Self-supervised learning of deep visual representations

Mathilde Caron

Université Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE GRENOBLE ALPES

Spécialité : **Mathématiques et Informatique**

Arrêté ministériel : 25 mai 2016
Présentée par

Mathilde CARON

Thèse dirigée par **Julien MAIRAL, chercheur, Inria**
préparée au sein du **Laboratoire Jean Kuntzmann**
dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Apprentissage Auto-Supervisé de Représentations Visuelles avec des Réseaux de Neurones Profonds

Self-Supervised Learning of Deep Visual Representations

Thèse soutenue publiquement le **9 décembre 2021,**
devant le jury composé de:

Julien Mairal
Chargé de recherche HDR, INRIA CENTRE GRENOBLE-RHONE ALPES, Directeur de thèse
Andrew Zisserman
Professeur, University of Oxford, Rapporteur
Alexei A. Efros
Professeur, University of California, Berkeley, Rapporteur
Cordelia Schmid
Directeur de recherche, Inria, Présidente
Diane Larlus
Ingénieur docteur, Naver Labs Europe, Examinatrice
Alexey Dosovitskiy
Ingénieur docteur, Google Brain, Examinateur

Membres invités :
Armand Joulin
Ingénieur docteur, Facebook AI Research, Co-encadrant
Piotr Bojanowski
Ingénieur docteur, Facebook AI Research, Co-encadrant

Contents

Abstract

Humans and many animals can see the world and understand it effortlessly which gives hope that visual perception could be realized by computers and Artificial Intelligence. More importantly, living beings acquire such an understanding of the visual world autonomously, without the intervention of a supervisor explicitly telling them what, where or who is to be seen. This suggests that visual perception can be achieved without too much explicit human supervision but simply by letting systems observe large amounts of visual inputs. In particular, this manuscript tackles the problem of self-supervised learning which consists in training deep neural networks without using any human annotations. Typically, neural networks require large amounts of annotated data, which have limited their applications in fields where accessing annotations is expensive or difficult. Moreover, manual annotations are biased towards a specific task and towards the annotators own biases, which can result in noisy and unreliable signals. Training systems without annotations could lead to better, more generic and robust representations. In this manuscript, we present different contributions to the fast-growing field of self-supervised visual representation learning.

In particular, we start by extending a promising category of self-supervised approaches, namely deep clustering, which trains deep networks while simultaneously mining groups of visually consistent images in a data collection. We then identify the limits of deep clustering methods such as their difficulty to scale to very large datasets or the fact that they are prone to trivial solutions. As a result, we propose novel and improved self-supervised methods that outperform their supervised counterparts on several benchmarks and exhibit interesting properties. For example, the resulting self-supervised networks contain generic representations that transfer well to different datasets and tasks. We also find that they contain explicit information about the semantic segmentation of an image. Finally, we make an effort throughout this manuscript to assess our self-supervised models *in the wild*, by training them on hundreds of millions of random unlabeled images from the Internet.

Keywords: self-supervised learning, computer vision, unsupervised learning, deep learning, artificial intelligence.

Résumé

Les humains et de nombreux animaux peuvent voir le monde et le comprendre sans effort, ce qui laisse espérer que la perception visuelle pourrait être réalisée par les ordinateurs et l'intelligence artificielle. Plus important encore, les êtres vivants acquièrent une telle compréhension du monde visuel de manière autonome, sans l'intervention d'un superviseur externe leur disant explicitement quoi, où ou qui doit être vu. Cela suggère que la perception visuelle peut être obtenue sans trop de supervision humaine explicite mais simplement en laissant les systèmes observer par eux-mêmes de grandes quantités d'entrées visuelles.

En particulier, ce manuscrit aborde le problème de l'apprentissage auto-supervisé. Cela consiste à entraîner des systèmes de réseaux de neurones profonds sans utiliser aucune annotation humaine. En règle générale, les réseaux de neurones nécessitent de grandes quantités de données annotées, ce qui a limité leurs applications dans les domaines où l'accès à ces annotations est coûteux ou difficile. De plus, les annotations manuelles sont biaisées vers une tâche spécifique et vers les propres biais de l'annotateur, ce qui peut entraîner des signaux bruités et peu fiables. Il en résulte que les systèmes d'entraînement n'utilisant pas d'annotations pourraient conduire à de meilleures représentations, plus génériques et plus robustes. Dans ce manuscrit, nous présenterons différentes contributions au domaine en pleine croissance de l'apprentissage auto-supervisé de représentations visuelles.

Nous commencerons par étudier une catégorie prometteuse d'approches auto-supervisées, à savoir le clustering profond, qui permet d'entraîner des réseaux de neuronnes tout en trouvant des groupes d'images visuellement cohérents dans une collection de données. Nous identifierons ensuite les limites de ces méthodes de clustering profond telles que leur difficulté à passer à l'échelle ou le fait qu'elles sont souvent sujettes à des solutions triviales. En conséquence, nous proposerons de nouvelles méthodes auto-supervisées qui surpassent leurs homologues supervisées sur plusieurs benchmarks et présentent des propriétés intéressantes. Par exemple, les réseaux auto-supervisés ainsi obtenus contiennent des représentations génériques qui transfèrent bien pour résoudre diverses tâches sur

d'autres ensembles de données. Ils contiennent également des informations explicites sur la segmentation sémantique d'une image. Finalement, nous évaluerons également nos modèles auto-supervisés sur des données brutes, en les entraînant sur des centaines de millions d'images non supervisées sélectionnées aléatoirement sur Internet.

Mots-clés: apprentissage auto-supervisé, vision par ordinateur, apprentissage non-supervisé, apprentissage profond, intelligence artifielle.

Acknowledgments

Acknowledgments are usually keenly anticipated by the reader and they arguably constitute one of the most important part of a PhD manuscript. But writing them is a difficult exercise. Indeed, trying to express the debt of tremendous gratitude I owe to all who have helped making this journey so rewarding and exciting is pretty wild and the following few hundred words will never capture all I want them to.

Without further ado, I would like to thank my supervisors, Piotr, Julien and Armand, for being the best. I feel very proud to have worked and grown with you during these past almost-four years and I cannot thank you enough for everything you have brought to me. Our achievements would simply not have been possible without your guidance, advice, countless ideas, patience and great rigor. Armand, I especially admire your vision, leadership, determination and inclination for setting high expectations throughout our different projects. I also want to thank you for your support and for advocating for me on multiple occasions. Julien I feel very fortunate to have ended up in your team at Thoth, which I think is a great and thriving working environment thanks to the way you manage it. Piotr, I thank you for creating such a relationship of trust between us. I owe you everything and I think you know how crucial your support has been during my PhD...

Finally, I thank all three of you for gathering such an incredible jury for my defense. I cannot say how proud and honored I am to defend in front of researchers whom I admire and whose works have influenced and sometimes even directly seeded the contributions presented in this manuscript. I thank the different jury members for dedicating of their precious time to review the manuscript and attend the defense.

Throughout my PhD, I have had the chance to collaborate with different colleagues at Facebook and I would like to thank them: Ari, Benjamin, Daniel, Ed, Edouard, Gabriel, Herve, Kanika, Lucas, Marc, Matthew, Mike, Nicolas Ballas, Nicolas Carion, Patrick, Sreya, Theo, Timothee and Vasil. At the very beginning of this journey was Matthijs, whom I thank for giving me the chance to intern at FAIR before the PhD while I did not know anything, and for his continued support throughout my PhD. I also want to thank and acknowledge

Priya and Ishan for merging our efforts into a fruitful and exciting collaboration. Cocktails and ski are long overdue ;)! I was also very lucky to collaborate on different projects with other fellow PhD students like Alaa who knows so many papers and always keeps up with the latest approaches; Gautier who patiently taught me the rudiments of NLP; Hugo whose work impresses me so much and who beats me in GPU consumption on the FAIR cluster and Mido who is so nice, inclusive, hardworking and talented.

Overall, I would like to thank all my colleagues at Facebook for creating such an amazing workplace. I have loved each of our conversations and I hope we will stay in touch. In particular, I thank Y-Lan, Joelle and Herve whom I admire a lot and consider as mentors; I thank Laurens, Ross and Antoine for taking the time to discuss my plans after the PhD; I thank Alexis and Camille for believing in me when I was a master student; I thank Alexandre for on-boarding me and answering all my stupid questions during my internship; I thank Jakob, Matthijs and Timothee for making FAIR Grenoble a real thing; I thank Francisco for helping me all the time with PyTorch; I thank Yann LeCun for featuring me in prime-time TF1; last but not least, I thank the entire PhD crew at FAIR Paris and in particular Leonard for his hilarious memes and Pierre, Louis and Othman for collecting the best and the worst from public datasets. Finally I am grateful to the CIFRE program and to Nathalie, Patricia and Pierre-Louis thanks to whom I have had this amazing opportunity to work within FAIR while being a PhD student.

I wish to thank particularly the people who have helped me preparing the *after-PhD* during these last few months. My special thanks go to Louis; I cannot express how valuable your support has been throughout the entire process. I want to thank Jeremy, Timothee and Yana for helping me reaching the bar in coding interviews, I think they remember that I was not anywhere near that point when we started! I also thank Ahmet, Angela, Alexandre, Alexis, Armand, Guillaume, Herve, Ishan, Mido and Piotr for their general advice, feedback and help during this long process.

Throughout these three years at Inria Grenoble, I was lucky to cross the path of amazing PhD students, engineers and researchers. I have really appreciated the friendly and open-minded atmosphere of the Thoth team and I will miss among others the long tables at "Le Champ de l'Etoile", the coffee breaks and the amazing retreats organized by Julien and Nathalie. It is very important to me to thank my day-to-day companions of the team at Inria: Andrei, Đ.Khuê, Daan, Ekaterina, Hubert, Heesung, Jocelyn, Jules, Juliette, Karteek, Lina, Matthieu, Nathalie, Nikita, Pauline, Pierre-Louis, Ricardo, Roman and Zhiqi. I would like to give special thanks to the (extended) "team Juju" founded back in 2018; to Alberto for being so unique and for admonishing raclette-based dermatology advice; to Alexandre and Minttu for being the best caving companions; to Bruno for being an excellent Dixit player

and for singing diligently; to Bulent for enjoyable coffee meetings; to Dexiong for the best karaoke, tea, hotpot and jiaozi sessions and for reminding Valentin and I we couldn't speak Chinese if our lives depended on it; to Enrico for insightful meme explanations; to Florent for being a great office-mate; to Gedeon for seeking the sun; to Gregoire for fervently tagging our mark _|| ||_ everywhere and to ally against our mutual enemy; to Konst for introducing me to (rainy) hikes in Grenoble; to Margot for obviously favoring my defense over her husband's one; to Masha for being extraordinary and for providing the most delicious lemon pie ever; to Michael for always being nice and for fun conversations over baguettes; to Pia who has encouraged me to color my hair; to Pierre for daring to go outside of his comfort zone at the retreat; to Sasha for great wine tasting skills and matching sense of competition; to Timothee for amazing blind test performance; to Theo for short coffee breaks and for supposedly making my life better; to Thomas for exquisite vim configs; to Valentin for dedicating so much effort into optimizing my office ergonomics; to Vlad for Russian candies and political regime debates; and finally I choose not to thank Houssam since he has been trying to organize a ski event the same day as the defense ;). I did not know anyone in Grenoble when arriving three years ago and I realize how fortunate I am to have forged such bonds of friendship and solidarity within the team.

My friends and family have given me a lot of support throughout this journey and I would like to thank in particular Aurelie for always being present; Chloe for her words in difficult times, her constant motivation and the best double dates with Antoine; Corentin for listening to my incessant complaints; Marion for offering me the most welcomed breaks during the PhD; Sedrique for always backing me up and never judging me; Timothee and Lauriane for being awesome; and all my friends of the Kes with a particular shout-out to Gwendo and Fabien for making all the way to Grenoble to cheer on me. I also thank the friends and family members that have supported me throughtout this journey and that I have not mentioned above: Antoine, Clement, Laetitia, Marc, and all my cousins, uncles and aunts; Stephane, Catherine, Boris and the entire Lucas family; and also Dasha, Elodie, Jack, Johanna, Ludo, Mayra, Olivier and Zach. I thank and apologize to the people that I might have failed to mention in these ackowledgements...

Perhaps the most extraordinary part of this PhD journey has been to encounter Thomas, with whom I have shared every single piece of it and who already knows all I would like to tell him. Finally, I thank my dear parents and my beloved sister Chloe a thousand times for their endless support and love throughout all my life.

Chapter 1

Introduction

1.1 Visual Representation Learning

Designing data representation is a major component of many Artificial Intelligence (AI) systems and has evolved significantly in the last decades. The performance of a system depends heavily on the quality of the *data representation* it is given in input. For example, as humans, arithmetic will be more easily and quickly performed if quantities are represented with Arabic numerals rather than binary representations or Roman numerals. Likewise, in Machine Learning (ML), input representations play a central role. For example when a simple machine learning system is tasked to identify the risk for prostate cancer [Stamey et al., 1989], it does not interact with the patient directly. Instead, it inputs a set of different variables such as clinical and demographic conditions gathered by a specialist. This set of variables constitutes the patient *representation* seen by the ML algorithm, which can then make prediction by learning how these different variables interact with each other. However, simple ML algorithms do not act on the representation itself. In this manuscript instead, we will consider more advanced AI systems, able to build their own internal representations from raw input data.

Visual representations. Although studying representations is important in all domains of AI, we focus primarily on computer vision applications in this manuscript and more particularly on image recognition. Generally speaking, our goal is to improve the visual perception of AI systems, with the hope that it matches, or even outperforms, the natural ability of humans to perceive the visual world. Indeed, most living beings can see the world and understand it effortlessly without requiring tremendous amounts of energy, which motivates that visual perception could be realized by a computer. Intuitively, we would like

an AI to reproduce the mental representations that we spontaneously have when looking at a scene. These high-level mental representations allow us to reason and understand what/where/how/who is represented in front of our eyes.

In computer vision, we refer to the representations extracted from raw images by *visual representations*, *features* or *descriptors*. Examples of such features can be specific structures and patterns, like edges, points or objects. They can also be given by a feature extractor function (or "encoder") that maps raw pixel values to a vector of fixed dimension which plays the role of the image representation. Good image featurization is difficult because it is not a well defined problem: it is not clear what information exactly needs to be extracted nor how to extract it. For example, let us imagine we want to create an application that detects dogs in images. Then, it might be a good idea to have the presence of muzzle in the image as a feature. However, describing precisely a muzzle in an image is tedious: it has a well defined shape but muzzle realizations in pixels are infinite due to varying lighting conditions, shadows, occlusions, etc.

The first generation of methods tackling the difficult problem of visual representations includes involved algorithms such as SIFT [Lowe, 2004] and HOG [Dalal and Triggs, 2005] that were obtained by manual design. These handcrafted extraction pipelines produce features with desirable properties like invariance to scale, illumination or rotation.

Deep learning. Another solution is to *learn* the visual representations directly from the data. A major shift of paradigm from machine learning to deep learning is to not only learn how to make prediction given a representation but to learn this representation itself from raw inputs, in an end-to-end fashion. Such learned representations usually result in much better performance compared to the handcrafted ones. They are also inherently more dynamic: deep learning models can learn automatically the set of features useful for a new given task while it can take years for researchers to discover these descriptors by successive trials and errors. Actually, the ability to learn and extract representations is directly baked in the deep neural network architectures used in deep learning. For example, a multi-layer perception (MLP) maps input to output by composition of successive parametrized simple functions. As a result, the output of each function (or "layer") can be thought of as a new representation of the input, which is then fed to the following layer. This way, the AI system can discover for itself intermediate representations that are useful for solving the considered task as illustrated in Figure 1.1. For example, the "XOR" classification problem is a well-known task that cannot be solved by a linear model. Yet, a very simple deep network, namely a MLP, can solve this problem since it has the ability to learn how to transform the data into an intermediate hidden representation that is linearly separable.

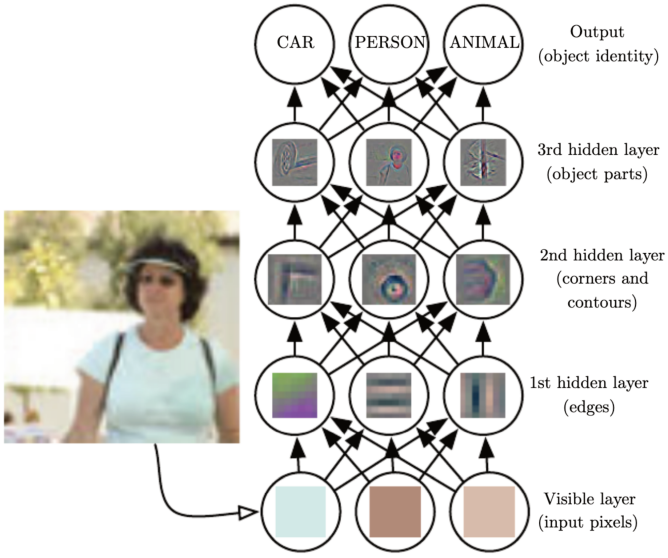


Figure 1.1 – **Illustration of the visual representations learned by deep neural networks.** It is difficult for a simple AI system to solve tasks operating directly on raw pixel values. In contrast, deep networks have the intrinsic ability to build hierarchical representations of increasing complexity. The output of each function (or "layer") can be thought of as a new representation of the input, which is then fed to the following layer. The images in this illustration give an idea of the kind of features captured by each layer. With pixel values as input, the first layer typically detect edges. Given the edges representations of the first layer, the following layer focus on more complex structures such as corners or extended object boundaries; and so on so forth as we move deeper into the network. Figure adapted from Zeiler and Fergus [2014] and Goodfellow et al. [2016].

Supervised representation learning. Learning representations directly from the data with deep learning is a very effective strategy and in the last decade, deep neural networks have become the building blocks of most computer vision applications [Carreira et al., 2016, Chen et al., 2016, Krizhevsky et al., 2012, Ren et al., 2015, Weinzaepfel et al., 2013]. These are trained on large amount of data in a *supervised* manner which means that they process pairs of inputs and desired outputs, called the *annotations* or *labels*. For example, in the case of an AI tasked to detect dogs in images, the supervised learning paradigm requires each training image to come with an annotation indicating the presence of dogs or even their exact localization. These labels are usually obtained thanks to the labor of human annotators [Deng et al., 2009, Everingham et al., 2010], though semi-automatic alternatives exist [Kuznetsova et al., 2020, Mahajan et al., 2018]. Overall, annotations can take on a wide variety of forms, with different degrees of precision and complexity depending on the

Image Classification | Object Detection | Semantic Segmentation | Instance Segmentation

Figure 1.2 – **Supervised learning tasks.** Examples of different types of annotations used in supervised learning tasks with various levels of complexity. Complexity can be thought of as how much time the considered annotation (for example labels, bounding boxes or pixel-level categorization) require an annotator to produce. Figure adapted from Lin et al. [2014] and Lucas [2020].

considered tasks (see Figure 1.2).

A widespread and effective recipe to solve AI tasks is to re-use the representations learned from a large supervised dataset to solve different tasks on other datasets. We can draw an analogy with how, as humans, we constantly re-use and transfer skills in different contexts: for example a tennis champion will likely be better at badminton than somebody who has never practiced any sport before. Likewise, a network trained for a large image classification benchmark like ImageNet [Deng et al., 2009] will perform better when tasked to localize or segment objects compared to the same network initialized from scratch [Sharif Razavian et al., 2014]. This pipeline can be decomposed into two distinct stages: (i) the *pre-training* of the network where (hopefully) general-purpose representations are learned and (ii) the *transfer* where the learned representations are used to solve some downstream tasks. While for the first stage, *supervised* pre-training has been the dominant paradigm in computer vision for a while, we are now witnessing a transition to a new generation of visual pre-trained representations, learned without using any manual annotations by means of *self-supervision*. Interestingly, we remark that this transition is more mature in other fields of AI like Natural Language Processing for example where self-supervised pre-training is now commonplace [Brown et al., 2020, Devlin et al., 2018].

Self-supervised learning. Generally speaking, unsupervised, or *self-supervised* representation learning works by defining a learning task that does not rely on manual labels. For example the task might be to colorize images given greyscale inputs [Zhang et al., 2016], to predict missing words from a sentence [Devlin et al., 2018] or to arrange shuffled frames in a video clip [Misra et al., 2016a]. The solution to these problems does not need to be provided by a human annotator since it is directly a part of the input signal. Performing well on such annotation-free tasks should enable the system to learn structure about the input signal and generic representations that are useful for solving downstream tasks. The

work performed in this thesis is part of a general effort within the research community to improve self-supervised representation learning.

1.2 Why Learning Without Labels?

The goal of self-supervised representation learning is to automatically uncover some structure and understanding from images without having to label them one by one with human annotators. This setting might seem artificial, after all there already *are* large and richly annotated datasets like ImageNet [Deng et al., 2009]. In this section, we argue that studying self-supervised learning is important for several theoretical and practical considerations.

Towards true intelligence. When observing a collection of images, as humans we naturally and autonomously remark that similar structures, concepts and patterns appear across images without the intervention of a supervisor telling us what is to be seen. For instance, when observing photographs of an unknown object, we will be able to recognize it in future photographs, though not to name it, without being explicitly told that these are images representing the same object. Intuitively, we hypothesize that a *truly* intelligent system should be able to learn and reason about the visual world without the intervention of explicit human supervision, but simply by letting the AI observe and perceive the world. This way, studying self-supervised approaches could be a step towards more intelligent and autonomous systems.

Data biases. Another consideration is the fact that annotating data is inherently an ill-defined and ambiguous process. Indeed, there are usually several forms of valid annotations and the arbitrary decisions taken by the annotator in the process induce noise and biases in the dataset. A self-supervised approach should suffer less from this inevitable bias and unreliable signal introduced by collecting supervised data. However, using self-supervised methods can remove the bias in the annotations only and will not impact the biases present in the images themselves. Overall, we hypothesize that pre-training systems without any annotations could lead to more generic and robust representations.

Labels are expensive. From a down-to-earth perspective we observe that getting annotations is almost always expensive, which encourages the development of methods that train without them. Self-supervised training allows to acquire representations from any

specific domain without requiring the tedious design of a large fully-supervised dataset for the considered domain.

For example, in histopathology, we surprisingly observe that most works in the literature [Courtiol et al., 2018] have been relying on the visual representations learned on ImageNet, an object-centric dataset with an over-representation of dogs [Deng et al., 2009]. A natural assumption would be to think that features learned on natural-looking images mostly representing dogs are unlikely to transfer well to the medical domain due to the evident discrepancy between the considered domains. However, in the absence of large annotated datasets of histopathology data and in the absence of mature self-supervised approaches capable of successfully leveraging unlabeled data, this has been the most commonly adopted solution so far [Courtiol et al., 2018]. Hence, we hypothesize that designing better self-supervised approaches could lead to major performance leaps in similar fields where accessing annotations is difficult.

In addition, self-supervised learning may be a more dynamic and flexible strategy compared to the supervised approach. Whenever we encounter a new domain or modality, relying on supervised learning means that we need to constitute a new large annotated dataset. Let us imagine a new image capture modality appears, for instance RGB + a weak sense of depth (this is what dual-lens cameras on mobile phones produce). It would be extremely tedious to select and re-annotate the million of ImageNet images for this modality. Overall, the need for large amounts of annotations have limited the applications of neural networks, which calls for improving methods that work without labels.

Tons of unlabeled data. Last but not least, it is worth noting that unlabeled, raw data *already exist* in tremendous quantities on the Internet: for example, millions of images are uploaded on social media platforms every single day. Leveraging this data with supervised learning would require an unrealistic amount of manual annotations, despite the expert knowledge in crowd-sourcing accumulated by the community over the years. This is not scalable. Another way of using this data is to replace manual annotations by raw metadata, like hashtags or localization coordinates prediction. For example using raw metadata as a supervisory signal has been shown to perform well [Joulin et al., 2016, Sun et al., 2017], even surpassing ImageNet pre-training when trained on billions of images [Mahajan et al., 2018]. However, metadata are not always available, and when they are, they do not necessarily cover the full extent of a dataset. All in all, this motivates the design of methods that can be trained on internet-scale datasets with no supervision at all.

1.3 Outline and Contributions

Let us outline the organization of the manuscript: after a detailed overview of the different methods for visual self-supervised representation learning in Chapter 2, we present the different contributions conducted during this PhD program to the fast-growing field of self-supervised learning.

1.3.1 Deep clustering for representation learning on uncurated data

As we will detail in Chapter 2, most of the research in self-supervised learning has primarily focused on the design of new pretext tasks [Doersch et al., 2015, Dosovitskiy et al., 2016, Noroozi and Favaro, 2016, Zhang et al., 2016]. Several interesting observations and results have emerged from these works but they have not exhibited features competitive with supervised representations obtained from labels classification. This suggests that the task of labels classification is sufficient for pre-training networks, provided that suitable labels are available. We note that in a supervised dataset, the labels partition the images into different groups (i.e. the classes). Based on this observation, in Chapter 3 we build upon self-supervised methods that aim at discovering such groups automatically by means of clustering. More precisely, our work revisits the DeepCluster framework which was published prior to the beginning of this PhD program in Caron [2018], Caron et al. [2018]. DeepCluster is a simple clustering method that jointly learns the parameters of a neural network and the cluster assignments, or "pseudo-labels" of the resulting features. Promising performance was reported in Caron et al. [2018] but most experiments and explorations were conducted in a highly controlled setting by training on ImageNet, a curated dataset made of carefully selected images to form well-balanced and diversified classes [Deng et al., 2009]. Simply discarding the labels does not undo this careful selection, as it only removes part of the human supervision. Because of that, previous works that have experimented with non-curated raw data report a degradation of the quality of features [Caron et al., 2018, Doersch et al., 2015]. The first contribution of this manuscript is to explore visual representations learning from unlabeled and non-curated datasets with deep clustering. We focus on the YFCC100M dataset [Thomee et al., 2015] as a source of uncurated data which contains 99 million images from the Flickr photo-sharing website. This dataset is unbalanced, with a "long-tail" distribution of hashtags contrasting with the well-behaved label distribution of ImageNet. For example in YFCC100M, *guenon* and *baseball* correspond to labels with 1300 associated images in ImageNet, while there are respectively 226 and $256,758$ images associated with these hashtags in YFCC100M. Our goal in Chapter 3 is to understand if trading manually curated data for scale leads to an improvement in the feature quality.

Outline. We start Chapter 3 by giving some preliminaries on DeepCluster [Caron et al., 2018], a method for self-supervised representation learning with clustering published just before this PhD program. Then, we extend DeepCluster to the large-scale training on uncurated data which results in the improved DeeperCluster model. Finally, we establish the limitations of deep clustering, which we will tackle in the following chapter.

Publication. Chapter 3 is largely based on the paper *"Unsupervised pre-training of image features on non-curated data"*, Mathilde Caron, Piotr Bojanowski, Julien Mairal and Armand Joulin, International Conference on Computer Vision, *ICCV 2019* (see [Caron et al., 2019]). The code to reproduce results presented in this chapter is publicly available at `https://github.com/facebookresearch/deepcluster` and at `https://github.com/facebookresearch/DeeperCluster`.

1.3.2 Overcoming deep clustering limitations with SwAV

Models combining representation learning and clustering, i.e. Deep(er)Cluster, learn good representations that improve the performance of neural networks on downstream tasks compared to training from scratch as detailed in Chapter 3. However, deep clustering in its original formulation yields a number of major limitations. Among these is the lack of scalability. Indeed, before each epoch a new clustering is performed to provide pseudo-labels to use during the epoch. This iterative process relies on the size of dataset and does not scale to very large-scale trainings where only one or two epochs are typically performed. A straightforward way to overcome this limitation would be to use an online version of k-means [Zhan et al., 2020]. However, similarly to Asano et al. [2020] and Chen et al. [2020d], we have been questioning the very importance of k-means clustering into our model. Other limitations include the use of empirical tricks to avoid trivial parametrizations or the fact that dependence in data augmentation is crucial but only implicitly and not properly accounted for. We address these different limitations by designing a new model, SwAV, that learns features by directly comparing the descriptors of different crops of a same image. We impose an equipartition constraint of the descriptors thanks to Optimal Transport [Cuturi, 2013] to prevent all images in a mini-batch to collapse to the same representation. Specifically, SwAV simultaneously clusters the data while enforcing consistency between cluster assignments produced for different augmentations (or "views") of the same image. We validate our model by improving the self-supervised state of the art on ImageNet, as well as surpassing supervised pre-training on a large benchmark of transfer tasks and datasets.

Outline. Chapter 4 introduces SwAV, a method for online self-supervised learning based on matching distorted views of a same image. First, we present the method and probe its

performance on ImageNet before assessing SwAV in a more realistic scenario by training on 1 billion uncurated images from Instagram.

Publication. Chapter 4 is primarily based on the paper *"Unsupervised learning of visual features by contrasting cluster assignments"*, Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski and Armand Joulin, Conference on Neural Information Processing Systems, *NeurIPS 2020* (see [Caron et al., 2020]). Code and models for SwAV can be found at `https://github.com/facebookresearch/swav`. This chapter also contains pieces of work publicly available in the technical report *"Self-supervised Pre-training of Visual Features in the Wild"*, Priya Goyal, Mathilde Caron, Benjamin Lefaudeux, Min Xu, Pengchao Wang, Vivek Pai, Mannat Singh, Vitaliy Liptchinsky, Ishan Misra, Armand Joulin and Piotr Bojanowski (see [Goyal et al., 2021]). Note that we also conducted and published an extension of SwAV to semi-supervised learning in *"Semi-Supervised Learning of Visual Features by Non-Parametrically Predicting View Assignments with Support Samples"*, Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Armand Joulin, Nicolas Ballas and Michael Rabbat, International Conference on Computer Vision, *ICCV 2021* (see [Assran et al., 2021]) which we do not present in this manuscript for conciseness.

1.3.3 Improving self-supervised learning with vision transformers

In Chapter 5, we take an orthogonal direction by exploring if we can use the progress in network architectures to improve self-supervised representations. Indeed, transformers [Vaswani et al., 2017] have recently emerged as an alternative to convolutional neural networks (convnets) for visual recognition [Dosovitskiy et al., 2020, Touvron et al., 2020, Zhao et al., 2020]. Their adoption has been coupled with a training strategy inspired by natural language processing (NLP), that is, pre-training on large quantities of data and finetuning on the target dataset [Devlin et al., 2018, Radford et al., 2019]. The resulting Vision Transformers (ViT) [Dosovitskiy et al., 2020] are competitive with convnets but, they currently do not have substantial benefits over convnets: they are computationally more demanding, require more training data, and there is no evidence of particular properties arising in their features.

In this final contribution in Chapter 5, we question if a reason for these setbacks has been the use of supervision when pre-training these networks on images and explore self-supervised pre-training as an alternative. The motivation is that a core feature of BERT pre-training [Devlin et al., 2018] is the use of a self-supervised task where the network learns to predict masked words in a sentence. These words provide a richer signal to learn

from than a single label per sentence. The problem is similar in the case of images since image-level supervision often reduces the diversity in an image to a single concept from a predefined set of a few thousand categories [Russakovsky et al., 2015]. However, as shown in Dosovitskiy *et al*. Dosovitskiy et al. [2020], directly using the BERT pretext task has limited success and may not be the optimal strategy for images. We address this problem by investigating if self-supervised methods originally designed for convnets provide additional benefits to the features produced by ViTs. During this process, we have identified interesting properties that do not emerge with supervised ViTs, nor with convnets:

— Self-supervised ViT features explicitly contain the scene layout and, in particular, object boundaries. This information is directly accessible in the self-attention modules of the last block.

— Self-supervised ViT features perform particularly well with a basic nearest neighbors classifier (k-NN) *without any finetuning, linear classifier nor data augmentation*, achieving 78.3% top-1 accuracy on ImageNet.

Outline. Chapter 5 starts by introducing a new method for self-supervised learning, DINO, specifically designed to work well with vision transformers. We then expose different interesting properties that emerge when training vision transformers in a self-supervised setting. Overall we find that combining vision transformers and self-supervised learning bring considerable boosts of performance in the standard benchmark and reveals intriguing properties.

Publication. Chapter 5 is based on the paper *"Emerging properties in self-supervised vision transformers"*, Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski and Armand Joulin, International Conference on Computer Vision, *ICCV 2021* (see [Caron et al., 2021]). The code is available at `https://github.com/facebookresearch/dino`.

Finally, to wrap up, we summarize in Chapter 6 our contributions and give a (subjective) overview of the current challenges in self-supervised learning.

Chapter 2

Related work

Self-supervised representation learning lives at the intersection of two broad domains of deep learning: unsupervised learning and representation learning. In the words of Goodfellow et al. [2016], the former refers to *most attempts to extract information from a distribution that do not require human labor to annotate examples*. On the other hand, the latter is the problem of learning a good feature extractor from the data. By "good", we mean a function that produces general-purpose visual representations useful for solving downstream tasks. In our work, we choose to instantiate this featurization function with a parametrized deep neural network and we task ourselves to learn its parameters (or *weights*) without using any manual annotations.

The prominent way of *learning* the weights of a neural network is to minimize a cost function using mini-batch stochastic gradient descent [Bottou, 2012] and backpropagation to compute the gradients of the cost with respect to the network weights [LeCun et al., 1998]. In the supervised case, this cost (or *task*, *loss*, *objective*) function is largely defined by the annotations and simply measures how well the output of the network matches the annotated ground truth. However, in the absence of annotations, defining a learning task is an open problem.

A solution is to use a *surrogate* (or *pretext*) task automatically generated from unlabeled data. When trained for this task, the network shall hopefully acquire an understanding of the structure of the scene and objects present in the images and, as a result, produce features that perform well on downstream tasks [Ahmed et al., 2008]. In this section, we give a non-exhaustive overview of the numerous methods developed in the image recognition literature to tackle the challenging problem of designing good label-free pretext tasks.

2.1 Generating Images

First of all, a classical unsupervised task is to ask a deep model to reconstruct its input data, and even to generate new probable inputs. We refer to this class of methods as *generative* approaches since they output in the image space and directly model the data generating function. Intuitively, the generative surrogate task of "*predicting the data*" should encourage the model to understand the data and build useful representations of it. For example, as humans, painting an existing, or imaginary, house requires some mental representation of what a house is and of its structure. Likewise, fitting the data generating distribution shall require a deep model to uncover some underlying structure of the data and to build rich, abstract representations of it. In this section, we briefly describe popular generative models like autoencoders or generative adversarial networks.

2.1.1 Autoencoders

A quintessential example of an unsupervised representation learning approach is the autoencoder model [Bengio et al., 2007, Hinton and Salakhutdinov, 2006, Huang et al., 2007, Masci et al., 2011, Vincent et al., 2008]. It is composed of two parts: an encoder that maps some given input to an intermediate representation, and a decoder that inputs this representation and outputs back into the input space. The two networks are trained together with a reconstruction objective. In other words, the encoded-then-decoded output has be as close as possible as the original input. It is important to constrain somehow the intermediate representation otherwise both networks could trivially learn the identity mapping. Examples of such constraints include dimension bottleneck [Hinton and Salakhutdinov, 2006], data corruption as in denoising autoencoders [Vincent et al., 2008] or information bottleneck as in variational autoencoders (VAE) [Kingma and Welling, 2013].

Interestingly, VAEs belong to a class of generative approaches, namely the *latent variable* models, that propose to explain entirely the data through different and distinct factors of variations. Such factors may be the nature of the represented object, the lighting conditions or the angle of the camera for example. The factors of variations, or latent variables, can be directly viewed as representations of the data as they have the power of describing any image from the data generating function. Follows the *conditional independence* assumption which states that all pixels of an image can be generated at once (i.e. independently from each other) given the knowledge of the image's corresponding latent variables. This is in contrast with the autoregressive models which we mention later in Section 2.1.3 that generate each pixel sequentially given all the previously generated pixels.

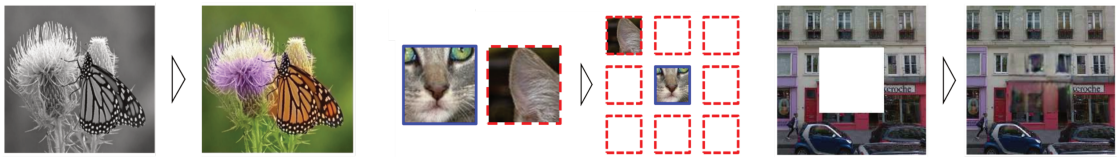
2.1.2 Generative adversarial networks

Another latent variable based generative model is the popular generative adversarial network (GAN) proposed by Goodfellow et al. [2014]. A GAN is made up of two neural networks, namely the generator and the discriminator, that compete against one another. The generator produces images given vectors of the latent space while the discriminator has to determine whether the images it sees are real (i.e. come from the dataset) or not (i.e. have been generated by the generator). This way, the generator is encouraged to produce realistic images in order to fool the discriminator. Interestingly, some experiments on GANs by Radford et al. [2015] have shown that their latent space captures semantic variations in the data distribution and that the representations learned by the generator and discriminator can be transferred to other visual tasks. However, the performance gains are (arguably) moderate. Moreover, it is unclear in a GAN what component of the model to re-use as a feature encoder (the generator maps from latent space to image space and the discriminator is trained to separate real from generated images). To overcome this difficulty, Donahue et al. [2016] and Dumoulin et al. [2016] concurrently propose to enhance the representation learning capacity of the original GAN by adding an auxiliary encoder, which acts as a feature encoder. The purpose of this encoder is to predict directly from images their corresponding semantic representations in the latent space. In other words, it learns the inverse mapping of the generator, which allows to produce competitive visual features. More recently, GAN implementation and training have improved drastically thanks to thorough and large-scale explorations [Brock et al., 2018]. As expected, this has also led to improved performance for their internal representations [Donahue and Simonyan, 2019].

2.1.3 Autoregressive models

Autoregressive methods drop the latent variables strategy and simply generate pixels sequentially based on the previously generated pixels. A successful implementation with convolutional neural networks is the Pixel-CNN of Oord et al. [2016]. Recently, Chen et al. [2020a] have proposed to train a transformer [Vaswani et al., 2017] for the autoregressive next pixel prediction task. The resulting representations have been the most competitive within the generative representation learning family so far.

Overall, generative modeling is a promising and fruitful direction in unsupervised representation learning. However, fitting the data generating function is a very complex learning task with a number of considerable challenges (see [Lucas, 2020] for a comprehensive overview). Moreover, we may wonder if being able to predict *everything* is necessary to learn useful visual representations. For example, the fact that in the supervised case, labels classification is enough to bring about good representations motivates the development of



Colorization
Zhang et al. [2016]

Relative patch position
Doersch et al. [2015]

Inpainting
Pathak et al. [2016]

Figure 2.1 – **Self-supervised pretext tasks obtained by manipulation of data.** Examples of different surrogate tasks built by extracting supervisory signal from the data. The illustrations are adapted from their corresponding publications.

approaches relying on somewhat simpler surrogates that do not require predicting every single pixel of an image. We present in the following different strategies to design non purely-generative unsupervised pretext tasks directly from the input data.

2.2 Pretext Tasks from Data

In this section, we detail methods to extract target labels directly from the data, without requiring human annotators as illustrated in Figure 2.1. Indeed, we will see how we can replace the annotations by the "pseudo-labels" of a pretext task by direct manipulation of the input images or by leveraging some external information like motion or sound for example. The resulting pseudo-labels (or targets) can take a variety of forms. For example, they can be categorical labels associated with a multi-class problem, as when predicting the geometrical transformation of an image [Gidaris et al., 2018, Zhang et al., 2019] or the ordering of a set of patches [Noroozi and Favaro, 2016]. Or they can be continuous variables associated with a regression problem, as when predicting image color [Zhang et al., 2016] or entire patches [Pathak et al., 2016]. Interestingly, we note that like in supervised learning, these pseudo-labels are fixed during training and the quality of the learned features entirely depends on their relevance.

2.2.1 Colorization

First, a quintessential example of self-supervised representation learning is colorization [Larsson et al., 2016, 2017, Zhang et al., 2016]. Given a colorless input (greyscale or lightness), a deep neural network is tasked to recover the image colors ("ab" or "RGB" channels for example). We emphasize that this task is automatically generated from a colored collection of images since the color information is already present and hence does

not need to be provided by annotators. Intuitively, performing well on the surrogate colorization task requires a semantic understanding of the extent of objects and scenes present in the images, which encourages the network to learn high level features. However, color does not always exist in images (in the medical domain for example) and consequently Zhang et al. [2017] have proposed to recast colorization to the more generic scenario of cross-channel prediction.

2.2.2 Spatial cues

Another landmark strategy in self-supervised learning is to manipulate and exploit the spatial layout of images to generate surrogate tasks [Doersch et al., 2015]. For instance, inspired by word2vec [Mikolov et al., 2013] in NLP, Doersch et al. [2015] propose to train a network to predict the relative position of a pair of patches from the same image as illustrated in the middle panel of Figure 2.1. Intuitively, this difficult task should force the network to learn rich representations of object parts and their spacial arrangement. For example, being able to predict that a car wheel patch is usually located below a windscreen patch requires some understanding of the concept of vehicles. A number of follow-up works [Kim et al., 2018, Lee et al., 2017, Mundhenk et al., 2018, Santa Cruz et al., 2017] have proposed several extensions and refinements along the seminal work of Doersch et al. [2015]. Of particular interest, Noroozi and Favaro [2016] extend the relative patch position prediction task to 9 patches, sampled on a 3×3 grid in the image. The task of the network is to predict the permutation applied to the set of shuffled patches. Goyal et al. [2019] show that making this task even more complex by increasing the number of possible patch permutations usually improves the quality of the resulting representations.

Another self-supervised approach exploiting spatial cues is the *inpainting* task (see right panel of Figure 2.1). Pathak et al. [2016] propose to train a network to generate an image region from its surroundings. A network encodes the context of the image into a compact latent representation while a decoder inputs that latent representation and fills in the missing image patch pixels. The connection between these two networks is performed with a channel-wise fully connected layer that allows propagation of information across channels. The authors use a combination of two different losses: (i) a reconstruction loss that tends to give blurry patches since it is safer to predict the mean of the distribution, and (ii) an adversarial loss that encourages sharpest and more realistic results.

Overall, these works exploiting spatial layout in images have particularly inspired one of the contribution of this thesis, the multi-crop training, which we present in Chapter 4 (see Section 4.2.4).

Gidaris et al. [2018]

Pathak et al. [2017]

Figure 2.2 – **(left): RotNet.** Rotation prediction with the self-supervised approach RotNet. **(right): Unsupervised segmentation.** From left to right, we show (i) a video frame, (ii) the pseudo ground truth segmentation masks obtained with an offline unsupervised segmentation algorithm and (iii) the segmentation learned by the self-supervised deep network. Interestingly, while the input segmentations are often noisy and inaccurate, the network manages to produce significantly better and smoother segmentations. Figures adapted from their corresponding papers.

2.2.3 Predicting geometric transformations

Another popular self-supervised surrogate task is to recognize the geometric transformation applied to an input image [Gidaris et al., 2018, Zhang et al., 2019]. In particular, Gidaris et al. [2018] have shown that, exploiting the bias in the way humans generally take pictures, good features can be obtained by training a deep network to discriminate between different image rotations. This pretext task corresponds to a multi-class classification problem with four categories: rotations in $\{0°, 90°, 180°, 270°\}$. Each input is randomly rotated and associated with a target that represents the angle of the applied rotation as illustrated in the left panel of Figure 2.2. The authors argue that in order for a network to recognize the rotation applied to an image it requires to understand the objects depicted in the image, such as their location, their type or their pose. Despite its extreme simplicity, the rotation prediction task enables to learn good representations as further empirically demonstrated in the work of Kolesnikov et al. [2019]. There have been several follow-up works in this direction [Feng et al., 2019, Qi et al., 2019], for example combining RotNet with generative modeling [Chen et al., 2019] or clustering [Caron et al., 2019].

2.2.4 Learning from motion

Other works propose to exploit motion (from videos for example) to design self-supervised pretext tasks. For example, echoing the direction on geometric transformations

prediction, Agrawal et al. [2015] train a network to predict the camera transformation between pairs of images. Another example is the work of Jayaraman and Grauman [2015] that learn image embedding from unlabeled video. Starting from egocentric video together with observer ego-motion signals, they train a system on a "view prediction" task, to learn equivariant visual features that respond predictably to observer ego-motion. In this target equivariant feature space, pairs of images related by the same ego-motion are related by the same feature transformation too. Similarly, Xiong et al. [2021] encourage the features to obey the same flow transformation as input image pairs.

Taking a different approach, Wang and Gupta [2015] propose a task where patches from a same video should be close in feature space. More precisely, from an anchor patch in a video, they form a positive pair by tracking it along thirty consecutive frames and form a negative pair with a random (or more sophistically selected) patch. The task of the network is to bring the positive pair closer together and to pull away the negative pair. The work of Wang and Gupta [2015] echoes the view-invariant features paradigm which we describe later in Section 2.3 and uses motion as a way of generating different "views" of a common concept.

A different approach is that of Pathak et al. [2017] who use motion to mine unsupervised segmentations from videos. Then, a deep network is trained to predict those "ground truth" segmentations from images. As shown in the corresponding paper and in Figure 2.2 (right), the unsupervised segmentations are noisy and of quite poor quality. However, because they do not exhibit consistent or systematic error patterns, they may be seen as perturbations around a good quality segmentation. The motivation of Pathak et al. [2017] is to hypothesize that a deep network will not fit this noise but output something closer to the underlying correct segmentation. This is qualitatively verified in Figure 2.2 (right) where we observe that the learned segmentations are of better quality that the noisy ground-truth segmentations used during training.

Though promising, a pitfall of these approaches relying on motion is the dependency on auxiliary information or offline algorithms, which might make them impractical in some cases. For example, if metadata about the camera is not de facto present in a visual collections, it might turn out to be as least as tedious to collect as the manual annotations used in supervised learning.

2.2.5 Other cues

Many other cues have been exploited in order to design self-supervised tasks, for example audio [Owens et al., 2016], counting instances [Noroozi et al., 2017] or physical interaction [Pinto et al., 2016]. Interestingly, Li et al. [2016] leverage the first generation of handcrafted visual representations to learn representations with deep networks. Concretely,

they build a graph of nearest neighbors on SIFT [Lowe, 1999] and Fisher Vector [Sánchez et al., 2013] descriptors. From this graph, they infer positive image pairs from cycle consistency and negative image pairs from large geodesic distances (accumulated weights along the shortest path between the two considered nodes). The task of the network is to learn an embedding in which images semantically similar (positive pairs) are close while images semantically different (negative pairs) are distant.

2.2.6 Combining cues

The nature of what the features capture and focus on is likely to differ from a self-supervised surrogate task to another. As a result, different strategies for combining multiple cues have been explored [Doersch and Zisserman, 2017, Wang et al., 2017]. Wang et al. [2017] propose to construct an affinity graph with two types of edges: inter-instance invariance (similar viewpoint) and intra-instance invariance (similar instance). Inter-instance edges are found with the relative patch prediction task of Doersch et al. [2015] while intra-instance edges are mined through tracking a patch along thirty frames of a video like in Wang and Gupta [2015]. From this affinity graph, they infer positive (resp. negative) pairs that need to be brought together (resp. pulled away) by the network. Doersch and Zisserman [2017] also aim at leveraging the diversity of the learned representation by combining different pretext tasks. A novelty in the approach of Doersch and Zisserman [2017] is to use a form of regularization that encourages the network to separate group of features useful for different tasks. In other words, the network determines which combination of layers to use for each of the pretext tasks.

Overall, we have described a number of diverse self-supervised tasks proposed in the literature in the past years. Lastly, we will describe in more details self-supervised approaches that encourage view-invariant representations [Dosovitskiy et al., 2014]. This paradigm has been the most successful lately [Caron et al., 2020, Chen et al., 2020b, Grill et al., 2020, Misra and Maaten, 2020] and is the seed to most of the contributions presented in this manuscript [Caron et al., 2018, 2019, 2020, 2021].

2.3 View-Invariant Features

The seminal work of Dosovitskiy et al. [2014] propose to learn representations that model view-invariant factors. This is motivated by the fact that the way we view a scene does not usually affect its semantic content. For example, the meaning or purpose of an image of a dog is most of the time preserved regardless of the position of the camera

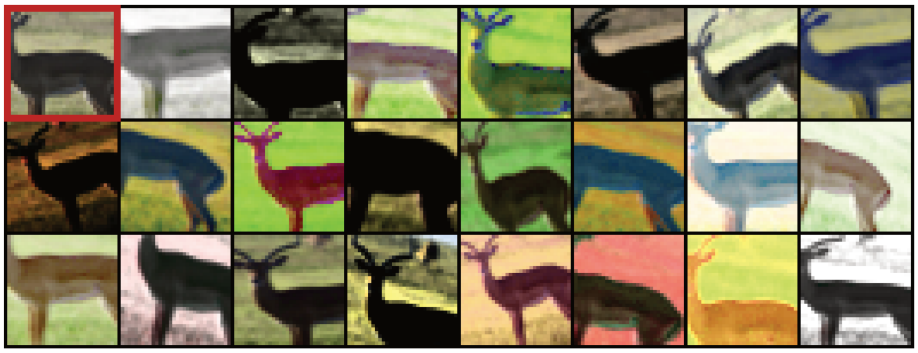


Figure 2.3 – **Illustration of the view-invariant learning paradigm.** Different data transformations are applied to an original image of the dataset (here framed in red). A deep network is trained to produce features invariant to these input distortions. Figure adapted from Dosovitskiy et al. [2014].

(high-angle or low-angle shots for instance) or the lighting conditions. Luckily for us, many factors of variation in natural looking images can be easily simulated with data transformations (see Figure 2.3).

Data augmentation has primarily been used in the context of supervised learning to augment the size of a dataset by creating additional "fake" training data. It consists in generating different versions (or "views", "crops", "patches") of a given image by means of transformations like translation, rotation, jitering, scaling, cropping, etc. In the context of supervised learning, one has to pay attention to apply operations that preserve the label of the original image. For example, optical character recognition tasks require recognizing the difference between "b" and "d", so horizontal flips might not be appropriate ways of augmenting datasets for these tasks. Overall, in supervised learning, data augmentation is a well-known and effective regularization technique to improve the generalization of models.

Dosovitskiy et al. [2014] operate a crucial shift of paradigm: data augmentation is not merely considered as a regularization tool but becomes a central component of the model, enabling systematic ways to design self-supervised prediction tasks. In other words, the hypothesis of Dosovitskiy et al. [2014] takes data augmentation to the extreme and uses it to automatically generate a surrogate task encouraging embeddings invariant to distortions of the input sample. Overall, this work on view-invariant representations has greatly influenced and inspired the work presented in this manuscript as well as numerous works in the self-supervised community which we describe in the remaining of this section.

2.3.1 Original parametric implementation

Dosovitskiy et al. [2014, 2016] implement the view-invariant paradigm as an instance classification approach. The problem is formulated as an "N-way" classification problem where N is the size of the dataset. In other words, each image acts as a distinct class and a deep network is trained to discriminate between them up to data augmentations. More concretely, let us denote by f_θ the neural network mapping, where θ is the set of corresponding parameters. We refer to the vector obtained by applying this mapping to an image as feature or representation. Given a training set $X = \{x_1, x_2, \dots, x_N\}$ of N images, we recall that our goal is to find a parameter θ^* such that the mapping f_{θ^*} produces good general-purpose features. In the context of supervised learning, each image x_n is associated with a label y_n in $[0, k-1]$ which represents the image's membership to one of k possible predefined classes. A parametrized classifier g_W predicts the correct labels on top of the features $f_\theta(x_n)$. The parameters W of the classifier and the parameter θ of the mapping are then jointly learned by minimizing the following per-example negative log likelihood loss on temperature-softmax outputs:

$$\min_{\theta,W} -g_W\left(f_\theta(x_n)\right)_{y_n} / \tau + \log \sum_{j=0}^{k-1} \exp\left(g_W\left(f_\theta(x_n)\right)_j / \tau\right) \tag{2.1}$$

Note that we explicit here the per-example loss only and in practice we minimize its expectation over the training dataset. In the work of Dosovitskiy et al. [2014], a random transformation t is applied to each original dataset image x_n. We denote by $\mathcal{T}$ the set of all possible data transformations. Overall, the following per-example instance classification objective is minimized:

$$\min_{\theta,W} \mathbb{E}_{t\sim\mathcal{T}}\left[-g_W\left(f_\theta(t(x_n))\right)_n / \tau + \log \sum_{j=0}^{k-1} \exp\left(g_W\left(f_\theta(t(x_n))\right)_j / \tau\right)\right], \tag{2.2}$$

where the label y_n of each transformed image is simply the id n of the original image. Explicitly learning a classifier g_W to discriminate between all images does not scale well with the number of images N. For example on ImageNet, W would be of size $1,281,167 \times d$ where d is the feature dimension. To overcome this major limitation, a number of approaches propose to use a noise contrastive estimator (NCE) [Gutmann and Hyvärinen, 2010] to directly compare instances instead of classifying them [Wu et al., 2018]. These methods are collected under the umbrella of "contrastive" representation learning.

2.3.2 Contrastive implementations

Contrastive methods drop learning a parametrized instance-level classifier g_W altogether. Instead, the task becomes to retrieve positive pairs of features out of a pool of negative pairs. Concretely, given an anchor feature representation $f_\theta(t(x_n))$, a positive pair is formed by featurizing a different view of the same instance x_n (denoted v^+). Negative pairs are formed by featurizing a set v^- of k different instances from the dataset. Contrastive implementations like that of Wu et al. [2018], He et al. [2020] or Chen et al. [2020b] minimize a per-example objective of the following form:

$$\min_{\theta,W} \mathbb{E}_{t\sim\mathcal{T}} \left[-f_\theta(t(x_n))^\intercal v^+/\tau + \log \sum_{v\in v^-} \exp\left(f_\theta(t(x_n))^\intercal v/\tau\right) \right], \tag{2.3}$$

where the different feature vectors are normalized. This loss can be interpreted as the log loss of a "$|v^-|+1$"-way softmax-based classifier that tries to classify the anchor as v^+. This contrastive implementation requires comparing features from a large number of images simultaneously and different strategies have been proposed to that end. For example, Wu et al. [2018] store positive element v^+ and negative elements v^- into a memory bank composed of computations from the previous epochs. As a result, this approach relies on the size of the dataset and does not scale to very large-scale trainings. He et al. [2020] propose an online memory bank strategy as a workaround. Computations from the previous iterations are stored in a rolling memory bank. Positive and negatives features are computed with an auxiliary dedicated momentum network which is an exponential moving average version of the main network. This ensures consistency between the features of the memory bank. Another approach is to simply use the examples from the same mini-batch as positive and negatives [Chen et al., 2020b]. Contrastive loss functions can also be based on other forms, such as margin-based losses and variants of NCE losses [Bachman et al., 2019, Henaff, 2020, Hjelm et al., 2018, Oord et al., 2018, Tian et al., 2020a, Wang and Gupta, 2015].

2.3.3 Other implementations

A caveat of the contrastive approach is that it requires comparing features from a large number of images simultaneously to have good performance. Interestingly, a number of approaches have implemented the "view-invariant" principle without falling back on the contrastive formulation. Among these methods are the SwAV and DINO contributions presented respectively in Chapters 4 and 5.

At the root of this thesis is the work of Bojanowski and Joulin [2017] which implements the instance discrimination problem by means of discriminative clustering [Bach and

Harchaoui, 2008]. Discriminative clustering optimizes a grouping of the data into distinct clusters that, if used as pseudo-labels to learn a classifier, produces the maximum of separability (as measured by the loss value at an optimal classifier). Note that it is crucial to add constraints to avoid the trivial solution where all examples have the same pseudo-label. Different classifiers can be used, leading to different implementations of the discriminative clustering framework. For instance, Bach and Harchaoui [2008] use a linear classifier with a ridge regression loss. Bojanowski and Joulin [2017] use a deep network that classifies into a set of N fixed targets uniformly arranged on the unit sphere in feature space. This way, the loss encourages representations to be uniformly spread out in the feature space. The approach iterates between training the network to classify the examples into their assigned targets and obtaining new target assignments for the representations. As different views are used for assigning and predicting the targets, the model trains implicitly for view-invariant representations. This work has influenced greatly the contributions presented in this manuscript (Chapters 3 and 4).

Interestingly, recent works have shown that we can instantiate the view-invariant learning paradigm without explicitly discriminating between images. Grill et al. [2020] propose a metric-learning formulation called BYOL, where features are trained by matching them to representations obtained with a momentum encoder. It has been shown that methods like BYOL work even without a momentum encoder, at the cost of a drop of performance [Chen and He, 2020, Grill et al., 2020]. We note that the contribution presented in Chapter 5 completes the interpretation initiated in BYOL of self-supervised learning as a form of Mean Teacher self-distillation [Tarvainen and Valpola, 2017] with no labels. Several other works echo this direction, showing that one can train features by using whitening [Ermolov et al., 2020] or redundancy reduction of the features [Zbontar et al., 2021]. The latter work proposes a loss function that measure the cross-correlation matrix between the representations of two distorted versions of a sample, and makes it as close to the identity matrix as possible. This causes the embedding vectors of distorted versions of a sample to be similar, while minimizing the redundancy between the components of these vectors.

2.3.4 Grouping variants

We have seen that most of the implementations of the view-invariant paradigm follow a *per-instance* discrimination framework. However, pulling representations from different instances apart can imper the learned embeddings if it turns out that these instances are actually semantically similar. This limitation was already identified in the analysis of Dosovitskiy et al. [2014]. As a workaround, they propose to use clustering to group the instances with strong similarities into a single pseudo-class instead of having a pseudo-class per instance as in their original formulation. Their results show that this variant improves

performance of the model and foresee that *potentially, using even more data or performing clustering and network training within a unified framework could further improve the quality of the learned features*. This analysis has influenced greatly the development of deep clustering approaches which we detail in Chapter 3.

Other works propose to model inter-examples similarities thanks to clustering. Some methods rely explicitly on a clustering loss [Asano et al., 2020, Caron et al., 2018, Xie et al., 2016, Yang et al., 2016] to learn simultaneously image features and clusters. The approach of Yang et al. [2016] iterates over a clustering and a training stages. The former optimizes a grouping of the representations given by the network. The latter trains the network to classify the images into the pseudo-labels given by the group assignments. Yang et al. [2016] use agglomerative clustering, which starts with a large number of very small clusters and merges them progressively. However, Douze et al. [2017] show that this choice of clustering method is not optimal when working with large-scale datasets.

2.3.5 Importance of spatial cues

An effective direction to improve performance of the view-invariant paradigm is to consider transformations targeting distortions of the spatial layout of an image (i.e. cropping and scaling for example). Indeed, many works have exploited spatial cues to induce desired invariants or properties in the representations [Hjelm et al., 2018, Misra and Maaten, 2020, Oord et al., 2018]. For example, Hjelm et al. [2018] and Bachman et al. [2019] define a surrogate task where local representations of an image (i.e. representation of a small image portion) need to be predicted from a global image descriptor (i.e. representation of a large portion) obtained with another view of that image. Besides, Oord et al. [2018] and Hénaff et al. [2019] encourage matching of adjacent non-overlapping views of a same image. Interestingly, Chen et al. [2020b] highlight the importance of the random scaling and cropping data augmentation (`torchvision.transforms.RandomResizedCrop` in PyTorch [Paszke et al., 2019]) and show that using it to generate different matching views naturally encompasses the two learning tasks above (i.e. "global-to-local" and "adjacent" view predictions) and many more. Intuitively, encouraging "crop-invariant" features is an interesting idea. Concretely, it means that a crop of an image needs to be predicted from another crop of the same image. To do so, the network has to either output a constant representation for every possible input (which is prevented by different constraints or designs) or to actually output a representation about that image. Importantly, this representation needs to be about the image in general and not just about the particular considered crop since it then needs to be predicted from another completely independent crop of that same image. Based on that observation, one contribution of this thesis is to encourage *holistic* "local-to-global" matching of the views, which we show to be very

effective for downstream classification tasks (see details of multi-crop in Section 4.2.4).

2.3.6 Other variants and refinements

Finally, a number of variants and refinements have been proposed to the original view-invariant self-supervised paradigm. For instance, some works [Kalantidis et al., 2020, Zhuang et al., 2019] propose different strategies to mine hard negatives in the contrastive learning implementation (Section 2.3.2). Other works ablate and analyze the different invariant injected through the data augmentation pipeline [Purushwalkam and Gupta, 2020, Tian et al., 2020b] or propose to match more elaborate representations, e.g., Bag-of-Words [Gidaris et al., 2020a,b]. Other related directions, beyond the scope of this manuscript, are the focus on designing *dense* local representations [Jabri et al., 2020, Pinheiro et al., 2020], or the use of different modalities as a way of generating the different "views" of a common concept [Alayrac et al., 2020, Wang and Gupta, 2015].

Chapter 3

Deep Clustering

We have seen in Chapter 2 that research in self-supervised learning has primarily mainly focused on generative modeling [Radford et al., 2015] or on designing surrogate tasks for which the labels can be computed automatically by manipulation of the input data [Doersch et al., 2015, Dosovitskiy et al., 2016, Noroozi and Favaro, 2016, Zhang et al., 2016]. These works pioneered the field of visual self-supervision but, at the time of their publication, showed moderate results compared to that of pre-training with supervised ImageNet classification. This suggests that the task of *classification* is promising for representation learning, given that labels are provided to that end.

A trivial but important observation is to see that the labels of ImageNet partitions the dataset into distinct groups (i.e. the classes). The seed of this chapter is to ask whether we can find those groups automatically by means of *clustering*. As a matter of fact, clustering, a class of unsupervised learning methods, has been extensively applied and studied in computer vision. However, only little work has been done to adapt it to the end-to-end training of visual features on large scale datasets. In this chapter, we start by giving some preliminaries on deep clustering with DeepCluster, a clustering method presented at ECCV in Munich [Caron et al., 2018] (prior to the PhD program) that jointly learns the parameters of a neural network and the cluster assignments of the resulting features.

Second, we study representation learning with deep clustering in an *uncurated* scenario. We train our model on 96 million images from YFCC100M [Thomee et al., 2015], achieving state-of-the-art results among unsupervised methods on standard benchmarks, which confirms the potential of self-supervised learning in a realistic scenario, when only uncurated raw data are available. This effort was presented at ICCV in Seoul [Caron et al., 2019] and is the base of this chapter.

3.1 Introduction

3.1.1 Self-supervised representation learning with clustering

Pre-trained convolutional neural networks, or convnets, have become the building blocks in most computer vision applications [Carreira et al., 2016, Chen et al., 2016, Ren et al., 2015, Weinzaepfel et al., 2013]. They produce excellent general-purpose features that can be used to improve the generalization of models learned on a limited amount of data [Sharif Razavian et al., 2014]. The existence of ImageNet [Deng et al., 2009], a large fully-supervised dataset, has been fueling advances in pre-training of convnets. However, Stock and Cisse [2018] have presented empirical evidence that the performance of state-of-the-art classifiers on ImageNet is largely underestimated, and little error is left unresolved. This explains in part why the performance has been saturating despite the numerous novel architectures proposed in recent years [Chen et al., 2016, Dosovitskiy et al., 2020, El-Nouby et al., 2021, He et al., 2015, Huang et al., 2016]. As a matter of fact, ImageNet is relatively small by today's standards; it "only" contains a million images that cover the specific domain of object classification. A natural way to move forward is to build a bigger and more diverse dataset, potentially consisting of billions of images. This, in turn, would require a tremendous amount of manual annotations, despite the expert knowledge in crowd-sourcing accumulated by the community over the years. Replacing labels by raw metadata leads to biases in the visual representations with unpredictable consequences [Misra et al., 2016b]. This calls for methods that can be trained on internet-scale datasets with no supervision.

Unsupervised learning has been widely studied in the Machine Learning community [Friedman et al., 2001], and algorithms for clustering, dimension reduction or density estimation are regularly used in computer vision applications [Joulin et al., 2010, Shi and Malik, 2000, Turk and Pentland, 1991]. For example, the "bag of features" model uses clustering on handcrafted local descriptors to produce good image-level features [Csurka et al., 2004]. A key reason for their success is that they can be applied on any specific domain or dataset, like satellite or medical images, or on images captured with a new modality, like depth, where annotations are not always available in quantity. Several works have shown that it was possible to adapt unsupervised methods based on density estimation or dimension reduction to deep models [Goodfellow et al., 2014, Kingma and Welling, 2013], leading to promising all-purpose visual features [Bojanowski and Joulin, 2017, Donahue et al., 2016]. Despite the primeval success of clustering approaches in image classification, very few works [Xie et al., 2016, Yang et al., 2016] have been proposed to adapt them to the end-to-end training of convnets, and never at scale. An issue is that clustering methods have been primarily designed for linear models on top of fixed features,

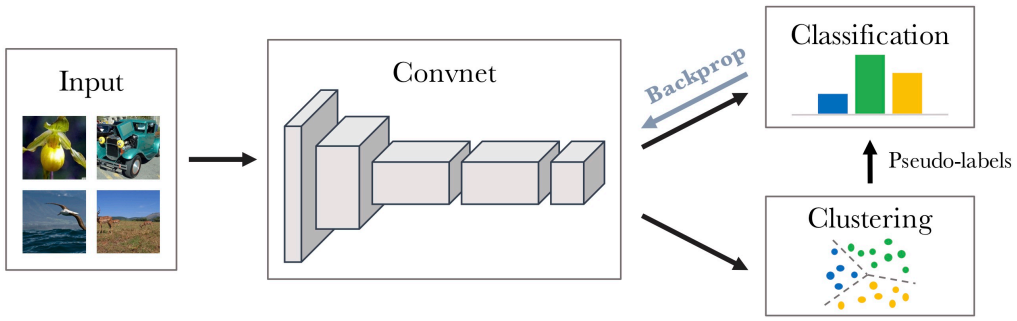


Figure 3.1 – **Illustration of DeepCluster.** We iteratively cluster deep features and use the cluster assignments as pseudo-labels to learn the parameters of the convnet.

and they scarcely work if the features have to be learned simultaneously. For example, learning a convnet with k-means would lead to a trivial solution where the features are zeroed, and the clusters are collapsed into a single entity.

In preliminaries, we review DeepCluster, a self-supervised approach that show that it is possible to obtain useful general-purpose visual features with a clustering framework. DeepCluster, summarized in Figure 3.1, consists in alternating between clustering of the image descriptors and updating the weights of the convnet by predicting the cluster assignments. For simplicity, we focus on k-means in this chapter, but other clustering approaches can be used, like Power Iteration Clustering (PIC) [Lin and Cohen, 2010]. The overall pipeline is sufficiently close to the standard supervised training of a convnet to reuse many common tricks [Ioffe and Szegedy, 2015]. Despite its simplicity, DeepCluster achieves significantly higher performance than the state of the art among self-supervised methods on standard transfer tasks at the time of the publication (i.e. 2018).

3.1.2 Training on uncurated data

However, these first results are obtained by training DeepCluster on ImageNet (without labels). Designing large fully-annotated datasets like ImageNet has required a significant effort from the community, not only due to the cost of manual labeling but also due to the compilation and cleansing of the data. As a result, we discuss the use of ImageNet, a highly curated dataset, as a training set for unsupervised models. It contains properly cropped images of a pre-selected number of classes, and nothing else. As it is designed for a fine-grained classification task, this dataset provides well-balanced and diversified data. Discarding the labels does not undo this careful selection, and only removes part of the human supervision.

The challenge tackled is this chapter is to learn good general-purpose representations

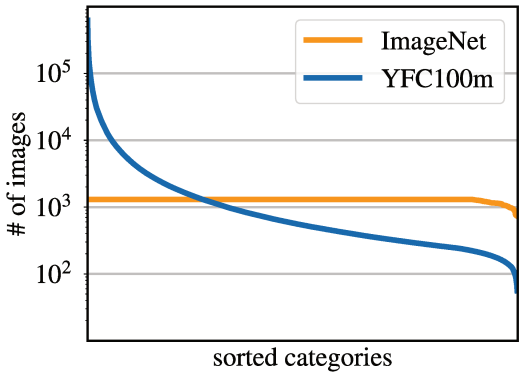


Figure 3.2 – **Comparison of the hashtag distribution in YFCC100M** [Thomee et al., 2015] **with the label distribution in ImageNet** [Deng et al., 2009]. YFCC100M dataset contains social media from the Flickr website. The content of this dataset is very unbalanced, with a "long-tail" distribution of hashtags contrasting with the well-behaved label distribution of ImageNet. For example, *guenon* and *baseball* correspond to labels with 1300 associated images in ImageNet, while there are respectively 226 and 256, 758 images associated with these hashtags in YFCC100M.

from large-scale raw data available on the Internet. This is an effort towards building representations that can be learned on any dataset, without requiring it to be annotated or curated. This way, one could adapt the feature learning dataset to the visual domain of the transfer task dataset. In this work, we choose to focus on social data, publicly and easily available in bulk on the Internet. Specifically, we use data from YFCC100m dataset [Thomee et al., 2015] which contains 96 million images from the Flickr photo sharing website. This dataset is dramatically unbalanced and ill-conditioned with a "long-tail" distribution (see Figure 3.2). Some recent efforts have been made to learn from weak supervisory signal provided by GPS coordinates [Weyand et al., 2016] or tags [Joulin et al., 2016, Mahajan et al., 2018] associated with the data. In contrast, in this work we show that it is actually possible to learn features with good transferability from the raw, uncurated and unlabeled data only.

To that end, we propose to extend DeepCluster specifically for leveraging large amount of raw data. Indeed, we hypothesize that training on large-scale non-curated data requires (i) model complexity to increase with dataset size; (ii) model stability to data distribution changes. A simple yet effective solution is to combine methods from two domains of self-supervision: clustering and pretext task by data manipulation. Since clustering methods, like DeepCluster, build supervision from *inter-image* similarities, the task at hand becomes inherently more complex when the number of images increases. In addition, DeepCluster

captures finer relations between images when the number of clusters scales with the dataset size. Clustering approaches infer target labels at the same time as features are learned. Thus, target labels evolve during training, making clustering-based approaches unstable. Furthermore, these methods are sensitive to data distribution as they rely directly on cluster structure in the underlying data. Explicitly dealing with unbalanced category distribution might be a solution but it assumes that we know the distribution of the latent classes. We design our method without this assumption. On the other hand, other methods use pretext task by predicting pseudo-labels automatically extracted from input signals [Doersch et al., 2015]. For instance, methods like RotNet [Gidaris et al., 2018] (described in Section 2.2.3), leverage *intra-image* statistics to build supervision, which are often independent of the data distribution. However, the dataset size has little impact on the nature of the task and on the performance of the resulting features. A solution to leveraging larger datasets require manually increasing the difficulty of the self-supervision task [Goyal et al., 2019]. Our approach automatically increases complexity through the clustering strategy.

Our approach, DeeperCluster, automatically generates targets by clustering the features of the entire dataset, under constraints derived from the rotation prediction pretext task. Due to the "long-tail" distribution of raw non-curated data, processing huge datasets and learning a large number of targets is necessary, making the problem challenging from a computational point of view. For this reason, we propose a hierarchical formulation that is suitable for distributed training. This enables the discovery of latent categories present in the "tail" of the image distribution.

3.1.3 Related work

Several approaches related to our work learn deep networks with no supervision and are detailed in Chapter 2. Here, we give a brief overview of methods that combine clustering and unsupervised learning with deep networks or approaches that train on large-scale uncurated data.

Clustering and deep unsupervised learning. Coates and Ng [2012] also use k-means to pre-train convnets, but learn each layer sequentially in a bottom-up fashion, while we do it in an end-to-end fashion. Other clustering losses [Dosovitskiy et al., 2014, Liao et al., 2016, Xie et al., 2016, Yang et al., 2016] have been considered to jointly learn convnet features and image clusters but they have never been tested on a scale to allow a thorough study on modern convnet architectures. Liao et al. [2016] combines the autoencoder reconstruction loss (detailed in Section 2.1.1) with a clustering regularization term. They benchmark different clustering strategies: Spatial clustering corresponds to the grouping of the pixels in

a given layer over all localizations and examples; channel co-clustering allows to group the different feature maps (channels) in a given layer across examples; and sample clustering refers to the clustering of the set of representations given by a network over the whole dataset. Of particular interest, Yang et al. [2016] iteratively learn convnet features and clusters with a recurrent framework. Their model offers promising performance on small datasets but may be challenging to scale to the number of images required for convnets to be competitive. Closer to our work, Bojanowski and Joulin [2017] learn visual features on a large dataset with a loss that attempts to preserve the information flowing through the network [Linsker, 1988]. Their approach discriminates between all images in a similar way as exemplar SVM [Malisiewicz et al., 2011], while we are simply clustering them.

Learning on non-curated datasets. Some methods [Chen and Gupta, 2015, Gomez et al., 2017, Ni et al., 2015, Tian et al., 2021] aim at learning visual features from non-curated data streams. They typically use metadata such as hashtags [Joulin et al., 2016, Sun et al., 2017] or geolocalization [Weyand et al., 2016] as a source of noisy supervision. In particular, Mahajan et al. [2018] train a network to classify billions of Instagram images into predefined and clean sets of hashtags. They show that with little human effort, it is possible to learn features that transfer well to ImageNet, even achieving state-of-the-art performance if finetuned. As opposed to our work, they use an extrinsic source of supervision that had to be cleaned beforehand.

3.2 Preliminaries: DeepCluster

We start this chapter by re-introducing the DeepCluster framework which is the seed to the contributions presented in the following of the chapter.

3.2.1 DeepCluster methodology

After a short introduction to the supervised learning of convnets, we describe the DeepCluster approach as well as the specificities of its optimization.

Preliminaries. Modern approaches to computer vision, based on statistical learning, require good image featurization. In this context, convnets are a popular choice for mapping raw images to a vector space of fixed dimension. When trained on enough data, they constantly achieve the best performance on standard classification benchmarks [He et al., 2015, Krizhevsky et al., 2012]. We denote by f_θ the convnet mapping, where θ is the set of corresponding parameters. We refer to the vector obtained by applying this mapping to an

image as feature or representation. Given a training set $X = \{x_1, x_2, \dots, x_N\}$ of N images, we want to find a parameter θ^* such that the mapping f_{θ^*} produces good general-purpose features.

These parameters are traditionally learned with supervision, i.e. each image x_n is associated with a label y_n in $\{0,1\}^k$. This label represents the image's membership to one of k possible predefined classes. A parametrized classifier g_W predicts the correct labels on top of the features $f_\theta(x_n)$. The parameters W of the classifier and the parameter θ of the mapping are then jointly learned by optimizing the following problem:

$$\min_{\theta,W} \frac{1}{N} \sum_{n=1}^{N} \ell\left(g_W\left(f_\theta(x_n)\right), y_n\right), \tag{3.1}$$

where ℓ is the multinomial logistic loss, also known as the negative log-softmax function. This cost function is minimized using mini-batch stochastic gradient descent [Bottou, 2012] and backpropagation to compute the gradient [LeCun et al., 1998].

Unsupervised learning by clustering. When θ is sampled from a Gaussian distribution, without any learning, f_θ does not produce good features. However the performance of such random features on standard transfer tasks, is far above the chance level. For example, a multi-layer perceptron classifier on top of the last convolutional layer of a random AlexNet achieves 12% in accuracy on ImageNet while the chance is at 0.1% [Noroozi and Favaro, 2016]. The good performance of random convnets is intimately tied to their convolutional structure which gives a strong prior on the input signal. The idea of this work is to exploit this weak signal to bootstrap the discriminative power of a convnet. We cluster the output of the convnet and use the subsequent cluster assignments as "pseudo-labels" to optimize Eq. (3.1). This deep clustering (DeepCluster) approach iteratively learns the features and groups them.

Clustering has been widely studied and many approaches have been developed for a variety of circumstances. In the absence of points of comparisons, we focus on a standard clustering algorithm, k-means. Preliminary results with other clustering algorithms indicates that this choice is not crucial. k-means takes a set of vectors as input, in our case the features $f_\theta(x_n)$ produced by the convnet, and clusters them into k distinct groups based on a geometric criterion. More precisely, it jointly learns a $d \times k$ centroid matrix C and the cluster assignments y_n of each image n by solving the following problem:

$$\min_{C \in \mathbb{R}^{d \times k}} \frac{1}{N} \sum_{n=1}^{N} \min_{y_n \in \{0,1\}^k} \|f_\theta(x_n) - C y_n\|_2^2 \quad \text{such that} \quad y_n^\top 1_k = 1. \tag{3.2}$$

Solving this problem provides a set of optimal assignments $(y_n^*)_{n \leq N}$ and a centroid matrix

C^*. These assignments are then used as pseudo-labels; we make no use of the centroid matrix.

Overall, DeepCluster alternates between clustering the features to produce pseudo-labels using Eq. (3.2) and updating the parameters of the convnet by predicting these pseudo-labels using Eq. (3.1). This type of alternating procedure is prone to trivial solutions; we describe how to avoid such degenerate solutions in the next section.

Avoiding trivial solutions. The existence of trivial solutions is not specific to the unsupervised training of neural networks, but to any method that jointly learns a discriminative classifier and the labels. Discriminative clustering suffers from this issue even when applied to linear models [Xu et al., 2005]. Solutions are typically based on constraining or penalizing the minimal number of points per cluster [Bach and Harchaoui, 2008, Joulin and Bach, 2012]. These terms are computed over the whole dataset, which is not applicable to the training of convnets on large scale datasets. In the following paragraphs, we briefly describe the causes of these trivial solutions and give simple and scalable workarounds.

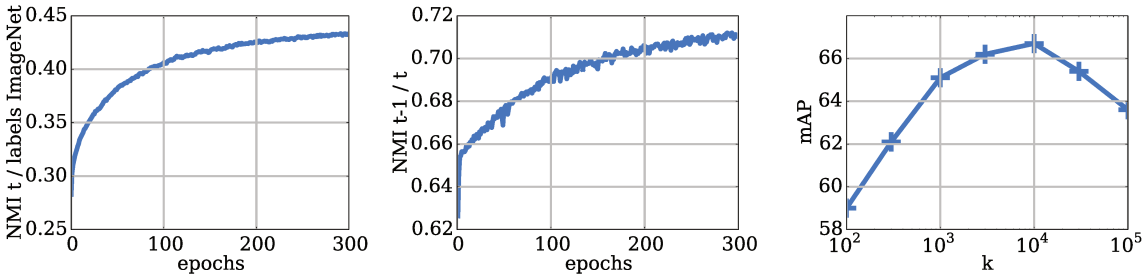
A discriminative model learns decision boundaries between classes. An optimal decision boundary is to assign all of the inputs to a single cluster [Xu et al., 2005]. This issue is caused by the absence of mechanisms to prevent from empty clusters and arises in linear models as much as in convnets. A common trick used in feature quantization [Johnson et al., 2017] consists in automatically reassigning empty clusters during the k-means optimization. More precisely, when a cluster becomes empty, we randomly select a non-empty cluster and use its centroid with a small random perturbation as the new centroid for the empty cluster. We then reassign the points belonging to the non-empty cluster to the two resulting clusters.

If the vast majority of images is assigned to a few clusters, the parameters θ will exclusively discriminate between them. In the most dramatic scenario where all but one cluster are singleton, minimizing Eq. (3.1) leads to a trivial parametrization where the convnet will predict the same output regardless of the input. This issue also arises in supervised classification when the number of images per class is highly unbalanced. For example, metadata, like hashtags, exhibits a Zipf distribution, with a few labels dominating the whole distribution [Joulin et al., 2016]. A strategy to circumvent this issue is to sample images based on a uniform distribution over the classes, or pseudo-labels. This is equivalent to weight the contribution of an input to the loss function in Eq. (3.1) by the inverse of the size of its assigned cluster.

Implementation details. For comparison with previous works at the time of the publication of DeepCluster paper, we use a standard AlexNet [Krizhevsky et al., 2012] architecture. It consists of five convolutional layers with 96, 256, 384, 384 and 256 filters; and of three fully connected layers. We remove the Local Response Normalization layers and use batch normalization [Ioffe and Szegedy, 2015]. We also consider a VGG-16 [Simonyan and Zisserman, 2014] architecture with batch normalization. Unsupervised methods often do not work directly on color and different strategies have been considered as alternatives [Doersch et al., 2015, Noroozi and Favaro, 2016]. We apply a fixed linear transformation based on Sobel filters to remove color and increase local contrast [Bojanowski and Joulin, 2017, Paulin et al., 2015]. We train DeepCluster on ImageNet [Deng et al., 2009] unless mentioned otherwise. It contains $1,281,167$ images uniformly distributed into $1,000$ classes. We cluster the central cropped images features and perform data augmentation (random horizontal flips and crops of random sizes and aspect ratios) when training the network. This enforces invariance to data augmentation which turn out to be crucial for DeepCluster and relates to the view-invariant self-supervised paradigm [Dosovitskiy et al., 2014] detailed in Section 2.3. The network is trained with dropout [Srivastava et al., 2014], a constant step size, an ℓ_2 penalization of the weights θ and a momentum of 0.9. Each mini-batch contains 256 images. For the clustering, features are PCA-reduced to 256 dimensions, whitened and ℓ_2-normalized. We use the k-means implementation of Johnson et al. [2017]. Note that running k-means takes a third of the time because a forward pass on the full dataset is needed. One could reassign the clusters every n epochs, but we found out that our setup on ImageNet (updating the clustering every epoch) was nearly optimal. On Flickr, the concept of epoch disappears: choosing the trade-off between the parameter updates and the cluster reassignments is more subtle. On ImageNet, we train the models for 500 epochs, which takes 12 days on a Pascal P100 GPU for AlexNet. We select hyperparameters on a down-stream task, i.e., object classification on the validation set of PASCAL VOC with no fine-tuning.

3.2.2 Probing DeepCluster on ImageNet

We compare our approach to previous state-of-the-art models on standard benchmarks on Pascal VOC dataset before studying the behavior of DeepCluster during training. Finally, we qualitatively assess the filters learned with DeepCluster. Note that in this section all models are trained on ImageNet without labels which is a highly curated dataset. While it provides a controlled setting, allows easier comparison with previous state of the art and helps understanding the impact of the labels on the performance of a network, one has to keep in mind the limitations of such a setting as described in Section 3.1.2. We study the uncurated scenario in the following section of this chapter (i.e. Section 3.3).



(a) Clustering quality (b) Cluster reassignment (c) Influence of k

Figure 3.3 – **DeepCluster preliminary studies.** (a): evolution of the clustering quality along training epochs; (b): evolution of cluster reassignments at each clustering step; (c): validation mAP classification performance for various choices of k.

Transfer on PASCAL VOC. We do a quantitative evaluation of DeepCluster on image classification, object detection and semantic segmentation on PASCAL VOC. The relatively small size of the training sets on PASCAL VOC ($2,500$ images) makes this setup closer to a "real-world" application, where a model trained with heavy computational resources, is adapted to a task or a dataset with a small number of instances. Detection results are obtained using `fast-rcnn`[1]; segmentation results are obtained using the code of Shelhamer *et al.*[2]. For classification and detection, we report the performance on the test set of PASCAL VOC 2007 and choose our hyperparameters on the validation set. For semantic segmentation, following the related work, we report the performance on the validation set of PASCAL VOC 2012. Table 3.1 summarized the comparisons of DeepCluster with other feature-learning approaches on the three tasks. We observe that DeepCluster outperforms previous unsupervised methods (published prior the publication of this work) on all three tasks, in every setting. The improvement with fine-tuning over the state of the art is the largest on semantic segmentation (7.5%). On detection, DeepCluster performs only slightly better than previously published methods. Interestingly, a fine-tuned random network performs comparatively to many unsupervised methods, but performs poorly if only FC6-8 are learned. For this reason, we also report detection and segmentation with FC6-8 for DeepCluster and a few baselines. These tasks are closer to a real application where fine-tuning is not possible. It is in this setting that the gap between our approach and the state of the art is the greater (up to 9% on classification).

Relation between clusters and labels. We measure the information shared between two different assignments A and B of the same data by the Normalized Mutual Information

1. https://github.com/rbgirshick/py-faster-rcnn 2. https://github.com/shelhamer/fcn.berkeleyvision.org

	Classification		Detection		Segmentation	
Method	FC6-8	ALL	FC6-8	ALL	FC6-8	ALL
ImageNet labels	78.9	79.9	–	56.8	–	48.0
Random-rgb	33.2	57.0	22.2	44.5	15.2	30.1
Random-sobel	29.0	61.9	18.9	47.9	13.0	32.0
Pathak et al. [2016]	34.6	56.5	–	44.5	–	29.7
Donahue et al. [2016]*	52.3	60.1	–	46.9	–	35.2
Pathak et al. [2017]	–	61.0	–	52.2	–	–
Owens et al. [2016]*	52.3	61.3	–	–	–	–
Wang and Gupta [2015]*	55.6	63.1	32.8†	47.2	26.0†	35.4†
Doersch et al. [2015]*	55.1	65.3	–	51.1	–	–
Bojanowski and Joulin [2017]*	56.7	65.3	33.7†	49.4	26.7†	37.1†
Zhang et al. [2016]*	61.5	65.9	43.4†	46.9	35.8†	35.6
Zhang et al. [2017]*	63.0	67.1	–	46.7	–	36.0
Noroozi and Favaro [2016]	–	67.6	–	53.2	–	37.6
Noroozi et al. [2017]	–	67.7	–	51.4	–	36.6
DeepCluster	**72.0**	**73.7**	**51.4**	**55.4**	**43.2**	**45.1**

Table 3.1 – **Comparing DeepCluster with the state of the art.** Comparison of DeepCluster to state-of-the-art (prior to 2018) self-supervised feature learning on classification, detection and segmentation on PASCAL VOC. Description of the concurrent self-supervised approaches are detailed in Chapter 2. * indicates the use of the data-dependent initialization of Krähenbühl et al. [2015]. †: numbers for other methods produced by us.

(NMI), defined as:

$$\mathrm{NMI}(A; B) = \frac{\mathrm{I}(A; B)}{\sqrt{\mathrm{H}(A)\mathrm{H}(B)}}$$

where I denotes the mutual information and H the entropy. This measure can be applied to any assignment coming from the clusters or the true labels. If the two assignments A and B are independent, the NMI is equal to 0. If one of them is deterministically predictable from the other, the NMI is equal to 1. Figure 3.3(a) shows the evolution of the NMI between the cluster assignments and the ImageNet labels during training. It measures the capability of the model to predict class level information. Note that we only use this measure for this analysis and not in any model selection process. The dependence between the clusters and the labels increases over time, showing that our features progressively capture information

	Clustering algorithm	Classification		Detection		Segmentation	
		FC6-8	ALL	FC6-8	ALL	FC6-8	ALL
DeepCluster	k-means	72.0	73.7	51.4	55.4	43.2	45.1
DeepCluster	PIC	71.0	73.0	53.6	54.4	42.4	43.8

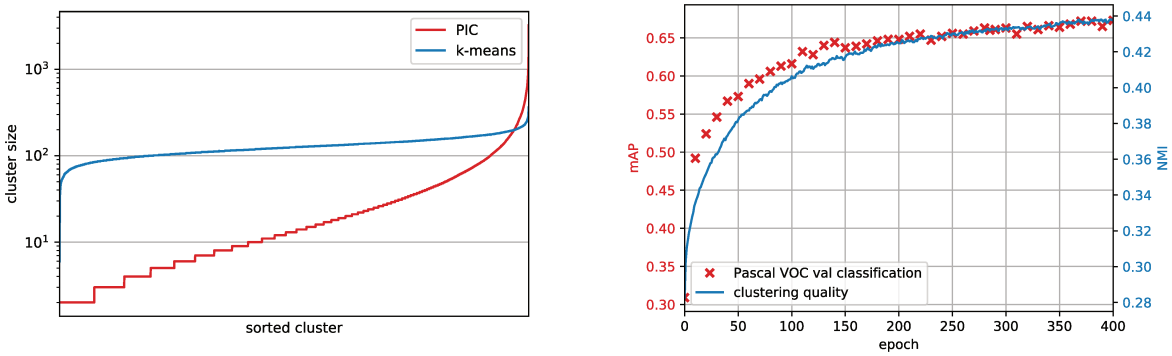
Table 3.2 – **Impact of changing the clustering algorithm.** Evaluation of PIC versus k-means for DeepCluster on PASCAL VOC transfer tasks.

related to object classes.

Number of reassignments between epochs. At each epoch, we reassign the images to a new set of clusters, with no guarantee of stability. Measuring the NMI between the clusters at epoch $t-1$ and t gives an insight on the actual stability of our model. Figure 3.3(b) shows the evolution of this measure during training. The NMI is increasing, meaning that there are less and less reassignments and the clusters are stabilizing over time. However, NMI saturates below 0.8, meaning that a significant fraction of images are regularly reassigned between epochs. In practice, this has no impact on the training and the models do not diverge.

Choosing the number of clusters. We measure the impact of the number k of clusters used in k-means on the quality of the model. We report the same down-stream task as in the hyperparameter selection process, i.e. mAP on the PASCAL VOC 2007 classification validation set. We vary k on a logarithmic scale, and report results after 300 epochs in Figure 3.3(c). The performance after the same number of epochs for every k may not be directly comparable, but it reflects the hyper-parameter selection process used in this work. The best performance is obtained with $k = 10,000$. Given that we train our model on ImageNet, one would expect $k = 1000$ to yield the best results, but apparently some amount of over-segmentation is beneficial.

Second, we report in Table 3.2 the results for the different PASCAL VOC transfer tasks with a model trained with the PIC version of DeepCluster. For this set of transfer tasks, the models trained with k-means and PIC versions of DeepCluster perform in comparable ranges.



(a) k-means versus PIC

(b) Feature evolution

Figure 3.4 – (a): **Different cluster distributions.** Sizes of clusters produced by the k-means and PIC versions of DeepCluster at the last epoch of training. (b): **Quality of the features as a function of the amount of training.** In red: validation mAP PASCAL VOC classification performance. In blue: evolution of the clustering quality.

Alternative clustering algorithm. We consider Power Iteration Clustering (PIC) [Lin and Cohen, 2010] as an alternative clustering method. It has been shown to yield good performance for large scale collections by Douze et al. [2017].

Since PIC is a graph clustering approach, we generate a nearest neighbor graph by connecting all images to their 5 neighbors in the Euclidean space of image descriptors. We denote by $f_\theta(x)$ the output of the network with parameters θ applied to image x. We use the sparse graph matrix $G = \mathbb{R}^{n \times n}$. We set the diagonal of G to 0 and non-zero entries are defined as

$$w_{ij} = e^{-\frac{\|f_\theta(x_i) - f_\theta(x_j)\|^2}{\sigma^2}}$$

with σ a bandwidth parameter. In this work, we use a variant of PIC [Douze et al., 2017] that does:

1. Initialize $v \leftarrow [1/n, ..., 1/n]^\top \in \mathbb{R}^n$;
2. Iterate
$$v \leftarrow N_1(\alpha(G + G^\top)v + (1 - \alpha)v),$$
where $\alpha = 10^{-3}$ is a regularization parameter and $N_1 : v \mapsto v/\|v\|_1$ the L1-normalization function;
3. Let G' be the directed unweighted subgraph of G where we keep edge $i \to j$ of G such that
$$j = \text{argmax}_j w_{ij}(v_j - v_i).$$

If v_i is a local maximum (ie. $\forall j \neq i, v_j \leq v_i$), then no edge starts from it. The clusters are given by the connected components of G'. Each cluster has one local maximum.

An advantage of PIC clustering is not to require the setting beforehand of the number of clusters. However, the parameter σ influences the number of clusters: when it is larger, the edges become more uniform and the number of clusters decreases, and the other way round when σ increased. In the following, we set $\sigma = 0.2$.

First, we give an insight about the distribution of the images in the clusters. We show in Figure 3.4(a) the sizes of the clusters produced by the k-means and PIC versions of DeepCluster at the last epoch of training (this distribution is stable along the epochs). We observe that k-means produces more balanced clusters than PIC. Indeed, for PIC, almost one third of the clusters are of a size lower than 10 while the biggest cluster contains roughly 3000 examples. In this situation of very unbalanced clusters, it is important in our method to train the convnet by sampling images based on a uniform distribution over the clusters to prevent the biggest cluster from dominating the training.

Stopping criterion. We monitor how the features learned with DeepCluster evolve along the training epochs on a down-stream task: object classification on the validation set of PASCAL VOC with no fine-tuning. We use this measure to select the hyperparameters of our model as well as to check when the features stop improving. In Figure 3.4(b), we show the evolution of both the classification accuracy on this task and a measure of the clustering quality (NMI between the cluster assignments and the true labels) throughout the training. Unsurprisingly, we notice that the clustering and features qualities follow a similar dynamic.

Influence of data pre-processing with Sobel filtering In this section we experiment with our method on raw RGB inputs and evaluate the impact of the Sobel filtering. The difficulty of learning convnets on raw images has been noted before [Bojanowski and Joulin, 2017, Doersch et al., 2015, Noroozi and Favaro, 2016, Paulin et al., 2015]. We provide some insights into the reasons why Sobel filtering (illustrated in Figure 3.5) is crucial to obtain good performance with our method.

In our preliminary experiments we have observed that the performance of DeepCluster on raw RGB images degrades significantly. For example, training a MLP on top of the DeepCluster frozen pre-trained features on an AlexNet (following the experiment protocol of Noroozi and Favaro [2016]) lead to 44.0% top-1 accuracy with Sobel filtering and 37.1% without (-6.9%). In order to better understand why this happens, in Figure 3.6 we randomly select a subset of 3000 clusters and sort them by standard deviation to their mean color. If the standard deviation of a cluster to its mean color is low, it means that the images

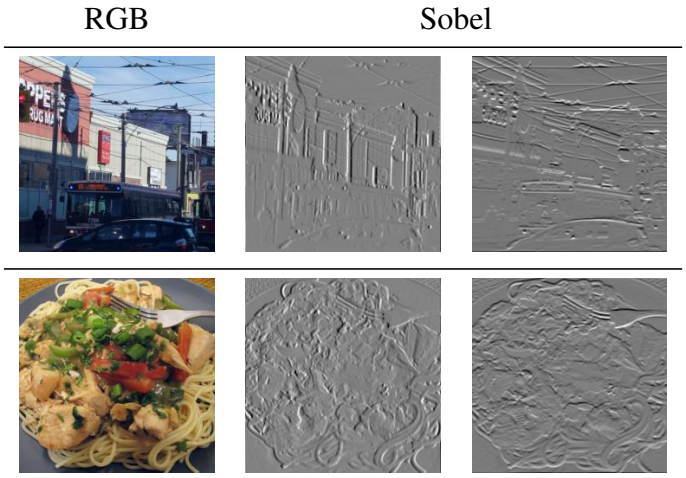


Figure 3.5 – **Illustration of Sobel filtering.** Visualization of two images pre-processed with Sobel filter. Sobel gives a 2 channels output which at each point contain the vertical and horizontal derivative approximations.

of this cluster tend to have a similar colorization. Indeed, we show in Figure 3.6 (right) examples of these clusters that have a low standard deviation to the mean color. We observe in Figure 3.6 (left) that the clustering on features learned with DeepCluster focuses much more on color than the clustering performed on other self-supervised approach (namely RotNet [Gidaris et al., 2018]). Indeed, clustering by color and low-level information produces balanced clusters that can easily be predicted by a convnet. As a result, clustering by color constitutes a solution to DeepCluster formulation. However, as we want to avoid an uninformative clustering essentially based on colors, we remove some part of the input information by feeding the network with the image gradients instead of the raw RGB image (see Figure 3.5). Interestingly, we have observed that using Sobel filter during RotNet training has almost no impact. Finally, note that more recent approaches replace Sobel filtering by strong color jittering augmentation [Chen et al., 2020b].

Visualizing first layer filters. Figure 3.7 shows the filters from the first layer of an AlexNet trained with DeepCluster on raw RGB images and images pre-processed with a Sobel filtering. As shown in the left panel of Figure 3.7, most filters capture only color information that typically plays a little role for object classification [Van De Sande et al., 2010]. Filters obtained with Sobel pre-processing act like edge detectors.

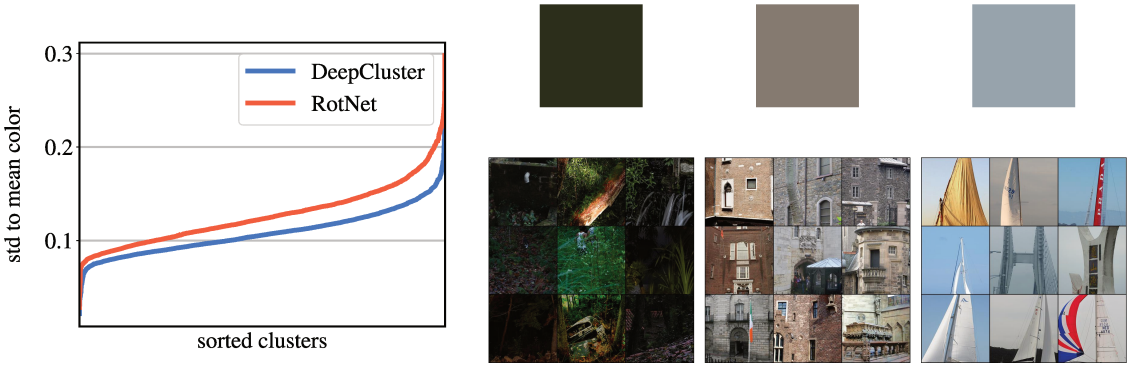


Figure 3.6 – **Analysis of the influence of the color information in the clusters produced by DeepCluster. (left)** Sorted standard deviations to clusters mean colors. If the standard deviation of a cluster to its mean color is low, the images of this cluster have a similar colorization. **(right)** Examples of clusters with an uniform colorization across their images. For each cluster, we display the corresponding mean color of the cluster.

Visualizing filters from deeper layers. We assess the quality of the representations learned by the VGG-16 convnet with DeepCluster. To do so, we learn an input image that maximizes the mean activation of a target filter [Erhan et al., 2009, Zeiler and Fergus, 2014]. We follow the process described by Yosinski et al. [2015] with a cross entropy function between the target filter and the other filters of the same layer. Starting from Gaussian noise, we iteratively update the input to maximize the response of a target filter, applying on the image a Gaussian blur every 5 updates and a ℓ_2 penalization. In Figure 3.8, we show these synthetic images as well as the 9 top activated images from a subset of 1 million images from YFCC100M. We observe that the filters, learned without any supervision, capture quite large and complex structures. In addition, in Figure 3.9, we display synthetic images that correspond to filters that seem to focus on human characteristics.

3.3 Training on Uncurated Images

In this section, we focus on self-supervised learning on uncurated data, which is closer to a real-world scenario than training on ImageNet without labels. First, we describe how we extend DeepCluster to deal with large uncurated content. Second, we show some results when training Deep**er**Cluster on $96M$ images from Flickr. Finally, we assess the quality and properties of the resulting clustering.

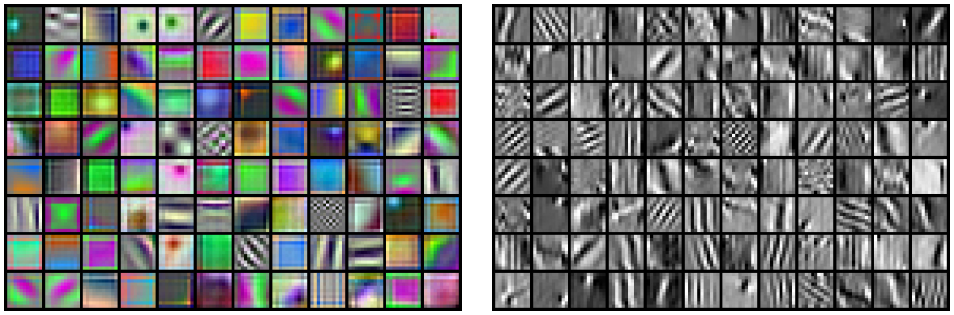


Figure 3.7 – **Visualizing first layer filters.** Filters from the first layer of an AlexNet trained on unsupervised ImageNet on raw RGB input (left) or after a Sobel filtering (right).

3.3.1 DeeperCluster methodology

In this section, we describe how we extend deep clustering in order to scale up to large numbers of uncurated images and targets. To that end, we propose to combine self-supervised learning based on pretext tasks (for intra-image modeling) and on clustering (for inter-image modeling).

Combining pretext tasks and clustering. We assume that the inputs $x_1, \dots, x_N$ are rotated images, each associated with a target label r_n encoding its rotation angle and a cluster assignment y_n. The cluster assignment changes during training along with the visual representations. We denote by $\mathcal{R}$ the set of possible rotation angles and by $\mathcal{Y}$, the set of possible cluster assignments. A way of combining rotation prediction with deep clustering is to add the corresponding losses. However, summing these losses implicitly assumes that classifying rotations and cluster memberships are two independent tasks, which may limit the signal that can be captured. Instead, we work with the Cartesian product space $\mathcal{R} \times \mathcal{Y}$, which can potentially capture richer interactions between the two tasks. We get the following optimization problem:

$$\min_{\theta, W} \frac{1}{N} \sum_{n=1}^{N} \ell(r_n \otimes y_n, W f_\theta(x_n)), \tag{3.3}$$

where ℓ is a loss function. Note that any clustering or self-supervised approach with a multi-class objective can be combined with this formulation. For example, we could use a self-supervision task that captures information about tiles permutations [Noroozi and Favaro, 2016] or frame ordering in a video [Wang and Gupta, 2015]. However, this formulation does not scale in the number of combined targets, i.e., its complexity is $O(|\mathcal{R}||\mathcal{Y}|)$. This limits the use of a large number of cluster or a pretext task with a large output space [Zhang

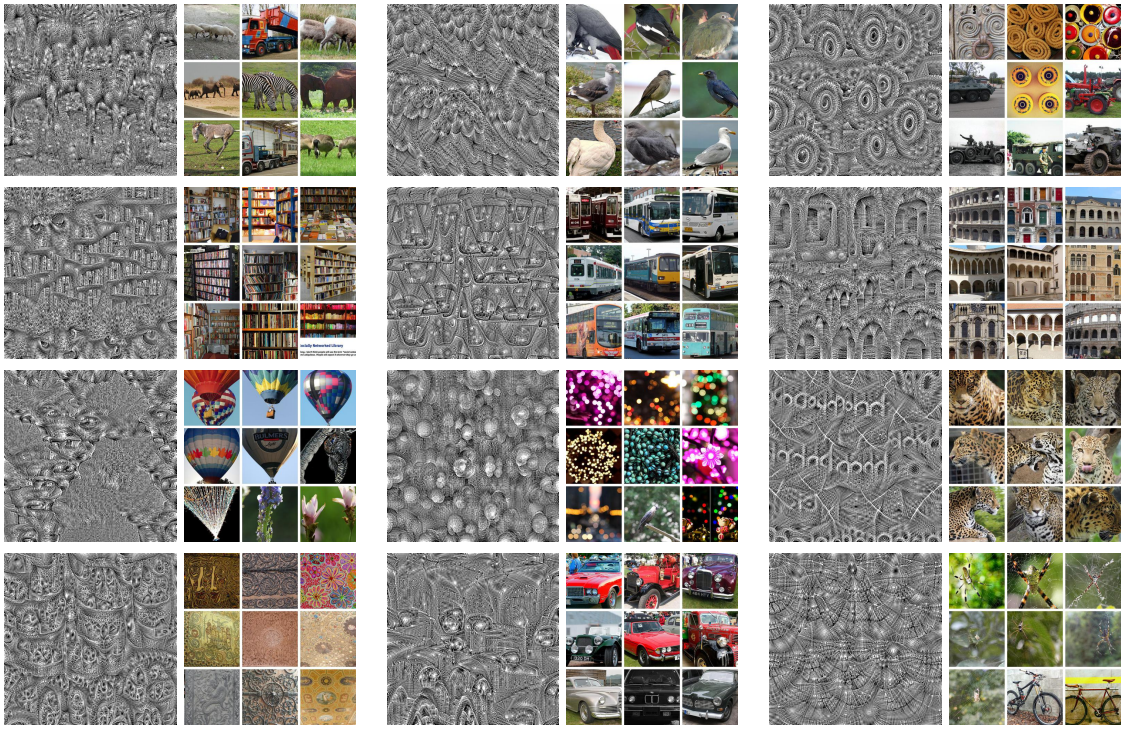


Figure 3.8 – **Visualizing filters from deeper layers.** Filter visualization and top 9 activated images (immediate right to the corresponding synthetic image) from a subset of 1 million images from YFCC100M for target filters in the last convolutional layer of a VGG-16 trained with DeepCluster.

et al., 2019]. In particular, if we want to capture information contained in the tail of the distribution of non-curated dataset, we may need a large number of clusters. We thus propose an approximation of our formulation based on a scalable hierarchical loss that it is designed to suit distributed training.

Scaling up to large number of targets. Hierarchical losses are commonly used in language modeling where the goal is to predict a word out of a large vocabulary [Brown et al., 1992]. Instead of making one decision over the full vocabulary, these approaches split the process in a hierarchy of decisions, each with a smaller output space. For example, the vocabulary can be split into clusters of semantically similar words, and the hierarchical process would first select a cluster and then a word within this cluster.

Following this line of work, we partition the target labels into a 2-level hierarchy where we first predict a super-class and then a sub-class among its associated target labels. The first

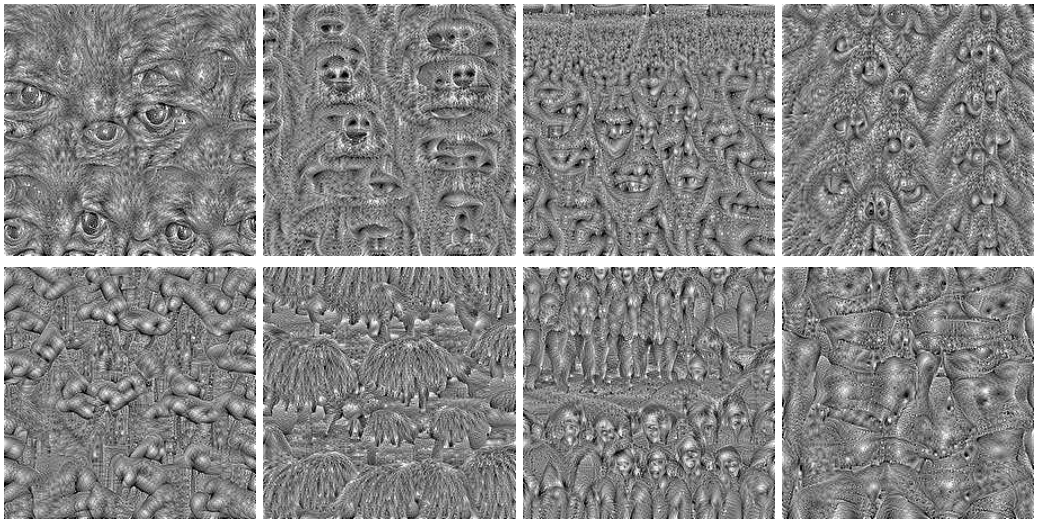


Figure 3.9 – **Visualizing filters from deeper layers with human attributes.** Filter visualization by learning an input image that maximizes the response to a target filter [Yosinski et al., 2015] in the last convolutional layer of a VGG-16 convnet trained with DeepCluster. Here, we manually select filters that seem to trigger on human characteristics (eyes, noses, faces, fingers, fringes, groups of people or arms).

level is a partition of the images into S super-classes and we denote by y_n the super-class assignment vector in $\{0,1\}^S$ of the image n and by y_{ns} the s-th entry of y_n. This super-class assignment is made with a linear classifier V on top of the features. The second-level of the hierarchy is obtained by partitioning *within each super-class*. We denote by z_n^s the vector in $\{0,1\}^{k_s}$ of the assignment into k_s sub-classes for an image n belonging to super-class s. There are S sub-class classifiers $W_1, \dots, W_S$, each predicting the sub-class memberships within a super-class s. The parameters of the linear classifiers $(V, W_1, \dots, W_S)$ and θ are jointly learned by minimizing the following loss function:

$$\frac{1}{N}\sum_{n=1}^{N}\left[\ell\Big(V f_\theta(x_n), y_n\Big) + \sum_{s=1}^{S} y_{ns}\ell\left(W_s f_\theta(x_n), z_n^s\right)\right], \tag{3.4}$$

where ℓ is the negative log-softmax function. Note that an image that does not belong to the super-class s does not belong either to any of its k_s sub-classes.

Choice of super-classes. A natural partition would be to define the super-classes based on the target labels from the self-supervised task and the sub-classes as the labels produced by clustering. However, this would mean that each image of the entire dataset would be

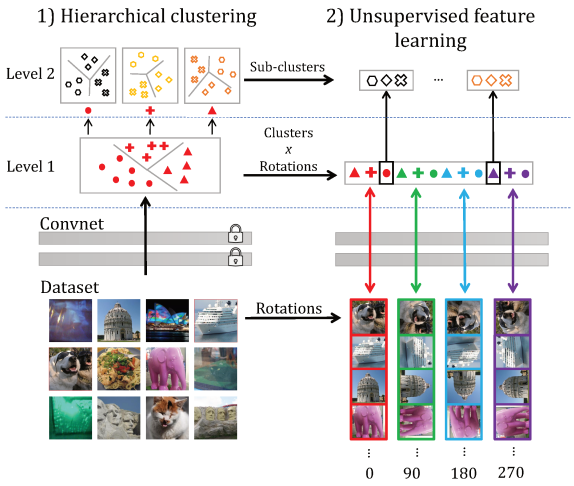


Figure 3.10 – **Illustration of DeeperCluster.** DeeperCluster alternates between a hierarchical clustering of the features and learning the parameters of a convnet by predicting both the rotation angle and the cluster assignments in a single hierarchical loss.

present in each super-class (with a different rotation), which does not take advantage of the hierarchical structure to use a bigger number of clusters.

Instead, we split the dataset into m sets by running k-means with m centroids on the full dataset every T epochs. We then use the Cartesian product between the assignment to these m clusters and the angle rotation classes to form the super-classes. There are $4m$ super-classes, each associated with the subset of data belonging to the corresponding cluster (N/m images if the clustering is perfectly balanced). These subsets are then further split with k-means into k sub-classes. This is equivalent to running a hierarchical k-means with rotation constraints on the full datasets to form our hierarchical loss. We typically use $m = 4$ and $k = 80$k, leading to a total of 320k different clusters split in 4 subsets. Our approach, "DeeperCluster", shares similarities with DeepCluster but is designed to scale to larger datasets. We alternate between clustering the non-rotated images features and training the network to predict both the rotation applied to the input data and its cluster assignment among the clusters corresponding to this rotation (Figure 3.10).

Distributed training. Building the super-classes based on data splits lends itself to a distributed implementation that scales well in the number of images. Specifically, when optimizing Eq. (3.4), we form as many distributed communication groups of p GPUs as the number of super-classes, i.e., $G = 4m$. Different communication groups share the parameters θ and the super-class classifier V, while the parameters of the sub-class classifiers

$W_1, \dots, W_S$ are only shared within a communication group. Each communication group s deals only with the subset of images and the rotation angle associated with the super-class s.

Distributed k-means. Every T epochs, we recompute the super and sub-class assignments by running two consecutive k-means on the entire dataset. This is achieved by first randomly splitting the dataset across different GPUs. Each GPU is in charge of computing cluster assignments for its partition, whereas centroids are updated across GPUs. We reduce communication between GPUs by sharing only the number of assigned elements for each cluster and the sum of their features. The new centroids are then computed from these statistics. We observe empirically that k-means converges in 10 iterations. We cluster 96M features of dimension 4096 into $m = 4$ clusters using 64 GPUs (1 minute per iteration). Then, we split this pool of GPUs into 4 groups of 16 GPUs. Each group clusters around 23M features into 80k clusters (4 minutes per iteration).

Implementation details. The loss in Eq. (3.4) is minimized with mini-batch stochastic gradient descent [Bottou, 2012]. Each mini-batch contains 3072 instances distributed across 64 GPUs, leading to 48 instances per GPU per mini-batch. We use dropout, weight decay, momentum and a constant learning rate of 0.1. We reassign clusters every 3 epochs. We use the Pascal VOC 2007 classification task without finetuning as a downstream task to select hyper-parameters. In order to speed up experiments, we initialize the network with RotNet trained on YFCC100M. Before clustering, we perform a whitening of the activations and ℓ_2-normalize each of them. We use standard data augmentations, i.e., cropping of random sizes and aspect ratios and horizontal flips [Krizhevsky et al., 2012]). We use the VGG-16 architecture [Simonyan and Zisserman, 2014] with batch normalization layers. We pre-process images with a Sobel filtering. We train our models on the 96M images from YFCC100M [Thomee et al., 2015] that we managed to download. We use this publicly available dataset for research purposes only.

3.3.2 Results

In this set of experiments, we compare RotNet, DeepCluster and DeeperCluster performance when trained on curated and non-curated datasets. We also evaluate the impact of the size of the dataset and number of clusters in DeeperCluster.

Comparison with RotNet and DeepCluster. In Table 3.3, we compare DeeperCluster with DeepCluster and RotNet when a linear classifier is trained on top of the last convolutional layer of a VGG-16 on several datasets. For reference, we also report previously

Method	Data	ImageNet	Places	VOC2007
Supervised	ImageNet	70.2	45.9	84.8
Wu et al. [2018]	ImageNet	39.2	36.3	-
RotNet [Gidaris et al., 2018]	ImageNet	32.7	32.6	60.9
DeepCluster	ImageNet	**48.4**	37.9	71.9
RotNet [Gidaris et al., 2018]	YFCC100M	33.0	35.5	62.2
DeepCluster	YFCC100M	34.1	35.4	63.9
DeeperCluster	YFCC100M	45.6	**42.1**	**73.0**

Table 3.3 – **Comparison between DeeperCluster, RotNet and DeepCluster when pre-trained on curated and non-curated dataset.** We report the accuracy on several datasets of a linear classifier trained on top of features of the last convolutional layer. All the methods use the same architecture. DeepCluster does not scale to the full YFCC100M dataset, we thus train it on a random subset of 1.3M images.

published numbers Wu et al. [2018] with a VGG-16 architecture. We do not perform any finetuning or layer selection. We average-pool the features of the last layer resulting in representations of 8192 dimensions. Our approach outperforms both RotNet and DeepCluster, even when they are trained on curated datasets (except for ImageNet classification task where DeepCluster trained on ImageNet yields the best performance). More interestingly, we see that the quality of the dataset or its scale has little impact on RotNet while it has on DeepCluster. This is confirming the intuition that pretext tasks methods based on data manipulation are more robust than clustering to a change of dataset distribution.

Influence of dataset size and number of clusters. To measure the influence of the number of images on features, we train models with 1M, 4M, 20M, and 96M images and report their accuracy on the validation set of the Pascal VOC 2007 classification task (FC68 setting). We also train models on 20M images with a number of clusters that varies from 10k to 160k. For the experiment with a total of 160k clusters, we choose $m = 2$ which results in 8 super-classes. In Figure 3.11, we observe that the quality of our features improves when scaling both in terms of images and clusters. Interestingly, between 4M and 20M of YFCC100M images are needed to meet the performance of our method on ImageNet. Augmenting the number of images has a bigger impact than the number of clusters. Yet, this improvement is significant since it corresponds to a reduction of more than 10% of the relative error w.r.t. the supervised model.

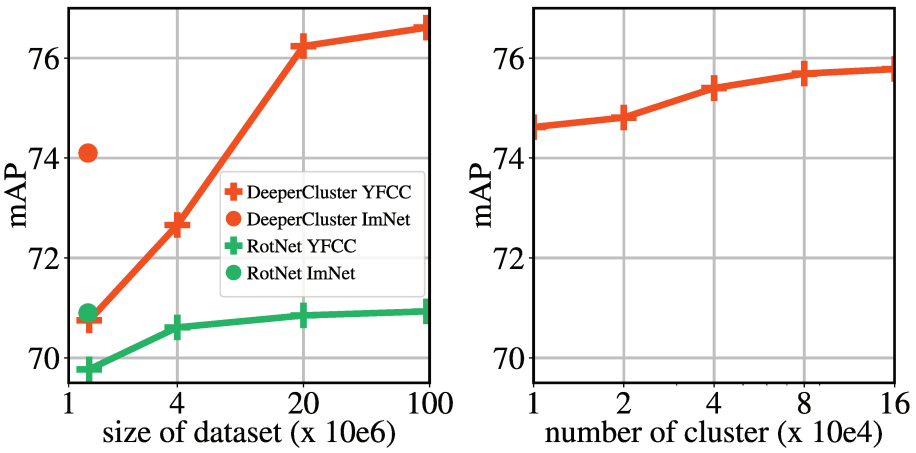


Figure 3.11 – **Influence of amount of data (*left*) and number of clusters (*right*) on the features quality.** We report validation mAP on Pascal VOC classification task (FC6 8 setting).

3.3.3 Quality of the clusters

In addition to features, our method provides a clustering of the input images. We evaluate the quality of these clusters by measuring their correlation with existing partitions of the data. In particular, YFCC100M comes with many different metadata. We consider hashtags, users, camera and GPS coordinates. If an image has several hashtags, we pick as label the least frequent one in the total hashtag distribution. We also measure the correlation of ours clusters with labels predicted by a classifier trained on ImageNet categories. We use a ResNet-50 network [He et al., 2016], pre-trained on ImageNet, to classify the YFCC100M images and we select those for which the confidence in prediction is higher than 75%. This evaluation omits a large amount of the data but gives some insight about the quality of our clustering in object classification.

In Figure 3.12, we show the evolution during training of the normalized mutual information (NMI) between our clustering and different metadata, and the predicted labels from ImageNet. The higher the NMI, the more correlated our clusters are to the considered partition. For reference, we compute the NMI for a clustering of RotNet features (as it corresponds to weights at initialization) and of a supervised model. First, it is interesting to observe that our clustering is improving over time for every type of metadata. One important factor is that most of these commodities are correlated since a given user takes pictures in specific places with probably a single camera and use a preferred fixed set of hashtags. Yet, these plots show that our model captures in the input signal enough information to predict these metadata at least as well as the features trained with supervision.

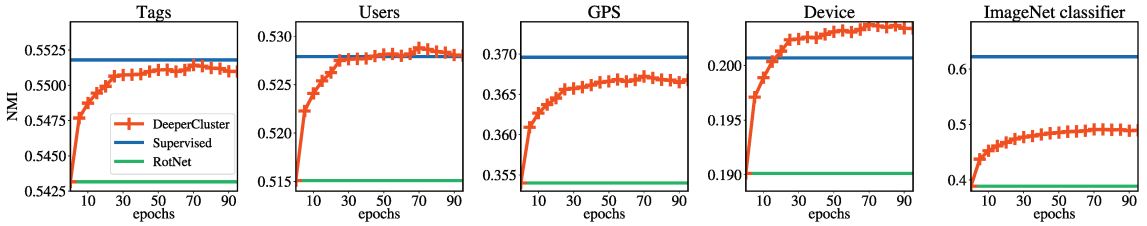


Figure 3.12 – **Evaluating the clustering quality on YFCC100m.** Normalized mutual information between our clustering and different sorts of metadata: hashtags, user IDs, geographic coordinates, and device types. We also plot the NMI with an ImageNet classifier labeling.

We visually assess the consistency of our clusters in Figure 3.13. We display 9 random images from 8 manually picked clusters. The first two clusters contain a majority of images associated with tag from the head (first cluster) and from the tail (second cluster) in the YFCC100M dataset. Indeed, 418.538 YFCC100M images are associated with the tag *cat* whereas only 384 images contain the tag *elephantparadelondon* (0.0004% of the dataset). We also show a cluster for which the dominant hashtag does not correlate visually with the content of the cluster. As already mentioned, this database is non-curated and contains images that basically do not depict anything semantic. The dominant metadata of the last cluster in the top row is the device ID *CanoScan*. As this cluster is about drawings, its images have been mainly taken with a scanner. Finally, the bottom row depict clusters that are pure for GPS coordinates but unpure for user IDs. It results in clusters of images taken by many different users in the same place: tourist landmarks.

3.4 Discussion and Limitations

In this chapter, we have presented a clustering approach for the self-supervised learning of deep networks, and its extension to deal with large amount of uncurated data. DeepCluster iterates between clustering with k-means the features produced by the deep network and updating its weights by predicting the cluster assignments as pseudo-labels in a discriminative loss. If trained on a curated dataset like ImageNet, it achieves performance that are significantly better than the previous state of the art at the time of the publication (i.e. 2018) as can be seen in Section 3.2.2. When trained on $96M$ of uncurated data (see Section 3.3), our approach surpasses unsupervised methods trained on curated datasets, which validates the potential of unsupervised learning in applications where annotations are scarce or curation is not trivial. Overall, we can interpret these promising results as a proof of concept of the fact that it is possible to use clustering for self-supervised learning.

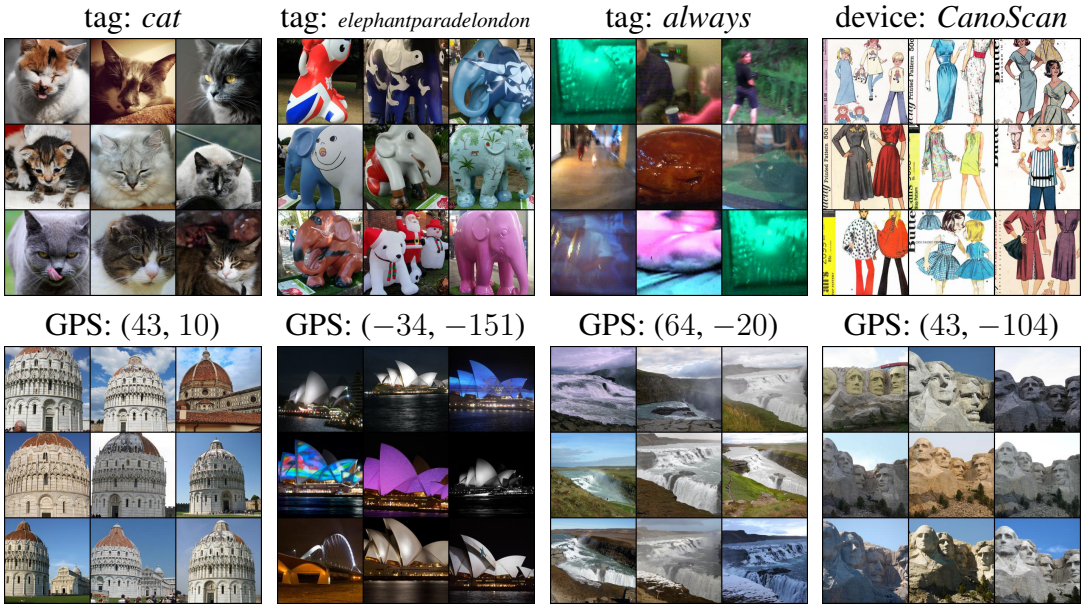


Figure 3.13 – **Cluster visualizations.** We randomly select 9 images per cluster and indicate the dominant cluster metadata. The bottom row depicts clusters pure for GPS coordinates but unpure for user IDs. As expected, they turn out to correlate with tourist landmarks. No metadata is used during training.

Yet, in its current form, the proposed approach has some major limitations which we detail in the following.

Limited scalability. A first limitation of Deep(er)Cluster is its inefficiency and limited scalability. Indeed, as shown in Eq. (3.1), both a linear classifier g_W and the network weights θ are trained to classify the images into their corresponding pseudo-label between two assignments. Intuitively, this classification layer represents *prototypes* for the different pseudo-classes. However, since there is no mapping between two consecutive cluster assignments, the prototypes learned for an assignment becomes irrelevant for the following one and hence need to be learned again from scratch. In practice, we indeed re-set the classification layer g_W to random weights whenever we have a new set of pseudo-labels. This considerably disrupts the stability of the network training, makes it quite inefficient and, as a result, less scalable.

Overall, the approach is not much scalable beyond the results that we have already shown with DeeperCluster. Indeed, we have seen that it is possible to train the models on large dataset such as YFCC100m but that is very costly and going beyond this scale would

be quite tedious with the current model. The main problem is that our approach depends directly on the size of the dataset. The approach alternates between some epochs of training and a clustering phase that is very costly since it requires to perform a forward pass on the entire dataset to collect the descriptors to cluster. This iterative process relies on the notion of epoch and on the size of the dataset. Indeed, to have good performance, we need to iterate a consequent number of times in order to refine the pseudo-labels regularly. If the dataset is not too large, like when training on ImageNet, we can refine the pseudo-labels between each epoch: this happens hundreds of time in a typical training. Now let us imagine we have a huge dataset where we can only afford a couple of passes on the dataset. This means that the pseudo-labels will be refined only a handful of times. Worse, if we have an even bigger dataset where we can only afford two passes, then the pseudo-labels are basically never refined. During the whole training (which consists in only one epoch in this case), we train with the first pseudo-labels which have been obtained from a randomly initialed network. As these pseudo-labels are likely to be semantically very poor, the resulting features will surely be pretty bad as well. Overall, a natural way to overcome the limitation of limited scalability is to adapt the method to work with an online version of k-means as done by Zhan et al. [2020] for example. However, given that we have observed in Table 3.2 that the choice of the clustering algorithm does not impact the performance much, similarly to Asano et al. [2020] and Chen et al. [2020d] we question the role of k-means and clustering altogether in DeepCluster.

Do we really need k-means? We remark that k-means clustering provide two distinct outputs: (i) the cluster assignments that we use as pseudo-labels (i.e. $y_n \in \{0,1\}^k$ in Eq. (3.2)) and (ii) a set of centroids (i.e. $C \in \mathbb{R}^{d \times k}$ in Eq. (3.2)). However, we make no use of the latter and only care about the former. This means that we are using only one component of k-means. Follows that we intuit that using k-means, an extremely simple clustering algorithm, might already be an overkill and we could find a simpler way to obtain the pseudo-labels that we use during training. We will see in the following chapter that we can directly use the output of the deep network to provide pseudo-labels, without resorting to an external clustering algorithm.

Empirical tricks to avoid trivial solution. As detailed previously in Section 3.2.1, our formulation is prone to trivial solution and to the "collapse" of the network to a constant output (thus all images have the same representation). We use heuristics to avoid this situation but it would be desirable to have a more principled formulation that prevents collapse by design.

		Random flips	Cropping	Performance
1	*Default*	Horizontal	Random cropping and scaling	12.5%
2		Vertical	Random cropping and scaling	11.1%
3		✗	Random cropping and scaling	12.4%
4		Horizontal	Random cropping	3.5%
5		Vertical	Random cropping	3.5%
6		✗	Random cropping	2.1%
7		Horizontal	✗	0.9%
8		Vertical	✗	0.9%
9		✗	✗	0.6%

Table 3.4 – **Impact of data augmentation (random flips and cropping).** We report the performance of different models, measured as the proportion of intra-class edges into a nearest neighbor graph on the descriptors on ImageNet. DeepCluster models are trained for 50 epochs on ImageNet without labels with AlexNet. We observe that random cropping and scaling is crucial for the method to learn semantic representations.

Implicit importance of data augmentation. Last but not least, we describe a final major limitation: we have presented data augmentation merely as an implementation detail in this chapter while it turns out to be a crucial component of the method. Indeed, let us conduct a final experiment where we ablate the different data transformations used during DeepCluster training. In Table 3.4, we experiment with three different flips: vertical, horizontal (both occurring with a probability of 0.5) and no flip. We also benchmark three different types of cropping: (i) (default cropping) a crop of random size (0.08 to 1.0 of the original size) and random aspect ratio (of 3/4 to 4/3 of the original aspect ratio), (ii) crop of fixed size with a random localization and (iii) no cropping, i.e. we take a central crop, as used for the clustering phase. We observe in Table 3.4 that the only configurations giving good features are that using random cropping and scaling (rows 1, 2 and 3). This transformation is thus crucial for DeepCluster. We also notice that flipping does not play a key role and that we could remove it. Using vertical flip degrades the performance which is consistent with the method of Gidaris et al. [2018] that trains the network to be discriminative to this transformation rather than invariant.

Overall, this dependence in the data augmentation reinforces the interpretation of our framework as a variant of the view-invariant paradigm as described in Section 2.3.4. In the following chapter, we will make data cropping more central to the model which will allow us to have substantial boosts of performance.

Chapter 4

The Self-Supervised SwAV Approach

In this chapter, we present a new method for self-supervised representation learning, SwAV, that overcomes the difficulties and limitations of both deep clustering (as detailed in Section 3.4) and contrastive representation learning (as mentioned in Section 2.3.2). Compared to deep clustering, SwAV works online (i.e. at the mini-batch level), scales to unlimited amount of data, has guarantees that prevent mode collapse and does not rely on an external clustering algorithm. Compared to contrastive methods, SwAV does not rely on "negative samples" which makes it more practical since it works without requiring a large memory bank [He et al., 2020, Wu et al., 2018] nor large mini-batches [Chen et al., 2020b]. SwAV yields excellent results when trained on ImageNet without labels, achieving 75.3% top-1 accuracy with a standard ResNet-50 [He et al., 2016] on the ImageNet linear evaluation protocol [Zhang et al., 2016]. This performance was more than 4 points above the state of the art at the time of the publication. Perhaps more importantly, SwAV features transfer very well to various downstream tasks, outperforming supervised features on classification and detection tasks on all the considered datasets. This work was presented at NeurIPS in 2020 [Caron et al., 2020].

Similarly to our previous study on deep clustering (see Section 3.3), we assess SwAV at scale when trained on uncurated data. We consider billions of Instagram images and train a very deep network with billions of parameters. This effort was described in a technical report (see Goyal et al. [2021]).

4.1 Introduction

As detailed in the first parts of this manuscript (see Chapter 1 and Chapter 2), self-supervised learning aims at obtaining features without using manual annotations and has rapidly been closing the performance gap with supervised pretraining in computer vi-

sion [Caron et al., 2018, Chen et al., 2020b, He et al., 2020, Misra and Maaten, 2020]. Many successful methods build upon the instance discrimination task that considers each image of the dataset (or "instance") and its transformations as a separate class [Dosovitskiy et al., 2016] (see a review in Section 2.3). This task yields representations that are able to discriminate between different images, while achieving some invariance to image transformations. Contrastive formulations of this view-invariant paradigm rely on a combination of two elements: (i) a contrastive loss [Hadsell et al., 2006] and (ii) a set of image transformations. The contrastive loss removes the notion of instance classes by directly comparing image features while the image transformations define the invariances encoded in the features. Both elements are essential to the quality of the resulting networks [Chen et al., 2020b, Misra and Maaten, 2020] and our work improves upon both the objective function and the transformations.

The contrastive loss explicitly compares pairs of image representations to push away representations from different images while pulling together those from transformations, or views, of the same image. Since computing all the pairwise comparisons on a large dataset is not practical, most implementations approximate the loss by reducing the number of comparisons to random subsets of images during training [Chen et al., 2020b, He et al., 2020, Wu et al., 2018]. An alternative to approximate the loss is to approximate the task—that is to relax the instance discrimination problem [Dosovitskiy et al., 2014]. For example, clustering-based methods discriminate between groups of images with similar features instead of individual images [Caron et al., 2018]. The objective in clustering is tractable, but has its own limitations as detailed in Section 3.4. For instance, it does not scale well with the dataset as it requires a pass over the entire dataset to get image cluster assignments that are used as pseudo-labels during training. In this chapter, we use a different paradigm and propose to compute pseudo-labels in a much simpler fashion: the pseudo-label used as target for a view is directly the normalized output of another view of the same image. This formulation encourages distorted views of a same image to match in feature space while not relying on explicit pairwise feature comparisons as in the contrastive implementations. Specifically, we propose a simple "swapped" prediction problem where we predict the pseudo-label (or "assignment") of a view from the representation of another view. We learn features by **Sw**apping **A**ssignments between multiple **V**iews of the same image (**SwAV**). Unlike deep clustering, the features and the pseudo-labels are learned online, allowing our method to scale to potentially unlimited amounts of data. Compared to contrastive methods, SwAV works with small and large batch sizes, does not rely on direct feature comparisons (i.e. "positives" and "negatives") and hence does not need a large memory bank [Wu et al., 2018] nor a momentum encoder [He et al., 2020].

Besides this scalable implementation of the view-invariant paradigm, we also propose an improvement to the image transformations. Most methods are trained to align the

representations of two independent and identically distributed views of an image. We hypothesize that this is sub-optimal for two reasons. First, there has been evidence that comparing more views than two during training improves the resulting model [Misra and Maaten, 2020]. Second, matching views from *different distributions* (as opposed to "identically distributed") could be a way to enforce desired properties in the learned representations. We propose that some views have access to only a narrow, local crop of the original image while other views cover a global and large area of that image. Thus, SwAV learns to extrapolate and generalize from the local view to the global content of the image, hence encouraging *holistic* representations. This strategy, called *multi-crop*, is simple and boosts considerably the performance of different self-supervised methods.

We validate our contributions by evaluating our method on several standard self-supervised benchmarks. Overall, in this chapter, we present the following contributions:

— We propose a new and scalable implementation of the view-invariant self-supervised paradigm that improves performance by $+2\%$ on ImageNet over contrastive implementations. Unlike contrastive methods, it works in both large and small batch settings without a large memory bank or a momentum encoder.

— We introduce the multi-crop strategy to increase the number of views of an image with limited computational or memory overhead. Multi-crop encourages local-to-global matching of the representations. We observe a consistent improvement of between 2% and 4% on ImageNet with this strategy on several self-supervised methods.

— Combining both technical contributions into a single model, SwAV, we improved the performance of self-supervised methods by $+4.2\%$ on ImageNet. Our features outperform supervised ImageNet pretraining on multiple downstream tasks. At the time of the publication, this was the first method to do so without finetuning the features, i.e., only with a linear classifier on top of frozen features.

— We explore if self-supervision lives to its expectation by training large models on random, uncurated images with no supervision. Our final SwAV model trained with a RegNetY [Radosavovic et al., 2020] with 1.3B parameters on 1B random images achieves 84.2% top-1 accuracy confirming that self-supervised learning works in a real world setting.

4.2 SwAV Methodology

Our goal is to learn visual features in an online fashion without supervision. To that effect, we propose a new, online implementation of the view-invariant paradigm (introduced in Section 2.3) which consists in learning representations that are invariant to distortions of the input sample. Our formulation aims primarily at overcoming both the limitations

of the contrastive-based implementations, that rely on direct feature comparison, and of the clustering-based methods [Asano et al., 2020, Caron et al., 2018] like DeepCluster (see Chapter 3), that are offline. This means that they alternate between a cluster assignment step where image features of the entire dataset are clustered, and a training step where the cluster assignments, i.e., "pseudo-labels" are predicted for different image views. Unfortunately, these methods are not suitable for online learning as they require multiple passes over the dataset to compute the image features necessary for clustering.

In this section, we describe an alternative implementation where we directly use the output of the network as a pseudo-label to predict from another view of the same image. We enforce explicit constraints on these pseudo-labels to prevent the network from learning a constant mapping. Indeed, if the network outputs a constant pseudo-label regardless of its input then it is of course trivial to predict.

4.2.1 Matching views through pseudo-labeling

SwAV works by computing a pseudo-label from an augmented version of the image and predicting it from other augmented versions of the same image. More precisely, given an input image, we take two different random augmentations of that same image, resulting in crops x_t and x_s. The augmented views are mapped to a K-dimensional vector representation by applying a non-linear mapping f_θ parameterized by θ. We denote by $\mathbf{z} = f_\theta(x)$ the output of the network which we interpret as a vector of scores for K latent pseudo-classes or "clusters". To transform this vector of scores into a soft pseudo-label over K dimensions, we adjust it based on constraints on the other mini-batch scores in order to prevent the collapse of the representations. We will explicit these constraints and describe the adjustment process in the following paragraph, but for now let us denote by $\mathbf{q}_t$ the resulting adjusted and normalized pseudo-label obtained from $\mathbf{z}_t$. We train the network to predict this pseudo-label $\mathbf{q}_t$ from the representation of another view $\mathbf{z}_s$. A probability distribution $\mathbf{p}_s$ is obtained by normalizing $\mathbf{z}_s$ with a standard softmax function:

$$\mathbf{p}_s^{(i)} = \frac{\exp(\mathbf{z}_s^{(i)}/\tau)}{\sum_{k=1}^{K} \exp(\mathbf{z}_s^{(k)}/\tau)}, \tag{4.1}$$

with $\tau > 0$ a temperature parameter that controls the sharpness of the output distribution. We minimize the cross-entropy loss w.r.t. the parameters θ of the feature encoder and considered the pseudo-label $\mathbf{q}_t$ *fixed* (i.e. we apply a `stop-gradient` operator):

$$\min_\theta H(\mathbf{q}_t, \mathbf{p}_s), \tag{4.2}$$

where $H(\mathbf{q}, \mathbf{p}) = -\sum_{i=1}^{K} \mathbf{q}^{(i)} \log \mathbf{p}^{(i)}$.

4.2.2 Online pseudo-labels with optimal transport

In order to make our method online, we compute the pseudo-labels using only the image features within a batch. We enforce that all the examples in a batch are equally partitioned into the K different dimensions or "clusters". This equipartition constraint ensures that the pseudo-labels for different images in a batch are distinct, thus preventing the trivial solution where every image has the same pseudo-label. We denote by B the mini-batch of feature vectors $\mathbf{Z} = [\mathbf{z}_1, \dots, \mathbf{z}_B]$ and by $\mathbf{Q} = [\mathbf{q}_1, \dots, \mathbf{q}_B]$ the desired matrix of pseudo-labels. We establish the following constraints for $\mathbf{Q}$:

$$\mathcal{Q} = \left\{ \mathbf{Q} \in \mathbb{R}_+^{K \times B} \mid \mathbf{Q}\mathbf{1}_B = \mathbf{1}_K, \mathbf{Q}^\top \mathbf{1}_K = \mathbf{1}_B \right\}, \tag{4.3}$$

where $\mathbf{1}_K$ denotes the vector of ones in dimension K. These constraints mean that each $\mathbf{q}_n$ should sum to 1 (i.e. it can be interpreted as an assignment over the K dimensions) and that, on average, each dimension "is used $\frac{B}{K}$ times in the batch". As we derive $\mathbf{Q}$ from $\mathbf{Z}$, we want $\mathbf{Q}$ to stay as close to Z as possible while satisfying the constraints above. More precisely, we have the following problem:

$$\max_{\mathbf{Q} \in \mathcal{Q}} \operatorname{Tr}\left(\mathbf{Q}^\top \mathbf{Z}\right) + \varepsilon H(\mathbf{Q}), \tag{4.4}$$

where H is the entropy function, $H(\mathbf{Q}) = -\sum_{ij} \mathbf{Q}_{ij} \log \mathbf{Q}_{ij}$ and ε is a parameter that controls the smoothness of the solution $\mathbf{Q}$. We observe that a strong entropy regularization (i.e. using a high ε) generally leads to a trivial solution where all samples collapse into an unique uniform pseudo-label. Hence in practice we keep ε low.

Once a continuous solution $\mathbf{Q}^*$ to Prob. (4.4) is found, a discrete pseudo-label can be obtained by using a rounding procedure [Asano et al., 2020]. However, in our setting we find that using the discrete pseudo-label performs worse than using the soft continuous ones (we lose 0.5% in the linear evaluation on ImageNet). An explanation is that the rounding needed to obtain discrete assignments is a more aggressive optimization step than gradient updates. While it makes the model converge rapidly, it leads to a worse solution. We thus preserve the soft pseudo-label $\mathbf{Q}^*$ instead of rounding it. These soft assignments $\mathbf{Q}^*$ are the solution of Prob. (4.4) over the set $\mathcal{Q}$ and takes the form of a normalized exponential matrix [Cuturi, 2013]:

$$\mathbf{Q}^* = \operatorname{Diag}(\mathbf{u}) \exp\left(\mathbf{Z}/\varepsilon\right) \operatorname{Diag}(\mathbf{v}), \tag{4.5}$$

where $\mathbf{u}$ and $\mathbf{v}$ are re-normalization vectors in $\mathbb{R}^K$ and $\mathbb{R}^B$ respectively. The re-normalization vectors are computed using a small number of matrix multiplications using the iterative Sinkhorn-Knopp algorithm [Cuturi, 2013]. In practice, we observe that using only 3 iterations is fast and sufficient to obtain good performance. Indeed, this algorithm can be efficiently implemented on GPU, and the alignment of 4K features to 3K codes takes 35ms

in our experiments. Interestingly, the parameter ε plays a similar role of controlling the smoothness of the exponential in Eq. (4.5) as the temperature τ in Eq. (4.2).

Overall, the optimal transport algorithm Cuturi [2013] used to adjust the scores $\mathbf{Z}$ is implemented simply with the following PyTorch style code.

```
# Z is B-by-K
# eps is Sinkhorn regularization parameter
Q = exp(Z / eps)
for _ in range(num_iters): # 1 iter of Sinkhorn
  # total weight per dimension (or cluster)
  c = sum(Q, dim=0, keepdim=True)
  Q /= c

  # total weight per sample
  n = sum(Q, dim=1, keepdim=True)
  # Q sums to 1 for each sample (assignment)
  Q /= n
return Q
```

As a matter of fact, when performing a single Sinkhorn iteration (`num_iters=1`) this is equivalent to taking a softmax function in the batch dimension and the implementation can be highly simplified:

```
Q = softmax(Z / eps, dim=0)
Q /= sum(Q, dim=1, keepdim=True)
```

Intuitively, the `softmax` operation on the batch axis allows to select for each dimension (or "cluster") its best match within the batch.

4.2.3 Working with small batches

When the number B of batch features is too low, our constraint to equally partition the batch into the K prototypes does not really make sense anymore. Therefore, when working with small batches, we use features from the previous batches to augment the size of $\mathbf{Z}$ in Prob. (4.4). This is similar to the memory bank mechanism used in contrastive methods He et al. [2020], Wu et al. [2018]. Then, we only use the pseudo-labels of the batch features in our training loss. In practice, we store around 3K features, i.e., in the same range as the number of code vectors. This means that we only keep features from the last 15 batches with a batch size of 256, while contrastive methods typically need to store the last 65K instances obtained from the last 250 batches to maintain good performance [He et al., 2020].

4.2.4 Multi-crop: local-to-global matching

As noted in prior works [Chen et al., 2020b, Misra and Maaten, 2020] and as observed in our previous experiments of Section 3.4, comparing random crops of an image plays a central role by capturing information in terms of relations between parts of a scene or an object. Instead of sampling a pair of matching views identically distributed as in Chen et al. [2020b], He et al. [2020], Wu et al. [2018], we design a cropping strategy that encourage local-to-global correspondences in the features. This way, the task of the network is to extrapolate a situation from a narrow restrained view. More precisely, from a given image, we generate a set V of different views. This set contains two *global* views, x_1^g and x_2^g and several *local* views of smaller resolution. Only the *global* views are used to compute a pseudo-label which then needs to be predicted from each of the other views of the same image. We minimize the per-example following loss:

$$\min_{\theta} \sum_{x_t \in \{x_1^g, x_2^g\}} \sum_{\substack{x_s \in V \\ x_s \neq x_t}} H(\mathbf{q}(x_t), \mathbf{p}(x_s)). \tag{4.6}$$

This loss is general and can be used on any number of views, even only 2. However, we find that using 2 global views at resolution 224^2 covering a large (for example greater than 50%) area of the original image, and several local views of resolution 96^2 covering only small areas (for example less than 50%) of the original image brings considerable boosts of performance. We refer to this setting as the basic parametrization of SwAV, unless mentioned otherwise. As a matter of fact, using low resolution images ensures only a small increase in the compute cost.

4.2.5 Implementation details

We implement in SwAV the generic implementation improvements used in SimCLR [Chen et al., 2020b], i.e., LARS [You et al., 2017], cosine learning rate [Loshchilov and Hutter, 2016] and the MLP projection head [Bachman et al., 2019]. We train SwAV with stochastic gradient descent using large batches of 4096 different instances. We distribute the batches over 64 V100 16Gb GPUs, resulting in each GPU treating 64 instances. The temperature parameter τ is set to 0.1 and the Sinkhorn regularization parameter ε is set to 0.05 for all runs. We use a weight decay of 10^{-6} and a learning rate of 4.8 which is linearly ramped up during the first 10 epochs. We find empirically that freezing the last layer of the network during the first epoch of training improves stability. We synchronize batch-normalization layers across GPUs using the optimized implementation with kernels through CUDA/C-v2 extension from `apex` [1]. We also use `apex` library for training with

1. github.com/NVIDIA/apex

mixed precision [Micikevicius et al., 2017]. Overall, thanks to these training optimizations (mixed precision, kernel batch-normalization and use of large batches), 100 epochs of training for our best SwAV model take approximately 6 hours. Similarly to previous works [Chen et al., 2020b,e, Li et al., 2020], we use a projection head consisting of a 2-layer MLP on top of the backbone features. Note that SwAV is more suitable for a multi-node distributed implementation compared to contrastive approaches SimCLR or MoCo. The latter methods require sharing the feature matrix across all GPUs at every batch which might become a bottleneck when distributing across many GPUs. On the contrary, SwAV requires sharing only matrix normalization statistics (sum of rows and columns) during the Sinkhorn algorithm. For multi-crop, we perform crops of random sizes, scales and aspect ratios. Specifically we use the `RandomResizedCrop` method from `torchvision.transforms` module of PyTorch with `s=(0.14, 1)` scaling parameters for the global views and `s=(0.05, 0.14)` parameters for the local ones. Then, we resize global views to 224×224 pixels, unless specified otherwise (we use 160×160 resolutions in some of our experiments). Local views are resized to 96×96 resolution. Finally, we apply random horizontal flips, color distortion and Gaussian blur to each resulting crop, exactly following the SimCLR implementation [Chen et al., 2020b]. Our code is publicly available at https://github.com/facebookresearch/swav.

4.3 Main Results

We analyze the features learned by SwAV trained on ImageNet without labels by transfer learning on multiple datasets.

4.3.1 Evaluating the unsupervised features on ImageNet

We evaluate the features of a ResNet-50 [He et al., 2016] trained with SwAV on ImageNet by two experiments: linear classification on frozen features and semi-supervised learning by finetuning with few labels. When using frozen features (Figure 4.1 (left)), SwAV outperforms the state of the art by $+4.2\%$ top-1 accuracy and is only 1.2% below the performance of a fully supervised model. Note that we train SwAV during 800 epochs with large batches (4096). We refer to Figure 4.2 (right) for results with shorter trainings and to Table 4.3 for experiments with small batches.

On semi-supervised learning (see Table 4.1), SwAV outperforms other self-supervised methods and is on par with state-of-the-art semi-supervised models [Sohn et al., 2020], despite the fact that SwAV is not specifically designed for semi-supervised learning. Note that in Assran et al. [2021], we extend SwAV methodology to a semi-supervised setting,

Method	Arch.	Param.	Top1
Supervised	R50	24	76.5
Zhang et al. [2016]	R50	24	39.6
Noroozi and Favaro [2016]	R50	24	45.7
Wu et al. [2018]	R50	24	54.0
Donahue and Simonyan [2019]	R50	24	56.6
Zhuang et al. [2019]	R50	24	58.8
He et al. [2020]	R50	24	60.6
Asano et al. [2020]	R50	24	61.5
Misra and Maaten [2020]	R50	24	63.6
Hénaff et al. [2019]	R50	24	63.8
Li et al. [2020]	R50	24	65.9
Chen et al. [2020b]	R50	24	70.0
Chen et al. [2020e]	R50	24	71.1
SwAV	R50	24	**75.3**

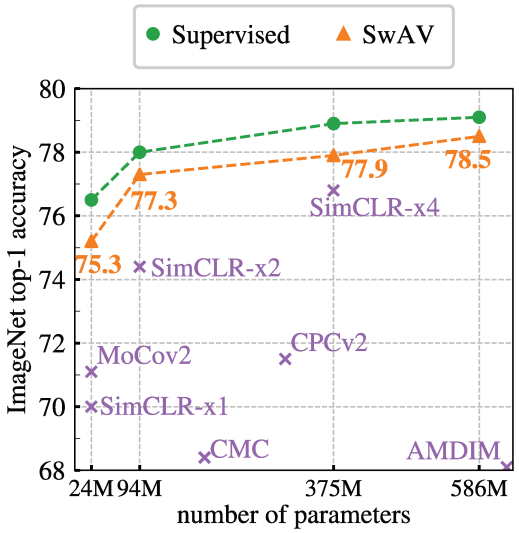


Figure 4.1 – **Linear classification on ImageNet.** Top-1 accuracy for linear models trained on frozen features from different self-supervised methods available at the time of SwAV publication. **(left)** Performance with a standard ResNet-50. **(right)** Performance as we multiply the width of a ResNet-50 by a factor $\times 2$, $\times 4$, and $\times 5$.

which allows to improve drastically the performance. As a matter of fact, as SimCLR and MoCov2 [Chen et al., 2020e], we use the training improvements introduced by SimCLR [Chen et al., 2020b] (see Section 4.2.5 for details). These technical improvements explain part of the performance gap with other previous works.

Variants of ResNet-50. Figure 4.1 (right) shows the performance of multiple variants of ResNet-50 with different widths [Kolesnikov et al., 2019]. The performance of our model increases with the width of the model, and follows a similar trend to the one obtained with supervised learning. When compared with concurrent work like SimCLR, we see that SwAV reduces the difference with supervised models even further. Indeed, for large architectures, our method shrinks the gap with supervised training to 0.6%.

4.3.2 Transferring unsupervised features to downstream tasks

We test the generalization of ResNet-50 features trained with SwAV on ImageNet (without labels) by transferring to several downstream vision tasks. In Table 4.2, we

	Method	1% labels Top-1	1% labels Top-5	10% labels Top-1	10% labels Top-5
	Supervised	25.4	48.4	56.4	80.4
Methods using label-propagation	UDA [Xie et al., 2020a]	-	-	68.8	88.5
	FixMatch [Sohn et al., 2020]	-	-	**71.5**	89.1
Methods using self-supervision only	PIRL [Misra and Maaten, 2020]	30.7	57.2	60.4	83.8
	PCL [Li et al., 2020]	-	75.6	-	86.2
	SimCLR [Chen et al., 2020b]	48.3	75.5	65.6	87.8
	SwAV	**53.9**	**78.5**	70.2	**89.9**

Table 4.1 – **Semi-supervised learning on ImageNet with a ResNet-50.** We finetune the model with 1% and 10% labels and report top-1 and top-5 accuracies. We use the same splits as Chen et al. [2020b].

compare the performance of SwAV features with ImageNet supervised pre-training and with self-supervised features available at the time of SwAV publication. First, we report the linear classification performance on the Places205 [Zhou et al., 2014], VOC07 [Everingham et al., 2010], and iNaturalist2018 [Van Horn et al., 2018] datasets. Our method outperforms supervised features on all three datasets. We observe that SwAV is the first self-supervised method to surpass ImageNet supervised features on these datasets. Second, we report network finetuning on object detection on VOC07+12 using Faster R-CNN [Ren et al., 2015] and on COCO [Lin et al., 2014] with DETR [Carion et al., 2020]. DETR is a recent object detection framework that reaches competitive performance with Faster R-CNN while being conceptually simpler and trainable end-to-end. We use DETR because, unlike Faster R-CNN [He et al., 2019], using a pretrained backbone in this framework is crucial to obtain good results compared to training from scratch [Carion et al., 2020]. In Table 4.2, we show that SwAV outperforms the supervised pretrained model on both VOC07+12 and COCO datasets. Note that this is line with previous works that also show that self-supervision can outperform supervised pretraining on object detection [Gidaris et al., 2020a, He et al., 2020, Misra and Maaten, 2020]. Overall, our SwAV ResNet-50 model surpasses supervised ImageNet pretraining on all the considered transfer tasks and datasets. We have released this model so other researchers might also benefit by replacing the ImageNet supervised network with our model.

	Linear Classification			Object Detection		
	Places205	VOC07	iNat18	VOC07+12 (Faster R-CNN R50-C4)	COCO (Mask R-CNN R50-FPN)	COCO (DETR)
Supervised	53.2	87.5	46.7	81.3	39.7	40.8
Self-supervised methods						
Gidaris et al. [2020a]	45.0	64.6	-	-	-	
He et al. [2020]	46.9†	79.8†	31.5†	81.5	-	
Misra and Maaten [2020]	49.8	81.1	34.1	80.7	-	
Li et al. [2020]	49.8	84.0	-	-	-	
Gidaris et al. [2020a]	51.1	79.3	-	81.3	-	
Chen et al. [2020b]	53.3†	86.4†	36.2†	-	-	
Chen et al. [2020e]	52.9†	87.1†	38.9†	82.5	39.8	42.0†
SwAV	**56.7**	**88.9**	**48.6**	**82.6**	**41.6**	**42.1**

Table 4.2 – **Transfer learning on downstream tasks.** Comparison between features from ResNet-50 trained on ImageNet with SwAV or supervised learning. We also report numbers from other self-supervised methods available at the time of SwAV publication († for numbers from other methods run by us). We consider two settings. (1) Linear classification on top of frozen features. We report top-1 accuracy on all datasets except VOC07 where we report mAP. (2) Object detection with finetuned features on VOC07+12 `trainval` using Faster R-CNN [Ren et al., 2015] and on COCO [Lin et al., 2014] using Mask R-CNN [He et al., 2017] or DETR [Carion et al., 2020]. We report the most standard detection metrics for these datasets: AP_{50} on VOC07+12 and AP on COCO.

4.3.3 Training with small batches

We train SwAV with small batches of 256 images on 4 GPUs and compare with MoCov2 and SimCLR trained in the same setup. In Table 4.3, we see that SwAV maintains state-of-the-art performance even when trained in the small batch setting. Note that SwAV only stores a queue of $3,840$ features. In comparison, to obtain good performance, MoCov2 needs to store $65,536$ features while keeping an additional momentum encoder network. When SwAV is trained using $2\times160 + 4\times96$ crops, SwAV has a running time $1.2\times$ higher than SimCLR with 2×224 crops and is around $1.4\times$ slower than MoCov2 due to the additional back-propagation [Chen et al., 2020e]. Hence, one epoch of MoCov2 or SimCLR is faster in wall clock time than one of SwAV, but these methods need more epochs for good downstream performance. Indeed, as shown in Table 4.3, SwAV learns much faster and reaches higher performance in $4\times$ fewer epochs: 72% after 200 epochs

Method	Mom. Encoder	Stored Features	multi-crop	epoch	batch	Top-1
SimCLR		0	2×224	200	256	61.9
MoCov2	✓	65,536	2×224	200	256	67.5
MoCov2	✓	65,536	2×224	800	256	71.1
SwAV		3,840	$2\times160 + 4\times96$	200	256	72.0
SwAV		3,840	$2\times224 + 6\times96$	200	256	72.7
SwAV		3,840	$2\times224 + 6\times96$	400	256	**74.3**

Table 4.3 – **Training in small batch setting.** Top-1 accuracy on ImageNet with a linear classifier trained on top of frozen features from a ResNet-50. All methods are trained with a batch size of 256. We also report the number of stored features, the type of cropping used and the number of epochs.

(102 hours) while MoCov2 needs 800 epochs to achieve 71.1%. Increasing the resolution and the number of epochs, SwAV reaches 74.3% with a small batch size, a small number of stored features and no momentum encoder. Finally, note that SwAV can be combined with a momentum mechanism and we leave these explorations to the next chapter.

4.4 Ablation Study and Analyses

Improving deep clustering approaches. In this section, we re-implement and improve two clustering-based models in order to assess if they can compete with well-known contrastive methods such as SimCLR [Chen et al., 2020b]. In particular, we consider two clustering-based models: DeepCluster-v2 and SeLa-v2, which are obtained by applying various training improvements introduced in other self-supervised learning papers to DeepCluster of Caron et al. [2018] and SeLa of Asano et al. [2020]. Among these improvements are the use of stronger data augmentation [Chen et al., 2020b], MLP projection head [Bachman et al., 2019], cosine learning rate schedule [Misra and Maaten, 2020], use of temperature-based softmax [Wu et al., 2018], memory bank [Wu et al., 2018], multi-clustering [Asano et al., 2020], etc. The implementation of DeepCluster-v2 is publicly available at `https://github.com/facebookresearch/swav/main_deepclusterv2.py`. Besides, we also improve DeepCluster model by introducing explicit comparisons to k-means centroids, which increase stability and performance. Indeed, a main issue in DeepCluster is that there is no correspondence between two consecutive cluster assignments. Hence, the final classification layer learned for an assignment becomes irrelevant for the following one and thus needs to be re-initialized from scratch at each epoch. This considerably disrupts

Method	Top-1 2x224	Top-1 2x160+4x96	Δ
Supervised	76.5	76.0	−0.5
Contrastive-instance approaches			
SimCLR	68.2	70.6	+2.4
Clustering-based approaches			
SeLa-v2	67.2	71.8	+4.6
DeepCluster-v2	70.2	74.3	+4.1
SwAV	70.1	74.1	+4.0

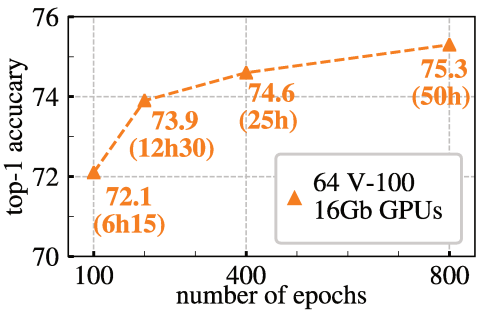


Figure 4.2 – Top-1 accuracy on ImageNet with a linear classifier trained on top of frozen features from a ResNet-50. **(left) Comparison between clustering-based and contrastive instance methods and impact of multi-crop.** Self-supervised methods are trained for 400 epochs and supervised models for 200 epochs. **(right) Performance as a function of epochs.** We compare SwAV models trained with different number of epochs and report their running time based on our implementation.

the convnet training. In DeepCluster-v2, instead of learning a classification layer predicting the cluster assignments, we perform explicit comparison between features and centroids.

Comparing clustering with contrastive instance learning. In Figure 4.2 (left), we make a best effort fair comparison between clustering-based and contrastive instance (SimCLR) methods by implementing these methods with the same data augmentation, number of epochs, batch-sizes, etc. In this setting, we observe that DeepCluster-v2 outperform SimCLR by 2% without multi-crop and by 3.5% with multi-crop. This suggests the learning potential of clustering-based methods over instance classification as also observed in the analyses of Dosovitskiy et al. [2014].

Advantage of SwAV compared to DeepCluster-v2. In Figure 4.2 (left), we observe that SwAV performs on par with DeepCluster-v2. In addition, we train DeepCluster-v2 in SwAV best setting (800 epochs - 8 crops) and obtain 75.2% top-1 accuracy on ImageNet (versus 75.3% for SwAV). However, unlike SwAV, DeepCluster-v2 is not online which makes it impractical for extremely large datasets. For billion scale trainings for example, a single pass on the dataset is usually performed [He et al., 2020]. DeepCluster-v2 cannot be trained for only one epoch since it works by performing several passes on the dataset to

Method	multi-crop	time / 100 epochs	peak memory / GPU
SimCLR	2×224	4h00	8.6G
SwAV	2×224	4h09	8.6G
SwAV	$2 \times 160 + 4 \times 96$	4h50	8.5G
SwAV	$2 \times 224 + 6 \times 96$	6h15	12.8G

Table 4.4 – **Computational cost.** We report time and GPU memory requirements based on our implementation for different models trained during 100 epochs.

regularly update centroids and cluster assignments for each image.

As a matter of fact, DeepCluster-v2 can be interpreted as a special case of our proposed swapping mechanism: swapping is done across epochs rather than within a batch. Given a crop of an image DeepCluster-v2 predicts the assignment of another crop, which was obtained at the previous epoch. SwAV swaps assignments directly at the batch level and can thus work online.

Applying the multi-crop strategy to different methods. In Table 4.2 (left), we report the impact of applying our multi-crop strategy on the performance of a selection of other methods. We see that the multi-crop strategy consistently improves the performance for all the considered methods by a significant margin of $2-4\%$ top-1 accuracy. Interestingly, multi-crop seems to benefit more clustering-based methods than contrastive methods. We note that multi-crop does not improve the supervised model.

Impact of longer training. In Figure 4.2 (right), we show the impact of the number of training epochs on performance for SwAV. We train separate models for 100, 200, 400 and 800 epochs and report the top-1 accuracy on ImageNet using the linear classification evaluation. We train each ResNet-50 on 64 V100 16GB GPUs and a batch size of 4096. While SwAV benefits from longer training, it already achieves strong performance after 100 epochs, i.e., 72.1% in 6h15.

Running times. In Table 4.4, we report compute and GPU memory requirements based on our implementation for different settings. As described in Section 4.2.5, we train each method on 64 V100 16GB GPUs, with a batch size of 4096, using mixed precision and `apex` optimized version of synchronized batch-normalization layers. We report results with ResNet-50 for all methods. In Figure 4.3, we report SwAV performance for different training lengths measured in hours based on our implementation. We observe that after only 6 hours of training, SwAV outperforms SimCLR trained for 1000 epochs (40 hours

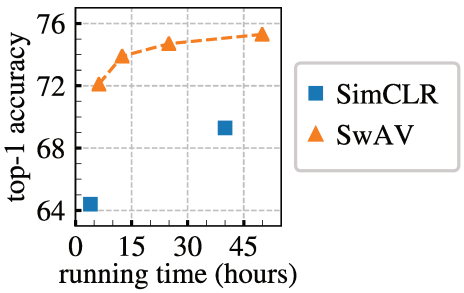


Figure 4.3 – **Running times.** Top-1 ImageNet accuracy for linear models trained on frozen features. We report SwAV performance for different training lengths measured in hours based on our implementation. We train each ResNet-50 models on 64 V100 16GB GPUs with a batch size of 4096.

based on our implementation) by a large margin. If we train SwAV for longer, we see that the performance gap between the two methods increases even more.

Number K of different dimensions	300	1,000	3,000	10,000	30,000	100,000
Top-1	72.8	73.6	73.9	74.1	73.8	73.8

Table 4.5 – **Impact of output dimensionality.** Top-1 ImageNet accuracy for linear models trained on frozen features.

Impact of output dimensionality. In Table 4.5, we evaluate the influence of the output dimensionality in SwAV. This is analogous to the number of clusters in DeepCluster (see Figure 3.3(c) in Section 3.2.2). We train ResNet-50 with SwAV for 400 epochs with $2 \times 160 + 4 \times 96$ crops and evaluate the performance by training a linear classifier on top of frozen final representations. We observe in Table 4.5 that varying the dimensionality by an order of magnitude (3k-100k) does not affect much the performance (at most 0.3 on ImageNet). This suggests that the number K of clusters or latent classes has little influence as long as it is "enough". Throughout the chapter, we train SwAV with $K = 3000$.

4.5 SwAV Pre-Training in the Wild

Self-supervised research has been motivated by the fact that developing a model that can learn with no supervision could benefit from large amount of data; and adapt to any random internet image. So far, self-supervised learning with SwAV or with other approaches [Chen et al., 2020b, Grill et al., 2020] has achieved impressive results in controlled environments (i.e. ImageNet without labels). In this section, we test if these promising achievements hold in the settings these methods were designed for, i.e., is it possible to learn high-quality visual features using self-supervised training on random internet images?

Method	Frozen	Finetuned
Random weights	15.0	76.5
SimCLR [Chen et al., 2020b]	60.4	77.2
SwAV	**66.5**	**77.8**

Table 4.6 – **Pretraining ResNet-50 with SwAV on uncurated data from Instagram.** Top-1 accuracy on ImageNet for pretrained models on an uncurated set of 1B random Instagram images. We compare ResNet-50 pretrained with either SimCLR or SwAV on two downstream tasks: linear classification on frozen features or finetuned features.

4.5.1 Scaling self-supervised learning

Scaling the data. First, we pre-train SwAV on an uncurated dataset of 1 billion random public non-EU images from Instagram by keeping the same architecture as in our previous experiments, i.e., ResNet-50. This is to test if SwAV can serve as a pre-training method for supervised learning. In Table 4.6, we measure the performance of ResNet-50 models when transferring to ImageNet with frozen or finetuned features. We compare SwAV with a randomly initialized network and with a network pre-trained on the same data using SimCLR [Chen et al., 2020b].

First, we observe that SwAV maintains a similar gain of 6% over SimCLR as when pre-trained on ImageNet (see Figure 4.1), showing that our improvements do not depend on the data distribution. We also see that pre-training with SwAV on random images significantly improves over training from scratch on ImageNet ($+1.3\%$). This result is in line with Caron et al. [2019] and He et al. [2020]. This preliminary experiment motivates the exploration of the limits of pre-training as we increase not only the dataset size but also the model capacity. We consider the variants of the ResNeXt architecture [Xie et al., 2017] as in Mahajan et al. [2018] and also models from the RegNet family [Radosavovic et al., 2020] which we describe in the following paragraph.

Scale efficient model family: RegNetY. Scaling data and model capacity jointly requires using architectures that are efficient in terms of both memory and runtime. RegNets are a family of models designed for this purpose and we briefly describe them in this paragraph. We refer to Radosavovic et al. [2020] for more details. RegNets are a family of architectures defined by a design space of convnets consisting of 4 stages, with each stage containing a series of identical blocks, while keeping the structure of their blocks fixed – namely the residual bottleneck block of He et al. [2016]. Here, we focus on the RegNetY architectures, that add a Squeeze-and-excitation op [Hu et al., 2018] to the standard RegNets to further

Pretraining	Arch.	#param	Top-1
Same architecture			
Scratch	RX101-32x8d	91M	79.6
Hashtag pred. [Mahajan et al., 2018]	RX101-32x8d	91M	82.6
SwAV	RX101-32x8d	91M	81.6
Different architectures			
Hashtag pred. [Mahajan et al., 2018]	RX101-32x48d	831M	85.4
SwAV	RG128	693M	83.8
SwAV	RG256	1.3B	**84.2**

Table 4.7 – **Comparison with weakly-supervised pretraining on curated data.** We compare pretraining a ResNeXt101-32dx8d with self-supervision on *random* images with pretraining on filtered images *labeled* with hashtags that are similar to ImageNet classes [Mahajan et al., 2018]. We report top-1 accuracy on ImageNet with finetuning. For completeness, we also report the best performance reported with larger architectures.

improve their performance. The RegNetY model family is parameterized by 5 parameters, allowing the search of a good instance with a certain number of FLOPs with reasonable resources. The models we used were all searched on ImageNet using the same procedure as Radosavovic et al. [2020]. We believe our results can further be improved by searching for RegNetYs directly on our self-supervised pre-training task. Our model of focus is the `RegNetY-256GF` architecture. Its parametrization is given by the scaling rules of RegNets [Radosavovic et al., 2020]:

$$w_0 = 640, w_a = 230.83, w_m = 2.53, \text{group width} = 373$$

It has 4 stages with stage depths (2, 7, 17, 1) and stage widths (528, 1056, 2904, 7392), leading to a total of 695.5M parameters. It takes 6125ms for a single training iteration over $8,704$ images on 512 V100 32GB NVIDIA GPUs. Training this model on 1 billion images requires $114,890$ training iterations for a batch size of $8,704$ images, summing to 8 days of training over 512 GPUs.

4.5.2 Comparing to weakly-supervised pre-training

Many online images have some metadata, e.g., hashtags or geo-localization, that can be leveraged during pre-training. In particular, Mahajan et al. [2018] show that pre-training by predicting a curated set of hashtags can greatly improve the quality of the resulting visual

features. Their approach requires to filter images and only works in the presence of textual metadata. In Table 4.7, we compare our SwAV pre-training on *random* images to theirs on the same architecture, a ResNeXt101-32x8d, with finetuning. For completeness, we also report their best number with their largest architecture. First, we observe that both pre-trainings improve top-1 accuracy over a model trained from scratch, showing in general the benefits of pre-training. Our approach is also in the same ballpark as theirs even though we do not rely on data curation nor supervision. Note that, when the features are frozen, their approach maintains high performance on ImageNet, with 81.6% top-1 accuracy while our model performance drops significantly. This result is not surprising: they pre-train on data that follows the same concepts as ImageNet classes and thus the learned features are more aligned with the target distribution. Since we pre-train our model on random images, we require a full-finetuning step of 35 epochs to adapt to the target distribution. This experiment shows that the benefits of pre-training with finetuning exist even if the features come from a different image distribution.

Chapter 5

Self-Supervised Vision Transformers

Like the contributions presented in this manuscript so far, most of the research in self-supervised learning has been conducted on convolutional networks (convnets). In this final chapter, we explore if self-supervised learning provides new properties to Vision Transformer (ViT) [Dosovitskiy et al., 2020] that stand out compared to convnets. Beyond the fact that adapting self-supervised methods to this architecture works particularly well, we make the following observations: first, self-supervised ViT features contain explicit information about the semantic segmentation of an image, which does not emerge as clearly with supervised ViTs, nor with convnets. Second, these features are also excellent k-NN classifiers, reaching 78.3% top-1 on ImageNet with a small ViT. Our study also underlines the importance of momentum encoder [He et al., 2020], multi-crop training [Caron et al., 2020], and the use of small patches with ViTs. We implement our findings into a simple self-supervised method, called DINO, which we interpret as a form of self-**di**stillation with **no** labels. We show the synergy between DINO and ViTs by achieving 80.1% top-1 on ImageNet in linear evaluation with ViT-Base. This chapter is based on a work published at ICCV 2021 (see Caron et al. [2021]).

5.1 Introduction

Transformers [Vaswani et al., 2017] have recently emerged as an alternative to convnets for visual recognition [Dosovitskiy et al., 2020, Touvron et al., 2020, Zhao et al., 2020]. Their adoption has been coupled with a training strategy inspired by natural language processing (NLP), that is, pretraining on large quantities of data and finetuning on the target dataset [Devlin et al., 2018, Radford et al., 2019]. The resulting Vision Transformers (ViT) [Dosovitskiy et al., 2020] are competitive with convnets but, they have not yet delivered clear benefits over them: they are computationally more demanding, require more

training data, and their features do not exhibit unique properties.

In this paper, we question whether the muted success of Transformers in vision can be explained by the use of supervision in their pre-training. Our motivation is that one of the main ingredients for the success of Transformers in NLP was the use of self-supervised pretraining, in the form of close procedure in BERT [Devlin et al., 2018] or language modeling in GPT [Radford et al., 2019]. These self-supervised pretraining objectives use the words in a sentence to create pretext tasks that provide a richer learning signal than the supervised objective of predicting a single label per sentence. Similarly, in images, image-level supervision often reduces the rich visual information contained in an image to a single concept selected from a predefined set of a few thousand categories of objects [Russakovsky et al., 2015].

While the self-supervised pretext tasks used in NLP are text specific, many existing self-supervised methods have shown their potential on images with convnets [Caron et al., 2020, Chen et al., 2020b, Grill et al., 2020, He et al., 2020]. They typically share a similar structure but with different components designed to avoid trivial solutions (collapse) or to improve performance [Chen and He, 2020]. In this work, inspired from these methods, we study the impact of self-supervised pretraining on ViT features. Of particular interest, we have identified several interesting properties that do not emerge with supervised ViTs, nor with convnets:

— Self-supervised ViT features explicitly contain the scene layout and, in particular, object boundaries. This information is directly accessible in the self-attention modules of the last block.

— Self-supervised ViT features perform particularly well with a basic nearest neighbors classifier (k-NN). For instance, a small ViT achieves 78.3% top-1 accuracy on Imagenet with a k-NN classifier on frozen features, *without any finetuning, linear classifier nor data augmentation*. This property is not as prominent when using convnets, nor when training ViT networks with other self-supervised components.

The emergence of segmentation masks seems to be a property shared across self-supervised methods. However, the good performance with k-NN only emerge with DINO. Another finding from our study is the importance of using smaller patches with ViTs to improve the quality of the resulting features.

Overall, our findings about the importance of certain components for ViT lead us to design a simple self-supervised approach that can be interpreted as a form of knowledge **di**stillation [Hinton et al., 2015] with **no** labels. The resulting framework, DINO, simplifies self-supervised training by directly predicting the output of a teacher network—built with a momentum encoder—by using a standard cross-entropy loss. Interestingly, our method can work with only a centering and sharpening of the teacher output to avoid

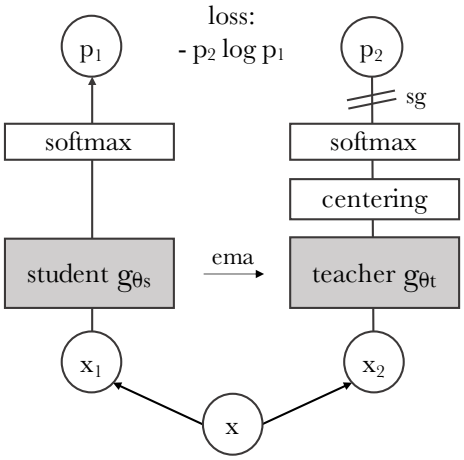


Figure 5.1 – **Self-distillation with no labels.** We illustrate DINO in the case of one single pair of views (x_1, x_2) for simplicity. The model passes two different random transformations of an input image to the student and teacher networks. Both networks have the same architecture but different parameters. The output of the teacher network is centered with a mean computed over the batch. Each networks outputs a K dimensional feature that is normalized with a temperature softmax over the feature dimension. Their similarity is then measured with a cross-entropy loss. We apply a stop-gradient (sg) operator on the teacher to propagate gradients only through the student. The teacher parameters are updated with an exponential moving average (ema) of the student parameters.

collapse, while other popular components such as predictor [Grill et al., 2020], equipartition constraints [Caron et al., 2020] or contrastive loss [He et al., 2020] add little benefits in terms of stability or performance. Of particular importance, our framework is flexible and works on both convnets and ViTs without the need to modify the architecture, nor adapt the internal normalizations [Richemond et al., 2020].

We further validate the synergy between DINO and ViT by outperforming previous self-supervised features on the ImageNet linear classification benchmark with 80.1% top-1 accuracy with a ViT-Base with small patches. We also confirm that DINO works with convnets by matching the state of the art with a ResNet-50 architecture. Finally, we discuss different scenarios to use DINO with ViTs in case of limited computation and memory capacity. In particular, training DINO with ViT takes just two 8-GPU servers over 3 days to achieve 76.1% on ImageNet linear benchmark, which outperforms self-supervised systems based on convnets of comparable sizes with significantly reduced compute requirements [Caron et al., 2020, Grill et al., 2020].

5.2 Related work

As we interpret our method as a kind of unsupervised self-distillation, we give a brief overview of related work about supervised knowledge distillation. We also quickly review the Vision Transformer architecture.

5.2.1 Self-training and knowledge distillation.

Self-training aims at improving the quality of features by propagating a small initial set of annotations to a large set of unlabeled instances. This propagation can either be done with hard assignments of labels [Lee et al., 2013, Xu et al., 2020, Yalniz et al., 2019] or with a soft assignment [Xie et al., 2020b]. When using soft labels, the approach is often referred to as knowledge distillation [Buciluǎ et al., 2006, Hinton et al., 2015] and has been primarily designed to train a small network to mimic the output of a larger network to compress models. Xie et al. [2020b] have shown that distillation could be used to propagate soft pseudo-labels to unlabeled data in a self-training pipeline, drawing an essential connection between self-training and knowledge distillation. Our work builds on this relation and extends knowledge distillation to the case where no labels are available. Previous works have also combined self-supervised learning and knowledge distillation [Chen et al., 2020c, Fang et al., 2021, Noroozi et al., 2018, Shen et al., 2021], enabling self-supervised model compression and performance gains. However, these works rely on a *pre-trained* fixed teacher while our teacher is dynamically built during training. This way, in our work, knowledge distillation instead of being used as a post-processing step to self-supervised pre-training, is directly cast as a self-supervised objective. Finally, our work is also related to *co-distillation* [Anil et al., 2018] where student and teacher have the same architecture and use distillation during training. However, the teacher in *co-distillation* is also distilling from the student, while it is updated with an average of the student in our work.

5.2.2 Vision transformers

The Transformer architecture has originally been proposed in the context of machine translation by Vaswani et al. [2017]. Since its introduction, this architecture has been successfully applied to sentence representation [Devlin et al., 2018], language modeling [Radford et al., 2019] and more recently speech recognition [Baevski et al., 2020]. While there have been many attempts at adapting the Transformer architecture to images [Child et al., 2019, Cordonnier et al., 2020, Parmar et al., 2018], it is only recently that standard Transformers have obtained competitive results on challenging image classification datasets [Dosovitskiy et al., 2020, Zhao et al., 2020]. In particular, Dosovitskiy et al. [2020]

model	blocks	dim	heads	#tokens	#params	im/s
ResNet-50	–	2048	–	–	23M	1237
ViT-S/16	12	384	6	197	21M	1007
ViT-S/8	12	384	6	785	21M	180
ViT-B/16	12	768	12	197	85M	312
ViT-B/8	12	768	12	785	85M	63

Table 5.1 – **Networks configuration.** "Blocks" is the number of Transformer blocks, "dim" is channel dimension and "heads" is the number of heads in multi-head attention. "# tokens" is the length of the token sequence when considering 224^2 resolution inputs, "# params" is the total number of parameters (without counting the projection head) and "im/s" is the inference time on a NVIDIA V100 GPU with 128 samples per forward.

show that a patch based Transformer is particularly suited for image classification. While these models originally required a lot of annotated data, Touvron et al. [2020] show that they also achieve competitive performance when trained on ImageNet alone by means of strong regularization and by guiding their training with a convnet through distillation.

Overall, we refer the reader to Vaswani et al. [2017] for details about Transformers and to Dosovitskiy et al. [2020] for its adaptation to images. In this chapter, we follow the implementation used in `DeiT` [Touvron et al., 2020]. We summarize the configuration of the different networks used in this chapter in Table 5.1. The ViT architecture takes as input a grid of non-overlapping contiguous image patches of resolution $N \times N$. In this paper we typically use $N = 16$ ("/16") or $N = 8$ ("/8"). The patches are then passed through a linear layer to form a set of embeddings. We add an extra learnable token to the sequence [Devlin et al., 2018, Dosovitskiy et al., 2020]. The role of this token is to aggregate information from the entire sequence. We refer to this token as the class token `[CLS]` for consistency with previous works [Devlin et al., 2018, Dosovitskiy et al., 2020, Touvron et al., 2020], even though it is not attached to any label nor supervision in our case. The set of patch tokens and `[CLS]` token are fed to a standard Transformer network with a "pre-norm" layer normalization [Chen et al., 2018, Klein et al., 2017]. The Transformer is a sequence of self-attention and feed-forward layers, paralleled with skip connections. The self-attention layers update the token representations by looking at the other token representations with an attention mechanism [Bahdanau et al., 2014].

5.3 DINO Methodology

5.3.1 SSL with knowledge distillation

The framework used for this work, DINO, shares the same overall structure as most recent self-supervised approaches [Caron et al., 2020, Chen et al., 2020b, Chen and He, 2020, Grill et al., 2020, He et al., 2020] following the view-invariant paradigm described in Section 2.3. However, our method shares also similarities with knowledge distillation [Hinton et al., 2015] and we present it under this angle. We illustrate DINO in Figure 5.1 and propose a pseudo-code implementation in Algorithm 1.

Knowledge distillation is a learning paradigm where we train a student network g_{θ_s} to match the output of a given teacher network g_{θ_t}, parameterized by θ_s and θ_t respectively. Given an input image x, both networks output probability distributions over K dimensions denoted by P_s and P_t. The probability P is obtained by normalizing the output of the network g with a softmax function. More precisely,

$$P_s(x)^{(i)} = \frac{\exp(g_{\theta_s}(x)^{(i)}/\tau_s)}{\sum_{k=1}^{K} \exp(g_{\theta_s}(x)^{(k)}/\tau_s)}, \tag{5.1}$$

with $\tau_s > 0$ a temperature parameter that controls the sharpness of the output distribution, and a similar formula holds for P_t with temperature τ_t. Given a fixed teacher network g_{θ_t}, we learn to match these distributions by minimizing the cross-entropy loss w.r.t. the parameters of the student network θ_s:

$$\min_{\theta_s} H(P_t(x), P_s(x)), \tag{5.2}$$

where $H(a, b) = -a \log b$.

In the following, we detail how we adapt the problem in Eq. (5.2) to self-supervised learning. First, we construct different distorted views, or crops, of an image with multi-crop strategy (see Section 4.2.4). More precisely, from a given image, we generate a set V of different views. This set contains two *global* views, x_1^g and x_2^g and several *local* views of smaller resolution. All crops are passed through the student while only the *global* views are passed through the teacher, therefore encouraging "local-to-global" correspondences. We minimize the loss:

$$\min_{\theta_s} \sum_{x \in \{x_1^g, x_2^g\}} \sum_{\substack{x' \in V \\ x' \neq x}} H(P_t(x), P_s(x')). \tag{5.3}$$

This loss is general and can be used on any number of views, even only 2. However, we follow the standard setting for multi-crop by using 2 global views at resolution 224^2 covering a large (for example greater than 50%) area of the original image, and several local views of resolution 96^2 covering only small areas (for example less than 50%) of

Algorithm 1 DINO PyTorch pseudo-code w/o multi-crop.

```
# gs, gt: student and teacher networks
# C: center (K)
# tps, tpt: student and teacher temperatures
# l, m: network and center momentum rates
gt.params = gs.params
for x in loader: # load a minibatch x with n samples
    x1, x2 = augment(x), augment(x) # random views

    s1, s2 = gs(x1), gs(x2) # student output n-by-K
    t1, t2 = gt(x1), gt(x2) # teacher output n-by-K

    loss = H(t1, s2)/2 + H(t2, s1)/2
    loss.backward() # back-propagate

    # student, teacher and center updates
    update(gs) # SGD
    gt.params = l*gt.params + (1-l)*gs.params
    C = m*C + (1-m)*cat([t1, t2]).mean(dim=0)

def H(t, s):
    t = t.detach() # stop gradient
    s = softmax(s / tps, dim=1)
    t = softmax((t - C) / tpt, dim=1) # center + sharpen
    return - (t * log(s)).sum(dim=1).mean()
```

the original image. Both networks share the same architecture g with different sets of parameters θ_s and θ_t. We learn the parameters θ_s by minimizing Eq. (5.3) with stochastic gradient descent.

Teacher network. Unlike knowledge distillation, we do not have a teacher g_{θ_t} given *a priori* and hence, we build it from past iterations of the student network. We study different update rules for the teacher in Section 5.6.2 and show that freezing the teacher network over an epoch works surprisingly well in our framework, while copying the student weight for the teacher fails to converge. Of particular interest, using an exponential moving average (EMA) on the student weights, i.e., a momentum encoder [He et al., 2020], is particularly well suited for our framework. The update rule is $\theta_t \leftarrow \lambda\theta_t + (1-\lambda)\theta_s$, with λ following a cosine schedule from 0.996 to 1 during training [Grill et al., 2020]. Originally the momentum encoder has been introduced as a substitute for a queue in contrastive learning [He et al., 2020]. However, in our framework, its role differs since we do not have a queue nor a contrastive loss, and may be closer to the role of the mean teacher used in self-training [Tarvainen and Valpola, 2017]. Indeed, we observe that this teacher performs a form of model ensembling similar to Polyak-Ruppert averaging with an exponential decay [Polyak and Juditsky, 1992, Ruppert, 1988]. Using Polyak-Ruppert averaging for model ensembling is a standard practice to improve the performance of a model [Jean et al., 2014]. We observe that this teacher has better performance than the student throughout the training, and hence, guides the training of the student by providing target features of higher quality. This dynamic was not observed in previous works [Grill et al., 2020, Richemond

et al., 2020].

Network architecture. The neural network g is composed of a backbone f (ViT [Dosovitskiy et al., 2020] or ResNet [He et al., 2016]), and of a projection head h: $g = h \circ f$. The features used in downstream tasks are the backbone f output. The projection head consists of a 3-layer multi-layer perceptron (MLP) with hidden dimension 2048 followed by ℓ_2 normalization and a weight normalized fully connected layer [Salimans and Kingma, 2016] with K dimensions, which is similar to the design from SwAV [Caron et al., 2020]. We have tested other projection heads and this particular design appears to work best for DINO. We do not use a predictor [Chen and He, 2020, Grill et al., 2020], resulting in the exact same architecture in both student and teacher networks. Of particular interest, we note that unlike standard convnets, ViT architectures do not use batch normalizations (BN) by default. Therefore, when applying DINO to ViT we do not use any BN also in the projection heads, making the system *entirely BN-free*.

Avoiding collapse. Several self-supervised methods differ by the operation used to avoid collapse, either through contrastive loss [Wu et al., 2018], clustering constraints [Caron et al., 2018, 2020], predictor [Grill et al., 2020] or batch normalizations [Grill et al., 2020, Richemond et al., 2020]. While our framework can be stabilized with multiple normalizations [Caron et al., 2020], it can also work with only a centering and sharpening of the momentum teacher outputs to avoid model collapse. As shown experimentally in Section 5.6.3, centering prevents one dimension to dominate but encourages collapse to the uniform distribution, while the sharpening has the opposite effect. Applying both operations balances their effects which is sufficient to avoid collapse in presence of a momentum teacher. Choosing this method to avoid collapse trades stability for less dependence over the batch: the centering operation only depends on first-order batch statistics and can be interpreted as adding a bias term c to the teacher: $g_t(x) \leftarrow g_t(x) + c$. The center c is updated with an exponential moving average, which allows the approach to work well across different batch sizes as shown in Section 5.6.5:

$$c \leftarrow mc + (1-m)\frac{1}{B}\sum_{i=1}^{B} g_{\theta_t}(x_i), \tag{5.4}$$

where $m > 0$ is a rate parameter and B is the batch size. Output sharpening is obtained by using a low value for the temperature τ_t in the teacher softmax normalization.

5.3.2 Implementation and evaluation protocols

In this section, we provide the implementation details to train with DINO and present the evaluation protocols used in our experiments.

Implementation details. We pretrain the models on the ImageNet dataset [Russakovsky et al., 2015] without labels. We train with the adamw optimizer [Loshchilov and Hutter, 2018] and a batch size of 1024, distributed over 16 GPUs when using ViT-S/16. The learning rate is linearly ramped up during the first 10 epochs to its base value determined with the following linear scaling rule [Goyal et al., 2017]: $lr = 0.0005 * \text{batchsize}/256$. After this warm-up, we decay the learning rate with a cosine schedule [Loshchilov and Hutter, 2016]. The weight decay also follows a cosine schedule from 0.04 to 0.4. The temperature τ_s is set to 0.1 while we use a linear warm-up for τ_t from 0.04 to 0.07 during the first 30 epochs. We follow the data augmentations of BYOL [Grill et al., 2020] (color jittering, Gaussian blur and solarization) and multi-crop [Caron et al., 2020] with a bicubic interpolation to adapt the position embeddings to the scales [Dosovitskiy et al., 2020, Touvron et al., 2020]. The code and models to reproduce our results is publicly available at `https://github.com/facebookresearch/dino`.

Evaluation protocols. Standard protocols for self-supervised learning are to either learn a linear classifier on frozen features [He et al., 2020, Zhang et al., 2016] or to finetune the features on downstream tasks. For linear evaluations, we apply random resize crops and horizontal flips augmentation during training, and report accuracy on a central crop. For finetuning evaluations, we initialize networks with the pretrained weights and adapt them during training. However, both evaluations are sensitive to hyperparameters, and we observe a large variance in accuracy between runs when varying the learning rate for example. We thus also evaluate the quality of features with a simple weighted nearest neighbor classifier (k-NN) as in Wu et al. [2018]. We freeze the pretrain model to compute and store the features of the training data of the downstream task. The nearest neighbor classifier then matches the feature of an image to the k nearest stored features that votes for the label. We sweep over different number of nearest neighbors and find that 20 NN is consistently working the best for most of our runs. This evaluation protocol does not require any other hyperparameter tuning, nor data augmentation and can be run with only one pass over the downstream dataset, greatly simplifying the feature evaluation.

Method	Arch.	Param.	im/s	Linear	k-NN
Supervised	RN50	23	1237	79.3	79.3
SCLR [Chen et al., 2020b]	RN50	23	1237	69.1	60.7
MoCov2 [Chen et al., 2020e]	RN50	23	1237	71.1	61.9
InfoMin [Tian et al., 2020b]	RN50	23	1237	73.0	65.3
BarlowT [Zbontar et al., 2021]	RN50	23	1237	73.2	66.0
OBoW [Gidaris et al., 2020b]	RN50	23	1237	73.8	61.9
BYOL [Grill et al., 2020]	RN50	23	1237	74.4	64.8
DCv2 [Caron et al., 2020]	RN50	23	1237	75.2	67.1
SwAV [Caron et al., 2020]	RN50	23	1237	**75.3**	65.7
DINO	RN50	23	1237	**75.3**	**67.5**
Supervised	ViT-S	21	1007	79.8	79.8
BYOL* [Grill et al., 2020]	ViT-S	21	1007	71.4	66.6
MoCov2* [Chen et al., 2020e]	ViT-S	21	1007	72.7	64.4
SwAV* [Caron et al., 2020]	ViT-S	21	1007	73.5	66.3
DINO	ViT-S	21	1007	**77.0**	**74.5**
Comparison across architectures					
SCLR [Chen et al., 2020b]	RN50w4	375	117	76.8	69.3
SwAV [Caron et al., 2020]	RN50w2	93	384	77.3	67.3
BYOL [Grill et al., 2020]	RN50w2	93	384	77.4	–
DINO	ViT-B/16	85	312	78.2	76.1
SwAV [Caron et al., 2020]	RN50w5	586	76	78.5	67.1
BYOL [Grill et al., 2020]	RN50w4	375	117	78.6	–
BYOL [Grill et al., 2020]	RN200w2	250	123	79.6	73.9
DINO	ViT-S/8	21	180	79.7	**78.3**
SCLRv2 [Chen et al., 2020c]	RN152w3+SK	794	46	79.8	73.1
DINO	ViT-B/8	85	63	**80.1**	77.4

Table 5.2 – **Linear and k-NN classification on ImageNet.** We report top-1 accuracy for linear and k-NN evaluations on the validation set of ImageNet for different self-supervised methods. We focus on ResNet-50 and ViT-small architectures, but also report the best results obtained across architectures. * are run by us. We run the k-NN evaluation for models with official released weights. The throughput (im/s) is calculated on a NVIDIA V100 GPU with 128 samples per forward. Parameters (M) are of the feature extractor.

5.4 Comparing with SSL Frameworks on ImageNet

We first validate the DINO framework used in this study with the standard self-supervised benchmark on ImageNet. We consider two different settings: comparison with the same architecture and across architectures.

Comparing with the same architecture. In top panel of Table 5.2, we compare DINO with other self-supervised methods with the same architecture, either a ResNet-50 [He et al., 2016] or a ViT-small (which follows the design of DeiT-S [Touvron et al., 2020]). The choice of ViT-S is motivated by its similarity with ResNet-50 along several axes: number of parameters (21M vs 23M), throughput (1237/sec VS 1007 im/sec) and supervised performance on ImageNet with the training procedure of Touvron et al. [2020] (79.3% VS 79.8%). First, we observe that DINO performs on par with the state of the art on ResNet-50, validating that DINO works in the standard setting. When we switch to a ViT architecture, DINO outperforms BYOL, MoCov2 and SwAV by +3.5% with linear classification and by +7.9% with k-NN evaluation. More surprisingly, the performance with a simple k-NN classifier is almost on par with a linear classifier (74.5% versus 77.0%). This property emerges only when using DINO with ViT architectures, and does not appear with other existing self-supervised methods nor with a ResNet-50.

Comparing across architectures. On the bottom panel of Table 5.2, we compare the best performance obtained across architectures. The interest of this setting is not to compare methods directly, but to evaluate the limits of a ViT trained with DINO when moving to larger architectures. While training a larger ViT with DINO improves the performance, reducing the size of the patches ("/8" variants) has a bigger impact on the performance. While reducing the patch size do not add parameters, it still leads to a significant reduction of running time, and larger memory usage. Nonetheless, a base ViT with 8×8 patches trained with DINO achieves 80.1% top-1 in linear classification and 77.4% with a k-NN classifier with $10\times$ less parameters and $1.4\times$ faster run time than previous state of the art [Chen et al., 2020c].

5.5 Properties of Self-Supervised ViT

We evaluate properties of the DINO features in terms of nearest neighbor search, retaining information about object location and transferability to downstream tasks.

			$\mathcal{R}$Ox		$\mathcal{R}$Par	
Pretrain	Arch.	Pretrain	M	H	M	H
Sup. [Revaud et al., 2019]	RN101+R-MAC	ImNet	49.8	18.5	74.0	**52.1**
Sup.	ViT-S/16	ImNet	33.5	8.9	63.0	37.2
DINO	ResNet-50	ImNet	35.4	11.1	55.9	27.5
DINO	ViT-S/16	ImNet	41.8	13.7	63.1	34.4
DINO	ViT-S/16	GLDv2	**51.5**	**24.3**	**75.3**	51.6

Table 5.3 – **Image retrieval.** We compare the performance in retrieval of off-the-shelf features pretrained with supervision or with DINO on ImageNet and Google Landmarks v2 (GLDv2) dataset. We report mAP on revisited Oxford and Paris. Pretraining with DINO on a landmark dataset performs particularly well. For reference, we also report the best retrieval method with off-the-shelf features [Revaud et al., 2019].

5.5.1 Nearest neighbor retrieval with DINO ViT

The results on ImageNet classification have exposed the potential of our features for tasks relying on nearest neighbor retrieval. In this set of experiments, we further consolidate this finding on landmark retrieval and copy detection tasks.

Image Retrieval. We consider the revisited [Radenović et al., 2018a] Oxford and Paris image retrieval datasets [Philbin et al., 2008]. They contain 3 different splits of gradual difficulty with query/database pairs. We report the Mean Average Precision (mAP) for the Medium (M) and Hard (H) splits. In Table 5.3, we compare the performance of different *off-the-shelf* features obtained with either supervised or DINO training. We freeze the features and directly apply k-NN for retrieval. We observe that DINO features outperform those trained on ImageNet with labels.

An advantage of SSL approaches is that they can be trained on any dataset, without requiring any form of annotations. We train DINO on the 1.2M clean set from Google Landmarks v2 (GLDv2) [Weyand et al., 2020], a dataset of landmarks designed for retrieval purposes. DINO ViT features trained on GLDv2 are remarkably good, outperforming previously published methods based on off-the-shelf descriptors [Revaud et al., 2019, Tolias et al., 2015].

Copy detection. We also evaluate the performance of ViTs trained with DINO on a copy detection task. We report the mean average precision on the "strong" subset of the INRIA

Method	Arch.	Dim.	Resolution	mAP
Multigrain [Berman et al., 2019]	ResNet-50	2048	224^2	75.1
Multigrain [Berman et al., 2019]	ResNet-50	2048	largest side 800	82.5
Supervised [Touvron et al., 2020]	ViT-B/16	1536	224^2	76.4
DINO	ViT-B/16	1536	224^2	81.7
DINO	ViT-B/8	1536	320^2	**85.5**

Table 5.4 – **Copy detection.** We report the mAP performance in copy detection on Copydays "strong" subset [Douze et al., 2009]. For reference, we also report the performance of the multigrain model [Berman et al., 2019], trained specifically for particular object retrieval.

Copydays dataset [Douze et al., 2009]. The task is to recognize images that have been distorted by blur, insertions, print and scan, etc. Following prior work Berman et al. [2019], we add 10k distractor images randomly sampled from the YFCC100M dataset [Thomee et al., 2015]. We perform copy detection directly with cosine similarity on the features obtained from our pretrained network. The features are obtained as the concatenation of the output `[CLS]` token and of the GeM pooled [Radenović et al., 2018b] output patch tokens. This results in a 1536d descriptor for ViT-B. Following Berman et al. [2019], we apply whitening on the features. We learn this transformation on an extra 20K random images from YFCC100M, distinct from the distractors. Table 5.4 shows that ViT trained with DINO is very competitive on copy detection.

Image classification. In Table 5.5, we evaluate the frozen representations given by ResNet-50 or ViT-small pre-trained with DINO with two evaluation protocols: linear or k-NN. For both evaluations, we extract representations from a pre-trained network without using any data augmentation. Then, we perform classification either with weighted k-NN or with a linear regression learned with `cyanure` library [Mairal, 2019]. In Table 5.5 we see that ViT-S accuracies are better than accuracies obtained with RN50 both with a linear or a k-NN classifier. However, the performance gap when using the k-NN evaluation is much more significant than when considering linear evaluation. For example on ImageNet 1%, ViT-S outperforms ResNet-50 by a large margin of $+14.1\%$ with k-NN evaluation. This suggests that transformers architectures trained with DINO might offer more model flexibility that benefits the k-NN evaluation. K-NN classifiers have the great advantage of being fast and light to deploy, without requiring any domain adaptation. Overall, ViT trained with DINO provides features that combine particularly well with k-NN classifiers.

	Logistic			k-NN		
	RN50	ViT-S	Δ	RN50	ViT-S	Δ
Inet 100%	72.1	75.7	3.6	67.5	74.5	7.0
Inet 10%	67.8	72.2	4.4	59.3	69.1	9.8
Inet 1%	55.1	64.5	9.4	47.2	61.3	14.1
Pl. 10%	53.4	52.1	-1.3	46.9	48.6	1.7
Pl. 1%	46.5	46.3	-0.2	39.2	41.3	2.1
VOC07	88.9	89.2	0.3	84.9	88.0	3.1
FLOWERS	95.6	96.4	0.8	87.9	89.1	1.2
Average Δ			2.4			**5.6**

Table 5.5 – **k-NN and linear evaluation for ViT-S/16 and ResNet-50 pre-trained with DINO.** We use ImageNet-1k [Russakovsky et al., 2015] ("Inet"), Places205 [Zhou et al., 2014], PASCAL VOC [Everingham et al., 2010] and Oxford-102 flowers ("FLOWERS") [Nilsback and Zisserman, 2008]. ViT trained with DINO provides features that are particularly k-NN friendly.

Class Representation As a final study of the k-NN property, we propose to look at the distribution of ImageNet concepts in the feature space from DINO. We represent each ImageNet class with the average feature vector for its validation images. We reduce the dimension of these features to 30 with PCA, and run t-SNE with a perplexity of 20, a learning rate of 200 for 5000 iterations. We present the resulting class embeddings in Figure 5.2. Our model recovers structures between classes: similar animal species are grouped together, forming coherent clusters of birds (top) or dogs, and especially terriers (far right).

5.5.2 Discovering the semantic layout of scenes

As shown qualitatively in Figure 5.3, our self-attention maps contain information about the segmentation of an image. In this study, we measure this property on a standard benchmark as well as by directly probing the quality of masks generated from these attention maps.

Video instance segmentation. In Table 5.6, we evaluate the output patch tokens on the DAVIS-2017 video instance segmentation benchmark [Pont-Tuset et al., 2017]. We follow the experimental protocol in Jabri et al. [2020] and segment scenes with a nearest-neighbor

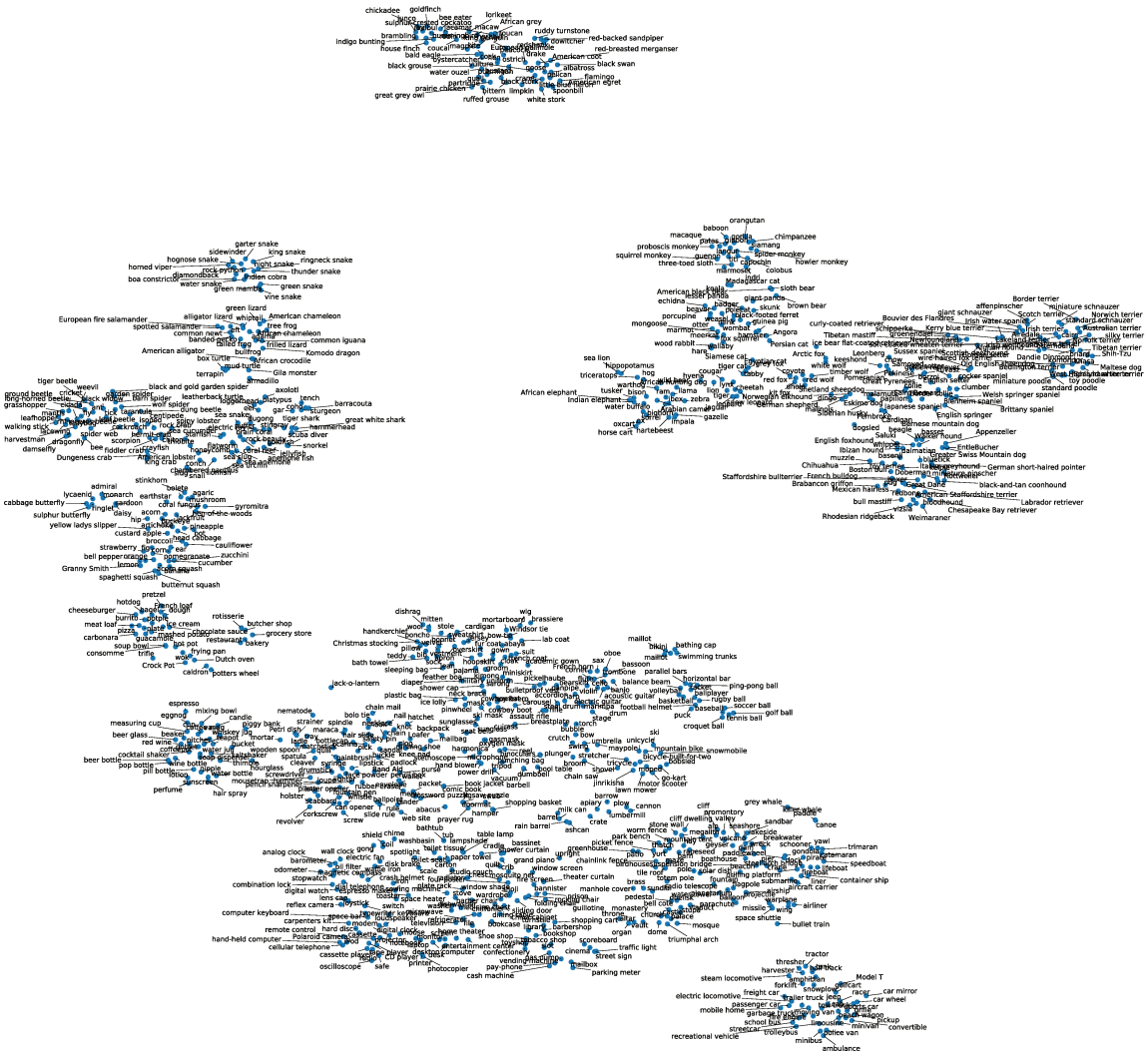


Figure 5.2 – **t-SNE visualization of ImageNet classes as represented using DINO.** For each class, we obtain the embedding by taking the average feature for all images of that class in the validation set.

between consecutive frames; we thus do not train any model on top of the features, nor finetune any weights for the task. We observe in Table 5.6 that even though our training objective nor our architecture are designed for dense tasks, the performance is competitive on this benchmark. Since the network is not finetuned, the output of the model must have

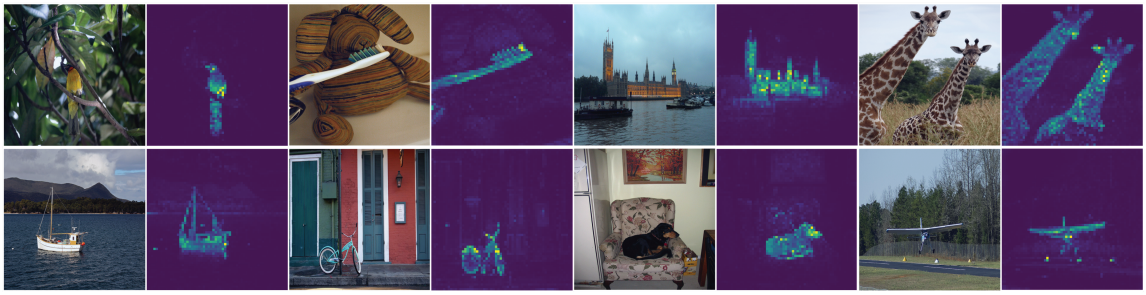


Figure 5.3 – **Self-attention from a Vision Transformer with 8×8 patches trained with no supervision.** We look at the self-attention of the `[CLS]` token on the heads of the last layer. This token is not attached to any label nor supervision. These maps show that the model automatically learns class-specific features leading to unsupervised object segmentations.

retained some spatial information. Finally, for this dense recognition task, the variants with small patches ("/8") perform much better (+9.1% $(\mathcal{J}\&\mathcal{F})_m$ for ViT-B).

Probing the self-attention map. In Figure 5.4, we show that different heads can attend to different semantic regions of an image, even when they are occluded (the bushes on the third row) or small (the flag on the second row). Visualizations are obtained with 480p images, resulting in sequences of 3601 tokens for ViT-S/8. In Figure 5.5, we show that a supervised ViT does not attend well to objects in presence of clutter both qualitatively and quantitatively. We report the Jaccard similarity between the ground truth and segmentation masks obtained by thresholding the self-attention map to keep 60% of the mass. Note that the self-attention maps are smooth and not optimized to produce a mask. Nonetheless, we see a clear difference between the supervised or DINO models with a significant gap in terms of Jaccard similarities. Note that self-supervised convnets also contain information about segmentations but it requires dedicated methods to extract it from their weights [Gur et al., 2020]. We also evaluate the self-attention maps with the Jaccard similarity criterion on ViT-S/16 trained with other self-supervised methods. In particular, we obtain 45.1 for DINO without multi-crop, 46.3 for MoCo-v2, 47.8 for BYOL and 46.8 for SwAV. These results are similar to that obtained with DINO, which shows that the properties about salient self-attention maps in ViT is observed across different self-supervised methods.

Method	Data	Arch.	$(\mathcal{J}\&\mathcal{F})_m$	$\mathcal{J}_m$	$\mathcal{F}_m$
Supervised					
ImageNet	INet	ViT-S/8	66.0	63.9	68.1
STM [Oh et al., 2019]	I/D/Y	RN50	81.8	79.2	84.3
Self-supervised					
CT [Wang et al., 2019]	VLOG	RN50	48.7	46.4	50.0
MAST [Lai et al., 2020]	YT-VOS	RN18	65.5	63.3	67.6
STC [Jabri et al., 2020]	Kinetics	RN18	67.6	64.8	70.2
DINO	INet	ViT-S/16	61.8	60.2	63.4
DINO	INet	ViT-B/16	62.3	60.7	63.9
DINO	INet	ViT-S/8	**69.9**	**66.6**	**73.1**
DINO	INet	ViT-B/8	**71.4**	**67.9**	**74.9**

Table 5.6 – **DAVIS 2017 Video object segmentation.** We evaluate the quality of frozen features on video instance tracking. We report mean region similarity $\mathcal{J}_m$ and mean contour-based accuracy $\mathcal{F}_m$. We compare with existing self-supervised methods and a supervised ViT-S/8 trained on ImageNet. Image resolution is 480p.

	Cifar_{10}	Cifar_{100}	INat_{18}	INat_{19}	Flwrs	Cars	INet
ViT-S/16							
Sup. [Touvron et al., 2020]	**99.0**	89.5	70.7	76.6	98.2	92.1	79.9
DINO	**99.0**	**90.5**	**72.0**	**78.2**	**98.5**	**93.0**	**81.5**
ViT-B/16							
Sup. [Touvron et al., 2020]	99.0	90.8	**73.2**	77.7	98.4	92.1	81.8
DINO	**99.1**	**91.7**	72.6	**78.6**	**98.8**	**93.0**	**82.8**

Table 5.7 – **Transfer learning by finetuning pretrained models on different datasets.** We report top-1 accuracy. Self-supervised pretraining with DINO transfers better than supervised pretraining.

5.5.3 Transfer learning on downstream tasks

In Table 5.7, we evaluate the quality of the features pretrained with DINO on different downstream tasks. We compare with features from the same architectures trained with supervision on ImageNet. We follow the protocol used in Touvron et al. [2020] and finetune the features on each downstream task. We observe that for ViT architectures, self-supervised

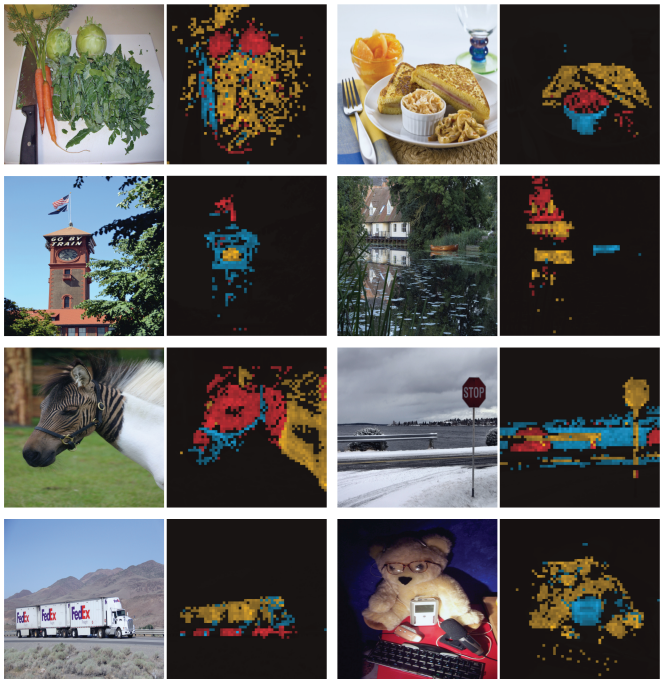


Figure 5.4 – **Attention maps from multiple heads.** We consider the heads from the last layer of a ViT-S/8 trained with DINO and display the self-attention for `[CLS]` token query. Different heads, materialized by different colors, focus on different locations that represents different objects or parts.

pretraining transfers better than features trained with supervision, which is consistent with observations made on convolutional networks (see Section 4.3.2 and Table 4.2). Finally, self-supervised pretraining greatly improves results on ImageNet (+1-2%).

5.5.4 Self-supervised ImageNet pre-training of ViT.

In this experiment, we evaluate the impact of pre-training a supervised ViT model with our method. In Table 5.8, we compare the performance of supervised ViT models that are initialized with different pretraining or guided during training with an additional pretrained convnet. The first set of models are pretrained with and without supervision on the large curated dataset composed of 300M images. The second set of models are trained with hard knowledge distillation from a pretrained supervised RegNetY [Radosavovic et al., 2020]. The last set of models do not use any additional data nor models, and are initialized either randomly or after a pretraining with DINO on ImageNet. Compare to random initialization,

	Random	Supervised	DINO
ViT-S/16	22.0	27.3	45.9
ViT-S/8	21.8	23.7	44.7

Figure 5.5 – **Segmentations from supervised versus DINO.** We visualize masks obtained by thresholding the self-attention maps to keep 60% of the mass. On top, we show the resulting masks for a ViT-S/8 trained with supervision and DINO. We show the best head for both models. The table at the bottom compares the Jaccard similarity between the ground truth and these masks on the validation images of PASCAL VOC12 dataset.

pretraining with DINO leads to a performance gain of +1%. This is not caused by a longer training since pretraining with supervision instead of DINO does not improve performance. Using self-supervised pretraining reduces the gap with models pretrained on extra data or distilled from a convnet.

5.6 Ablation Study and Analyses of DINO

In this section, we empirically study DINO applied to ViT. The model considered for this entire study is ViT-S.

5.6.1 Ablation of different model components

We show the impact of adding different components from self-supervised learning on ViT trained with our framework. All models are trained for 300 epochs.

In Table 5.9, we report different model variants as we add or remove components.

Pretraining				
method	data	res.	tr. proc.	Top-1
Pretrain on additional data				
MMP	JFT-300M	384	[Dosovitskiy et al., 2020]	79.9
Supervised	JFT-300M	384	[Dosovitskiy et al., 2020]	84.2
Train with additional model				
Rand. init.	-	224	[Touvron et al., 2020]	83.4
No additional data nor model				
Rand. init.	-	224	[Dosovitskiy et al., 2020]	77.9
Rand. init.	-	224	[Touvron et al., 2020]	81.8
Supervised	ImNet	224	[Touvron et al., 2020]	81.9
DINO	ImNet	224	[Touvron et al., 2020]	82.8

Table 5.8 – **ImageNet classification with different pretraining.** Top-1 accuracy on ImageNet for supervised ViT-B/16 models using different pretrainings or using an additional pretrained convnet to guide the training. The methods use different image resolution ("res.") and training procedure ("tr. proc."), i.e., data augmentation and optimization. "MPP" is *Masked Patch Prediction*.

First, we observe that in the absence of momentum, our framework does not work (row 2) and more advanced operations, SK for example, are required to avoid collapse (row 9). However, with momentum, using SK has little impact (row 3). In addition, comparing rows 3 and 9 highlights the importance of the momentum encoder for performance. Second, in rows 4 and 5, we observe that multi-crop training and the cross-entropy loss in DINO are important components to obtain good features. We also observe that adding a predictor to the student network has little impact (row 6) while it is critical in BYOL to prevent collapse [Chen and He, 2020, Grill et al., 2020].

Relation to SwAV. In Table 5.10, we further evaluate the differences between DINO and SwAV: the presence of the momentum encoder and the operation on top of the teacher output. In absence of the momentum, a copy of the student with a stop-gradient is used. We consider three operations on the teacher output: `Centering`, `Sinkhorn-Knopp` or a `Softmax` along the batch axis. The `Softmax` is similar to a single Sinkhorn-Knopp iteration as detailed in Section 4.2.2. First, these ablations show that using a momentum encoder significantly improves the performance for ViT (3 versus 6, and 2 versus 5). Second, the momentum encoder also avoids collapse when using only centering (row 1).

	Method	Mom.	SK	MC	Loss	Pred.	k-NN	Lin.
1	DINO	✓	✗	✓	CE	✗	72.8	76.1
2		✗	✗	✓	CE	✗	0.1	0.1
3		✓	✓	✓	CE	✗	72.2	76.0
4		✓	✗	✗	CE	✗	67.9	72.5
5		✓	✗	✓	MSE	✗	52.6	62.4
6		✓	✗	✓	CE	✓	71.8	75.6
7	BYOL	✓	✗	✗	MSE	✓	66.6	71.4
8	MoCov2	✓	✗	✗	INCE	✗	62.0	71.6
9	SwAV	✗	✓	✓	CE	✗	64.7	71.8

SK: Sinkhorn-Knopp, MC: Multi-Crop, Pred.: Predictor
CE: Cross-Entropy, MSE: Mean Square Error, INCE: InfoNCE

Table 5.9 – **Important component for self-supervised ViT pretraining.** Models are trained for 300 epochs with ViT-S/16. We study the different components that matter for the k-NN and linear ("Lin.") evaluations. For the different variants, we highlight the differences from the default DINO setting. The best combination is the momentum encoder with the multi-crop augmentation and the cross-entropy loss. We also report results with BYOL [Grill et al., 2020], MoCo-v2 [Chen et al., 2020e] and SwAV [Caron et al., 2020].

In the absence of momentum, centering the outputs does not work (4) and more advanced operations are required (5, 6). Overall, these ablations highlight the importance of the momentum encoder, not only for performance but also to stabilize training, removing the need for normalization beyond centering.

Relation to MoCo-v2 and BYOL. In Table 5.11, we present in further details the impact of ablating components that differ between DINO, MoCo-v2 and BYOL: the choice of loss, the predictor in the student head, the centering operation, the batch normalization in the projection heads, and finally, the multi-crop augmentation. The loss in DINO is a cross-entropy on sharpened softmax outputs (CE) while MoCo-v2 uses the InfoNCE contrastive loss (INCE) and BYOL a mean squared error on l2-normalized outputs (MSE). No sharpening is applied with the MSE criterion. Though, DINO surprisingly still works when changing the loss function to MSE, but this significantly alters the performance (see rows (1, 2) and (4, 9)). We also observe that adding a predictor has little impact (1, 3). However, in the case of BYOL, the predictor is critical to prevent collapse (7, 8) which is consistent with previous studies [Chen and He, 2020, Grill et al., 2020]. Interestingly, we observe that the teacher output centering avoids collapse without predictor nor batch

	Method	Momentum	Operation	Top-1
1	DINO	✓	`Centering`	76.1
2	–	✓	`Softmax(batch)`	75.8
3	–	✓	`Sinkhorn-Knopp`	76.0
4	–		`Centering`	0.1
5	–		`Softmax(batch)`	72.2
6	SwAV		`Sinkhorn-Knopp`	71.8

Table 5.10 – **Relation to SwAV.** We vary the operation on the teacher output between centering, a softmax applied over the batch dimension and the Sinkhorn-Knopp algorithm. We also ablate the Momentum encoder by replacing it with a hard copy of the student with a stop-gradient as in SwAV. Models are run for 300 epochs with ViT-S/16. We report top-1 accuracy on ImageNet linear evaluation.

normalizations in BYOL (7, 9), though with a significant performance drop which can likely be explained by the fact that our centering operator is designed to work in combination with sharpening. Finally, we observe that multi-crop works particularly well with DINO and MoCo-v2, removing it hurts performance by $2-4\%$ (1 versus 4 and, 5 versus 6). Adding multi-crop to BYOL does not work out-of-the-box (7, 10) and further adaptation may be required.

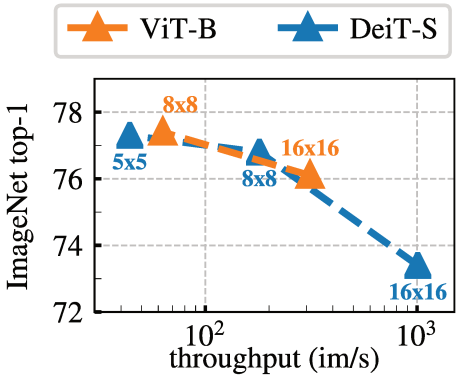


Figure 5.6 – **Effect of Patch Size.** k-NN evaluation as a function of the throughput for different input patch sizes with ViT-B and ViT-S. Models are trained for 300 epochs.

Importance of the patch size. In Figure 5.6, we compare the k-NN classification performance of ViT-S models trained with different patch sizes, 16×16, 8×8 and 5×5. We also compare to ViT-B with 16×16 and 8×8 patches. All the models are trained for 300

	Method	Loss	multi-crop	Center.	BN	Pred.	Top-1
1	DINO	CE	✓	✓			76.1
2	–	MSE	✓	✓			62.4
3	–	CE	✓	✓		✓	75.6
4	–	CE		✓			72.5
5	MoCov2	INCE			✓		71.4
6		INCE	✓		✓		73.4
7	BYOL	MSE			✓	✓	71.4
8	–	MSE			✓		0.1
9	–	MSE		✓			52.6
10	–	MSE	✓		✓	✓	64.8

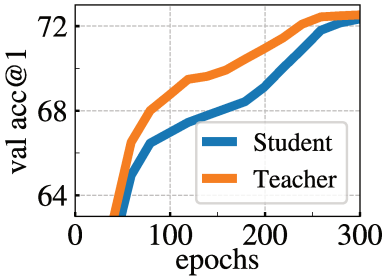
Table 5.11 – **Relation to MoCo-v2 and BYOL.** We ablate the components that differ between DINO, MoCo-v2 and BYOL: the loss function (cross-entropy, CE, versus InfoNCE, INCE, versus mean-square error, MSE), the multi-crop training, the centering operator, the batch normalization in the projection heads and the student predictor. Models are run for 300 epochs with ViT-S/16. We report top-1 accuracy on ImageNet linear evaluation.

epochs. We observe that the performance greatly improves as we decrease the size of the patch. It is interesting to see that performance can be greatly improved without adding additional parameters. However, the performance gain from using smaller patches comes at the expense of throughput: when using 5×5 patches, the throughput falls to 44 im/s, *vs* 180 im/s for 8×8 patches.

5.6.2 Impact of the choice of teacher network

In this ablation, we experiment with different teacher network to understand its role in DINO. We compare models trained for 300 epochs using the k-NN protocol.

Building different teachers from the student. In Figure 5.7 (right), we compare different strategies to build the teacher from previous instances of the student besides the momentum teacher. First we consider using the student network from a previous epoch as a teacher. This strategy has been used in a memory bank [Wu et al., 2018] or as a form of clustering hard-distillation [Asano et al., 2020, Caron et al., 2018, Chen et al., 2020d]. Second, we consider using the student network from the previous iteration, as well as a copy of the student for the teacher. In our setting, using a teacher based on a recent version of the



Teacher	Top-1
Student copy	0.1
Previous iter	0.1
Previous epoch	66.6
Momentum	72.8

Figure 5.7 – **Analyses on the choice of teacher network.** Top-1 accuracy on ImageNet validation with k-NN classifier. **(left)** Comparison between the performance of the momentum teacher and the student during training. **(right)** Comparison between different types of teacher network. The momentum encoder leads to the best performance but is not the only viable option.

student does not converge. This setting requires more normalizations to work. Interestingly, we observe that using a teacher from the previous epoch does not collapse, providing performance in the k-NN evaluation competitive with existing frameworks such as MoCo-v2 or BYOL. While using a momentum encoder clearly provides superior performance to this naive teacher, this finding suggests that there is a space to investigate alternatives for the teacher.

Analyzing the training dynamic. To further understand the reasons why a momentum teacher works well in our framework, we study its dynamic during the training of a ViT in the left panel of Figure 5.7. A key observation is that this teacher constantly outperforms the student during the training, and we have observed the same behavior when training with a ResNet-50. This behavior has not been observed by other frameworks also using momentum [Grill et al., 2020, He et al., 2020], nor when the teacher is built from the previous epoch. We propose to interpret the momentum teacher in DINO as a form of Polyak-Ruppert averaging [Polyak and Juditsky, 1992, Ruppert, 1988] with an exponentially decay. Polyak-Ruppert averaging is often used to simulate model ensembling to improve the performance of a network at the end of the training [Jean et al., 2014]. Our method can be interpreted as applying Polyak-Ruppert averaging during the training to constantly build a model ensembling that has superior performances. This model ensembling then guides the training of the student network [Tarvainen and Valpola, 2017].

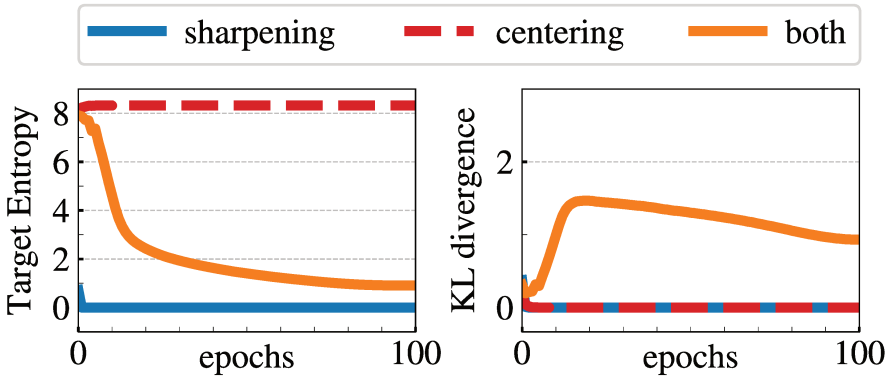


Figure 5.8 – **Collapse study. (left)**: evolution of the teacher's target entropy along training epochs; **(right)**: evolution of KL divergence between teacher and student outputs.

5.6.3 Avoiding collapse

We study the complementarity role of centering and target sharpening to avoid collapse. There are two forms of collapse: regardless of the input, the model output is uniform along all the dimensions or dominated by one dimension. The centering avoids the collapse induced by a dominant dimension, but encourages an uniform output. Sharpening induces the opposite effect. We show this complementarity by decomposing the cross-entropy H into an entropy h and the Kullback-Leibler divergence ("KL") D_{KL}:

$$H(P_t, P_s) = h(P_t) + D_{KL}(P_t|P_s). \tag{5.5}$$

A KL equal to zero indicates a constant output, and hence a collapse. In Figure 5.8, we plot the entropy and KL during training with and without centering and sharpening. If one operation is missing, the KL converges to zero, indicating a collapse. However, the entropy h converges to different values: 0 with no centering and $-\log(1/K)$ with no sharpening, indicating that both operations induce different form of collapse. Applying both operations balances these effects as shown in the following experiments on the sharpening parameter τ_t.

Sharpening. We enforce sharp targets by tuning the teacher softmax temperature parameter τ_t. In Table 5.12, we observe that a temperature lower than 0.06 is required to avoid collapse. When the temperature is higher than 0.06, the training loss consistently converges to $ln(K)$. However, we have observed that using higher temperature than 0.06 does not collapse if we start the training from a smaller value and increase it during the first epochs. In practice, we use a linear warm-up for τ_t from 0.04 to 0.07 during the first 30 epochs

τ_t	0	0.02	0.04	0.06	0.08	$0.04 \rightarrow 0.07$
k-NN top-1	43.9	66.7	69.6	68.7	0.1	69.7

Table 5.12 – **Impact of teacher target sharpening.** Sharpening plays a crucial role in preventing collapse. Hence, DINO is particularly sensitive to collapse at the beginning of training. Our experiments seem to suggest to seek for the maximum temperature that does not collapse.

of training. Finally, note that $\tau \rightarrow 0$ (extreme sharpening) correspond to the `argmax` operation and leads to one-hot hard distributions.

m	0	0.9	0.99	0.999
k-NN top-1	69.1	69.7	69.4	0.1

Table 5.13 – **Impact of online centering.** We study the impact of the smoothing parameters in the update rule for the center c used in the output of the teacher network (see Eq. (5.4)). The convergence is robust to a wide range of smoothing, and the model only collapses when the update is too slow, i.e., $m = 0.999$.

5.6.4 Compute requirements

In Table 5.14, we detail the time and GPU memory requirements when running ViT-S/16 DINO models on two 8-GPU machines. We report results with several variants of multi-crop training, each having a different level of compute requirement. We observe in Table 5.14 that using multi-crop improves the accuracy / running-time trade-off for DINO runs. For example, the performance is 72.5% after 46 hours of training without multi-crop (i.e. 2×224^2) while DINO in $2\times224^2 + 10\times96^2$ crop setting reaches 74.6% in 24 hours only. This is an improvement of $+2\%$ while requiring $2\times$ less time, though the memory usage is higher ($15.4G$ versus $9.3G$). We observe that the performance boost brought with multi-crop cannot be caught up by more training in the 2×224^2 setting, which shows the value of the "local-to-global" augmentation. Finally, the gain from adding more views diminishes (+.2% form $6\times$ to $10\times$ 96^2 crops) for longer trainings.

Overall, training DINO with Vision Transformers achieves 76.1 top-1 accuracy using two 8-GPU servers for 3 days. This result outperforms state-of-the-art self-supervised systems based on convolutional networks of comparable sizes with a significant reduction

	100 epochs		300 epochs		
multi-crop	top-1	time	top-1	time	mem.
2×224^2	67.8	15.3h	72.5	45.9h	9.3G
$2\times224^2 + 2\times96^2$	71.5	17.0h	74.5	51.0h	10.5G
$2\times224^2 + 6\times96^2$	73.8	20.3h	75.9	60.9h	12.9G
$2\times224^2 + 10\times96^2$	74.6	24.2h	76.1	72.6h	15.4G

Table 5.14 – **Time and memory requirements.** We show total running time and peak memory per GPU ("mem.") when running ViT-S/16 DINO models on two 8-GPU machines. We report top-1 ImageNet val acc with linear evaluation for several variants of multi-crop, each having a different level of compute requirement.

of computational requirements [Caron et al., 2020, Grill et al., 2020]. Our code is available to train self-supervised ViT on a limited number of GPUs.

5.6.5 Training with small batches

bs	128	256	512	1024
top-1	57.9	59.1	59.6	59.9

Table 5.15 – **Effect of batch sizes.** Top-1 with k-NN for models trained for 100 epochs without multi-crop.

In Table 5.15, we study the impact of the batch size on the features obtained with DINO. We also study the impact of the smooth parameter m used in the centering update rule of Eq. (5.4) in Table 5.13. We scale the learning rate linearly with the batch size [Goyal et al., 2017]: $lr = 0.0005 * \text{batchsize}/256$. Table 5.15 confirms that we can train models to high performance with small batches. Results with the smaller batch sizes ($bs = 128$) are slightly below our default training setup of $bs = 1024$, and would certainly require to re-tune hyperparameters like the momentum rates for example. Note that the experiment with batch size of 128 runs on only 1 GPU. We have explored training a model with a batch size of 8, reaching 35.2% after 50 epochs, showing the potential for training large models that barely fit an image per GPU.

5.6.6 Ablation study on the projection head

Similarly to other self-supervised frameworks, we observe that using a projection head [Bachman et al., 2019] improves greatly the accuracy of our method. The projection

head starts with a n-layer multi-layer perceptron (MLP). The hidden layers are 2048d and are with Gaussian error linear units (GELU) activations. The last layer of the MLP is without GELU. Then we apply a ℓ_2 normalization and a weight normalized fully connected layer [Chen and He, 2020, Salimans and Kingma, 2016] with K dimensions. This design is inspired from the projection head with a "prototype layer" used in SwAV [Caron et al., 2020]. We do not apply batch normalizations. In this set of experiments, we study the effect of the l2-normalization bottleneck, the number of linear layers in the projection head, the output dimension K, the choice of activation unit and the impact of adding batch normalizations.

BN-free system. Unlike standard convnets, ViT architectures do not use batch normalizations (BN) by default. Therefore, when applying DINO to ViT we do not use any BN

ViT-S, 100 epochs	heads w/o BN	heads w/ BN
k-NN top-1	69.7	68.6

also in the projection heads. In this table we evaluate the impact of adding BN in the heads. We observe that adding BN in the projection heads has little impact, showing that BN is not important in our framework. *Overall, when applying DINO to ViT, we do not use any BN anywhere, making the system entirely BN-free.* This is a great advantage of DINO + ViT to work at state-of-the-art performance without requiring any BN. Indeed, training with BN typically slows down trainings considerably, especially when these BN modules need to be synchronized across processes [Caron et al., 2019, 2020, Grill et al., 2020, He et al., 2020].

L2-normalization bottleneck in projection head. We illustrate the design of the projection head with or without l2-normalization bottleneck in Figure 5.9. We evaluate the

# proj. head linear layers	1	2	3	4
w/ l2-norm bottleneck	–	62.2	68.0	69.3
w/o l2-norm bottleneck	61.6	62.9	0.1	0.1

accuracy of DINO models trained with or without l2-normalization bottleneck and we vary the number of linear layers in the projection head. With l2 bottleneck, the total number of linear layers is $n + 1$ (n from the MLP and 1 from the weight normalized layer) while without bottleneck the total number of linear layers is n in the head. In this table, we report ImageNet top-1 k-NN evaluation accuracy after 100 epochs pre-training with ViT-S/16. The output dimensionality K is set to 4096 in this experiment. We observe that DINO training fails without the l2-normalization bottleneck when increasing the depth of the projection head. L2-normalization bottleneck stabilizes the training of DINO with deep projection head. We observe that increasing the depth of the projection head improves

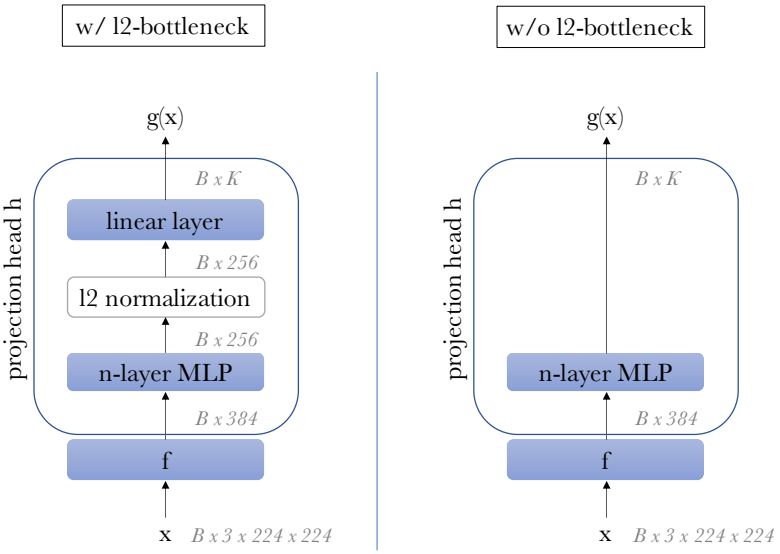


Figure 5.9 – **Projection head design w/ or w/o l2-norm bottleneck.**

accuracy. Our default is to use a total of 4 linear layers: 3 are in the MLP and one is after the l2 bottleneck.

Output dimension. In this table, we evaluate the effect of varying the output dimensionality K. We observe that a large output dimensionality improves the performance. We note

K	1024	4096	16384	65536	262144
k-NN top-1	67.8	69.3	69.2	69.7	69.1

that the use of l2-normalization bottleneck permits to use a large output dimension with a moderate increase in the total number of parameters. Our default is to use K equals to 65536 and $d = 256$ for the bottleneck.

GELU activations. By default, the activations used in ViT are Gaussian error linear units (GELU). Therefore, for consistency within the architecture, we choose to use GELU also

ViT-S, 100 epochs	heads w/ GELU	heads w/ ReLU
k-NN top-1	69.7	68.9

in the projection head. We evaluate the effect of using ReLU instead of GELU in this table and observe that changing the activation unit to ReLU has relatively little impact. It is likely that re-tuning the hyperparameters would allow to recover the 0.8% performance gap between our default and the heads with ReLU.

crops	2×224^2		$2 \times 224^2 + 6 \times 96^2$	
eval	k-NN	linear	k-NN	linear
BYOL	66.6	71.4	59.8	64.8
SwAV	60.5	68.5	64.7	71.8
MoCo-v2	62.0	71.6	65.4	73.4
DINO	**67.9**	**72.5**	**72.7**	**75.9**

Table 5.16 – **Multi-crop applied to different self-supervised frameworks.** We report top-1 accuracy on ImageNet for different methods using two different configurations of multi-crop: (i) 2×224^2 refers to 2 global crops and no local crops, (ii) $2 \times 224^2 + 6 \times 96^2$ refers to 2 global crops and 6 local crops of resolution 96 by 96 pixels.

5.6.7 Ablation study on multi-crop

In this section, we study a core component of DINO: multi-crop training which we introduced in previous Chapter 4 (see Section 4.2.4).

Range of scales in multi-crop. We use the `RandomResizedCrop` in PyTorch for generating the different views. We sample two global views with scale range $(s, 1)$ before resizing them to 224^2 and 6 local views with scale sampled in the range $(0.05, s)$ resized to 96^2 pixels. Note that we arbitrarily choose to have non-overlapping scaling range for the global and local views following the original design of SwAV. However, the ranges could definitely be overlapping and experimenting with finer hyperparameters search could lead to a more optimal setting. In this table, we vary the parameter s that controls the range of scales used in multi-crop and find the optimum to be around 0.3 in our experiments. We note that this is higher than the parameter used in SwAV which was of 0.14.

$(0.05, s)$, $(s, 1)$, s:	0.08	0.16	0.24	0.32	0.48
k-NN top-1	65.6	68.0	69.7	69.8	69.5

Multi-crop in different self-supervised frameworks. Finally, we compare the impact of multi-crop with ViT trained with different recent self-supervised learning frameworks, namely MoCo-v2 [Chen et al., 2020e], BYOL [Grill et al., 2020] and SwAV [Caron et al., 2020]. For fair comparisons, all models are pretrained either with two 224^2 crops or with multi-crop [Caron et al., 2020] training, i.e. two 224^2 crops and six 96^2 crops for each image. We report k-NN and linear probing evaluations after 300 epochs of training. In Table 5.16, we observe that multi-crop does not benefit all frameworks equally, which has

been ignored in benchmarks considering only the two crops setting [Chen and He, 2020]. The effectiveness of multi-crop depends on the considered framework, which positions multi-crop as a core component of a model and not a simple "add-ons" that will boost any framework the same way. Without multi-crop, DINO has better accuracy than other frameworks, though by a moderate margin (1%). Remarkably, DINO benefits the most from multi-crop training ($+3.4\%$ in linear eval). Interestingly, we also observe that the ranking of the frameworks depends on the evaluation protocol considered.

Chapter 6

Conclusion

In conclusion, we summarize the contributions presented in this manuscript before giving an overview of some of the open problems and challenges in the field of self-supervised learning.

6.1 Summary of Contributions

Deep clustering. In Chapter 3, we have introduced deep clustering and presented an extension to this framework: the DeeperCluster method. DeeperCluster iterates between clustering with k-means the features produced by the network and updating its weights by predicting the cluster assignments as pseudo-labels. If trained on large uncurated datasets like YFCC100M for example, it achieves high performance on several standard transfer tasks. Indeed, DeeperCluster outperforms the state of the art at the time of the publication and is getting very close to supervised methods, while not surpassing them. We have identified at the end of Chapter 3 the main limitations of deep clustering (see Section 3.4) that we would need to overcome with the goal of eventually outperforming supervised pre-training. These limitations include inefficiency, limited scalability, under-explored dependence in the data transformations and use of empirical tricks to avoid feature collapse.

SwAV. Subsequently, in Chapter 4, we have proposed a new and improved model for self-supervised learning, SwAV, that overcomes the limitations of deep clustering. As a matter of fact, SwAV also overcomes the problems posed by the popular contrastive implementations described in Section 2.3.2. SwAV is an implementation of the view-invariant feature learning paradigm (introduced in Section 2.3) that, unlike contrastive methods, does not rely on direct feature comparison. This makes it more practical as it does not

need "negative pairs", large mini-batches or memory banks. Within our SwAV framework, we also introduce multi-crop training which encourages the emergence of holistic representations through a local-to-global matching. We show with extensive experiments that multi-crop allows important performance gains. At the time of its publication in Caron et al. [2020], SwAV is state-of-the-art on ImageNet among the self-supervised learning methods. Perhaps more importantly, we show that SwAV features transfer very well to several downstream tasks on different datasets, outperforming supervised features on all the considered transfer benchmarks. Finally, we have assessed SwAV in an uncontrolled setting by training on uncurated "in the wild" images. We have shown in this context the importance of scaling both dataset size and model capacity to make up for the lack of curation. Our final large-scale SwAV model, trained with a 1.3B parameters architecture on 1B random images from Instagram, achieves 84.2% top-1 accuracy on ImageNet. This confirms that self-supervised learning can work in a real world setting to some extent.

Self-supervised transformers. Finally, in Chapter 5, we have shown the potential of self-supervised pre-trained Vision Transformers, achieving performance that are comparable with the best convnets specifically designed for this setting while using much less parameters. We have also seen emerged two properties that can be leveraged in future applications: the quality of the features in k-NN classification has a potential for image retrieval, matching or similarity search. Second, the presence of information about the scene layout in the features can also benefit weakly supervised or fully unsupervised image segmentation. However, the main result of Chapter 5 is to show that we have evidence that self-supervised learning could be the key to developing a BERT-like model based on transformer architectures, which would allow to push the limits of visual features.

6.2 Open Problems in Self-Supervised Learning

The last decade has seen tremendous progress in self-supervised learning. At this point in time and at the end of this manuscript, one is entitled to ask if this is a solved field. Our (subjective) answer to that is a clear *no* because much remains to be done to tackle the open challenges raised by self-supervised learning. Also, we remark that most of the research so far has been conducted in very controlled settings, still far from real-world applications. In this final section, we give an overview of some of the open questions and remaining challenges in self-supervised learning.

6.2.1 Self-supervised versus supervised pre-training

First of all, in this manuscript (see Sections 4.3.2 and 5.5.3 for example), consistently with other works in the self-supervised community [He et al., 2020, Misra and Maaten, 2020], we have seen that self-supervised pre-training leads to better performance on downstream tasks compared to supervised methods. However, these results might not imply that we can replace supervised pre-training with self-supervised one just yet. This is because at the moment, the current self-supervised systems are much slower than the supervised ones to learn representations of equal quality. Self-supervised models require much more compute and epochs to reach convergence compared to supervised training on ImageNet. The performance improvements observed with self-supervised learning is probably not significant enough to justify the need for such an increase in the compute requirements. At this stage, if there are some labels available then it is likely better to use them. Hopefully, future improvements will make self-supervised pre-training more efficient and, as a result, more accessible to the community.

Actually, improving pre-training efficiency has been the focus of a collaboration conducted during the time of the PhD program but not presented in this manuscript (see Assran et al. [2021]). In this work, we propose a semi-supervised approach inspired by SwAV [Caron et al., 2020], that achieves state-of-the-art performance on ImageNet with either 10% or 1% of the training instances labeled while using only 200 epochs of training, which is $4\times$ less than previous works. These results suggest that, at least in the short term, the best way to go might not be purely self-supervised but to better leverage small amount of annotated data.

6.2.2 Towards real-world self-supervised learning

In this section, we mention several directions to make self-supervised learning work in more realistic scenario.

Uncurated data. A major limitation of the current self-supervised systems is that they are typically trained in the highly controlled setting of using ImageNet without labels. However, as mentioned at length in this manuscript, this does not reflect a real-world situation of *unsupervised* learning since removing the labels only removes part of the human supervision. We have shown in Sections 3.3 and 4.5 some trainings on raw, "never-before-labeled" images but the conclusions of our study is arguably slightly underwhelming. Indeed, our results show that it is possible to learn good representations from uncurated data but they also exhibit a strong diminishing return with the dataset scale. It is possible

that we would need to completely re-think our current self-supervised frameworks to make systems work better in uncontrolled settings with long tail distributions.

Actually, the current most successful methods are all declination of the same view-invariant paradigm, which might have led recently to certain performance saturation in the benchmarks. Ways to move forward might be to seek progress in orthogonal directions (this is what we propose in Chapter 5 by considering self-supervised features with Vision Transformers) or to explore different and original self-supervised paradigms.

Unsupervised recognition tasks. The well-known recipe used in this manuscript to solve visual tasks is twofold: first a network is pre-trained without using any annotations before being transferred to a downstream task. While the first stage does not rely on annotations, the second one often still requires a lot of labels, especially when transferring to dense recognition tasks. Indeed, even if networks trained with this paradigm have shown good performance on dense-level recognition problems such as object detection, they need to be heavily adapted to these tasks, which is not practical in real applications. In contrast, we have seen in Chapter 5 that it is possible to obtain excellent performance for ImageNet classification task without any finetuning (see Section 5.4), which is akin to a *zero-shot* setting. This probably can be explained by the fact that methods developed in this manuscript work with *holistic* image representations, which are especially suited for global tasks like classification. This raises the question if we can obtain similar zero-shot results on dense tasks. Our preliminary results in Chapter 5 suggest that unsupervised dense segmentation, matching or localization might be possible, which can represent promising future directions.

Multi-modal learning. The models developed in this manuscript operate on unlabeled images only. However most real applications are *multi-modal* in the sense that they deal with data coming from different modalities. For example a video uploaded on a social media platform will usually come with one of several additional data stream like a soundtrack, a caption, some hashtags, geolocalization coordinates, emojis, etc. This motivates for developing better multi-modal systems capable of ingesting different media of data. Intuitively, these plural modalities should complement one another and could allow to learn richer and more generic representations.

6.2.3 Is self-supervised learning fully unsupervised?

Finally, we can question if the research in self-supervised learning is really *fully unsupervised*. In some aspects, we can argue that self-supervised learning still relies on some human labor and is not totally label-agnostic. For example, we still use a significant

amount of labels during hyperparameter searches and model explorations to develop self-supervised systems.

Handcrafted data transformations. In most modern and successful self-supervised approaches, the data augmentation scheme strongly drives the performance of the system. Indeed, the details of view generation are crucial and require a careful design. For example, Chen et al. [2020b] have conducted extensive ablations on the data augmentation pipeline which has boosted considerably the performance on ImageNet pre-training. These optimized pipelines are tuned on a certain kind of data and downstream tasks and are not likely to transfer to other domains (medical images for example). In addition, they are very *handcrafted*. We can criticize that the labor needed to annotate the data is being replaced by a different kind of labor which consists in manually tuning the data transformations to the considered task and application. It would be interesting to automate the search for an optimal data augmentation pipeline, in a similar manner as Auto-augment [Cubuk et al., 2018] in supervised learning.

Model selection. In standard supervised learning, model selection is usually performed by looking at the loss value on a held-out validation set. In self-supervised learning though, the loss is not necessarily aligned with the final objective which is to learn representations useful for downstream tasks. Thus, the model is typically validated by looking at the transfer performance of the network to supervised downstream tasks. As a result, the current self-supervised methods still use some supervision in a sense, and might actually be over-fitted to the supervised task used for validation. A future and challenging research question could be to find ways to select self-supervised models without using supervision.

Bibliography

P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *ICCV*, 2015.

A. Ahmed, K. Yu, W. Xu, Y. Gong, and E. Xing. Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks. In *ECCV*, 2008.

J.-B. Alayrac, A. Recasens, R. Schneider, R. Arandjelovic, J. Ramapuram, J. De Fauw, L. Smaira, S. Dieleman, and A. Zisserman. Self-supervised multimodal versatile networks. In *NeurIPS*, 2020.

R. Anil, G. Pereyra, A. Passos, R. Ormandi, G. E. Dahl, and G. E. Hinton. Large scale distributed neural network training through online distillation. *arXiv preprint arXiv:1804.03235*, 2018.

Y. M. Asano, C. Rupprecht, and A. Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *ICLR*, 2020.

M. Assran, M. Caron, I. Misra, P. Bojanowski, A. Joulin, N. Ballas, and M. Rabbat. Semi-supervised learning of visual features by non-parametrically predicting view assignments with support samples. In *ICCV*, 2021.

F. R. Bach and Z. Harchaoui. Diffrac: a discriminative and flexible framework for clustering. In *NeurIPS*, 2008.

P. Bachman, R. D. Hjelm, and W. Buchwalter. Learning representations by maximizing mutual information across views. *arXiv preprint arXiv:1906.00910*, 2019.

A. Baevski, H. Zhou, A. Mohamed, and M. Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *arXiv preprint arXiv:2006.11477*, 2020.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *NeurIPS*, 2007.

M. Berman, H. Jégou, V. Andrea, I. Kokkinos, and M. Douze. MultiGrain: a unified image embedding for classes and instances. *arXiv preprint arXiv:1902.05509*, 2019.

P. Bojanowski and A. Joulin. Unsupervised learning by predicting noise. In *ICML*, 2017.

L. Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*. 2012.

A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai. Class-based n-gram models of natural language. *Computational linguistics*, 1992.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil. Model compression. In *SIGKDD*, 2006.

N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

M. Caron. Unsupervised representation learning with clustering in deep convolutional networks, 2018.

M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018.

M. Caron, P. Bojanowski, J. Mairal, and A. Joulin. Unsupervised pre-training of image features on non-curated data. In *ICCV*, 2019.

M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020.

M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik. Human pose estimation with iterative error feedback. In *CVPR*, 2016.

L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.

M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever. Generative pretraining from pixels. In *ICML*, 2020a.

M. X. Chen, O. Firat, A. Bapna, M. Johnson, W. Macherey, G. Foster, L. Jones, N. Parmar, M. Schuster, Z. Chen, et al. The best of both worlds: Combining recent advances in neural machine translation. *arXiv preprint arXiv:1804.09849*, 2018.

T. Chen, X. Zhai, M. Ritter, M. Lucic, and N. Houlsby. Self-supervised gans via auxiliary rotation loss. In *CVPR*, 2019.

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020b.

T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. Hinton. Big self-supervised models are strong semi-supervised learners. In *NeurIPS*, 2020c.

W. Chen, S. Pu, D. Xie, S. Yang, Y. Guo, and L. Lin. Unsupervised image classification for deep representation learning. *arXiv preprint arXiv:2006.11480*, 2020d.

X. Chen and A. Gupta. Webly supervised learning of convolutional networks. In *ICCV*, 2015.

X. Chen and K. He. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, 2020.

X. Chen, H. Fan, R. Girshick, and K. He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020e.

R. Child, S. Gray, A. Radford, and I. Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

A. Coates and A. Y. Ng. Learning feature representations with k-means. In *Neural networks: Tricks of the trade*. 2012.

J.-B. Cordonnier, A. Loukas, and M. Jaggi. On the relationship between self-attention and convolutional layers. In *ICLR*, 2020.

P. Courtiol, E. W. Tramel, M. Sanselme, and G. Wainrib. Classification and disease localization in histopathology using only global labels: A weakly-supervised approach. *arXiv preprint arXiv:1802.02212*, 2018.

G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, 2004.

E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NeurIPS*, 2013.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

C. Doersch and A. Zisserman. Multi-task self-supervised visual learning. In *ICCV*, 2017.

C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015.

J. Donahue and K. Simonyan. Large scale adversarial representation learning. *arXiv preprint arXiv:1907.02544*, 2019.

J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *NeurIPS*, 2014.

A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *TPAMI*, 2016.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

M. Douze, H. Jégou, H. Sandhawalia, L. Amsaleg, and C. Schmid. Evaluation of gist descriptors for web-scale image search. In *CIVR*, 2009.

M. Douze, H. Jégou, and J. Johnson. An evaluation of large-scale methods for image instance and class discovery. *arXiv preprint arXiv:1708.02898*, 2017.

V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

A. El-Nouby, H. Touvron, M. Caron, P. Bojanowski, M. Douze, A. Joulin, I. Laptev, N. Neverova, G. Synnaeve, J. Verbeek, et al. Xcit: Cross-covariance image transformers. *arXiv preprint arXiv:2106.09681*, 2021.

D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 2009.

A. Ermolov, A. Siarohin, E. Sangineto, and N. Sebe. Whitening for self-supervised representation learning. *arXiv preprint arXiv:2007.06346*, 2020.

M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

Z. Fang, J. Wang, L. Wang, L. Zhang, Y. Yang, and Z. Liu. Seed: Self-supervised distillation for visual representation. 2021.

Z. Feng, C. Xu, and D. Tao. Self-supervised representation learning by rotation feature decoupling. In *CVPR*, 2019.

J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*. 2001.

S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018.

S. Gidaris, A. Bursuc, N. Komodakis, P. Pérez, and M. Cord. Learning representations by predicting bags of visual words. In *CVPR*, 2020a.

S. Gidaris, A. Bursuc, G. Puy, N. Komodakis, M. Cord, and P. Pérez. Online bag-of-visual-words generation for unsupervised representation learning. *arXiv preprint arXiv:2012.11552*, 2020b.

L. Gomez, Y. Patel, M. Rusiñol, D. Karatzas, and C. Jawahar. Self-supervised learning of visual features through embedding images into text topic spaces. In *CVPR*, 2017.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

I. J. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. 2016.

P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

P. Goyal, D. Mahajan, A. Gupta, and I. Misra. Scaling and benchmarking self-supervised visual representation learning. In *ICCV*, 2019.

P. Goyal, M. Caron, B. Lefaudeux, M. Xu, P. Wang, V. Pai, M. Singh, V. Liptchinsky, I. Misra, A. Joulin, et al. Self-supervised pretraining of visual features in the wild. *arXiv preprint arXiv:2103.01988*, 2021.

J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *NeurIPS*, 2020.

S. Gur, A. Ali, and L. Wolf. Visualization of supervised and self-supervised neural networks via attribution guided factorization. *arXiv preprint arXiv:2012.02166*, 2020.

M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *International Conference on Artificial Intelligence and Statistics*, 2010.

R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006.

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *ICCV*, 2017.

K. He, R. Girshick, and P. Dollár. Rethinking imagenet pre-training. In *ICCV*, 2019.

K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

O. Henaff. Data-efficient image recognition with contrastive predictive coding. In *ICML*, 2020.

O. J. Hénaff, A. Srinivas, J. De Fauw, A. Razavi, C. Doersch, S. Eslami, and A. v. d. Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019.

G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 2006.

R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.

F. J. Huang, Y.-L. Boureau, Y. LeCun, et al. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *CVPR*, 2007.

G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

A. Jabri, A. Owens, and A. A. Efros. Space-time correspondence as a contrastive random walk. In *NeurIPS*, 2020.

D. Jayaraman and K. Grauman. Learning image representations tied to ego-motion. In *ICCV*, 2015.

S. Jean, K. Cho, R. Memisevic, and Y. Bengio. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*, 2014.

J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

A. Joulin and F. Bach. A convex relaxation for weakly supervised classifiers. *arXiv preprint arXiv:1206.6413*, 2012.

A. Joulin, F. Bach, and J. Ponce. Discriminative clustering for image co-segmentation. In *CVPR*, 2010.

A. Joulin, L. Van Der Maaten, A. Jabri, and N. Vasilache. Learning visual features from large weakly supervised data. In *ECCV*, 2016.

Y. Kalantidis, M. B. Sariyildiz, N. Pion, P. Weinzaepfel, and D. Larlus. Hard negative mixing for contrastive learning. In *NeurIPS*, 2020.

D. Kim, D. Cho, D. Yoo, and I. S. Kweon. Learning image representations by completing damaged jigsaw puzzles. In *WACV*, 2018.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*, 2017.

A. Kolesnikov, X. Zhai, and L. Beyer. Revisiting self-supervised visual representation learning. In *CVPR*, 2019.

P. Krähenbühl, C. Doersch, J. Donahue, and T. Darrell. Data-dependent initializations of convolutional neural networks. *arXiv preprint arXiv:1511.06856*, 2015.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

A. Kuznetsova, M. H. M. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, T. Duerig, and V. Ferrari. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV*, 2020.

Z. Lai, E. Lu, and W. Xie. Mast: A memory-augmented self-supervised tracker. In *CVPR*, 2020.

G. Larsson, M. Maire, and G. Shakhnarovich. Learning representations for automatic colorization. In *ECCV*, 2016.

G. Larsson, M. Maire, and G. Shakhnarovich. Colorization as a proxy task for visual understanding. In *CVPR*, 2017.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. 1998.

D.-H. Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, 2013.

H.-Y. Lee, J.-B. Huang, M. Singh, and M.-H. Yang. Unsupervised representation learning by sorting sequences. In *CVPR*, 2017.

D. Li, W.-C. Hung, J.-B. Huang, S. Wang, N. Ahuja, and M.-H. Yang. Unsupervised visual representation learning by graph-based consistent constraints. In *ECCV*, 2016.

J. Li, P. Zhou, C. Xiong, R. Socher, and S. C. Hoi. Prototypical contrastive learning of unsupervised representations. *arXiv preprint arXiv:2005.04966*, 2020.

R. Liao, A. Schwing, R. Zemel, and R. Urtasun. Learning deep parsimonious representations. In *NeurIPS*, 2016.

F. Lin and W. W. Cohen. Power iteration clustering. In *ICML*, 2010.

T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

R. Linsker. Towards an organizing principle for a layered perceptual network. In *NeurIPS*, 1988.

I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

I. Loshchilov and F. Hutter. Fixing weight decay regularization in adam. 2018.

D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

T. Lucas. *Deep generative models : over-generalisation and mode-dropping*. Theses, 2020.

D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten. Exploring the limits of weakly supervised pretraining. In *ECCV*, 2018.

J. Mairal. Cyanure: An open-source toolbox for empirical risk minimization for python, c++, and soon more. *arXiv preprint arXiv:1912.08165*, 2019.

T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011.

J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. *Artificial Neural Networks and Machine Learning*, 2011.

P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

I. Misra and L. v. d. Maaten. Self-supervised learning of pretext-invariant representations. In *CVPR*, 2020.

I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *ECCV*, 2016a.

I. Misra, C. L. Zitnick, M. Mitchell, and R. Girshick. Seeing through the human reporting bias: Visual classifiers from noisy human-centric labels. In *CVPR*, 2016b.

T. N. Mundhenk, D. Ho, and B. Y. Chen. Improvements to context based self-supervised learning. In *CVPR*, 2018.

K. Ni, R. Pearce, K. Boakye, B. Van Essen, D. Borth, B. Chen, and E. Wang. Large-scale deep learning on the yfcc100m dataset. *arXiv preprint arXiv:1502.03409*, 2015.

M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, 2008.

M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.

M. Noroozi, H. Pirsiavash, and P. Favaro. Representation learning by learning to count. In *ICCV*, 2017.

M. Noroozi, A. Vinjimoor, P. Favaro, and H. Pirsiavash. Boosting self-supervised learning via knowledge transfer. In *CVPR*, 2018.

S. W. Oh, J.-Y. Lee, N. Xu, and S. J. Kim. Video object segmentation using space-time memory networks. In *ICCV*, 2019.

A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional image generation with pixelcnn decoders. *arXiv preprint arXiv:1606.05328*, 2016.

A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

A. Owens, J. Wu, J. H. McDermott, W. T. Freeman, and A. Torralba. Ambient sound provides supervision for visual learning. In *ECCV*, 2016.

N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran. Image transformer. In *ICML*, 2018.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan. Learning features by watching objects move. In *CVPR*, 2017.

M. Paulin, M. Douze, Z. Harchaoui, J. Mairal, F. Perronin, and C. Schmid. Local convolutional features with unsupervised training for image retrieval. In *ICCV*, 2015.

J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*, 2008.

P. O. Pinheiro, A. Almahairi, R. Y. Benmalek, F. Golemo, and A. Courville. Unsupervised learning of dense visual representations. In *NeurIPS*, 2020.

L. Pinto, D. Gandhi, Y. Han, Y.-L. Park, and A. Gupta. The curious robot: Learning visual representations via physical interactions. In *ECCV*, 2016.

B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 1992.

J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017.

S. Purushwalkam and A. Gupta. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. *arXiv preprint arXiv:2007.13916*, 2020.

G.-J. Qi, L. Zhang, C. W. Chen, and Q. Tian. Avt: Unsupervised learning of transformation equivariant representations by autoencoding variational transformations. In *ICCV*, 2019.

F. Radenović, A. Iscen, G. Tolias, Y. Avrithis, and O. Chum. Revisiting oxford and paris: Large-scale image retrieval benchmarking. In *CVPR*, 2018a.

F. Radenović, G. Tolias, and O. Chum. Fine-tuning cnn image retrieval with no human annotation. *TPAMI*, 2018b.

A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.

I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár. Designing network design spaces. In *CVPR*, 2020.

S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

J. Revaud, J. Almazán, R. S. Rezende, and C. R. d. Souza. Learning with average precision: Training image retrieval with a listwise loss. In *ICCV*, 2019.

P. H. Richemond, J.-B. Grill, F. Altché, C. Tallec, F. Strub, A. Brock, S. Smith, S. De, R. Pascanu, B. Piot, et al. Byol works even without batch statistics. *arXiv preprint arXiv:2010.10241*, 2020.

D. Ruppert. Efficient estimations from a slowly convergent robbins-monro process. Technical report, 1988.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

T. Salimans and D. P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. 2016.

J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. *IJCV*, 2013.

R. Santa Cruz, B. Fernando, A. Cherian, and S. Gould. Deeppermnet: Visual permutation learning. In *CVPR*, 2017.

A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *CVPR workshops*, 2014.

Z. Shen, Z. Liu, J. Qin, L. Huang, K.-T. Cheng, and M. Savvides. S2-bnn: Bridging the gap between self-supervised real and 1-bit neural networks via guided distribution calibration. *arXiv preprint arXiv:2102.08946*, 2021.

J. Shi and J. Malik. Normalized cuts and image segmentation. *TPAMI*, 2000.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

K. Sohn, D. Berthelot, C.-L. Li, Z. Zhang, N. Carlini, E. D. Cubuk, A. Kurakin, H. Zhang, and C. Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *NeurIPS*, 2020.

N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014.

T. Stamey, J. Kabalin, J. McNeal, I. Johnstone, F. Freiha, E. Redwine, and N. Yang. Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate. ii. radical prostatectomy treated patients. *The Journal of urology*, 1989.

P. Stock and M. Cisse. Convnets and imagenet beyond accuracy: Explanations, bias detection, adversarial examples and model criticism. In *ECCV*, 2018.

C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017.

A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv preprint arXiv:1703.01780*, 2017.

B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. Yfcc100m: The new data in multimedia research. *arXiv preprint arXiv:1503.01817*, 2015.

Y. Tian, D. Krishnan, and P. Isola. Contrastive multiview coding. In *ECCV*, 2020a.

Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola. What makes for good views for contrastive learning? In *NeurIPS*, 2020b.

Y. Tian, O. J. Henaff, and A. v. d. Oord. Divide and contrast: Self-supervised learning from uncurated data. *arXiv preprint arXiv:2105.08054*, 2021.

G. Tolias, R. Sicre, and H. Jégou. Particular object retrieval with integral max-pooling of cnn activations. *arXiv preprint arXiv:1511.05879*, 2015.

H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *CVPR*, 1991.

K. Van De Sande, T. Gevers, and C. Snoek. Evaluating color descriptors for object and scene recognition. *TPAMI*, 2010.

G. Van Horn, O. Mac Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie. The inaturalist species classification and detection dataset. In *CVPR*, 2018.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008.

X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015.

X. Wang, K. He, and A. Gupta. Transitive invariance for self-supervised visual representation learning. In *ICCV*, 2017.

X. Wang, A. Jabri, and A. A. Efros. Learning correspondence from the cycle-consistency of time. In *CVPR*, 2019.

P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *ICCV*, 2013.

T. Weyand, I. Kostrikov, and J. Philbin. Planet-photo geolocation with convolutional neural networks. In *ECCV*, 2016.

T. Weyand, A. Araujo, B. Cao, and J. Sim. Google landmarks dataset v2-a large-scale benchmark for instance-level recognition and retrieval. In *CVPR*, 2020.

Z. Wu, Y. Xiong, S. X. Yu, and D. Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 2018.

J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, 2016.

Q. Xie, Z. D. Dai, E. Hovy, M.-T. Luong, and Q. V. Le. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*, 2020a.

Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le. Self-training with noisy student improves imagenet classification. In *CVPR*, 2020b.

S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

Y. Xiong, M. Ren, W. Zeng, and R. Urtasun. Self-supervised representation learning from flow equivariance. *arXiv preprint arXiv:2101.06553*, 2021.

L. Xu, J. Neufeld, B. Larson, and D. Schuurmans. Maximum margin clustering. In *NeurIPS*, 2005.

Q. Xu, T. Likhomanenko, J. Kahn, A. Hannun, G. Synnaeve, and R. Collobert. Iterative pseudo-labeling for speech recognition. *arXiv preprint arXiv:2005.09267*, 2020.

I. Z. Yalniz, H. Jégou, K. Chen, M. Paluri, and D. Mahajan. Billion-scale semi-supervised learning for image classification. *arXiv preprint arXiv:1905.00546*, 2019.

J. Yang, D. Parikh, and D. Batra. Joint unsupervised learning of deep representations and image clusters. In *CVPR*, 2016.

J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

Y. You, I. Gitman, and B. Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *ICML*, 2021.

M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

X. Zhan, J. Xie, Z. Liu, Y.-S. Ong, and C. C. Loy. Online deep clustering for unsupervised representation learning. In *CVPR*, 2020.

L. Zhang, G.-J. Qi, L. Wang, and J. Luo. Aet vs. aed: Unsupervised representation learning by auto-encoding transformations rather than data. *arXiv preprint arXiv:1901.04596*, 2019.

R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *ECCV*, 2016.

R. Zhang, P. Isola, and A. A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, 2017.

H. Zhao, J. Jia, and V. Koltun. Exploring self-attention for image recognition. In *CVPR*, 2020.

B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *NeurIPS*, 2014.

C. Zhuang, A. L. Zhai, and D. Yamins. Local aggregation for unsupervised learning of visual embeddings. In *ICCV*, 2019.