of clustering. The over-segmentation requires extra supervision during post-labelling. From this point of view, unsupervised continuous learning is closely related to the notion of novelty detection.

Novelty detection for non-stationary data is essential, since an autonomous agent must detect changes in its input while dynamically adapting to possible variations in its environment. We have previously reviewed several state-of-the-art change detection approaches in the literature in the section 2.5.3. Among these approaches, the Page-Hinckley test allows detecting abrupt changes.

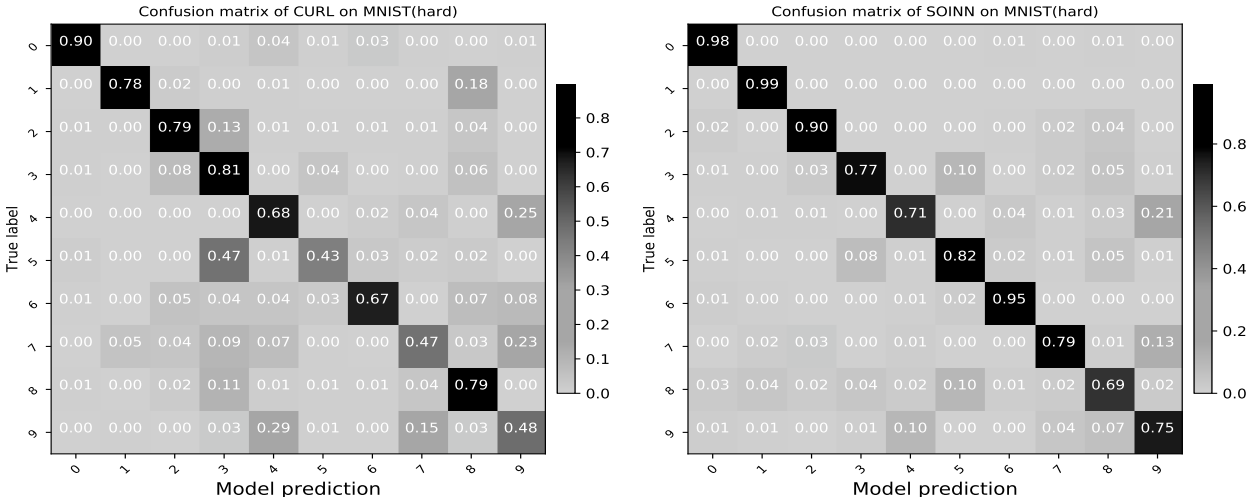


Figure 3.27: Confusion matrix averaged over 3 runs and composition of clusters between ground truth and predicted cluster components on MNIST and with the hard transition protocol (seq. 1) for CURL (left), SOINN (right)

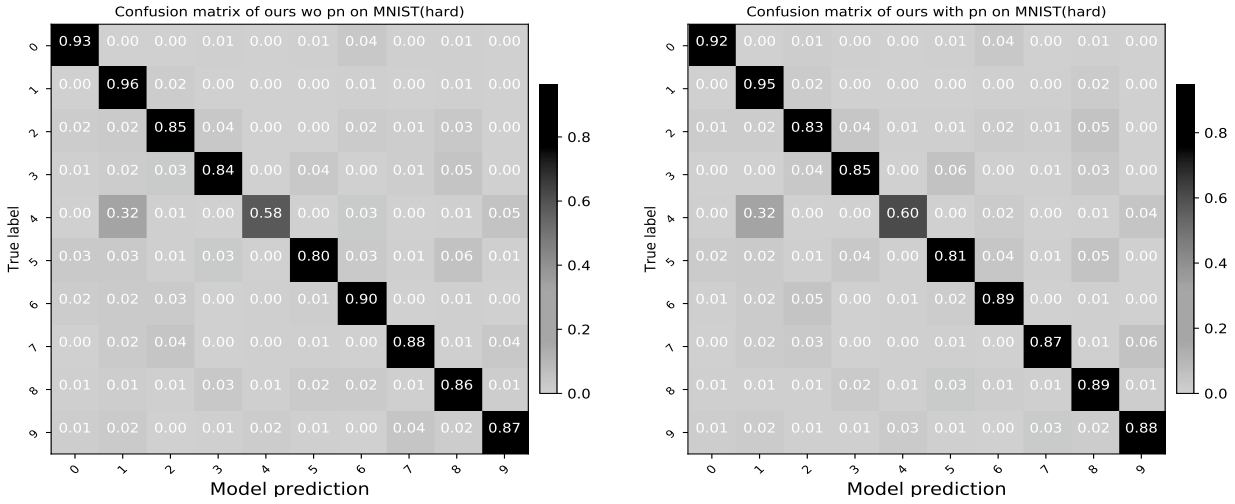


Figure 3.28: Confusion matrix averaged over 3 runs and composition of clusters between ground truth and predicted cluster components on MNIST and with the hard transition protocol (seq. 1) "ours w/o p_n" (left) and "ours with p_n" (right) (the darker the cell the more instances it represents).

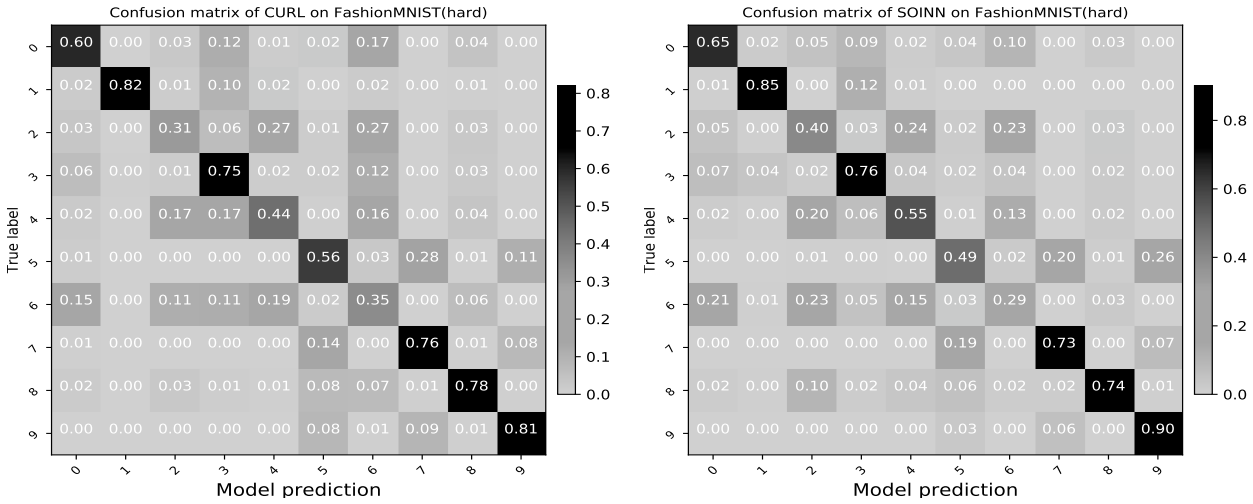


Figure 3.29: Confusion matrix averaged over 3 runs and composition of clusters between ground truth and predicted cluster components on Fashion-MNIST and with the hard transition protocol (seq. 1) for CURL (left), SOINN (right)

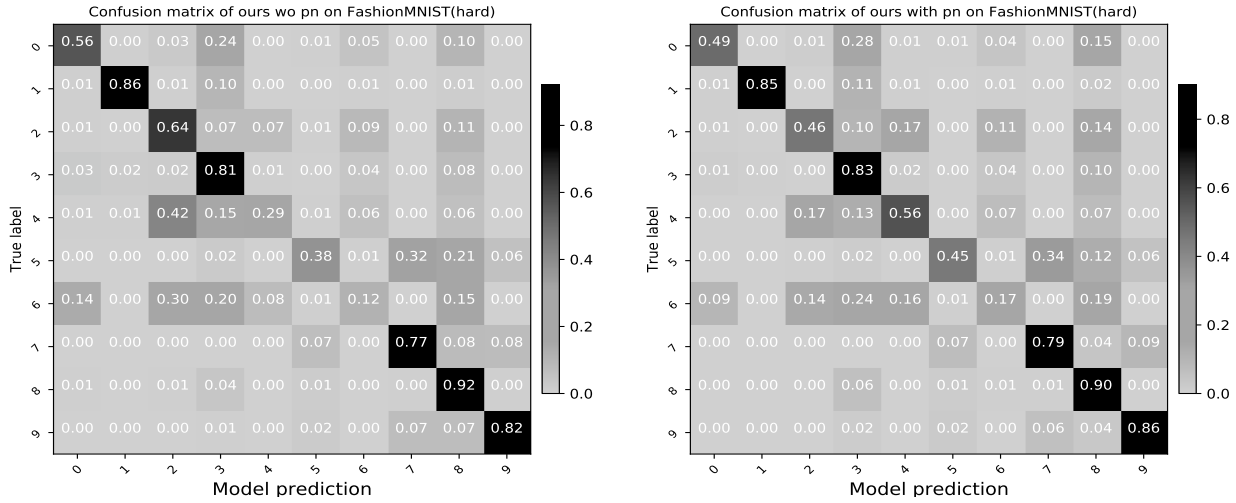


Figure 3.30: Confusion matrix averaged over 3 runs and composition of clusters between ground truth and predicted cluster components on Fashion-MNIST and with the hard transition protocol (seq. 1) for "ours w/o p_n" (left) and "ours with p_n" (right) (the darker the cell the more instances it represents).

As our first contribution for novelty detection, we use the Page-Hinckley test to detect category changes. In this model, we have considered the hypothesis of temporal consistency where objects are presented in sequential order, class by class. Meanwhile, as a first step, we consider a simplification where previously learnt object classes do not reappear later during training. The Page-Hinckley test is applied to ELBO loss, which was compared with a threshold resulting in a Heaviside step function as an indicator of novelty, with possible smoothing strategies like running average to reduce the impact of possible fluctuations in the indicator. With the application of Page-Hinckley test on this indicator, our model can detect abrupt changes that correspond to a new category.

In our proposed model, the novelty detection helps the learning by eliminating false positives that would lead to excessive component creation. As an unsupervised continual learning approach, CURL also detects changes, yet through the application of a simple thresholding approach. During clustering, this results in false positive change detection and over-segmentation. The self-supervision based on Page-Hinckley test helps to create a clustering that is close to the real distribution of different classes and avoids dividing the same category into subcategories.

Our model is evaluated on the MNIST and Fashion-MNIST dataset in the class-incremental scenario, for which we have tested two protocols. The first protocol is a baseline to compare different models and variants of the model. We have chosen to optimize different models regarding both the ARI (or the number of created components) and the classification accuracy. With a larger number of components in clustering, a higher classification accuracy can be obtained when combined with majority vote, due to subdivisions of clusters that result in clusters with more purity. Our approach improves autonomy through eliminating excessive components and to avoid over-segmentation, which requires less supervision during relabelling since the number of clusters is close to the number of categories.

We have also compared with another scenario, called the hard protocol, by creating hard transitions that are formed by similar categories that arrive one after the other. The performance of clustering of our approach has not seen dramatic decrease on the hard protocol. However, in addition, we have also remarked that changing the order of presence in the sequence, especially in the hard protocol, has slightly increased the number of generated examples that are easily confused but presented at the beginning of the sequence. This is a possible element that can impact the learning performance.

Regarding the scenarios that an autonomous agent can face during object recognition, we will extend the model in a way that objects that reappear later can be identified by the agent, so that objects can be seen multiple times. This is a more realistic but also more challenging scenario as the agent not only needs to detect novelty but also needs to be deployed in a way that it is capable to recognize learnt objects without repeatedly reallocating unnecessary new resources. This scenario with more complexity is explored and will be presented more in detail in the following chapter. And regarding these use cases, we will represent our second contribution.

Chapter 4

Continual object representation learning with novelty detection and recognition

4.1 Introduction

Previously in chapter 3, we proposed a model that is an extension of Curl, integrating the Page-Hinckley test for novelty detection to guide training. This model is limited to a scenario where each class is seen only once. As an autonomous agent may come across the same class several times, it needs to not only be able to detect new classes, but also recognize previously learned ones. Compared to the previous problem setting in chapter 3, the review of categories actually raises more challenges: without the review of categories, the model only needs to detect unseen categories. The detection of abrupt changes of Page-Hinckley will partially suit this use new case, but with the review of categories, the model will need to consider novelty detection in a way that optimizes the trade-off between detecting new categories and recognizing those already learned. This means to avoid detecting all category changes as new, regardless of whether they are learned or unknown, while also clustering learned classes online.

Indeed, the scenario with the review of objects is more complex compared to the previous scenario explained in chapter 3, the difference lies in the recognition of objects online. To learn semantic object representation and to estimate statistics of latent variable distributions properly and dynamically, which are close to the distributions of different categories, the model is required to learn semantic object representation.

Moreover, the problem shares the same constraints such as catastrophic forgetting as in common continual learning scenarios, as it is explained in chapter 2 in section 2.3.2.3. During the learning process, the model's representation of a class that has already been learned tends to shift in embedding space due to the introduction of new classes and optimizations. This results in the forgetting or failure to recognize the class. Likewise, the model should not store infinitely past training examples in a continual learning scenario. Therefore, maintaining the autonomy of the model in this online scenario can be more delicate, as the model needs to detect the number of clusters by recognizing learned categories and rejecting unknown ones to create new classes. In the continual learning literature is introduced in chapter 2, Curl, as a baseline, considers the review of objects implicitly by approximating the indicator of novelty with the loss function during training. However, it often produces excessive clusters due to the lack of automatic determination of the number of categories and the over-segmentation of different clusters, creating false positives and requiring extra effort for post-labelling. Moreover, the model should reject an unknown category that it has not learned, referred to as novelty detection problems while staying within the context of continuous learning. It is often crucial to estimate the probability to be an unknown category. As outlined in chapter 2, existing approaches are often poorly calibrated due to the closed world assumption, which is an estimation of probability considering only known categories.

In this chapter, we will first show the drawbacks of classical confidence metrics, in section 4.2.3 the maximum activation of posterior estimation of VAE and entropy-based metrics in novelty detection and object recognition. We will introduce a statistical hypothesis test, the Hotelling t-squared test, in section 4.2 which presents promising results in the preliminary test and its integration in our model in section 4.3.1. The combination of our model and the Hotelling t-squared test equally enables self-supervision of the learning process, as described in section 4.3.2 - the model can either determine the identity of a category or can create a new class to adapt to the change in the input distribution change that corresponds to a new class.

4.2 Challenges in new-class detection

4.2.1 Problem definition

To detect new classes is essential to implement self-supervision, as we have previously seen in chapter 3. From a pure novelty detection aspect, one can mathematically formulate the novelty D_n determined by the model using a confidence measure $M(x)$. For example, if the confidence (or certainty) related to an input x is smaller than a threshold θ_M, it is rejected as unknown and D_n is 1, otherwise the example is considered from a known class and D_n is 0:

$$D_n = \begin{cases} 0, & \text{if } M(x) \leq \theta_M \\ 1, & \text{otherwise.} \end{cases} \tag{4.1}$$

This is the problem we addressed in the previous chapter. But here, our targeted task is less constrained and more complex. An autonomous agent may perceive data streams that are less structured. Not only do we need to detect novelty but also to recognize the learned categories. This problem is referred to as "open set recognition" in the literature, as previously introduced in section 2.5.2 in chapter 2. That means, if one needs to further classify learned classes and detect unknown classes, for a given instance, the model will predict its category y_p if it is a known one.

Let us take the case of our model and the case of CURL (to give a more intuitive example). In CURL, the VAE models different categories with a Gaussian mixture. For the category variable denoted y and the input denoted x, the model predicts the category for an input with specific output dense layer after the encoder applying the softmax function $q(y|x)$ (see chapter 2 section 2.4 for more details). Then $argmax(q(y|x))$ (with $q(y|x)$ estimated by the VAE) predicts the category of an input, and the input is rejected if novelty is detected. The determination of D_n in Eq. 4.1 therefore becomes Eq. 4.2 to predict y_p which includes the recognition:

$$y_p = \begin{cases} \text{argmax}(q(y|x)), & \text{if } M(x) \geq \theta_M \\ \textit{new class label}, & \text{otherwise.} \end{cases} \tag{4.2}$$

An essential part of the problem in both novelty detection and open set recognition is to define a metric $M(x)$ or a reliable approach that penalizes outliers or new objects and allows their rejection.

4.2.2 Illustration of challenges through an example

Novelty detection and recognition are challenging when considered together, especially in the continual scenario. In the literature, as mentioned in 2.5.2, some researchers have pointed out the issue that for examples of unknown classes many existing models will predict a known class with high confidence. This is because most methods are designed under a closed-world assumption. Hence, the model decides whether to accept or reject the estimation based on the known categories while estimating their probability. But this estimation is often poorly calibrated, which highly affects the scores. Unknown classes are sometimes hard to separate and are sometimes confused with the already known ones. Especially in classification for example, where the softmax function is applied regarding existing categories, and unknown categories are thus poorly calibrated. This also concerns CURL, i.e. our base model, where the prediction of $q(y|x)$ also uses a softmax layer.

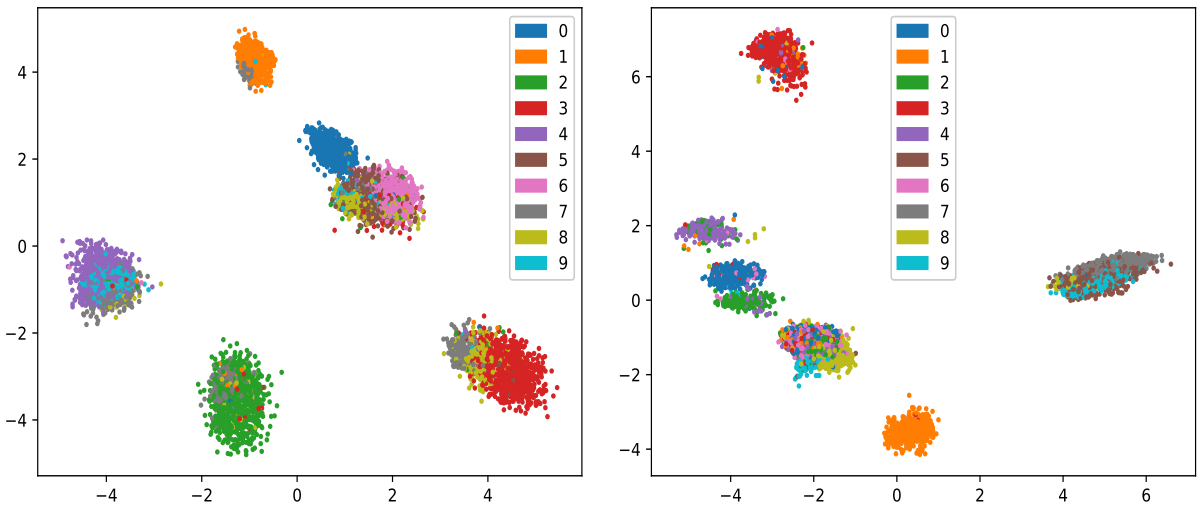


Figure 4.1: PCA-projected latent variables for our model on learned (0-6) and unknown categories (7-9): MNIST (left) and Fashion-MNIST (right) with normalization as preprocessing. Latent variables are normalized with regard to mean and standard deviation of the latent variable on the training set. One can observe that examples of unknown categories, on MNIST and Fashion-MNIST are projected onto that of learned ones.

As an illustration, we provide a simple example of the novelty detection performance of this model by showing the results on the MNIST and Fashion-MNIST datasets for rejecting unknown categories. Considering training the model only on the part of the classes that existed in the dataset and evaluating the entire dataset, the classes used for training will be considered as known, with the rest considered as unknown. Here, we train on categories numbered 0-6 and evaluate all the categories from 0-9 in order to test different confidence metrics on both datasets. Our model, as introduced in the previous chapter 3 in section 3.2, uses the ground truth changes of categories as supervision, which is normally self-supervised by a model determining the internal label y_m in Eq. 3.19. This supervision is only applied in this simple case to demonstrate the challenge of open set recognition. The use of ground truth change here is to discard the influence of the Page-Hinckley test that we have integrated as the novelty detection process since our objective is to study the distribution of latent variables projected into 2D space for unknown categories. We illustrate the PCA projections of 10 classes on the test set on both MNIST and Fashion-MNIST to show if the model can separate learned categories 0-6 from unknown categories 7-10. From figure 4.1, it can be remarked that the projection of unknown categories 7-9 are mixed with learned categories 0-6. Typically, on the MNIST dataset, the class '9' is projected on the class '4'. On the Fashion-MNIST dataset, the category '9' and category '7' are once again projected on the category '5'. While this confusion can be explained by the similarities between classes, it also shows the limitations of novelty detection in this scenario, as the examples of unknown categories are projected onto that of learning objects.

The problem of novelty detection or open-set recognition urges us to search for an effective confidence measure $M(x)$. In the following section, we will show some of the common metrics applied as confidence measure $M(x)$ and compare their performance in a preliminary test.

4.2.3 Preliminary tests for different metrics

We will study here experimentally if new classes can be detected for a given input example by applying a threshold θ_M on common confidence metrics $M(x)$, as shown in both Eq. 4.1 and Eq. 4.2 in section 4.2.1.

To compare the performances of different metrics as a preliminary study, we continue to use the approach explained in the section 4.2. In this case, we show the preliminary results of several common

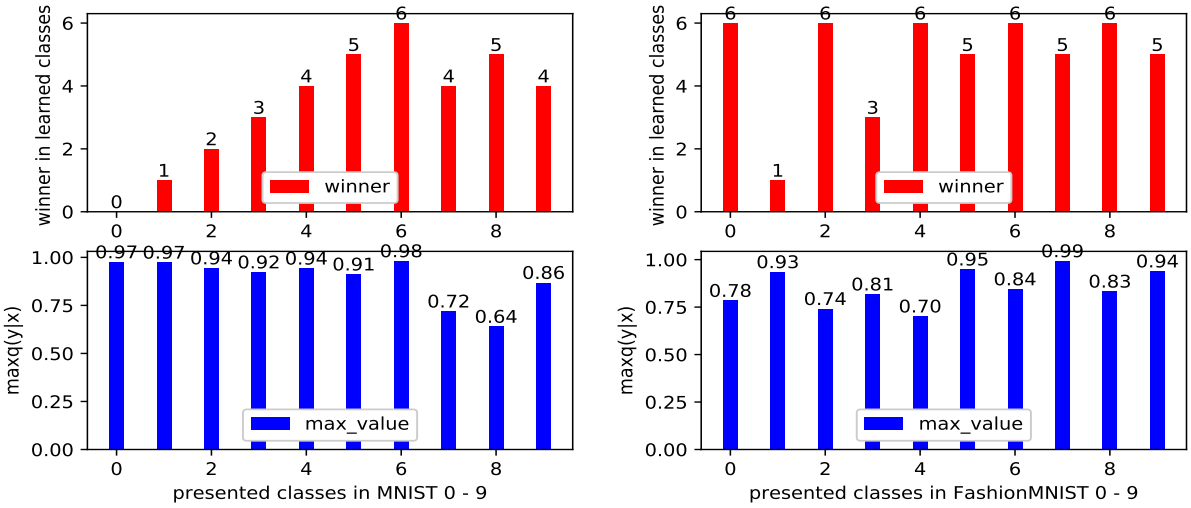


Figure 4.2: The average values of $M(x) = \max(q(y|x))$ (bottom) on the test set of MNIST (left) and Fashion-MNIST (right), and the majority class (marked as "winner") (top).

metrics on the MNIST and Fashion-MNIST datasets. We train our model as explained in chapter 3 while replacing the self-supervision internal label y_m with the ground truth changes to isolate the influence of the proposal of the Page-Hinckley test and to demonstrate the performance of different metrics. Our model is trained only on a limited number of classes. We take the simple example of training on classes 0 to 6 and evaluate the entire test set containing examples from classes 0 to 9. Thus, in this example, classes 0 to 6 are considered known classes, while classes 7 to 9 are considered unknown classes.

4.2.3.1 Metric 1: Maximum activation

First, we study the performance of the metric based on $q(y|x)$ as a baseline. We use the maximum activation of $q(y|x)$ for the probability of belonging to the most probable cluster, its equation is shown in Eq. 4.3.

$$\begin{aligned} M(x) &= \max(q(y|x)) \\ q(y|x) = \text{softmax}(A(x)) &= \frac{e^{-A(x)}}{\sum_1^K e^{-A(x)}} \ , \end{aligned} \tag{4.3}$$

where K is the number of learned categories. Also, the number of activated components as the model is trained on the ground truth category changes in this scenario. $A(x)$ the activation *before* the softmax function and $q(y|x)$ the activation or cluster prediction of the VAE *after* the softmax function. As a common disadvantage, the softmax function normalizes the output of the VAE for a prediction between 0 and 1 for all the K existing categories, but, in our scenario, it also projects the rest of the $(10 - K)$ categories onto learned K categories, making the unknown categories poorly calibrated. In figure 4.2, we illustrate the mean for each class for the proposed metric of Eq. 4.3, the maximum activation of $q(y|x)$. It can be remarked that on the Fashion-MNIST dataset, the metric gives relatively poor performance, since it mixes known with unknown categories. Another problem can arise from intra-class similarities, especially for the Fashion-MNIST dataset in which visual similarities exist between different categories of clothes.

4.2.3.2 Metric 2: Entropy

Another common approach in the literature is entropy, based on the information theory, as we have previously introduced in the section 2.5 in chapter 2. Here, we study a second metric defined on the

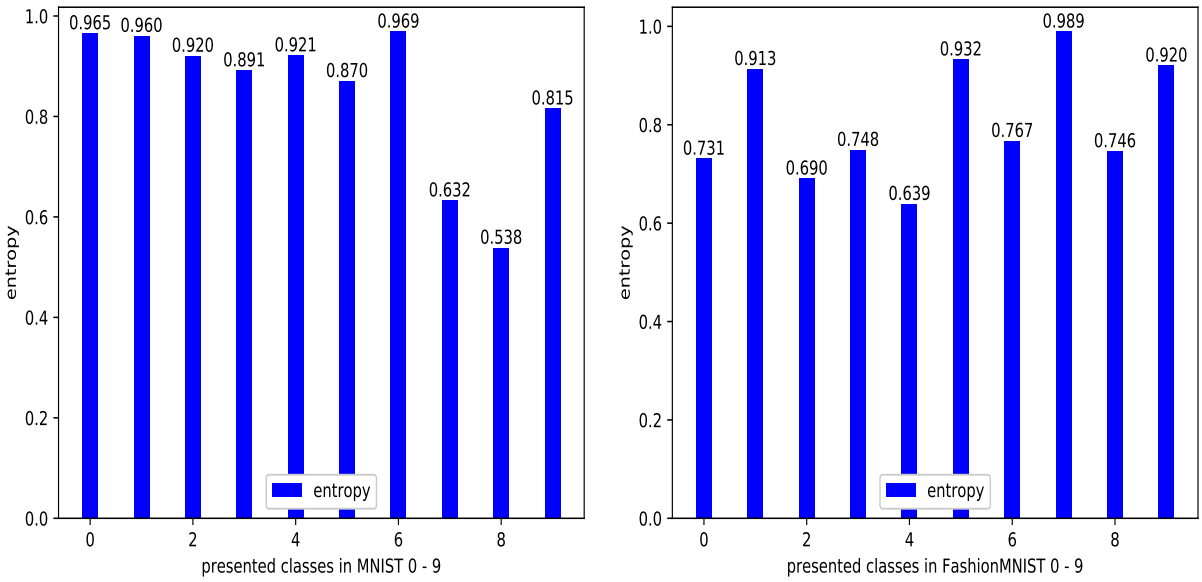


Figure 4.3: The average values for the entropy-based confidence metric (Eq. 4.4) on the test sets of MNIST left) and Fashion-MNIST (right).

entropy of $q(y|x)$ over all classes K. Intuitively, this measures the uncertainty, where the prediction of a single class with probability 1.0 has the lowest entropy (highest confidence) and a uniform distribution over all classes has the highest entropy (lowest confidence). In the latter case, an example is poorly modeled and therefore a possible new class candidate. In Eq. 4.4, as in the previous metric, y is the category variable and x the input of the model, n_a represents the number of components created by the model.

$$\begin{aligned} E_{max} &= -\sum_{k=1}^{K} 1/K * log(1/K)) = -log(1/K)) \\ Entropy &= -\sum_{k=1}^{K} q(y=k|x) log(q(y=k|x)) \\ M(x) &= 1 - Entropy/E_{max} \end{aligned} \tag{4.4}$$

We use this metric to measure the entropy of output prediction of the model $q(y|x)$, for a well-modeled example, the ideal case is that $q(y|x)$ has a high score by activating only one single component representing a single category. But for a poorly modeled example, the $q(y|x)$ will be dispersed while activating different categories with low scores. The worst case, although it rarely occurs in practice, $q(y|x)$ is uniformly distributed on different components of $q(y|x)$, with K the number of learned categories, As the model here is trained with ground truth class changes, K will also be the number of activated components and $1/K$ their probability. In order to bound $M(x)$ and be able to easily fix a threshold, we normalize by the corresponding entropy E_{max}.

In figure 4.3, we show the results of the mean metric value for different classes. As introduced in section 4.2.3, the classes 0-6 are learned while 7-9 are unknown.

In fact, the entropy in Eq. 4.4 is a metric that is deduced from $q(y|x)$. This explains the reason for its performance in certain unknown classes. For example, $q(y|x)$ activates an unknown category with a high score, indicating $q(y|x)$ is little dispersed in other categories since $1 - max(q(y|x))$ is low, this eventually leads to a false high confidence with the entropy-based metric value. One can remark that certain unknown categories are getting high scores compared to learned categories–their difference is not significant enough to easily find a threshold method to reject unknown categories. For example, in the

case of class '9' on MNIST, voted as class '4' with a high metric value. For the Fashion-MNIST, class '9' is getting a high score and is not separable from the class '5'. This can partially be explained by the calibration of the softmax function. When the model has learned categories 0-6, the softmax function will be computed with regard to 7 learned classes instead of 10 classes, therefore the calibration is poor for unknown classes. This illustrates the limitation of common novelty detection metrics based on a simple threshold mechanism.

Eventually, the robustness of both Metric 1 in section 4.2.3.1 and Metric 2 in this section is limited by the capacity of the model prediction $q(y|x)$. For this reason, we looked at other metrics providing more robustness.

4.2.3.3 Metric 3: Entropy-based on the histogram of the class

As we pointed out in the previous section, the instance-based prediction $q(y|x)$ of the model is not robust enough to provide a confidence metric for unknown category detection, as some unknown categories are heavily confused with the learned ones. The limitation of the model's posterior estimation based on the $q(y|x)$ indicator has motivated us to search for a different approach. An idea to alleviate this problem is to use a metric based on a batch of instances so that it can reduce noises brought by the $q(y|x)$ prediction error on some instances. We therefore computed the entropy on the statistics over a given batch to replace the instance-based prediction of $q(y|x)$. The purpose is to have a stronger estimation of statistics.

Concretely, for a batch of examples of a category, we propose a metric based on the $q(y|x)$ over the entire batch, with b the batch size and n_i the number of examples predicted as class i. Eq. 4.5 shows the definition of this metric. It corresponds to the entropy over the batch histogram of model predictions over the class IDs.

$$\begin{aligned} E_{max} &= -log(1/K)) \\ Entropy_{bch} &= -\sum_{i}^{K} \frac{n_i}{b} * log(\frac{n_i}{b}) \\ M(x) &= 1 - Entropy_{bch}/E_{max} \end{aligned} \tag{4.5}$$

In Eq. 4.5, as in previous metrics, K refers to the number of learned categories and also the number of activated components. And for a batch of examples of the same category, we count over the batch model predictions and compute the histogram of model prediction for each possible learned category. We use the n_i to refer to the number of instances inside the batch predicted as a certain category i, and b refers to the batch size. In summary, for this metric, the entropy is computed on the histogram of the model predictions averaged over a batch. The results for MNIST and Fashion-MNIST are shown in figure 4.4. Again, similar problems can be observed as with the other metrics. As previous preliminary works showed, the likelihood of unknown categories with deep neural networks may be very high, this is also true for our VAE model. When studying the distribution of latent vectors z, there are large overlaps between known and unknown classes, illustrated in figure 4.1.

Instead of using metrics based on model posterior estimation $q(y|x)$, we will investigate an approach based on the statistics of the latent variable z learned by the Variational Autoencoder. As discussed in chapter 3, we suppose that the distribution of latent variable or each category follows a (single component) Gaussian distribution. This permits us to use a statistical hypothesis test, the Hotelling t-squared test, which permits both novelty detection and the recognition of learned categories. It is usually applied on a small set of continuous samples for hypothesis testing, i.e. a batch, from the data stream.

4.2.4 The Hotelling t-squared test

The Hotelling t-squared test [72] is a statistical test for multivariate normal distributions. The two-sample Hotelling t-squared test determines if two sets of samples correspond to the same distribution according to the empirical means and covariance matrices of the two sets. Formally, let $z_y = \{z_{1y}, \ldots, z_{n_y}\}$ be samples drawn from a multivariate normal distribution $\mathcal{Z}_y \sim N(\mu_y, \Sigma_y)$. Let $z_b = \{z_{1b}, \ldots, z_{n_b}\}$ be samples drawn independently of another multivariate normal distribution, $\mathcal{Z}_b \sim N(\mu_b, \Sigma_b)$. The Hotelling

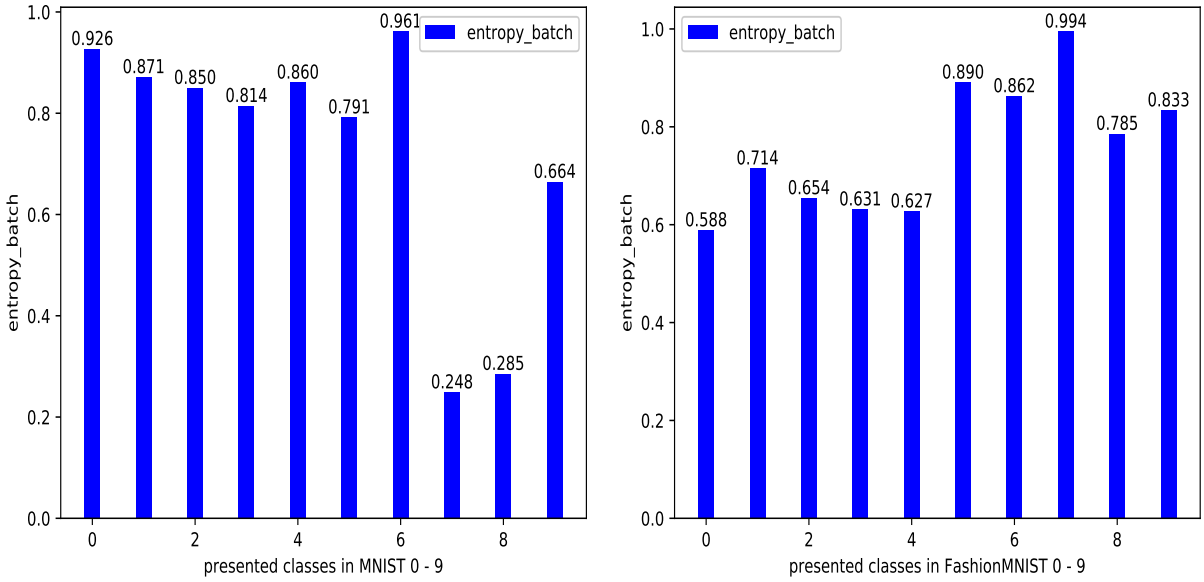


Figure 4.4: Batch-averaged entropy computed with the histogram of batch prediction for each class (Eq. 4.5) for MNIST (left) and Fashion-MNIST (right).

t-squared test determines if the distribution of $\mathcal{Z}_y$ and $\mathcal{Z}_b$ are the same. The sizes of two samples are respectively n_y and n_b, $\bar{z}_y$ the mean of samples z_y, and $\bar{z}_b$ the mean of samples z_b, then t^2 statistics of the test can be computed as:

$$t^2 = \frac{n_y * n_b}{n_y + n_b}(\bar{z}_y - \bar{z}_b)^T \widehat{\Sigma}^{-1}(\bar{z}_y - \bar{z}_b) \,, \tag{4.6}$$

with $\widehat{\Sigma}$ being the pooled covariance matrix determined by

$$\widehat{\Sigma} = \frac{(n_y - 1)\widehat{\Sigma}_y + (n_b - 1)\widehat{\Sigma}_b}{n_y + n_b - 2} \,, \tag{4.7}$$

and $\widehat{\Sigma}_y$ the covariance matrix of z_y, and $\widehat{\Sigma}_b$ the covariance matrix of samples z_b. Intuitively, the t^2 statistics can be assimilated to a distance. Besides, if $\widehat{\Sigma}$ was not pooled equally over the covariance of the samples, it would be the Mahalanobis distance, and if, in addition, $\widehat{\Sigma}$ was a diagonal matrix it would be a normalized euclidean distance. Here, the $\widehat{\Sigma}$ is pooled to consider the sample sizes of both the statistics. The larger t^2 statistics, the smaller is the probability that the two sample sets correspond to the same distribution. The t^2 distribution follows the F distribution up to a factor, where d is the dimension of vectors in samples:

$$\frac{n_y + n_b - d - 1}{(n_y + n_b - 2)d} t^2 \sim F(d, n_y + n_b - 1 - d) \tag{4.8}$$

Note that the Hotelling t-squared test assumes multivariate normal distributions for both samples in the test. Taking the application in our learning scenario as an example, the VAE learns the latent variables z for different categories. We suppose that the latent variable z learned by the VAE for one category can be modeled by a *single* (Gaussian) component as previously mentioned in section 4.3.1. In this case, in our model, the z_y and z_b will respectively be latent vectors resulting from classes y and a new batch b. The purpose is to determine for the input batch b and latent variables in the batch z_b, the most probable category y that the batch belongs to. Thus, our null hypothesis H_0 is that the two means μ_y, μ_b of object classes y and input batch b are equal, which implies that the distributions from which y and b are sampled share the same means and covariance.

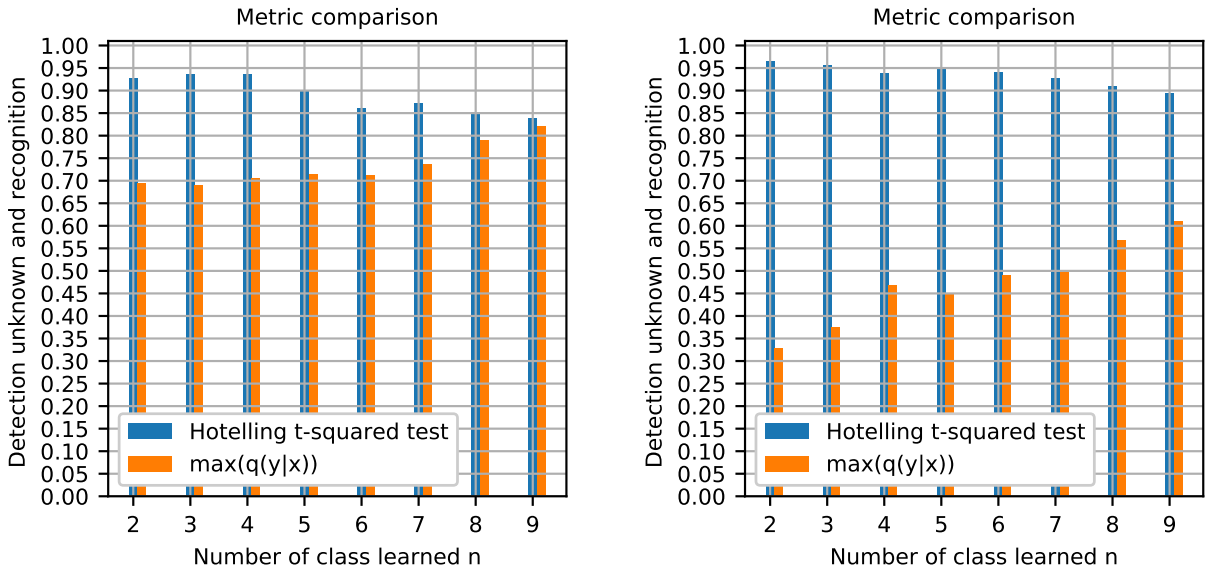


Figure 4.5: Offline open-set recognition accuracy (i.e. new-class detection + known-class recognition) on MNIST (left) and Fashion-MNIST (right) averaged over 3 runs, comparing our proposed Hotelling t-squared test with softmax.

The null hypothesis H_0 is that the two samples are from the same distribution. It is rejected if the left-hand side of Eq. 4.8, referred to as the p-value, is below a critical value. The critical value is used to define the confidence interval by thresholding the p-value computed by the Hotelling t-squared test. For the choice of its threshold, we will take into account its statistical meaning among all the samples. A common choice is a 95% confidence interval, and this implies 0.05 as the threshold for p-value. Although this is a sensible choice, empirically we have tested other p-values for this experiment (lower than 0.05) but did not notice significant differences.

To compare the performance of the Hotelling t-squared test with other state-of-the-art approaches based on the method of thresholding, we take a simple example for illustration. We consider the case where our model is trained on ground truth class changes, only on the part of the classes to give an illustration and to compare different novelty detection approaches. Concretely, we train our model on a limited number of n categories, integrating the online estimation of the means and covariance matrix of each class. During training, we use the ground truth labels as an oracle for novelty detection and show the performance of novelty detection for the remaining $10 - n$ unknown classes (without oracle) and the recognition of the n learned classes using the batches of the test set. We take $n = 7$, which implies learning on 0-6 and evaluating on 0-9 on the test set, to compare the performance of the Hotelling t-squared test with the maximum softmax function activation $\max(q(y|x))$ of posterior estimation, as introduced in section 4.2.3. In preliminary offline studies of novelty detection and recognition, the Hotelling t-squared test outperformed the other metric as shown in figure 4.5 in terms of open set recognition, evaluating both the detection of unknown categories and the recognition of learned ones. The performance is computed following the logic of Eq. 4.2, but in the case of using the Hotelling t-squared test to predict the category of a given batch, the Hotelling t-squared test will associate the batch to the component with the maximum p-value, instead of using the $\max(q(y|x))$, and unknown categories are detected by comparing the maximum p-value among all the existing categories with a threshold (0.05). That is, if all null hypotheses of all known classes are rejected, we consider the sample batch (and the following) to be coming from a new class. In this preliminary test, the n_y is set to the real number of examples of the training set in the category, around 6000 on the MNIST dataset (according to the class distribution) and 6000 on the Fashion-MNIST dataset. The n_y is chosen to be the same as the number of examples in each category, and the performance of detection and recognition can vary eventually according to the choice of n_y since it influence the computation of p-value in the Hotelling t-squared test. The test set

is presented batch by batch (of size $n_b = 100$) giving the average accuracy in figure 4.5 over 3 runs. In our context, the Hotelling t-squared test shows a particular advantage under the presence of unknown categories that the model needs to distinguish from learning categories.

So far, we have applied the test in its standard stationary setting. That means, the distributions of the latent vectors do not change. However, with continual learning, the latent representations of previously seen objects are not fixed but change during the training of the VAE model. In the following sections, we will show how the Hotelling t-squared test is integrated into our model, especially for the continual learning scenario.

4.2.5 Adapting the Hotelling t-squared test to an online context

4.2.5.1 Novelty detection and on-line recognition

We suppose that in the input data stream, classes remain the same at least for some time. There is no immediate shift between classes, and the model will receive data category by category. As we have mentioned in Chapter 1, the hypothesis of continuity in data presence arises from temporal continuity; an object does not shift abruptly in the space in a very short period of time. In chapter 3, we make use of this hypothesis to apply the Page-Hinckley test in the continual learning scenario as a first step. However, in this chapter, we now also consider that objects can reappear multiple times in order to simulate what an autonomous agent can encounter in a realistic scenario. Our objective is to determine if the input belongs to a known class and recognize it or if it belongs to a new one and create a new component in the model.

With this regard, we further suppose that the images in a given input batch are from the same category. As a result, instead of the instance-by-instance detection or recognition, we compute the novelty detection and recognition for the entire batch and associate it to the most probable component based on the batch statistics estimated online for different categories. In practice, in the input data stream, a new class may come at any moment and does not necessarily coincide with the batch boundaries, so a batch will likely contain more than one class. In this case, there can be some bias in the estimation of the batch statistics. A mixture of classes within the batch will lead to a smaller p-value and to more created clusters. However, in this study, we chose to maintain the hypothesis that there is only a single class in each batch.

To decide if the current observation batch corresponds to a given class, we perform a two-sample Hotelling t-squared test [72], which is a statistical test for multivariate normal distributions explained in the previous section. Since there are multiple categories, to determine the category of the input batch, the Hotelling t-squared test is performed on all categories that have been trained before.

We compute the t^2 statistics as in Eq. 4.6. Mathematically, the t^2 statistics can further be approximated by the F distribution as in Eq. 4.8 by computing the critical value (p-value) of F statistics. Therefore, compared to the conventional softmax function, which may produce an incorrect positive response by confusing existing categories with unknown categories, the Hotelling t-squared test has a promising performance in novelty detection when making the Gaussian-distributed assumption regarding the latent variables in our learning scenario. It allows to decide if data comes from an unknown category from a statistical point of view by rejecting all null hypothesis, one for each learnt category.

Nevertheless, we point out that the Hotelling t-squared test, as well as many other hypothesis tests in the literature, are usually performed offline i.e. the model has access to the entire dataset in order to estimate the required statistics. The problem is that, in our continual learning setting, the embedding in the latent space of the VAE gradually evolves both for previously learned representation and the current one because the shared encoder is continuously updated. But computing the exact sample mean and covariance matrix at a given instant while storing all past images do not correspond to continual learning by definition. It is therefore crucial to adapt the Hotelling t-squared test to continual learning by estimating the statistics of each category online, as detailed in the next section. In addition in theory, the number of training examples is n_y, however, for the online scenario we will consider it as an experimentally determined parameter, since statistics are estimated online and not every training example is available to the model. And n_b is simply the batch size.

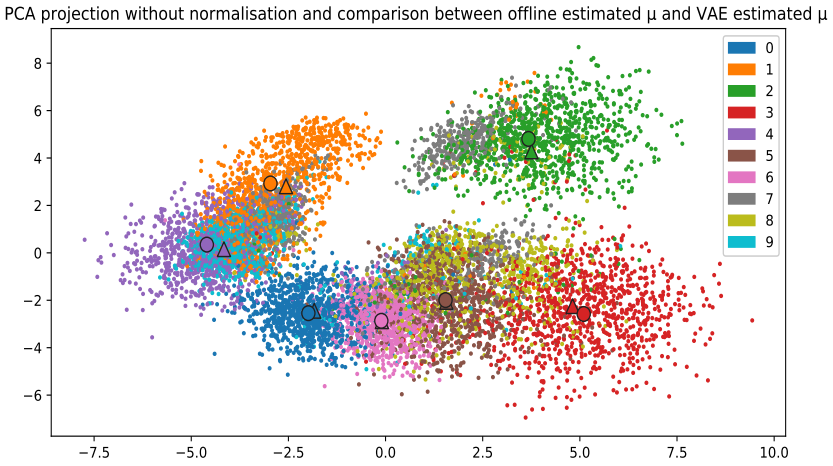


Figure 4.6: PCA-projected latent embeddings for our model on learned (0-6) and unknown categories (7-9) without normalization, and PCA projection of mean estimated offline (marked as triangles) on the training set and the estimation of VAE (marked as dots) on the MNIST dataset.

4.2.5.2 Online parameter estimation

VAE mean and covariance estimation

In our continual learning setting, the embedding in the latent space of the VAE gradually evolves. For the online estimation of the mean and covariance of the statistics for each category, one may first think of the μ and σ that are estimated by the VAE. Indeed, theoretically, the VAE estimations can be used directly as indicators, but in fact they appear to be too imprecise. Here we will give a simple example illustrating the mean and covariance matrix estimated by the VAE model. Taking the same scenario as in section 4.2.4, we integrate the covariance matrix estimated incrementally using a running average. When $n = 7$, the classes 7-9 are considered as unknown. To illustrate the mean estimated offline on the training set and the ones from the VAE, we project them onto 2 dimensions using PCA. Figure 4.6 shows the results. At the same time, we also illustrate the PCA projection on 2D of the latent variables on the test set to show if the estimation of mean can be used to reflect statistics of the test set. Contrary to what was illustrated in section 4.2, we took the sample example, but here we did not normalize the latent variables with respect to their mean and standard deviation. This is to show if the mean estimation of VAE makes sense or, on the contrary, is irrelevant. In figure 4.6, we mark the offline estimated mean as "triangles" and the ones from the VAE as dots. One can see that in the MNIST categories, the two estimated means correspond to each other in certain categories with a slight distance (for example, categories "1" and "3"). On the Fashion-MNIST dataset, however, the means of classes 2, 4, and 6 are very close, with a slightly larger difference between offline estimated means and the mean estimated by the VAE (e.g. between class 2 and 6). Using only the estimation of the VAE, it will be difficult to differentiate between these categories that seem to strongly overlap.

Concerning the estimation of covariance, ideally, the estimation of VAE should approximate a diagonal matrix due to the KL divergence term in the loss function. Therefore, we compare the offline estimated covariance on the training set, marked as "offline", with that of the estimation of the VAE, marked as "VAE" on the left hand. For the MNIST dataset, the illustration can be found in figure 4.9 corresponding respectively to the first and the last column. We also show the difference between offline and VAE on the right hand side in figure 4.9. Similarly, an illustration of the Fashion-MNIST dataset can be found in figure 4.10. A major difference can be observed between the offline estimated covariance and the estimation of the VAE.

This motivates us to search for a more effective way that can better estimate the statistics online,

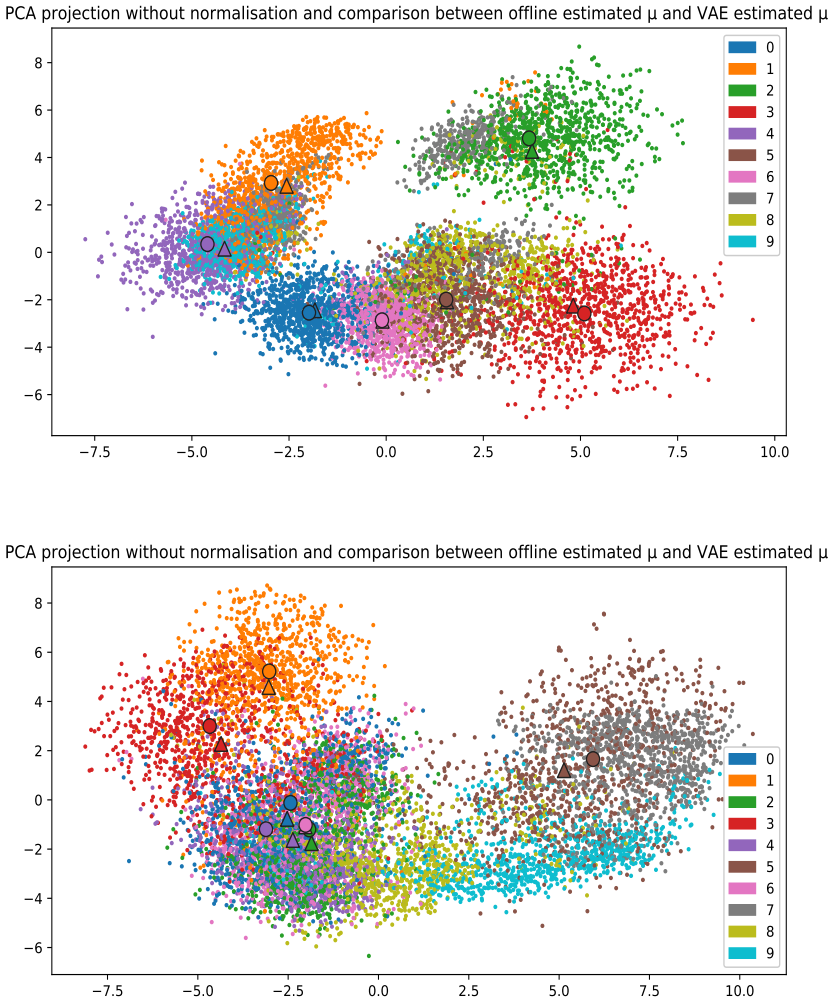


Figure 4.7: PCA-projected latent variables for our model on learned (0-6) and unknown categories (7-9) without normalization, and PCA projection of mean estimated offline (marked as triangles) and the estimation of VAE (marked as dots) on the Fashion-MNIST dataset. *Top:* training set. *Bottom:* test set.

in order to allow for using the Hotelling t-squared test to both recognize learned categories and detect unknown ones. In the following section, we will compare several approaches for the adaptation of online parameter estimation, which is mainly based on running averages.

Running average and adapted covariance

We approximate the offline mean and covariance matrices that we consider exact with an online estimation by using running averages. Since we do not store past observations, the running average is applied to the current class with real observations, as well as to all other classes with synthetic examples coming from the generative replay during training. To differentiate with the notation as it is used in section 4.2.4 for sample mean and covariance, we use the notation of $\bar{z}'_y$ and $\widehat{\Sigma}'_y$ for the online estimated mean and covariance matrix of class y:

$$\bar{z}'_y(t) = (1-\alpha)\bar{z}'_y(t-1) + \alpha z_y(t) \ , \tag{4.9}$$

$$\widehat{\Sigma}'_y(t) = (1-\alpha)\widehat{\Sigma}'_y(t-1) + \alpha(z_y(t) - \bar{z}'_y(t))^T(z_y(t) - \bar{z}'_y(t)) \ . \tag{4.10}$$

where $z_y(t)$ is the embedding of class y produced by the VAE. $\alpha \in (0,1)$ is a small update factor that controls the relevance of recent examples in the estimation of the running average.

Compared to an offline covariance matrix estimation, we found that this approximation slightly underestimates the actual variance, or has inhomogeneous diagonal entries that strongly activate only certain terms while having weak activations for others. This may lead to difficulties when inverting the matrix $\widehat{\Sigma}^{-1}$, resulting in too large values in the t^2 statistics (Eq. 4.6) and eventually to very small p-values for known classes. As a result, small p-values lead to false-positive rejections which imply useless component creations. Figure 4.8 illustrates this problem for the different approximations that we will explain in the following.

We show the offline estimated covariance on the training set, together with the online estimated covariance using running average noted as "entire covariance" in figure 4.9 on the MNIST dataset and in figure 4.10 on the Fashion-MNIST dataset for the first 7 classes.

We experimentally studied several possible variants of the covariance matrix so as to alleviate the problems that the model can encounter while inverting the matrix $\widehat{\Sigma}^{-1}$.

- *Entire covariance*: we will compare the first two proposals with the original online estimated covariance without adjustment $\widehat{\Sigma}'_y$, denoted as "entire covariance".

- *Diagonal*: the regularization of the KL divergence term during training tends to place most of the relevant entries on the diagonal. We will compare with the covariance matrix based only on the diagonal terms of the covariance matrix. Thus for this variant, we take the diagonal entries of the online estimated covariance matrix $\widehat{\Sigma}'_y$:

$$\widehat{\Sigma}^d_y = Diag(\widehat{\Sigma}'_y) \tag{4.11}$$

- *Shrunk covariance*: we apply an operation called shrinkage, frequently used in the literature to improve the estimation of covariance and to alleviate problems related to the inversion of covariance matrix. The shrinkage operation unifies the diagonal entries of the online estimated covariance matrix.

$$\widehat{\Sigma}^{sh}_y(t) = (1-\gamma)\widehat{\Sigma}'_y + \gamma * \frac{\mathrm{tr}(\widehat{\Sigma}'_y)}{d} I \ , \tag{4.12}$$

  with I the identity matrix, $\gamma \in (0,1)$ the shrinkage coefficient. When $\gamma = 1$, the covariance matrix is averaged on its diagonal entries. Using shrinkage, the diagonal entries of the running covariance matrix are more homogeneous and the difference between the eigenvalues is reduced to better approximate the covariance matrix and appropriately estimate the t^2 statistics. We illustrate of the covariance matrix with shrinkage operation applied on the running average in figure 4.9 and figure 4.10, marked as "shrunk covariance" with $\gamma = 0.1$:

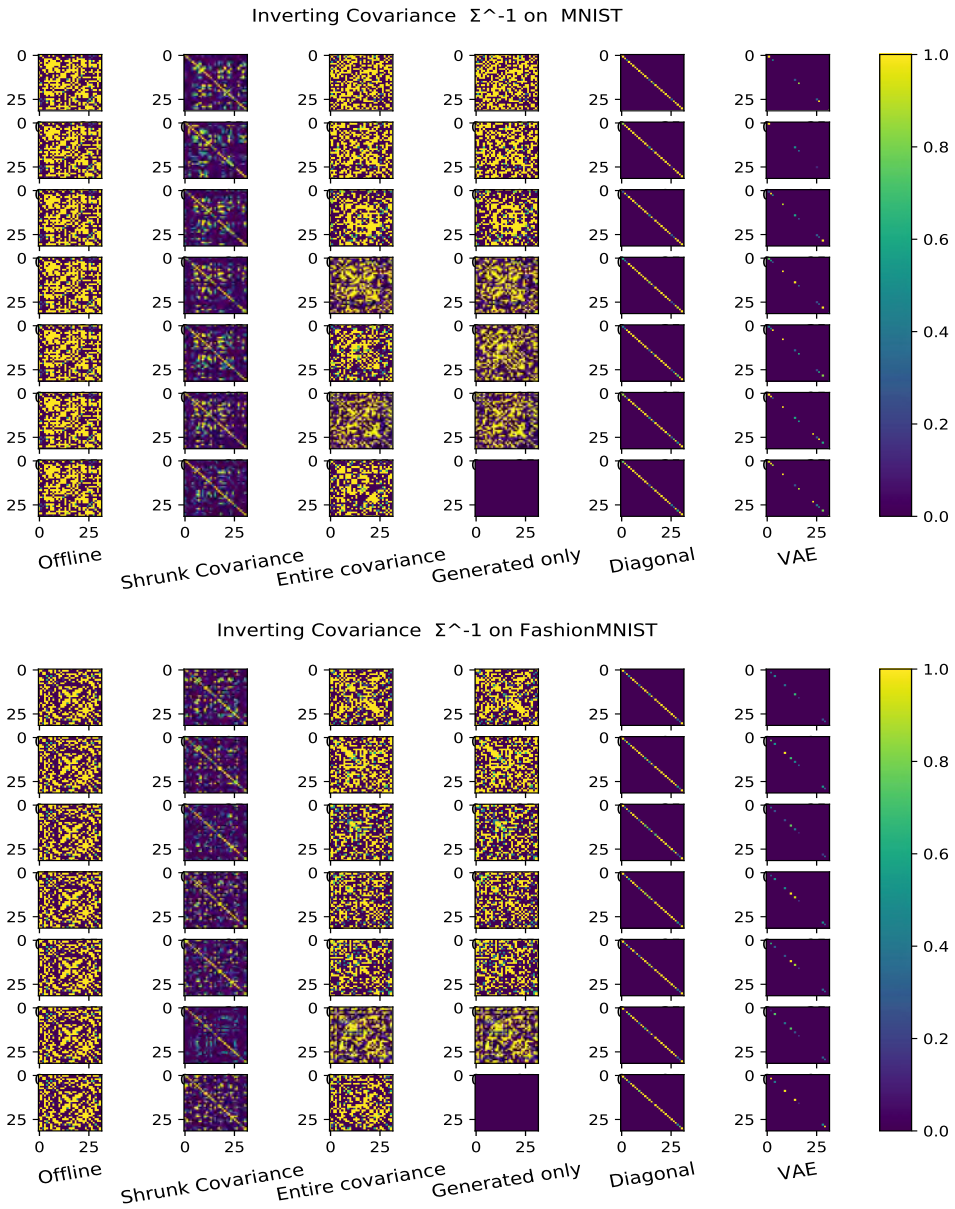


Figure 4.8: An illustration of inverted covariance matrix for different approximations on the MNIST and Fashion-MNIST datasets. Although in the Hotelling t-squared test, the inversion of the *pooled* covariance was used (see Eq. 4.7), this may give an intuition of the problem that may exist in inverting the matrix – the covariance matrices largely differ from each other in amplitude after inverting.

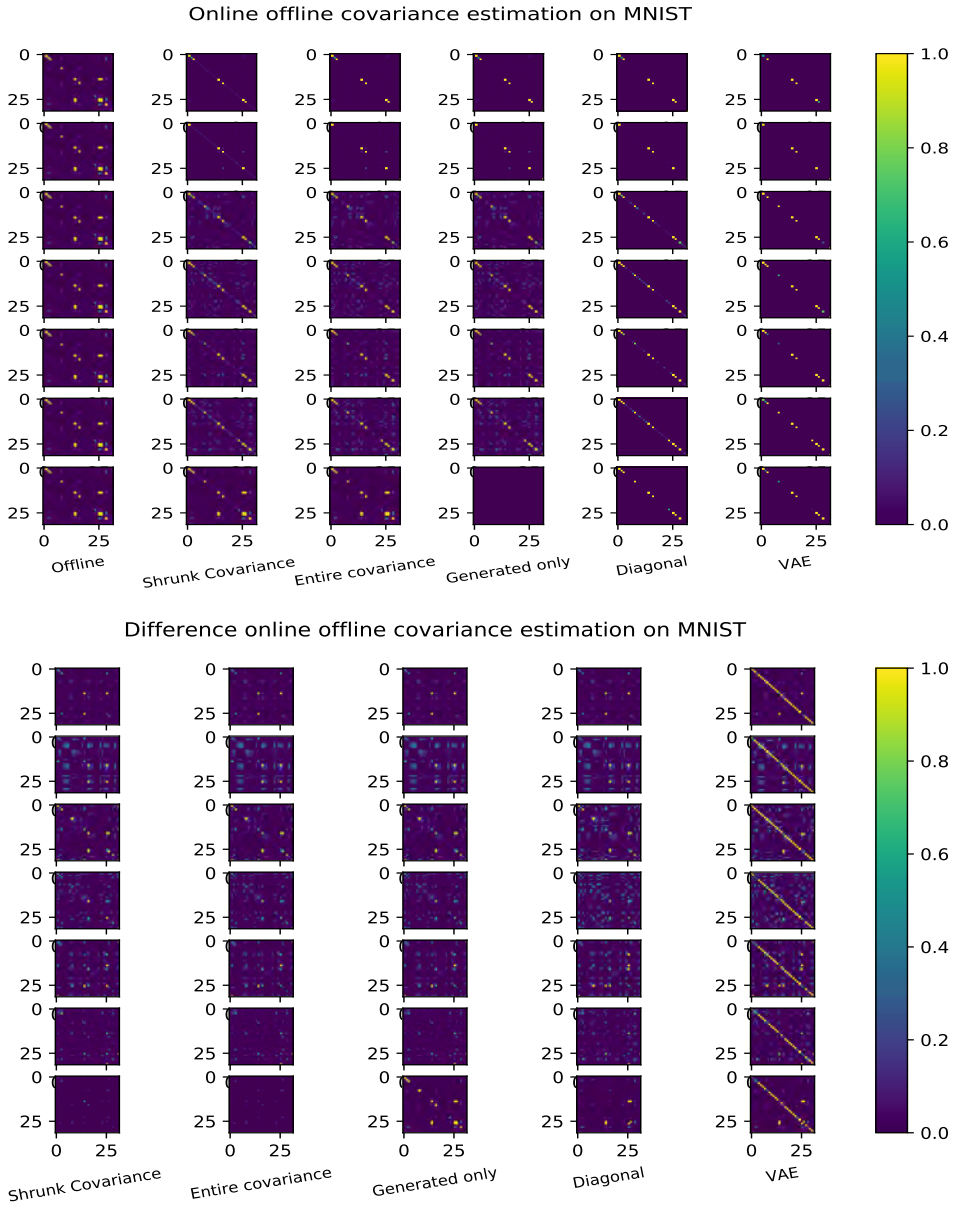


Figure 4.9: MNIST: Results of different variants for online covariance estimation (top) and their element-wise difference to the offline covariance matrix (bottom) for the first seven classes (rows of the plot). *Column 1:* Offline (exact) covariance. *Column 2:* Applied shrinkage. *Column 3:* Entire covariance. *Column 4:* Estimation on generated examples only. *Column 5:* Forcing non-diagonal entries to zero. *Column 6:* Internal VAE estimation.

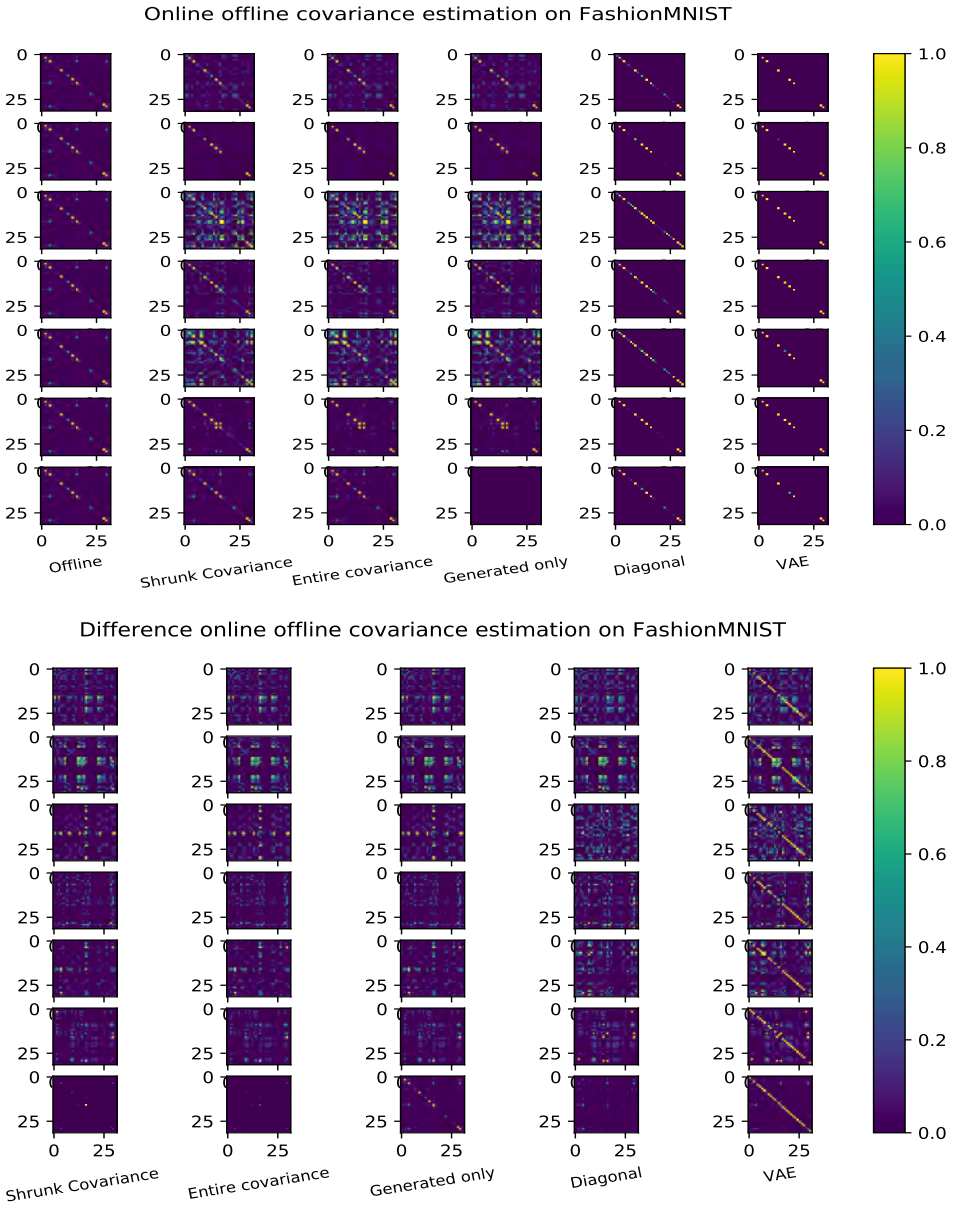


Figure 4.10: Fashion-MNIST: Results of different variants for online covariance estimation (top) and their element-wise difference to the offline covariance matrix (bottom) for the first 7 classes (rows of the plot). *Column 1:* Offline (exact) covariance. *Column 2:* Applied shrinkage. *Column 3:* Entire covariance. *Column 4:* Estimation on generated examples only. *Column 5:* Forcing non-diagonal entries to zero. *Column 6:* Internal VAE estimation.

- *Generated examples only*: there is eventually the possibility of estimating the covariance matrix only on generated batches issued from the generative replay mechanism of our VAE model. Since, during online training, real examples are only available for the currently trained class, this does not affect the way the covariance matrices are updated for previous classes. But their initialisation is different compared to the "entire covariance" method. We illustrate the results of this variant in figure 4.9 and in figure 4.10 on the MNIST and Fashion-MNIST test sets (column denoted as "generated only"). In addition, to detect novelty in the input data, the model should at least correctly estimate the statistics of the current category. Since the generated examples were only for those past ones, this makes this variant more difficult to apply in practice.

Overall, looking at the differences to the offline (exact) covariances, the VAE estimate is the worst, and the covariance matrix estimated from only generated examples also has some large errors, especially on the diagonal. The most accurate estimates are the ones for the entire covariance or only its diagonal, and the shrunk covariance for both datasets.

We then used these online estimation methods to evaluate their effectiveness in our approach for novelty detection and object recognition using the Hotelling t-squared test as explained in section 4.2.5.1. We only compared the first three variants for the online estimation of covariance, since from illustration in figure 4.9 and in figure 4.10 one can remark that the covariance matrix estimated on generated batches gives worse results. Thus, we mainly focus on the comparison between "shrunk covariance", "entire covariance", and "diagonal" using the same protocol as introduced in section 4.2.4. While learning is restricted to $n = (1 \dots 7)$ classes with $10 - n$ classes remaining unknown by the agent, we will compare the estimation of the Hotelling t-squared test on both already seen classes and new ones. For a learned category among n, the model should categorize the instance correctly with a p-value that is superior to a threshold. The model has to detect unknown instances with a small p-value so that they can be considered an outlier. One can note that, although this is only a preliminary test under slightly easier conditions, it illustrates the efficiency of the Hotelling t-squared test when combined with the selected covariance estimator.

In figure 4.11, we show the results of the entire variances (Eq. 4.10), the shrunk covariance (with a shrinkage operation applied as in Eq. 4.12 on the entire variance $\widehat{\Sigma}'_y$), and diagonal entries $\widehat{\Sigma}_{y_d}$ of the entire matrix as in Eq. 4.11, "noted" as diagonal in figure 4.11. In this preliminary test, the value of n_y is fixed empirically to 20 which gives the best overall performances, and the choice of n_y will potentially makes the performance of detection and recognition different, since it impact the computation of p-value which is further compared with the threshold fixed at 0.05 (in this preliminary test). For both datasets, the covariance estimation with shrinkage ($\gamma = 0.1$) outperforms the entire covariance and the diagonal covariance estimate. Note that the accuracy measures both outlier detection and learned category recognition, and the "entire covariance" estimation classifies all test batches as unknown, leading to a decreasing rate. To give a more refined analysis of the results shown in figure 4.11, which was a combination of the detection and recognition, we further separately evaluate the test for unknown categories and the detection and recognition for known categories as shown in figure 4.12. This shows that with all the three variants of covariance estimation, the Hotelling t-squared test can effectively detect novelty among unknown classes with a slight decrease for some classes of the Fashion-MNIST dataset. For the recognition of learned categories, "entire covariance" performs very poorly as all p-values are below the fixed threshold (0.05), hence examples of classes are considered novel. As previously pointed out in section 4.2.5.2 figure 4.8, inverting the entire covariance, lead to large amplitude values and results in too little estimated p-value to correctly recognize learned categories. Using only the diagonal entries partly solved this problem. Nevertheless, the recognition accuracy is much below the one from the method using shrinkage.

Overall, our proposal of online estimation with running averages and shrinkage gives the best empirical results. The shrinkage operation allows for the best tradeoff between novelty detection and online recognition accuracy. The model neither rejected all the learned categories as unknown nor mixed unknown categories with the learned ones. For this reason, we choose to use the online estimation with shrinkage for the following sections.

Finally, in order to integrate this approach for continual learning, we can rewrite Eq. 4.7 as an online Hotelling t-squared test. We simply use the online estimated means using running averages $\bar{z}'_y$ as in

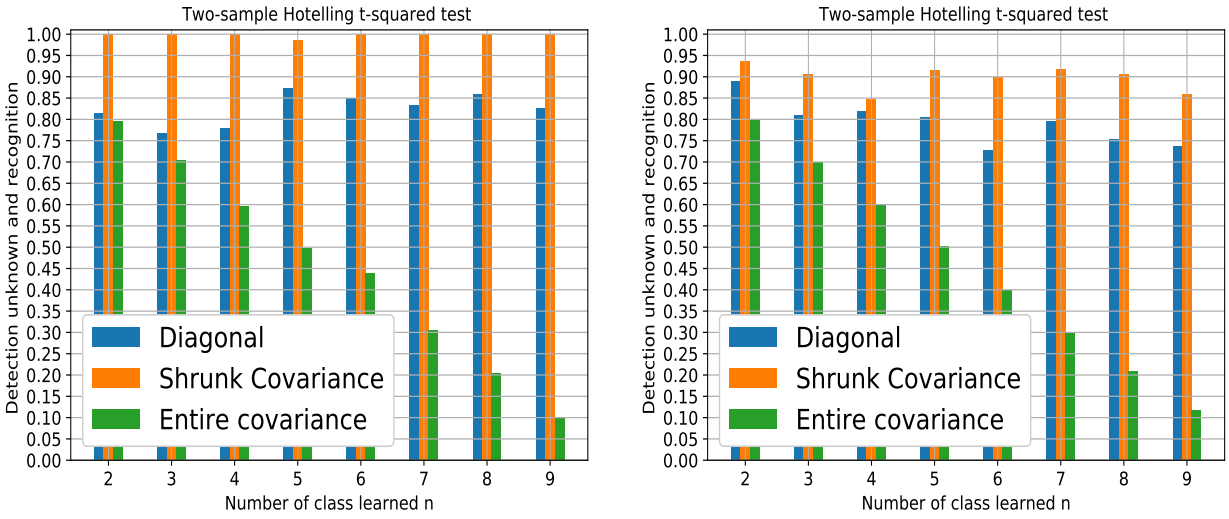


Figure 4.11: The accuracies of detection of unknown classes and the recognition of learnt classes of our approach with 3 variants for online covariance estimation and varying number of learnt classes for MNIST (left) and Fashion-MNIST (right).

Eq. 4.9 and the $\widehat{\Sigma}_y^{sh}$, the online estimated covariance matrices with shrinkage operation as in Eq. 4.10 and in Eq. 4.12 for each learnt class y. With $\widehat{\Sigma}$ integrating the shrunk covariance $\widehat{\Sigma}_y^{sh}$:

$$\widehat{\Sigma} = \frac{(n_y - 1)\widehat{\Sigma}_y^{sh} + (n_b - 1)\widehat{\Sigma}_b}{n_y + n_b - 2} , \tag{4.13}$$

with t^2 statistics:

$$t^2 = \frac{n_y * n_b}{n_y + n_b}(\bar{z}'_y - \bar{z}_b)^T \widehat{\Sigma}^{-1}(\bar{z}'_y - \bar{z}_b) , \tag{4.14}$$

In the following sections, we will introduce our proposed model. In section 4.3.1 we will start by explaining the overall architecture of our model, meanwhile, details on the application of the Hotelling t-squared test to create new classes and to recognize learned ones will be presented in section 4.3.2.

4.3 Proposed Model

4.3.1 Overall architecture and learning

We continue to use CURL's model as a base architecture, a VAE that learns an embedding of different object categories, each represented with a GMM. For the details of CURL, please refer to section 2.4. In addition to the embedding z which is conditioned on the class number (i.e. cluster) y, CURL learns to predict the class probability of a given input x: $q(y|x)$ with a dense neural network layer, for which the maximum can be used for classification. But like many other works in the literature, $q(y|x)$ is mapped through a softmax function, which may not be considered as a real estimation of the probability, and is limited to estimating the input belonging to an unknown category due to "the closed-world assumption". Thus, the main change of our proposal is based on a mechanism enabling both novelty detection, i.e. detection of unknown categories, and object recognition, i.e. recognizing categories that were learned previously. This allows us to self-supervise the learning process through an internal label constructed by the model.

We suppose that for each category, the latent variable will follow a Gaussian distribution modeled by a single component. This induces that each correctly detected category represents one object. This hypothesis is contrary to CURL, which uses a Gaussian mixture for each category, as explained previously

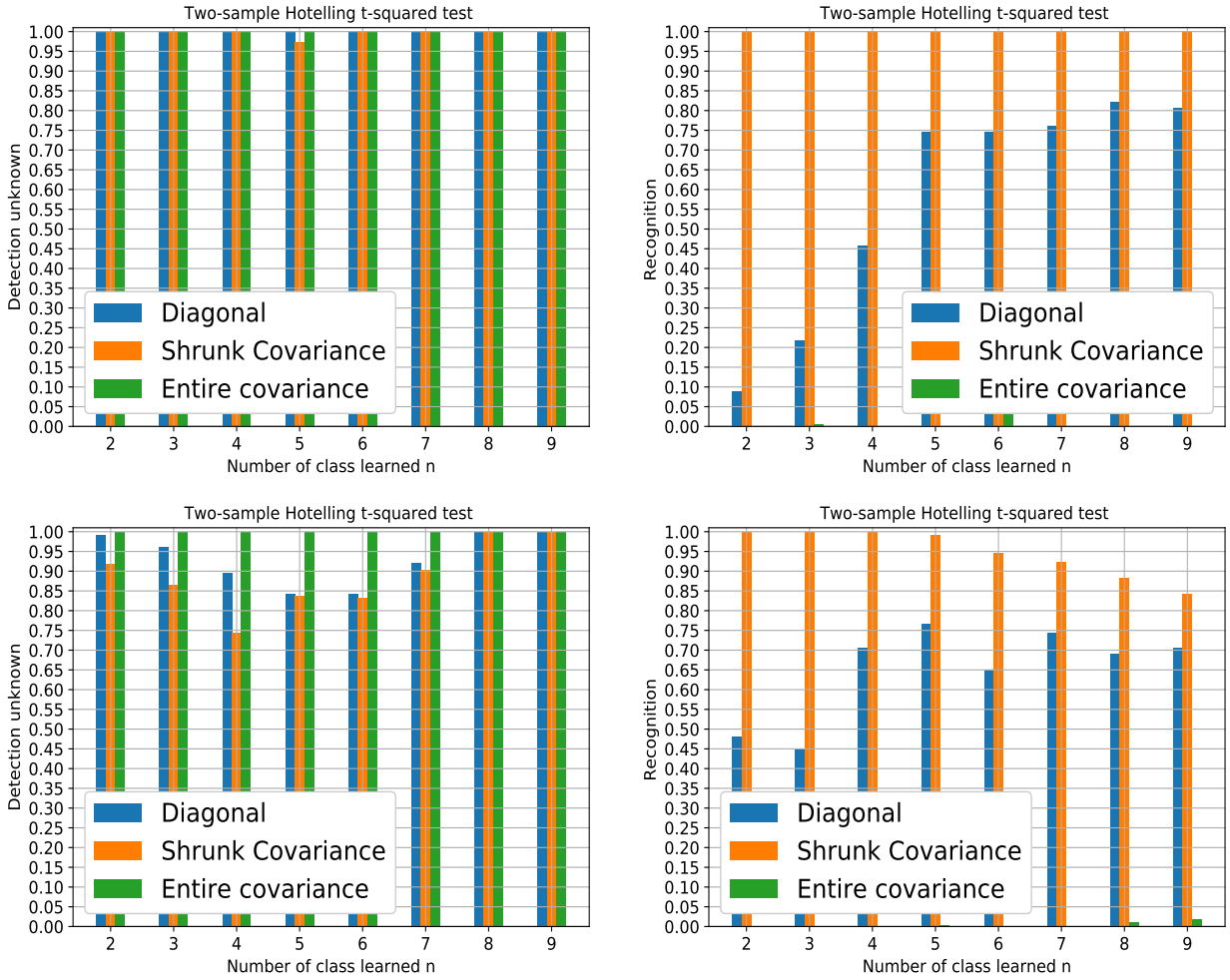


Figure 4.12: Accuracies for novelty detection (left) and recognition (right) for $n = (2 \dots 9)$ known and $10 - n$ unknown classes. Novelty detection is evaluated for all unknown classes and recognition for all learnt classes of the MNIST test set (top) and Fashion-MNIST test set (bottom).

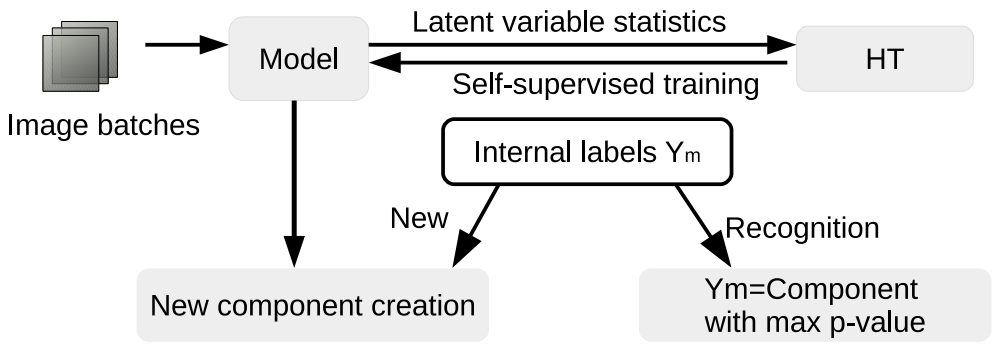


Figure 4.13: Hotelling t-squared test for self-supervision: the model estimates online latent variable statistics of different categories. For each batch the Hotelling t-squared test decides if it is a new category or a known one and, in that case, which one. This enables to create a self-supervision signal that allows learning visual representations with autonomy.

in section 2.4 in chapter 2 and in section 3.2 in chapter 3. Although, in theory, GMMs have a higher expressive power, in practice they may also lead to over-segmentation (i.e. too many components per class) and overfitting. Moreover, we argue that using fewer (cluster) components will facilitate the assignment of object category labels to components, where in the ideal case, there is a unique one-to-one mapping between components and object categories. Thus minimizing the supervision of the system and increasing its autonomy.

We therefore target a model that learns visual object representations continuously in order to recognize them in an autonomous way. That is, it should determine the category of already seen objects and introduce new classes when necessary. It infers the identity $y_m \in \mathbb{N}$ of an input category automatically, and uses y_m as self-supervision to optimize the supervised ELBO loss during training, in the same manner it is done with the model previously presented in chapter 3:

$$E_{sup}(x) = \log q(y = y_m|x) + \log p(x|\tilde{z}^{y_m}, y = y_m) - KL(q(z|x, y = y_m)||p(z|y = y_m)) \ , \quad (4.15)$$

but with the possibility to recognize previously seen objects, and continue the learning of them. A detailed explanation about the internal label inference y_m is presented in the next section.

Similarly to the model of CURL, new class candidates are stored in an outlier buffer. Once the outlier buffer reaches its limit of size, a new component is created. An overall illustration of the model can be found in figure 4.13.

4.3.2 Self supervision with the Hotelling t-squared test

We illustrate in figure 4.13 the implication of the Hotelling t-squared test in our model. The model builds its internal representation during training to estimate statistics for each category and uses the Hotelling t-squared test to detect whether the category of input examples already exists or not, and, if it exists, which is the most likely one that the example belong to. In this way, the model can be self-supervised and learn continuously and autonomously. We perform a two-sample Hotelling t-squared test between each input batch and all existing categories. We continue to use the same notation as in

section 4.2.4, for all the training inputs in the batch, we compare the corresponding latent variable z_b learned by VAE with latent variables z_y of all existing categories, $\bar{z}_y$ and $\bar{z}_b$ are the sample means of latent variables for a category y and for the input batch respectively. Concretely, for an input batch and all existing categories, the Hotelling t-squared test estimates the p-value following Eq. 4.8 the statistics of the F distribution explained in section 4.2.4. Among the p-values p_i for all existing categories i, the maximum p-value corresponds to the most probable category. If it is above a predefined threshold, the input batch will be assigned to that category. If it is below, the input batch is considered unknown, and a new class is created.

Thus, the internal label for self-supervision is defined as $y_m \in \mathbb{N}$ as:

$$y_m = \begin{cases} \text{argmax}_i(p_i), & \text{if } p_i \geq \theta_p \\ y_{max} + 1, & \text{otherwise,} \end{cases} \tag{4.16}$$

where y_{max} is the largest known class index. The model should determine if the data belong to an existing category, which implies recognition of a category given that y_m is associated with the category with the maximum p-value or, on the contrary, if none of the known categories has the same data distribution as the input batch, indicating that a new class has been detected. Then a new component is created, and y_m is assigned to its index. If the category of the input batch is considered learnt, our model can recognize it by determining the most probable category using the Hotelling t-squared test and computing the p-values of each component.

4.4 Experimental evaluation

4.4.1 Datasets

We choose to experiment on the same datasets as in the previous chapter, i.e. the MNIST [91], Fashion-MNIST [179], and EMNIST datasets [33]. MNIST and Fashion-MNIST are composed respectively of handwritten digits and clothes, and both datasets have 10 categories of objects. EMNIST contains images of letters and digits, but there are more categories in the EMNIST dataset, 47 categories instead of 10. Compared to the MNIST and Fashion-MNIST datasets, EMNIST allows showing the impact of having more categories. Detailed descriptions of these datasets can be found in the previous chapter 3.3.1. Before presenting the results, we will explain some common experimental hyperparameter settings in section 4.4.2, the evaluation measures in section 4.4.3 and the different model variants in section 4.4.4.

4.4.2 Hyperparameters

In the following, we present some more details on the training settings. For MNIST and Fashion-MNIST, each category is trained for $2 \times 5000 = 10000$ iterations in total. Compared to the experimental setting used in chapter 3, a class is presented twice, but for each class the number of iterations is divided by two, but the total number of iterations on which the category is trained remains the same (10000). For each iteration, a batch of 100 images is presented to the model. For MNIST and Fashion-MNIST, the total number of steps of the training set is 100000 ($2 \times 10 \times 5000$). For the EMNIST dataset, since there are more (47) categories, each category is trained for 3000 iterations, 1500 iterations for each of the two phases resulting in a total number of training iterations of 141000 ($2 \times 47 \times 1500$). The batch size is set to 100 for all the datasets, MNIST, Fashion-MNIST, and EMNIST.

The main setting of experiments on CURL and our proposal uses the VAE model presented in 2.4 in chapter 2 and in chapter 3. The VAE structure chosen by CURL in the original paper [133] is based on an MLP and is the only studied structure. Concerning the number of hidden layers, the number of neurons in each hidden layer, and the dimension of the latent variable, we use the same *MLP structure* as CURL: an encoder of layer sizes $\{1200, 600, 300, 150\}$ and a decoder of sizes $\{500, 500\}$. However, for comparison, we will consider also a CNN-based VAE to replace the MLP, since the CNN is the most used architecture in the literature and outperforms MLP in terms of robustness when it comes to visual tasks [94]. The CNN structure is inspired by [80, 168], which applies a CNN-based VAE for anomaly

detection and shares some common points with our use cases of learning for novelty detection and object recognition. For the encoder, we use kernels of size 3×3, with stride 2 and the following number of channels: $\{64, 128, 512\}$. For the decoder we use kernels of size 3×3, stride 2 with the following number of channels: $\{256, 64, 16\}$.

For hyperparameters concerning training and optimization, we use the Adam optimizer and set the learning rate to 10^-3. Other hyperparameters that are more related to a specific scenario or model will be detailed in the following sections, where we also give a description of how these hyperparameters are chosen.

Concerning the outlier buffer which stores possible outlier candidates, its size is set to 200 for our model and ours with/wo p_n P-H+HT, and "CURL + HT". For these models, we took a larger outlier buffer compared to the original size of 100, since the Page-Hinckley test or Hotelling test are performed on this outlier buffer (containing consecutive examples), and we slightly increase the buffer to contain 2 batches and avoid potential fluctuations. For CURL, we have empirically compared the buffer sizes of 100 and 200. In fact, regarding the way CURL stores outliers, enlarging the buffer size will be similar to changing the outlier detection threshold while fixing the buffer size, and the buffer size 100 gave better results for the given threshold. For this reason, we set the buffer size to 100 for CURL. For the parameters related to the Hotelling t-squared test, as explained in section 4.2.5.1, the batch size is 100, thus $n_b = 100$, and n_y is empirically set to 20. For online parameter estimation, $\alpha_h t$ is set to 0.999 determined empirically.

4.4.3 Evaluation metrics

To evaluate the quality of the learned representation, we compute different standard metrics, as we have mentioned in chapter 3 section 3.3.4: the classification accuracy (the label of each component is obtained by post-labelling via majority vote on the training set), Adjusted Mutual Information (AMI, see Eq. 3.23 in section 3.3.4) and the Adjusted Random Index (ARI, see Eq. 3.25 in section 3.3.4) measuring the correspondence of the learned clustering w.r.t. the Ground Truth. We also report the number of components created during training and the homogeneity score (see Eq. 3.26). We continue to use the *per-instance* classification accuracy computed on model predictions of individual examples, same as in section 3.3.4, as it is a common clustering metric. However, our model considers that all examples of a batch belong to the same category, as mentioned in section 4.2.5.1, which amounts to a class assignment *per batch*. For a fair comparison, the other models are also evaluated batch by batch via a majority vote of predictions within each batch. For the models without Hotelling t-squared test, the by-batch accuracy is computed with the majority vote of $max(q(y|x))$ over all x in the batch. But for the models including the Hotelling t-squared test, the by-batch accuracy is computed with the prediction corresponding to the component with the maximum p-value.

4.4.4 Model variants

In section 4.3.1, we have introduced our proposal that an adapted on-line version of the Hotelling t-squared test to self-supervised training in the scenario with a possible review of categories. To demonstrate the effectiveness of self-supervision, we perform an ablation study where we further compare with a model based on the combination between CURL and the Hotelling t-squared test of our model, named *CURL + HT*. In this model, the Hotelling t-squared test is applied for the creation of components, instead of the original novelty detection based on comparing the ELBO loss to a threshold. In the CURL+HT model, the Hotelling t-squared test intervenes only in new component creation and thus does not affect learning directly (except for the number of components). Another model, as a variant of ours, is to combine the Page-Hinckley test with the Hotelling t-squared test. We named it *ours with/wo p_n P-H+HT* . This variant detects a new class if the Page-Hinckley test raises a positive response and, at the same time, no previous categories are recognized by the Hotelling t-squared test. For SOINN with SIFT features, we use the same setting as explained in section 4.2.2. However, one needs to note that as SOINN learns the topology online from the input data, the creation and deletion of nodes is actually different from other models, i.e. the threshold is not determined by hand but rather determined automatically with respects to its distance with the neighbors. And the period to insert and delete noisy nodes can also affect the number of nodes as it will decide the frequency to eliminate noisy nodes and to create new ones.

We also compared with the approach called STAM [154], briefly presented in section 2.3.3. STAM is another state-of-the-art unsupervised continual learning model. Its difference with CURL is that STAM does not use neural networks, but a hierarchy of layers based on the clustering of local input images patches at different scales. Compared to the original setting of the STAM paper, we did several slight modifications to stay comparable to other models, CURL and ours. We set the number of classes that the model will see to 1 (instead of 2 in the original implementation), for the scenario where objects are presented class by class. Moreover, we evaluate STAM in the clustering mode, and evaluate the entire test set (to replace the sampling and evaluation process in the original implementation of STAM), in the same way as the other models are evaluated. The clustering of STAM is unsupervised and stays independent from the use of annotation. The clustering mode also implies a spectral clustering on the learned embedding after training and a precomputed similarity matrix. The number of clusters was fixed by hand to 10, although there are some approaches to automatically determine the number of clusters, we decided to apply the approach that was applied in the original paper for a fair comparison. However, compared to other approaches, the predefined number of clusters can give higher scores in metrics like AMI or ARI, which are metrics that influenced by the number of clusters. Thus, this manual setting of cluster numbers introduces a little bias that may give a slight advantage to STAM w.r.t. the other approaches doing this automatically, although their hyperparameters are tuned to come more or less close to this number.

4.4.5 Baseline scenario

In this section, we study the performance of our model and other state-of-the-art models using a simple transition protocol. The first case we consider is a simple scenario that is used as a baseline for comparison between different models. In this protocol, all categories are seen once, one after the other, and then a second time in the same order. This protocol is called the simple transition protocol and explained in more detail in the following.

4.4.5.1 Protocol

In this protocol, each class is presented twice in the learning sequence. Sequences were created based on a class-incremental use case, but a class may reappear, and the model is supposed to recognize it. Training is divided into two phases. In the first part, the model sees all training classes, class by class in a certain order. During this phase, new classes are presented successively, and the model needs to detect novelty in the sequence corresponding to each class change. At the second phase of the sequence, the categories learned in the first phase are replayed to the agent in the same order. The objective of the model in this phase is to recognize all categories correctly and not detect them as new. Examples of the same class are thus grouped together and presented in a random order during the training of that class. And the different classes are presented in a (fixed) random order. Transitions between similar categories are avoided here. The similarity is determined empirically (from the composition of clusters) as it will be discussed more extensively in section 4.4.7.1. Note that, during the training of one class, a training example (image) may be presented multiple times, which is the common procedure in class-incremental continual learning.

Table 4.1 shows the three tested sequences for the MNIST and Fashion-MNIST datasets. For each phase, the sequences were presented in the same order as previously evaluated in section 3.3.5 in chapter 3. In addition, we used the same MLP structure as CURL [133] to fairly compare both models. Nevertheless, as previously mentioned in chapter 3, the number of components created during training is an important factor that can affect the overall performance. As an example, figure 3.5 illustrates the influence of over-segmented clusters combined with the majority vote as post-labelling. When clustering models have created different number of components, the model with more clusters tends to create clusters with less confusion inside. Therefore, the number of clusters is worth studying because of the possible effects it can have on clustering performance. These models were learned with different hyperparameters to influence the number of components (i.e. clusters) created in total, in order to qualitatively analyze the relationship between clustering performance and the number of components. In this protocol, we evaluate CURL, CURL+HT, ours and eventually also SOINN and STAM. But at the same time, we consider our model's

seq	phase	MNIST										FASHION-MNIST									
1	phase 1	3	6	4	2	9	5	1	8	7	0	3	6	9	4	8	2	5	1	7	0
	phase 2	3	6	4	2	9	5	1	8	7	0	3	6	9	4	8	2	5	1	7	0
2	phase 1	9	8	5	7	6	4	3	2	1	0	9	8	7	6	3	4	1	2	5	0
	phase 2	9	8	5	7	6	4	3	2	1	0	9	8	7	6	3	4	1	2	5	0
3	phase 1	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
	phase 2	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9

Table 4.1: Tested sequences (class indices) in the simple transition protocol.

variants combining Page-Hinckley and Hotelling t-squared tests to be similar to ours, so it is not included in this study.

To study the possible impact on the models' clustering performance, we have experimented different threshold values to make the new-class detection more or less sensitive, leading to the variation of the number of components created during learning. For CURL, we vary the threshold for the ELBO loss (see Eq. 2.20), which is an indicator for new class candidates. We used the same setting as in chapter 3. For our model and CURL + HT (CURL combined with the HT test, as introduced previously), it is the p-value resulting from the two-sample Hotelling t-squared test (see Eq. 4.8) that is involved in novelty detection. Therefore, we vary the threshold of the p-value. Concretely, we have chosen the threshold to test by partially following the dichotomy between the threshold that creates the minimum and the maximum number of components. This dichotomy ensures that we have different numbers of components in a given range. However, for CURL, the relationship between the number of components created and the threshold of the ELBO indicator is not linear; the number of components created using these thresholds can be rather dispersed. Other values of the threshold were chosen in such a way that it refines the number of components on the curve (from a predetermined number of components we will test, we find the threshold that corresponds to its two closest adjacent components, taking the middle point as a threshold that needs to be evaluated). We limit the maximum number of components the model can create to avoid unlimited expansion of the model. This reduces the memory and computation footprint of the model. The tested values of thresholds for different models are listed in table 4.2 creating different numbers of components as displayed in the table 4.3.

model	tested threshold MNIST								tested threshold Fashion-MNIST							
CURL (ELBO θ)	-300	-250	-225	-200	-187	-175	-150	-125	-400	-375	-362	-350	-337	-325	-300	
ous (p-value θ_p)	0.05	0.5	0.8	0.9	0.98	0.999	0.9999	0.99995	0.05	0.5	0.7	0.8	0.9	0.98	0.999	0.9999	0.99995
CURL+HT (p-value θ_p)		0.05	0.1	0.15	0.25	0.3	0.35	0.37		0.05	0.065	0.07	0.08	0.1	0.11	0.12

Table 4.2: Tested threshold of ELBO θ for CURL and threshold of p-value θ_p used to create different components in our approach.

model	tested number of components MNIST	tested number of components Fashion-MNIST
CURL	9 16 28 49 56 66 107 150	32 48 65 78 118 122 170
Ours	10 17 20 24 33 48 70 78	15 20 27 32 34 48 71 111 127
CURL+HT	20 24 25 36 49 54 150	21 40 48 51 92 113 170

Table 4.3: Obtained number of components created during learning (by applying the thresholds in Table 4.2)

In this study, we fix n_c the maximum number of components that can be reached by the model for CURL and ours, to 150 for MNIST and to 170 for Fashion-MNIST respectively. The maximum number of components that can be created is limited by an upper bound, which ought to be set high enough that it is not exceeded by the number of components created during training. In theory, this choice of maximum number of components can be arbitrarily high depending on the choice of threshold. But in practice, it increases the required memory as these components are pre-allocated. In our experiments,

this limit ($n_c = 150 \dots 170$) is almost never attained.

The SOINN model is a special case. The SOINN model was originally developed to learn the topology of datasets, and it differs from a clustering approach in that the model creates a greater number of nodes, which are different from clusters, and the threshold for creating new nodes is calculated automatically. However, to vary the number of nodes, we vary the frequency (period) of node insertion and deletion (taking 2, 3, 6, 10 on MNIST and on the Fashion-MNIST) (which was originally 100), with a "shortened" edge age fixed at 5 (which was originally 30). We reduce the period of node insertion/deletion such that the model creates as few nodes as possible, and we show how the number of nodes will impact the clustering performance. This differs slightly from the original setting of the model which focused more on the learning of topology. But in order to make this method comparable to the other clustering approaches, we artificially decreased the number of created nodes.

Another model that we compare with is STAM [154]. STAM proposes two modes in the original paper, a classification mode and a clustering mode. Here we choose the clustering mode and show the impact of the number of clusters during clustering. In STAM, we first compute the similarity matrix, which is a pairwise distance matrix on which spectral clustering is performed. In fact, we perform multiple spectral clusterings while varying the number of clusters k within a certain range, between 2 clusters and 65 clusters to illustrate the impact of the number of clusters on the performance. As with the other models, we then apply a majority vote (instead of the hierarchy voting that was originally applied in STAM) to each cluster to determine its most representative class. The majority vote is used as with all other models, and this avoids to add potential bias to the results related to voting.

In the section that follows, we will discuss how these thresholds allow different numbers of components in the model, and how we select hyperparameters with the optimal number of components.

4.4.5.2 Influence of the number of components

To remind, our model and the one of CURL is based on a VAE that automatically creates a certain number K of Gaussian components y during the unsupervised learning (see Eq. 2.20 for CURL and Eq. 3.20 for our approach), implemented as different dense layer heads in the VAE encoder. These components will naturally correspond to different clusters of the input data in the latent space and have a role in the overall classification performance of the models. Here we investigate the relationship between the number of components and clustering performance evaluated with AMI, ARI and accuracy, for our approach, CURL and CURL + HT. Figure 4.14 shows this relationship on MNIST and figure 4.15 on Fashion-MNIST.

As the learning proceeds, in particular with CURL, the higher the ELBO threshold is, the more susceptible it becomes to responding to poorly modeled examples that represent either general outliers or a possible new category. But when more and more components are created for the same object category during training, the model avoids optimizing existing components with newer instances and may not learn to generalize over the overall intra-class variability. In an extreme case, a component is trained with only very few instances and thus risks overfitting. Nevertheless, in practice there is some benefit in creating more components. In particular, for CURL and CURL+HT, by optimizing Eq. 2.20 the marginalization over components, the KL category regularizer leads to the fact that with a larger number of components K, the model reduces the chances of confusing different categories. Thus, with more cluster components, CURL and CURL+HT tend to achieve better classification accuracy globally at the cost of more subdivisions in the same category and thus lower AMI and ARI scores which measure the similarity between model prediction and ground truth labels. This phenomenon is evident in figure 4.14 and figure 4.15. However, an increase in component creation of the model leads to components that mostly do not correspond to real category changes and negatively impacts clustering autonomy, because creating more clusters will require more post-labelling effort and more supervision to achieve comparable clustering performance.

A model that tends to over-segment different categories will penalize the ARI metric value when instances belonging to the same category in the ground truth label set are separated and predicted differently. The Pearson correlation shows the relationship between the ARI score and the number of components, which is between -1 and 1, where the values -1 or 1 indicate a linear relationship between two factors. In our model and CURL, the correlation between the ARI metric and the number of components

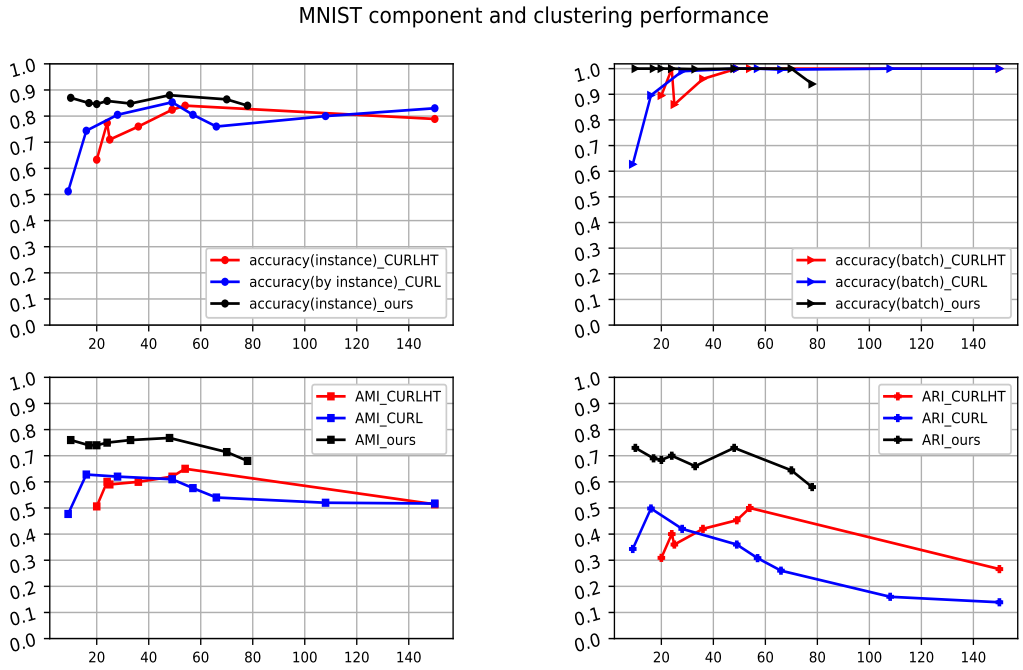


Figure 4.14: The influence of the number of components on clustering performance (accuracy by-instance, accuracy by-batch, AMI and ARI), 1 run for each point for our model, CURL and CURL+HT on the MNIST dataset.

is nearly -1. There is an almost linear relationship between ARI and the number of components, and for both CURL and ours, a lower number of components results in higher ARI scores at comparable clustering accuracy. But a model that has too few components not including all the presented object categories also decreases the clustering performance since there are at least two categories that lie in the same cluster. For this reason, this linearity is respected only when the classification accuracy is also optimized.

Globally, CURL and CURL+HT perform worse than our model on both AMI and ARI metrics. We can observe that our model achieves comparable classification accuracy with only a few clusters, when we consider that each category is modeled by a single Gaussian component in our model. The comparison between our model and CURL+HT validates our model which uses an self-supervised ELBO loss. Despite the comparable classification accuracy, CURL+HT gets a higher ARI score than CURL because the Hotelling t-squared test is used instead of the simple thresholding in the original model of CURL, and the hypothesis testing partially eliminates excessive components and subdivisions. In contrast, for the CURL+HT model, there is little correlation between the ARI score and the number of components. Hotelling t-squared test was only used to detect and create new components in the CURL+HT model during training and did not affect the model's optimization or category prediction.

We illustrate the impact of the number of nodes of SOINN in figure 4.16 on the MNIST dataset and in figure 4.17 on the Fashion-MNIST dataset. This illustration is separated from all other models, because the range of variation of nodes in SOINN is not the same as other models due to its topology learning purpose, the number of nodes reached by SOINN can be larger than all other models. At 140 nodes, the performance of SOINN reaches its optimum on all the metrics.

In the case of STAM, we illustrate the evolution of classification accuracy by instance, the accuracy by batch and the AMI and ARI scores respectively in figure 4.18 on the MNIST dataset and in figure 4.19 on the Fashion-MNIST dataset.

As with the other models, better clustering accuracy is obtained for more clusters, and when the

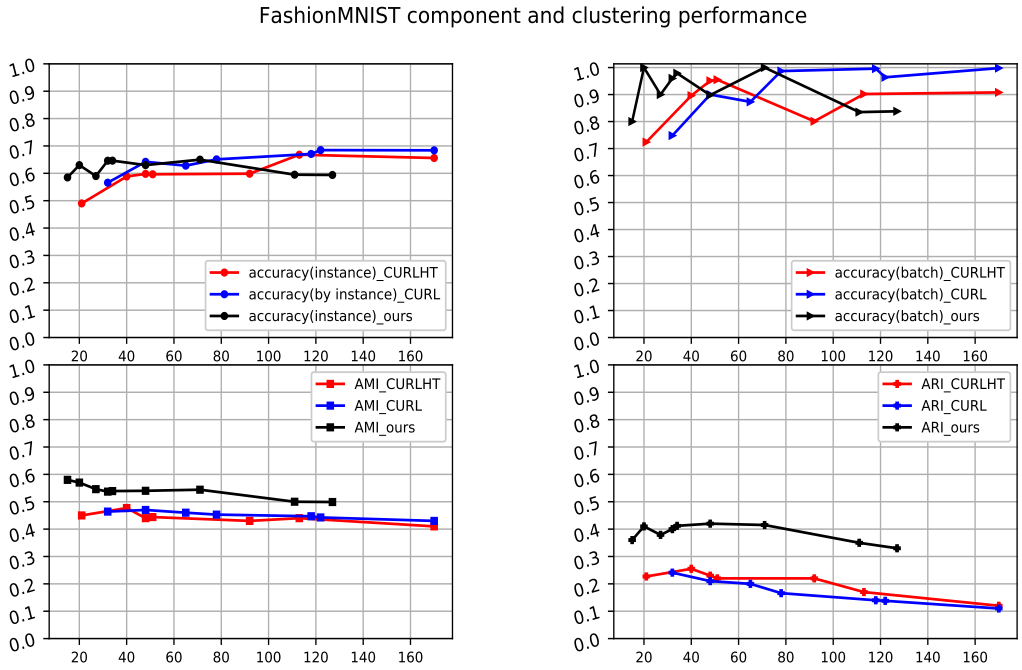


Figure 4.15: The influence of the number of components on clustering performance (accuracy by-instance, accuracy by-batch, AMI and ARI), 1 run for each point for our model, CURL and CURL+HT on the Fashion-MNIST dataset.

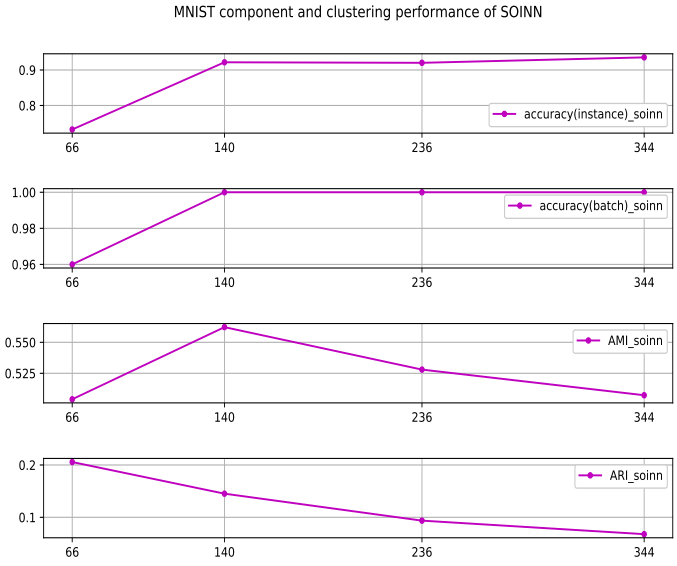


Figure 4.16: The influence of the number of components on clustering performance (accuracy by-instance, accuracy by-batch, AMI and ARI), 1 run for each point for the model of SOINN on the MNIST dataset.

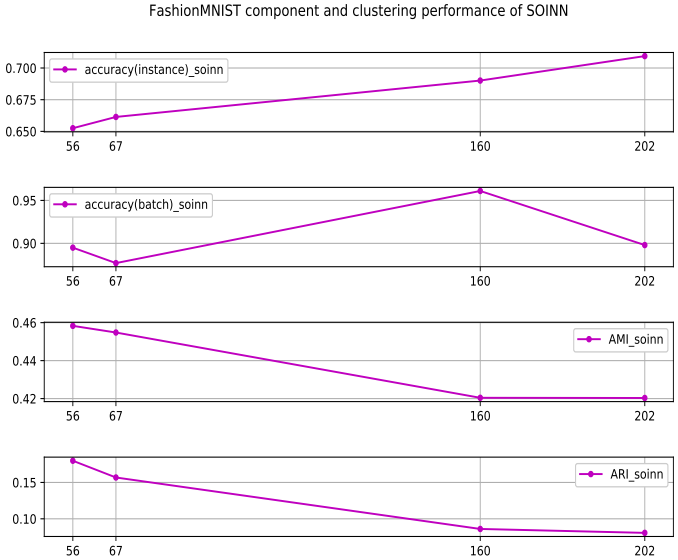


Figure 4.17: The influence of the number of components on clustering performance (accuracy by-instance, accuracy by-batch, AMI and ARI), 1 run for each point for the model of SOINN on the Fashion-MNIST dataset.

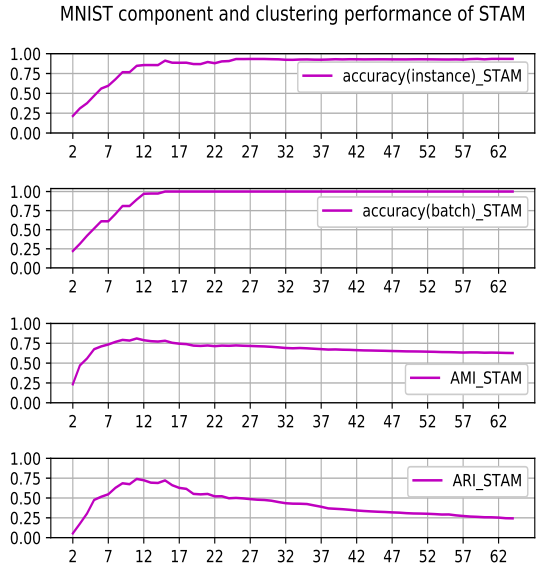


Figure 4.18: The influence of the number of components on clustering performance (accuracy by-instance, accuracy by-batch, AMI and ARI), 1 run for each point for the model of STAM on the MNIST dataset.

number of clusters exceeds 25, the classification accuracy outperforms both our model and CURL (which reaches around 93% for classification accuracy and close to the optimum accuracy of SOINN), but the AMI and ARI drop as the number of clusters increases and the accuracy reaches saturation.

We conclude that there should be a compromise between clustering and classification performance, depending on the number of components created. It is essential to decide on a criterion to determine

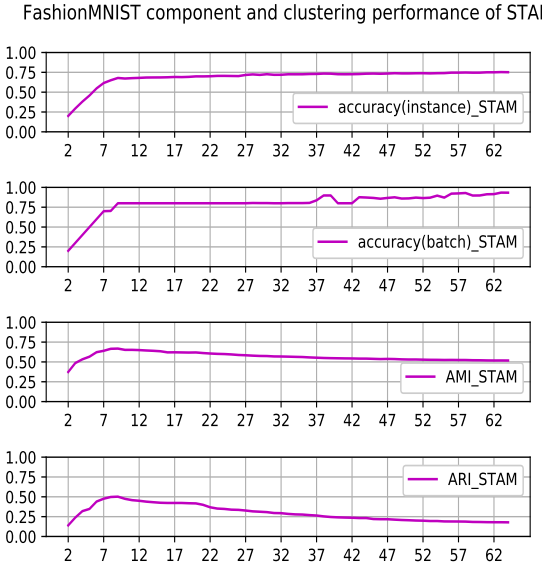


Figure 4.19: The influence of the number of components on clustering performance (accuracy by-instance, accuracy by-batch, AMI and ARI), 1 run for each point for the model of STAM on the Fashion-MNIST dataset.

the appropriate hyperparameters. For example, classification accuracy, AMI, and ARI scores are all evaluation metrics that need to be considered at the same time. Obviously, no model can meet all three of them simultaneously. Neither do we want a model reaching the best classification accuracy, but heavily over-segments the categories requiring extra effort and supervision in post-labelling, nor do we want a model with highest ARI score but mediocre classification performance.

Therefore, as a simple compromise, we choose a criterion that optimizes the mean of the classification accuracy metric and the ARI score. This enables us to find hyperparameters that allow both better classification accuracy and low component numbers. If two hyperparameter settings give comparable ARI and accuracy values (when the difference is less than 0.01 in our experiments), we choose the model with the fewest components.

4.4.5.3 Optimal hyperparameter choices

We illustrate the chosen indicator, the mean accuracy and the ARI score for MNIST dataset in figure 4.20 and for the Fashion-MNIST dataset in figure 4.21. We choose the hyperparameter setting that reaches the optimum indicator of mean accuracy and ARI score, while giving the lowest number of components and staying close to the real number of categories. For CURL, on MNIST, $\theta = -250.0$ (for 17 components), on Fashion-MNIST $\theta = -375$ (for 48 components). For our model, a thresholding method on the p-value estimated by the Hotelling t-squared test detects novelty from online estimated statistics of latent variables. The threshold is fixed at 0.05 for ours on MNIST, since it gives an (almost) optimum mean of accuracy and ARI scores while properly detecting the number of components. The difference between the indicators at 10 components and at 48 components is little significant ($<$0.01, which can also be accounted for by statistical variations). On the Fashion-MNIST dataset, it can be seen that the optimal performance was given by the threshold of p-value at 0.5 (creating 20 components), which outperforms that of a threshold at 0.05 (creating 15 components). The threshold for p-value defines the confidence zone. Nevertheless, from a broader perspective, if the model rejects instances with ambiguities and creates more components, this helps reduce confusion by reallocating more neuronal resources. On the Fashion-MNIST dataset, preliminary studies found that the latent variables z constructed for some categories were statistically close (as illustrated in figure 4.1). Therefore, there are more outliers due

to the confusion between categories. As a result, the model will need a higher threshold to reject more instances that are falsely assigned with a high probability but are in fact outliers. As a side effect, this may also lead to the false rejection of known classes, thus creating more components. With 20 components (at a p-value threshold of 0.5), the model seems to create a clearer separation between categories than with 15 components (at a threshold p-value of 0.05), while the p-value threshold is again determined by our optimization criteria. In the section 4.4.5.7, we will give a more detailed insight into how the choice of the threshold could influence the clustering performance and novelty detection performances.

On both datasets, the same hyperparameters were applied to other model variants based on the Hotelling t-squared test, our model PH+HT and CURL+HT setting p-value threshold to 0.05. For ours with/wo p_n P-H+HT, using the same hyperparameters as ours facilitates the comparison to show the role of the Page-Hinckley test. Furthermore, by using the same hyperparameters for CURL + HT, we can validate the use of self-supervised ELBO loss and the hypothesis of a single Gaussian component by category. For the experiments in the following sections using the MLP-based VAE, we will continue to use the hyperparameters that we have determined here and that provide a good compromise between accuracy and number of components.

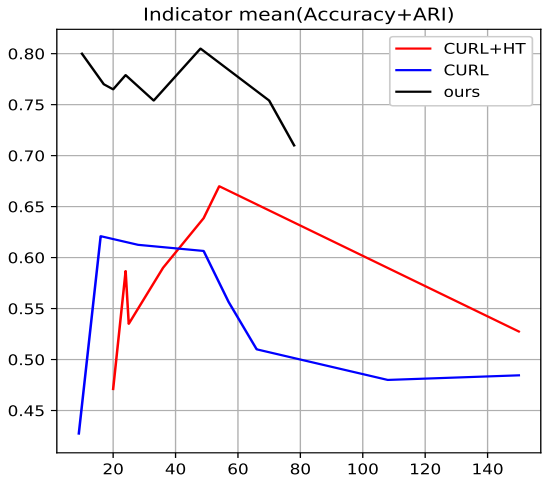


Figure 4.20: Our criteria to choose the optimal hyperparameters: the mean of accuracy and ARI score vs the number of components, evaluated on MNIST for CURL, ours and CURL+HT.

For other models with different novelty detection mechanisms, SOINN and STAM similar strategies are applied to choose hyperparameter setting. For SOINN, on both datasets, we set the maximum node age to 5 and the insert/delete node period to 3 regarding the compromise taking into account both the classification accuracy and the ARI score. We choose 3 for the insert/delete node period because the algorithm will need to compare its first and second winner. Therefore, this choice should be greater than 2. This hyperparameter setting is the same on both datasets. This results in 140 nodes on the MNIST dataset and 66 nodes on the Fashion-MNIST dataset respectively. For STAM, as can be seen in figures 4.22 and 4.23, the best comprise is reached at 15 components on the MNIST dataset. However, to stay with the minimum number of clusters permitting a comparable indicator value, we choose number of clusters as 11. On the Fashion-MNIST, at 9 clusters, the mean of accuracy and ARI score as an indicator is at its best.

4.4.5.4 Overall Results

We compare the results of all methods: SOINN, CURL, CURL+HT, ours PH + HT and STAM, using the optimal hyperparameters determined in the previous section. For a detailed descriptions of these models, refer to section 4.4.4. Table 4.4 shows the result for the MNIST dataset of the models by using the optimal choices of hyperparameters that induce the highest mean of classification accuracy and

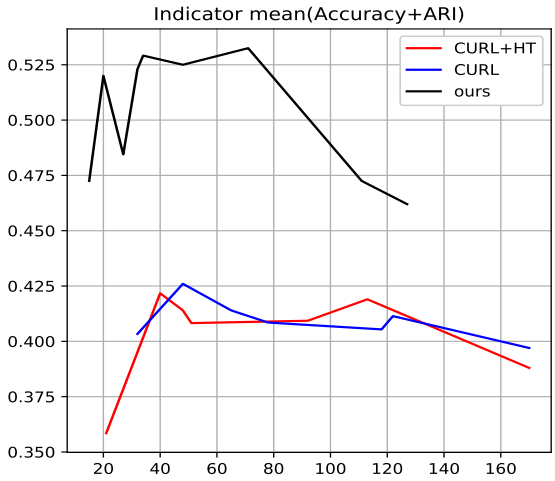


Figure 4.21: Our criteria to choose the optimal hyperparameters: the mean of accuracy and ARI score vs the number of components, evaluated on Fashion-MNIST for CURL, ours and CURL+HT.

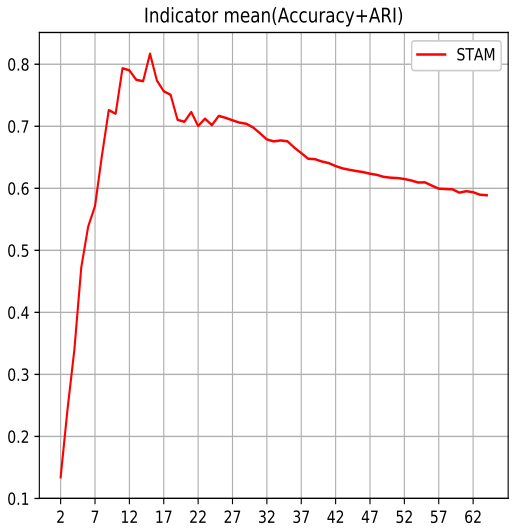


Figure 4.22: Our criteria to choose the optimal hyperparameters: the mean of accuracy and ARI score vs the number of components, evaluated on MNIST for STAM.

ARI score for each method. For CURL this compromise gives better AMI and ARI scores but induces also a decrease in classification accuracy. In addition, since model prediction performance may be distributed differently over classes, the by-batch accuracy could be different from the by-instance accuracy due to the average that is computed by instance or by batch. Especially, the by-batch accuracy may be larger when the prediction performance on different classes vary. At the same time, the prediction of Hotelling could in some cases outperform that of $max(q(y|x))$, which makes the by batch accuracy different when using the prediction of $max(q(y|x))$ or the Hotelling t-squared test applying the online estimated statistics.

Our model obtains higher classification accuracy, AMI and ARI score compared to CURL on the MNIST dataset. Particularly, the number of components created by our model is much closer to the actual number of categories, thereby greatly improving the AMI and ARI scores, while batch accuracy

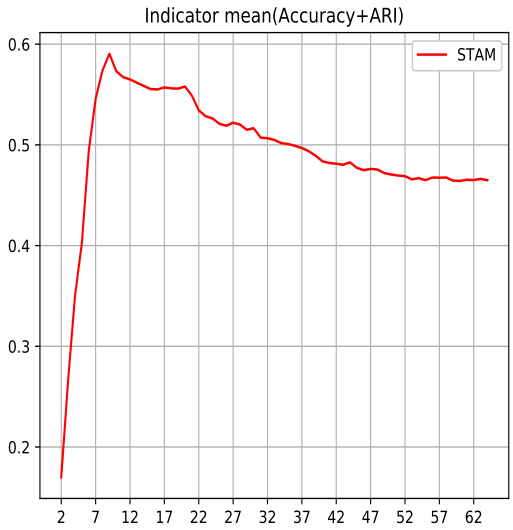


Figure 4.23: Our criteria to choose the optimal hyperparameters: the mean of accuracy and ARI score vs the number of components, evaluated on Fashion-MNIST for STAM.

model	AMI	ARI	# components	accuracy (batch)	accuracy (instance)	homogeneity
CURL [133]	0.598 ± 0.055	0.42 ± 0.087	20.0 ± 2.16	0.916 ± 0.016	0.72 ± 0.035	0.66 ± 0.05
CURL + HT	0.6 ± 0.04	0.41 ± 0.025	22.6 ± 1.69	0.87 ± 0.04	0.72 ± 0.06	0.678 ± 0.057
SOINN SIFT [46]	0.538 ± 0.016	0.155 ± 0.007	126.7 ± 9.42	$\mathbf{1.0 \pm 0.0}$	$\mathbf{0.86 \pm 0.04}$	$\mathbf{0.816 \pm 0.04}$
STAM [154]	$\mathbf{0.81 \pm 0.01}$	0.73 ± 0.01	11	0.89 ± 0.0	0.84 ± 0.01	$\mathbf{0.81 \pm 0.01}$
Ours	0.778 ± 0.012	$\mathbf{0.769 \pm 0.02}$	11 ± 1.41	$\mathbf{1.0 \pm 0.0}$	$\mathbf{0.887 \pm 0.009}$	0.777 ± 0.012
Ours P-H with p_n+ HT	0.77 ± 0.005	0.75 ± 0.019	$\mathbf{10.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{0.878 \pm 0.009}$	0.769 ± 0.006
Ours P-H wo p_n+HT	0.758 ± 0.008	0.735 ± 0.016	$\mathbf{10.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{0.8698 \pm 0.006}$	0.757 ± 0.009

Table 4.4: Comparison with the state of the art on the MNIST test set (averaged over 3 runs) Mean±SD.

remains at 100%. This facilitates post-labelling and improves the autonomy from the point of view that less supervision is required. And when we combine CURL with our novelty detection based on the Hotelling t-squared test (CURL + HT), with the equivalent number of components as CURL, it shows similar clustering performance. The Hotelling t-squared test is only used here to detect new categories, but it does not affect model training except for the number of components. And by combining the Page-Hinckley test with the Hotelling t-squared test, in ours with/wo p_n P-H+HT (with or without p_n smoothing), the Page-Hinckley test reduces even more the number of components since it helps detect abrupt changes. The model uses both the Page-Hinckley test and the Hotelling t-squared test to determine when to create a new component to eliminate components that correspond to the false detection of the Hotelling t-squared test. This combination eventually helps improve the autonomy of the model by detecting more accurately the true number of categories. However, this approach shows no significant improvement in the other measures compared to our approach ("Ours") that only uses the Hotelling t-squared test..

For SOINN with SIFT with chosen hyperparameter settings, at higher number of nodes, SOINN gets a classification accuracy (both by instance and by batch) that is statistically close to that of ours. Moreover, SOINN is not capable of automatically detecting the number of clusters, the approaches create "nodes" but were still far away from the true number of categories in the dataset. For this reason, SOINN gives much lower AMI and ARI scores due to over-segmentation.

Concerning the performance of STAM, the model performs a spectral clustering on learned embedding

model	AMI	ARI	# components	acc. (batch)	acc. (batch HT)	acc. (instance)	homogeneity
CURL [133]	0.47 ± 0.0063	0.19 ± 0.0152	55 ± 4.97	$\mathbf{0.9 \pm 0.003}$	-	0.63 ± 0.007	$\mathbf{0.62 \pm 0.00023}$
CURL + HT	0.473 ± 0.0054	0.253 ± 0.008	31.3 ± 6.34	$\mathbf{0.86 \pm 0.04}$	-	0.58 ± 0.006	0.545 ± 0.015
SOINN [46] SIFT	0.452 ± 0.004	0.147 ± 0.008	74.0 ± 8.52	$\mathbf{0.9 \pm 0.021}$	-	$\mathbf{0.665 \pm 0.012}$	$\mathbf{0.626 \pm 0.017}$
STAM [154]	$\mathbf{0.66 \pm 0.01}$	$\mathbf{0.48 \pm 0.02}$	**9**	0.74 ± 0.04	-	$\mathbf{0.66 \pm 0.02}$	$\mathbf{0.62 \pm 0.01}$
Ours	0.56 ± 0.006	0.39 ± 0.0147	21.33 ± 1.25	0.73 ± 0.05	$\mathbf{0.897 \pm 0.004}$	0.61 ± 0.0087	0.53 ± 0.001
Ours P-H with p_n+HT	0.55 ± 0.02	0.39 ± 0.01	$\mathbf{11.67 \pm 1.89}$	0.79 ± 0.09	$\mathbf{0.873 \pm 0.053}$	0.62 ± 0.02	0.53 ± 0.02
Ours P-H wo p_n +HT	0.54 ± 0.06	0.36 ± 0.06	$\mathbf{11 \pm 0.82}$	0.69 ± 0.07	0.782 ± 0.09	0.56 ± 0.07	0.5 ± 0.06

Table 4.5: Comparison with the state of the art on the Fashion-MNIST test set (averaged over 3 runs) Mean±SD.

with a prefixed number of clusters chosen as defined previously. Its classification accuracy is several points lower than for our approach, but it gets a higher AMI score on the MNIST dataset. Compared to other approaches, its classification/clustering phase is decoupled from the embedding learning phase, and the embedding is composed of distance vectors that measure the distance between patches and centroids. But the clustering will need all the embedding to be extracted at a time and is not incremental, since while computing the distance-based embedding, the centroids are learned and updated over time. Clustering is therefore not completely online. In addition, it did not score as well as our model in the by batch classification accuracy, although the classification accuracy is close in our model and in STAM. This is due to different classification performances in different classes, as will be shown in the confusion matrix that STAM performed well in classifying some categories but tend to confuse the others. The difference is reduced by averaging over instances.

On the Fashion-MNIST dataset (table 4.5), our model detects better the number of categories, but at the cost of a slight drop in by-instance accuracy compared to CURL, but about the same by-batch accuracy. It may be explained by the fact that the Fashion-MNIST dataset is more difficult and by the fact that a higher number of clusters facilitates high accuracy (since mixing classes within one component is reduced), as was demonstrated in section 4.4.5.2. Due to more clusters, the prediction of CURL is also with more homogeneity in each cluster. In the opposite case, STAM does not automatically detect the number of components. STAM has a higher by-instance accuracy but a much lower by-batch accuracy on the Fashion-MNIST dataset; and AMI and ARI scores are also better. Note that our approach that combines the Hotelling t-squared test with the Page-Hinckley test for novelty detection ("P-H with p_n + HT") creates a number of components that is much closer to the real number of object categories with a similar performance on the other measures. This suggests that integrating Page-Hinckley test in our approach could lead to a slightly better model in terms of the number of created components. However, as there is no significant improvement in the other measures (e.g. accuracy) which we consider more important, we preferred to keep the simpler model ("Ours") in the following.

To analyze these results in more detail we further studied the confusion matrices and the composition of clusters for the different approaches. Figure 4.24 shows the results of CURL, our model and "ours PH w/wo p_n+HT" for MNIST. And figures 4.26 and 4.25 show the results for CURL and ours, respectively, on Fashion-MNIST. We only give an illustration over 1 run for the confusion matrix and the composition of cluster for each model, because the component-class association can vary over time, and the number of components differs over different runs. The composition of clusters is illustrated by component following a component creation order. In addition, due to this marginalization, the composition of clusters of CURL did not form a "diagonal-like" matrix, since during training, CURL optimizes an unsupervised ELBO loss that marginalizes over categories such that components that are created earlier can be (accidentally) re-used later for other object categories. Overall, on the MNIST dataset, our model can separate different categories, yet still, some confusion was created with similar categories such as, some '9' are predicted as '7' in CURL, and for both '4', '7' the model predicts '9'. And in addition, CURL tends to confuse the categories "3" with "5" and "8", and recognize "5" as "3" . On the Fashion-MNIST dataset, however, more confusion can be observed among categories '2', '4', and '6'. These confusion cases correspond to similar categories of clothes that are both visually close and hard to separate by the model. CURL still tends to over-segment clusters into numerous subclusters. As one can observe, in our model, as we assumed one component per category on the composition of clusters, at the beginning of learning, the model has not yet stabilized and the latent variable modelled for different categories can be noisy.

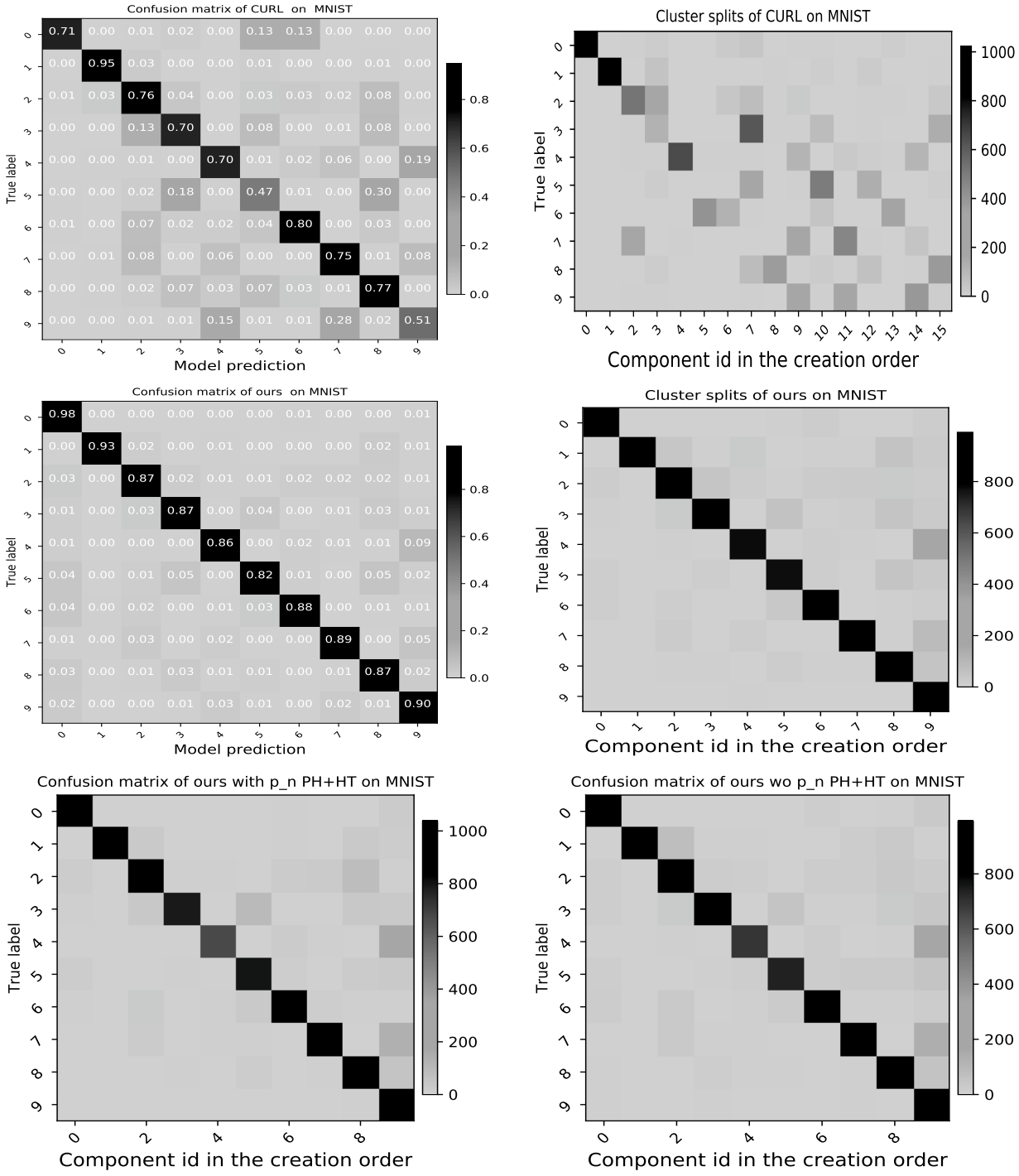


Figure 4.24: Confusion matrix averaged over 3 runs and composition of clusters on 1 run of CURL (above) and ours (middle) for simple transition protocol using MLP on MNIST. In the last row, the composition of clusters for "ours PH with/without p_n+HT" on MNIST.

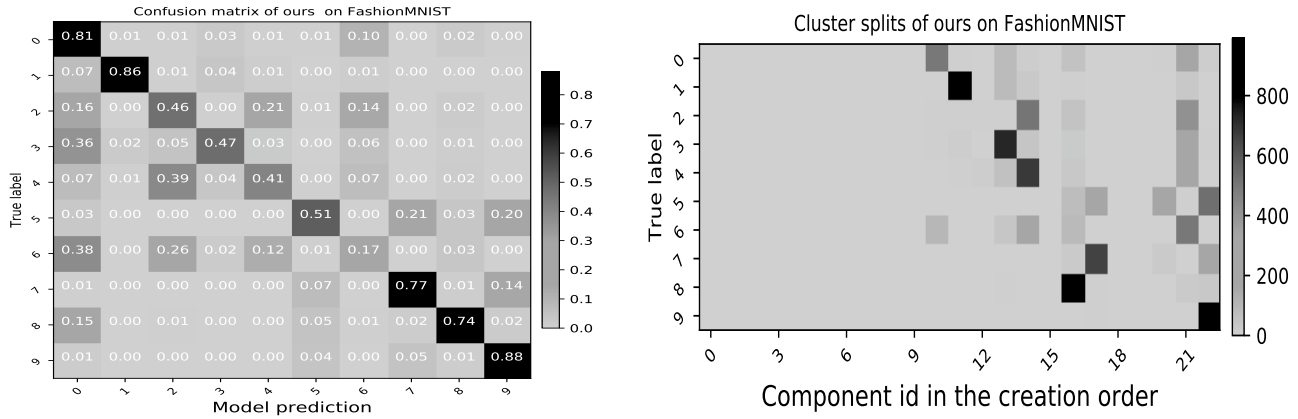


Figure 4.25: Confusion matrix averaged over 3 runs and composition of clusters on 1 run of our model using MLP for the simple transition protocol using MLP on Fashion-MNIST.

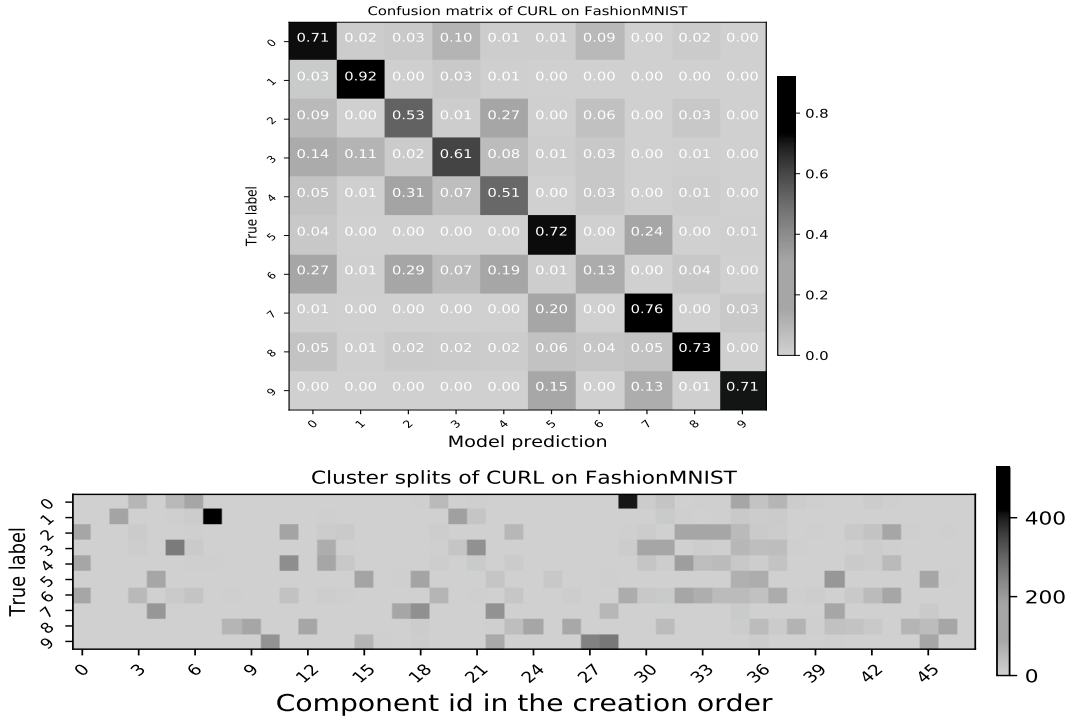


Figure 4.26: Confusion matrix averaged over 3 runs and composition of clusters on 1 run of CURL for simple transition protocol using MLP on Fashion-MNIST.

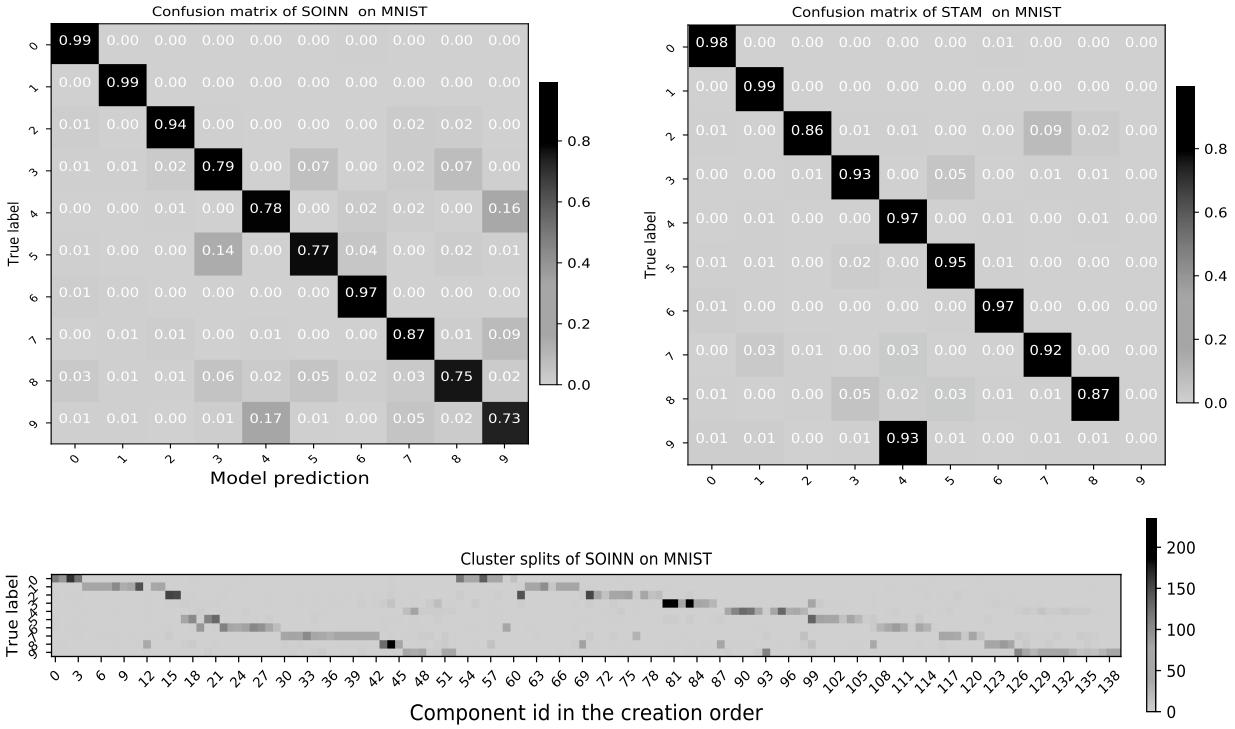


Figure 4.27: Confusion matrix and split of clusters of SOINN with SIFT (top left and bottom) and STAM (top right) using modified hyperparameters on the MNIST dataset.

The estimation of the statistics of the category is poor when the model has only been trained on a few data points and was far away from the true latent variable distribution. For this reason, for our model, especially in the illustrated case of the Fashion-MNIST dataset, there is an "almost empty" case that corresponds to false detection and poor quality components while few examples are predicted to belong to this component. However, after training for a while, the model stabilized and the prediction of the first category was condensed on the last created component while optimizing the supervised ELBO loss. Similarly, during presenting an object that is already learned, if a new component is created for an existing category, the prediction mainly tends to be the component created later due to the cross-entropy term in the loss function. In addition, when our model is combined with the Page-Hinckley test, the model detects new categories by using both the Page-Hinckley test and the Hotelling t-squared test. When the Page-Hinckley test detects abrupt changes on the Fashion-MNIST dataset and combines it with the Hotelling t-squared test, the empty cases at the beginning of learning are partially eliminated, but also result in a slight decrease in accuracy since the model also creates some false negative detection of the category '2'.

Concerning the split of the clusters of SOINN, the predictions as one can remark from the figure 4.27 and in figure 4.28, the model predictions are separated into two diagonal lists of entries when the classes are presented for a second time. However, although the mechanism of inserting nodes in SOINN enables finding a winner among existing nodes, it does not perform well for the recognition of a learned category presented in the past. As a result, new nodes are created for the categories that the model has already learned. However, this can also be caused by our modification of hyperparameters, which is obtained regarding the compromise between classification accuracy and ARI score, and the insertion/deletion period has been reduced so that it can result in the creation of nodes for a learned category. At the same time, for the Fashion-MNIST dataset typically, the categories "2", "4" and "6" are still mixed. For STAM, as one can observe from the confusion matrix on the MNIST dataset, for the category "1" the model over-segments "1" into 3 clusters but tends to mix categories "3" and "5" and categories "4" and "9" when it is evaluated in its clustering mode using spectral clustering. On Fashion-MNIST, the

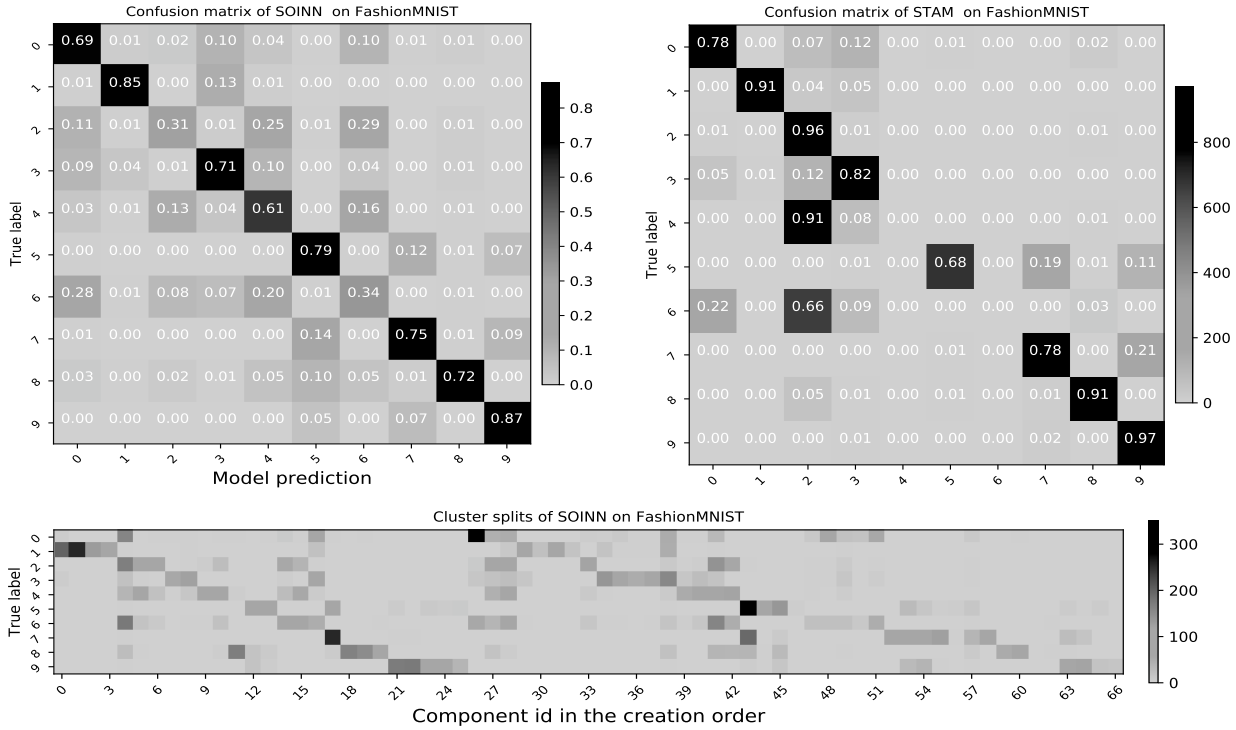


Figure 4.28: Confusion matrix and split of clusters of SOINN with SIFT (top left and bottom) and STAM (top right) using modified hyperparameters on the Fashion-MNIST dataset.

over-segmentation is for categories "1" and "5", but categories "2", "4", and "6" are still mixed. But compared to other models, the categories "2", "4" ,"6" are almost completely predicted as "2"; this has also led to lower by batch accuracy compared to other models such as CURL or SOINN; but the classification accuracy by instance is close due to better classification performance on other categories such as "8" and "9".

We also use the t-SNE projection in figure 4.29 to illustrate the 2D projection of latent variables of ours and CURL on the MNIST dataset and in figure 4.30 on the Fashion-MNIST dataset. We only illustrate ours and CURL as they are the most representative, and for other model clustering such as ours with/wo p_n P-H+HT combining the Page-Hinckley test with Hotelling to detect new categories, we consider that they are similar to ours. Similarly, for the t-SNE projection, our model creates a clustering that is coherent to reflect the true class separation of the dataset, by properly distinguishing different categories. However, for the clustering of CURL, for a single category, '1' for example, is separated into several sub-clusters and creates more intra-class dispersion in the distribution of latent variables.

The t-SNE projection of latent variables on the Fashion-MNIST gets more confusion on both our model and that of CURL, for example, for the category "2", "4" and "6", which is coherent with the confusion matrix; these correspond also to the categories that share many similarities. The t-SNE projection of CURL also shows that the model separated the categories '7' and '9' for example, into many categories.

4.4.5.5 Required amount of supervision for post-labelling

In this section, we will show the importance of annotated examples, following the protocol as explained in the section 3.3.10 in chapter 3, varying and sampling different number of annotated examples n_l. Nonetheless, for an autonomous agent, a smaller number of clusters is preferred since when only a few annotated examples are available for post-labelling, the model with more autonomy has less dependence on the annotation and therefore gets better clustering performance. We demonstrate the influence of the number of annotated training data used in post-labelling in figure 4.31 and in figure 4.32. As previously

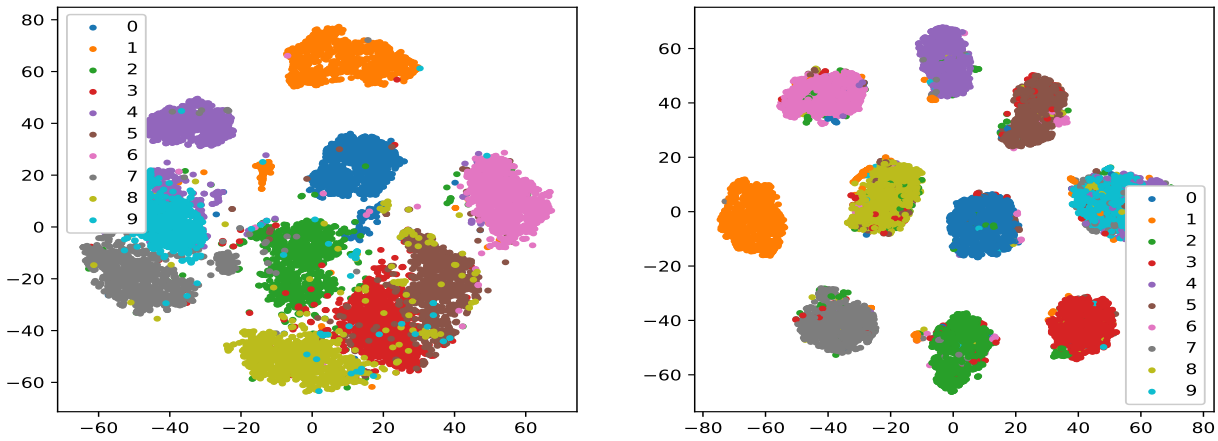


Figure 4.29: 2D t-SNE projection of embeddings of CURL (left) and our model (right) on the MNIST dataset.

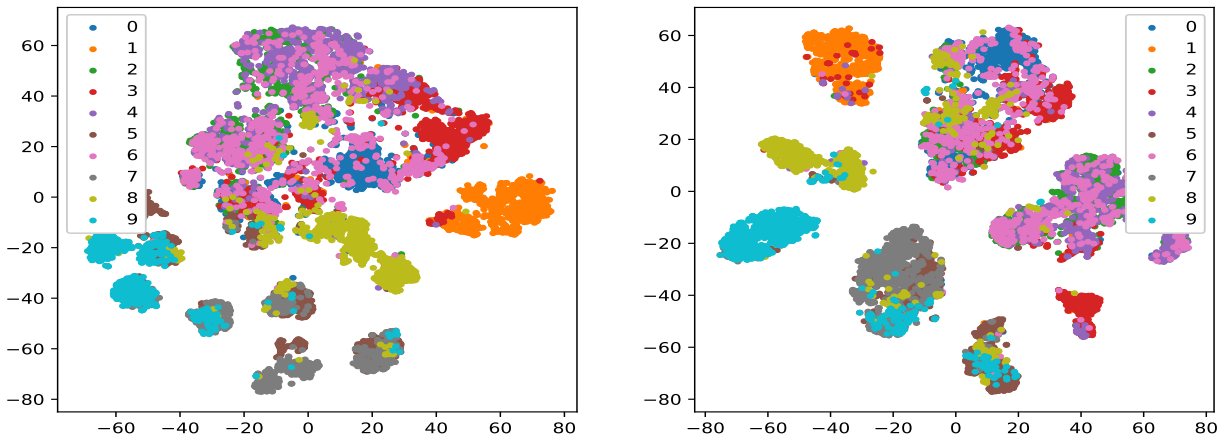


Figure 4.30: 2D t-SNE projection of embeddings of CURL (left) and our model (right) on the Fashion-MNIST dataset.

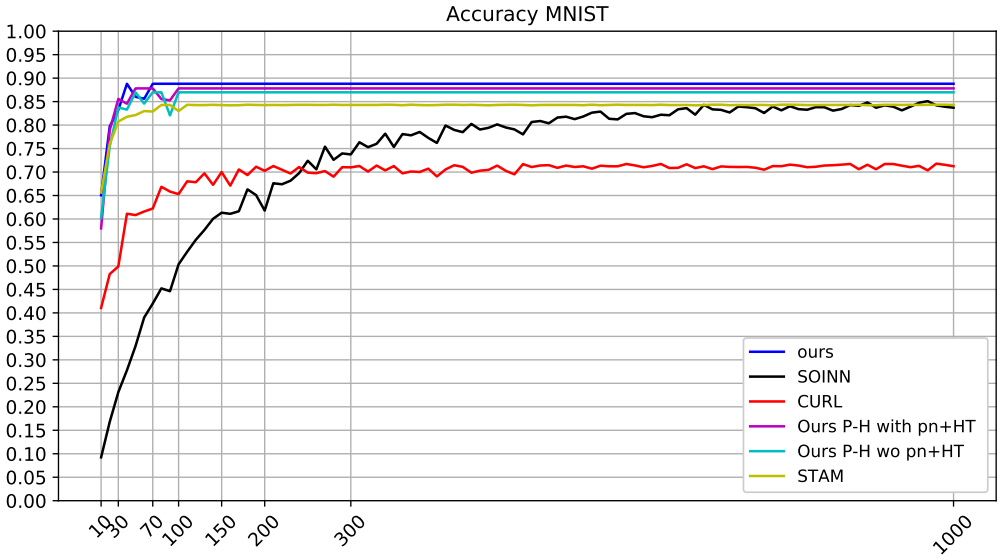


Figure 4.31: The influence of the number of annotated examples in the training set for post-labelling, evolution of classification accuracy by instance on the test set for MNIST.

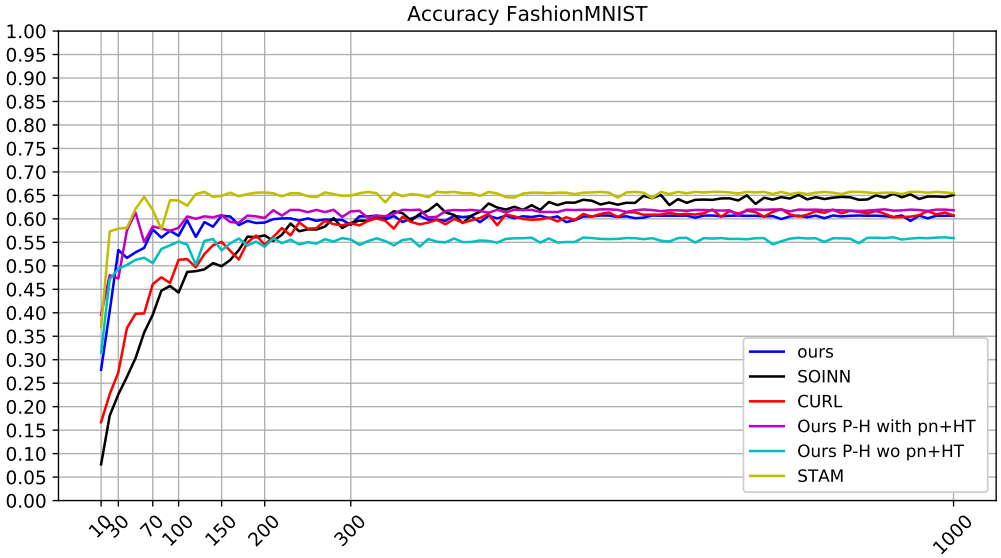


Figure 4.32: The influence of the number of annotated examples in the training set for post-labelling, evolution of classification accuracy by instance on the test set for Fashion-MNIST.

defined, post-labelling is the process of assigning to each cluster the corresponding class during evaluation to compute classification accuracy. We choose to highlight several values on the x-axis, which correspond to values that are close to the number of clusters or nodes for each model; the x-axis is not linear, since different models show different speeds of convergence depending on the number of clusters. This is because the cluster ID is determined by the model and can be arbitrary (for example, it depends on when and how the model creates a cluster, and when there are more clusters than the true number of categories in the dataset). Both SOINN and CURL require a larger number of annotated examples

before finally reaching their optimum accuracy due to more clusters created during training, however, our approach only needs a few points. Intuitively, when more clusters are created and the model prediction is dispersed into different clusters, to correctly associate each cluster to its corresponding class, the number of annotated examples should be sampled in a way that at least several points predicted as that cluster were covered so that when performing a majority vote for the component, it will reflect properly the most probable class. As a result, when more clusters are created for CURL and SOINN, these two models will need more annotated examples before reaching their optimum performance. The speed of convergence to the optimum performance is related to the time constant close to the number of clusters.

However, for the model of STAM, in the classification mode it is possible to use a different number of annotated examples to associate to a category; but in the clustering mode, as we have shown previously, the number of clusters is fixed, and the clustering is performed on an embedding that consists of distance vectors from centroids. For this reason, its speed of convergence is independent of the number of clusters.

Overall, our approach achieves a higher accuracy with fewer annotated examples compared to most state-of-the-art methods. For Fashion-MNIST, only STAM outperforms our model. However, here we only report the by-instance accuracy, and we have seen in table 4.5 that our approach outperforms STAM in terms of batch-level accuracy on Fashion-MNIST.

4.4.5.6 Evolution during learning of recognition and detection performances

We further evaluate the evolution of the recognition and novelty detection performance for each category change in the input corresponding to the ground truth. This allows us to show the evolution of performance in different models, which can also show if there is catastrophic forgetting during learning, for example, if the model fails to recognize a learned category later when new classes are acquired.

We use *detection accuracy by class* to show if the model detects new classes during training, i.e. the binary classification of "known" and "unknown" classes, where 'known' classes are those who have already been learned and unknown classes are those never been presented in the learning sequence at a specific moment of class change. We use thresholds as described previously in section 4.4.5.3 for new category detection and creation during training. For CURL, the detection will be false if the ELBO loss for an unknown category is superior to the threshold (as outliers should be poorly modeled); for our model, the detection is false if a p-value estimated by the Hotelling t-squared test is superior to a threshold (0.05) for an unknown class. Other evaluation includes *classification accuracy by class* to evaluate the clustering performance on learnt categories, i.e. known classes. Both metrics are evaluated by batch, to choose a common metric to compare all models, since the Hotelling t-squared test requires the statistics of the batch to decide on acceptance or rejection as a new category. In figure 4.33 and 4.34, we illustrate the evolution of model performance during training on MNIST and Fashion-MNIST for our approach and CURL. Each line represents the evolution of detection or classification accuracy by class. The lines follow the order in which they are learned. One can look at these figures in two directions. In the figure of classification accuracy by class, one can observe horizontally if there is a noticeable decrease in classification accuracy at any point in time compared with the first time the class was learned, it illustrates forgetting and the model fails to maintain its performance. This is similar to the maintenance of metric performance in the literature. On the other hand, vertically, one can observe the effect of integrating a class (if all classes are not learned by the agent) to compare the performance before and after learning the new classes. Similarly, in the literature, there are metrics like backward transfer, which refers to how learning new classes will impact the performance of the old ones; on the other hand, there are also metrics like forward transfer, which shows how learning a class will impact those already observed by the agent; however, this is not evaluated in this section, since we only analyze the classification accuracy of each ground truth class change after learning the class. One can still refer to the local changes in detection accuracy. This can be seen as a type of forward transfer. The novelty detection performance is impacted in this case when an unknown object is more easily confused with a learned one and is difficult to detect; but reversely, if the model manages to detect this change, it could be easier to converge and helps learning. And more generally, after the tenth class change, the classification accuracy can see a local increase, this is typically due to representing the class for a second time.

For this reason, we include only our model and CURL in the illustration. In our model, we observe that each new class is detected globally at the right time on the MNIST dataset and that the classification

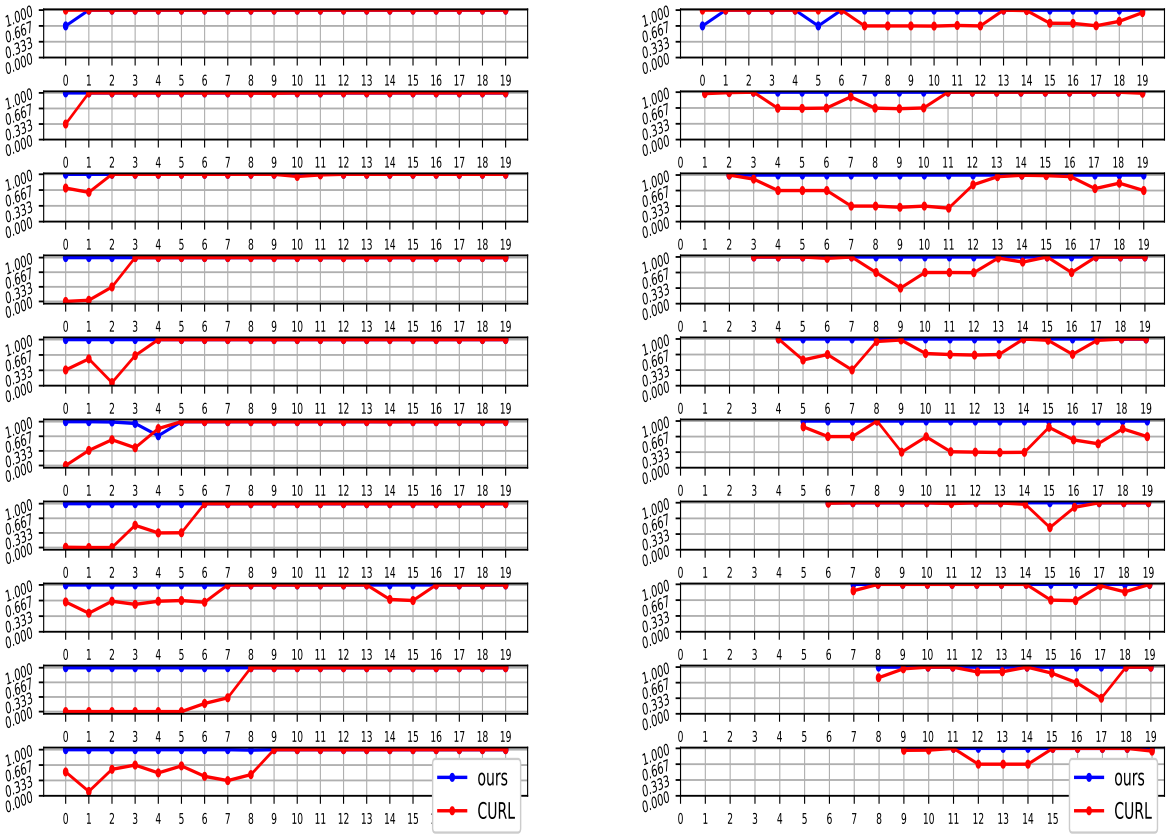


Figure 4.33: The comparison between CURL and ours for evolution of performance in detection (left) and classification by batch (right) for MNIST. For each class, we represent the evolution separately in a line and the evaluation is done over the test set at every ground truth category change. Thresholds for detection/classification (by batch) are the same as in section 4.4.5.3.

accuracy does not drastically decrease during the evolution of the class changes. On the Fashion-MNIST dataset, there were more local variations of detection accuracy and classification accuracy due to the presence of similar classes that are easily confused during learning. And globally our model outperforms CURL in novelty detection since for the i-th line, our model gets higher detection accuracy than CURL before the i-th class changes that correspond to the first time a class is introduced. However, for CURL the detection of new categories, the model has used a rather simple thresholding mechanism, which most of the time does not reject learned examples (since these examples are well modeled and have higher ELBO objectives than the threshold). However, among the unknown new category candidates, there is more variance in the value of the ELBO objective (some are above, some are below the ELBO threshold to detect new categories); for this reason, for unknown categories the detection accuracy is lower than for learned categories. On the other hand, if for an unknown category, its ELBO objective is comparatively high, it could be both a source of false detection and a source of confusion during recognition with a learned category. This is logical since the threshold of ELBO for CURL is chosen with respect to both the ARI (which is correlated with the number of components) and the classification accuracy since in an extreme case we may fix the threshold as the upper bound of the ELBO loss for a learned category after learning has stabilized, this will detect all unknown categories but create excessive components at

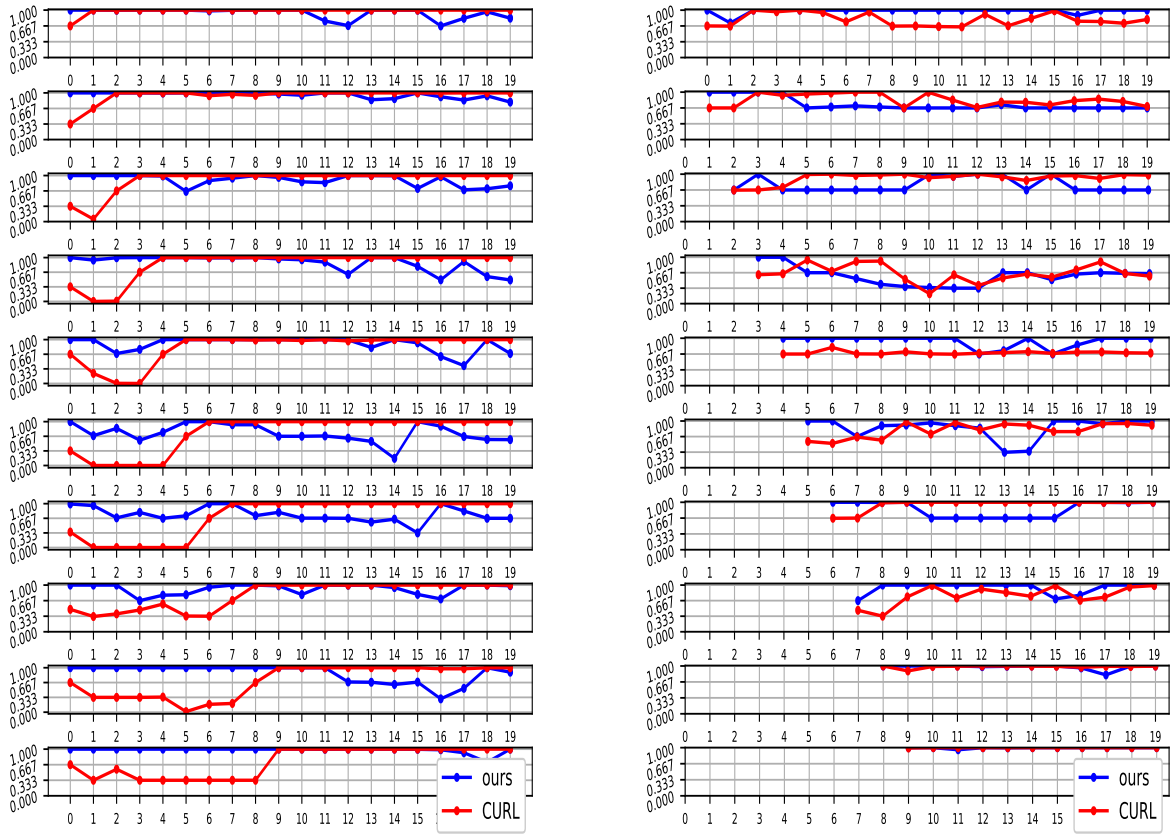


Figure 4.34: The comparison between CURL, ours for evolution of performance in detection (left) and classification by batch (right) for Fashion-MNIST. For each class, we represent the evolution separately in a line and the evaluation is done over the test set at every ground truth category change. Thresholds for detection/classification (by batch) are the same as in section 4.4.5.3.

the same time. While CURL may display some false detections on unknown categories with the chosen threshold, this is not a problem since the model will store those detected as outliers until it fills the outlier buffer (which size is 100) to create a new category.

4.4.5.7 Influence of the p-value threshold

In this section we would like to give an illustration of how the threshold of the p-value choice would influence the model performance, on the Fashion-MNIST dataset. From table 4.6, one can remark that although our model gets similar by instance accuracy, the by batch accuracy differs, and at the threshold for the p-value 0.5, our model gets higher by batch classification accuracy. This is because some categories that are heavily confused, such as "2", "4" and "6" get a slightly better separation with higher p-value threshold.

We illustrate an example of a split of clusters, together with the correctly classified proportion and the class report on the Fashion-MNIST to illustrate why the difference in by batch classification accuracy could be observed for the case of p-value threshold of 0.5 in figure 4.35 and p-value threshold of 0.05 in figure 4.36. As explained previously, with the prediction using the Hotelling t-squared test, the

model	AMI	ARI	# components	accuracy (batch)	accuracy(batch HT)	accuracy(instance)	Homogeneity
Ours (p=0.5)	0.56 ± 0.006	$\mathbf{0.39 \pm 0.0147}$	21.33 ± 1.25	0.73 ± 0.05	$\mathbf{0.897 \pm 0.004}$	$\mathbf{0.61 \pm 0.0087}$	$\mathbf{0.53 \pm 0.001}$
Ours (p=0.05)	$\mathbf{0.57 \pm 0.0}$	$\mathbf{0.395 \pm 0.03}$	13.33 ± 0.94	0.7 ± 0.047	0.83 ± 0.047	$\mathbf{0.601 \pm 0.02}$	$\mathbf{0.53 \pm 0.019}$
Ours P-H with p_n+HT (p=0.5)	0.55 ± 0.02	$\mathbf{0.39 \pm 0.01}$	11.67 ± 1.89	$\mathbf{0.79 \pm 0.09}$	$\mathbf{0.87 \pm 0.05}$	$\mathbf{0.62 \pm 0.02}$	$\mathbf{0.53 \pm 0.02}$
Ours P-H with p_n+HT (p=0.05)	$\mathbf{0.581 \pm 0.002}$	0.383 ± 0.004	8.33 ± 0.47	0.7 ± 0.02	0.767 ± 0.047	0.588 ± 0.009	0.527 ± 0.01
Ours P-H wo p_n +HT (p=0.5)	0.54 ± 0.06	0.36 ± 0.06	11 ± 0.82	0.69 ± 0.07	0.782 ± 0.09	0.56 ± 0.07	0.5 ± 0.06
Ours P-H wo p_n +HT (p=0.05)	0.525 ± 0.07	0.354 ± 0.07	$\mathbf{9.3 \pm 2.05}$	0.64 ± 0.1	0.7 ± 0.14	0.56 ± 0.08	0.483 ± 0.08

Table 4.6: Comparison on Fashion-MNIST (averaged over 3 runs) of different variants with different p-value thresholds Mean±SD.

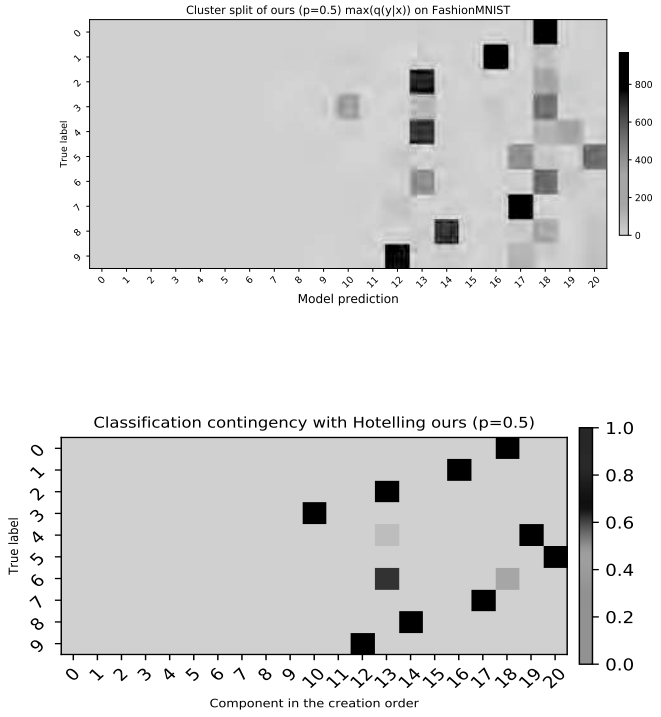


Figure 4.35: Comparison between the composition of clusters predicted using $max(q(y|x))$ on 1 run (seq 2) of our model on MNIST (at p-value threshold 0.5) (above). Cluster splits predicted with Hotelling t squared test that are correctly affected (the component with maximum p-value that is coherent with majority vote labelling) for each class (below).

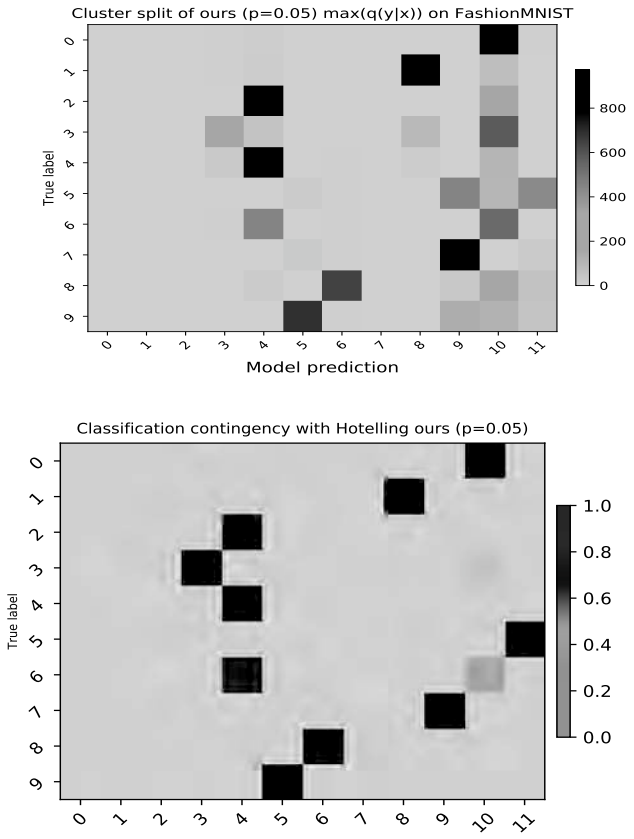


Figure 4.36: Comparison between composition of clusters $max(q(y|x))$ on 1 run (seq 2) of our model on Fashion-MNIST (at p-value threshold 0.05) (above). Cluster splits predicted with Hotelling t squared test of batches (the component with maximum p-value that is coherent with majority vote labelling) for each class (below).

test compares the input batch to all existing categories, for which a component is considered a category. However, on the Fashion-MNIST dataset, when threshold for p-value is at 0.5, although "2", "4" and "6" are still heavily mixed compared to threshold of p-value at 0.05, a little partition of "4" is separated onto component "19", making at least 1 component associate with category 2 and 4; but when the threshold of $p-value$ is at 0.05 as there are fewer components and "2" and "4" gets confused, when we affect each component to its most represented class the class 2 becomes 0 since it is not affected to any component due to the majority vote that associates clusters with categories. Furthermore, when we use the Hotelling t-squared test to classify, we will affect each input batch by comparing p-values of the test of all existing categories, and the classification will be counted correct if it is in coincidence with the majority vote. However, since no component is affected in the case of category "2", it has an impact on the computation of by batch accuracy when compared to the case where the p-value threshold is 0.5. This results in a different by batch classification accuracy. Reduced confidence zones are more likely to reject the input batch for the more easily confused categories, thus avoiding training a new category onto an existing category and adding confusion to both categories.

As in the previous section, in figure 4.37, we illustrate the evolution of detection accuracy and classification accuracy by batch on the Fashion-MNIST datasets. From the change in detection accuracy, one could observe the impact of the choice of threshold on novelty detection before and after a category that has already been learned. With a p-value threshold of 0.5, the model gets better performance in detecting unknown categories, but at the cost of falsely rejecting categories that are already learned. The change of the p-value threshold actually changes the confidence zone, but the advantage is that when there are similarities between classes, at p-value threshold 0.5, the model is more likely to detect novelty and create a new category, therefore reducing the confusion between similar classes. This has also led to better classification performances for these categories.

In this section, we have used the MLP based VAE as a baseline to compare with different models, as this is the original model architecture in CURL. However, compared to MLPs, CNNs are more frequently used in common state-of-the-art models when dealing with images. In the next section, we will use CNN to replace MLP to study the impact of model architecture, which was not explored in the original CURL paper.

4.4.6 Using a Convolutional VAE

In this section, we show the performance of another neural network structure by changing the MLP from the original CURL architecture. We choose to test a CNN-based model architecture on a simple transition protocol, since CNNs are usually applied in visual tasks, and give a better performance on such tasks. We choose the CNN architecture explained in section 4.4.

4.4.6.1 Influence of the number of components

We compare the performance of using a CNN-based VAE with that of an MLP-based VAE, to show the impact of the model architecture on learning performance. Therefore, we only illustrate the performance of the models that share similar and comparable architectures, CURL and ours, which are also the models with the best clustering performance.

As we showed previously with MLP-based VAE, the number of components created during learning impacts the learning performance. Especially for models like CURL that learn a Gaussian Mixture for each category, changing the number of components will influence the capacity of the model for distribution estimation. For this reason, we will further illustrate the relationship between clustering performance and the number of components in the modified structure using a CNN-based VAE, which is similar to that explained in section 4.4.5.1. The idea is to vary the ELBO threshold for CURL and the p-value threshold in our model (the tested thresholds are listed as shown in table 4.7) to create a model with a different number of components, as shown in table 4.8. We use the same approach to choose the different thresholds to study, as it is introduced in section 4.4.5.1.

We have also fixed an upper bound on the number of components the model can contain, as we did in our previous protocol. However, as we have mentioned, this upper bound should be high enough to ensure component creation when detected, but those that are not activated or created (not created for

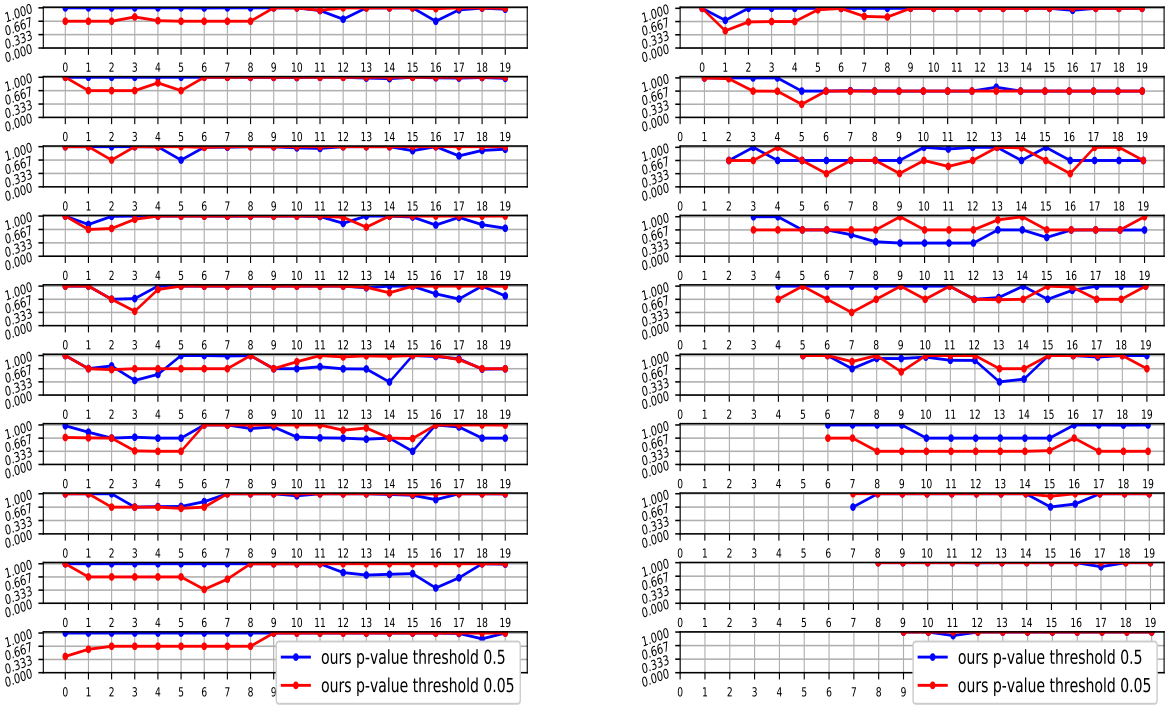


Figure 4.37: The comparison between our approach with two different p-value thresholds (0.5 and 0.05) for the evolution of performance in detection (left) and classification by batch (right) for Fashion-MNIST. For each class, we represent the evolution on a separate line, and the evaluation is done over the test set at every ground truth category change. The thresholds for detection / classification are the same as in section 4.4.5.3.

model	tested threshold MNIST	tested threshold Fashion-MNIST
CURL (ELBO θ)	-200 -150 -143 -135 -128 -120 -100	-350 -325 -300 -287 -275 -260 -250
ours (p-value θ_p)	0.05 0.8 0.99 0.999 0.9999 0.99995	0.05 0.8 0.95 0.999 0.9999 0.99995

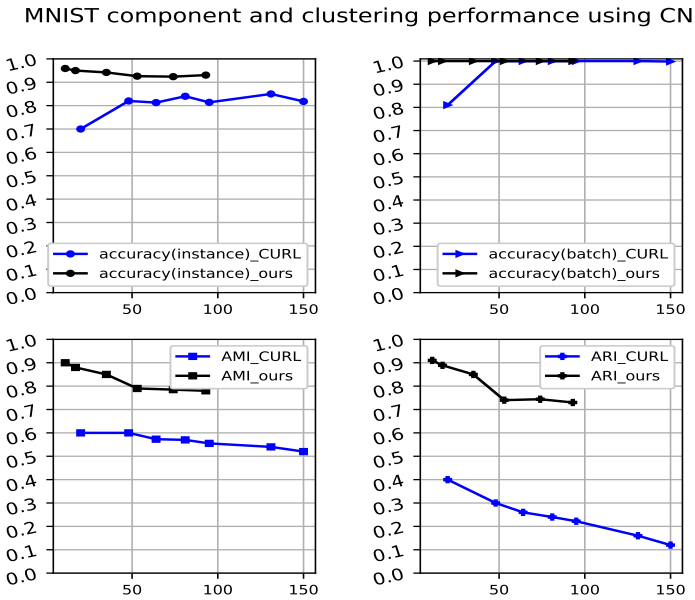
Table 4.7: Tested threshold to create different numbers of components using a CNN-based VAE.

model	tested number of components MNIST	tested number of components Fashion-MNIST
CURL	20 48 64 81 95 131 150	37 56 86 106 120 144 168
Ours	11 17 35 53 74 93	15 29 40 60 96 103

Table 4.8: Tested number of components resulting from thresholds in Table 4.7 and created during learning using a CNN-based VAE.

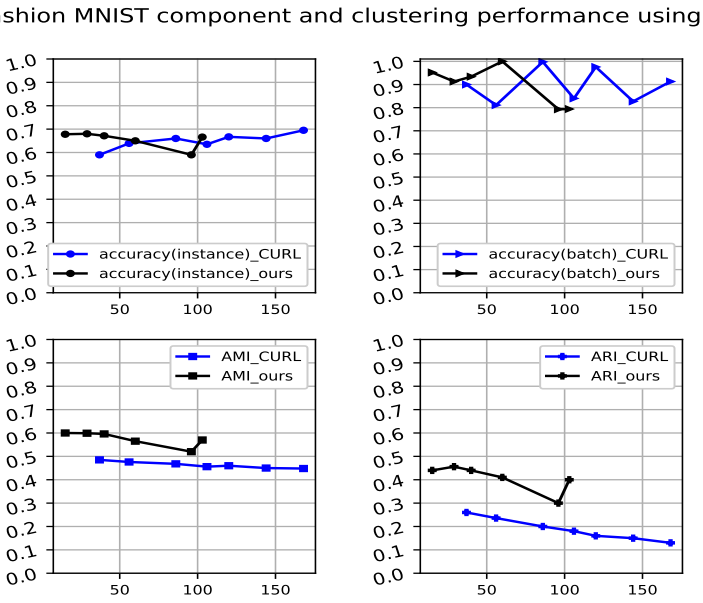
real actually) have a negligible impact on the learning performance. For this reason, we can choose an arbitrary maximum number of components that can include all possible new category detection, but not too large as those non-activated are never in use.

We illustrate the impact of the number of components on the clustering on the MNIST (in figure 4.38) and on Fashion-MNIST dataset (in figure 4.39).



,

Figure 4.38: The influence of the number of components on clustering performance using our CNN-based VAE for the simple transition protocol, evaluated on accuracy, AMI and ARI for MNIST (1 run for each point).



,

Figure 4.39: The influence of the number of components on clustering performance using our CNN-based VAE for the simple transition protocol, evaluated on accuracy, AMI and ARI for Fashion-MNIST (1 run for each point).

One can see that the number of components has a large impact on the clustering performance. Also, on can observe that our model is almost everywhere superior to CURL, whatever the number of created components. As previously shown with the MLP-based VAE, the correlation between the ARI score and the number of components was close to 1. This shows that a model with larger number of components is more likely to penalize the ARI score. As more components imply more subdivision of the cluster, the metrics of AMI and ARI are more sensitive to the number of clusters. It can also be observed that for CURL, a greater number of components results in better by-instance classification accuracy and a reduced chance of confusing different categories. But there could still be some local variations in the by batch classification accuracy due to a varying per-class performance. Especially on the Fashion-MNIST dataset, even if the by-instance classification accuracy is close, the by-batch accuracy still could be different when there are more or less confusions in different classes.

We will need to find a hyperparameter setting that does not create too much subdivision (which may be reducing the model) while maintaining the model's classification accuracy. This has also motivated us to choose hyperparameter settings that optimize an indicator that takes into account both the classification accuracy and the ARI score. As with the MLP-based model, we use the mean of classification accuracy and ARI score as an indicator to optimize both our model and CURL.

In table 4.9, we list the hyperparameters for CURL and our model. The threshold θ of the ELBO loss is set to -150 (which creates 48 components on MNIST) and -325 (which creates 56 components on Fashion-MNIST), respectively, on the MNIST and Fashion-MNIST datasets, using CNN-based VAE. For our model on the MNIST and Fashion-MNIST datasets, the p-value threshold is fixed to 0.05.

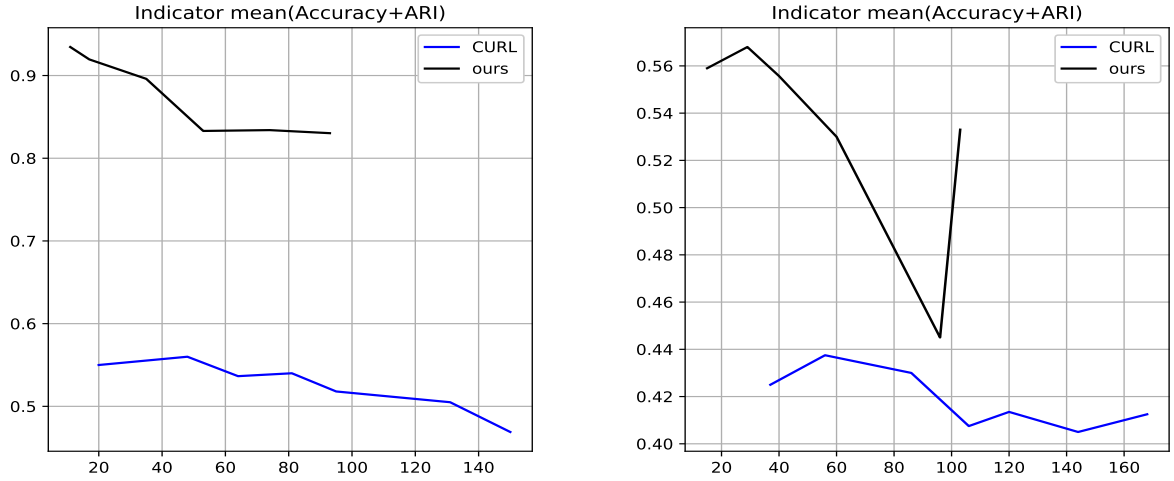


Figure 4.40: The mean of the accuracy and ARI score as a function of the number of components. This criteria is used for optimal hyperparameter choice for the MNIST (left) and the Fashion-MNIST (right) datasets.

4.4.6.2 Model comparison

Overall results

We illustrate our model and CURL using CNN-based VAE in table 4.10 and table 4.11 to show the impact of changing the model architecture.

Here, we only studied the models that gave the best performance in the previous sections and that are comparable. One can observe that, for our model with CNN, better clustering performance can be obtained on the MNIST dataset. Our model with CNN-based VAE outperforms CURL with CNN on all metrics. Our CNN-based model creates a comparable number of components; the number of clusters

dataset	model	θ (ELBO)	p-value(HT)
MNIST	ours (HT)	–	0.05
	CURL	-150	-
Fashion-MNIST	ours (HT)	–	0.05
	CURL	-325	-

Table 4.9: For the models with the Hotelling t-squared test in the scenario with review of objects with our *CNN-based VAE*, We show the threshold for the ELBO loss θ used in each model, and the hyperparameters for the Hotelling t-squared test.

model	AMI	ARI	# components	accuracy(batch)	accuracy(instance)	homogeneity
CURL [133] (MLP)	0.598 ± 0.055	0.42 ± 0.087	20.0 ± 2.16	0.916 ± 0.016	0.72 ± 0.035	0.66 ± 0.05
CURL (CNN)	0.58 ± 0.015	0.279 ± 0.037	51.6 ± 4.5	0.967 ± 0.024	0.7897 ± 0.023	0.766 ± 0.015
STAM [154]	0.81 ± 0.01	0.73 ± 0.01	$\mathbf{11}(*)$	0.89 ± 0.0	0.84 ± 0.01	0.81 ± 0.01
Ours (MLP)	0.778 ± 0.012	0.769 ± 0.02	$\mathbf{11 \pm 1.41}$	$\mathbf{1.0 \pm 0.0}$	0.887 ± 0.009	0.777 ± 0.012
Ours (CNN)	$\mathbf{0.886 \pm 0.009}$	$\mathbf{0.894 \pm 0.011}$	$\mathbf{11 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{0.95 \pm 0.006}$	$\mathbf{0.888 \pm 0.008}$

Table 4.10: Comparison with the state of the art on MNIST (averaged over 3 runs) with the CNN-based VAE model Mean±SD.

created during learning stays close to the real distribution. For our model, better clustering performance is obtained using CNN than using the MLP-based model. And the improvement in clustering performance of both by-batch accuracy and by-instance accuracy can be observed. Our model outperforms STAM on the MNIST dataset, since its local patch-based features may not be discriminant enough for this dataset. However, our CNN-based VAE is more capable of learning the semantic representation modeled by latent variables.

On Fashion-MNIST, our model outperforms CURL in terms of by-batch and by-instance classification accuracy. STAM has slightly higher AMI, ARI and homogeneity scores since those metrics are sensitive to the number of clusters. Its accuracy by-instance is comparable, but the accuracy by-batch of STAM is much lower than for our approach,

For CURL, CNN-based VAE improves the clustering performance at the cost of increasing the number of components on MNIST and with a reduced ARI score and increased by-instance and by-batch classification accuracy, since CNN helps better extract representation and features. However, on Fashion-MNIST, the clustering performance with a CNN-based CURL model is not improved.

Confusion matrices

The composition of clusters of our model and CURL using CNN-based VAE on MNIST and Fashion-MNIST dataset is illustrated in figure 4.41 - figure 4.42; in figure 4.43 for CURL and in figure 4.44 for our approach.

Each illustration shows the components following the order of their creation, and each cluster composition reflects a single run, as the number of created components can vary from run to run. Moreover, as CURL optimizes an unsupervised ELBO loss, the marginalization over different categories makes its composition of clusters separated, while the component category association may vary throughout time.

model	AMI	ARI	# components	accuracy(batch)	accuracy(batch HT)	accuracy(instance)	homogeneity
CURL [133] (MLP)	0.47 ± 0.0063	0.19 ± 0.0152	55 ± 4.97	$\mathbf{0.9 \pm 0.003}$	-	0.63 ± 0.007	$\mathbf{0.62 \pm 0.00023}$
CURL (CNN)	0.47 ± 0.003	0.22 ± 0.0157	62.33 ± 8.26	0.85 ± 0.0262	-	0.62 ± 0.0168	$\mathbf{0.63 \pm 0.014}$
STAM [154]	$\mathbf{0.66 \pm 0.01}$	$\mathbf{0.48 \pm 0.02}$	$\mathbf{9}$	0.74 ± 0.04	-	$\mathbf{0.66 \pm 0.02}$	$\mathbf{0.62 \pm 0.01}$
Ours (MLP)	0.56 ± 0.006	0.39 ± 0.0147	21.33 ± 1.25	0.73 ± 0.05	0.897 ± 0.004	0.61 ± 0.0087	0.53 ± 0.001
Ours (CNN)	0.62 ± 0.016	0.447 ± 0.009	14.67 ± 1.25	0.843 ± 0.024	$\mathbf{0.93 \pm 0.047}$	$\mathbf{0.675 \pm 0.009}$	0.599 ± 0.009

Table 4.11: Comparisons with the state of the art on Fashion-MNIST (averaged over 3 runs) with the CNN-based VAE model Mean±SD.

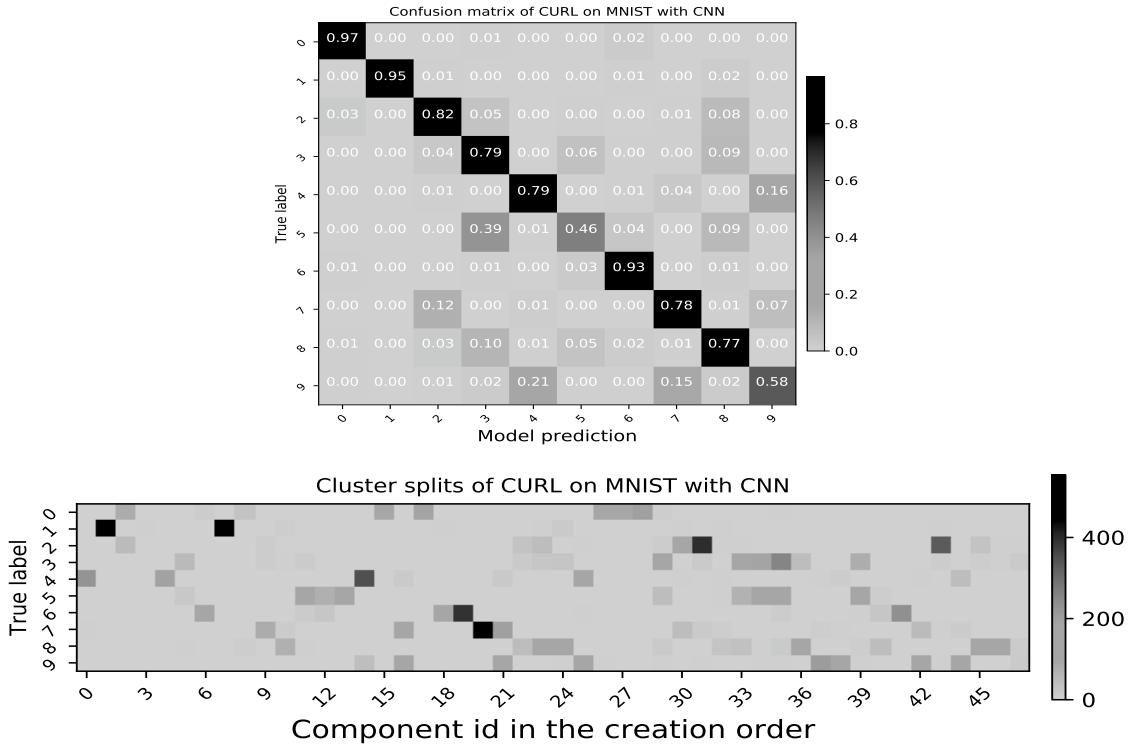


Figure 4.41: An example of composition of clusters on 1 run of CURL (bottom) and confusion matrix (top) using CNN for simple transition protocol on MNIST.

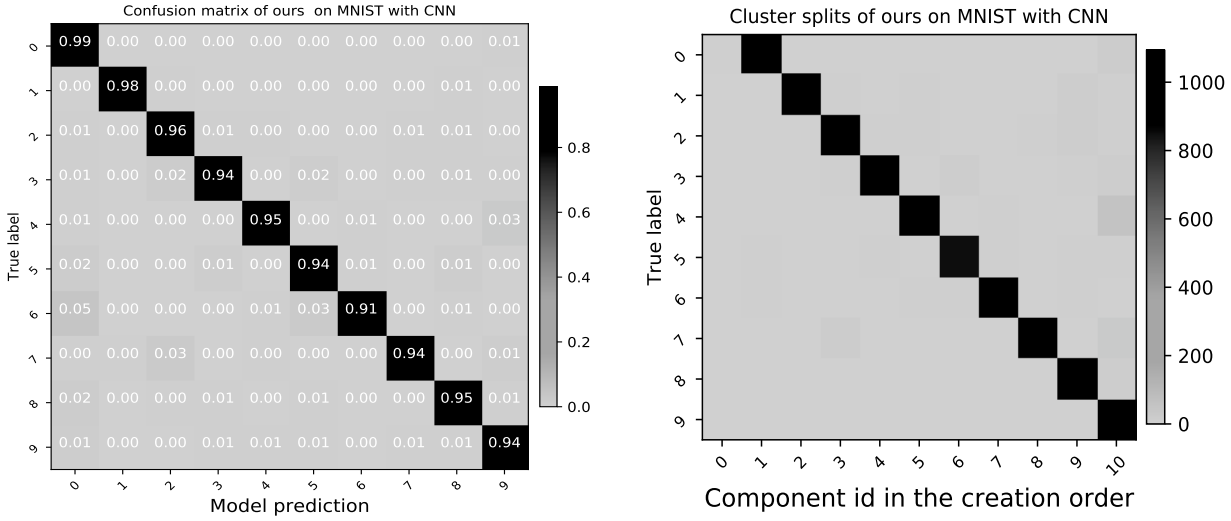


Figure 4.42: An example of composition of clusters on 1 run of our model (right) and confusion matrix (left) using CNN for simple transition protocol on MNIST.

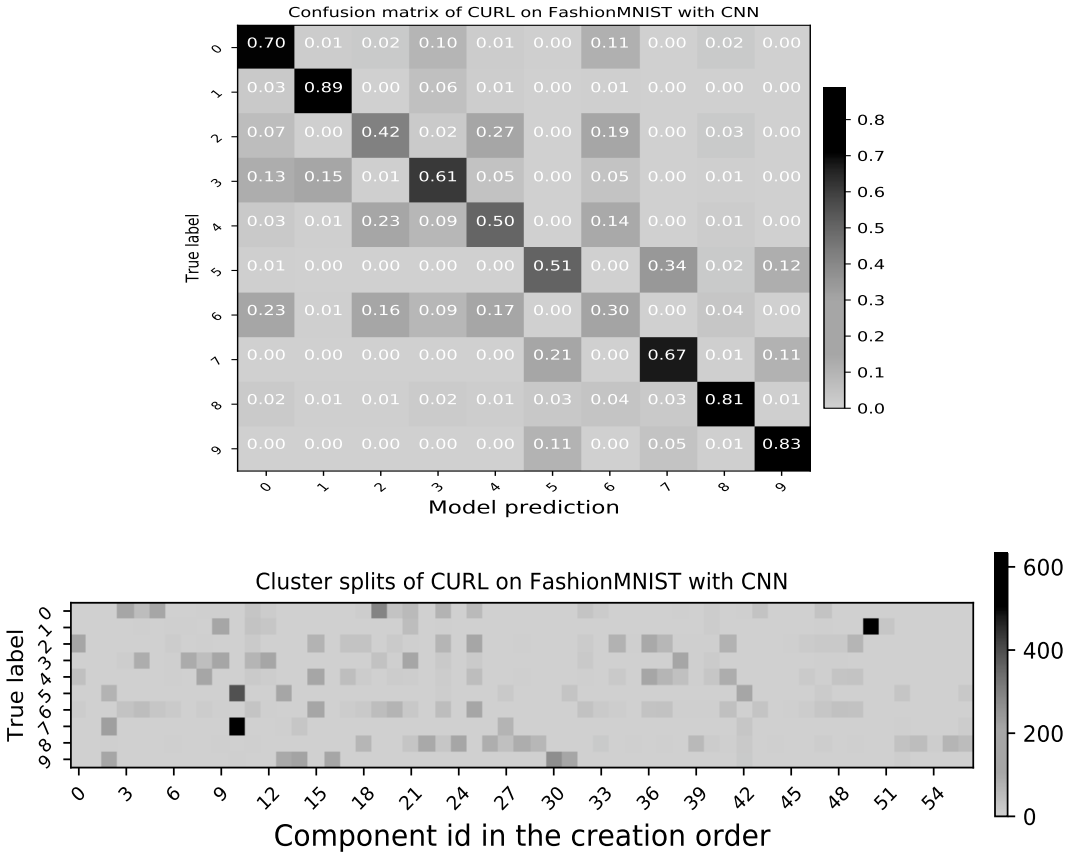


Figure 4.43: An example of composition of clusters on 1 run of CURL (bottom) and confusion matrix (top) using CNN for simple transition protocol on Fashion-MNIST.

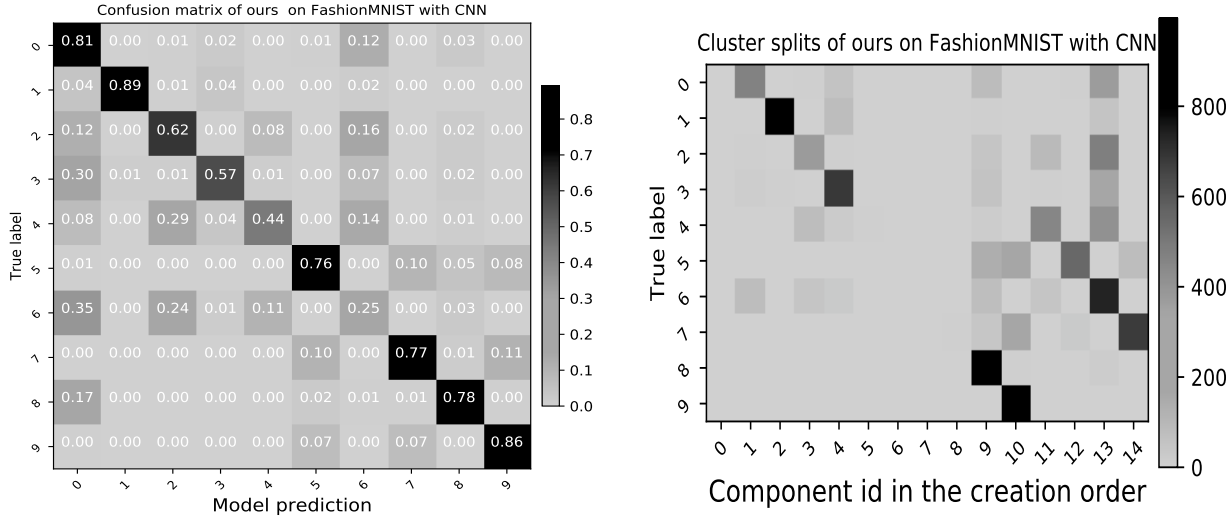


Figure 4.44: An example of composition of clusters on 1 run of our model (right) and confusion matrix (left) using CNN for simple transition protocol on Fashion-MNIST.

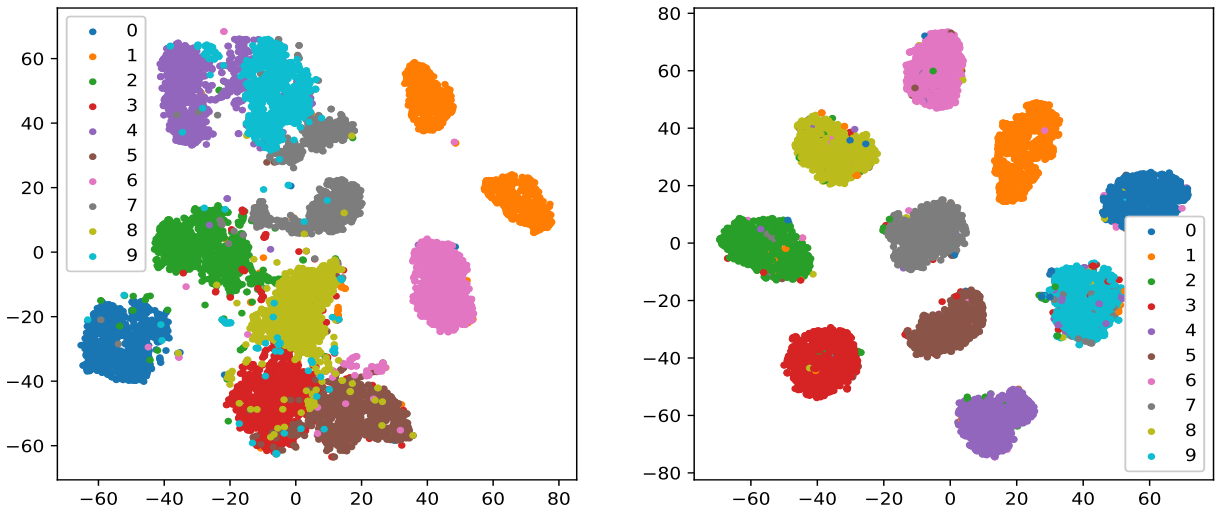


Figure 4.45: t-SNE projection of the embedding of CURL (left) and our model (right) on the MNIST dataset using CNN

For this reason, we only illustrate the composition of clusters of 1 run as an example for each model on each dataset. For our model, we remark that the confusion between similar categories ('2','4','6') on the Fashion-MNIST dataset persists in the composition of clusters, similar to when we used an MLP-based VAE. On the Fashion-MNIST dataset, the composition of clusters is not perfectly diagonal; meanwhile, "empty components" can also be observed. As we explained previously, our model optimizes ELBO loss with self-supervision, and it considers the hypothesis of 1 component per category. If the model creates excessive components, the category will be trained and optimized on the later created one, and the ancient one becomes a component of poor quality and is therefore empty or almost empty. And for the categories "4" to "7", new components are created during the review of objects, therefore, the total number of clusters is not exactly 10.

Clustering visualisation

We further show the t-SNE projections of latent variables on the MNIST dataset in figure 4.45 and on the Fashion-MNIST in figure 4.46. Similar results can be found as in the previous section using an MLP-based VAE. However, on the MNIST dataset, with the CNN-based model architecture, there are fewer confusions between different categories and the model creates a better clustering that is more robust.

In the following sections, we will only use the CNN-based VAE as the only model architecture for different experiments, since it is the architecture that allows better performance and enables better distinguishing of different categories.

Evolution of detection and classification accuracy

Similar to the previous section, in this section we illustrate by class detection and classification accuracy. This evolution illustrates the dynamics of the model's performance in both recognition and novelty detection. While local variations illustrate a backward transfer when a new class is integrated from learned to new classes and a forward transfer from learned to previously unlearned classes. Specifically, a decline in the by-class classification performance (compared to when the class was first learned) indicates forgetting.

We show their variations on the MNIST and Fashion-MNIST datasets, respectively, in figure 4.47 and in figure 4.48. Both models are not subject to catastrophic forgetting because they manage to maintain the classification accuracy globally compared to the performance when a class was first seen.

For the evolution of the detection accuracy, as one remark from the figure, CURL gets higher detection accuracy than before learning the classes. Therefore, the threshold used to determine poor modelled

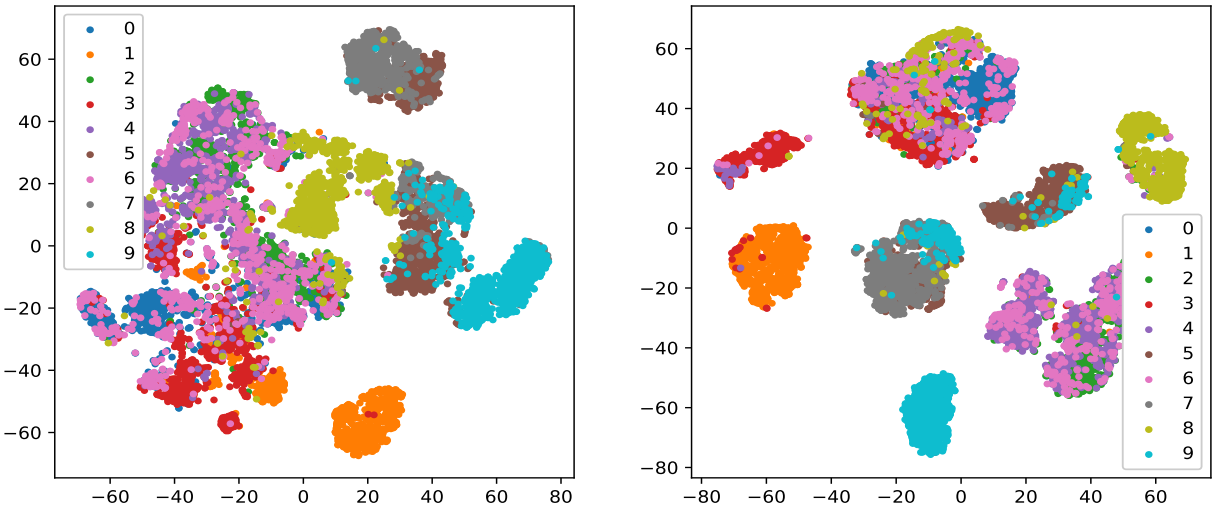


Figure 4.46: t-SNE projection of the embedding of CURL (left) and our model (right) on the Fashion-MNIST dataset using CNN

examples is actually lower than that would be used to filter out all the outliers or new class candidates, and for some new classes, even not yet seen, the ELBO loss exceeds the threshold. However, CURL creates a new category once its outlier buffer has been filled, even if there are some false negatives, a new category will still be created if there are enough outlier data to fill the buffer; but still, this can be a source of confusion and decrease recognition performance. Overall, our model achieves a higher detection accuracy for unseen categories, and it also better recognizes some already learnt object classes. CURL fails to recognize them from time to time and thus creates new components.

4.4.6.3 Influence of annotated examples

The dependence on the annotated examples to achieve comparable clustering performance is an important indicator. We illustrate for CURL and our model using the CNN-based VAE in the figure 4.49 and in the figure 4.50, the effort needed on both models for post-labelling on the MNIST and the Fashion-MNIST dataset. One can remark that with more components, CURL requires more annotated examples to finally reach its optimal classification accuracy during post-labelling.

4.4.7 Hard transitions

Previously, we have experimented with a simple protocol to demonstrate the performance of the model using two types of VAE model architectures, one using an MLP and another based on a CNN. From previous experiments, we observed that some categories are harder to separate by the model according to similarities defined by the confusion matrix. Based on this observation, in this section, we will in particular study if the setting where these similar categories follow each other, denoted as "hard transitions", challenges the effectiveness of novelty detection and recognition of the model in the continual learning scenario. We will show if the model manages to distinguish similar yet different categories and to detect novelty and use a CNN-based VAE, as it is the model architecture with the best clustering performance.

4.4.7.1 Hard sequences

We create a second protocol called the "hard-transition protocol". It consists of deliberately introduced hard transitions, in which similar categories are presented one after another. Like in the simple transition protocol, the learning sequences are composed of 2 phases, the detection of new categories and the recognition of learned ones, and the examples of the same category are presented multiple times.

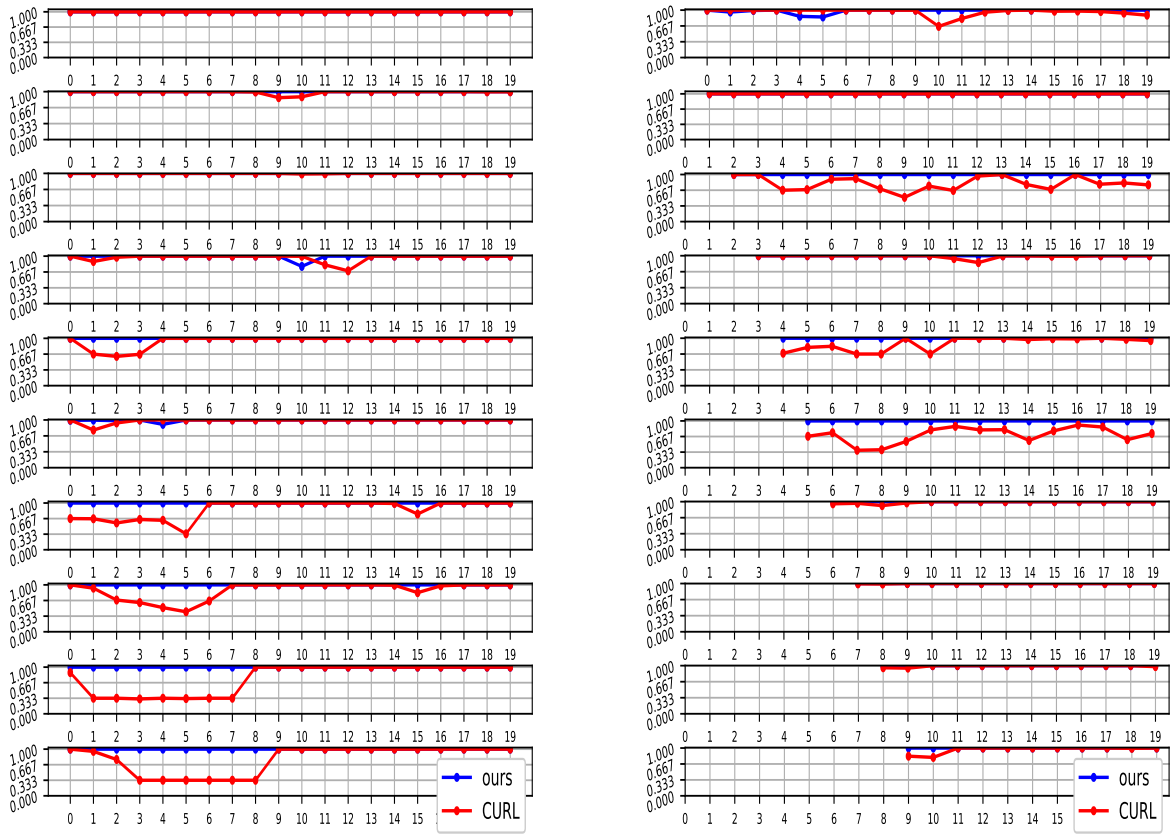


Figure 4.47: Comparison between CURL and our approach for the evolution of performance in detection (left) and by-batch classification (right) for MNIST using a CNN-based VAE.

We define similar categories as those that are hard to separate and show the most confusion with each other, derived from the composition of clusters of previous experiments in figure 4.41- 4.42, figure 4.43 - figure 4.43 and also from the preliminary studies shown in section 4.2. Typically, the categories '4', '7' and '9' are considered similar in MNIST, and the categories '2', '4' and '6' are considered similar in Fashion-MNIST, where we observed the mix between certain classes. Similar to *hard-transition protocol wo obj review* (explained in the previous chapter using the Page-Hinckley test), In this way, we will show whether the Hotelling t-squared test allows the separation between statistically close categories. For other details on how the hard transition protocol is constructed, see section 4.4.5.1. We hereby list 3 sequences as in table 4.12 that we test for MNIST and Fashion-MNIST.

Regarding other settings of the learning sequence, the number of iterations in each training period, we continue to use the same setting as in the *simple transition protocol*.

4.4.7.2 Results

We use the CNN-based VAE for our model in the hard-transition protocol, with which we have obtained the best result on the simple transition protocol and corresponds to the most frequent model architecture in the literature. In the following, we use the same CNN as in the previous section, and we use the same hyperparameters as in the simple transition protocol - as can be optimal for the trade-off

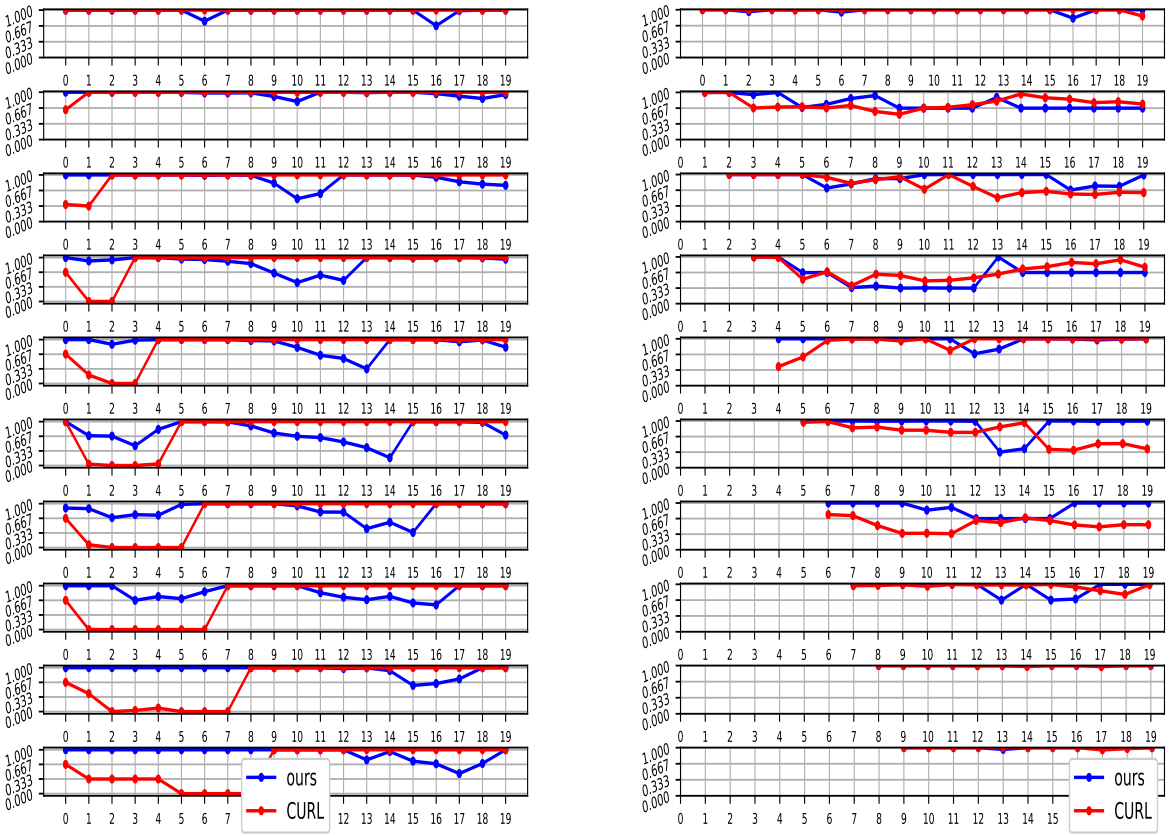


Figure 4.48: Comparison between CURL and our approach for the evolution of performance in detection (left) and by-batch classification (right) for Fashion-MNIST using a CNN-based VAE.

seq	phase	MNIST										Fashion-MNIST									
1	phase 1	7	1	9	4	3	5	2	0	8	6	2	4	6	7	5	9	8	1	0	3
	phase 2	7	1	9	4	3	5	2	0	8	6	2	4	6	7	5	9	8	1	0	3
2	phase 1	5	3	2	9	7	4	8	0	6	1	5	7	9	6	4	2	1	0	3	8
	phase 2	5	3	2	9	7	4	8	0	6	1	5	7	9	6	4	2	1	0	3	8
3	phase 1	8	0	6	5	3	2	9	7	4	1	1	0	3	2	4	6	7	5	9	8
	phase 2	8	0	6	5	3	2	9	7	4	1	1	0	3	2	4	6	7	5	9	8

Table 4.12: Tested sequences with the hard-transition protocol.

calculated by the mean classification accuracy and ARI score. Other models are not illustrated here, as our goal is to demonstrate that our approach using a CNN-based VAE model with the Hotelling t-squared test is also able to detect hard transitions. Since the hard-transition protocol is a harder scenario created from similar categories confused in the simple transition protocol, other models were studied only in the simple transition protocol.

On the MNIST and Fashion-MNIST, the results are shown in table 4.13 and table 4.14. We also illustrate the confusion matrix of CURL and our approach in figure 4.51- 4.53 on MNIST and in figure 4.54-4.56 on the Fashion-MNIST dataset.

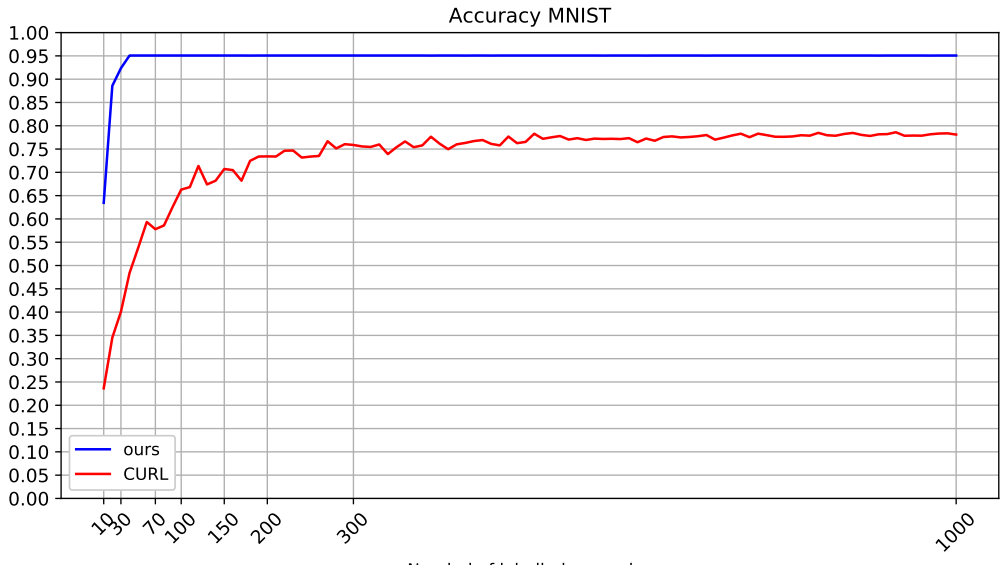


Figure 4.49: The influence of the number of annotated examples using a CNN-based model on MNIST.

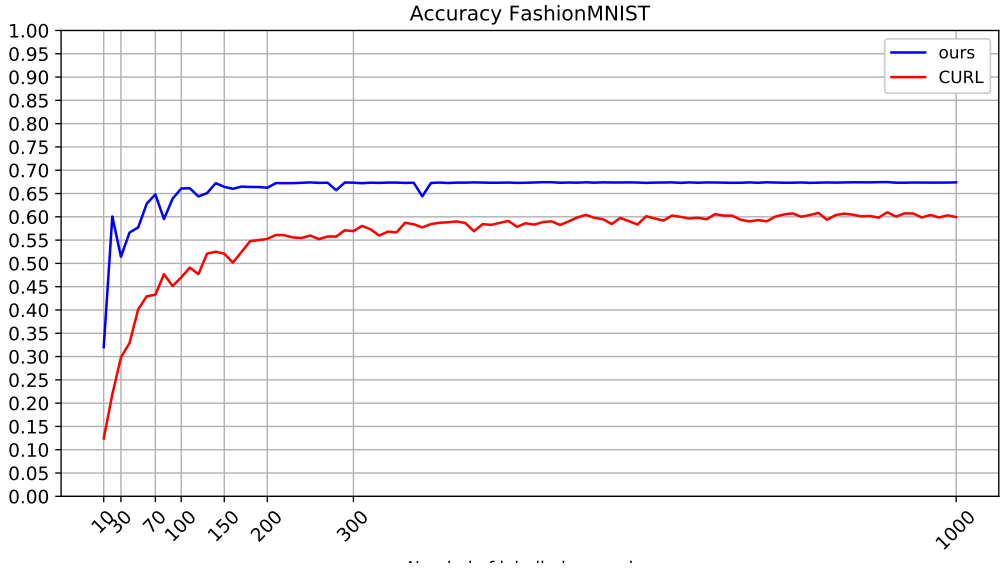


Figure 4.50: The influence of the number of annotated examples using a CNN-based model on Fashion-MNIST.

For the hard transition protocol, our model outperforms CURL in AMI and ARI scores on both MNIST and Fashion-MNIST datasets. Our model can create a number of components that is close to the number of categories in the dataset, while achieving better classification accuracy on both datasets. However, on the Fashion-MNIST dataset, CURL using a CNN outperforms our method in terms of by-batch classification accuracy. As discussed in the previous section and illustrated in figure 4.54 in figure 4.56, the creation of more components can improve the classification accuracy when combined with post-labelling of the majority vote. Subdivisions of clusters or over-segmentation lead to higher

model	AMI	ARI	# components	accuracy (batch)	accuracy(instance)	Homogeneity
CURL (CNN)	0.587 ± 0.006	0.296 ± 0.026	53.7 ± 6.13	0.97 ± 0.02	0.806 ± 0.022	0.778 ± 0.02
SOINN with SIFT	0.55 ± 0.007	0.15 ± 0.0078	131.67 ± 7.76	$\mathbf{1.0 \pm 0.0}$	0.89 ± 0.015	0.85 ± 0.0142
STAM [154]	0.806 ± 0.003	0.733 ± 0.002	$\mathbf{11}(*)$	0.89 ± 0.0	0.8433 ± 0.001	0.806 ± 0.003
Ours (CNN)	$\mathbf{0.8824 \pm 0.004}$	$\mathbf{0.89 \pm 0.005}$	$\mathbf{11.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{0.95 \pm 0.002}$	$\mathbf{0.88 \pm 0.005}$

Table 4.13: Comparisons with the state of the art on MNIST using the hard transition protocol and CNN (averaged over 3 runs) Mean±SD

model	AMI	ARI	# components	accuracy(batch)	accuracy(batch HT)	accuracy (instance)	Homogeneity
CURL (CNN)	0.47 ± 0.01	0.24 ± 0.01	54.33 ± 2.49	0.87 ± 0.0	-	0.62 ± 0.02	0.6 ± 0.02
SOINN with SIFT	0.46 ± 0.01	0.15 ± 0.01	81.67 ± 0.47	$\mathbf{0.9 \pm 0.0}$	-	$\mathbf{0.69 \pm 0.01}$	$\mathbf{0.64 \pm 0.01}$
STAM [154]	$\mathbf{0.66 \pm 0.006}$	$\mathbf{0.489 \pm 0.014}$	$\mathbf{9}(*)$	0.736 ± 0.045	-	0.66 ± 0.016	0.624 ± 0.006
Ours (CNN)	0.628 ± 0.006	$\mathbf{0.468 \pm 0.028}$	$\mathbf{12.6 \pm 1.25}$	0.832 ± 0.047	0.833 ± 0.047	0.66 ± 0.019	0.59 ± 0.018

Table 4.14: Comparisons with the state of the art on Fashion-MNIST using the hard transition protocol and CNN (averaged over 3 runs) Mean±SD

homogeneity within the cluster. Thus the chances of affecting the cluster to its true corresponding category increases, decreasing the possibility of confusion between categories.

4.4.7.3 An illustration of the evolution of pairwise distances between different components

In this section, we illustrate the pairwise distance between the online estimation of the mean over different categories and show how it evolves over time. As each component is created, when a new category is detected, the change in Euclidean distance between the mean estimated over categories will show the level of separation between them. Ideally, the euclidean distance should be maximized when it concerns different categories, and it could be considered as a measure of similarity. The calculation of the pairwise distance matrix is inspired by [154]. In particular, it will help us to compare learning in the simple transition protocol and the hard-transition protocol, and the impact of learning in these circumstances.

We illustrate the evolution of the pairwise distance matrix in a timely order in figure 4.59 on the MNIST dataset and in figure 4.60 on the Fashion-MNIST dataset for the simple transition protocol and for the hard transition protocol. The diagonal entries of the matrix contain only zeros and will not be studied, and the matrix is symmetric. Each matrix is calculated every 2000 steps, making 10000 steps each line from left to right; And this covers the end of learning.

During training, when we consider that a category is represented by a single component, ideally the pairwise Euclidean distance between different components should be large (the pairs with maximum distance are represented by lighter cells). A darker cell indicates that the distance between two estimations of means are close, which should happen as little as possible since this suggests confusion between two categories. However, many dark cells could be observed in the case of the Fashion-MNIST dataset due to the confusion of certain similar categories, for example, '2', '4' and '6'. Interestingly, in the Fashion-MNIST dataset, the position of dark cells has formed a sort of mosaic, but on the hard transition protocol, they lie in adjacent positions, because we have presented these similar categories one after the other.

For both protocols on both datasets, the distances between classes become larger (the cells become slightly brighter) as the model optimizes and learns different examples over time if one compares the color of a cell in its early stage (that lies on the left) with that of its color when trained over a certain time. This could be caused by the initialization of the component, as each time a component is created, the initialization copies the existing one that shares the most similarity with it; this also suggests that for the hard transition protocol, the initialization of a component is more likely to be originated from the one that is adjacent to it, when the model has seen similar categories. After training for a while, the brightness of the cell increases due to the model's learning. This suggests that the distances between the two components are larger and they are better separated.

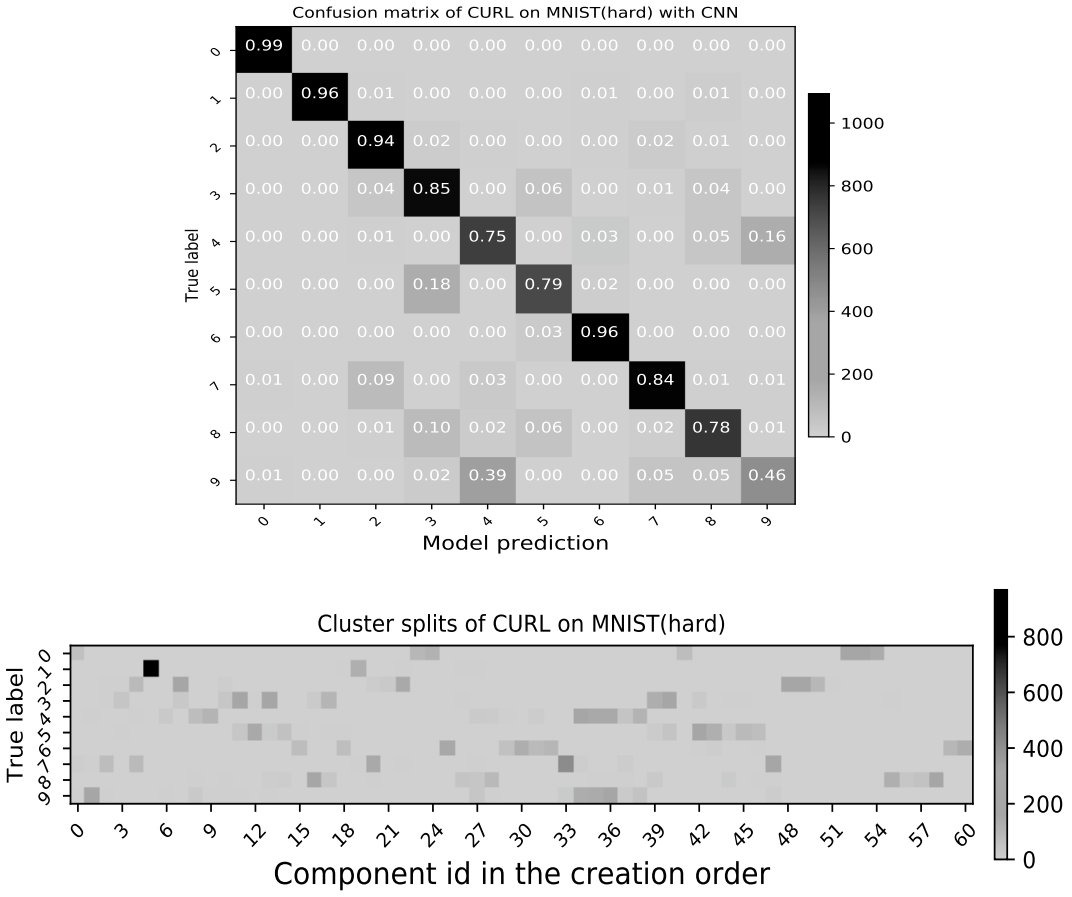


Figure 4.51: An example of the composition of clusters on 1 run (seq 1) of CURL (bottom) and confusion matrix (top) using CNN for the hard transition protocol on MNIST.

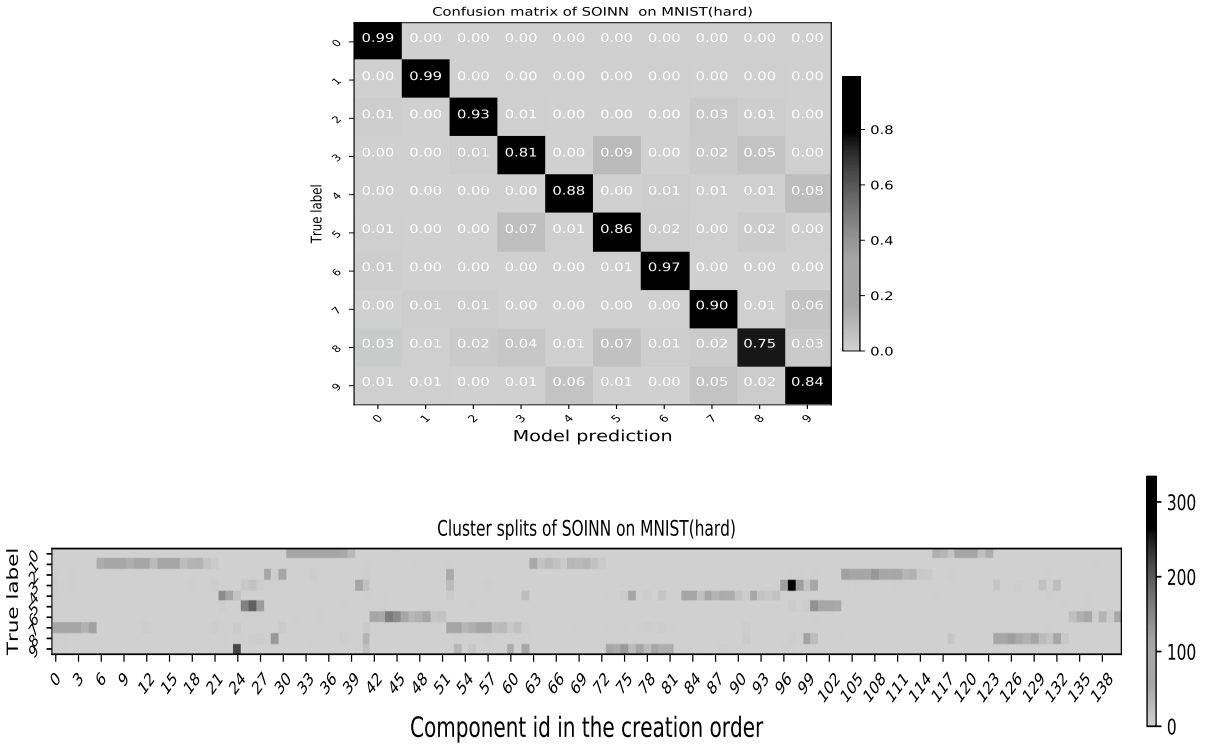


Figure 4.52: An example of the composition of clusters on 1 run (seq 1) of SOINN (bottom) and confusion matrix (top)for the hard transition protocol on MNIST.

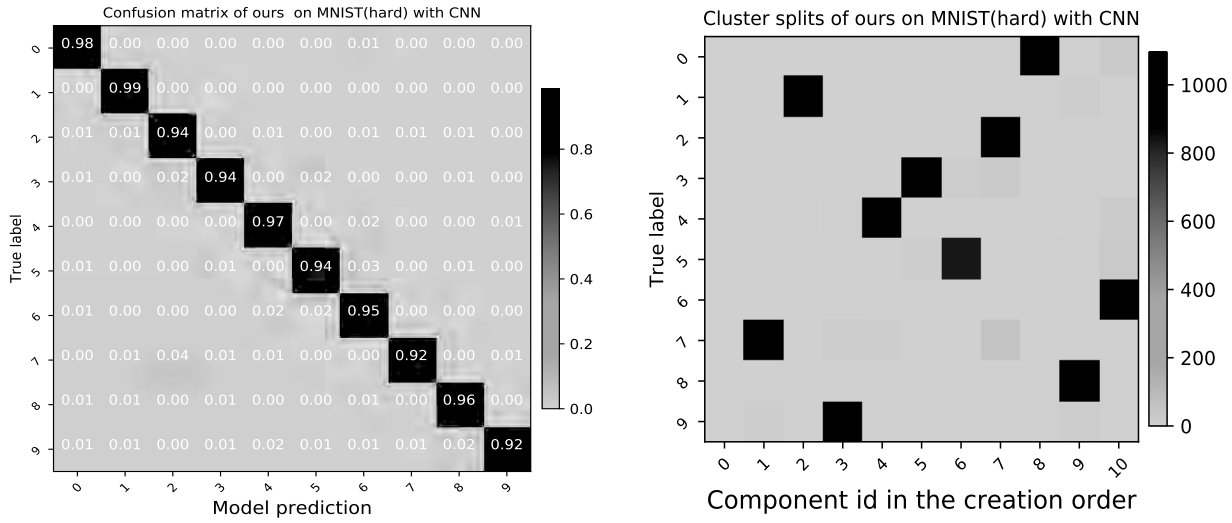


Figure 4.53: An example of the composition of clusters on 1 run (seq 1) of ours (right) and confusion matrix (left) using CNN for the hard transition protocol on MNIST.

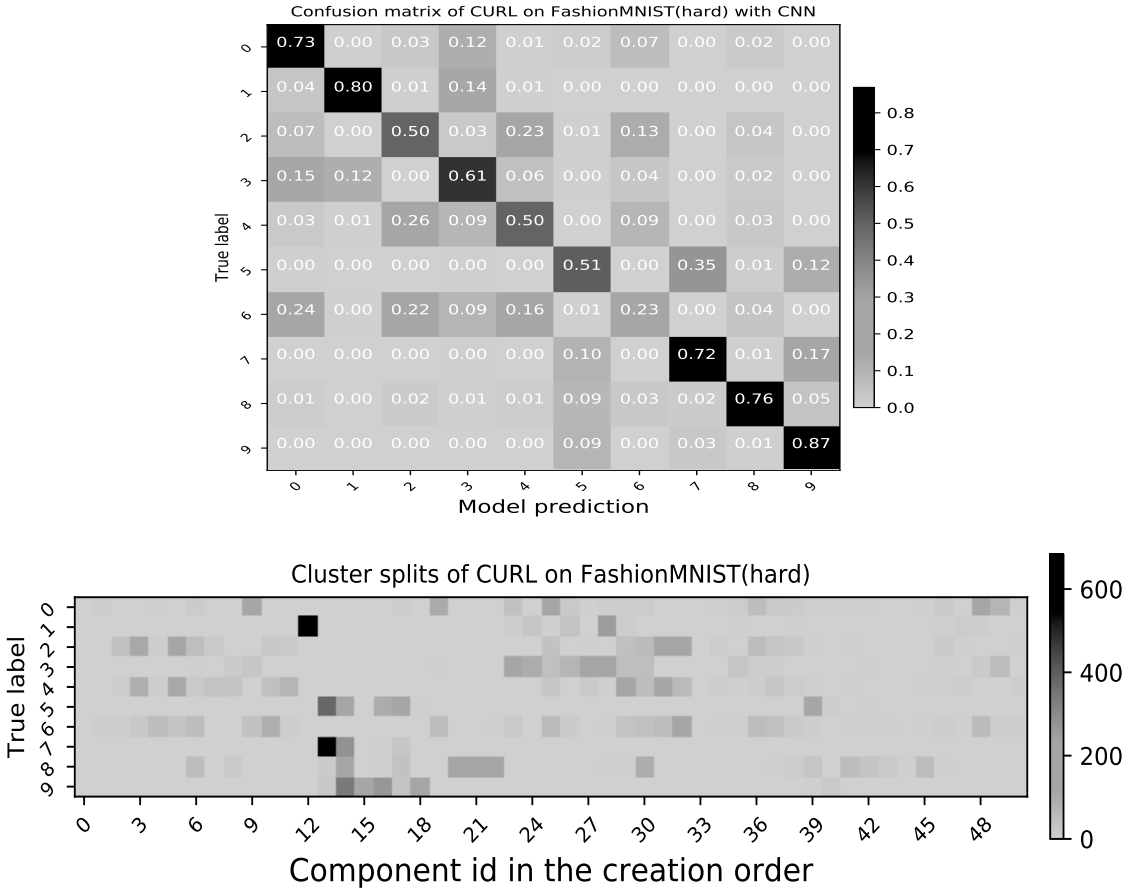


Figure 4.54: An example of the composition of clusters on 1 run (seq 1) of CURL (bottom) and confusion matrix (top) using CNN for the hard transition protocol on Fashion-MNIST.

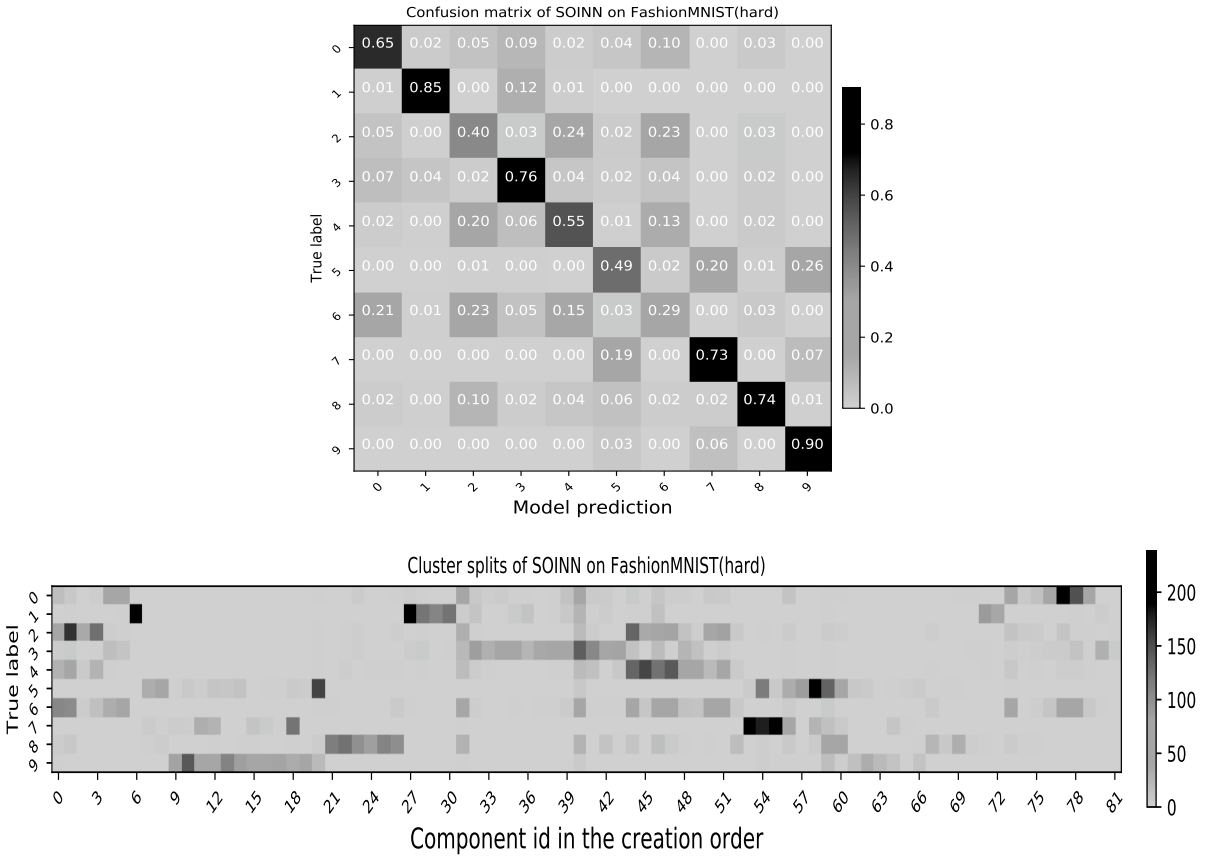


Figure 4.55: An example of the composition of clusters on 1 run (seq 1) of SOINN (bottom) and confusion matrix (top) for the hard transition protocol on Fashion-MNIST.

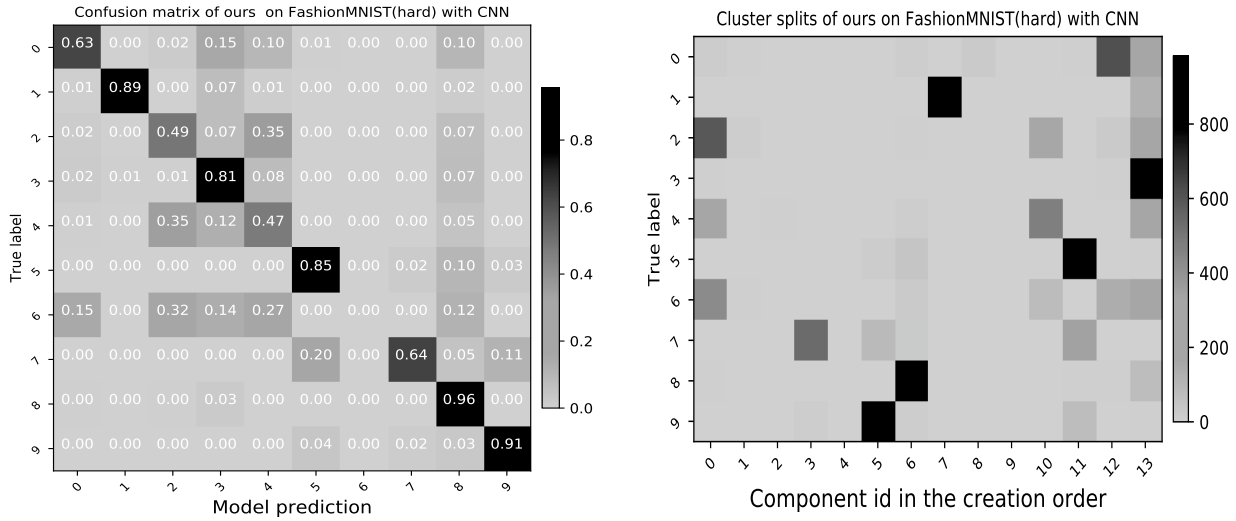


Figure 4.56: An example of the composition of clusters on 1 run (seq 1) of our model (right) and confusion matrix (left) using CNN for the hard transition protocol on Fashion-MNIST.

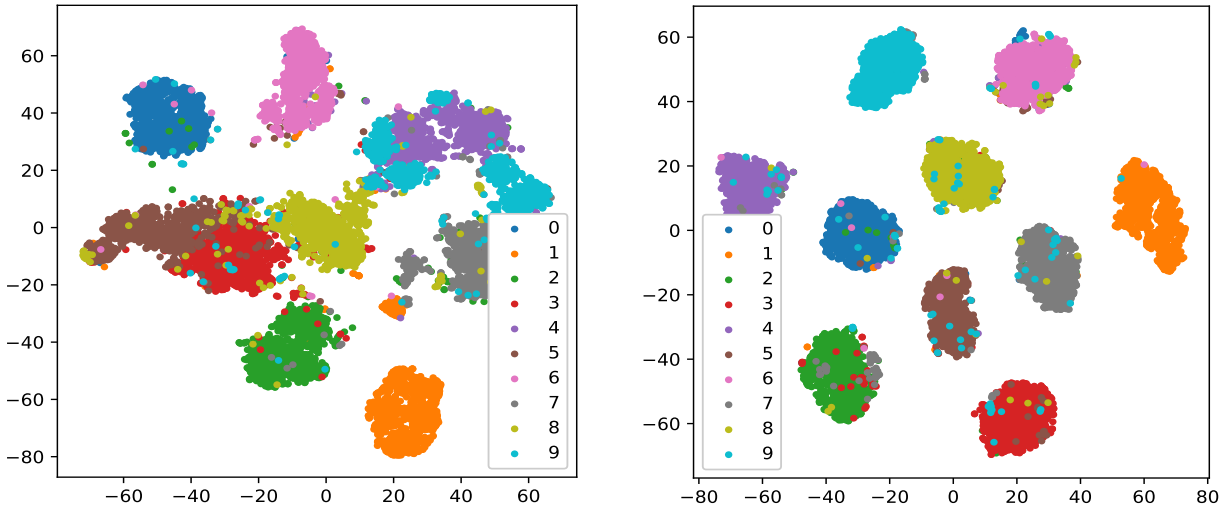


Figure 4.57: t-SNE projection of latent variables on 1 run (seq 1) of CURL (above) and our model (below) using CNN for the hard transition protocol on MNIST.

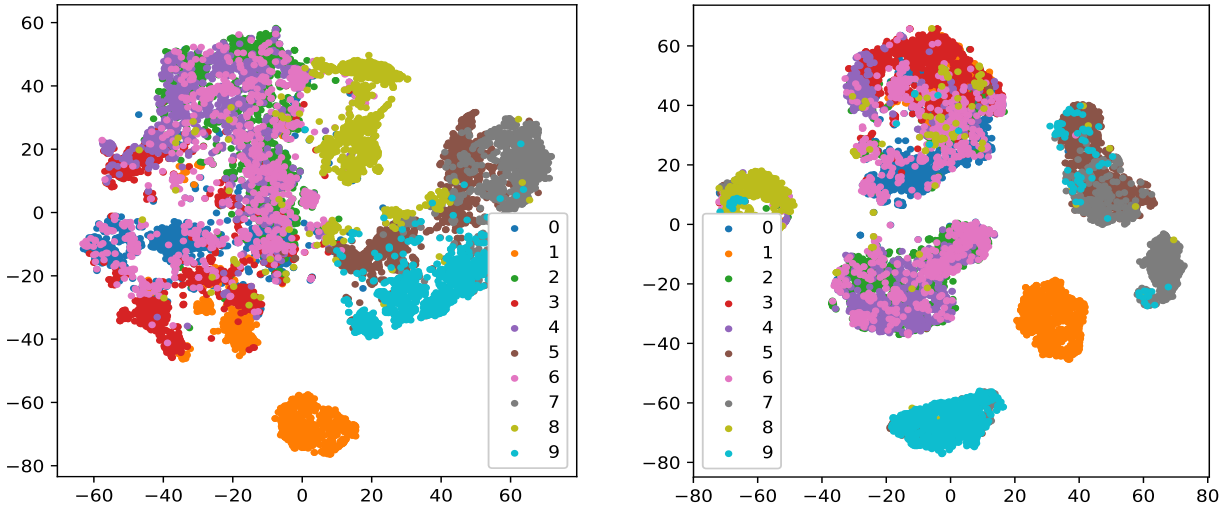


Figure 4.58: t-SNE projection of latent variables CURL (above) and our model (below) using CNN for the hard transition protocol on Fashion-MNIST.

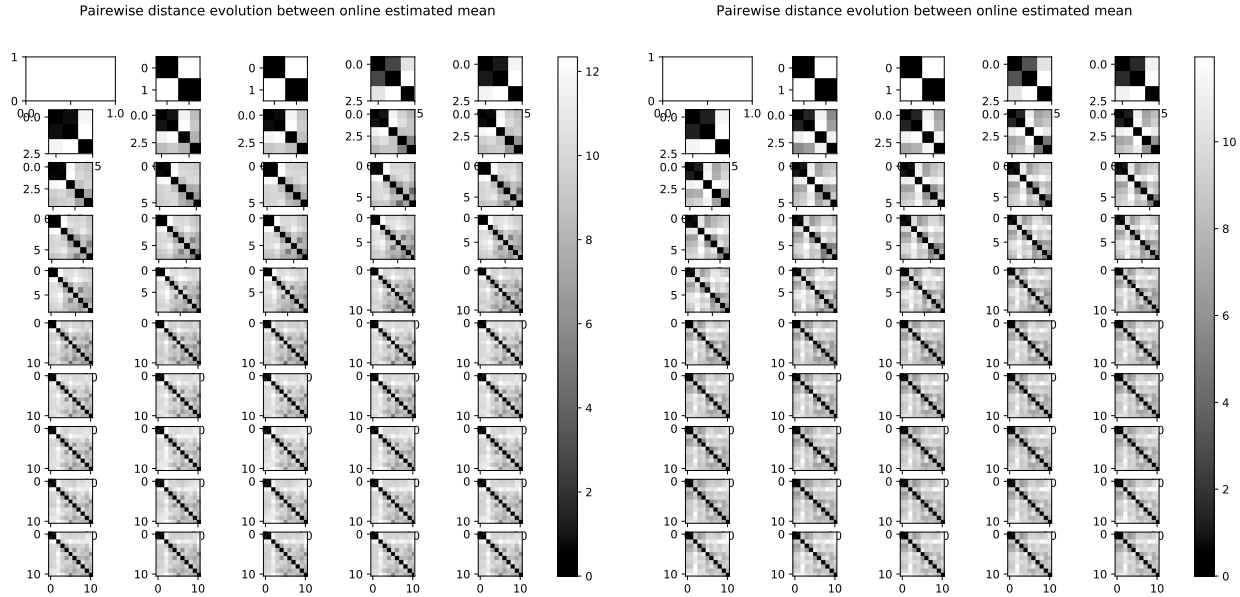


Figure 4.59: The evolution of similarity defined by pairwise distance between different means of components estimated online, to compare the evolution of similarities between a run on the simple protocol seq 1 (left) and hard protocol seq 1 (right) on the MNIST dataset.

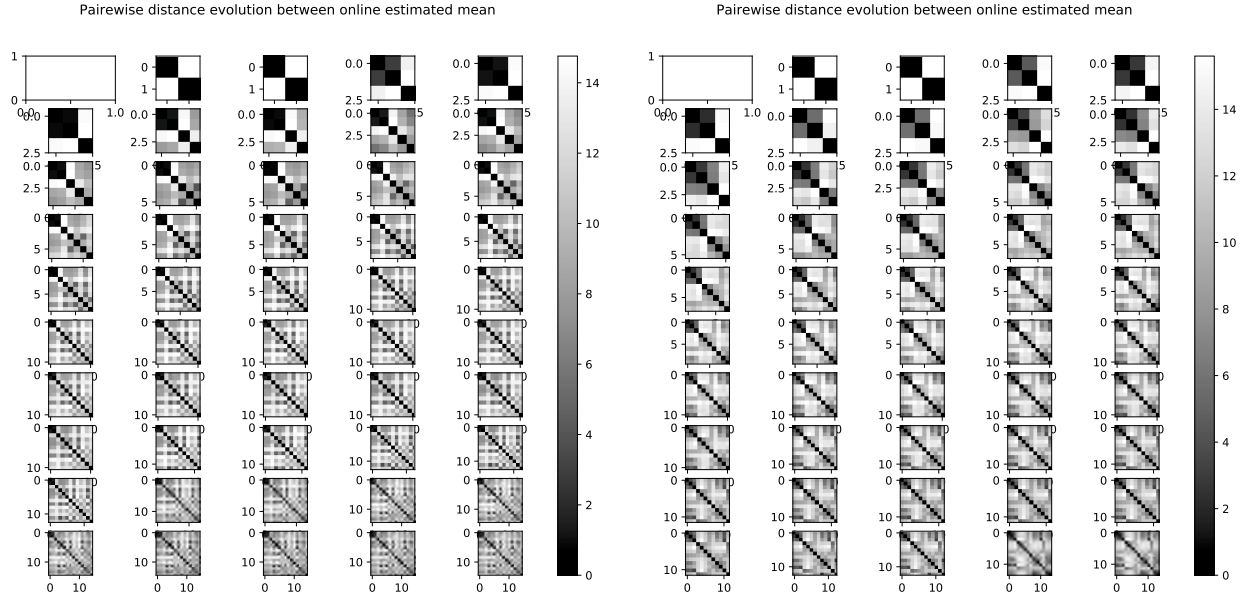


Figure 4.60: The evolution of similarity defined by the pairwise distance between different means of components estimated online, to compare the evolution of similarities, between a run on the simple protocol seq 1 (left) and hard protocol seq 1 (right) on the Fashion-MNIST dataset.

4.4.7.4 Evolution of detection and classification accuracy

We further illustrate the evolution of detection and classification accuracy, similar to that shown by the simple transition protocol with the MLP-based VAE in section 4.4.5.6. We show the evolution of detection and classification accuracy for our model and CURL on the hard-transition protocol in figure 4.61 on the

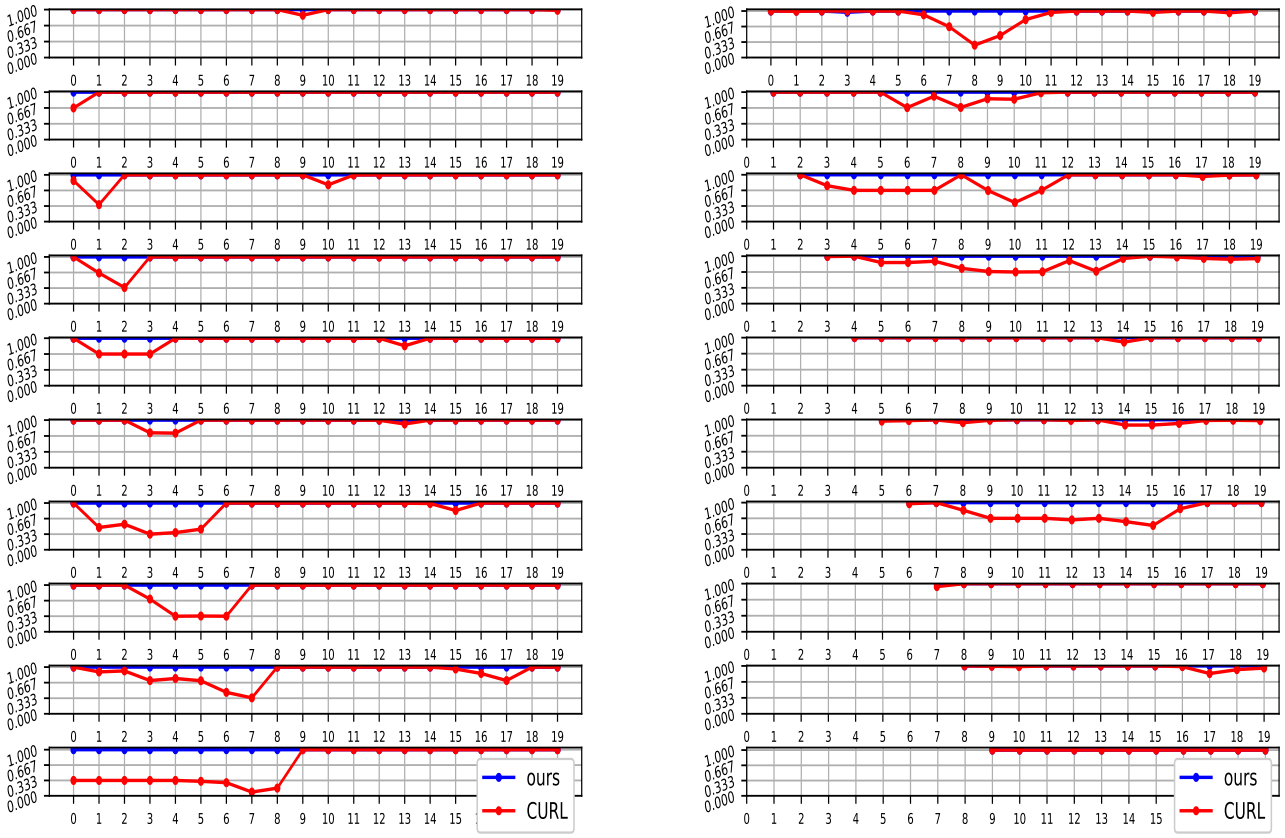


Figure 4.61: Comparison between CURL and ours for evolution of performance in detection (left) and classification by batch (right) for hard sequences, using the CNN model architecture for MNIST; each single line corresponds to the evolution computed by class at each ground truth category change moment.

MNIST dataset and in figure 4.62 on the Fashion-MNIST dataset. With hard transitions that present similar objects one after the other, our model can properly detect transitions on MNIST, yet more detection errors are found both for our model and for CURL on the Fashion-MNIST dataset.

Previously, we have studied the performance of our model and CURL on the dataset of MNIST and Fashion-MNIST, on two protocols, simple transition protocol and hard-transition protocol, a difficult protocol with hard transitions. However, on both MNIST and Fashion-MNIST datasets, there are only 10 classes; in the context of an autonomous agent, it can be exposed to a more complicated environment with more object categories. Therefore, we will study the performance of different models on a third dataset, EMNIST, with 47 categories instead of 10, to show the impact of this increase. We continue to use the previous model structure, the CNN-based VAE, since it was the optimum architecture with which we have obtained the optimal clustering performance.

4.4.8 Increasing the number of classes

In this section, we discuss the impact of an increase in the number of classes on the performance of the model. We use the Extended MNIST (EMNIST) dataset, which contains 47 categories. We compare

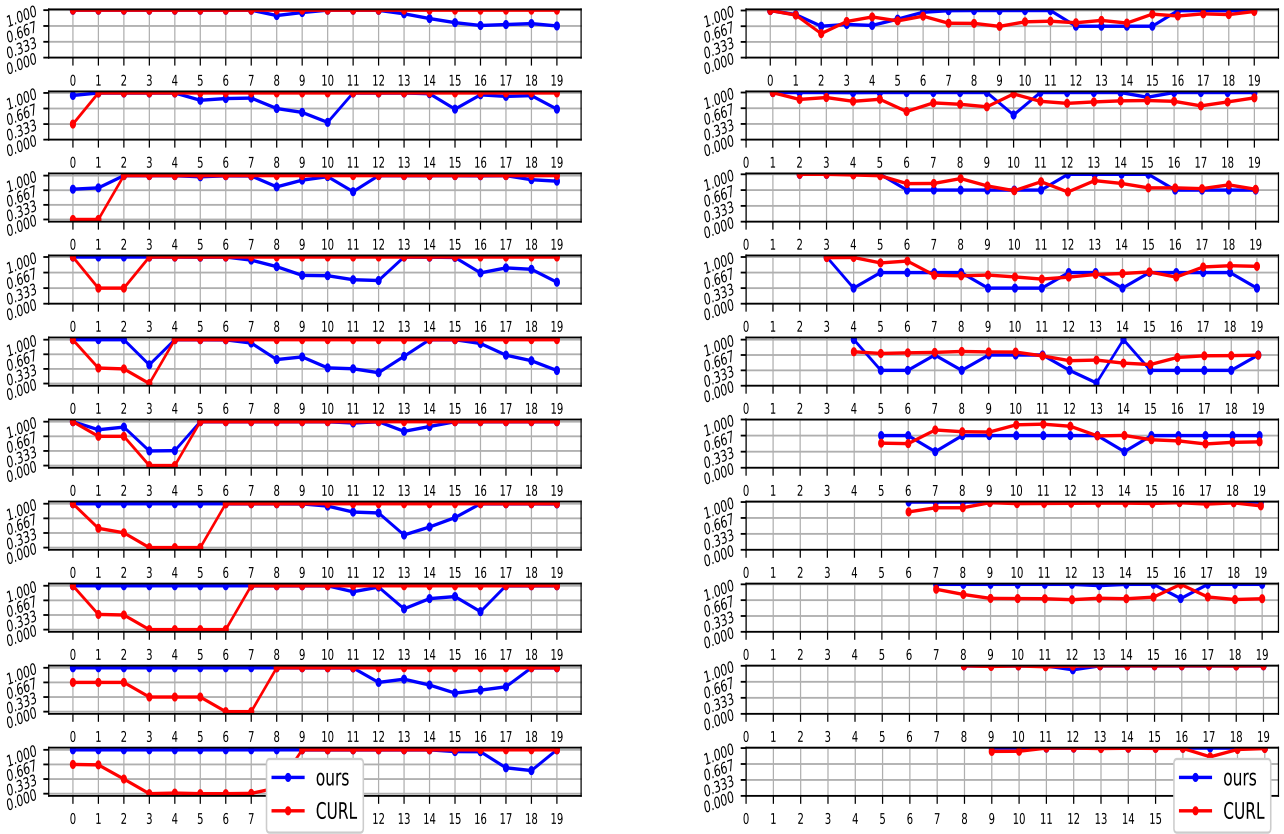


Figure 4.62: Comparison between CURL and ours for evolution of performance in detection (left) and classification by batch (right) for hard sequences, using the CNN model architecture for Fashion-MNIST; each single line corresponds to the evolution computed by class at each ground truth category change moment.

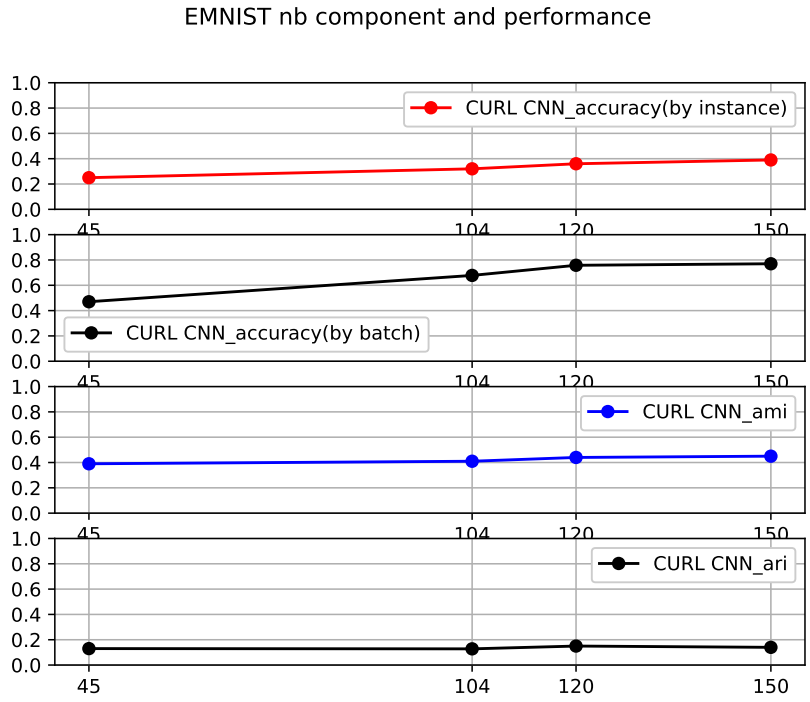


Figure 4.63: The impact of the number of components on the clustering performance of CURL on EMNIST, we have gradually varied the number of classes to learn by taking the first n classes, to evaluate their learning performances.

our model with CURL using a CNN-based VAE, which is among the best models we have discussed in the previous sections. We first give a detailed illustration of the variation of performance with a different number of classes (learn only several classes) to demonstrate the performance of comparable models, ours and CURL on the entire dataset with 47 categories. We continue to use the same CNN as in the previous section. We have tested several values of the threshold for ELBO loss of CURL as shown in table 4.15.

model	tested threshold EMNIST
CURL (ELBO θ)	-150 -200 -250

Table 4.15: Tested thresholds to create different number of components for CURL.

The relationship between the number of components and the clustering performance is shown in figure 4.63. We can observe that the performance of CURL is less sensitive to the number of components on the EMNIST dataset. For this reason, we only tested a limited number of points on EMNIST, and the upper and lower bounds are more constrained due to more categories presented in EMNIST. Since there are 47 categories, when the model creates fewer categories than 47, this indicates confusion between different classes; on the other hand, we also fix the upper bound of the number of components created during training to 150, to avoid too much creation of components during training, which is both time and resource consuming.

At 150 components, the optimum accuracy and the ARI score are reached, while the accuracy increases as the number of components increases. Although this seems contrary to what we have observed on the MNIST and Fashion-MNIST dataset, the EMNIST is a more complex dataset from the point of view that there are actually more categories in the dataset than in the MNIST and in the Fashion-MNIST dataset. For the ARI score, there are two possible sources of penalization: too many "pairs" that are classified by

error or over-segmentation that separates pairs in the same category into different categories. In the case of too few components, the model has hardly modeled correctly the distributions of different categories, with low classification accuracy, the model has also obtained low ARI score since there are too many "pairs" of outliers while using the ARI evaluation metric.

4.4.8.1 Optimal hyperparameter choices

Similar to previous experiments on MNIST and Fashion-MNIST, we choose the threshold that allows for better AMI and ARI scores while maintaining the classification accuracy. For the ELBO threshold for CURL, we choose to use $\theta = -150$ to allow the best trade-off between the number of components and clustering performance, getting the best AMI, ARI scores, and classification accuracy.

dataset	iter/period	model	θ (ELBO)	n_y(HT)	n_b(HT)
EMNIST	1500	ours (HT)	$-$	10	100
		CURL	-150	-	-

Table 4.16: For the models with Hotelling t-squared test in the scenario with review of objects with our *CNN-based VAE*. We show the threshold for ELBO loss θ used in each model, and the hyperparameters for the Hotelling t-squared test. The iter/period corresponds to the number of iterations during which each category is trained.

For the choice of other hyperparameters, the choice of n_y and α_{ht} is re-determined empirically on the EMNIST dataset. Since there are 47 categories in EMNIST, for a given total number of training iterations, it will be split into 47 categories in the continual learning scenario. Therefore, for each category, there are fewer training examples during fewer iterations. α_{ht} is related to the speed of update and the relevance of recent examples for the computation of running average, and n_y is the number of examples for training. we slightly decrease n_y and α_{ht}, taking n_y =10 for Hotelling t-squared test and $\alpha_{ht} = 0.95$ for the estimation of running average and running covariance. These two hyperparameters are decided empirically with regard to the Hotelling t-squared test's novelty detection and recognition performance.

4.4.8.2 Results

We will show the influence of number of object classes on the model performance on the simple transition protocol, by learning only a certain number of objects among all objects in the dataset. With n the number of classes presented among all in the learning sequence, by varying n classes, for example, taking n=$\{10, 20, 30, 47\}$, while fixing the other hyperparameters, for example, the order of the sequences and the number of iterations the model will be trained in each class. We illustrate the evolution of clustering performance as a function of the increase in the number of classes in figure 4.64. We can observe that the clustering performance decreases as the number of categories increases. When there are only 10 categories in the dataset, the performance of EMNIST on the simple transition protocol is comparable to that of MNIST on the simple transition protocol, while using CNN, but on the entire dataset of EMNIST with 47, the classification accuracy has decreased by half, while the decrease can also be observed in AMI and ARI scores. When the model sees the entire dataset with 47 categories, as it applies a generative replay strategy to alleviate catastrophic forgetting, it changes between real batches that are composed of images of the current categories, with generated batches that contain all past categories. At the end of the training, with 47 categories, the generated images will be split into categories from 0 to 46, leaving only a few generated images for each category. Therefore, it can be a possible reason for explaining the drop in accuracy, as the model only sees a few examples of previous categories.

In table 4.17, we also compare our model with CURL, SOINN and STAM on the entire dataset with 47 categories. Our model outperforms CURL and SOINN on AMI and ARI score and also has higher accuracy than CURL, with a slightly lower homogeneity score. Overall, the best results on this dataset were obtained by STAM, except the accuracy by batch compared to our approach using our proposed Hotelling t-squared test. Again, the number of clusters has been fixed manually to the number

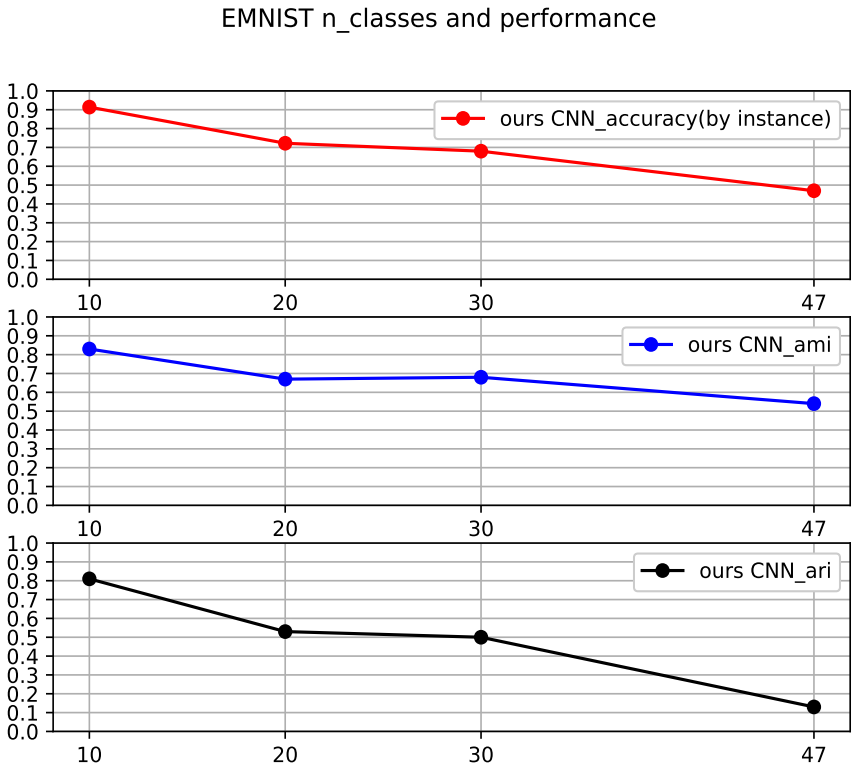


Figure 4.64: Study of scalability – the evolution of clustering performance as a function of increase in number of classes in the dataset of our (CNN-based) model on EMNIST dataset, learning limited classes: n=10 correspond to learning from class 0 to 9, n = 20 correspond to learning from 0 to 19, and n=47 the entire dataset.

model	AMI	ARI	# components	accuracy(batch)	accuracy(batch HT)	accuracy(instance)	homogeneity
CURL	0.443 ± 0.009	0.133 ± 0.009	150 ± 0.0	-	0.77 ± 0.024	0.383 ± 0.012	0.532 ± 0.008
SOINN SIFT	0.485 ± 0.008	0.153 ± 0.005	243 ± 12.0	-	$\mathbf{0.845 \pm 0.024}$	0.51 ± 0.015	$\mathbf{0.613 \pm 0.013}$
STAM [154]	$\mathbf{0.63 \pm 0.002}$	$\mathbf{0.384 \pm 0.004}$	**47**	-	0.743 ± 0.01	$\mathbf{0.55 \pm 0.005}$	$\mathbf{0.63 \pm 0.0006}$
Ours	0.54 ± 0.016	0.173 ± 0.026	76 ± 6.98	0.75 ± 0.03	0.788 ± 0.017	0.48 ± 0.007	0.515 ± 0.004

Table 4.17: Comparison with the state of the art on EMNIST (averaged over 3 runs) Mean±SD. (For STAM, the number of cluster is fixed manually to 47.)

.

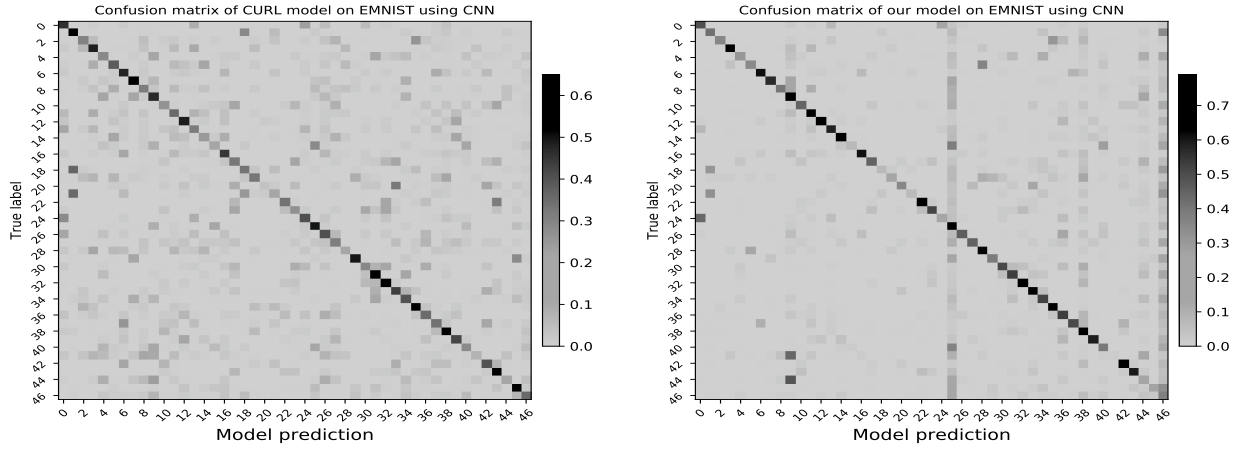


Figure 4.65: An example of the confusion matrix of clusters on 1 run for CURL (left) and ours (right) on EMNIST.

of ground truth classes (47) which may give it a slight advantage. Note that, here we did not optimize the hyperparameters to achieve the best compromise between ARI (or number of components) and accuracy. Thus, for our model, there may be a better configuration. But although there is a sensitivity to the number of classes, our approach globally outperforms the CURL and SOINN models. The decrease in CURL's performance could also be explained by reaching the maximum number of components 150 before the end of the training, as for the consistency of the model, we consider that the model does not expand infinitely. This can further be reflected by the difference between homogeneity score and the classification accuracy– during post-labelling we assign to each cluster its corresponding class by majority vote, and homogeneity score measures if the cluster contains only data from one single class, in the case of CURL, its homogeneity score higher than our model at the cost of over-segmentation during clustering. Although in the GMM modeled by the VAE, having more components enables better clustering performances in previous cases on MNIST and Fashion-MNIST, having more categories in the training set has actually made expansion of the model a bottleneck in the case of CURL. But our model is less constrained from this point of view while creating fewer components and reaching better clustering performances.

We also show the composition of clusters of CURL, our model and SOINN on the EMNIST dataset in figures 4.65 and 4.66. One can see that, globally, for our model, there is less confusion of classes than for CURL. It also seems better than SOINN, although the statistics in Table 4.17 show the opposite (i.e. a slightly higher overall by-batch and by-instance accuracy for SOINN). This may be due to a few clusters that "contain" a mixture of many classes (slightly darker columns in the confusion matrix). The prediction of CURL tends to disperse on different clusters, at the same time our model also creates a higher number of clusters than the real number of categories in the dataset. For our approach the prediction of each category is condensed into 1 cluster most of the time. Excessive components are created after approximately 24th cluster. This is partially due to the reduced capacity in recognition performances when the number of classes increases, as we have previously shown in figure 4.64.

The performance of the model is impacted by the number of classes during training and shows a decrease in clustering performance (classification accuracy) accompanied by a rise in component creation– it has also become more complicated to recognize a learned class when there are more classes and fewer real examples or generated examples for training, as mentioned earlier. In figure 4.67, we show the cumulative component count during training of our CNN-based model, i.e. the number of times each component has been predicted for the input data. One can see that the components created at the end are predicted half as often as those created in the first half as the most probable component. This is because they are created for a learned category that is reviewed for a second time. If there is already an existing component for this category, yet the model still created a new one, the component count could be divided into both components and are smaller than a recognized category. Consequently, this

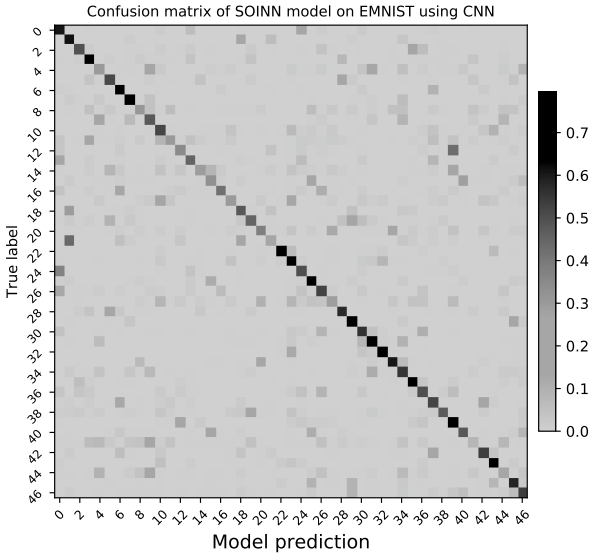


Figure 4.66: An example of the confusion matrix on 1 run for SOINN on EMNIST.

may affect how images are generated and the clustering performance. Ideally one should get a uniformly distributed prior. For our model, the distribution appears uniform only at the beginning of the learning process when the model has only seen a few classes, but the bar plots indicate the components created at the end of the training are unbalanced with regard to those created at the beginning.

Besides the impact of the number of classes, other more complicated cases may occur. The autonomous agent can perceive sequences of objects in which classes arrive in less structured orders. This motivated us to study another protocol in the next section, in which we imagine a more complicated scenario.

4.4.9 Mixing detection and recognition

In the context of an autonomous agent perceiving objects in an unknown environment, it will not always be able to anticipate when a learned class will reappear, since, for example, the objects may themselves be moving away and reenter its field of view. Although, in theory, it can interact with its environment, the agent may sometimes have limited control over the structure of the sequence. For this reason, we will consider a case where the review of (some) objects happens before presenting all the new objects, i.e. we do not divide the whole sequence into one detection and one recognition period of all classes but have several such periods with only a part of the available object classes. We define this protocol in more detail in the following section.

4.4.9.1 Mixed protocol

As an incremental step, instead of a single detection and recognition cycle, here, we define two phases, with K being the total number of object classes:

1. *Phase I*: Learning and recognizing n classes. During this learning stage, learning only n classes among all new classes and present these classes once again to the agent to study if the agent can recognize them.

2. *Phase II*: Then show the rest of the $K - n$ classes that are novel to the agent and repeat them for recognition

For example, on EMNIST, a possible sequence order can be learning 23 classes- recognition of 23 classes - learning 24 new classes - recognition of 24 classes, which leads to a learning sequence in the order of classes $\{0, 1, 2, 3, 4, ...22|0, 1, 2, 3, 4, ..., 22|23, 24...46|23, 24...46|\}$. That means, the model first sees 23

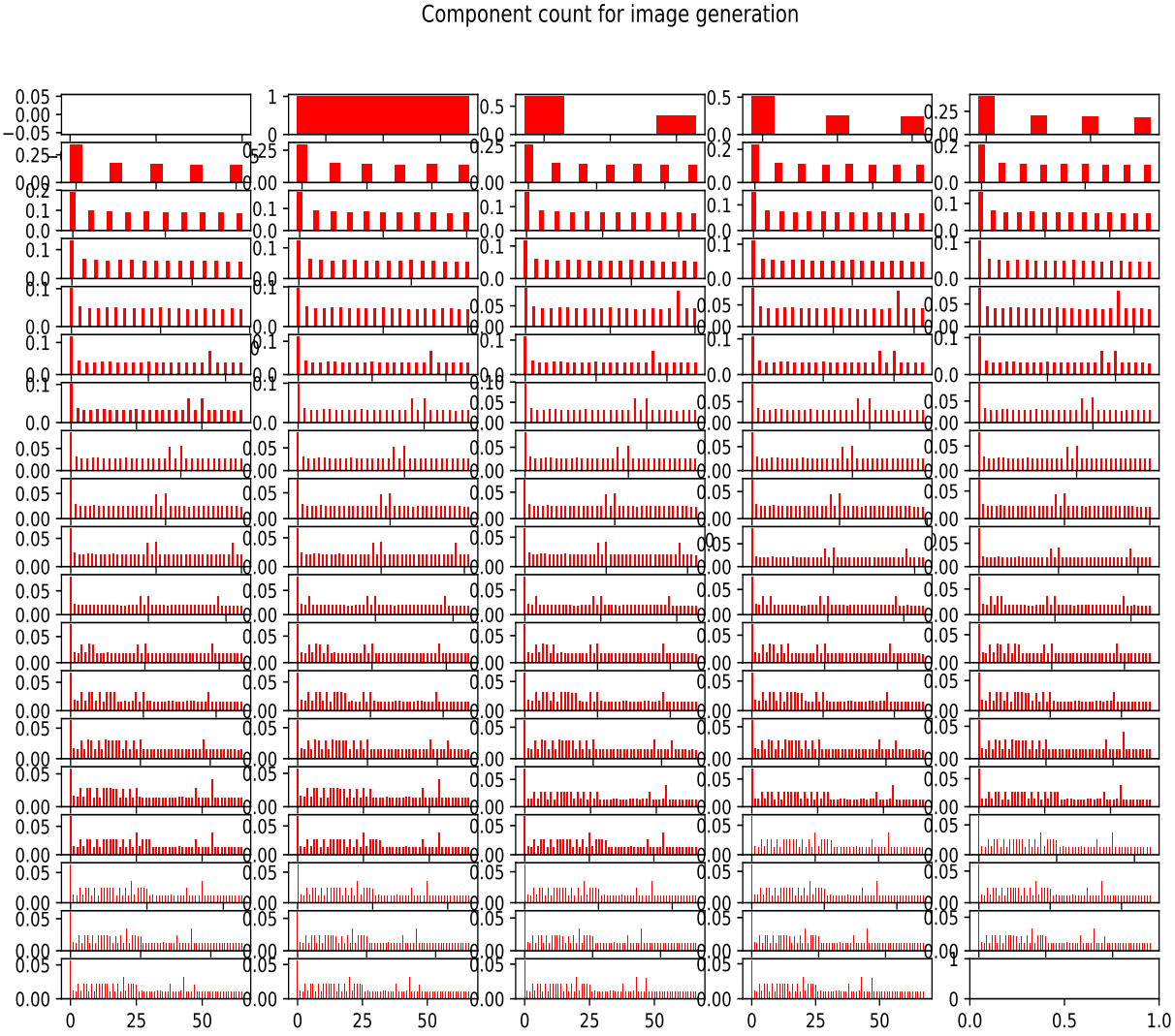


Figure 4.67: The distribution from which images are generated: each bar subplot correspond to the component count normalized by its sum, and they are plotted in a timely order from left to right. Evolution during training for component count on the EMNIST dataset that counts the number of times a component is predicted as the best for seq 1.

model	AMI	ARI	# components	accuracy(batch)	accuracy(batch HT)	accuracy(instance)	homogeneity
CURL	0.44 ± 0.014	0.132 ± 0.0166	150 ± 0.0	0.74 ± 0.01	-	0.37 ± 0.008	0.52 ± 0.0135
SOINN	0.47 ± 0.015	0.14 ± 0.01	245 ± 43.4	$\mathbf{0.785 \pm 0.072}$	-	0.483 ± 0.044	0.596 ± 0.033
STAM [154]	$\mathbf{0.63 \pm 0.004}$	$\mathbf{0.39 \pm 0.0044}$	**47**	0.75 ± 0.0125	-	$\mathbf{0.55 \pm 0.003}$	$\mathbf{0.63 \pm 0.004}$
Ours	0.555 ± 0.015	0.213 ± 0.046	66.0 ± 7.5	0.675 ± 0.036	0.76 ± 0.018	0.486 ± 0.02	0.529 ± 0.019

Table 4.18: Comparison with the state of the art on EMNIST (averaged over 3 runs) with the mixed protocol Mean±SD. (For STAM, the number of cluster is fixed manually to 47.)

novel classes, one after the other, to be detected as unknown each time when there is a class change and to recognize these learned classes. Afterwards, the agent learns the rest of the classes as new classes and to recognize them.

During both phases, the agent should be able to detect novelty and recognize learned classes, but compared to the previous learning scenarios, the classes learned and recognized in phase one are not seen again in phase II. EMNIST presents a particular challenge because the model must not forget the classes learned during Phase I. However, as we have previously illustrated, the increase in the number of categories affects the learning performance, as shown in figure 4.64 and the performance decreases for the number of categories situated above roughly 10.

Concerning other settings, the number of iterations and the training period of each class remain the same as in the first scenario. Here we demonstrate only the results of some of the models that are comparable using CNN, for the coherence with previous sections– since with CNN we manage to have the best performance compared to the use in which we use an MLP. The mixed protocol is evaluated on the EMNIST dataset since there are more categories and it is easier to demonstrate the impact of mixing detection and recognition in sequence structure.

4.4.9.2 Results

We continue to use the same CNN structure and hyperparameter settings as in the previous protocol on EMNIST. The overall results on the protocol mixing detection and recognition are shown in table 4.18. Compared to experiments in the previous section, the mix between detection and recognition for the mixed protocol does not change the total number of classes in EMNIST (47) but instead introduces more variations in the structure of the sequence in which different classes are presented. Compared to the experiments of EMNIST on the simple transition protocol, our model shows similar clustering performance. This indicates that the variation that we introduce by mixing detection and recognition, has no significant impact on the overall clustering performance of the model. STAM gives the best overall results, but with a slightly lower accuracy by batch. And again, the number of clusters has been fixed manually to the number of ground truth classes (47). As with the simple transition protocol for EMNIST in the previous section, here we did not optimise the hyperparameters to achieve the best compromise between ARI (or number of components) and accuracy. Thus, for our model, there may be a better configuration.

4.4.10 Conclusion

Novelty detection and object recognition, or, in other words, open set recognition, is a challenging problem due to the close world assumption of most state-of-the-art approaches. They often fail to estimate the probability of an input belonging to an unknown category, assigning high scores to an existing class and predicting them as learned by error. Moreover, in continual learning scenarios, the model does not have access to the entire dataset at hand and the statistics to decide whether it is an inlier or a new category need to be estimated online. The learned representation can continue to evolve while incrementally integrating new classes or concepts.

As a first step, we started by applying some common and traditional measures estimating confidence scores to detect new categories. As we have illustrated, these approaches have been constrained in their performance when it comes to new classes, since they are easily confused with old ones due to poor calibration, which is also a common problem in novelty detection or open set recognition.

Facing these challenges, we target a more robust manner to detect novelty, which is equally an approach that is applicable in the continual learning context. We propose to use the two-sample Hotelling t-squared test adapted online to both detect unknown categories and recognize learned ones, by using statistics related to a small batch that outperforms other state-of-the-art statistical approaches evaluated on both novelty detection and the recognition. Our model based on the two-sample Hotelling t-squared test is adapted during training by online parameter estimation, which computes statistics of data points of mean and covariance applying a running average. It compares the input batch with the statistics of all existing categories and decides whether the input batch belongs to the same category. This enables the self-supervision of our model to stay independent of ground truth labels. We also combine the covariance estimation with a shrinkage technique to facilitate inverting the matrix so that the model can properly estimate confidence scores based on the p-values for the input batches.

In this section, we have further experimented on three different datasets, MNIST, Fashion-MNIST, and EMNIST, by using three different protocols, to study different elements of the model that influence clustering performance and the model robustness: the number of components, hard transitions in the sequence or changing the structure of the sequence by mixing detection and recognition, and the increase in the number of classes. During clustering, the number of components impacts the learning performances since with more components, the model tends to obtain higher classification accuracy but over-segments the clusters. This requires more supervision during post-labelling therefore questioning the autonomy of the model. For this reason, and to make the studied models' performance comparable, we have chosen to take into account both the classification accuracy and ARI score by taking their mean as a trade-off. This trade-off thus considers both the classification accuracy and the number of components and, depending on the application context or other constraint, one could consider other trade-offs that favor either the former or the latter.

We have also compared types of model structures on a simple protocol, one based on an MLP and one based on a CNN. The MLP is an original setting of CURL, which we used as a baseline to compare different model variants with a simple protocol. Our model outperforms CURL by creating a clustering with the number of clusters close to the number of categories in the dataset. STAM outperforms our approach on some datasets, but it remains an approach whose number of components is not detected automatically but fixed manually. On the simple transition protocol, to compare different model architectures, we replace the MLP with a CNN to evaluate its robustness, since in the literature, CNN is a common choice for visual tasks. Our experiments show that the CNN-based VAE obtains better clustering performance compared to CURL and STAM, at least for MNIST and Fashion-MNIST. At the sample time, from the composition of clusters obtained from the previous protocol, we have constructed a second protocol that successively presents categories that are easy to be confused to create transitions that are hard to detect and are easily confused by the model. Despite the presence of hard transitions created deliberately in the sequence order, the learning performance did not decrease drastically. We have tested our model based on CNN and that of CURL on this hard protocol.

We have also tested the different approaches on the EMNIST dataset, where we were mainly interested in the performance with an increase in the number of categories. Facing this increase, our model is less robust and presents a drop in the clustering performance compared to the other two datasets. A possible reason is that when the number of classes increases, the generated examples for past learned categories have decreased and lead to a decrease in clustering performance, since during generative replay, the model can alternate between real batches and generated ones that cover all categories learned in the past. Still, our model outperforms CURL and SOINN on EMNIST. Although STAM gives excellent results on the three different datasets, it also has several drawbacks. In the clustering mode, that we used to allow for fair comparison in our experiments, it requires a spectral clustering that is computationally expensive and that may prohibit an online and real-time operation of the approach. Although on-line clustering methods exist, this would be a considerable modification of the original method. Moreover, these online methods are usually less accurate than off-line algorithms especially those that are not allowed to store all the previous data, and it is not clear what the impact on the overall performance would be. STAM has also a classification mode, that requires some labelled examples in order to assign labels to the internal cluster centroids used to classify new examples during post-labelling. The classification performance is comparable or slightly better in this mode, but in the experiments in the paper [154], 10 examples are needed per class, which amounts to 470 for EMNIST and 100 for MNIST, for example. This is

more than is needed for our approach, where around 20-30 annotated examples are enough to almost achieve the theoretically maximal accuracy for MNIST (see figure 4.49). Finally, unlike our MLP-based or CNN-based VAE, the feature extraction of STAM is more explicitly based on low-level patch intensities (aggregated into cluster centroids) – an approach that may theoretically lead to much more exuberant object representations and risks to overfit for more complex data (requiring many patch centroids), similar to Bag-of-Words-based approaches. But this needs to be investigated further.

Overall, the Hotelling t-squared test-based novelty detection and recognition has provided a supervision that guides learning, which avoids the over-segmentation of clusters and improves autonomy. However, when the number of classes increases, the performance of our model can still be improved in the future, for example by enhancing the generative replay strategy and increasing the number of generated examples of the past categories to better estimate the distribution of latent variables for different categories.

Chapter 5

Conclusion and perspectives

5.1 Conclusion and discussion

When an autonomous agent needs to build its representation of an unknown and unconstrained environment, its autonomy and its ability to learn without supervision are crucial capacities, since in many scenarios no extra external information sources are available. Besides, the agent also needs the capacity of generalization to apply its constructed knowledge to unseen data, which requires it to learn effective and meaningful representations. The dynamics of the unknown environment require the agent to be able to learn continuously while facing challenges like catastrophic forgetting and non i.i.d. data.

In this thesis, we have started by introducing from a more general aspect the cognitive theory and some neuroscience implications in AI which inspire advances in AI from a computer science aspect and also a bio-inspired one. This helps illustrate how humans can construct cognition and the implication of such development from a cognitive (psychology) perspective in learning for an autonomous agent. We have briefly introduced several possible sources of information for the autonomous agent, where the perception is an indispensable source in the field of computer vision and machine learning.

The main focus of this thesis is to target the problem of continual learning for object recognition. An important issue in this context is the concept drift, a common problem with a non-stationary data stream as input. Possible difficulties can be changes due to the arrival of new objects, or the evolving representation of old ones. But in unsupervised learning, a change in the object category is not revealed to the agent and requires the agent to detect it by itself. In addition, when it comes to detection of unknown categories most existing models are operating under the closed-world assumption, and especially neural networks can falsely attribute high confidence to an unknown category when it is calibrated onto learned ones, At the same time, the estimation of related statistics should be performed online without having the entire dataset at hand. Also, catastrophic forgetting remains an issue with state-of-the-art continual learning approaches.

To target the challenging continual learning problem, the core of our proposal lies in creating a self-supervision motivated by a novelty detection approach that guides the class-incremental continual learning of different classes. The novelty, or a possible new category candidate, is crucial, but it is difficult to detect since the model needs to estimate the statistics online in the targeted context. As here we only rely on passive perception as the only source of input, to separate different categories, the criteria are purely statistical based on learned representations. But we stated the hypothesis of temporal continuity in the input that constrains our problem and enables the use of novelty detection to formulate a self-supervised approach. Overall, our model is based on an existing approach called CURL that is composed of a VAE with dedicated latent encoders for different categories, to model the latent variable distribution with Gaussian mixtures. But as we have observed from experimental results, since CURL detects and creates new components by comparing the chosen novelty indicator, the ELBO loss, to a threshold, it provides only a limited solution to the automatic detection of the number of clusters, and tend to over-segment categories. To alleviate this problem, inspired by CURL, in our model the novelty detection results in a self-supervision that guides the model for training. Each category is modeled

by a single Gaussian component instead of the Gaussian mixture originally considered by CURL. The model alleviates catastrophic forgetting by a mixture of generative replay that alternates between real images and generative ones. Our aim is to more precisely detect category change which helps avoid over-segmentation and improves autonomy.

We have considered two cases. The first represents a scenario with a certain continuity in the data stream, where classes are presented one after the other but do not reappear for a second time. It is a rather simple learning scenario that we have targeted in the first place, before further proposing solutions in a global problem that considers the presence of objects in a more unconstrained environment. In this scenario, we aim to target a pure novelty detection case. Furthermore, to complete the first simplified learning scenario, we have considered a second case which is to consider a more complicated and less structured sequence that involves reappearing objects that the model should recognize by effectively modeling the representation of different categories. Meanwhile, the agent should not forget those representations that essential for the recognition of reappearing classes, despite the possible overall evolution of the learned representation due to the integration of new classes or tasks.

As a first proposal, we have adopted the Page-Hinckley test, a common choice of change detection in data streams. We use an indicator based on the Heaviside function that compares the model's internal ELBO loss to a threshold, or to eventually apply a smoothing technique based on the running average. The running average smoothing alleviates the fluctuation caused by intra-class variations to reduce potential false positive detection of novelty. The Page-Hinckley test compares the accumulated difference between the indicator and its average performance to find the moment of change. This allows the model's autonomous construction of an internal label that stays close to the real category changes and allows training under self-supervision, with more autonomy motivated by an improvement of detection in the number of categories and the moment of change during training and reduces the effort in post-labelling.

Our second axis of study is to include the review of objects in the learning scenario. Starting by comparing with common uncertainty metrics, we have applied the Hotelling t-squared test, a hypothesis test that detects both novelty and recognizes learning objects. The Hotelling t-squared test is operated offline, but we adapted the test online through online statistics estimation from the VAE modeled latent variables. The online estimation is challenging as the learned representation evolves for past categories and covariance estimation are more or less biased since only generated images are accessible for the past category. Our proposal is based on a running average and a running covariance combined with a shrinkage technique, which is integrated into the Hotelling t-squared test performed online. From these estimations, the Hotelling t-squared test is capable of predicting for each input batch by comparing the p-value to a threshold, whether a given input batch corresponds to a learned category or is to be rejected as unknown. This provides a self-supervision that guides the recognition of learning objects and the creation of new components for our model and allows learning with self-supervision that detects better the category changes under the reappearance of objects. Our proposal manages to alleviate the problem of calibration under the closed world assumptions in the common open set recognition problem, since the p-value has a better statistical significance and outperforms approaches like the frequently applied softmax in classification problems. We have experimentally compared both an MLP-based VAE, which is the original experimental setting of CURL, and in a second time, a CNN-based VAE since CNN is more widely applied in the field of computer vision.

For both proposals, we have studied several elements that are crucial for the model and its clustering performances: the impact of the number of components, the impact of hard transitions that are composed by similar categories that are easily confused by the model that we consider appearing one after the other. In addition, we have also illustrated the influence of a number of annotated examples in the post-labelling process that associates different clusters to classes, and we compare different models in terms of the need for annotated examples that show their autonomy in this aspect. We have also experimentally shown the impact of increasing number of categories and the different influences it may have on the studied models.

In both cases, the self-supervision has allowed us to learn a model that creates a number of clusters close to the real number of categories in the dataset. On the contrary, CURL tends to over-segment different clusters, although both cases depend on the choice of hyperparameters, since the new category detection and creation process is based on a relatively simple thresholding approach and needs to tune the threshold manually to give the optimal performance (according to the chosen metric). Note that in

our approach the threshold is on the p-value, thus it is easier to fix because the p-value is a probability and the standard threshold of 0.05 gave almost the optimal results in most cases. The other models we compared to are SOINN and STAM. The former learns the topology of classes, but the problem also lies in the number of nodes exceeding the number of categories, which can be seen as a sort of over-segmentation although the nodes in SOINN do not have exactly the same significance as clusters in the clustering approaches. STAM is an unsupervised continual learning approach, but in STAM, the feature (representation) learning stage and the clustering/classification stage are decoupled. Also, in the clustering mode, that we used in our experiments for fair comparison, the number of clusters is determined manually by the user, which may slightly bias the results, and the clustering is offline. CURL and SOINN on the other hand tend to produce an excessive number of components. In theory, this might enable these learning of a model with more expressive power and reduce the amount of parameter sharing (in the latent encoder) or overlapping between different categories. but this over-segmentation itself increases the amount of necessary labels in the post-labelling, i.e. it requires more annotated examples to correctly associate to each cluster its represented class. Our proposal better models the number (and separation) of categories in the dataset and outperforms CURL and SOINN by scoring higher in AMI and ARI scores and partly in accuracy. For STAM the results are superior to ours for some datasets: Fashion-MNIST and EMNIST, but inferior for MNIST which is supposed to be simpler. Thus there are still some questions that remain open regarding the general performance of capacities of STAM. Meanwhile, the scalability regarding the number of classes to learn remains a challenging problem in our model and that of CURL. Especially for CURL, creating more components allows better clustering accuracy but is also constrained when there are many categories in the dataset since the memory requirements for creating these models are quite large

Overall, our proposal based on the Page-Hinckley test allows online novelty detection using statistics of the chosen indicator, while the Hotelling t-squared test has allowed both novelty detection and recognition. Our proposals tackle the problem of continual learning with self-supervision, which ensures the agent's autonomy in learning in an unknown and unconstrained environment, although it may still be limited in some cases such as distinguishing statistically close objects. Although this work has not been applied to video streams and considering its application on image sequences formed by public datasets, we argue that video streams show (even) more continuity in observed objects. And we target the scenario with image sequences to study various related issues in the problem of unsupervised continual learning. This temporal correlation is helpful to find the transition between objects or to detect object boundary via the use of optic flow.

5.2 Perspectives: future research directions

In this thesis, we have studied continual learning for unsupervised object recognition. Our proposal is based on novelty detection as self-supervision to guide learning. The current model expands dynamically by creating a new latent encoder (component) when a new category is detected to avoid parameter sharing that may lead to catastrophic forgetting. It extends CURL by combining with the Page-Hinckley test and the Hotelling t-squared test. However, in the scale increase problem where one can not expand the model infinitely since numerous components also lead to more computational and memory resources. The increase in the number of categories to learn results in some catastrophic forgetting. But we can also consider other techniques in the literature, such as to activate only part of the network or to prune the network structure with certain modularity.

In the future, we will apply our work on datasets of video streams. This will change the dynamics of the input data since compared to images, the video stream is with more temporal auto-correlation and less intra-class variability. This will be challenging for the estimation of covariance since, in addition, the estimation of statistics should be online. In this setting, a combination of the two proposed approaches, the Page-Hinckley test for novelty detection and the Hotelling t-squared test for recognition (potentially with more shrinkage) may be better. But at the same time, the application on video streams should help get a step closer to the data stream humans or an autonomous agent perceive in "real life" and allows a wider range of applications for our model.

The strategy of alleviating catastrophic forgetting can be studied in the future. In the current work,

we have continued to use the same generative replay strategy as CURL, which uses generated examples. But from a more general point of view, the replay can occur in other forms, such as using a memory system for rehearsal. Such a memory system and a replay strategy with real images have provided the possibility to apply other cognitive and neuroscience-related strategies, since in a broader perspective, the use of memory in AI model is bio-inspired. Other neuroscience-related strategies include for example attention, which could define an exemplar selection criterion. The agent can choose what to replay. For example, a possible criterion as in [65] can be the similarity of a previous experience with the current category. From this point of view, both the selection of examples to store and the selective replay are important to build better continual learning models. Integrating this strategy into our model helps better model the distribution from which images are generated and replayed and improve the model's performances in modelling the representation of different categories and the estimation of prior and posterior distribution. In addition, the agent can also automatically learn a curriculum that enables better clustering performances and actively choose what to learn following certain motivation mechanisms such as intrinsic motivation. Moreover, this work is based on an approach that builds the representation of different objects from a perceptual input as the only source of information. But in future works, it will be interesting to imagine a mobile robot learning different objects and getting extra feedback from action and interaction for a complete cognitive model with multi-modality. In this manner, the agent separates objects through additional sources of information, which would certainly improve the performance in continual learning for object recognition in general.

Bibliography

[1] Mohammed Y Abbass, Ki-Chul Kwon, Nam Kim, Safey A Abdelwahab, Fathi E Abd El-Samie, and Ashraf AM Khalaf. A survey on online learning for visual tracking. *The Visual Computer*, 37:993–1014, 2021.

[2] Marcel R Ackermann, Marcus Märtens, Christoph Raupach, Kamil Swierkot, Christiane Lammersen, and Christian Sohler. Streamkm++ a clustering algorithm for data streams. *Journal of Experimental Algorithmics (JEA)*, 17:2–1, 2012.

[3] Sander Adam, Lucian Busoniu, and Robert Babuska. Experience replay for real-time reinforcement learning control. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(2):201–212, 2011.

[4] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154, 2018.

[5] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3366–3375, 2017.

[6] Erin L Allwein, Robert E Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of machine learning research*, 1(Dec):113–141, 2000.

[7] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017.

[8] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[9] Umut Asan and Secil Ercan. An introduction to self-organizing maps. In *Computational Intelligence Systems in Industrial Engineering*, pages 295–315. Springer, 2012.

[10] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Visual tracking with online multiple instance learning. In *2009 IEEE Conference on computer vision and Pattern Recognition*, pages 983–990. IEEE, 2009.

[11] Peter L Bartlett and Marten H Wegkamp. Classification with a reject option using a hinge loss. *Journal of Machine Learning Research*, 9(8), 2008.

[12] Peyman Bateni, Raghav Goyal, Vaden Masrani, Frank Wood, and Leonid Sigal. Improved few-shot visual classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14493–14502, 2020.

[13] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[14] Shawn Beaulieu, Lapo Frati, Thomas Miconi, Joel Lehman, Kenneth O Stanley, Jeff Clune, and Nick Cheney. Learning to continually learn. *arXiv preprint arXiv:2002.09571*, 2020.

[15] Eseoghene Ben-Iwhiwhu, Jeffery Dick, Nicholas A Ketz, Praveen K Pilly, and Andrea Soltoggio. Context meta-reinforcement learning via neuromodulation. *arXiv preprint arXiv:2111.00134*, 2021.

[16] Abhijit Bendale and Terrance Boult. Towards open world recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1893–1902, 2015.

[17] Abhijit Bendale and Terrance E Boult. Towards open set deep networks. In *IEEE conference on Computer Vision and Pattern Recognition*, pages 1563–1572, 2016.

[18] Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 17–36. JMLR Workshop and Conference Proceedings, 2012.

[19] Brandon Bennett, Derek R Magee, Anthony G Cohn, and David C Hogg. Enhanced tracking and recognition of moving objects by reasoning about spatio-temporal continuity. *Image and Vision Computing*, 26(1):67–81, 2008.

[20] Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 443–448. SIAM, 2007.

[21] Matthew Botvinick, Jane X Wang, Will Dabney, Kevin J Miller, and Zeb Kurth-Nelson. Deep reinforcement learning and its neuroscientific implications. *Neuron*, 107(4):603–616, 2020.

[22] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688, 1993.

[23] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[24] Louis-Charles Caron. *Combining Depth, Color and Position Information for Object Instance Recognition on an Indoor Mobile Robot*. PhD thesis, ENSTA ParisTech, 2015.

[25] Louis-Charles Caron, David Filliat, and Alexander Gepperth. Neural network fusion of color, depth and location for object instance recognition on a mobile robot. In *European Conference on Computer Vision*, pages 791–805. Springer, 2014.

[26] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 132–149, 2018.

[27] Robert Oliver Castle, Darren J Gawley, Georg Klein, and David William Murray. Towards simultaneous recognition, localization and mapping for hand-held and wearable cameras. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4102–4107. IEEE, 2007.

[28] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 233–248, 2018.

[29] Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. *SIAM Journal on Computing*, 33(6):1417–1440, 2004.

[30] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 2180–2188, 2016.

[31] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognitiqon*, pages 15750–15758, 2021.

[32] Zhiyi Chen and Tong Lin. Revisiting gradient episodic memory for continual learning. 2019.

[33] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: an extension of MNIST to handwritten letters. *CoRR*, abs/1702.05373, 2017.

[34] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[35] Celine Craye. *Intrinsic motivation mecanisms for incremental learning of visual saliency.* PhD thesis, Université Paris-Saclay (ComUE), 2017.

[36] Florin Cutzu and John K Tsotsos. The selective tuning model of attention: psychophysical evidence for a suppressive annulus around an attended item. *Vision research*, 43(2):205–219, 2003.

[37] James J DiCarlo, Davide Zoccolan, and Nicole C Rust. How does the brain solve visual object recognition? *Neuron*, 73(3):415–434, 2012.

[38] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80, 2000.

[39] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[40] Kai Olav Ellefsen, Jean-Baptiste Mouret, and Jeff Clune. Neural modularity helps organisms evolve to learn new skills without forgetting old skills. *PLoS computational biology*, 11(4):e1004128, 2015.

[41] David Filliat, Emmanuel Battesti, Stéphane Bazeille, Guillaume Duceux, Alexander Gepperth, Lotfi Harrath, Islem Jebari, Rafael Pereira, Adriana Tapus, Cedric Meyer, et al. Rgbd object recognition and visual texture classification for indoor semantic mapping. In *2012 IEEE international conference on technologies for practical robot applications (TePRA)*, pages 127–132. IEEE, 2012.

[42] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[43] Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *International Conference on Machine Learning*, pages 1920–1930. PMLR, 2019.

[44] Evelyn Fix and Joseph Lawson Hodges. Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review/Revue Internationale de Statistique*, 57(3):238–247, 1989.

[45] Jerome H Friedman. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378, 2002.

[46] S. Furao, T. Ogura, and O. Hasegawa. An enhanced self-organizing incremental neural network for online unsupervised learning. *Neural Networks*, 20(8), 2007.

[47] Shen Furao and Osamu Hasegawa. An incremental network for on-line unsupervised classification and topology learning. *Neural networks*, 19(1):90–106, 2006.

[48] Shuyang Gao, Rob Brekelmans, Greg Ver Steeg, and Aram Galstyan. Auto-encoding total correlation explanation. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1157–1166. PMLR, 2019.

[49] Zengan Gao. Application of cluster-based local outlier factor algorithm in anti-money laundering. In *2009 International Conference on Management and Service Science*, pages 1–4. IEEE, 2009.

[50] Andrew B Gardner, Abba M Krieger, George Vachtsevanos, Brian Litt, and Leslie Pack Kaelbing. One-class novelty detection for seizure analysis from intracranial eeg. *Journal of Machine Learning Research*, 7(6), 2006.

[51] Chuanxing Geng, Sheng-jun Huang, and Songcan Chen. Recent advances in open set recognition: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[52] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[53] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[54] Micah Goldblum, Steven Reich, Liam Fowl, Renkun Ni, Valeriia Cherepanova, and Tom Goldstein. Unraveling meta-learning: Understanding feature representations for few-shot tasks. In *International Conference on Machine Learning*, pages 3607–3616. PMLR, 2020.

[55] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[56] Jacqueline Gottlieb and Pierre-Yves Oudeyer. Towards a neuroscience of active sampling and curiosity. *Nature Reviews Neuroscience*, 19(12):758–770, 2018.

[57] Prasoon Goyal, Zhiting Hu, Xiaodan Liang, Chenyu Wang, and Eric P Xing. Nonparametric variational auto-encoders for hierarchical representation learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5094–5102, 2017.

[58] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[59] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

[60] Rasmus Boll Greve, Emil Juul Jacobsen, and Sebastian Risi. Evolving neural turing machines for reward-based learning. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 117–124, 2016.

[61] Frank E Grubbs. Sample criteria for testing outlying observations. *The Annals of Mathematical Statistics*, pages 27–58, 1950.

[62] Xiaoning Han, Shuailong Li, Xiaohui Wang, and Weijia Zhou. Semantic mapping for mobile robots in indoor scenes: A survey. *Information*, 12(2):92, 2021.

[63] Chris Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[64] Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, 2017.

[65] Tyler L Hayes and Christopher Kanan. Selective replay enhances learning in online continual analogical reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3502–3512, 2021.

[66] Tyler L Hayes, Giri P Krishnan, Maxim Bazhenov, Hava T Siegelmann, Terrence J Sejnowski, and Christopher Kanan. Replay in deep learning: Current approaches and missing biological elements. *arXiv preprint arXiv:2104.04132*, 2021.

[67] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[68] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[69] Donald Olding Hebb. *The organisation of behaviour: a neuropsychological theory*. Science Editions New York, 1949.

[70] Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*, 2018.

[71] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2016.

[72] H. Hotelling. The generalization of student's ratio. *Annals of Mathematical Statistics*, 2(3):360–378, 1931.

[73] Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *arXiv preprint arXiv:1810.12488*, 2018.

[74] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 97–106, 2001.

[75] Pedro R Mendes Júnior, Roberto M De Souza, Rafael de O Werneck, Bernardo V Stein, Daniel V Pazinato, Waldir R de Almeida, Otávio AB Penatti, Ricardo da S Torres, and Anderson Rocha. Nearest neighbors distance ratio open-set classifier. *Machine Learning*, 106(3):359–386, 2017.

[76] Leonid Karlinsky, Joseph Shtok, Sivan Harary, Eli Schwartz, Amit Aides, Rogerio Feris, Raja Giryes, and Alex M Bronstein. Repmet: Representative-based metric learning for classification and few-shot object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5197–5206, 2019.

[77] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[78] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *stat*, 1050:1, 2014.

[79] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A Rusu, K. Milan, J. Quan, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[80] Janis Klaise, Arnaud Van Looveren, Clive Cox, Giovanni Vacanti, and Alexandru Coca. Monitoring and explainability of models in production. *arXiv preprint arXiv:2007.06299*, 2020.

[81] Stefan Klanke, Sethu Vijayakumar, and Stefan Schaal. A library for locally weighted projection regression. *Journal of Machine Learning Research*, 2008.

[82] Jeremias Knoblauch, Hisham Husain, and Tom Diethe. Optimal continual learning has perfect memory and is np-hard. In *International Conference on Machine Learning*, pages 5327–5337. PMLR, 2020.

[83] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

[84] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[85] Soheil Kolouri, Nicholas Ketz, Xinyun Zou, Jeffrey Krichmar, and Praveen Pilly. Attention-based selective plasticity. *arXiv preprint arXiv:1903.06070*, 2019.

[86] J Zico Kolter and Marcus A Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *The Journal of Machine Learning Research*, 8:2755–2790, 2007.

[87] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[88] Dharshan Kumaran, Demis Hassabis, and James L McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, 20(7):512–534, 2016.

[89] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*, pages 6402–6413, 2017.

[90] Lucian Leahu, Phoebe Sengers, and Michael Mateas. Interactionist ai and the promise of ubicomp, or, how to put your box in the world without putting the world in your box. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 134–143, 2008.

[91] Yann LeCun. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[92] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[93] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[94] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE international symposium on circuits and systems*, pages 253–256. IEEE, 2010.

[95] Timothée Lesort, Massimo Caccia, and Irina Rish. Understanding continual learning settings with data distribution drift analysis. *arXiv preprint arXiv:2104.01678*, 2021.

[96] Aoxue Li, Tiange Luo, Tao Xiang, Weiran Huang, and Liwei Wang. Few-shot learning with global class representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9715–9724, 2019.

[97] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[98] S. Liang, Y. Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *ICLR*, 2018.

[99] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

[100] W. Liu, X. Wang, J. D. Owens, and Y. Li. Energy-based out-of-distribution detection. In *Advances in neural information processing systems*, 2020.

[101] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[102] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476, 2017.

[103] Viktor Losing, Barbara Hammer, and Heiko Wersing. Incremental on-line learning: A review and comparison of state of the art algorithms. *Neurocomputing*, 275:1261–1274, 2018.

[104] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.

[105] G Lowe. Sift-the scale invariant feature transform. *Int. J*, 2(91-110):2, 2004.

[106] Benno Lüders, Mikkel Schläger, and Sebastian Risi. Continual learning through evolvable neural turing machines. In *Nips 2016 workshop on continual learning and deep networks (cldl 2016)*, 2016.

[107] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[108] Adam H Marblestone, Greg Wayne, and Konrad P Kording. Toward an integration of deep learning and neuroscience. *Frontiers in computational neuroscience*, page 94, 2016.

[109] Charles E Martin and Praveen K Pilly. Probabilistic program neurogenesis. In *ALIFE 2019: The 2019 Conference on Artificial Life*, pages 440–447. MIT Press, 2019.

[110] Frank J Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78, 1951.

[111] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[112] Nawel Medjkoune, Frédéric Armetta, Mathieu Lefort, and Stefan Duffner. Autonomous object recognition in videos using siamese neural networks. In *EUCognition Meeting (European Society for Cognitive Systems) on" Learning: Beyond Deep Neural Networks"*, page 4, 2017.

[113] Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in psychology*, 4:504, 2013.

[114] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[115] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[116] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[117] Jacob Montiel, Jesse Read, Albert Bifet, and Talel Abdessalem. Scikit-multiflow: A multi-output streaming framework. *Journal of Machine Learning Research*, 19(72):1–5, 2018.

[118] Martin Mundt, Yong Won Hong, Iuliia Pliushch, and Visvanathan Ramesh. A wholistic view of continual learning with deep neural networks: Forgotten lessons and the bridge to active and open world learning. *arXiv preprint arXiv:2009.01797*, 2020.

[119] Fionn Murtagh. Multilayer perceptrons for classification and regression. *Neurocomputing*, 2(5-6):183–197, 1991.

[120] Medhini Narasimhan, Erik Wijmans, Xinlei Chen, Trevor Darrell, Dhruv Batra, Devi Parikh, and Amanpreet Singh. Seeing the un-scene: Learning amodal semantic maps for room navigation. In *European Conference on Computer Vision*, pages 513–529. Springer, 2020.

[121] Lawrence Neal, Matthew Olson, Xiaoli Fern, Weng-Keen Wong, and Fuxin Li. Open set learning with counterfactual images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 613–628, 2018.

[122] Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2(3):4, 2018.

[123] Nikunj C Oza and Stuart J Russell. Online bagging and boosting. In *International Workshop on Artificial Intelligence and Statistics*, pages 229–236. PMLR, 2001.

[124] Ewan S Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

[125] Simone Parisi, Victoria Dean, Deepak Pathak, and Abhinav Gupta. Interesting object, curious agent: Learning task-agnostic exploration. *Advances in Neural Information Processing Systems*, 34, 2021.

[126] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[127] Duc Truong Pham, Stefan Simeonov Dimov, and CD Nguyen. An incremental k-means algorithm. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 218(7):783–795, 2004.

[128] Jean Piaget. Part i: Cognitive development in children: Piaget development and learning. *Journal of research in science teaching*, 2(3):176–186, 1964.

[129] Marco AF Pimentel, David A Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, 2014.

[130] Agyztia Premana, Akhmad Pandhu Wijaya, and Moch Arief Soeleman. Image segmentation using gabor filter and k-means clustering method. In *2017 International Seminar on Application for Technology of Information and Communication (iSemantic)*, pages 95–99. IEEE, 2017.

[131] Guo-Jun Qi and Jiebo Luo. Small data challenges in big data era: A survey of recent progress on unsupervised and semi-supervised methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[132] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[133] Dushyant Rao, Francesco Visin, Andrei Rusu, Razvan Pascanu, Yee Whye Teh, and Raia Hadsell. Continual unsupervised representation learning. In *Advances in Neural Information Processing Systems*, pages 7645–7655, 2019.

[134] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. iCaRL: Incremental classifier and representation learning. In *CVPR*, pages 2001–2010, 2017.

[135] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[136] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[137] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[138] J. Ren, P. J. Liu, E. Fertig, J. Snoek, R. Poplin, M. A. DePristo, J. V. Dillon, and B. Lakshminarayanan. Likelihood ratios for out-of-distribution detection. In *Advances in Neural Information Processing Systems*, 2019.

[139] Ryne Roady, Tyler L Hayes, Ronald Kemker, Ayesha Gonzales, and Christopher Kanan. Are out-of-distribution detection methods effective on large-scale datasets? *arXiv preprint arXiv:1910.14034*, 2019.

[140] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 410–420, 2007.

[141] Martin Runz, Maud Buffier, and Lourdes Agapito. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 10–20. IEEE, 2018.

[142] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[143] Amir Saffari, Martin Godec, Thomas Pock, Christian Leistner, and Horst Bischof. Online multi-class lpboost. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3570–3577. Ieee, 2010.

[144] Nguyen Sao Mai. *A Curious Robot Learner for Interactive Goal-Babbling*. PhD thesis, Citeseer, 2013.

[145] Walter J Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult. Toward open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1757–1772, 2012.

[146] Bernhard Schölkopf, Robert C Williamson, Alexander J Smola, John Shawe-Taylor, John C Platt, et al. Support vector method for novelty detection. In *NIPS*, volume 12, pages 582–588. Citeseer, 1999.

[147] David Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178, 2010.

[148] John Searle. Chinese room argument. *Scholarpedia*, 4(8):3100, 2009.

[149] J. Serrà, D. Alvarez, V. Gómez, O. Slizovskaia, J. F. Núnez, and J. Luque. Input complexity and out-of-distribution detection with likelihood-based generative models. In *International Conference on Learning Representations*, 2020.

[150] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[151] Jianbing Shen, Xiaopeng Hao, Zhiyuan Liang, Yu Liu, Wenguan Wang, and Ling Shao. Real-time superpixel segmentation by dbscan clustering algorithm. *IEEE transactions on image processing*, 25(12):5933–5942, 2016.

[152] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[153] Jivko Sinapov, Taylor Bergquist, Connor Schenck, Ugonna Ohiri, Shane Griffith, and Alexander Stoytchev. Interactive object recognition using proprioceptive and auditory feedback. *The International Journal of Robotics Research*, 30(10):1250–1262, 2011.

[154] James Smith, Cameron Taylor, Seth Baer, and Constantine Dovrolis. Unsupervised progressive learning and the stam architecture. *arXiv preprint arXiv:1904.02021*, 2019.

[155] Niko Sunderhauf, Feras Dayoub, Sean McMahon, Ben Talbot, Ruth Schulz, Peter Corke, Gordon Wyeth, Ben Upcroft, and Michael Milford. Place categorization and semantic mapping on a mobile robot. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 5729–5736. IEEE, 2016.

[156] Niko Sünderhauf, Sareh Shirazi, Feras Dayoub, Ben Upcroft, and Michael Milford. On the performance of convnet features for place recognition. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 4297–4304. IEEE, 2015.

[157] James Supancic III and Deva Ramanan. Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 322–331, 2017.

[158] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[159] Nadeem Ahmed Syed, Huan Liu, and Kah Kay Sung. Incremental learning with support vector machines. 1999.

[160] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[161] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[162] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[163] David MJ Tax and Robert PW Duin. Support vector domain description. *Pattern recognition letters*, 20(11-13):1191–1199, 1999.

[164] Yee Whye Teh. Dirichlet process., 2010.

[165] Michael Tschannen, Olivier Bachem, and Mario Lucic. Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*, 2018.

[166] Cem C Tutum and Risto Miikkulainen. Adapting to unseen environments through explicit representation of context. *arXiv preprint arXiv:2002.05640*, 2020.

[167] Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

[168] Arnaud Van Looveren, Janis Klaise, Giovanni Vacanti, Oliver Cobb, and Ashley Scillitoe. Alibi detect: Algorithms for outlier, adversarial and drift detection, 2019.

[169] Vinay Kumar Verma, Gundeep Arora, Ashish Mishra, and Piyush Rai. Generalized zero-shot learning via synthesized examples. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4281–4289, 2018.

[170] Eli Verwimp, Matthias De Lange, and Tinne Tuytelaars. Rehearsal revealed: The limits and merits of revisiting samples in continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9385–9394, 2021.

[171] Sethu Vijayakumar, Aaron D'souza, and Stefan Schaal. Incremental online learning in high dimensions. *Neural computation*, 17(12):2602–2634, 2005.

[172] Sethu Vijayakumar and Stefan Schaal. Locally weighted projection regression: An o (n) algorithm for incremental real time learning in high dimensional space. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, volume 1, pages 288–293. Morgan Kaufmann, 2000.

[173] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

[174] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29:3630–3638, 2016.

[175] Heng Wang and Zubin Abraham. Concept drift detection for streaming data. In *2015 international joint conference on neural networks (IJCNN)*, pages 1–9. IEEE, 2015.

[176] Scott Wares, John Isaacs, and Eyad Elyan. Data stream mining: methods and challenges for handling concept drift. *SN Applied Sciences*, 1(11):1–19, 2019.

[177] Satosi Watanabe. Information theoretical analysis of multivariate correlation. *IBM Journal of research and development*, 4(1):66–82, 1960.

[178] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

[179] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *preprint arXiv:1708.07747*, 2017.

[180] Dongkuan Xu and Yingjie Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193, 2015.

[181] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *arXiv preprint arXiv:1411.1792*, 2014.

[182] Yang Yu, Wei-Yang Qu, Nan Li, and Zimin Guo. Open-category classification by adversarial sample generation. *arXiv preprint arXiv:1705.08722*, 2017.

[183] Zhongjie Yu, Lin Chen, Zhongwei Cheng, and Jiebo Luo. Transmatch: A transfer-learning scheme for semi-supervised few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12856–12864, 2020.

[184] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[185] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995, 2017.

[186] Da Zhang, Hamid Maei, Xin Wang, and Yuan-Fang Wang. Deep reinforcement learning for visual object tracking in videos. *arXiv preprint arXiv:1701.08936*, 2017.

[187] Jian Zhang, Zoubin Ghahramani, and Yiming Yang. A probabilistic model for online document clustering with application to novelty detection. *Advances in neural information processing systems*, 17:1617–1624, 2004.

[188] Jianming Zhang, Sarah Adel Bargal, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. *International Journal of Computer Vision*, 126(10):1084–1102, 2018.

[189] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. *ACM sigmod record*, 25(2):103–114, 1996.

[190] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[191] Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In *International Conference on Machine Learning*, pages 7693–7702. PMLR, 2019.

[192] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*, 2019.

UNIVERSITÉ DE LYON | INSA

FOLIO ADMINISTRATIF

THESE DE L'UNIVERSITE DE LYON OPEREE AU SEIN DE L'INSA LYON

NOM : DAI
(avec précision du nom de jeune fille, le cas échéant)

DATE de SOUTENANCE : 14/09/2022

Prénoms : Ruiqi

TITRE : Continual class-incremental learning for autonomous object recognition in image sequences

NATURE : Doctorat

Numéro d'ordre : 2022LYSEI0072

Ecole doctorale : ED InfoMaths

Spécialité : Informatique

RESUME :

For an agent, it is very challenging to autonomously elaborate and make use of a visual representation of its open environment. It is necessary to introduce new categories for unseen objects, and recognize already seen objects to refine and make evolve the representation, which is even more difficult when undergoing uncertainty and variability under uncontrolled operation conditions. The autonomy of the agent in classification, its adaptation to a changing environment as well as continual representation building are important properties of such a dynamic machine learning system. However, in some of the existing systems, especially with neural networks, the acquired representation tends to gradually drift away from initial observations which causes catastrophic forgetting. Another challenge comes from the decision whether incoming observations are new or belong to already seen objects. One has to detect novelty and introduce a new concept or class if necessary, and maintain the already acquired representation, i.e. recognize the object and make use of new observations appropriately, which is part of the more general problem of finding a meaningful and robust representation under the stability-plasticity dilemma.

In this thesis, we address these challenges by adopting a state-of-the-art continual unsupervised representation learning model based on a specific Variational Auto-Encoder (VAE) architecture that we extended and applied on sequences of images showing different objects. Assuming that, in a realistic environment, there is some continuity in the sequence of object observations, the first contribution is an algorithm that robustly detects when object class changes occur in the image stream based on the dynamic evolution of the observation likelihood. It is shown that the thus obtained clusters in the representation space are more consistent with real objects and therefore facilitate the classification of known objects. The second contribution replaces the first approach and further increases the autonomy of the model by introducing an algorithm based on the Hotelling t-squared statistical hypothesis test that is able to continuously detect class changes and simultaneously decides if the currently observed object is novel or already present in the learnt representation. Extensive experimental results on several image benchmarks show that this self supervised continual learning approach is able to build representations that favor the autonomy of the agent by minimising supervision while maintaining a high object recognition accuracy compared to the state of the art.

MOTS-CLÉS : Continual learning, object recognition, representation learning

Laboratoire (s) de recherche : LIRIS

Directeur de thèse: Stefan Duffner

Président de jury :

Composition du jury :

Nicole Vincent	PR	Université de Paris Descartes	Examinatrice
Céline Hudelot	PR	CentraleSupélec	Examinatrice
Thierry Château	PR	Université Clermont-Auvergne	Rapporteur
Patrick Reignier	PR	Univ. Grenoble Alpes	Rapporteur
Duffner Stefan	MCF-HDR	INSA Lyon	Directeur.rice de thèse
Armetta Frederic	MCF	Université Claude Bernard Lyon 1	Co-encadrant
Lefort Mathieu	MCF	Université Claude Bernard Lyon 1	Co-encadrant
Guillermin Mathieu	MCF	Université catholique de Lyon	Co-encadrant