Deep-learning for high dimensional sequential observations : application to continuous gesture recognition

Nicolas Granger

université PARIS-SACLAY

TELECOM SudParis

NNT : 2019SACLL002

Thèse de doctorat

Deep-Learning for High Dimensional Sequential Observations : Application to Continuous Gesture Recognition

Thèse de doctorat de l'Université Paris-Saclay
préparée à Télécom SudParis

École doctorale n°580 Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Robotique

Thèse présentée et soutenue à Évry, le 10/01/2019, par

Nicolas Granger

Composition du Jury :

Alice Caplier Professeur, Grenoble-INP (GIPSA-Lab)	Présidente du jury
Gilles Gasso Professeur, INSA de Rouen (LITIS)	Rapporteur
Fabien Moutarde Professeur, Mines ParisTech (CAOR)	Rapporteur
Hervé Bredin Chargé de recherche, Université d'Orsay (LIMSI)	Examinateur
Laurence Likforman Maître de conférences, HDR, Telecom ParisTech (S^2A)	Examinatrice
Mounîm A. El Yacoubi Directeur d'études, Télécom SudParis (ARMEDIA)	Directeur de thèse
Sylvain Le Corff Maître de conférences, Télécom SudParis (CITI)	Invité
Quoc Cuong Pham Responsable d'équipe, CEA (DIASI)	Invité

Abstract

The original impulse for this thesis came as a motivation to improve the intuitiveness of human–computer interfaces. In particular, machines should try to replicate human's ability to process streams of information continuously in real-time. Indeed, reading, listening to speeches or observing a live scene are all natural activities we perform spontaneously and use extensively to interact or communicate. However, the sub-domain of Machine Learning dedicated to recognition on time series remains barred by numerous challenges: modelling patterns simultaneously over time and within individual observations, dealing with high dimensional inputs from streams of observations, conforming to real-time specifications, etc. Nevertheless, this research field has progressed steadily over the last decades, with a recent renewal fuelled by advances on Neural Network models.

To support our studies on this subject, gesture recognition was selected as the exemplar application. This type of input presents several qualities in our eyes: firstly, gestures intermix static body poses and movements in a complex manner to convey information; secondly, gesture data is encoded under widely different modalities with low and high dimensional representations; finally, the lack of expertise in this field — compared to handwriting or speech recognition — emphasizes better the importance of automatically learning useful factors of variation, which conditions cross-domain re-usability.

The first part of our work examines two state-of-the-art temporal models used in the context of continuous sequence recognition, namely Hybrid Neural Network–Hidden Markov Models (NN-HMM) and Bidirectional Recurrent Neural Networks (BDRNN) with gated units. Instead of trying to improve their performances for a given task, this thesis puts more emphasis on analyzing shortcomings, advantages, similarities or influential properties. To do so, we reimplement the two within a shared test-bed for continuous sequence recognition which is more amenable to a fair comparative work. We propose adjustments to Neural Network training loss functions and the Hybrid NN-HMM expressions to accommodate for highly imbalanced data classes. Although most of recent contributions tend to prefer the Recurrent Neural Networks on the basis of superior performances, we demonstrate that both models can in fact perform competitively. However, our experiments also exhibit that Hybrid NN-HMM rely more on their input transformation modules, in particular on the existence of a short-term temporal pattern detector which we implement via Temporal Convolutions. Finally, we demonstrate inter-compatibility between the representation learning stages of both solutions, indicating a convergence of representations to encode factors of variations in the inputs.

Between humans, interactions and communications necessitate more that comprehension alone, as we also learn and adapt quickly to novelties in our environment: new words, voices,

symbols, etc. The pendant of this capability in Machine Learning is formalized under the one-shot learning setting, which has been the subject of relatively few research works so far, in particular for sequential inputs. To tackle this problem, we propose a model built around a Bidirectional Recurrent Neural Network. Its effectiveness is demonstrated by testing the discriminative performances on isolated gestures recordings from a sign language lexicon. We propose several improvements over this baseline by drawing inspiration from related works and evaluate their performances, exhibiting different advantages and disadvantages for each.

RÉSUMÉ

Cette thèse a pour but de contribuer à améliorer les interfaces Homme-machine. En particulier, nos appareils devraient répliquer notre capacité à traiter continûment des flux d'informations. En effet, nous lisons, écoutons et observons des scènes spontanément pour interagir ou communiquer. Cependant, le domaine de l'apprentissage statistique dédié à la reconnaissance de séries temporelles pose certains défis : la détection de phénomènes définis simultanément dans le temps ou dans l'instant, le traitement de flux de données de grandes dimensions, l'inférence en temps réel, etc. Néanmoins, la recherche dans ce domaine a progressé continuellement au cours des dernières décennies, avec un nouveau souffle apporté par les récents progrès avec les réseaux de neurones.

Pour appuyer notre étude de ce sujet, nous avons sélectionné la reconnaissance de gestes comme exemple applicatif. Ce type de données présente de multiples avantages à nos yeux : premièrement, la signification des gestes repose sur un mélange complexe de poses corporelles et de mouvements, deuxièmement, les gestes sont encodés sous des formes très variées avec des représentations en faible ou grande dimension ; enfin, la faible expertise présente dans ce domaine —comparé à l'écriture ou la parole— permet de mieux mettre en valeur l'importance de l'apprentissage automatique de représentations, qui conditionne la facilité à réutiliser un modèle sur des domaines différents.

La première partie de notre travail examine deux modèles temporels de l'état de l'art pour la reconnaissance continue sur des séquences, plus précisément l'hybride réseau de neurones–modèle de Markov caché (NN-HMM) et les réseaux de neurones récurrents bidirectionnels (BD-RNN) avec des unités commandées par des portes. Plutôt que de consacrer notre étude à l'optimisation des performances, cette thèse se focalise sur l'analyse des propriétés majeures caractérisant ces deux modèles. Pour ce faire, nous avons implémenté un environnement de test partagé qui est plus favorable à une étude comparative équitable. Nous proposons des ajustements sur les fonctions de coût utilisées pour entraîner les réseaux de neurones et sur les expressions du modèle hybride afin de gérer un large déséquilibre des classes de notre base d'apprentissage. Bien que les publications récentes semblent privilégier l'architecture BD-RNN, nous démontrons qu'il est possible d'obtenir des performances comparables avec l'autre approche. Néanmoins, nos expériences montrent aussi que le succès de l'hybride NN-HMM est conditionné sur la modélisation des entrées dans les premières couches du modèle. Celles-ci doivent en particulier modéliser les phénomènes temporels à court terme que nous détectons à l'aide de convolutions temporelles. Nous montrons aussi que ces représentations sont largement inter-compatibles entre les deux modèles, ce qui démontre une convergence dans la manière d'encoder les facteurs de variations des entrées.

La compréhension ne représente qu'un seul aspect des échanges et interactions entre humains. Nous utilisons aussi largement notre faculté à apprendre et à s'adapter rapidement à des nouveautés de notre environnement : de nouveaux mots ou symboles, de nouvelles voix, etc. L'équivalent de cette faculté en apprentissage statistique est formalisé par le paradigme de l'apprentissage dit « en un coup », qui a reçu une attention relativement faible de la part de la communauté scientifique, en particulier pour le traitement de données sous forme de séries temporelles. Pour aborder ce problème, nous proposons une architecture de modèle construite autour d'un réseau de neurones bidirectionnel. Son efficacité est démontrée par des tests de classification sur des gestes isolés issus d'un dictionnaire de langage des signes. À partir de ce modèle de référence, nous proposons de multiples améliorations inspirées par des travaux dans des domaines connexes, et nous étudions les avantages ou inconvénients de chacun.

Acknowledgement

It is an understatement to say that my family was very supportive during my studies, in particular during the rough times of prep school. Maman, Papa et Aurélie, you always worried I had to deal with my studies alone, but you'd be surprised to realize how much time people around me devote to logistics! You have compensated a hundred times more with your help, attention and kindness. A special thank is reserved to Mima: with the thousands of cookies you made during my studies, you have kept my energy high and my moral up (as everyone knows, chocolate contains antidepressant compounds!).

For as long as I can remember, machines have always exerted a fascination on me, either by their complexity, or through the elegance of a simple and efficient solution. My dream job is and has always been to "solve problems" with them. On the journey that led to this thesis, my dad played a significant role: by demystifying mechanics, he fostered my curiosity and set me en route to ever want to know more. Thanks to him, the naive and almost mystical observation of machines has been replaced by the pleasure of guessing and understanding their inner workings.

My desire to adopt a scientific approach in my work roots back to when I met Marcel Charron, to whom I would like to dedicate this work. As I struggled to make sense of mathematics in high school, he made them meaningful to me thanks to his incredible pedagogical skills. He taught me how to learn, showed me the beauty of a rigorous and well constructed reasoning and made intuitive the process of formalizing a real-life problem into a mathematical one. Scientists who have all been through this process will certainly appreciate the extent of this accomplishment. Yet to describe him as a teacher would be limitative. As a matter of principle, he would never accept payment for the courses he gave, and showed the greatest kindness to anyone around him. Marcel, I can wholeheartedly assert that you helped me set up the course of my life and define who I want to become, thank you so much.

With many very talented teachers during my studies, I have had the chance to extend my knowledge and have my attention drawn toward subjects that fit my tastes the best. I would like to express my deepest gratitude to Mounîm A. El Yacoubi, who first taught me about Machine learning, introduced me to the world of academic research, and then set up this thesis and directed me through it. We set ourselves ambitious goals and he has always kept me focused on the right path, even during periods of doubts. His relentless proofreading of my papers and presentations – including this manuscript – have substantially improved my explanation skill which is so essential in life, and for that I am very grateful. I also would like to thank the Institut

Mines Télécom and more particularly Bernadette Dorizzi who followed my application for the fundings which made this thesis possible.

Although I cannot list them all, it has been a great pleasure to meet all my colleagues. My impression is that passion grows by sharing it to others, and I have loved every discussion I had about other people's work and every exchange with fellow scientists during the seminars, conferences, summer schools, etc.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the last 50 years, the volume of interaction between humans, computers and the rest of the world has expanded immensely. Long gone is the time of physical terminals where a two-ways textual communication with instructions and feedback gave access to what could be summarized as a general purpose automaton. With diverse sensors, actuators, and increased computation power, computers have found uncountable applications in most domains of human activity. One fascinating branch of this expansion is Machine Learning, whose purpose is to equip computers with somewhat generic models capable to learn and generalize from observations like humans do.

Some compelling applications are now available for daily use by the laypeople: speech recognition for dictation or virtual assistants, handwriting recognition, etc. Furthermore, expert tasks have also integrated machine learning in a wide diversity of domains, for example: medicine with radiography or microscopic image segmentation [Ronneberger et al., 2015], mechanical failure detection with non-invasive railway damage probing [Lee et al., 2016], image de-noising [Lehtinen et al., 2018], and biometric identification [Taigman et al., 2014].

This thesis focuses on sequential data, in practice a stream or a time series of observations annotated by one or several labels possibly in a sequence as well. There are countless sources of sequential data in our daily activities on which Machine Learning could prove useful: speech, online handwriting, gestures, video recordings of objects, physical quantity measurements, etc. To set a more concrete objective to this thesis, gesture recognition from videos and body poses has been selected as an application example to illustrate and support the studies. The general motivation was to select a task that encompasses multiple modalities of various natures, low and high dimensional, and has a concrete application, in this case a human-computer interface.

The prediction of sequences can take different forms: in the isolated recognition task, a sequence of temporal observations containing a simple instance of a class is submitted for recognition, whereas in the continuous recognition task, the model can be charged to identify the successive instances of different gestures from a stream of input observations with possibly non-gesture segments in between the gesture instances. As an additional requirement, one may seek to obtain the temporal alignment of the instances (positioning beginning and end of gestures).

Another realistic setting is the one-shot or few-shot learning paradigm where only one or a few training instances are available for each class. Such a model could potentially learn new classes very quickly. Finally, one can foresee some use for the online learning and incremental learning paradigms for user or task adaptation. To illustrate these settings in a real world environment, one can imagine an industrial robot tasked to execute operations triggered by specific gestures interpreted as instructions. The one-shot learning and incremental learning aspects would help to program the robot for new instructions while the online learning could provide some level of adaptation with the human operator.

1.1 Overview of Gesture Recognition

1.1.1 Motivations

From the universally known gestures such as pointing or waving to the standardized military signs, gestures offers a flexible, practical and almost universal communication support which is encountered daily in our lives. Consequently, it is easy to imagine applications of a recognition system for human-computer interfaces in the line of existing solutions for speech and handwriting. The literature on the topic notably mentions video games [Ibañez et al., 2014], contact-less computer control in medical environment [Jacob and Wachs, 2014] and industrial robot control [Duan et al., 2017] as examples.

From a technical standpoint, recognizing gestures is a fascinating challenge. Indeed, the level of analysis and understanding of gestures is less extensive than in other fields with similar objectives such as handwriting or speech recognition, which explains the need for significant research effort on the modelling task. Moreover, appearance, shape and movement all play a role so that the information about gestures must integrate data sources of widely different structures such as colour video frames, depth maps, body pose...

Continuous Gesture Recognition is generally described as the spotting and recognition of segments carrying a specific meaning within a recorded stream of observations. These segments may contain static poses or dynamic movements of the body, arms, hands, and facial expression from a person who is trying to convey a message. Active communication through gestures naturally implies a certain level of cooperation from the subject: centred position in front of the camera with few or no occlusions, stable recording conditions to some extent without dramatic lighting changes or varying background for example. This is to be distinguished from activity recognition which tries to analyse a scene from a passive recording where the subject(s) need not take into account the presence of a camera or sensors. Typical activities include running, playing basketball, dancing, cooking, eating, observed by a sort of surveillance camera with a large depth of field. While closely related to gesture recognition, activity recognition often proceeds with different time scales, recording environments and needs to bear with additional difficulties resulting from the passive observation of subjects.

The closest related task to gesture recognition that involves physical action is probably sign language recognition, the pendant of speech recognition for hearing or speech impaired people, but it has received little modelling effort comparatively to its speech counterpart. Moreover, the syntax and the signs share an intricate and subtle relationship which renders continuous recognition particularly challenging. Practical applications will therefore realistically focus on simplified subtasks such as finger-spelling translation or lexicon-based translation[1].

In practice, Gesture Recognition more often draws the inspiration for its models from speech and handwriting recognition, two other sequence prediction tasks with which it shares sequentially structured inputs and similar prediction objectives. Consequently, a large corpora of research work is available from both domains, with foundational work dating back from the 80s.

1.1.2 Empirical Description of Gesture Data

The data for gesture recognition is typically composed of colour videos, depth maps videos and body parts coordinated in space. The advent of easily accessible depth cameras and most notably the Kinect sensor has greatly improved the availability of depth videos in gesture datasets. This new modality helps to disconnect pixels from separate 3D surfaces but which may appear as neighbours on the 2D frame matrix (for example distinguishing the arm in front of the body when the colour of the clothes is uniform on both). It is also more resilient to light conditions and motion blur. Furthermore, the depth map helps to accurately regress the body part coordinates in the 3D space as shown in Figure 1.1, which in turn proves to be a very informative cue for the recognition of gestures [D'Orazio et al., 2016].

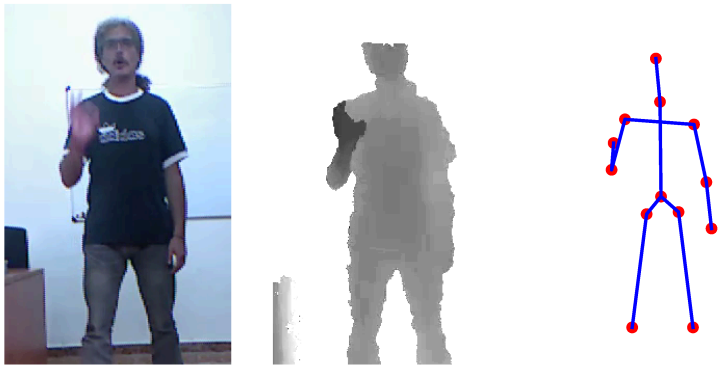


Figure 1.1 – Sample colour frame, depth map and body pose coordinates as returned by a kinect camera in the Montalbano v2 dataset [Escalera et al., 2014].

Some earlier works based on colour videos gave the subject coloured gloves to help locate and segment the hands from the rest of the image [Kelly et al., 2009]. Similarly, gloves have been equipped with accelerometers and gyroscopes to record the movement of the hands [Kim et al., 2009]. More recently, datasets with motion capture technology have been created in order to

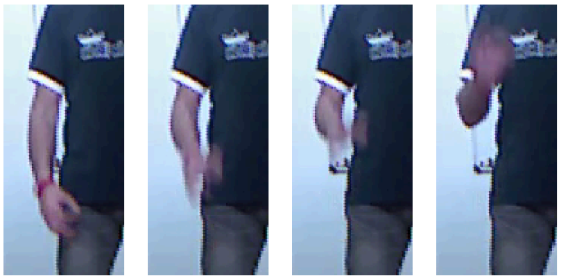
[1] one-to-one mapping of gestures with words

provide a very detailed and accurate positioning of the body parts [Braffort et al., 2015]. All of these methods substantially improve data measurement quality, but do so at the cost of usability thus restricting their use to laboratory experiments.

Video datasets typically contain recordings of medium quality: from 320×240 to 720×576 pixels in resolution and from 10 to 30 in frame rate. When available, the depth frames and body pose coordinates are synchronized with the colour frames. The subject upper body is usually centred in the frame so that it remains in sight. Some sources of variations in the recording may still remain: position of the light sources, distance from the camera to the body, various aspects of the environment such as background colours or surrounding objects. The subjects themselves bring their own share of variability: body size, skin colour, clothes, but also active variations such as the speed, the amplitude or the precision of the gestures. Figure 1.2 displays some illustrations of these issues.

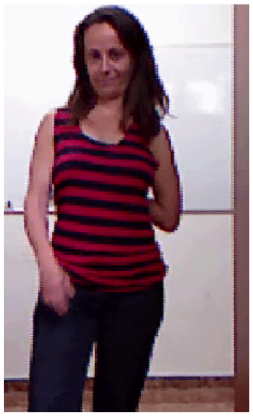
(a) cluttered environment

(b) poor illumination



(c) motion blur

(d) colour contrasts and colour inconsistencies (clothes, skin, etc.)



(e) occlusions

Figure 1.2 – Sources of variations and errors in video recordings. Samples are drawn from the Montalbano v2 dataset [Escalera et al., 2014]

For recognition, the annotations of a dataset should at least contain the list of gestures identified in each recording (or the class for the single instance in isolated recognition). Optionally, annotations may include temporal alignment providing the timestamps bounding each instance of gesture. Those frontier annotations obviously suffer from a certain level uncertainty introduced by transitioning movements that do not clearly belong to any class.

1.2 Continuous Recognition of Sequences

From a general perspective, continuous recognition of sequences belongs to the ensemble of classification problems. Formally, the objective is to predict a target y given an observation x. The *sequential* aspect means that x comes as a sequence of observations $x = (x_t)_{1 \leq t \leq T}$ where T designates the length of that particular sequence. All observations x_t contain the same type of data, usually one vector for each source of data or modality. For a large majority of practical cases, the observations are sampled uniformly along a dimension in space or time. As for the labels, three distinct cases arise in practice; first, the simpler isolated recognition task where each sequence bears a single class value y, for example a gesture label. In a more general case, a sequence contain $M < T$ class instances $(y_k)_{1 \leq k \leq M}$ to detect, for example a spoken word in audio recordings. This formulation is the most commonly studied one in speech and handwriting recognition, as it exonerate the annotators from providing a mapping between observations and targets; the existence of such a mapping is rarely obvious anyway since target instances may overlap or have uncertain frontiers. Moreover, temporal annotations require an unrealistic amount of annotation work for any reasonably large speech or handwriting datasets. Finally, the *continuous* recognition task we explore in this thesis corresponds to situations where observation time-steps $y = (y_t)_{1 \leq t \leq T}$ map one-to-one with a target label, therefore providing the temporal placement of instances to be detected. To avoid confusion: no hypothesis claims that y_t is *fully determined* by the observation at the same time-step x_t, in fact we will analyse the importance of temporal context in details through our experiments.

Besides the inherent challenges posed by classification, this class of problems is subject to additional difficulties. In absence of information about the temporal support related to a given target y_t (the set of observations which explain this particular outcome), one might need to process a very large section of the input if not the whole sequence to generate a valid prediction. Even under the hypothesis that only nearby time-steps relate to an output, the extent of the support for one class instance might vary from one sample to another and thus requires a model which can deal with inputs of variable size. The two-dimensional pendant of this issue is the recognition of objects at multiple scales in images. A second challenge stems from the dimension of the inputs, since their size essentially grows linearly with the number of time-steps per sequence. Contrary to image datasets where the sample size is determined by the recording device resolution, temporal sequences can have widely varying sizes depending on the durations. To put the dimension issue into perspective: 12 seconds of speech coded as a 26 feature vector every 5ms have the same dimension as an image of 256×256 pixels.

Depending on the desired application, a real-time inference constraint might be added, but more realistically the requirement will be relaxed to a constant delay, meaning the model can wait for a certain number of observations past a time-step t before returning a prediction for y_t.

Given the objectives and challenges stated above, a range of models have been conceived for various applications such as speech, handwriting or gesture recognition. It is generally safe to assume these models can be cast under the following pipeline (with different levels of integration between the modules):

Input data aquisition: This part provides a raw stream of observations over time (eg. videos frames, sound amplitude, etc.)

Feature extraction: This stage often comprises a set of basic pre-processing steps (standardization, differentiation over time) or more elaborate field-specific transformations (mel-spectrum or fourier transformations for speech, HOG features for RGB images, pose regression on depth maps, tracking, etc.).

With the advances of deep learning, a parametric model (often a Neural Network) can also provide suitable embeddings of the inputs for the upcoming recognition layers of the model. While such embedding functions are generally not fully interpretable, they are selected for their ability to provide semantically aware representations. The embedding should remain invariant to variations of the input data which are not relevant for the recognition task and should structure the representations so as to reflect principles of compositionality, semantic proximity, etc. For example: it is desirable that a characteristic hand shape from a gesture consistently binds to a given embedding vector regardless of the position of the subject in the image and independently of left or right handedness.

Temporal modeling: This part of the model is in charge of capturing any temporal pattern or temporal structure that may exist in the input. While this thesis will mainly focus on Hidden Markov Models [Baum and Petrie, 1966] and Recurrent Neural Networks, many other temporal models exist in the literature: Dynamic time warping [Vintsyuk, 1972], Conditional Random Fields [Sutton, 2012], Maximum Entropy Markov Models [McCallum et al., 2000] etc.

It should be noted that this stage can be made optional by the means of a prior segmentation step which identifies distinct monolithic parts of a sequence. Subsequent classification steps on these segments do not necessarily involve an explicit temporal model but may simply rely on aggregated statistics, for example Bag of Words representations [Wang et al., 2013].

Classifier: The separation between the temporal model and the classifier is not necessarily explicit, but a final module should eventually produce the decision about the classes and the temporal alignment of detected segments in the sequence if required. A vast number of models are available for this module but the choice is mainly conditioned by the structure of the temporal model and the desired type of outputs (isolated or continuous predictions, localized segments or simple event enumeration, etc.).

Focusing on building a relatively fast and light version of this pipeline, some works for isolated recognition have suggested using dynamic time warping (DTW) directly over hand-crafted features and run the comparison against reference prototypes [Li and Greenspan, 2011], therefore turning the classification into a re-identification task.

To relax the constraint of managing varying sequence durations in the model, several works have contributed to solutions producing representations of fixed size from a sequence by sampling a fixed number of key frames [Wanqing Li et al., 2008; Tripathi and Nandi, 2015], an approach that can incidentally provide invariance to execution speed and help control the dimension of the input for the classifier. Another straightforward solution to leverage classifiers for fixed data size is to simply use a sliding window over the input sequence, which, assuming the input window is sufficient to produce a confident decision, can effectively delegate the detection of temporal patterns to the classification module. A noteworthy illustration of this approach for continuous detection of gestures is given by [Neverova et al., 2014], where windows at multiple time-scales are used to quickly increase the context size and add some invariance to execution speed with a moderate computation cost.

Two temporal models are frequently used in state-of-the-art recognition models, Hidden Markov Models (HMM) and Recurrent Neural Networks. Among other similarities, both adopt an infinite recurrent process to model the evolution over time. Nevertheless, we also view these models as characteristic from two different schools of thoughts to solve the same underlying objective.

From a general perspective, HMMs build upon a probabilistic and generative approach focused on the observation data: judicious assumptions are made about the process that governs the environment and lead to these observations. These hypotheses also serve as simplifications which lead to more amenable inference computations within the probabilistic graphical model family. From an adequately designed model, a lot of insight is gained on this underlying process that can in turn help to solve a desired task, in our case continuous gesture recognition. HMMs are based on a combination of an observation model with a transitions model jointly optimized according to the maximum likelihood criterion [Baum and Petrie, 1966; Rabiner, 1989]. They provide a fairly interpretable and flexible class of models giving access to their internals for analysis and tuning guided by expertise acquired in the field. Training and inference can be done efficiently and robustly using dynamic programming. To leverage more complicated observation models, [Bourlard and Morgan, 1990] suggest some alterations to the HMM in order to trade the observation probability with a discriminative model, in practice a Neural Network. While the so-called Hybrid Neural Network Hidden Markov Model (NN-HMM) looses its generative capabilities in the process, the discriminative model is discharged from selecting hypotheses and learning the structure of observations. Those requirements can introduce additional learning effort and limitations that are only necessary to generate sample observations, but not directly relevant to the recognition task.

RNN and more generally Neural Networks proceed with a radically different approach, largely focused on the target task where modelling focuses more on representing data than

explaining it. Neural Network training revolves around the objective function that models the task in mathematical terms. The objective helps to learn the underlying model of the data as part of its optimization process and successfully does so on a large variety of tasks and data types. Based on the success of Hybrid NN-HMMs, one may wonder why the Recurrent Neural Networks, which are similar in their structure were not so successful until the last decade. In their vanilla implementation, the recurrent formulation unfortunately causes a rapid attenuation of signal coming from past observations and from the back-propagated error gradient [Hochreiter, 1998; Pascanu et al., 2013]. To circumvent the issue, [Hochreiter and Schmidhuber, 1997] proposed to replace the standard recurrent neurons with LSTM cells which are gated neurons that can dynamically reduce the apparent depth of the network with respect to time. With additional refinements [Gers et al., 2000; Gers and Schmidhuber, 2001], more appropriate objective functions [Graves et al., 2006, 2013] and increased computation power to train on large datasets, this class of model has eventually reached state-of-the-art performances and seen its popularity peak in a wide variety of sequence related problems: speech recognition [Graves and Jaitly, 2014], activity recognition [Donahue et al., 2014], video or image captioning [Xu et al., 2015], translation [Sutskever et al., 2014], etc.

1.3 One-Shot Learning of Gestures

Despite the name, one-shot learning principally formalizes the *inference and testing* conditions of a recognition model, within which a learning step occupies a predominant aspect. Indeed, this framework stipulates that, given a few samples from classes never seen before, a single instance in the most extreme one-shot case, a model should succeed in learning these classes and recognizing other instances.

How to learn from a few samples admittedly represents the central challenge of this problem. Indeed, traditional methods designed for large datasets representative of all observable variations of classes obviously fail in this setting, and new model and training techniques are needed to learn the most out of few training shots. Growing efforts contribute to bring the benefits of Neural Networks' robustness and adaptability to a setting with very little information to train-on. Nevertheless, a lot of attention is also invested into preparing the model prior to the one-shot learning session, which is why we view one-shot first as a testing paradigm rather than a learning one.

One-shot learning is relevant to gestures in terms of application and usage: one may wish to program home assistants or robots to react after specific gesture based instructions are given, in the same way we already do with voice based activation. Obviously, end-users won't participate in long and rigorous enrolment campaigns to provide numerous samples of their gesture based order, hence the need for one-shot capability. This thesis focuses on the isolated gesture recognition task, where a few labelled gestures instances is given as isolated data sequences, each from a class never observed before, and the model must correctly classify these gestures in an additional set of test sequences.

Compared to a recognition task in the regular paradigm, one-shot learning remains a barely investigated research field as of today. For example, we only found one publication studying Neural Networks with sequential data [Pei et al., 2016]. This is partly due to the difficulty of the task which requires a radically different approach to modelling and learning. However, recent publications [Santoro et al., 2016; Vinyals et al., 2016a] have laid out more precise specifications of the testing methodology, which help clarify the goals to achieve. The field is therefore quickly gaining momentum. Existing learning methods have been extended to accommodate one-shot conditions and new approaches have been developed: transfer learning, multi-task learning, meta-learning [Schmidhuber, 1987], etc.

1.4 Contributions

At the core of this thesis lies the idea to learn and detect sequentially structured patterns, more precisely temporal ones.

The first part of this thesis gathers work on continuous recognition of sequences, a research framework with a long history that has reached end-users in their daily life with speech recognition for example. Research on the subject is still very strong and active, pursuing numerous objectives including the expansion to new application fields or data types and the improvement of performances. Rather than focusing on optimization aspects, this work puts more emphasis on the analysis of two state-of-the-art temporal models, namely the Hybrid NN-HMM and Recurrent Neural Networks. With their spectacular progress and state-of-the-art performances, the latter have supplanted the former as a de-facto standard for continuous recognition. Yet both models share interesting similarities, a Neural Network based representation learning stage at the input to begin with. They also rely on state based hidden representations internally with discrete states and real vectors for the HMM and RNN respectively, and their inference algorithm both involve a recursion of linear transformations followed by a non-linearities. This work aims to inspect more finely the differences and similarities between Hybrid NN-HMM and RNN models using practical and realistic experiments able to demonstrate their advantges and weaknesses in practice. Gesture data shines for this purpose as it provides a variety of modalities and data representations with different levels of complexity.

Our contributions begin with the creation of a testing framework with a modular design that ensures a fair comparison between the two models while ensuring optimal training and inference conditions for both models. We run tests on a gesture dataset from a former competition with rigorous evaluation methods and compare our work to previous publications. Our design is validated by demonstrating state-of-the-art performances for both models, on body-pose data, video data and a combination thereof.

Following this essential step, we develop a series of studies that analyse and detail influential aspects of the models and training procedures that impact recognition performances. In particular, we propose alterations to the training procedures of both models to take into account and alleviate issues related to imbalanced representation of classes in the dataset.

Inversely, we perform a detailed post-training analysis of errors and parameter values for each model in order to reveal and explain important properties. Our experiments show that RNN models manage to achieve a good frame-wise accuracy while missing the concept of gesture duration nonetheless, which leads to noisy predictions and suboptimal performances on short gestures.

Although the literature often report Hybrid NN-HMM with inferior results compared to end-to-end RNN models, we show that most of this difference can be bridged by adding sufficient temporal context into the posterior state model. A comparative study is performed to ascertain the role of this context. We observe that RNNs rely very little on that context and feature a great robustness to the variation of input type and quality overall. We further verify this robustness through a series of Transfer Learning experiments that challenge the model with input representations that were not designed for the given model, but are known to contain relevant gesture information. These experiments also reveal that both models learn largely compatible representations of the input data within their respective representation learning stage.

The second part of this thesis concentrates on a different gesture recognition task: one-shot and few-shots learning of isolated gestures. Little work exists on one-shot sequence recognition in general. Our work contributes to this field by proposing a model based on Recurrent Neural Networks, which improves an existing proposition by [Pei et al., 2016] using a more modern and refined architecture. Using this model as a baseline, we propose several improvements by changing the inference method and training objectives. We also report experiments conducted with the Matching Network model [Vinyals et al., 2016b] on our gesture data. This model adopts the meta-learning principle in its design, a very promising area of research for one-shot learning.

Chapter 2

State of the Art

This chapter will introduce the state-of-the-art models for continuous sequence recognition, with a bias toward our select support application: Gesture Recognition. We first introduce the Hybrid Neural Network - Hidden Markov Model and explain how it can be trained and used for continuous sequence recognition. The same presentation is then provided for the Recurrent Neural Network. Finally, a section is dedicated to an ensemble of recent Neural Network techniques which participate substantially to the performances of our models.

Table 2.1 defines the notations used through this thesis, more specific notations will be introduced when needed to facilitate comprehension.

$\boldsymbol{v}, \mathbf{M}$	*(bold)*	multidimensional data
s	*(small + thin)*	sets, scalars
$\mathrm{S}, \mathrm{K}, \boldsymbol{\theta}$	*(straight)*	sets/constants/parameters
t, X	*(italic)*	variables/random variables
$\mathcal{Z}$	*(calligraphic)*	sample spaces, ensembles
$x \sim X$	x sampled from random variable X	
$p(X = x \mid Y = y; \boldsymbol{\theta})$	conditional probability of X given Y for a model parametrized by $\boldsymbol{\theta}$	
$p(x \mid y; \boldsymbol{\theta})$	implicit abbreviated version of the above.	
$E_X[f(X)]$	expectation	
$X \perp Y$	X independent of Y	
$[\boldsymbol{h}_1, \boldsymbol{h}_2]$	concatenation operator	
$\oplus, \odot$	element-wise sum, product	

Table 2.1 – Notations

2.1 Hybrid Neural Network – Hidden Markov Model

In this section, we introduce the Hybrid Neural Network — Hidden Markov Model (NN-HMM). This model is nowadays an industry standard solution for sequence recognition, for example in speech where it comes as a ready-to-use speech recognition module in several software libraries

such as CMUSphinx [Lamere et al., 2003] or HTK [Young and Young, 1994]. We will first introduce the standard HMM model, then introduce the common design choices for sequence prediction and finally detail the modifications needed to convert it to its hybrid formulation.

2.1.1 Hidden Markov Models: Definition

Hidden Markov Models define a family of Probabilistic Graphical Models for sequences of observations $\boldsymbol{X} = (\boldsymbol{x}_t)_{t\in[1..T]}$ with T the length of the sequence. The hidden aspect is brought in by adding series of latent state variables matching the observations at each time step $S = (s_t)_{t\in[1..T]}$. The HMM model is built on the assumption that the hidden states at each time-step govern the associated observations. In probabilistic words, this is enforced by setting a conditional independence of the observations given the latent variables: $\forall(t,t'), t \neq t', \boldsymbol{x}_t \perp \boldsymbol{x}_{t'} \mid s_t$. The HMM also simplifies the temporal model by enforcing a Markov transition assumption on the states: $\forall t, s_t \perp (s_1, .., s_{t-2}) \mid s_{t-1}$. The joint probability distribution therefore factorizes over the shared transition and observation models as:

$$p(\boldsymbol{X}, \boldsymbol{S}) = p(s_1)p(\boldsymbol{x}_1 \mid s_1)\prod_{t=2}^{T} p(s_t \mid s_{t-1})p(\boldsymbol{x}_t \mid s_t) \tag{2.1}$$

For the sake of clarity, we introduce a dummy state variable s_0 so that the initial state prior $p(s_1)$ can be merged into the product:

$$p(\boldsymbol{X}, \boldsymbol{S}) = \prod_{t=1}^{T} p(s_t \mid s_{t-1})p(\boldsymbol{x}_t \mid s_t) \tag{2.2}$$

The states variables take discrete values in [1 .. K] with K to be defined for the task. It is common to picture those values as K nodes in a graph and imagine that transitions from one state to another correspond to hops from a corresponding node to another. That transition model $p(s_t \mid s_{t-1})$ is governed by a categorical discrete distribution $s_t \sim Cat(\phi_{s_{t-1}})$ where $\phi_{i,j}$ is the probability of the transition from *i* to *j*.

To model the observations, many distributions are available; a popular starting point for continuous multidimensional observations such as speech features is often the Gaussian Mixture Model. Without loosing generality regarding the type of observation model, its parameters will be denoted by $\boldsymbol{\theta}$ for the remaining of this section, leading to the expression of the full joint distribution:

$$p(\boldsymbol{X}, \boldsymbol{S}; \phi, \boldsymbol{\theta}) = \prod_{t=1}^{T} p(s_t \mid s_{t-1}; \phi)p(\boldsymbol{x}_t \mid s_t; \boldsymbol{\theta}) \tag{2.3}$$

For the sake of clarity, however, the parameters will be omitted in the following of this thesis unless explicitly required.

2.1.2 Hidden Markov Models: Inference and training

Observation likelihood

To calculate the likelihood of a sequence of observations $p(\boldsymbol{x}_1, .., \boldsymbol{x}_T)$, one needs to eliminate the latent variables from the expression of the joint distribution:

$$p(\boldsymbol{x}_1, .., \boldsymbol{x}_T) = \sum_{(s_1,..,s_T)\in[1..\mathrm{K}]^T} p(\boldsymbol{x}_1, .., \boldsymbol{x}_T, s_1, .., s_T) \tag{2.4}$$

The combinations in the sum above grow exponentially with the length of the sequence rendering the direct naive summation impossible. Instead, a dynamic programming approach breaks the summation in a recurrent way using the *forward variable*:

$$\alpha_t(i) \stackrel{def}{=} p(\boldsymbol{x}_1, .., \boldsymbol{x}_t, s_t = i) \tag{2.5}$$

which is the probability to reach the state i at time t after generating the first t observations. This variable can take an efficient recursive formulation using the independence assumptions of the HMM:

$$\begin{aligned}
\alpha_{t+1}(j) &= p(\boldsymbol{x}_1, .., \boldsymbol{x}_t, \boldsymbol{x}_{t+1}, s_{t+1} = j) && (2.6)\\
&= p(\boldsymbol{x}_{t+1} \mid s_{t+1} = j) p(\boldsymbol{x}_1, .., \boldsymbol{x}_t, s_{t+1} = j) && (2.7)\\
&= p(\boldsymbol{x}_{t+1} \mid s_{t+1} = j) \sum_{i=1}^{\mathrm{K}} p(\boldsymbol{x}_1, .., \boldsymbol{x}_t, s_t = i, s_{t+1} = j) && (2.8)\\
&= p(\boldsymbol{x}_{t+1} \mid s_{t+1} = j) \sum_{i=1}^{\mathrm{K}} p(s_{t+1} = j \mid s_t = i) p(\boldsymbol{x}_1, .., \boldsymbol{x}_t, s_t = i) && (2.9)\\
\alpha_{t+1}(j) &= p(\boldsymbol{x}_{t+1} \mid s_{t+1} = j) \sum_{i=1}^{\mathrm{K}} p(s_{t+1} = j \mid s_t = i) \alpha_t(i) && (2.10)\\
& && (2.11)
\end{aligned}$$

with the initial values $\alpha_1(j) = p(\boldsymbol{x}_1, s_1 = j) = p(s_1 = j) p(\boldsymbol{x}_1 | s_1 = j)$ or $\alpha_0(j) = 1$ if a fictional initial state is used. Using this recursion up to $t = T$ takes only $\mathcal{O}(\mathrm{K}^2 T)$ operations and gives access to the observation likelihood:

$$p(\boldsymbol{x}_1, .., \boldsymbol{x}_T) = \sum_{i=1}^{\mathrm{K}} \alpha_T(i) \tag{2.12}$$

Maximum-likelihood state assignment: Viterbi algorithm

The Viterbi algorithm is a variation of the previous algorithm for the estimation of the most probable state sequence given a series of observations:

$$\boldsymbol{S}^* = \underset{\boldsymbol{S}\in[1..\mathrm{K}]^T}{\arg\max}\ p(\boldsymbol{S}, \boldsymbol{X}) \tag{2.13}$$

This is once again resolved through dynamic programming. Let $V_t(j)$ be the probability of the most probable path ending in state j at time t; we note $B_t(j)$ the succession of states in that path, then:

$$V_t(j) \overset{def}{=} \max_{\boldsymbol{s}_{1..t-1} \in [1..\mathrm{K}]^{t-1}} p(\boldsymbol{s}_{1..t-1}, s_t = j, \boldsymbol{x}_{1..t}) \tag{2.14}$$

$$\overset{def}{=} p(\boldsymbol{s}_{1..t} = B_t(j), \boldsymbol{x}_{1..t}) \tag{2.15}$$

$$= \max_{k \in [1..\mathrm{K}]} \quad p(s_t = j \mid s_{t-1} = k) p(x_t \mid s_t = j) \times \underbrace{\max_{\boldsymbol{s}_{1..t-2} \in [1..\mathrm{K}]^{t-2}} p(\boldsymbol{s}_{1..t-2}, s_{t-1} = k, \boldsymbol{x}_{1..t-1})}_{V_{t-1}(k)} \tag{2.16}$$

The complete algorithm for the Viterbi maximum likelihood path computation is detailed in Algorithm 2.1.

1: $V_0(j) = 1, \qquad j \in [1 \,..\, \mathrm{K}]$
2: $B_0(j) = []$
3: **for** $t = 1$ to T **do**
4: $\quad k = \arg\max_{k' \in [1..\mathrm{K}]} V_{t-1}(k') p(s_t = j \mid s_{t-1} = k')$
5: $\quad B_t(j) = [B_{t-1}(k), j]$
6: $\quad V_t(j) = V_{t-1}(k) p(s_t = j \mid s_{t-1} = k) p(x_t \mid s_t = j)$
7: **end for**
8: **return** $B_T(\arg\max_{k' \in [1..\mathrm{K}]} V_T(k'))$ ▷ backtracking

Algorithm 2.1 – Viterbi algorithm

Maximum Likelihood parameters optimization

Since latent variables are present in the model, the maximum likelihood optimization of ϕ and $\boldsymbol{\theta}$, the parameters of the observation and transition models, is usually achieved using the Expectation Maximization (EM) scheme, also called Baum-Welch algorithm in that context. The computation in the maximization step also involves dynamic programming methods, this time with an additional backward pass which won't be detailed here (please refer to [Bishop, 2006] for additional details).

2.1.3 Hybrid Formulation

In this section, we detail how to use Neural Networks to estimate the observation probabilities of the HMM $p(\boldsymbol{x}; \boldsymbol{\theta})$. The so-called hybrid implementation of this idea was brought by [Bourlard and Morgan, 1990; Morgan and Bourlard, 1995] and uses the Bayes rule on the observation likelihood $p(\boldsymbol{x}_t | s_t)$ in 2.3 to bring up a predictive state posterior term in the formula:

$$p(\boldsymbol{X}, \boldsymbol{S}) = \prod_{t=1}^{T} p(s_t \mid s_{t-1}) \frac{p(s_t \mid \boldsymbol{x}_t) p(\boldsymbol{x}_t)}{p(s_t)} \tag{2.17}$$

The Bayes rule gives rise to three terms: a categorical distribution of the state priors $p(s_t)$, a prior on the observations $p(\boldsymbol{x}_t)$ which are assumed to be independent identically distributed, and a predictive posterior model of the state probabilities given the observations $p(s_t|\boldsymbol{x}_t)$. In practice, the latter is taken as a Neural Network state classifier with a softmax output interpreted as probabilities.

This modification does not alter the inference algorithms apart from the expansion of the likelihood model into several terms. However, the training procedure now requires to assign values to the states in order to fit the Neural Network parameters and the state priors. Since the states are unobserved variables, it is common to employ yet again an Expectation Maximization, leading to the training procedure described in Algorithm 2.2.

Algorithm 2.2 – Training procedure for Hybrid Neural Network-Hidden Markov Models.

1: assign arbitrary[1] state targets $\tilde{y}$
2: **while** $\tilde{y}$ not converged **do**
3: **for** n epochs **do**
4: fit state posterior neural network $p(s_t = \tilde{y}_t \mid \boldsymbol{x}_t; \boldsymbol{\theta})$
5: **end for**
6: evaluate state priors $p(s_t = \tilde{y}_t; \boldsymbol{\theta})$
7: fit transition model $p(s_{t+1} \mid s_t; \phi)$
8: realign $\tilde{y}$ to maximize obs. likelihood for updated model (Viterbi)
9: **end while**

Two aspects of this algorithm require a special attention:

- Step 4 is a complete Neural Network training procedure by itself with multiple epochs, learning rate schedules, etc.
- The state targets $\tilde{y}$ are solely used to fit the Neural Network. They are not tied to any annotations and solely constrained by the maximum likelihood criterion; as such, the whole training procedure remains *unsupervised*. In the next section, amendments to this procedure are added to integrate partial supervision.

With the generic elements of the Hybrid NN-HMM introduced along with the training algorithm, the final requirement to produce a working recognition system is to elaborate a proper interpretation of the model for the recognition task.

2.1.4 Specialization for Subsequence Detection

Up to this point, the predictive capacity of the Hybrid NN-HMM model has not been mentioned nor have the target annotations been utilized; the training algorithm presented previously only maximizes the likelihood of the observations. Since this work is mostly interested in continuous

[1] A better initialization will reduce the number of training iterations needed. A fairly robust heuristic will be presented later on for the semi-supervised version of this algorithm.

recognition, the objective considered here is to find the boundaries and the label for events within a sequence of observations, or equivalently to classify each observation at each time step in conformance with the annotations, for example: the event might be a word uttered over a few time-steps within a longer recording of a speech; the moment and signification of this word is given during training and must be inferred for prediction.

The approach which is largely used in the literature requires no modification to the HMM model and simply relies on the values of the states to perform the predictions. More precisely each state value is dispatched into one of the classes to be predicted so that given the most likely path through the states for a series of observations, the mapped sequence of labels is also directly available.

The number of states must be at least equal to the number of classes but it is often greater in practice, resulting in multiple states for each class. This apparent redundancy is mostly needed for composite classes built from several elementary units which can be handled by different states. For example the gesture *"hello"* contains a raising arm motion, an open hand pose and a lowering arm motion which are all very distinct elementary parts of the same class. While the state posterior model could potentially capture all these sub-units into one state, having several distinct ones facilitate learning and lets the transition model learn the interactions that may exist between these states, such as an ordering or a repetition. Besides, expert knowledge and assumptions can be injected more easily into the transition model for simple interpretable concepts: for example one gesture may not transition to another half-way through its execution, therefore many transition probabilities can be zeroed out and excluded from the training process. Figure 2.1 details the typical state structures and assumptions adopted for a speech or gesture recognition transition model [Rabiner and Juang, 1986; Rabiner, 1989].

Now that the model can perform classification, the training procedure needs to be altered so as to maximize the likelihood of the observations *subject to predicting the annotated label sequence*. Algorithm 2.3 shows the updated training procedure for the Hybrid NN-HMM, modified for supervised learning.

In effect, the states from each class are still realigned but only within the boundaries of each instance of a class as illustrated on Figure 2.2.

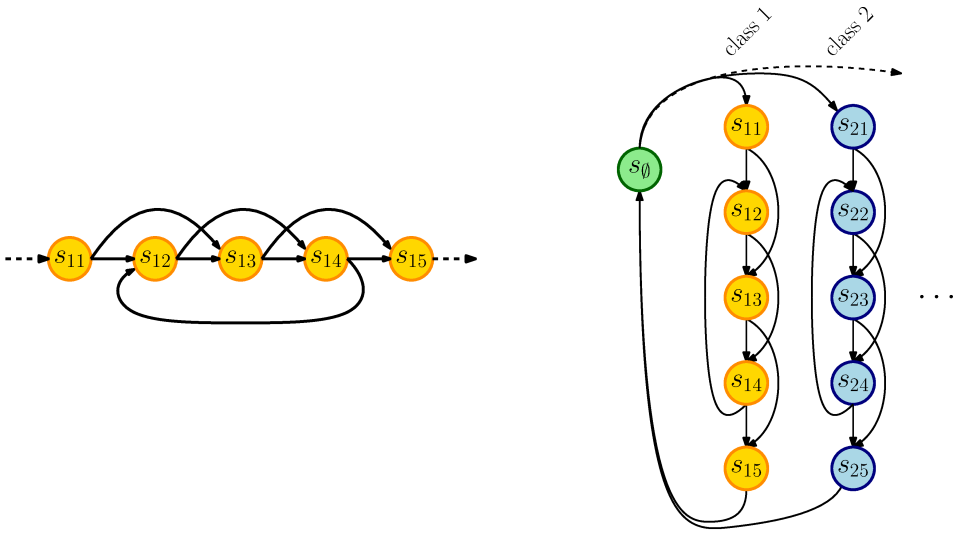


Figure 2.1 – Example of HMM state transitions for a speech or gesture recognition model.

(left) In speech recognition a common model of the phonemes (elementary units of speech) is often composed of 3 to 5 ordered sub-parts: a few core element surrounded by transitional parts. In the model, 3 to 5 states are allocated with the intent to map each sub-part. Left to right transition are added to account for the ordering between the initial transient part, the core and the final transient part. Optional core states and repetitions are implemented via skip and reverse transitions. All other transitions are impossible by default, so that the model cannot follow a path through states which contradicts the design of the temporal structure.

(right) Each group of states models one class and bears its label (colour-coded on the figure) so that transitions via these states indicate an occurrence of that class for the corresponding time-steps. In absence of transitions from one group to another, the HMM is forbidden to accept partial execution of any class. Additionally, a null state $s_\emptyset$ absorbs the silent moments of the sequence which do not have a special meaning and introduces a loop-back connection between the end and the beginning of the class states so that multiple occurrences can follow each others with blanks in between.

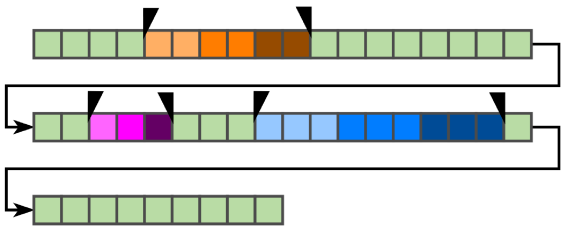


Figure 2.2 – Example of a state alignment constrained by class annotations.
The black marks show the delineation of the segment as provided by the annotator, the shades of a same colour denote different states from one class and the green colour symbolizes the null class.

let $(z_t)_{t\in[1..T]}$ be the sequence of labels
let $m : \tilde{y} \rightarrow \tilde{z}$ be the state-label mapping function
uniformly spread state targets $\tilde{y}$ within annotations $\forall 1 \leq t \leq T, m(\tilde{y}_t) = z_t$ (refer to Figure 2.2)
while $\tilde{y}$ not converged **do**
    for n epochs **do**
        fit state posterior neural network $p(s_t = \tilde{y}_t \mid \boldsymbol{x}_t; \boldsymbol{\theta})$
    end for
    evaluate state priors $p(s_t = \tilde{y}_t; \boldsymbol{\theta})$
    fit transition model $p(s_{t+1} \mid s_t; \phi)$ subj. to chosen hypotheses
    realign $\tilde{y}$ to maximize likelihood subj. to $\forall 1 \leq t \leq T, m(\tilde{y}_t) = z_t$
end while

Algorithm 2.3 – Training algorithm for Hybrid Neural Network-Hidden Markov Models.

2.2 Recurrent Neural Networks

2.2.1 Recurrent Neural Networks: Definition

Recurrent Neural Networks (RNN) are a reformulation of the basic feed-forward Neural Networks which can handle a variable-length sequence of inputs observations $(\boldsymbol{x}_t)_{t\in[1..T]}$:

$$\boldsymbol{h}_t = f(\mathbf{W}^\top [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t]) \tag{2.18}$$

where $(\boldsymbol{h}_t)_{t\in[1..T]}$ are the hidden state vectors, $\mathbf{W}$ are the parameters and f is a non linearity function such as tanh or the sigmoid function. Thanks to the recursive expression, the model itself is invariant through time translation but past inputs are still taken into account via the hidden states. At any given time-step t, recent inputs $(\boldsymbol{x}_{t-1}, \boldsymbol{x}_{t-2}, \dots)$ affect the value of the hidden state the most in practice, so the hidden states tend to behave as temporally conditioned embeddings of the recent or current inputs. Similarly to a regular feed-forward Neural Network layer, the output of the recurrent layer is composed of the hidden state vectors, a sequence in this case: $(\boldsymbol{h}_t)_{t\in[1..T]}$.

Sometimes, only the final hidden state $\boldsymbol{h}_T$ is used because its value is conditioned on all observations and might therefore summarize the information from the whole sequence. This is mostly relevant for non-continuous recognition where the sequence as a whole carries one label. A major advantage of this method is that it maps a variable length input to a fixed size vector which can be more easily fed into a subsequent classifier. However this technique introduces imbalance between the treatment of the first observations, which traverse many layers through time, and the final observations. By contrast, a simple averaging of all hidden vectors gives all time-steps a comparable importance, yet some of the first hidden vectors $(\boldsymbol{h}_0, \boldsymbol{h}_1, \dots)$ might be irrelevant due to the lack of information at this level of progression in the sequence.

In the most general case, all hidden vectors are tied to either subsequent layers, for example

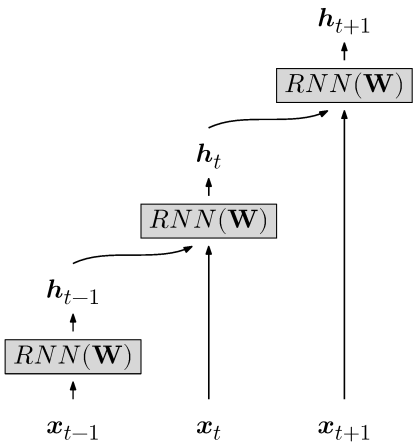


Figure 2.3 – Unfolded representation of a Recurrent Neural Network as a cascade of RNN cells with *shared* parameters.

another RNN, or to a mathematical operation, such as an aggregation function over time or a classifier. To train the parameters, a variation of the regular back-propagation called Back-Propagation Through Time [Werbos, 1988] recursively provides access to the derivatives with respect to the parameters and the inputs so that the recurrent layer as a whole can be considered as any regular Neural Network layer. Schematically, Back-Propagation Through Time "unrolls" Recurrent Neural Networks and then applies the regular back-propagation algorithm over the unfolded network graph as illustrated in Figure 2.3.

2.2.2 Gated Neurons for Long-Term Information Flow

Recurrent neural networks as introduced above are unfortunately notoriously difficult to train. Indeed, as the gradient of the loss at a given instant is back-propagated through time, it is multiplied at each time step by the parameter matrix and clipped by the non-linearity. Intuitively, one can imagine that these operations will attenuate the influence of the gradient in only a few time steps, therefore incapacitating the network from learning long range temporal patterns.

The favored solution to this issue as of today was introduced in [Hochreiter and Schmidhuber, 1997] as a shortcut mechanism with respect to time later refined by [Gers et al., 2000; Gers and Schmidhuber, 2001]. Their modification of the neurons into Long Short-Term Memory units allows a Recurrent Neural Network to dynamically reduce the number of transformations that happen between distant events. More precisely, a set of open gates can cut-off the influence of the input while closed gates transfer the hidden state to the next time-step mostly unchanged. This more complex take on the Recurrent Neural Network is nevertheless still compatible with back-propagation through time and can be used and trained like a vanilla RNN.

The LSTM unit obeys the following set of equations where $\mathbf{W}_\cdot, \mathbf{V}_\cdot, \mathbf{U}_\cdot, \mathbf{b}_\cdot$ are parameter

vectors and matrices, σ_g is the sigmoid function and σ_h the hyperbolic tangent:

$$f_t = \sigma_g(\mathbf{W}_f \boldsymbol{x}_t + \mathbf{V}_f \boldsymbol{c}_{t-1} + \mathbf{U}_f \boldsymbol{h}_{t-1} + \mathrm{b}_f) \tag{2.19}$$
$$i_t = \sigma_g(\mathbf{W}_i \boldsymbol{x}_t + \mathbf{V}_i \boldsymbol{c}_{t-1} + \mathbf{U}_i \boldsymbol{h}_{t-1} + \mathrm{b}_i) \tag{2.20}$$
$$o_t = \sigma_g(\mathbf{W}_o \boldsymbol{x}_t + \mathbf{V}_o \boldsymbol{c}_{t-1} + \mathbf{U}_o \boldsymbol{h}_{t-1} + \mathrm{b}_o) \tag{2.21}$$
$$\tilde{\boldsymbol{c}}_t = \sigma_h(\mathbf{W}_c \boldsymbol{x}_t + \mathbf{U}_c \boldsymbol{h}_{t-1} + \mathbf{b}_c) \tag{2.22}$$
$$\boldsymbol{c}_t = f_t \cdot \boldsymbol{c}_{t-1} + i_t \cdot \tilde{\boldsymbol{c}}_t \tag{2.23}$$
$$\boldsymbol{h}_t = \sigma_h(o_t \cdot \boldsymbol{c}_t) \tag{2.24}$$

Figure 2.4 gives a schematic representation to help visualize the logic behind the equations.

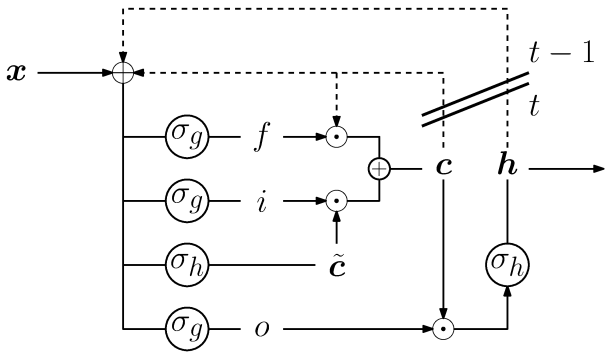


Figure 2.4 – Long Short-Term Memory unit. Parameters and affine transformations have been omitted for clarity.

The recurrence is built on a bipartite representation composed of the internal cell state $\boldsymbol{c}_{t-1}$ and the hidden state $\boldsymbol{h}_{t-1}$ which is also the output of the model. The cell update mechanism builds upon a recurrence over these two values and the fresh observation $\boldsymbol{x}_t$. With these three inputs, a new candidate cell value $\tilde{\boldsymbol{c}}_t$ is computed, along with two scalar gate values i_t and f_t. All of these computations are implemented as a single Neural Network layer, that is to say an affine transformation followed by a non-linearity. The tanh function is used as a non-linearity except for the gates which use a sigmoid. The input and forget gates i_t and f_t implement a dynamic compromise between updating the cell with $\tilde{\boldsymbol{c}}_t$ or keeping the previous value $\boldsymbol{c}_{t-1}$ as shown on Equation 2.23. Finally, the output $\boldsymbol{h}_t$ is computed from the updated cell state $\boldsymbol{c}_t$ using yet another neural network transformation. However, an additional output gate o_t is inserted before the non-linearity to throttle the amplitude of the neurons activity.

A simplified variant of the LSTM without a state cell has been proposed by [Cho et al., 2014] under the name Gated Recurrent Units. The authors observed that their cell performed comparatively to the LSTM at a fraction of the computational cost. Following the same rules of

notations as previously, its propagation rule is given by the following set of equations:

$$\boldsymbol{z}_t = \sigma_g(\mathbf{W}_z \boldsymbol{x}_t + \mathbf{U}_z \boldsymbol{h}_{t-1} + b_z) \qquad \triangleright \text{ update gates} \tag{2.25}$$

$$\boldsymbol{r}_t = \sigma_g(\mathbf{W}_r \boldsymbol{x}_t + \mathbf{U}_r \boldsymbol{h}_{t-1} + b_r) \qquad \triangleright \text{ reset gates} \tag{2.26}$$

$$\boldsymbol{h}_t = (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \sigma_h(\mathbf{W}_h \boldsymbol{x}_t + \mathbf{U}_h(\boldsymbol{r}_t \times \boldsymbol{h}_{t-1}) + b_h) \tag{2.27}$$

where $\odot$ is the element-wise product.

From now on, the RNN acronym will refer to the more general Recurrent Formulation of Neural Networks regardless of the presence of gated units; to avoid confusion, the original Recurrent Neural Networks without gates will be explicitly referred to as "vanilla"[1] RNN, and the other formulation will use the name of the cell (LSTM-RNN, GRU-RNN).

2.2.3 Architecture for Continuous Recognition

The recurrent neural networks introduced in the previous section admittedly benefit from a past context so as to return temporally aware representations or predictions, but this context is only oriented toward the past. Continuous detection without insight about future observations is similar to tackling the real-time prediction problem, an additional constraint which is not the matter of this thesis and was not imposed on the Hybrid NN-HMM model. Indeed, the prediction on the Hybrid NN-HMM does use future information brought in by the backtracking step of the Viterbi algorithm. A simple workaround to compensate for this lack of information in the RNN consists in using two networks: one reading the sequence from start to finish and another reading the reversed sequence as shown in Figure 2.5, a setup known as Bidirectional Recurrent Neural Network (BDRNN). The combination of the hidden states from the two (eg:

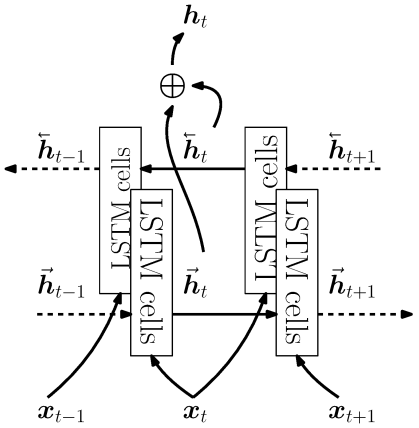


Figure 2.5 – Bidirectional Recurrent Neural Network

by concatenation or element-wise summation) gives a representation carrying both past and future context as desired.

[1] This adjective is commonly used in computer science to describe the default "flavour" of a concept.

One can compose multiple layers of Bidirectional Neural Networks by stacking them vertically over a sequence in order to increase the representational power of the model, in the same fashion as Multilayer Perceptron or Convolutional Neural Networks.

For very long sequences, it can be inconvenient to process a full sequence at once due to memory limitations and large padding overhead if multiple sequences of varying length are batched together (the longest sequence imposes the number of time-steps). Moreover, one may not want to wait for the end of the observation sequence to start reading predictions. Finally, the benefits of the temporal context around any given time-step t are normally limited to a neighbourhood $(t - \Delta_1, .., t + \Delta_2)$ which can be small in comparison of the sequence duration: $\Delta_1 + \Delta_2 \ll \mathrm{T}$. For example, in a full sentence of speech, observations from distant unrelated words most likely cannot help determine the label for the current word. As a result, the Bidirectional Recurrent Neural Network usually processes a sequence in overlapping chunks of fixed duration: $(\boldsymbol{x}_1, .., \boldsymbol{x}_c), (\boldsymbol{x}_{c-o}, .., \boldsymbol{x}_{2c-o}), (\boldsymbol{x}_{2c-2o}, .., \boldsymbol{x}_{2c-3o}), \ldots$ where c designates the chunk size and o the overlap. The chunk size c is selected so as to include more context than is presumable necessary. The edges of the outputs sequence are discarded to avoid returning predictions lacking from a sufficient amount of context, while the overlap makes sure that all time-steps of the sequence are eventually mapped to an output in one of the chunks.

Chunking solves all the aforementioned issues by compromising on the size of the temporal context; it caps the memory requirements to a known constant, and lets the model return its output within a constant delay since observations beyond the chunk will not influence its output. If the computation speed of the whole model remains sufficiently high and the latency introduced by the constant delay is judged negligible, this model can integrate a real-time prediction framework.

2.3 Recent Advances in Neural Network Training and Architectures

We compile here a series of short presentations for recent major contributions in the field of Neural Networks. Although the core of our work does not focus primarily on designing Neural Network modules and optimizing them, the methods presented in this section contribute noticeably to the performances of our proposed models.

Dropout

Dropout [Srivastava et al., 2014] is a noise based regularizer which disables (sets to zero) a fixed proportion r of the neurons in a layer before applying the non-linearity.

Dropout belongs to the class of regularizers which injects a destructive noise inside the model similarly to gaussian noise regularisation. However, randomly disabling neurons at each training iteration also amounts to temporarily train a different sub-model which contains only the retained neurons and merge it back into the full Network which is a form of model averaging.

Let $\mathcal{B}$ denote the Bernoulli distribution. For any neuron with output value $\tilde{h}$, the noisy output is given by:

$$h = \begin{cases} \frac{\tilde{h}}{1-r} & \text{if } d \sim \mathcal{B}(r) = 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.28}$$

The denominator scales up the non-dropped activations to preserve the average activation values over a whole layer regardless of the dropout rate r. This correction is needed for validation and testing where r is modified and set to 0.

The dropout rate, that is to say the proportion of disabled units, needs to be optimized for each problem, but rates up to 0.5 are not uncommon.

Rectified Linear Units

Activation functions turn the set of nodes outputs in Neural Networks, which are essentially a set of linear transformations, into a highly complex non-linear parametric function composed of multiple elementary blocks. To avoid computational issues and mimic biological neurons [Hodgkin and Huxley, 1952], bounded functions with an almost linear behaviour around zero have often been used, for example with the sigmoid or tanh.

More recently, a simpler activation function known as Rectified Linear Units [Maas et al., 2013] has become a de-facto standard for new Neural Network architectures:

$$ReLU(x) = max(0, x) \tag{2.29}$$

$$LeakyReLU(x) = 0.1 \times min(0, x) + max(0, x) \tag{2.30}$$

The leaky version is slightly more robust to the initial conditions during training. Indeed, its small slope on $\mathcal{R}^-$ avoids back-propagating zero gradient for neurons which are rarely activated as it sometimes happens during the first epochs.

Some of the motivations behind this non-linearity include:

- a non-symmetric behaviour which can implement more complex transformations
- a faster computation speed
- absence of shrinking or clipping effect on $\mathbb{R}^+ \rightarrow$ preservation of the back-propagated training signal
- increased activation sparsity and "dispersion"[1] [Willmore et al., 2000]
- improved performance metrics observed on various problems

[1] property of the activation distribution where input patterns are encoded over many different neurons instead of a few characteristic ones.

Batch Normalization

Batch Normalization [Ioffe and Szegedy, 2015] stems from the observation that gradient updates try to optimize all layers in parallel without taking care of the co-adaptation between them. As a layer learns new patterns, the distribution of its outputs changes therefore impacting subsequent layers. To reduce the impact of this shifting behaviour, Batch Normalization inserts a normalization step so as to ensure that the distributions of activations maintains a zero mean and unit standard deviation. The neuron outputs therefore becomes:

$$\forall k,\ h_k = f\left(\frac{a_k - E[a_k]}{\sqrt{VAR[a_k]}}\right) \tag{2.31}$$

where a_k represents the activation of the k-th neuron. The expectation and variance are empirically estimated on each minibatch during training whereas the correction during testing uses running statistics learnt during the training phase. This transformation is reported to not only accelerate training, its original purpose, but also to introduce some regularisation as well, therefore reducing the need for more destructive techniques based on noise injection.

Residual Networks

If the "Deep" aspect of Neural Networks has started to gain attention by 2009 [Krizhevsky, 2009], the number of layers in Neural Networks has truly soared with the introduction of gated layers [Srivastava et al., 2015; He et al., 2016]. Their foundational idea is arguably the same as the one introduced earlier for Recurrent Neural Networks with gated units, but this time the gating mechanism happens vertically across stacked layers of multilayer Neural Networks: instead of shorting time-steps as in RNN, gates render individual layers optional or "skippable" here. One proposition of this concept is given by Residual Layers from [He et al., 2016] where each block of a Neural Network proceeds by adding a correction to the output of the previous layer $\boldsymbol{h^l}$ instead of performing the usual affine transformation:

$$\boldsymbol{h}^{l+1} = Relu\left(\boldsymbol{h}^l \oplus \mathcal{B}lock(\boldsymbol{h}^l)\right) \tag{2.32}$$

where $\mathcal{B}lock$ contains a few Neural Network layers without the last non-linearity, which is moved after the element-wise summation with the input $\boldsymbol{h}^l$. Figure 2.6 provides an equivalent graphical representation of the Residual block. When $\mathcal{B}lock$ modifies the dimension of the hidden representation, for example with max-pooling or a layer with a different number of output neurons, the short-cut path is replaced with a very simple operation such as a linear transformation that returns compatible outputs.

Residual Networks introduce a short-cut path with little attenuation from the output down to the input layers. As a result, gradient descent is able to train very deep layers without major difficulties, and the original ResNet paper effectively demonstrates the feasibility of a thousand layer model.

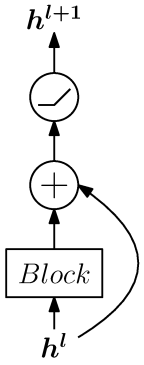


Figure 2.6 – Residual Block with its short-cut path.

For completeness, it should be mentioned that the residual blocks are normally not stacked directly one onto the other; instead, the final non-linearity inside a block is moved on top of the summation so as to also include the input $\boldsymbol{x}$, but since the ReLU function is used most of the time, the identity-like behaviour is preserved on most of the output values. Finally, the short-cut path may contain a simple transformation (no more than one Neural Network layer in practice) to accommodate for changes between the dimension of the input $\boldsymbol{x}$ and the output $\boldsymbol{h}$, for example when the block contains a max-pooling operation or when the number of neurons changes.

Accelerated Gradients

A large majority of the Neural Network models are trained via gradient descent. The algorithm updates the parameters $\boldsymbol{\Theta} \in \mathcal{R}^d$ of the Model by a small step of length $\lambda \in \mathcal{R}$ in the direction of the steepest descent of the loss function $L : \mathcal{R}^d \mapsto \mathcal{R}$, that is the opposite of the gradient vector. The surface of the loss function is estimated with the help of a training dataset X, Y, and the updates therefore look like:

$$\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - \lambda E_{X,Y}\left[\vec{\nabla}_{\boldsymbol{\Theta}} L(\boldsymbol{\Theta}, \boldsymbol{x}, y)\right] \tag{2.33}$$

Three different options exist to estimate the expectation value:

Batch updates use all available samples to produce the best possible estimation, but this takes a lot of time.

Stochastic updates, by contrast, use only one sample, with the assumption that for small step sizes, the surface of the loss function does not change quickly. Consequently, the average series of "updates" will approximately follow the path of the average "gradient", except that updates are more frequent and the descent might run faster.

Minibatches updates compromise between the above two points by averaging over a small subset of samples which is assumed to be representative of the whole dataset. This is by far the most commonly used option.

This generic yet reliable solution to Neural Network optimization suffers a few shortcomings related to the amplitude of the gradient and the sparsity of the updates with respect to the

different parameters. Some parameters related to less frequent patterns will indeed receive less frequent updates.

Some optimization algorithms try to escape flat regions of the loss function such as saddle points by using second order approximations of the gradient, but this class of method has not gained much traction in the Deep Neural Network community. Indeed, the second order approximation does not seem to apply well to this class of models in practice.

Instead, a family of "accelerated" gradient has been developed where the common principle is to adapt the learning rate for every parameter so as to avoid slow or over-confident updates for some subsets of the parameters. We briefly review here the Adaptative Moment Estimation (ADAM) [Kingma and Ba, 2014] which is used in our experiments. A more comprehensive definition along with a comparison to other variants of the gradient descent algorithms are provided in [Ruder, 2016].

For readability, we omit arguments other than parameters, and the gradient of the loss function at the t-th iteration with respect to the i-th parameter will be noted $g_{t,i} = \vec{\nabla}_{\theta_i} E\left[L(\mathbf{\Theta})\right]$. ADAM composes its updates from the expectation of past gradients $(g_{t'})_{t' \leq t}$ and squared gradients $(g_{t'}^2)_{t' \leq t}$. More precisely, it maintains an exponentially decaying average of both which can deal with a non-stationary distribution (the model changes at each update) while using little memory:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.34}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.35}$$

Due to the zero initialization $m_0 = 0$ and $v_0 = 0$, these quantities are biased toward zero, the authors demonstrate that the true estimates are given by:

$$\widehat{m}_t = \frac{m_t}{1 - \beta_1} \tag{2.36}$$

$$\widehat{v}_t = \frac{v_t}{1 - \beta_2} \tag{2.37}$$

Finally, the update rule is defined as:

$$\forall i, \quad \mathbf{\Theta}_{t+1,i} \leftarrow \mathbf{\Theta}_{t,i} - \frac{\lambda}{\sqrt{\widehat{v}_{t,i}} + \epsilon} \widehat{m}_{t,i} \tag{2.38}$$

with ϵ a small value to avoid instabilities in computations. The authors use the analogy of signal to noise ratio to describe the scaling factor $\sqrt{\widehat{v}_{t,i}} + \epsilon$ where the noise is estimated by $\widehat{v}_t$; the step size grows when a series of consistent updates have passed indicating a good signal and conversely decreases when an unexpected gradient value arrives. Additional arguments supporting this method are provided in their paper.

2.4 Brief Overview of Deep Learning

Before unveiling our contributions, this section draws the general outline of Deep Learning, which surrounds most of the work presented in this thesis. This section comes at the end of the chapter for different reasons. In terms of research interests, our work puts more emphasis on the comparison between Hybrid NN-HMM and RNN than on the "Deep" aspect of these models. In fact, the number of layers we use largely results from a pragmatic model selection when working with particular datasets and data types, and not from a deliberate architecture restriction. Without anticipating on our findings, it can be said that the properties of Deep Learning intervened more often in our analyses, observations and conclusions than in our research intentions. As a result, the purpose of this section is not to provide a detailed overview of Deep Learning but rather to exhibit certain aspects which influenced our work and our understanding of experimental results.

The Deep Learning Book [Goodfellow et al., 2016] stands out as an authoritative source on the matter, and provides this short definition of Deep Learning:

> Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones.

Deep learning in itself does not define a specific class of models or techniques, its existence in fact started in the continuity of the existing Machine Learning research. For a large part, the novelty lied on the commitment to have the model learn this hierarchy of concepts and to focus the research on solving the resulting optimization challenge. While the idea of nesting transformations was not completely new, it admittedly neglects theoretical work demonstrating that a single Neural Network layer is enough to approximate any function with arbitrary precision [Cybenko, 1989]. Instead, Deep Learning increases the capacity to abstract more complex factors of variations by adding more compositions into the computation graphs.

Representation Learning via Unsupervised Pre-training

Early works on the subject were rightfully concerned that the lack of training data would lead to dramatically over-fitted models and focused on unsupervised learning which can leverage large datasets of easily accessible unannotated data. These datasets are conceived to simulate the "true" data distribution: they contain samples representative of all inputs that might be observed later during testing. The Deep Belief Network (DBN, Figure 2.7) [Hinton et al., 2006a; Hinton and Salakhutdinov, 2006], from the family of probabilistic graphical models, represents a notable instance of these models. They are usually built by stacking several Restricted Boltzmann Machines (RBM), each trained successively to reconstruct their inputs based on an internal hidden representation composed of latent random variables. Although Deep Belief Networks are generative probabilistic graphical models, the inference of the top hidden units probabilities given the inputs happen to follow the same computation as a Neural Network. It is therefore

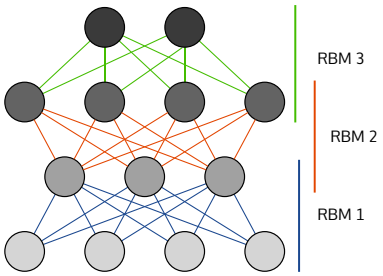


Figure 2.7 – Deep Belief Network

easy to fit a DBN unsupervisedly and use it to set the initial parameters of an equivalent Neural Network. Not only does the transition from DBN to Neural Network run smoothly in practice, but it also greatly reduces the number of training iterations needed by the full model to converge. In fact, [Hinton and Salakhutdinov, 2006] further observes that training from scratch may even fail for some datasets whereas pre-training renders the model viable. This multilayer stacked architecture illustrates perfectly the notion of nested hierarchy composed by a series of transformations.

A related model called Auto-Encoders [Vincent et al., 2010; Ronneberger et al., 2015] aims for the same unsupervised learning objective using this time a regular end-to-end Multilayer Neural Network trained to reconstruct its input. Due to the introduction of a low-dimensional "bottleneck" layer in the middle of the stack, the network cannot simply learn the identity function. To reconstruct the input, the network must generate efficient (i.e. compact) representations of the input at the bottleneck level that will discard irrelevant or redundant information but preserve concepts that explain the input and help reconstruct it[1].

Lifting Training Limitations

With the advent of large annotated datasets and the methods presented in the previous section (Dropout, ReLU activation, Batch Normalisation, Gated neurons), unsupervised pre-training seems to have lost some of its prevalence in later publications. In particular, the Computer Vision community benefits from particularly large datasets featuring millions of annotated samples. In that context, multilayer Neural Networks have demonstrated an impressive ability at learning deep hierarchies of representations, trained in a supervised fashion on object classification tasks. The sheer size and depth of the model alone does not explain this success. Convolutional layers and pooling [Lecun et al., 1998] helped share parameter values efficiently and implement known properties of images such as translation invariance into the models, eventually leading to a more efficient use of model parameters and a more efficient learning. The traditional approach to build a Neural Network from a stack of layers trained end-to-end was demonstrated to work well on surprisingly large models featuring as many as 19 layers and millions of parameters [Krizhevsky

[1] Note: not all information needed to reconstruct an input is relevant for a given task. On the opposite, notions such as translation invariance can be difficult to implement via the reconstruction metric which serves as objective function.

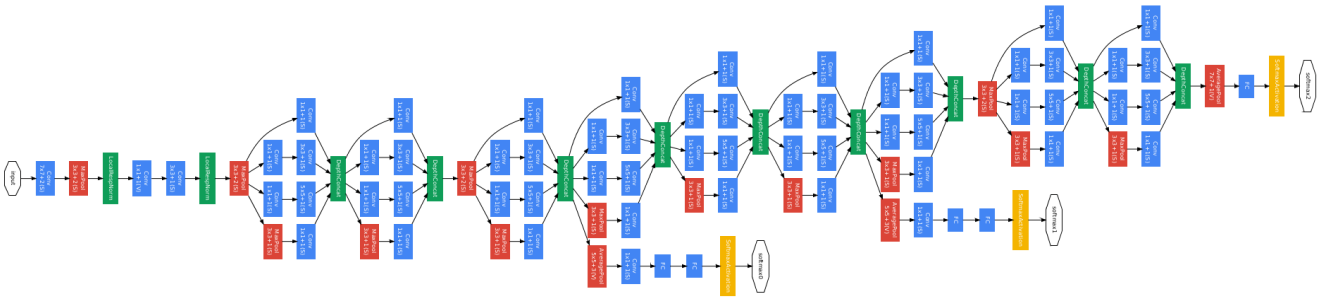


Figure 2.8 – GoogLeNet with 2 intermediate and one final outputs.

et al., 2012; Simonyan and Zisserman, 2014]. These deep Neural Networks learn hierarchies of more and more abstract features as indicated by the type of convolutional filters learned at each layer. By cutting these models and reading the representation vectors at a certain layer, a compromise can be found between the specialization for the training task and the level of abstraction to describe concepts present in the input. In practice, these models once decapitated of their specialized layers have been found to produce generic reusable features that perform well on a variety of Computer Vision tasks beyond the initial object recognition.

Architectural Novelties

Nevertheless, tackling larger problems by trying larger Networks remains prone to over-fitting issues. Besides, it should be acknowledged that the feasibility of the Deep Learning breakthrough was largely conditioned on the existence of fast computers and more precisely Graphical Processing Units (GPUs) reconverted for parallel computation, but the reduced rate of improvement these days motivates new axes of research to improve the performance/computation ratio. The remainder of this section describes contributions that we believe qualify into this general trend, including notably some of the techniques introduced in the previous section. For example, the Rectified Linear Units simplify the non-linearities to a binary mode transformation: either discard or copy. Residual connections enforce the assumption that representations can be formed by an iterative additive process instead of stand-alone transformations from one embedding space to another. Batch Normalization enforces an assumption on the statistics of neurons activations. In the inception Network [Szegedy et al., 2015], it is hypothesised that intermediate hidden states contain representations that are sufficiently elaborate to produce early guesses on the classification outcome over which loss functions are applied (Figure 2.8). Intermediate objectives help to reinforce the gradient signal for training in the lower parts of this 22 layers Networks. Some papers apply the divide and conquer principle by modularising their Neural Networks into specialized parts each optimized for one subtask, with an optional supervision on the subtask. For example, Spatial Transformer Networks [Jaderberg et al., 2015] use a Neural Network to eliminate deformations in images (scale, rotation, skew, etc.) before running an

object detection model. The latter will obviously benefit from the reduced variability of its inputs. A related example is brought by [Girshick et al., 2014] where a region proposal Network is trained to propose regions of interests that tightly enclose objects, the detection of which is delegated to a second Neural Network. In the same vein, a large body of work has investigated attention models which mimic the ability for humans to allocate brain resources on processing a targeted subset of the sensory inputs while discarding the rest. In these models, one module of the Neural Network is dedicated to select an attentional target and to produce a readout of the input that is subsequently fed into the remaining parts of the model. This approach is particularly useful when Natural Language is involved because the subject of the attention may evolve along the subject of a sentence; for example in image captioning [Xu et al., 2015].

All of these contributions tend to restrict the model in ways that do not hinder its capacity to learn representations but introduce simplifications and assumptions that help train more easily and more efficiently the Neural Networks.

Chapter 3

Comparing Hybrid Neural Network - Hidden Markov Models with Recurrent Neural Networks for Continuous Gesture Recognition

The latest developments in machine learning have seen the progressive replacement of Hidden Markov Models, even in their Hybrid formulation [Hinton et al., 2012], by Recurrent Neural Networks with Gated units in topics such as speech [Graves et al., 2006, 2013; Graves and Jaitly, 2014; Ravanelli et al., 2017], hand-writing [Graves and Schmidhuber, 2009; Bluche, 2015] or gesture recognition [Chai et al., 2016; Pigou et al., 2016; Plappert et al., 2017]. Gated units have now become a standard (all of the aforementioned publications use it) and the Bidirectional-RNN is almost systematically used when applicable, forming a robust and efficient go-to model to begin with on any experiments, at least as a strong baseline reference. In terms of recognition performances, the RNN models are often reported to perform better than the HMM, even under the Hybrid NN-HMM formulation which introduces Neural Networks into the posterior state model. However, most comparisons are run between largely different models with the RNN gaining the benefit of recent improvements in training techniques and model design. By contrast, the thesis work on handwriting recognition from [Bluche, 2015] only finds a narrow performance gap between these two models under comparable configurations. This chapter, which composes the core of this thesis, investigates suspected explanations for the differences, but also the existing similitude between Hybrid NN-HMM and RNN via a set of comparative experiments designed to target and illustrate these specific hypotheses. We also discuss the relative advantages of both models including (but not limited to) the detailed performances and failure cases.

At a broad level of observation the Hybrid NN-HMM and the Bidirectional RNN share some similarities: the use of a neural network to transform the input, a transition model to capture temporal structures, or even the existence of past and future context to back the prediction at any time-step. By contrast, the most obvious difference comes from the transition models themselves: the HMM receives a lot of expert knowledge in the form of a Markov transition hypothesis along with a trimmed structured state graph with many forbidden transitions which model known independence assumptions. One drawback of the Markov hypothesis is the limited amount of information transiting from one time-step to another via the transition model: only the index of the previous state.

The RNN transition model benefits from the flexibility and learning capacity of neural networks with a real state vector of fairly high dimension ($\mathcal{O}(100) - \mathcal{O}(1000)$) to carry information across time. However, Recurrent Neural Networks, like most Neural Networks, are hardly interpretable, and cannot enforce known hypotheses about the structure of the data as easily as the HMM.

Another difference stems from the learning procedure: the RNN features a simple and coherent back-propagation algorithm from the output down to the input. The Hybrid NN-HMM requires an iterative training procedure where the state posterior model is involved even though it is initially trained on largely suboptimal targets. The posterior state model remains relatively sensitive to the initialization heuristic and requires a careful balancing of the importance attributed to each state.

Based on these observations, we formulated a set of questions that could help understand the differences between the Hybrid NN-HMM and the bidirectional RNN:

1. Can the model learn by itself a representation from raw features which is suitable for the subsequent detection part? This ability is especially important for domains with little existing expertise and when working with complex high dimensional inputs.
2. How specific are the lower parts of the model with regard to the classification task? For many real-world applications, the training of the low level representation learning might be sub-optimal, for example due to the lack of training data or because the dataset is slightly biased from the actual distribution as is the case with transfer learning or artificially generated samples. This question is concerned with the degradation of performances in these cases where all the modules in the model were not fitted together, regardless of the ability of the model to do such a training (previous question).
3. How robust and flexible is the model regarding the nature of its inputs?
4. What are the limits of the temporal aspect in the model?

To shed some light on each of these issues, the models have been modularized as much as possible and a set of experiments has been devised so as to stress one particular aspect without interference from the others. More precisely a first experiment concerned with the standard end-to-end learning setting is run to establish a benchmark between the models, analyse their properties, weaknesses and differences. With the assurance that our models perform on par with

the results reported in the literature, the models are then retrained on data of various complexity and quality to exhibit robustness and adaptability. The inputs are changed from the simpler preprocessed body pose features to include cropped images of the hands either separately or jointly in a multi-modal configuration. The amount of temporal context is also varied to stress the short-term temporal modelling capacity. Finally, a set of transfer learning experiments are designed to explore the existence of model specific representation learning or on the contrary, the ability to learn generic reusable representations.

At inputs, colour video frames, depth maps and regressed body pose coordinates constitute the most commonly observed modalities in recent datasets from the field. They can be obtained reliably with off-the-shelves hardware which facilitates the construction of larger datasets and preserves the possibility of converting the recognition model into end-user applications easily. The colour frames provide aspect and shape information about the subject, all projected onto a 2D plane. Depth maps help to bring back the third spatial dimension which can be used to disambiguate the position of body parts and helps regress the body pose in space which itself provides a very informative cue when working with gestures.

The description of these experiments is preceded by several sections that provide a gradually detailed description starting from the state-of-the-art in gesture recognition, followed by a more detailed presentation of the supporting dataset data along with our data preprocessing pipeline to finally detail the architecture of our models and of the learning algorithms.

3.1 Continuous Gesture Recognition: a Review

This section reopens the analysis of continuous recognition methods this time with a more practical take geared toward continuous gesture recognition. The presentation follows the commonly observed preprocessing and recognition pipeline starting with data acquisition which we describe briefly, preprocessing and hand-crafted feature extraction, then representation learning and temporal modelling, to finally conclude with the classification stage.

3.1.1 Data Acquisition

In one sentence, gestures are defined by a specific succession and combination of held body pose, and movements. Depending on the task, "pose" description might reduce down to the relative positions of the main upper-body limbs (forearms, arms, torso), maybe including the full body posture with the legs, or even finer details (hand shape, head orientation, facial expression...). For a human, gesture recognition is purely based on visual cues but will implicitly eliminate appearance cues (color, texture, etc.) to focus on the underlying pose.

Logically, capturing data for automated gesture recognition should aim to extract as much information as possible about the pose. As mentioned already in section 1.1.2, wearable devices such as gloves with accelerometers and gyroscopes can provide this information with little data preprocessing. Motion capture also provides body pose data; although its acquisition pipeline

remains quite complex, it has been streamlined and perfected to high accuracy and reliability over the last decade thanks to widespread use in animated films and video games.

The majority of research papers focuses on data originating from cameras, mainly because recording requires little preparation from the subject and also due to the widespread availability of recording devices, which makes this approach the most realistic for real-world utilization. Although body pose cannot be readily extracted from camera feeds, algorithms have been implemented to extract it either from colour video frames [Toshev and Szegedy, 2014] or depth fields [Shotton et al., 2011]. Some gesture recognition models also skip these explicit pose regression stages in favour of an end-to-end video frame to gesture recognition. [Shi and Kim, 2017] notably trains a model with pose regression as a secondary task objective to help the model focus on this useful cue.

For this thesis, the decision was taken to opt whenever possible for options that reduce the amount of work needed to implement a production ready gesture recognition system. Consequently, the only data we used came from cameras available on the consumer market, more precisely Microsoft Kinects, which combine a colour and a depth field sensor.

3.1.2 Hand-crafted Features

An exhaustive overview of hand-crafted features would certainly go far beyond the scope of this manuscript, hence only a subset of the most illustrative and relevant methods is reported here; [Zhu et al., 2016; D'Orazio et al., 2016] provide comprehensive surveys on the matter.

Using the colour video frames, a family of interest points detectors has been created using either 2D techniques repeatedly applied over frames, or using 3D spatio-temporal volumes by stacking consecutive frames.

A category of these methods rely on detecting interesting points in whole sequences. Scale Invariant Feature Transformations (SIFT) is one such 2D method, which has been extended into Spatio Temporal Interest Points (STIP) in [Laptev, 2005] and other similar methods using various edges detection techniques. Dense tracking [Wang et al., 2013] propose to follow trajectories in space and time instead of isolated points. The information from the data around these points is then encoded using typical feature extractors such as HOG and HOF descriptors or 3D extensions for the depth map [Kläser et al., 2008], or even temporally aware representations. The data extracted around these points or trajectories is then aggregated using the Bag of Words technique to produce a fixed dimensional representation. Most of the time, the resulting feature vectors are low-dimensional or projected into a low-dimensional space via Principal Component Analysis and therefore require moderate computational resources, which makes them suitable for real-time processing or computationally limited applications as demonstrated in [Yin and Davis, 2014].

Our early exploratory work involved some of these techniques and revealed a series of limitations or shortcomings. Contrary to activity recognition, recognizing gestures sometimes requires a fine analysis of detailed shapes or movements of a particular body part. Points of

interest in their original formulation do not distinguish the semantic of points and statistical representations such as Bag-of-Words lack in subtlety for smaller but meaningful details like hand-shapes. We found that tracking and trajectory-based methods completely skipped fast displacements of the hands which can be essential to detect transitional movements. Conversely, held postures which also carry a semantic signification generate no trajectories and therefore need to be analysed via different features. Overall, these methods require a lot of expertise which generalizes poorly to other domains and does not contribute to the goals of this work, hence our decision to rule them out in this thesis.

The features extracted from body pose coordinates rely on typical feature engineering methods:

- temporal filtering [Seddik et al., 2014; Neverova et al., 2014] or curve fitting of trajectories [Koller et al., 2015]
- body joints coordinates (articulations between limbs and characteristic points such as hips and head) are often normalized around the centre of mass of the body, sometimes with a rotation to correct non-frontal subjects. [Seddik et al., 2014; Neverova et al., 2014] further propose to normalize limbs length to eliminate variations between subjects.
- orientation of the head, the torso and limbs [Miranda et al., 2014], angles formed by the limbs, which are often encoded by the cosinus and sinus values to ensure continuity between 0 and 2π
- pairwise distances between joints [Wu et al., 2012]
- first and second order derivatives, velocity and accelerations, [Lee and Kim, 1999] uses quantized directions and encodes feature into discrete states.
- and many other...

We will review our selection of features with more details as we present the preprocessing stage of our experiments.

3.1.3 Representation Learning of Observations

To produce a more abstract representation of the features, parametric models can be used to generate an embedding. This work deliberately focuses on Neural Networks for this task as they are predominant in the field as of today [Bengio et al., 2013], and also because they bind elegantly with both the Hybrid NN-HMM and RNN models we wish to study.

In the case of gestures, the expert knowledge is quite limited when it comes to find optimal features for the subsequent gesture detector. With video frame images in particular, the interactions between the pixel values and the gestures are extremely complex and operate in high dimensional spaces (do neighbouring pixels belong to the same object? Is the support deformable or rigid ? Do two pixels from successive frames correspond to the same moving object? etc.). For this type of problems, the so-called "Deep learning" methods regroup an ensemble of techniques and Neural Network architectures capable of processing inputs with little expertise and yet returning very compelling results. In fact, deep learning has helped improved the performances

in numerous domains beyond computer vision. One of its downsides, however, comes from the large amount of training data required in order to properly fit the parameters.

Besides the usual regularisation techniques (dropout, data augmentation, noise injection), a typical solution to resolve the lack of training data is to train the first layers of the model on another suitably large dataset. Indeed, these bottom layers often remain useful to produce sensible features on other datasets provided the underlying structure of the data remains similar. It should be noted that the task (regression, classification, detection, …) needs not be the same between the original and the secondary dataset. Using these 'pre-trained' parameters, the remaining parts of the model can be fitted on the original dataset. For example, [Koller et al., 2016a] reuses a Convolutional Neural Network model (CNN) designed for natural image classification in a sign language detector on video sequences.

Another solution for small datasets consists in pre-training the model in an unsupervised fashion, an approach which is mostly appealing on datasets with a lot of unlabelled data. Plenty of methods are mentioned in the literature, but to name a few: auto-encoders use self-supervision to train a neural network [Hinton and Zemel, 1994; Vincent et al., 2010], Restricted Boltzmann Machines [Hinton and Salakhutdinov, 2006] and Deep Belief Networks [Hinton et al., 2006b] fit on the data a generative graphical model that can be used to initialize neural network parameters (ex: [Wu et al., 2016] with body pose features for gesture recognition).

3.1.4 Temporal Modelling

The structure of temporal observation sequences shares with images the notion of local continuity between sample points: two consecutive observations are often closely related in the same way that two neighbouring pixels of a picture are. The analogy continues with translational invariance since temporal events often happen at random location within the sequence in the same way that objects might be placed anywhere within the field of view. A variable execution speed would result in a different scale of the pattern in recording which once again comes close to the apparent object size in the visual counterpart. As a result of these numerous shared properties, both fields have naturally shared and successfully adapted their techniques.

However, dealing with sequence of widely different durations (long or short recordings) is not rare, as long as individual patterns inside keep a similar structure, whereas managing images of very different size and resolution altogether in the same model is a more peculiar requirement in computer vision; the recording conditions are more or less standardized for most applications and datasets. Finally, continuous recognition of instance classes within a temporal sequence usually needs to run *without* a given number of instances to detect, therefore narrowing down the (still large) pool of potentially useful computer vision techniques.

Casting to Isolated Recognition

For the continuous recognition task, the model should account for the variable length of a sequence and the unknown number of instances to look for.

These two constraints can be isolated by dividing the model into a first sub-sequence proposal system which proposes locations for events to be detected. A classifier for *isolated* recognition can then predict the label for each sub-sequence proposal. Its task is made considerably simpler by having relaxed the continuous aspect of the problem in the previous module. A computer vision analogy is naturally found in the region proposal systems as in [Girshick et al., 2014] for example.

A simplified version of this two-step approach consists in predicting the classes from the vocabulary over a sliding window (systematic division of the sequence) and then fusing repeated class predictions into segments, a simple yet efficient solution as demonstrated on the Montalbano V2 gesture dataset by [Neverova et al., 2014; Pigou et al., 2016].

For a more refined selection of instance segments, blanks or silence detectors can sometimes help to find frontiers as is the case for [Wu et al., 2012]. On the other hand, active region proposal tries to detect where an instance starts and finishes. These two approaches do not account for uninterrupted chains of instances with specific transitions, as it happens in speech with successive phonemes of the same word.

Transition models which try to learn the pattern that connect two successive events are rarely used in practice; indeed, the number of combinations grows quadratically in the vocabulary size whereas the number of training pairs is inversely smaller with possibly missing combinations. Besides, detecting transitions remains essentially the same task as identifying the gestures themselves but with different targets, so the difficulty remains at least as high as originally. An elegant solution proposed by [Lee and Kim, 1999] takes the form of an adaptive threshold model which runs in parallel to a continuous detection system, both of which are implemented by an HMM in their work. The threshold model determines with its output the confidence level which must be attained by the detection system before its predictions can be accepted, defaulting to the null class otherwise. This discriminative method eliminates spurious detections which should fall into the null class and also handle transitions without explicitly learning them.

Reshaping and Re-indexing Inputs

The previous section introduced methods to reduce the problem from continuous to isolated recognition. Going one step further, some papers suggest to reduce the duration of the isolated segments to a fixed dimension, which is more practical to feed into classifier models.

To do so, a simple averaging along time might provide a naive form of aggregation. A geometrically decaying average such as the Motion History Images [Ahad, 2013] illustrated in Figure 3.1 can aggregate together successive observations with a focus on the more recent past.

To reduce rapidly the dimension of the input data without loosing information frame-wise, sub-sampling might be an option to consider. Besides straightforward uniform sampling, key-frame sampling detection builds on the assumption that most of the successive observations are not only correlated but moreover redundant, and thus only a handful of them are truly necessary to provide a compressed yet complete overview of a sequence [Bhuyan et al., 2004].

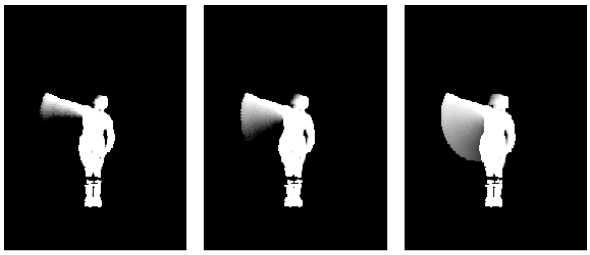


Figure 3.1 – Motion History Image on silhouette with various decay rates δ

Dynamic Time Warping also performs some sort of subsampling by dynamically reindexing the frames along a monotonically (not strictly) increasing index [Li and Greenspan, 2011; Konecny and Hagara, 2014].

Besides the inconvenience of manipulating variable length data, manipulating sequential data can sometimes lead to very high dimensional inputs, beyond the possibilities of many models and the hardware devices tasked to run them. Reducing the inputs as above will help to control the size of the data below the learning and processing capacities. When working with video images, one can sacrifice the holistic approach to focus on smaller regions of interest which are expected to carry most of the relevant information: for example the hands or the arms in gesture recognition [Neverova et al., 2014; Wu et al., 2016].

Statistical Temporal Models

Both previous subsections present methods that proceed mainly by circumventing the difficulties arising from modelling patterns in time. However, these methods often present some limitations as a result:

- Strong simplifying assumption on the data or targets.
- Lack of compatibility with end-to-end training, which weakens reuse and portability to other data types.
- Reliance on expertly designed features with the same issues as above.
- Poor compatibility with continuous on-line inference for real-time execution.

From the Probabilistic Graphical Model family, Hidden Markov Models provide a solution to all of the above issues and have demonstrated excellent performances over the last decades. Conditional Random Fields (CRF) offer a discriminative alternative and both models have seen numerous variations to improve learning of more complex hierarchical temporal structures. For example, [Lee and Kim, 1999] uses HMM with discrete distributions of gesture observations described by codewords. [Kelly et al., 2009; Wu and Shao, 2014] evaluates HMMs with a Mixture of Gaussian for the observation model on sign language and gesture recognition respectively. Their model demonstrates a successful adaptation of acoustic speech recognition models [Rabiner, 1989]. [Kelly et al., 2009] further improve their model to eliminate non-gesture segments with the addition of a threshold model

[Kelly et al., 2009; Yang and Lee, 2011] demonstrate the capabilities of CRF model for similar continuous recognition tasks. [Morency et al., 2007] proposes LDCRF, a variation of CRF with latent states. [Kurakin et al., 2012] demonstrates the effectiveness of a different probabilistic graphical model based on action graphs [Wanqing Li et al., 2008], which allows state sharing between gestures, thus reducing the number of states and therefore the amount of training data needed to fit the model.

More recent publications tend to favour the Hybrid HMM approach [Bourlard and Morgan, 1990]. For example, [Wu and Shao, 2014; Wu et al., 2016] uses a Deep Belief Network and a Convolutional Neural Network for the posterior state probabilities of a gesture recognition model taking body pose and video frames as inputs. Similarly, [Koller et al., 2016a,b] use a pre-trained Convolutional Neural Network to process videos of hand shapes from a sign language dataset.

Purely Neural Network-based temporal models only appeared in more recent publications for gesture and sign language recognition. Their architecture are structurally similar to their speech recognition counter-part with one or more layers of Bidirectional LSTM Recurrent Neural Networks. However, image based inputs require a more specific transformation to provide manageable input to the RNN layers. This transformation is automatically learnt in an end-to-end fashion by connecting Convolutional Neural Networks applied to each frame of the input as demonstrated by [Chai et al., 2016; Pigou et al., 2016]. Similar architectures are observed in action recognition [Donahue et al., 2014] and captionning [Xu et al., 2015][1].

3.2 Gesture Data and Preprocessing

3.2.1 Montalbano v2 Dataset

For the remaining part of this chapter, gesture recognition is chosen as the targeted application to illustrate continuous recognition on high-dimensional sequential data. Experiments are run on the Montalbano v2 dataset [Escalera et al., 2014], which was introduced for the 2014 edition of the Chalearn competition. This dataset is commonly used in the literature for its rigorous setting and several qualities that make it suitable for deep learning:

- official separation of the training, validation and testing sets
- somehow controlled recording conditions (at least lighting and distance to subject)
- precise annotations in time (beginning and end of gestures), and of course in class labels (Figure 3.2 shows a sample annotation)
- multi-modality: color frames, depth maps, and a reliable pose regression *in 3D* as shown in Figure 3.3
- fairly good recording quality: 640 $\times$ 480 resolution at 20 fps with subjects occupying a box of 300 $\times$ 500 pixels on average.

[1] This paper is concerned with static image captionning but processes them repeatedly in a sequence hence the relation with works on video data.

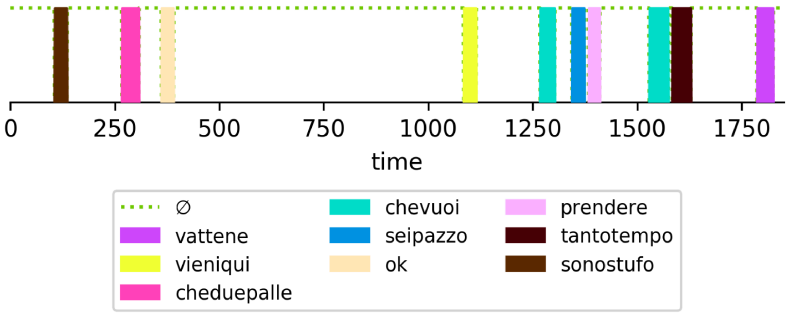


Figure 3.2 – Sample annotations for the Montalbano v2 dataset [Escalera et al., 2014]. Each frame is annotated by the labels of the active gesture or defaults to the null (=idle = ∅) category.

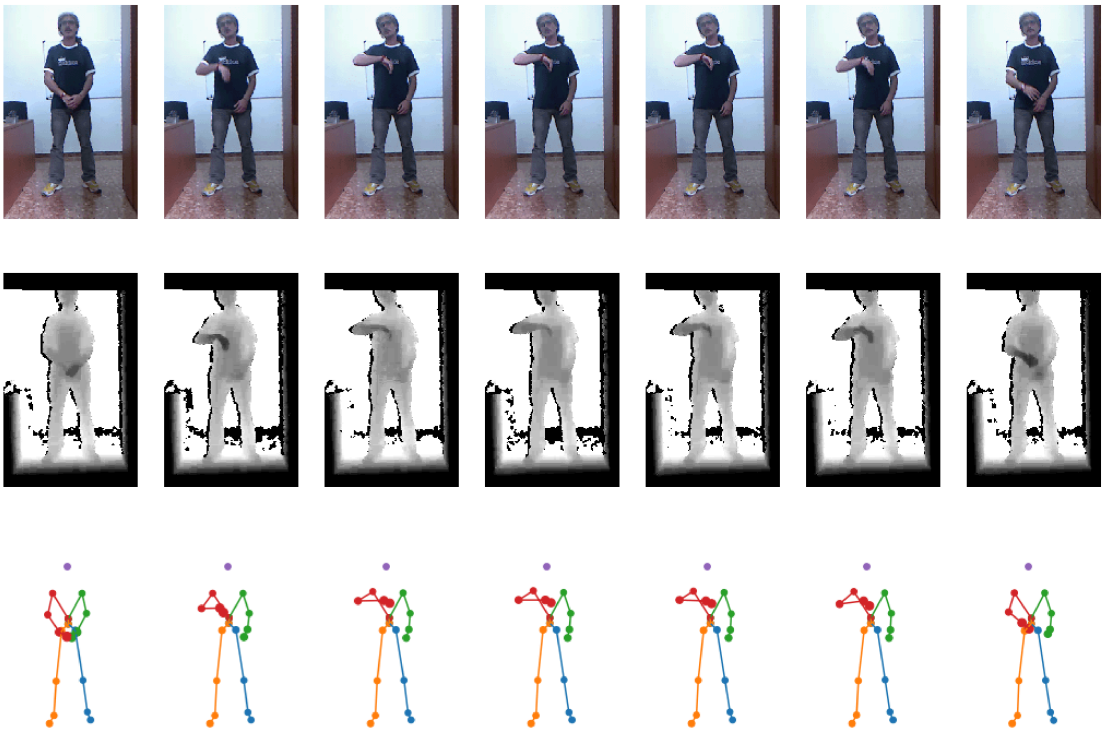


Figure 3.3 – Sample frames, depth maps and body pose coordinates

- large scale:
  - 940 gesture sequences with 1360 frames and 15 gesture instances per sequence on average
  - 13880 gesture instances across all 20 classes of the vocabulary as shown in Figure 3.4
  - 470 training gesture sequences containing 6847 individual gesture instances

Most of the gestures span between 20 and 60 frames with an average duration of 38.6 (Figure 3.5). Following up the discussion on temporal context and gated neurons in recurrent neural networks from the previous chapter, the necessary context to efficiently recognize a gesture might intuitively need to cover the last 20 to 60 observations so that the predictions for the end of a gesture are taken with all the related gesture data. For an RNN, this requires to successfully forward a signal through 20 to 60 effective layers over time, while grabbing incoming input data on the way, a very challenging prospect for a neural network unless special care is taken.

3.2.2 Data Augmentation

Data augmentation remains a very widespread way to regularize a parametric model and to make sure it will abide by known invariance. While video datasets may seem large and augmentation useless at first sight, it should be noted that successive frames often bring highly redundant information, and the same observation goes for neighbouring pixels within an image. Furthermore, with a majority of time spent by the subjects on resting idle between gestures, each label is only present between 1.6% and 2.5% of the 636727 frames in the training set depending on the class. The problem is not specific to gesture recognition but also commonly encountered in related domains such as speech recognition with silences.

To generate augmented recordings, we apply to each sequence a set of simple and intuitive transformations simultaneously for all modalities:

- flipping with a probability of 0.15 to help the model train on left-handed gestures normally performed as right-handed.
- random tilting in the range $-7°$ and $7°$
- random linear deformation of each spatial dimension from -15% to $+15\%$

A preview of these transformations is given in Figure 3.6. In all of our experiments, each training sequence is augmented four times using the above transformations in addition to the original unaltered version.

3.2.3 Data Preprocessing

The typical preprocessing adopted in this body of work takes its root from the commonly used methods reported in the literature [Seddik et al., 2014; Neverova et al., 2014; Wu et al., 2016]. Due to the comparative aspect of our work, this preprocessing is not optimized for one specific model. Instead, it is shared by all experiments to remove any interference with the matter of the research.

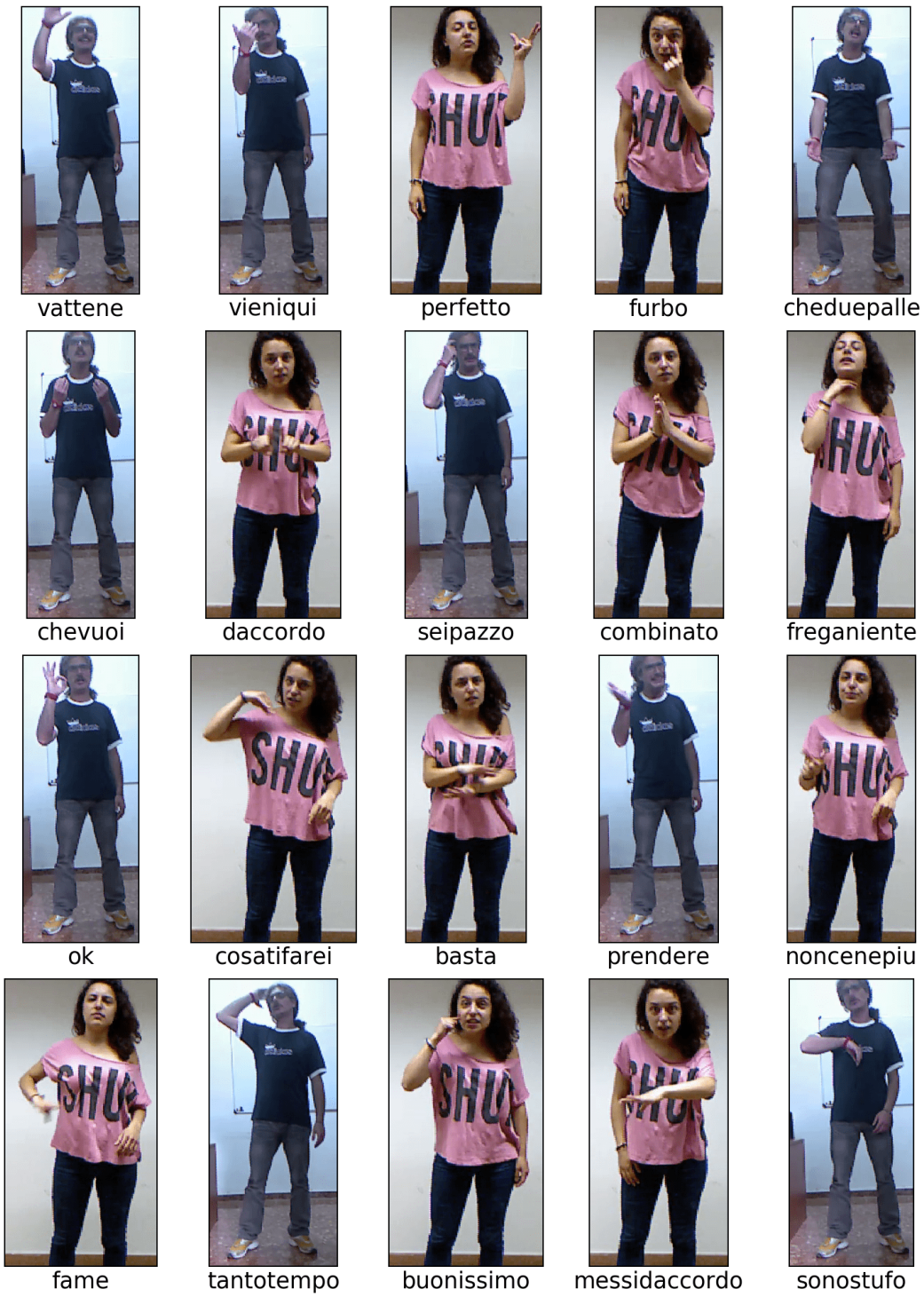


Figure 3.4 – Classes from the Montalbano v2 dataset

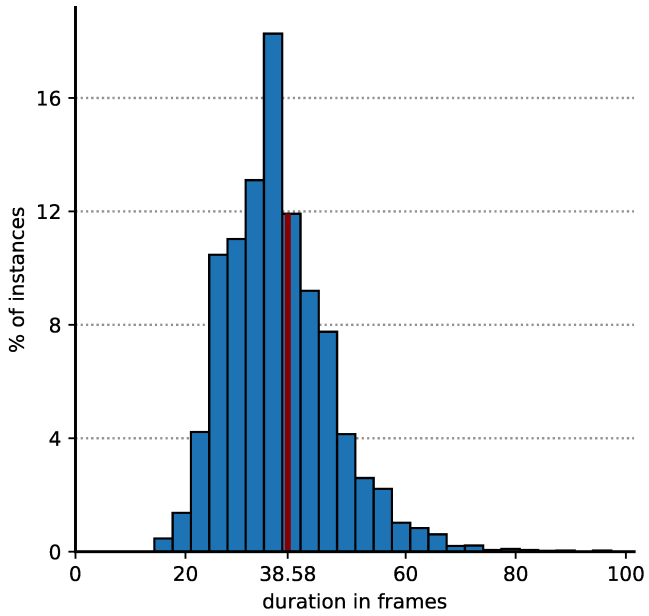


Figure 3.5 – Histogram of the gesture instance durations

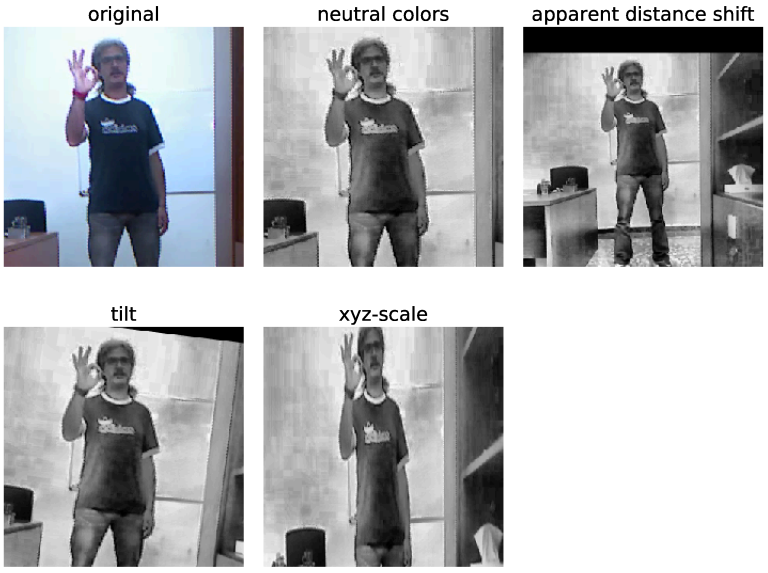


Figure 3.6 – Sample transformations and augmentations on image frames.

Body pose coordinates are obviously a very informative and therefore very desirable cue to recognize a gesture. The points of interest known as joints are: the hands, wrists, elbows, shoulders, hips, knees, ankles, foots, and the head. In this dataset only the 6 upper body joints (wrist, elbows and shoulders) and the head position are relevant so we deliberately ignore the others.

The pose regression provided by the Kinect device API is fairly robust, with seldom hand swaps when the hands are close to each other and/or occluded. Due to the absence of motion blur on the depth map, the system remains relatively robust even for a fast moving subject. However, the confidence on the pose detection sometime drops for other reasons (e.g. proximity with background, cluttered environment, unusual viewpoint), and no pose is provided at all. We use quadratic spline interpolation to infer plausible values for those missing poses. On 18 training sequences, however, the number of interpolated poses was judged too high to provide a sufficiently reliable representation of the actual poses; these recordings have consequently been completely eliminated as safety measure.

An additional Gaussian blur (with variance 1 in the experiments presented here) is used to smooth out pose detection noise and avoid spurious very fast movements between two frames.

From the raw position of the body joints, many features can be computed as illustrated with Figure 3.7:

- the positions relatively to the neck
- the first and second order derivatives of the joints
- the velocity (norm of the first derivatives vector)
- the pairwise differences and distances between joints
- the angle between the limbs (forearm-arm and arm-collar) and the vector orthonormal to that angle

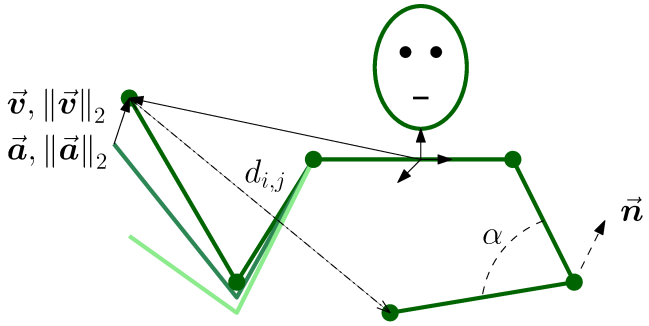


Figure 3.7 – Typical features extracted from the body pose coordinates

At each time-step, a vector with 248 dimensions is obtained. Those features then receive a typical normalization to remove the mean and variance. In an ablation study, we successively tested the removal of feature subsets, and concluded that position and speed are essential to the models, but all other features usually have a small positive impact, whereas no detrimental effect was found in adding all of them together.

Color images are first normalized to the apparent distance to the subject and therefore its size within the image. Holistic approaches will normally use a crop of the image around the upper body, but because most of the information from this input would be redundant with the body pose information[1], we will instead focus on small regions of interest around the hands to observe their shape in details, a complementary source of information.

Because the hand detections are often more noisy than the wrist ones, the region of interest is actually centred around the latter and a crop size of 32×32 pixels is chosen to add a sufficient spatial context.

Since colour is not a particularly reliable cue, the CLAHE contrast normalization [Pizer et al., 1987] is applied on each colour channel to remove lighting variations before switching to grey scale intensity images. The end result is visible in Figure 3.8.

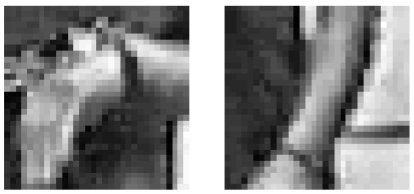


Figure 3.8 – Hand crops used as image features. Note that the right hand (image on the left) is flipped so that the model will not need to learn separate representations for left and right hands. The left hand crop presents a typical pose regression error where the predicted forearm length is smaller than in reality.

Depth maps processing is often similar to colour frames. Sometimes, the depth is simply added as a fourth colour channel. For competitive work, one would want to use them as a complementary or primary source of information because of their intrinsic qualities and the added information they may contain. For the work presented here, the benefit over the colour frames alone was judged too small to justify the extra computation cost.

3.3 Models Layout and Architecture

3.3.1 Representation Learning

Given the endeavour of this work to limit as much as possible interferences between the model components during our comparative study, it would be desirable to isolate each part and each aspect of the models in order to run specific and fair comparative experiments.

With the pre-processed body pose features, an RNN-based model could run directly above and process the sequences without transforming the features any further. RNN have been used

[1]This assumption is backed by the comparative experiments in [Pigou et al., 2016] where their holistic model barely benefits from additional pose features.

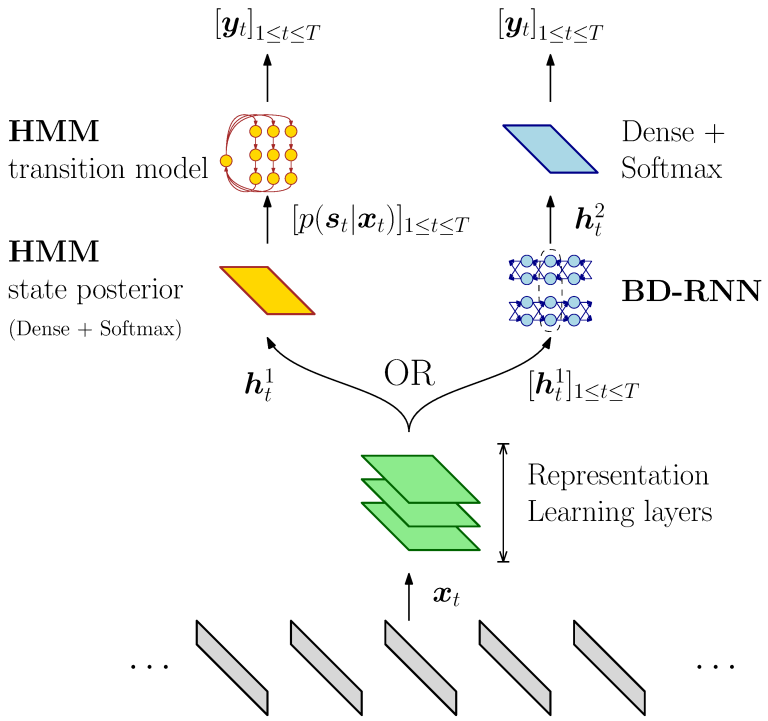


Figure 3.9 – Experimental setup with one shared representation learning path and the Hybrid NN-HMM or the BDRNN model

several times without time-step-wise representation learning as it has been demonstrated multiple times in speech recognition [Graves and Schmidhuber, 2009; Graves et al., 2013; Bluche, 2015] and even gestures with low-dimensional representation of hand shape features [Chai et al., 2016]. For raw images, however, a higher level and more synthetic low-dimensional representation is clearly required. In practice, another set of neural network layers operating in a frame-wise fashion provides this representation, for example a Convolutional Neural Network over the cropped hand images [Donahue et al., 2014]. Even for the body pose features, a Multilayer Neural Network can provide representation vectors more specifically tied to the task and model [Wu et al., 2012]. For the Hybrid NN-HMM model, these representation learning layers can integrate the state posterior Neural Network as its bottom section. For both models, the modification simply changes the architecture of a Neural Network but does not necessitate modifications in the training procedures.

To avoid introducing any unfair differences between the two models, we propose to use the same *shared*[1] feed-forward neural network operating in a frame-wise fashion as a representation learning model. Its outputs are directly fed into the Recurrent layers for the BD-RNN model while a single dense state classification layer is added to complete the state posterior model $p(s_t|\boldsymbol{x}_t)$ of the Hybrid NN-HMM as shown in Figure 3.9.

To avoid confusion: the representation learning model, the NN-HMM state posterior model and the dense layer above the BD-RNN all work in a frame-wise fashion, albeit the representa-

[1] "shared" only refers to the architectural aspect, no parameter tying is involved and the training procedures are completely distinct.

tion also include contextual observations as will be detailed later. By contrast, only the HMM transition model and the Bidirectional RNN layers process their input sequentially.

The same architecture is enforced below the BD-RNN branch and the Hybrid NN-HMM branch. However, the parameters are retrained unless specified otherwise. This shared architecture with free parameters means that the same learning capacity is given outside of the temporal model themselves, yet the final results will account for the ability to perform end-to-end learning.

Before unveiling the specific architectural details, it should be noted that state posterior models rarely run in a pure frame-wise fashion in order to achieve the best results. Depriving the model from temporal context might seem like a necessary condition to the independence assumption between the observations of the HMM. This hypothesis is used to factor the observation likelihood terms of the joint probability density function in equation 2.2. But it can rarely be enforced in practice because this hypothesis is often broken at the input level already due to the high correlation between successive observations. On the contrary, the best models usually provide some temporal context in order to improve the reliability of state posterior estimations, in particular with respect to input noise and frame ambiguity.

Consequently, the models in this work combine vectors from successive time-steps over a sliding window before returning the input representation vector. Our representation Learning module more specifically adopts the approach proposed by [Pigou et al., 2016], where the window of input values is aggregated into a fixed dimension vector by a dense neural network layer instead of a mean or maximum pooling strategy. In practice, this transformation is most efficiently implemented by a 1D convolutional neural network layer called *Temporal Convolution* (TC) in that context.

The basis of inspiration for our architecture is [Wu et al., 2016] which also uses a Hybrid NN-HMM model, and [Neverova et al., 2014] since both of these papers evaluate their models on the Montalbano v2 dataset. The representation learning model of the former is initialized by unsupervised pre-training using a Deep Belief Network mapped onto the representation learning layers, a technique popularized by [Hinton and Salakhutdinov, 2006].

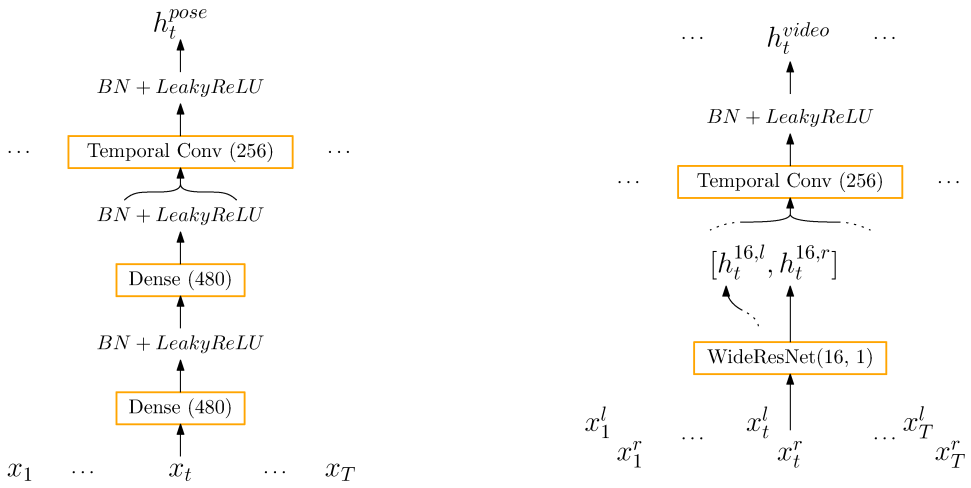
Our models only use supervised training but leverage more recent techniques such as batch-normalization [Ioffe and Szegedy, 2015] and Rectified Linear Units [Hahnloser et al., 2000; Nair and Hinton, 2010] which render pre-training optional. In fact the architecture presented here uses fewer neurons than [Neverova et al., 2014] which operates on the same data modality, yet our model still needs dropout regularization (with a dropping rate of 0.2 between the layers) to prevent over-fitting, an evidence that the training procedure and models perform efficiently.

The specific architecture of the representation learning model for the body pose features has the following specification (illustrated with Figure 3.10a):

- Two dense layers with 480 neurons taking frame-wise features followed by a Batch-Normalization Layer and a Leaky Rectification[1] [Maas et al., 2013].

[1] a variant of the ReLU non-linearity with a small slope on $\mathcal{R}^-$: $f(x) = max(0, x) + 0.01 \times min(0, x)$

- A temporal convolution layer with 256 filters also followed by a batch-normalization and a leaky rectification; its details and purpose will be detailed later in section 3.6.



(a) Body pose model

(b) Hand crops model

Figure 3.10 – Architecture of the embedding models Neural Networks. BN stands for Batch Norm, ReLU for Rectified Linear Units

For the video image frames, the noisy detection of the hand positions calls for a model that is invariant to small shifts, which is why a dense Neural Network is eliminated in favour of its Convolutional formulation. To our knowledge, CIFAR10 or CIFAR100 [Krizhevsky, 2009] are the closest well known computer vision datasets. Indeed our data possesses 21 classes or 101 states (counting in the non-gesture class) and 32 by 32 pixels images which is comparable to the 10 or 100 classes with colour images of the same size. As a result, the selected model for representation learning is chosen as a ResNet Convolutional Neural Network [He et al., 2016], or more precisely its enhanced wide-ResNet [Zagoruyko and Komodakis, 2016] variation as shown in Figure 3.11. This family of Convolutional Neural Networks is very flexible and achieves excellent performances on CIFAR10 or CIFAR100. Our work uses a reduced configuration of the original wide-ResNet to prevent over-fitting issues observed during our initial tests. Each of the two hands are processed separately by the CNN shown in Figure 3.10b, and the resulting hidden vectors are then concatenated to provide the input to a temporal convolution layer like previously. To summarize, the representation learning layers for the video frames are composed of:

- A stacking of the left and right hands as separate observations into the minibatch.
- An extraction of visual features via a Wide-ResNet CNN in the minimal 16-1 configuration for each hand.
- A concatenation of the left and right hand hidden vectors.
- A Temporal Convolution layer with 256 filters followed by a batch-normalization and a leaky rectification.

All architectures presented here were established through iterative refinements and cross validation of the meta-parameters (number of neurons, dropout rate, ...).

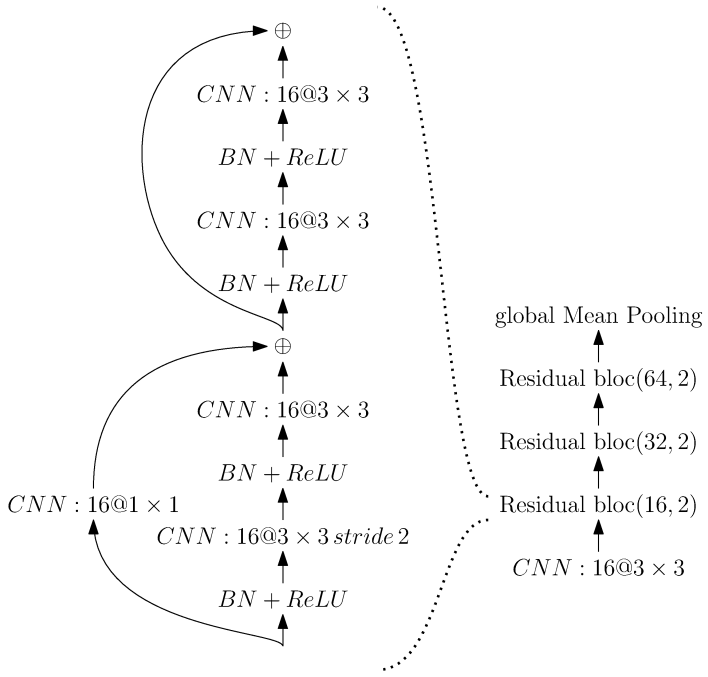


Figure 3.11 – Wide ResNet Neural Network in 16-1 configuration

3.3.2 Hybrid NN-HMM

This model is composed of two parts, the posterior of the states and the model of the transitions between them, which corresponds to the left path in Figure 3.9. Since we consider the first Neural Network layers as part of a separate Representation Learning module, the state posterior model is conceptually reduced to a single dense layer with a softmax non-linearity However, this point of view should not occult the fact that the whole stack of layers is trained altogether, and the separation between the two stages is only meant to help understand the role played by each part. As for the transition model, the state of the art standard organization as presented in section 2.1.4 is adopted; more precisely, our model uses the transition model presented in Figure 3.12 with:

1. 5 states by gesture class
2. linear transition structure between the states
3. 1-state skip transitions (Bakis Model)
4. a loop-back transition from fourth to the second state to account for repetitions within a gesture.

3.3.3 Bidirectional Recurrent Neural Network with Gated Recurrent Units

To oppose the hybrid NN-HMM model, a single layer of Bidirectional Recurrent Neural Networks is chosen, with 172 Gated Recurrent Units (GRU) as shown in the right path of the full model architecture shown in Figure 3.9

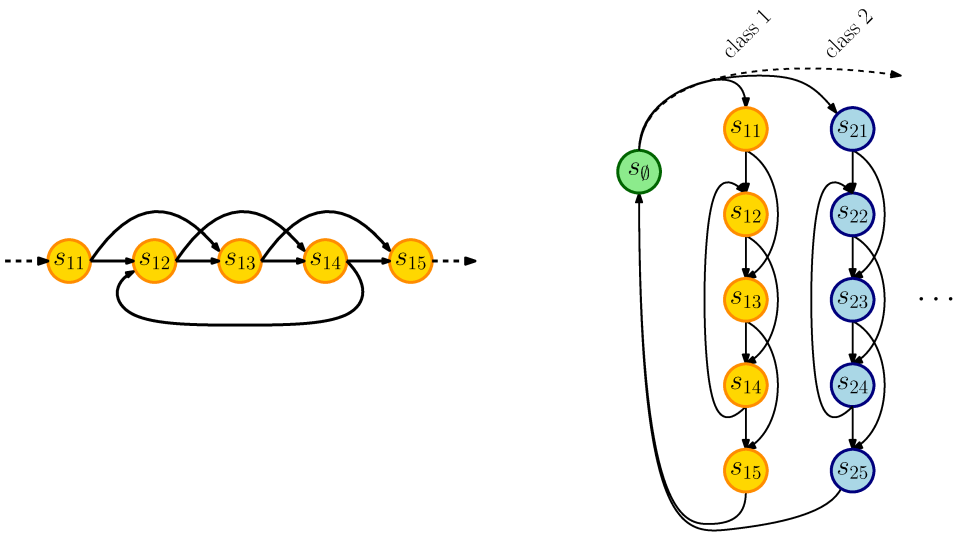


Figure 3.12 – HMM State transitions model

The selected model might seem relatively small compared to other examples from the literature, but this choice is motivated by several arguments:

- the single layer architecture[1] mimicks the forward-backward algorithm used for inference on the HMM in a Neural Network way. The comparison will therefore show how a target driven Neural Network compares with a structured handcrafted probabilitic model with exact inference.
- the computation cost is in the same order of magnitude since the forward-backward pass requires 2 * 101 steps vs 2 * 172 steps for the RNN. Both the number of parameters and operations are higher on the RNN but this is compensated during predictions at runtime by the high level of optimization of Neural Network computations on GPUs devices. Inversely, we do not want to size up the HMM with additional states as we have found no evidence that it would improve performances. In fact, adding more states would divide the available data further between the states introducing a risk of over-fitting the state posterior Neural Network model.
- GRU units are obviously preferred over a vanilla RNN implementation since their gating mechanism allows a more efficient training and learning of long range events. In the Montalbano v2 dataset case, the range between related time-steps is reckoned to follow the durations of gestures, that is to say around 40 steps. Compared to the more widespread LSTM units, the difference in terms of learning quality is often found negligible (at least for the purpose of our comparative study), yet GRUs are slighlty faster to train hence the choice for these units here.

The final part of the classification process is carried out by a single dense layer with a softmax non-linearity which transforms the hidden states of the two RNNs into (20 + 1)-class prediction

[1] since the two opposing RNNs are not stacked on top of each other, there is no composition of successive layers, hence the single layer qualification.

vectors. The modularized design of the models and the experiments led to the decision of not using temporal convolution at this point since it would off-load an unknown portion of the temporal modelling work from the BD-RNN module. As a result, this layer is simply repeated across time and only takes the hidden states of the BD-RNN at one time step to issue the final prediction.

3.4 Training Set-up and Parameters

For most experiments, the training procedure does not differ much from the standards observed in the literature. There is however one aspect which rarely receives more than little attention in publications but was deemed to have a fairly noticeable impact in our experiments nevertheless: managing class imbalance.

To get an intuition about why class imbalance might affect the training quality, Figure 3.13 shows the count of frames by class in the dataset: the dataset used here contains a predominant non-gesture class with an order of magnitude more samples than the other classes. The next section explains how this problem is dealt with in our experiments. Then, several sections provide detailed descriptions of our training procedures for completeness and reproducibility.

3.4.1 Training with imbalanced classes using loss re-weighting

Neural Network based classifiers such as the state posterior model or the RNN used in this work usually do not cope very well with imbalanced data. In practice, a Neural Network trained naively will be overloaded by samples from the a-priori most likely class ans simply predict "$\emptyset$" most of the time; a behaviour that satisfies the accuracy or the cross-entropy objectives, but will fail to fulfil related metrics such as the Jaccard Index. In particular, the existence of a distinct null class in our problem introduces numerous types of errors unseen on regular classification problems:

- false negative: a gesture frame is classified as non-gesture
- false positive: a non-gesture is mistakenly classified as a gesture
- misclassification: a gesture is classified as another one
- a gesture can be improperly delimited in time or completely missed

Some use-cases will require to penalize some errors more than others but training is often performed based on the cross-entropy loss function anyway because it is differentiable and numerically stable. With $(y_t)_{1\leq t\leq T}$ a sequence of T target labels in a vocabulary of V classes, $(\tilde{y}_t)_{1\leq t\leq T}$ a sequence of predicted class probability vectors, the cross entropy loss is given by:

$$L_{CE}(y,\tilde{y}) = \sum_{t=1}^{T}\sum_{j=1}^{\mathrm{V}} -\mathbb{1}_{y_t=j}\log(\tilde{y}_{t,j}) \tag{3.1}$$

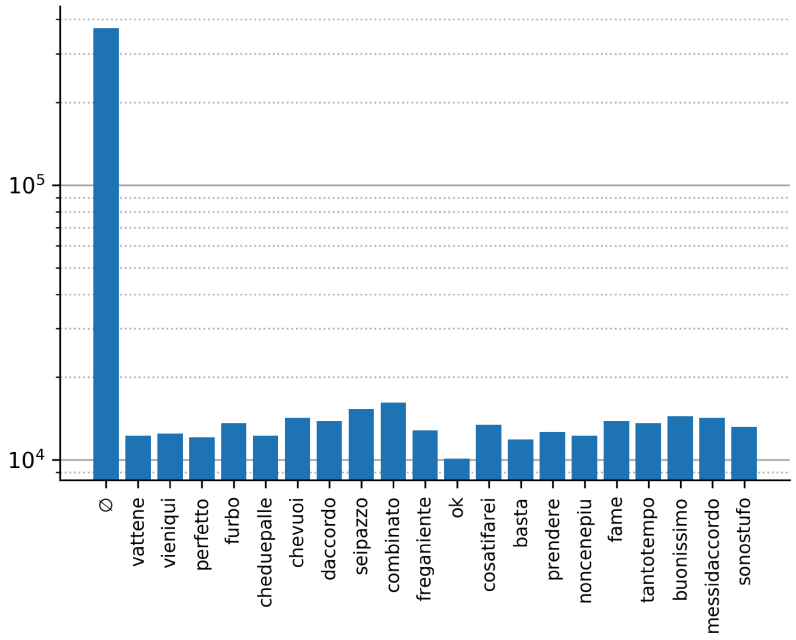


Figure 3.13 – Number of frame occurrences by class in the Montalbano v2 dataset

To artificially control the effect of imbalance, multiple papers propose to sub-sample or over samples the training dataset. Unfortunately, this technique is hardly manageable for the RNN which takes whole sequences with mixed labels in it.

Instead, since most loss functions (including the cross entropy used in this work) are written as a sums over the losses at individual time-steps, one can selectively put a coefficient on each term depending on the target label. This correction is chosen as the number of classes times the inverse frequency of the class smoothed by a hyper-parameter α to avoid over-correction. Accordingly, the cross entropy loss function becomes:

$$L'_{CE}(y,\tilde{y}) = \sum_{t=1}^{T}\sum_{j=1}^{\mathrm{V}} -\frac{1}{\mathrm{V}\cdot f_j^{\alpha}}\,\mathbb{1}_{y_t=j}\log(\tilde{y}_{t,j}) \qquad \alpha \in [0,1] \tag{3.2}$$

where f_j is the frequency of class j in the training set.

Let X, Y designate the samples as random variables of the true data distribution, $\tilde{Y} = Net(X)$ the prediction from the Neural Network. The expectation of the re-weighted loss with $\alpha = 1$ is equivalent to that of re-sampling the dataset with a uniform class distribution (time steps

omitted for clarity):

$$\begin{aligned} E_{(X,Y)}\left[L'_{CE}(y,\tilde{y})\right] &= E_{Y,X|Y}\left[\sum_{j=1}^{\mathrm{V}} -\frac{1}{\mathrm{V}\cdot f_j}\cdot \mathbb{1}_{y=j}\cdot\log(\tilde{y}_j)\right] \\ &= \sum_{j'=1}^{\mathrm{V}} p(Y=j')\cdot E_{X|Y=j'}\left[-\frac{1}{\mathrm{V}\cdot p(y=j)}\cdot\mathbb{1}_{y=j}\cdot\log(\tilde{y}_j)\right] \\ &= \sum_{j'=1}^{\mathrm{V}} \frac{1}{\mathrm{V}}\cdot E_{X|Y=j'}\left[-\mathbb{1}_{y=j}\cdot\log(\tilde{y}_j)\right] \\ &= E_{X,Y\sim\mathcal{U}(\mathrm{V})}\left[L_{CE}(y,\tilde{y})\right] \end{aligned} \tag{3.3}$$

Yet, contrary to re-sampling, the selection of training sequences is unconstrained and accepts sequences with any labels in any proportion.

Hybrid NN-HMM meta-parameters

The training procedure follows algorithm 3.1. For the neural network, the accelerated ADAM

1: assign state labels uniformly within annotated gesture boundaries (Figure 2.2)
2: initialize Neural Network with random parameters, HMM forward and skip transitions probabilities with 0.15 and 0.05 respectively.
3: $i := 0$
4: **repeat**
5: fit state posterior model for 20 epochs ▷ 7 suffice for $i \geq 2$
6: fit transition probabilities until convergence
7: realign state labels within annotated gesture boundaries ▷ Viterbi
8: estimate state priors
9: **if** $i = 0$ **then**
10: reset Neural Network with random parameters
▷ *this step is not mandatory but considering that the first realignment step may substantially alter the targets, a full retraining from scratch is used as a safety measure*
11: **end if**
12: **until** convergence of target state assignment

Algorithm 3.1 – Detailed training procedure of the Hybrid NN-HMM model

gradient (defined in Section 2.3) is used with a base learning rate at 0.001 decreased by a factor 0.3 everytime the loss begins to stall. This heuristic offers a compromise between fast convergence during an initial phase and finer adjustment later during the training iterations, and thus reduces the overall number of training iterations.

Note that due to the large imbalance in the data, the re-weighting trick introduced in the previous section is used when fitting the state posterior model, in this case with a smoothing factor $\alpha = 0.7$ established through cross validation.

In parallel, the division by priors $p(s_t)$ in the expression of the Hybrid NN-HMM joint distribution causes problems due to the extreme discrepancy between the prior of the gesture classes

and the null one: $\forall l \neq \emptyset, p(s = l) \ll p(s = \emptyset)$. The neural network state posterior is never sufficiently confident to compensate for this term, even more so after introducing the (necessary) loss re-weighting. The measurable consequence shows up in the form of a collapsed likelihood ($\frac{p(s_t|\boldsymbol{x}_t)p(\boldsymbol{x}_t)}{p(s_t)}$) for the null class as illustrated in Figure 3.14. Indeed, the neural network based state posterior model is never sufficiently confident and underestimates $p(s_t|\boldsymbol{x}_t)$ which decreases abruptly the likelihood due to the large prior $p(s_t = \emptyset)$. In fact, the division by the priors which is supposed to un-bias the model is largely redundant and over-correcting the work done by loss re-weighting. As a result, the priors are simply assumed to be uniform in our version of the Hybrid NN-HMM, and the neural network implements what we would call a pseudo-likelihood.

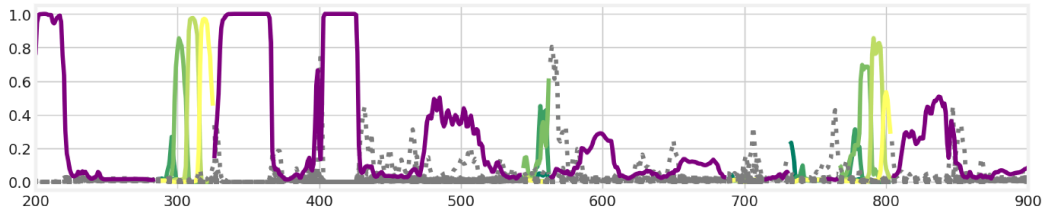


Figure 3.14 – Pseudo-likelihood of gesture observations over time. Only the curves associated to the actual annotated gesture are coloured at each time-step, and they should be above the others for a correct prediction. The purple one which stands for the null class often falls below others.

Bidirectional Recurrent Neural Network

The BDRNN cannot take the full length sequences as input (at least for the version that takes images in input). As a result the sequences are chunked into overlapping pieces of length up to 128 with some warmup frames provisioned at the beginning and the end of segments to eliminate edge effects. This chunking is also used during evaluation, and we were not able to exhibit any significant degradation due to the edge effect, so we assumed the number of warm-up frames was sufficient.

Again, the training procedure uses the ADAM accelerated gradient with an initial learning rate of 0.001 reduced by a factor 0.3 every time the loss begins to stall.

Post-Processing of Abnormal Detections

Before computing the evaluation metric on the predicted sequences of labels, a simple filter is run to eliminate spurious very short or overly long detections. Indeed, the objective function we use to train the Neural Networks operates in a frame-wise fashion and thus fails to prevent detections with improbable durations, either very short or very long. Figure 3.15 anticipates on the experimental sections by showing a sample prediction output from the BDRNN model where very short spiky detections are clearly visible. For evaluation metrics that take into account the number and order of the detections such as the Levenstein distance [Levenshtein, 1966] or the

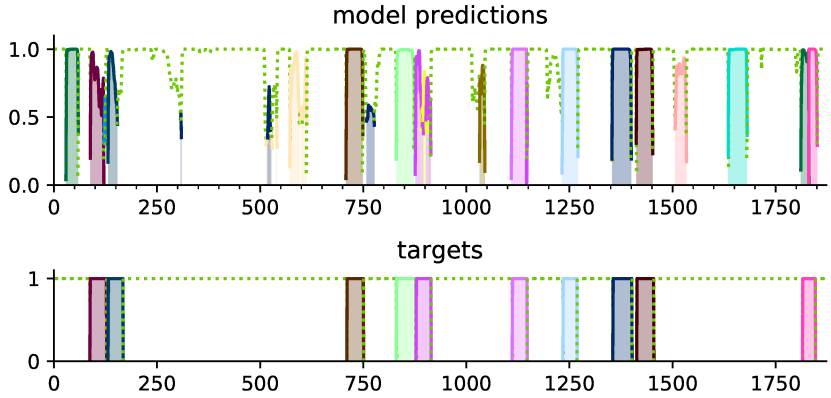


Figure 3.15 – Sample predictions from the BDRNN model on body pose data. At each time-step, only the class with the highest score (the prediction) is plotted. Gesture classes use distinct colours and the non-gesture category uses a green dashed line.

Jaccard Index [Jaccard, 1902], these noisy predictions can prove particularly harmful because they introduce whole new instances in the sequence of detected gestures regardless of how short they are.

As shown in Figure 3.16, the accuracy of the RNN on very short or very long gesture predictions is lower than simply defaulting to no detection at all. Consequently, we have introduced a post-processing step where these predictions are simply discarded and replaced by the non-gesture class. The lower and upper limits on the frame count by gesture have been cross-validated

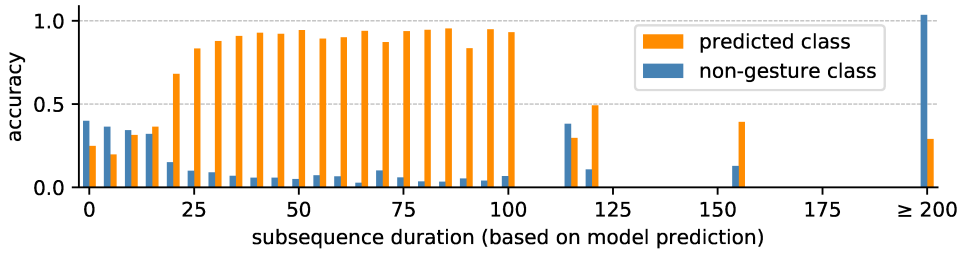


Figure 3.16 – Accuracy during gestures discriminated by the duration of the detection.

to approximately 20 and 300 respectively.

Despite its apparent simplicity, this transformation noticeably improves the JI metric on the RNN-based model which tends to produce slightly noisier outputs overall. As expected, the HMM transition model is more resilient to this phenomenon since the self transitions of the states implicitly encode into the model the expected duration of each states and by extension of each gesture. Still, the results were marginally but consistently improved by this filtering across experiments, so the filtering step was kept for the Hybrid NN-HMM model as well. To further improve the filtering quality, one may want to use a more elaborate smoothing method such as using a combination of an RNN followed by an HMM as used in [Koller et al., 2017], but this

goes beyond the scope of this work and would intermix the two models under analysis, which is undesirable.

3.4.2 Details on Design and Implementation

To minimize interferences with the comparison of the models, we have shown that our models use shared representation learning layers. With the same intent, we use the same loss function, learning rate and learning rate decay strategy for the training of the state posterior Neural Network and BD-RNN models. More generally, the meta-parameters including the Neural Network architectures have been selected to simultaneously maximize the performances for both models. This constraint made it difficult to directly reuse existing model parts from the literature, either from Gesture recognition or other related subjects. For example, we noted that the Representation learning layers trained via the Hybrid NN-MMM set-up were more prone to overfitting than they were in the BD-RNN model.

To optimize the numerous meta-parameters that define our models architecture (number or type of layers, number neurons, activation functions, learning rate, etc.), a greedy cross-validation strategy was used: starting from a sub-optimal Hybrid NN-HMM and RNN, settings have evolved to the current configuration by iteratively selecting one of the most influential parameters and cross-validating its optimal value. This is the only conceivable strategy we found considering the combinatorial complexity of this optimization problem, and the heterogeneity of settings to analyse.

Although our models have been redesigned for the purpose of our experiments, other publications have served as a source of inspiration as mentioned in the previous sections, with some notable similarities and differences along the prediction pipeline:

1. Our body pose features are similar to the ones used in [Neverova et al., 2014; Seddik et al., 2014; Wu et al., 2016], with pairwise joint distances in particular. We applied a few simplifications compared to what was reported in the first two papers. The overall body size was normalized, but not the limbs lengths as we feared it could distort the pose (cross hands) and worsen pose regression errors. We found the Azimuth and Bending angles to describe the limb orientation counter-intuitive, and replaced them with coordinates of the vector normal to the plane formed by two consecutive limbs (noted $\vec{n}$ in Figure 3.7). This three-dimensional representation matches the one of positions and pairwise differences but also removes issues with angle continuity between 0 and 2π.
   Because we restrict the computation of features to the upper body parts joints, the dimension of our final feature vector is only 248 instead of 860 for [Neverova et al., 2014].
2. For the video module, [Pigou et al., 2016] uses a large rough crop of the whole section of the body on each side, which relaxes the need for a fine body pose detection. Like [Neverova et al., 2014], we normalize the apparent distance but use a local contrast normalization algorithm instead of normalizing the mean and standard deviation on the cropped images, as the latter behaves differently depending on the distribution of colours in the clothes and

background more than the hands. Unlike both [Neverova et al., 2014] and [Wu et al., 2016], we do not detect the dominant hand but rather augment the dataset to generate additional left-handed samples. Indeed, a recognition system in real-life conditions would lack access to this information which requires accumulating movement statistics over several gestures. Besides, we do not think the subjects should be constrained to consistently use their dominant hand. In an attempt to minimize the computational cost of our model, our crop size is set to 32 instead of 72×72 in [Neverova et al., 2014] and 64×64 [Wu et al., 2016]. This also matches the input size Residual Convolutional Networks were designed for. Finally, we restrict our input to the colour images converted to grey-scale and ignore the depth unlike other publications. We are convinced that adding this modality would not change the conclusions of our studies except for raw performance values, and discarding it saved a considerable amount of computation time which was better invested into the comparative analyses.

3. Although raw performances were, indeed, a secondary concern for our comparative study, we sought to validate the correctness of our model by trying to achieve the best possible performances within the conditions imposed by our experiments. To do so, our models combine several ideas or innovations. While these reused model parts have been discussed previously, here is a summary for convenience: the temporal convolution was suggested and tested by [Pigou et al., 2016], the multi-modal fusion strategy was proposed in [Neverova et al., 2014], and the Hybrid NN-HMM model is similar the one in [Wu et al., 2016], but also simply follows the standard transition model established over the last decade [Pisoni, 1988]. Finally our models make use of various innovations such as Rectified Linear unit activations, Batch Normalization, Dropout, Residual Units, Gated Recurrent units and ADAM accelerated gradient.

For most parts, the code of our experiments has been written from scratch in Python: data preprocessing, data sampling and loading, model architectures, some specific model parts such as Temporal Convolution, the loss functions, and training routines with the Hybrid NN-HMM procedure in particular. However, a lot of credit should go to several supporting libraries, all open-source and publicly accessible, that provide optimized implementations of various elementary algorithms:

- OpenCV [Bradski, 2000] for some of the image preprocessing operations
- Theano [Theano Development Team, 2016] for fast GPU execution and automatic differentiation of computation graphs.
- Lasagne [Dieleman et al., 2015] for the implementation of Neural Networks layers, accelerated gradients, and regularization operations.
- Pomegranate [Schreiber, 2018] for the implementation of HMM inference and training algorithms.
- Scikit-learn [Pedregosa et al., 2011] for various machine learning algorithms.

The manipulation of variable length sequences with high throughput requirements during preprocessing, training, and inference, has proved to be particularly challenging. Indeed, the dataset could not fit into memory so on-demand data loading, chunking, transformation and recomposition was needed. Most of this logic has been externalized into a separate library called *SeqTools*[1] which will be maintained for reuse in future projects.

All the scripts were executed on a standard desktop computer equipped with an NVIDIA 1080Ti GPU, an Intel Xeon E5-1620v3 CPU, and the operating system ran a Linux kernel. On this setup, the longest experiments took no more than four days.

3.5 Study 1: End-to-end Learning

In this section the models are simply trained "from scratch" according to the procedures described previously without any modifications. This experiment serves three purposes: 1 – compare the ability of the models for end-to-end learning, down to the input layers, 2 – establish a baseline to build on for the following experiments and 3 – verify that the performances on our implementation of the models is on par with the results reported by other publications.

To evaluate the model, the accuracy metric (Equation 3.4) gives an intuitive idea of the model performances in terms of frame-wise correctness. However, it makes no distinction between the different gesture classes and the non-gesture class. The reported score is computed by taking the mean accuracy of the testing sequences, which is given by:

$$Acc(y,\tilde{y}) = \frac{1}{T}\sum_{t=1}^{T} \mathbb{1}(y_t = \tilde{y_t}) \tag{3.4}$$

where $(y_t)_{1\leq t\leq T}$ and $(\tilde{y})_{1\leq t\leq T}$ contain the annotated and predicted labels for a sequence of T time-steps.

Besides accuracy, the Jaccard index (JI, Equation 3.7) is also reported as it was originally the scoring method selected for the Chalearn 2014 competition for which the Montalbano v2 dataset was assembled. Contrary to the edit distance which is popular in sequence recognition, this metric not only penalizes classification errors but also mismatched alignments between detections and targets; its value for one sequence is determined by the following expressions:

$$\mathcal{V} = \{y\} \cup \{\tilde{y}\} \setminus \{\emptyset\} \qquad \triangleright \text{ set of active classes} \tag{3.5}$$

$$\forall_{l\in\mathcal{V}} \quad JI_l(y,\tilde{y}) = \frac{\sum_t \mathbb{1}(y_t = l)\cdot\mathbb{1}(\tilde{y}_t = l)}{\sum_t \max(\mathbb{1}(y_t = l), \mathbb{1}(\tilde{y}_t = l))} \tag{3.6}$$

$$JI(y,\tilde{y}) = \frac{1}{|\mathcal{V}|}\sum_{l\in\mathcal{V}} JI_l(y,\tilde{y}) \tag{3.7}$$

where $|\bullet|$ is the cardinal function, $\{\bullet\}$ designates the set of uniquely distinct elements from an ensemble. The final reported score is the empirical estimation of the Jaccard Index on all the

[1] currently hosted at https://github.com/nlgranger/SeqTools

sequences of the dataset. This averaging notably discards any variation of length or difficulty between sequences.

Both metrics are evaluated and averaged over the whole training, validation or testing dataset to provide the final metric. Following the rules of the original competition, the length of the sequences is not taken into account in the computation of the average.

3.5.1 Recognition Based on Body Pose Features

Table 3.1 summarizes the results for the Hybrid NN-HMM and the BD-RNN model using the body pose features only. These inputs lie in a relatively low dimension space (248) and are known to carry a lot of information about the gestures to detect. This is therefore the simplest experiment to run at first. The proximity with state-of-the-art results demonstrates that the restrictions to use a common preprocessing pipeline and representation learning module has not impacted the performances. On the contrary, both models achieve state-of-the-art performances and the Hybrid NN-HMM only leaves a slight edge to the RNN on this experiment.

Model	**Accuracy**		**JI**	
	validation	test	validation	test
Hybrid NN-HMM (no filtering)	0.908		0.778	
BDRNN (no filtering)	0.919		0.787	
[Wu et al., 2016] Hybrid NN-HMM			0.783	0.779
Hybrid NN-HMM	0.911	0.912	0.789	0.788
BDRNN	0.921	0.922	0.814	0.811
[Neverova et al., 2016] DNN				0.831

Table 3.1 – Accuracy and Jaccard Index metrics with body pose features

As mentioned previously, the post-processing step that filters out gesture detections of abnormal durations brings notable improvements to the JI score of the BDRNN. Our understanding on this issue focuses on the architecture and training algorithms which probably fail to enforce intuitive notions of compositionality between concepts as a human would expect. In this case, the RNN does not naturally capture elementary notions such as decoupling the temporal extension from the signification of the gestures. Some multilayer Neural Network may "happen" to provide interpretable behaviours (characteristic filters, responses to patterns, etc.), but our experiment shows that architectural constraints such as a recurrent formulation or a temporal convolution can be overridden in order to maximize the objective function.

Although the end-results are quite close after post-processing, the global metrics hide a difference in behaviour between the models which can be exhibited by comparison of the confusion matrices in Figure 3.17. The difference between the confusion matrices is plotted as a heatmap so that a blue shade at coordinates (i, j) indicates that more observations with label $y = i$ were classified as $\hat{y} = j$ by the Hybrid NN-HMM than by the BDRNN, and inversely for the red

colours. The confusion matrices were normalized row-wise so that each element (i, j) contains the empirical estimation of the error probability $p(\hat{y} = j | y = i)$, hence the small scale of the values observed in the figure. The first row and column correspond to the non-gesture class $\emptyset$ which understandably does not share the same error patterns as the remaining classes. The Hybrid NN-HMM actually generates 1.5% fewer misclassification errors between the ges-

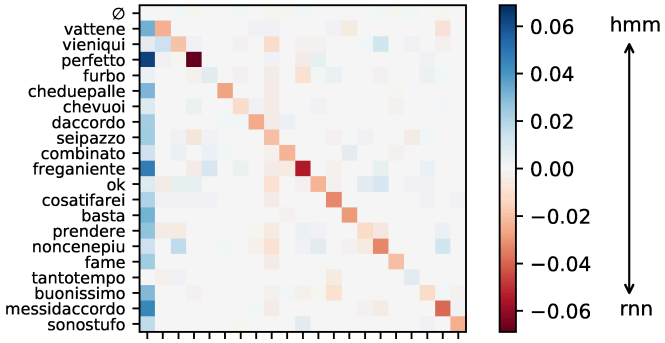


Figure 3.17 – Difference between the confusion matrices of the Hybrid NN-HMM and the BD-RNN models on the body pose modality experiment. Prior to the computation of the difference, a row-wise normalization is applied to each row of the confusion matrices in order to eliminate prior class imbalance from the visualization.

ture classes as denoted by the slight red dominance outside de diagonal, but more often fails to detect the gestures at all (blue dominance in the first column). Inversely, the row-wise normalization in the confusion matrix should not occlude that the slight advantage of the BDRNN on the non-gesture class (first row) concerns *many* frames, as explained in Section 3.4.1. Any effort exercised to rebalance this shortcoming of the Hybrid NN-HMM resulted in greater deterioration from the other error types, namely non classes detected as gestures or misclassification between gestures. The selection of hyper-parameters we have presented and in particular the reweighing of the loss function correspond to the best compromise we could find for that issue.

As a conclusive remark, it should be noted that the RNN does not systematically beat the Hybrid NN-HMM on all classes, as indicated by blue diagonal elements in Figure 3.17, suggesting a possible complementarity between the models.

3.5.2 Recognition Using Video Features

To stress the ability to learn the bottom layers even further, another experiment is run with the cropped hand images which contain high dimensional data of low-level features (pixel intensity). Apart from considering a Convolutional Neural Network for the image inputs, the rest of the models and training procedures are left unchanged.

In terms of complexity, the CNN contains $3.4e5$ parameters, which is fairly compact for a computer vision model, but stacks 17 layers instead of respectively $4.7e5$ parameters and 3 representation learning layers for the body part (Figures 3.10a and 3.10b). In conclusion the

CNN might prove to be much more complex to train and should uncover any shortcoming in the end-to-end learning capacity of each model.

As reported in Table 3.2, the gap between the Hybrid NN-HMM and the RNN is not negligible on both evaluation metrics under this more complex architecture. The analysis of the errors do not reveal any particular point of failure but rather an overall performance gap.

Model	Accuracy		JI	
	validation	test	validation	test
Hybrid NN-HMM	0.890	0.890	0.748	0.745
BDRNN	0.909	0.909	0.791	0.789

Table 3.2 – Accuracy and Jaccard Index metrics with hand crops video input

3.5.3 Analysis of Models and Interpretation of Prediction Errors

To understand better our models, we analyse into more details the values of their parameters and the most frequent error types to propose an interpretation thereof. Overall, both models present the same kind of errors, so we do not necessarily provide the details for both models unless the distinction contributes to the analysis.

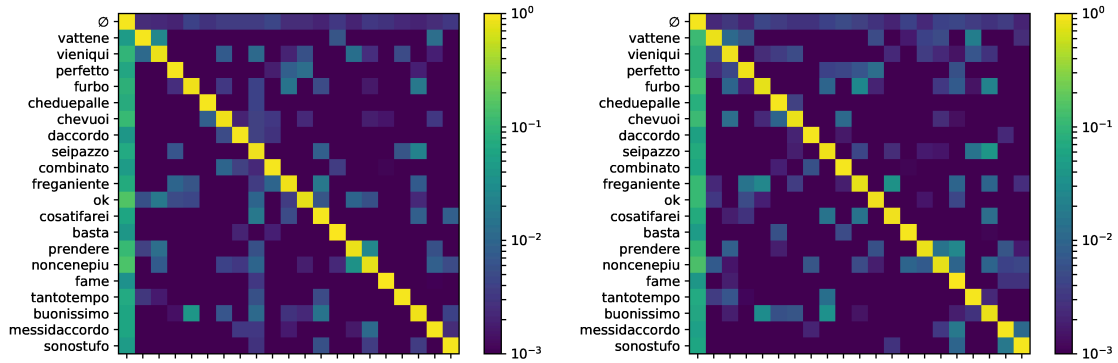
Error Types and Statistics

For the RNN model on body pose data, the classification errors are divided into:

- 47% non-gesture frames classified as gestures. This category includes early detections, late terminations or spurious detection of non-gestures segments that might contain movements outside of the gesture vocabulary.
- 25% gesture frames into non-gesture. For example late detections, early terminations or complete failures to detect any gesture at all.
- 28% gesture frames confused with other gestures, that is to say that a gesture was detected but incorrectly classified.

As mentioned previously, the balance between the number of errors for gestures and non-gesture frames is largely controlled by the α parameters of our reweighing scheme in Equation 3.2. Its value was cross validated to maximize the Jaccard Index metrics and set to 0.7 for the RNN and slightly lower at 0.6 for the HMM since the latter faces more class imbalance with 100 gesture states instead of 20 gesture classes.

By looking into more details at the confusion matrix in Figure 3.18, we observe that some classes are more problematic on both input modalities. In fact, these mistakes are largely understandable considering the similarity between some pairs of classes in the data itself as shown in Figure 3.19: some gestures have both similar pose and hand shapes.



(a) Body pose data **(b)** Hand crops inputs

Figure 3.18 – Confusion matrices for BDRNN model

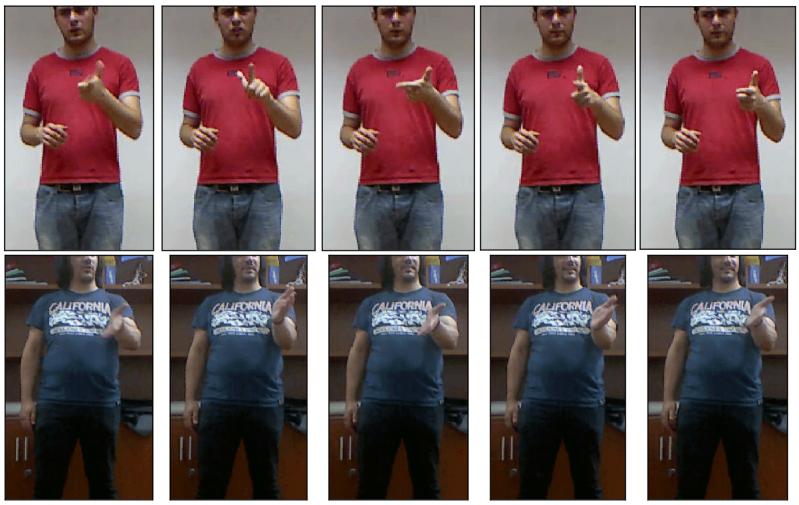
Due to the choice of the Jaccard Index as specified in the Charlearn competition, the detection of spurious gestures which do not exist in the sequence is particularly harmful for the score as it increases by one the denominator in its expression (Equation 3.7). Section 3.4.1 already mentioned how this property led us to specifically introduce a post-processing filtering step on our model predictions. In a language related problem, generating out-of-context words is obviously problematic, but we consider targeting this type of error as a shortcoming of the evaluation metric in the context of gestures sequences, where class instances are drawn randomly and not according to some syntax; if a language model existed, it could be incorporated into the continuous recognition system and limit these spurious detections. We nonetheless complied to the rules of the original competition in order to compare our work with prior contributions. On average, each prediction sequence contains 0.74 spurious classes, and this number drops down to 0.25 after automatic elimination of the overly short or long detections.

The spread out distribution of the performance metric across sequences (Figure 3.20) reveals that some sequences are more problematic as a whole, leading to very low sequence-wise Jaccard Indexes, a phenomenon which we explain by missing pose detections due to suboptimal recording conditions or peculiarities such as a non-standard resting position with arms crossed or hands in the back. Finally, some recordings contain non-annotated movements such as using a hand to move the hair or the clothes, both of which are difficult to distinguish from the actual gesture classes. These events remain rare according to our observations and the model obviously fails to generalize on them. Training specifically to manage these situations would likely bear little benefits considering the rarity and oddity of each occurrence.

Local Origins of Errors

In the previous section, the analysis was focused on the different error types regarding the classes and labels. We now track the origin of these errors at a finer scale and try to correlate them to known quantities.

Since a large proportion of errors imply the non-gesture class, we assumed that part of the explanation could be found in alignment issues at the beginning and the end of the gesture. Yet



(a) noncenepiu vs prendere

(b) buonissimo vs furbo

(c) fame vs sonostufo

Figure 3.19 – Pairs of samples from easily confused classes

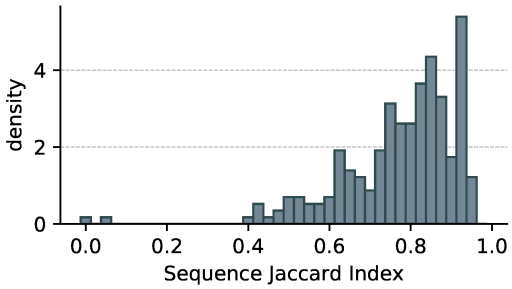


Figure 3.20 – Histogram of *sequence-wise* Jaccard Index scores, the spread-out distribution indicates recognition difficulty also results from the properties of a whole recording and not only of local gesture-wide phenomena.

most of the time, a correct classification goes along with a nearly perfect alignment, although Figure 3.21 shows a slight tendency to anticipate and respectively delay the beginning and the end of the gestures which is common to both of our models.

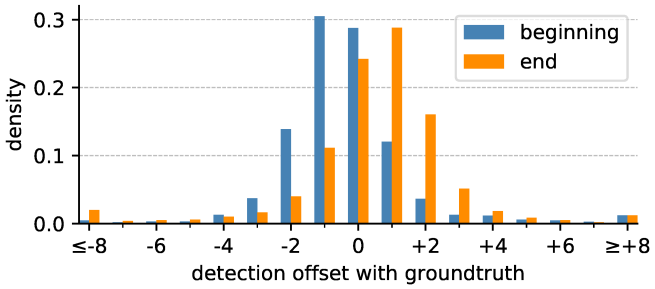


Figure 3.21 – Distribution of delays between predicted and annotated gesture boundaries. Alignment quality is assumed to require a proper classification in the first place, therefore detections matching less than 50% of the annotated frames are eliminated to avoid disrupting the statistics with outliers.

Figure 3.21 does not provide insights about the reason of these imprecise alignments. By studying the position of errors and distinguishing the gestures by duration instead of label, we gain more insights about the differences between the Hybrid NN-HMM and the BR-RNN models. Figure 3.22 shows a 2D histogram plotted as a heat map of the accuracy against the duration of the annotated gestures and the frames within these gestures. To facilitate the interpretation, time (or frames numbers) is normalized between 0 and 1 with 0 the beginning the gestures according to the annotation and 1 the end. The BD-RNN suffers more difficulties on the first rows than the HMM model, which means this model struggles more to detect short gestures as shown by the upper heat-map in Figure 3.22. On the opposite, the Hybrid NN-HMM model sees its performances gradually dropping as the duration of the gestures increases, in particular on the edges of the subsequence as shown by the darker upper corners. Given the previous analyses on the error types, we believe the state posterior model is never sufficiently confident to prevent the Viterbi algorithm from switching too quickly to the non-gesture state, a problem

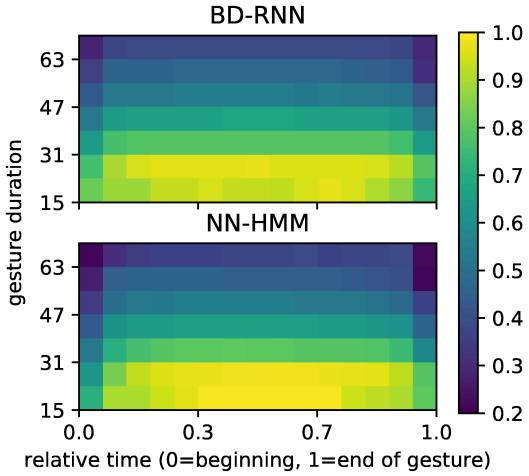


Figure 3.22 – Colour coded accuracy within annotated gesture boundaries. Horizontal axis represents execution progress, vertical bins aggregate gestures by total duration.

aggravated by the fact that transitioning poses often look like the resting position. As the duration of gestures increases, the number of affected frames grows leading to the observed issue: detections start too late and terminate too early as confirmed by the distribution of boundary delays for long sequences in Figure 3.23.

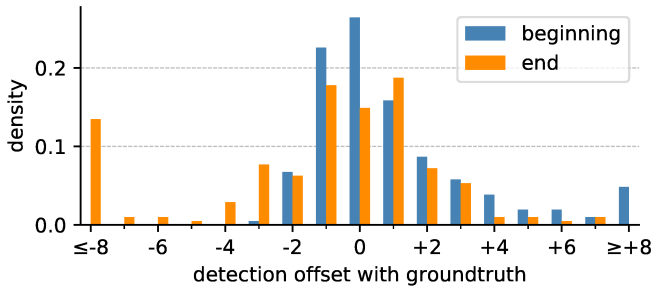


Figure 3.23 – Distribution of delays between predicted and annotated gesture boundaries *for gestures longer than 55 time-steps*.

HMM Transition Model

Diving into deeper details, we analyse the HMM transition model (Figure 3.12) and the learnt transition probabilities. On body pose data, the HMM does not exploit the backward transitions but does use the skip ones as denoted by the zero values below the diagonal and the non-zero values on the second upper diagonal of the transition matrices from Figure 3.24. Because the patterns modelled by some states do not necessarily appear in all instances of a gesture, these transitions allow the HMM to follow a path that skips these states. The backward transitions seem to be used when the model trains on hand cropped images. We therefore suspect that repetitive patterns such as hand waving are too fine to be observed in the body-pose data, which

admittedly suffers from detection noise. In both cases, the transition model does not present

(a) Body pose inputs.

(b) Hand crops inputs.

Figure 3.24 – HMM State transition probabilities. Element (i, j) encodes $p(s_t = j | s_{t-1} = i)$. The diagonal therefore contains the self transitions, the upper triangle the forward transitions and the lower triangle the reverse ones. The last column $j = 6$ contains transitions probabilities toward the non-gesture class $\emptyset$.

strong variations between states or gestures, a sign that all states tend to capture the same number of frames on average. The equal role played by all states might find its roots in the re-weighting mechanism of the loss function that we introduced while solving the class balance issue. It might forbid the posterior model from completely ignoring a state and therefore prevents any state from ever disappearing during the realignment steps of the training process. With the hand crops as inputs, the transitions show a greater diversity of values with the loop-back transition probability often taking small yet non-negligible values, which indicates that the model might follow these paths to model repetitions of structured patterns inside gestures.

The ability of the state posterior model to learn all 101 states without much ambiguity is demonstrated by its confusion matrix which presents a remarkably low error rate on the validation set (Figure 3.25). In an ablation experiment, the state classifier without transition model demonstrated that it could classify gestures by looking at the label associated with the most likely state at each time step; in that configuration, the ablated model achieves an accuracy of 0.873 and a Jaccard index of 0.705 on the body pose validation sequence. Although the score might

seem high considering the lack of a proper temporal model, the gap to reach the performances of the full model cannot be filled easily. Indeed, even with a temporal convolution raised to 29 time-steps, the model only achieves a Jaccard Index of 0.754, which implies that the role of the transition model cannot be trivially emulated by a feed-forward Neural Network with a lot of temporal context. In [Neverova et al., 2014] for example, the lack of a recurrent temporal model is compensated by a non-trivial combination of temporal convolutions at multiple time-scales.

The first and final states of each gesture often model a generic transition with the core, and thus are understandably confusable between gesture classes; in the data itself, these transitional moments are not very precise both in execution and annotations. On the state confusion matrix in Figure 3.25 these errors are denoted by non-zero values outsize the diagonal boxes which regroup states by gesture classes. More generally, when two gestures contain easily mistaken patterns encoded by some states, the HMM can still disambiguate the gesture label using the likelihood of the whole succession of states since transitions between partially executed gestures are forbidden. Obviously, this filtering finds its limits on completely similar gestures. These problematic situations are visible on the confusion matrix when a whole diagonal of 5 states is active inside the square block corresponding to another gesture.

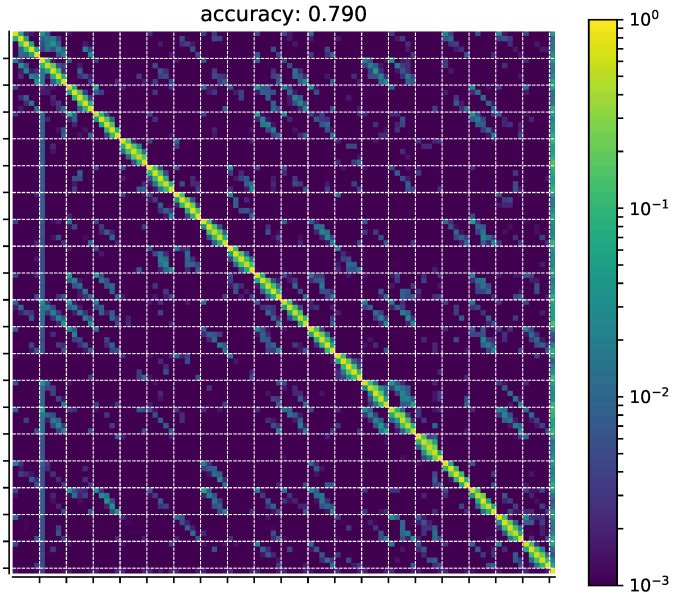


Figure 3.25 – State posterior confusion matrix. Each element (i, j) encodes the probability of having the posterior state model predicting the state j for an item labelled i (row-wise normalised confusion matrix). The grid encloses groups of states corresponding to each gesture and the non-gesture state is placed at the end on index 101.

3.5.4 Recognition Using Multi-Modal Inputs

To further expand the complexity of the model and exploit more fully the available data, an intermediate fusion experiment is run that uses both body pose and video images of the hands

simultaneously. Basically, both inputs are read simultaneously and the embeddings h_t^{pose} and h_t^{video} from Figure 3.10, are concatenated to form a joint representation vector. The remaining parts of the models are left unchanged.

Since the video image part is much slower to train and certainly much more complicated, a competition between the modalities may occur that would lead to under-fitting the CNN and investing all the training effort on the body-pose side of the model. To avoid this issue, the Modrop technique introduced by [Neverova et al., 2014] is used: it is a variation of the Dropout technique that completely drops the whole feature vector for one modality so that the model is forced to deal with the remaining ones. This method was selected because it preserves the architecture of our models and we do not believe the Modrop technique will impact differently the Hybrid NN-HMM or the BDRNN model. In our case, the body pose features are dropped 30% of the time whereas the rate is set to 10% only for the hands images, due to the high computational cost of training the whole model, these values have only been very validated very coarsely.

Table 3.3 summarizes the results for all experiments with the additional results for the fusion set-up. One can observe that both models benefit from having the two modalities together, meaning that they are both able to improve their performances given additional data, even if it comes with a completely different structure and nature. The Hybrid NN-HMM partially reduces the difference of performance with the RNN compared to the video image experiment, but it is difficult to conclude whether the Hybrid NN-HMM can improve better on complex multi-modal data or if the BD-RNN model has simply hit its maximum learning capacity on this specific dataset. Indeed the results by the BD-RNN model are close to the state of the art results reported in the literature for comparable inputs.

Model	**Modality**	**Accuracy**		**JI**	
		validation	test	validation	test
[Wu et al., 2016] Hybrid NN-HMM	P			0.783	0.779
Hybrid NN-HMM	P	0.911	0.912	0.789	0.788
BD-RNN	P	0.921	0.922	0.814	0.811
[Neverova et al., 2016] DNN	P				0.831
Hybrid NN-HMM	(P)V	0.890	0.890	0.748	0.745
BDRNN	(P)V	0.909	0.909	0.791	0.789
[Neverova et al., 2016] CNN	(P)VD				0.836
[Pigou et al., 2016] BD-RNN	VD				0.906
Hybrid NN-HMM	PV	0.926	0.927	0.829	0.826
BDRNN	PV	0.935	0.934	0.852	0.846
[Neverova et al., 2016] DNN + CNN	PVD				0.868

Table 3.3 – Accuracy and Jaccard Index metrics on various modalities: Pose P, implicit pose for bounding box placement (P), colour images V and depth maps D.

3.6 Study 2: Reliance on Temporal Context in Representations

Role of the Temporal Convolution: Theory and Practice

This experiment requires a preliminary analysis of the role taken by the Temporal Convolution (TC). Located at the top of the representation learning module (Figure 3.10), this layer adds temporal context into the representations. Indeed, the input samples scarcely contain any context beyond the Gaussian de-noising and the derivative features for the body-pose. These inputs are initially processed on a frame-wise basis by the representation learning layers, but it is commonly admitted that using a small window instead of a single time-step improves the quality and stability of the representation [Franco et al., 1997; Pigou et al., 2016].

Following the example of [Pigou et al., 2016] and [Neverova et al., 2014][1], we take a window of features over time and apply a Neural Network layer as a transformation instead of merely aggregating the features with max-pooling or mean-pooling. As can be seen in Figure 3.26, this operation corresponds to a 1D convolution hence the name: Temporal convolution. Not only

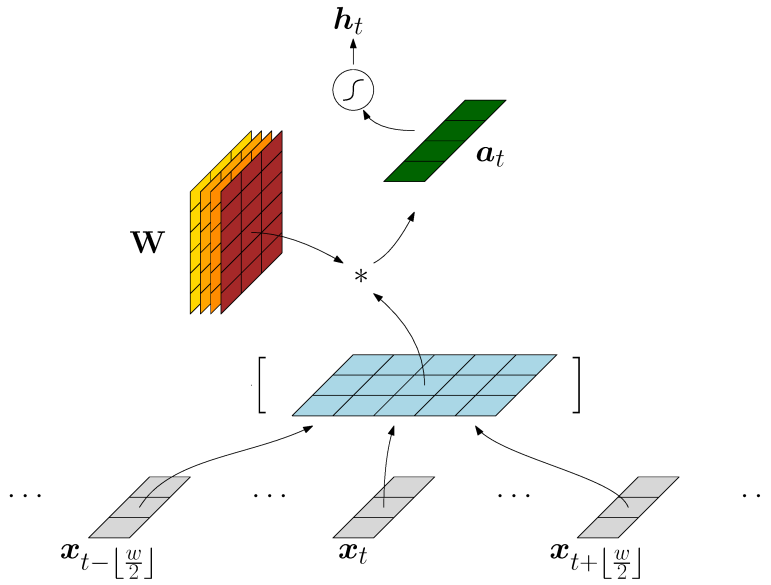


Figure 3.26 – Temporal Convolution layer.
$\boldsymbol{x}$ denotes the input sequence, w the width of the temporal window, $[\dots]$ the concatenation operator and W the battery of filters.

does the grouping and aggregating operations decrease the sensitivity to noise, but the linear transformation followed by a non-linearity can also learn short temporal patterns which would otherwise remain absent from the representation.

To understand the type of patterns learnt by the TC layer, Figure 3.27 displays the weights it has learnt during the end-to-end learning experiment on body pose data. The model seems

[1] Althought the temporal convolution is not explicitly mentioned, parts of the model they use are mathematically equivalent to a (strided) temporal convolution

to learn a great variety of patterns with many contrast detectors or spikes detectors of different width located at various positions within the observation window. The spikes on the right of the window with small weights elsewhere indicate a filter that focuses more attention on features from the near future and conversely the left ones correspond to previous observations. Similarly to 2D image filters, filters with two contrasted bands will roughly implement a temporal differentiation.

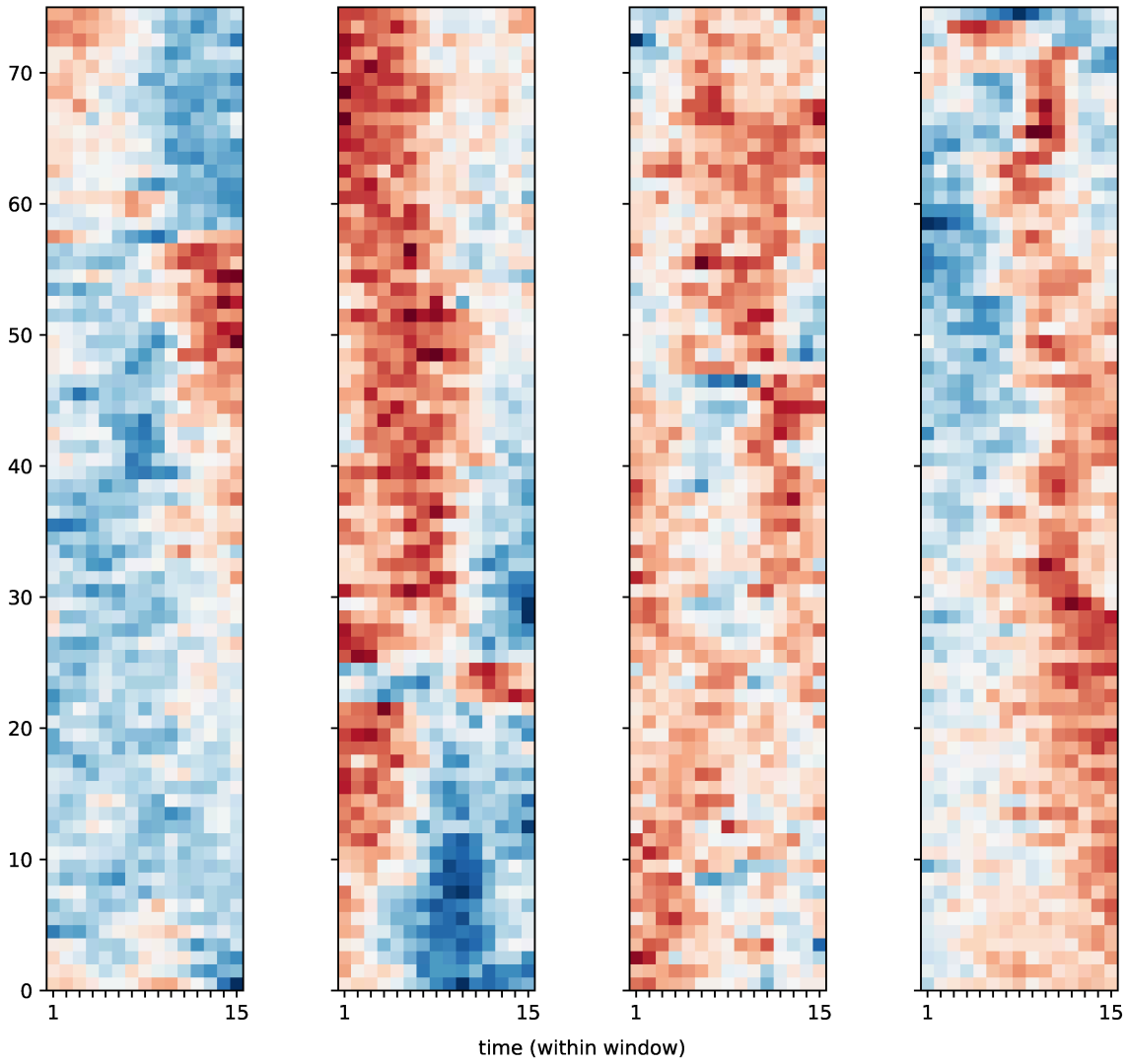


Figure 3.27 – Rows from the Temporal Convolution filters in the body pose BD-RNN model. Each filter is composed of a combination of these rows, one for each dimension of the input vector. The rows have been sorted using a one-dimensional t-SNE projection [Maaten and Hinton, 2008] to facilitate interpretation, but the actual filter banks contain a combination of these.

Experiments and Results

For both the Hybrid NN-HMM and the RNN based models, the temporal convolution improves the performances and off-loads the next detection stages from processing short temporal patterns. Meanwhile, the Hybrid NN-HMM and the BD-RNN can focus on long-range temporal structures. Since the same Temporal Convolution layer is used for both models, the comparison between the two keeps its fairness. However, it would be useful to stress the reliance of the two temporal models on this preliminary step. Indeed, little reliance would indicate robustness to the quality of the input, and to the temporal scale of patterns to be detected, two desirable properties when expert knowledge about that input is lacking. As a result, a series of experiments is run with varying context sizes to measure the impact on the final detection metrics. Since the number of parameters in the temporal convolution is proportional to the width of the window, additional experiments have been run with strided convolutions or fewer filters so that the number of parameters remains comparable. The experiment is run over the body-pose features for which both models are known to perform similarly well.

Observing the performance metrics reported in Figure 3.28, the Hybrid NN-HMM displays a notable and continuous degradation of performances as the window size diminishes. This means the transition model of the HMM by itself is not capable of capturing the finer short-term temporal patterns. Moreover, the filtering effect of the transition model, which disambiguates sequences of states by looking at the whole sequence, fails to compensate the lower posterior model accuracy which drops from 0.790 with a window of 15 frames down to 0.708 with 3 frames. By contrast, the RNN-based model demonstrates an impressive resilience to the amount of temporal context provided. We attribute the slight score regression at the largest window size to over-fitting. To summarize, the Hybrid NN-HMM relies on its state posterior model to learn and process small abstract temporal patterns before running the transition model. The latter enforces an intuitively sound structure on the gestures. Early temporal modelling remains a mostly optional stage for the RNN or can be reduced to a bare minimum. It is therefore not surprising that many papers decide to skip any preprocessing layers or temporal context-based embedding and use RNN layers right from the start in their model [Graves et al., 2006, 2013; Graves and Jaitly, 2014; Bluche, 2015; Plappert et al., 2017]. One should ponder the value of this advantage to the fact that short sequence filtering (Section 3.5.1) is necessary to eliminate numerous spurious detections: the RNN manages to learn and detect temporal patterns of any length but fails to strongly enforce a major characteristic of gestures which is their duration.

Interestingly, the number of parameters in the Temporal Convolution layer has little impact on the metrics which implies that it is indeed the context size alone that causes the observed variations and not the learning capacity or some under-fitting effect.

Our results rejoin similar observations made by [Bluche, 2015] on handwriting recognition. However, their experiments introduced context by concatenation of input frames instead of temporal convolution which is more easily interpretable, and they have not assessed the impact of added parameters on under/over fitting. By contrast, [Pigou et al., 2016] advocates using temporal convolutions in combination with Bidirectional LSTM. His experiments compare

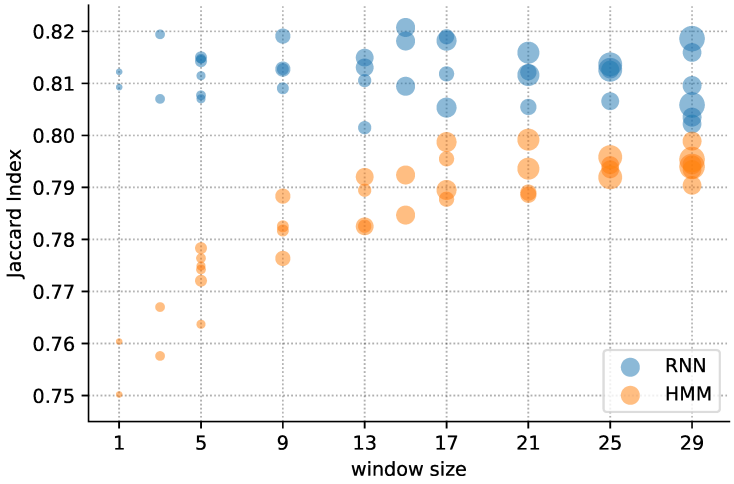


Figure 3.28 – Jaccard Index over the validation set under varying temporal context sizes. Marker size is proportional to the number of parameters.

the architecture with a drastically reduced model using a one time-step wide window. Without additional experiments, we hypothesise that the simpler model, having fewer parameters, may have suffered from under-fitting, or that temporal convolution only shines in a multi-layer architecture allowing it to learn structured temporal patterns.

Before concluding this study, we would like to address the question of why previous experiments have used the Temporal Convolution of width 15 when it is not the most optimal for the Hybrid NN-HMM model. We reinstate that our objective is the comparison of the temporal models RNN and HMM, a strong contextual embedding with a large Temporal Convolution would reduce the modelling load from these model and negate the point of our work. We also justify this choice as a sweet spot in terms of performance vs computation resources.

Temporal Context as a Substitute for Bidirectional Conditional Embedding

The assumption adopted so far was that Temporal Convolution gives information from the short-term past and future observation to the upper parts of the model. With a non-negligible window size (the average gesture duration is only 39 frames), one may wonder whether most of the sequential modelling is not taken care of by the Temporal Convolution. A second aspect of interest is the interaction between training the temporal model, HMM or RNN, and training the Temporal Convolution.

To improve the understanding of both aspects, we propose an ablation study where the BD-RNN is replaced by a single Recurrent Neural Network. In this configuration, the classifier receives information about past observations like previously: via the Hidden state vector of the RNN and the left[1] half of the Temporal Convolution filter. In absence of a reversed RNN,

[1] the actual weights are reversed in the convolution operations, but flipped for convenience in our visualizations and explanations.

information about future inputs is only available via the right half of the input window which is limited to $(w - 1)/2$ observations, where w is the filter width.

In this configuration, the performance severely drops with a testing Jaccard Index of 0.763 and an accuracy of 0.902, therefore confirming the importance of the Temporal model. But more importantly, we can observe in Figure 3.29 the synergy between the RNN layer and the temporal convolution sharing the temporal modelling work. Many filters tend to pack the larger values on the right-most filter weights which translates to reading the most recent time-steps available. In other words, the Temporal Convolution takes over the work of the missing reverse RNN by trying to read future observations. Since this phenomenon was not observed with the

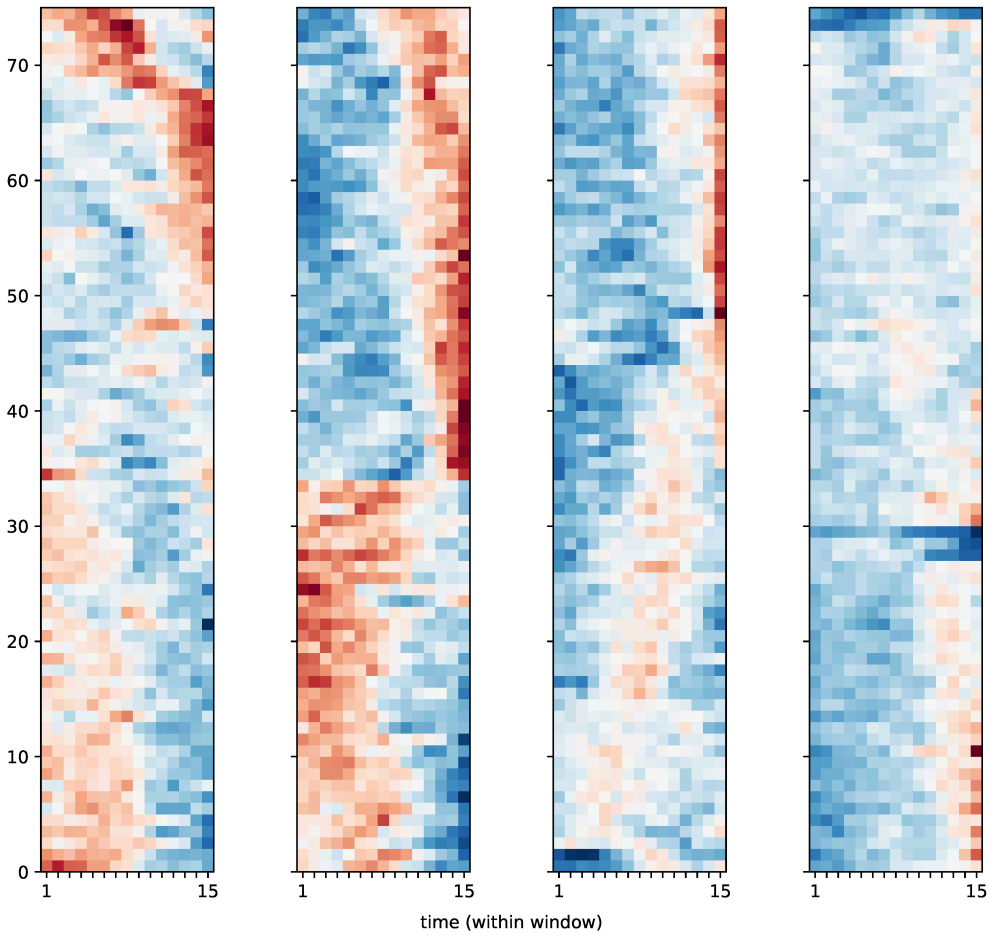


Figure 3.29 – Filters from a Temporal Convolution layer with mono-directional RNN based model.

Bidirectional RNN, we conclude that the RNN layers are indeed responsible for the long-range modelling, while the Temporal Convolution normally focuses on short term patterns within its scope.

General Remarks

Before concluding this study, a reopening of the discussion on what is actually learnt by the RNN seems appropriate. In the first end-to-end experiment, we observed that the RNN based model did not seem to learn general properties of gestures when maximising the training objective, for instance their average duration. We observe here that the GRU-RNN can simultaneously model a broad range of inputs types including short-term and long-term patterns. Moreover, the GRU-RNN demonstrates flexibility by compensating the limitations from the recurrent layer with other model parts when possible, here with Temporal Convolution. It turns out that, even though the model actually seems to learn a hierarchy of features, it failed to deduce general characteristics of gestures (what is their expected duration) because it was not compelled to do so by the objective function, and because the architecture of the Neural Network is not restrictive enough to enforce it.

3.7 Study 3: Specialization and Generality of Learnt Representations

Transfer Learning Experiments

The previous experiment varied almost directly the input quality of the temporal model via the context size parameter. But another interesting aspect to explore is the specificities and qualities of the representations learnt and then extracted from the input in order to feed the hybrid NN-HMM and BDRNN models respectively.

The so called *Deep* Neural networks family is often reported to draw some of its success from their ability to jointly train representation learning and task specific layers in an adequate manner. Besides, the learnt representations of the input produced by the intermediate layers demonstrate fairly strong generalization capabilities across different data sources from related domains. It is therefore not uncommon to see a Computer Vision model pre-trained on a large dataset and subsequently reused in other Computer vision tasks where the amount of data is not sufficient to properly fit a full end-to-end Neural Network.

The experiments from this section aim to observe the relationship between the temporal models, either the HMM's transition model or the BD-RNN, and the preceding representation learning layers. More precisely, the experiments are conceived to exhibit the reliance on a model-specific representation and the ability to produce generic and reusable features. For that purpose, we propose to exchange the representations learnt through each of the two temporal models. "Representation" here designates the last hidden vector of the representation learning layers. Figure 3.30 summarizes the modified training and inference procedure in the case of the Hybrid NN-HMM to BD-RNN transfer experiment.

Neural networks have often demonstrated a propensity to facilitate transfer learning of representations, especially in the domain of computer vision where large enough annotated datasets

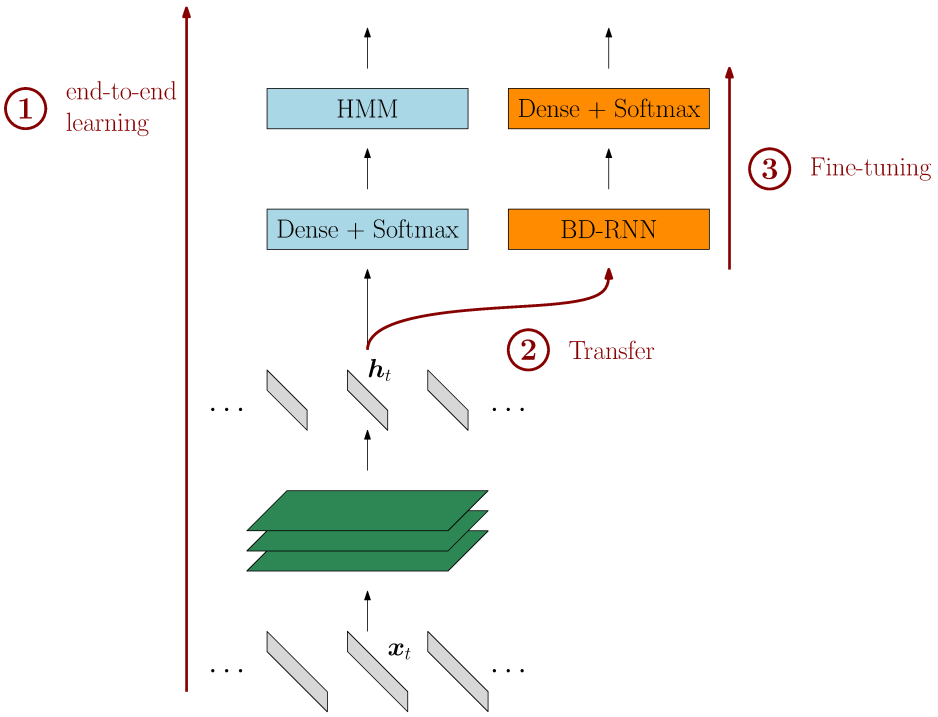


Figure 3.30 – Main steps and architecture of the Hybrid NN-HMM to BD-RNN transfer learning experiment: ① end-to-end learning as presented in section 3.5; ② freezing of the bottom layers parameters with deactivation of the noise regularization and transfer into the second model, when the state posterior probabilities are used, it is the posterior values $p(s_t|\boldsymbol{x}_t)$ that are rerouted into the BD-RNN; ③ training of the upper layers following the same procedure as the end-to-end experiment.

used to be scarcely available and would therefore be re-used to pre-train large models before fine-tuning on the actual task. In this case, the same dataset is used before and after the transfer, and only the model is changed so a fair level of generality and re-usability is expected between the embedding vectors. By contrast, any specificity required from the embedding vectors of either model may result in catastrophic performances loss during the transfer learning experiment.

Besides the adaptability and generality of representations featured by each model, the comparative study of the performances between Hybrid NN-HMM and BD-RNN is pushed further. In particular, we would like to verify whether the recurrent layers alone explains the slight advantage of the RNN based model reported in section 3.5 or if end-to-end training is necessary to achieve maximum performances. For that reason, no fine-tuning is performed after the transfer so that a model cannot unlearn the features from the other model and imprint new ones instead.

Table 3.4 compiles the results of the transfer learning experiments. In all configurations, training over the transferred features incurs a small drop in performances compared to end-to-end learning, but the loss is small enough to assume compatibility between the different representations. For both the body pose and video images of the hands, the RNN based model performs equally or better than the HMM fitted over the same data vectors via its posterior model. This confirms the assumption that recurrent neural networks are slightly more capable at learning temporal patterns than the state transition model regardless of the posterior model, and that RNN are robust with regards to the nature of their inputs.

Model	Source	Accuracy	JI	Δ_{source}	$\Delta_{end\text{-}to\text{-}end}$
HMM	RNN pose embeddings	0.899	0.752	−0.059	−0.036
RNN	HMM pose embeddings	0.918	0.794	0.007	−0.016
HMM	RNN hands embeddings	0.876	0.707	−0.081	−0.038
RNN	HMM hands embeddings	0.900	0.758	0.013	−0.031

Table 3.4 – Performance metrics in transfer learning experiments. Δ_{source} indicates the difference with the Jaccard Index of the end-to-end trained model that provided the representations, $\Delta_{end-to-end}$ the difference with the model trained end-to-end from scratch.

For completeness, and although it is not directly related to the objective of this experiment, a special note should be given about the hyper-parameters in these experiments. To facilitate early stopping and the selection of the best iteration, the learning rate had to be reduced so that the BD-RNN-based architecture would not overfit already in the first few training epochs. Besides, the re-weighting parameters on the loss functions in equation 3.2 was readjusted down to $\alpha = 0.2$ from its original value of 0.7, which equates to almost disabling the re-weighting of observed classes frequencies. It would therefore seem that Neural Networks are only affected by imbalanced classes when the input layers are being trained. This hypothesis remains open for further investigations.

BDRNN as a Transition Model over Posterior State Probabilities

In this experiment, we consider the posterior state probabilities $p(s_t|\boldsymbol{x}_t)$ as a form of representation for the input observation sequences. While it might seem unusual to pass such an input to a Neural Network, these values constitute the actual input to the temporal (transition) model $p(s_t|s_{t-1})$ contrary to the representation vectors that we used previously. Neural networks normally process real vectors of values distributed in a small range around zero, these representations are distributed meaning that information is encoded across multiple dimensions, and sometimes structured so that the similarity or difference between vectors has a semantic interpretation. By contrast, the state probability vectors $p(s_t|\boldsymbol{x}_t)$ complies to different constraints: unit sum value, high sparsity with many unrelated dimensions (states), relatively low dimension.

This experiment tests whether the BD-RNN model can mimic the HMM transition model given the state probability vectors. The logic of the previous experiment remains untouched apart from the slightly different transfer.

Table 3.5 summarizes the results of this experiments. We observe that even though a drop in performances is observed, the RNN copes surprisingly well with this special type of inputs, maintaining the same level of performances that the HMM achieved.

Model	Source	Accuracy	JI	Δ_{source}	$\Delta_{end\text{-}to\text{-}end}$
HMM	RNN pose embeddings	0.899	0.752	−0.059	−0.036
RNN	HMM pose embeddings	0.918	0.794	0.007	−0.016
RNN	HMM pose state posteriors	0.910	0.782	−0.006	−0.029
HMM	RNN hands embeddings	0.876	0.707	−0.081	−0.038
RNN	HMM hands embeddings	0.900	0.758	0.013	−0.031
RNN	HMM hands state posteriors	0.892	0.747	0.002	−0.042

Table 3.5 – Performance metrics for the BDRNN trained over posterior state probabilities.

3.8 Discussions and Remarks

Comparison between Hybrid NN-HMM and BDRNN

With this series of parallel studies, a comparison of two temporal models has been proposed on a fair test-bed that goes as far as sharing the representation learning layers to eliminate as many uncontrolled variables as possible. Our transfer learning experiments show that both models can be split into separate parts, with the representation layers being more or less interchangeable. The temporal models are therefore not tied to a specific representation or do not lead to learn highly specific features that cannot be found or used by the other model.

We have shown that Hybrid NN-HMM models can perform on-par with BD-RNN solutions, although it takes some targeted optimizations to do so. Having implemented the models our-

selves, we were able to pinpoint influential meta-parameters and settings. Different influential training aspects are reported, notably how to manage the priors and class imbalance (that particular issue also affects the BD-RNN model to some extent). Our experiments showed that one of the apparent shortcomings of the HMM model is its reliance on providing temporal context of sufficient size to the state posterior model. This latter model sees its performances greatly enhanced by the addition of context, which in turn benefits to whole Hybrid NN-HMM model. Optimizing this particular aspect through the representation learning layers yields substantial improvements and constitutes a central point of interest in the design of Hybrid NN-HMM models.

In most of our experiments, the BD-RNN models equipped with gated units tend to perform at least slightly better than the HMM counterpart, an unsurprising outcome in light of numerous other results reported in the literature on continuous sequence recognition.

A more interesting aspect for us stems from the robustness of the RNN with gated units in general: with end-to-end learnt features or posterior state probabilities, with or without temporal context, the BD-RNN often maintain fairly good results and demonstrates a remarkable robustness and adaptability. Nevertheless, throughout the extended testing campaign of experiments, the black-box nature of the Neural Networks made optimisation and debugging very challenging at multiple occasions.

Deep Learning and its Impact on Model Properties

A particularly interesting property that showed up through our experiments is the decoupling between interpretable parameters or values and the actual behaviour of the Neural Network. For instance, the combination of Temporal convolution, recurrent formulation of the RNN layers, and hierarchy of patterns from short-term to long-term that we observe all-together fail to prevent absurdly short or noisy predictions. Instead, Neural Networks seem to be largely driven by the objective function, and desired properties should be preferably encoded through this mean. To summarise: architectural decisions unlock the capability to learn particular patterns more than they constrain the model to follow them.

This personal observation rejoins other works. For example, the contributors of the Inception Network remark:

> One must be cautious though: although the Inception architecture has become a success for computer vision, it is still questionable whether this can be attributed to the guiding principles that have lead to its construction. Making sure of this would require a much more thorough analysis and verification. (Szegedy et al., 2015)

[Zhang et al., 2016] offers an original series of experiments to directly analyse this problematic, but we believe that many other publications follow an unstated objective of gaining more control over what is learnt by Neural Networks. It should be noted that this hypothesis does not contradict other explicitly stated objectives such as performance maximization, computation efficiency or training speed acceleration, but merely characterizes contributions with a shared orientation.

To support this idea, we resume in the next paragraphs a brief overview of the evolutions in Neural Network based Computer Vision over the last decade.

The first breakthrough with Deep Neural Networks was achieved by stacking up many layers with the intention to capture the compositional nature of visual object [LeCun et al., 2015]: bottom layers extract low-level features such as edges that get gradually combined and organised into more abstract and elaborate features by successive layers along the way up to the final classification layers. Having solved some major difficulties in training this class of model (with improved non-linearity functions, gradients descent methods, regularization techniques, and selection of sufficiently large datasets), it has become possible to exploit the exponential modelling complexity that comes from stacking many layers. This architecture shares some similarities with the Primary visual cortex of animals which is often quoted as a source of inspiration for the Artificial Neural Network equivalent [Fukushima, 1980; Lecun et al., 1998]. The intuitive motivation is supported by how easily the generic middle level features extracted by these CNN models can be reused across a whole spectrum of Computer Vision applications: one can split the model at a given layer to satisfy a certain level of compromise between specialization and re-usability.

Nevertheless, after a brief period of development that lead to particularly large and deep models such as VGGNet [Simonyan and Zisserman, 2014], later evolutions of Neural Networks have focused on more targeted supervision and more efficient computation paths. For example, the GoogLeNet model with its 22 layers uses multiple intermediate outputs, each bearing a training objective so that deeper layers receive a more direct supervision signal than if the final output alone was guiding the training process. Gated architectures such as Residual Neural Networks [He et al., 2016; Zagoruyko and Komodakis, 2016] reorganise the usual stacked architecture by inserting short-cut paths into the computation graph so that the model can decide whether or not one layer is necessary for the computation of the output prediction. Dense Neural Networks [Huang et al., 2017] push the concept one step further by systematically binding each layer with the outputs of all the preceding ones. Reversing the viewpoint on this method from top to bottom, the gating method can be interpreted as yet another way of connecting the objective function more directly to the lower layers of the model.

Most of the above-mentioned techniques rely on static computation graphs, that is to say the series of transformations from the input to the prediction is mostly deterministic (with the notable exception of noise injection for regularization and minor random transformations). Long Short-Term memory units [Hochreiter and Schmidhuber, 1997] still use a static execution graph but introduce some variability with the use of dynamic gate values.

Static graphs introduce fundamental computational limitations in terms of computational cost, but also tend to restrict the model to a more monolithic nature. More recently, a lot of effort has been invested on partial model execution and more specifically attention models. Similarly, to what the human brain does, attention in neural networks purposely restricts the extent of inputs taken into consideration to a specific point in space or time in order to meet computational requirements. With an appropriate selection of attentional target, it is assumed

that the model will be able to provide a correct prediction at a fraction of the cost it would take to process all available inputs. Besides, the elimination of irrelevant surrounding information might reduce input noise and therefore prediction errors. These attention models add more supervision into how the predictions are constructed and lead to more interpretable models as a side effect.

Finally, a related approach activates parts of the model instead of reading parts of the input. A notable example is given by [Shazeer et al., 2017] where a model containing multiple billion parameters is trained but only uses a manageable portion of it at a time. The authors of this paper propose a model and a training method that distributes the modelling work across a large mixture of experts out of which only a sparse combination is used at any given time. Once again, we can view this method as a strategy to control better what is learnt in each part of the model: here the use of experts introduces more specialized sections into the model.

To conclude, we do not believe one of these techniques will specifically predominate, but we point out that gaining more control on what Neural Networks learn may represent a substantial part of future researches in this field.

Chapter 4

One-Shot Learning for Gesture Recognition

One-shot and few-shots learning is defined as the process of learning and generalizing new concepts from a single or a few examples, for instance, learning to recognize new classes of objects never seen before from single images [Fei-Fei et al., 2006], or new words from single utterances in speech recognition [Lake et al., 2014, 2015; Santoro et al., 2016]. Plenty of applications may benefit from one-shot capable models, ranging from biometric verification [Chopra et al., 2005] to robotic arm programming [Duan et al., 2017].

For a long time, one-shot learning has remained beyond the reach of many supervised learning models for any practical application, as most of them require numerous samples from each label in order to grasp the underlying factors of variations. Such models trained naively over a few samples rapidly over-fit because they are not able to distinguish sample-specific features from generalizable concepts. Consequently, one-shot or even few-shots learning has often been considered an inherent advantage of human intelligence over machine learning.

It is reckoned that humans succeed in few-shot learning by exploiting a life-long worth of experience and general knowledge. This knowledge helps to promptly prune irrelevant features from new observations and instead focus on pertinent concepts for the task to achieve. For example, a human will quickly abstract away the notion of colour and viewpoint when tasked to separate images of dogs and cats. This ability to quickly prioritize concepts in preparation for a task is called *inductive bias* [Caruana, 1993; Thrun, 1998] which is more specifically defined as "any basis for choosing one generalization over another, other than strict consistency with the observed training instances" [Mitchell, 1980]. Inductive bias is now observable in virtually all Machine Learning paradigms related to one-shot or few-shots learning, either implicitly or explicitly, and constitutes a solid basis to help overcome the challenge of learning with little data.

The work presented in this section will cover sequence recognition algorithms, where plausible applications may include robotic assistants controlled via gestures or voice. Such devices should be able to learn new commands or tasks tailored to its user without requiring an extensive enrolment stage or any expertise regarding Machine Learning. In this mindset, the work

presented here will focus on isolated sequence recognition with a small but dynamic vocabulary of gestures, each learnt from one or a few examples. Nevertheless, all effort invested to reduce the amount of training data necessary to fit a model will strongly benefit to the field of sequential data recognition as a whole. Indeed, the acquisition and annotation of its datasets is often slow, tedious and expensive.

The following section will introduce the main trends and notable contributions in the field. Our work primarily focuses on sequential data, but since this aspect has received remarkably little attention in the context of one-shot learning, the presentation will concentrate on more general topics still applicable to our case. In particular, we will exhibit the common canvas on which Neural Network based solutions build up and then summarize the different axes of research followed by recent contributions.

After a statement of our objectives, we will provide a description of the supporting dataset for our experiments. Then, we will introduce our baseline model and a series of suggested improvements. Several studies follow to assess the impact of these propositions and feed a discussion on predominant properties which affect the design of one-shot capable sequence recognition models. Finally, we report experiments that we run using Matching Networks [Vinyals et al., 2016b], a model which follows the principles of meta-learning [Schmidhuber, 1987; Hochreiter et al., 2001] for its design.

4.1 Overview of One-Shot Learning

In general, a one-shot or few-shots capable classifier is only requested to distinguish a few classes, usually in the order of $\mathcal{O}(10)$. But to do so, only S annotated samples are provided for each class to train on, where S is known as the number of training shots and can be as little as one. The model is then tested and evaluated on testing samples from these classes like any regular classifier.

Recent publications have adopted the term "episode" [Santoro et al., 2016] to standardize the evaluation procedure of a one-shot capable model. An episode comprises several steps:

1. Sampling of a vocabulary subset $\mathcal{V}^{ep}$ from a large pool of available categories $\mathcal{V}$.
2. Sampling of S annotated training shots for each class in $\mathcal{V}^{ep}$, noted (X^{ep}, Y^{ep}).
3. One-shot or few-shots learning on the training shots (X^{ep}, Y^{ep}).
4. Episodic testing on additional samples drawn from the same vocabulary subset $\mathcal{V}^{ep}$.

A formal definition is provided later in section 4.2.

One must insist on the distinction between any training phase prior to episodes and the training phase in step 3 above. Early training typically aims to acquire background knowledge or inductive bias which is kept across episodes. By contrast, the episodic framework is a *testing* procedure which aims to evaluate the capacity of the model to learn new classes it has never seen before. In other words, testing the generalization capacity is the true purpose of the episodic procedure, whereas one-shot learning is only one step of each episode, albeit the most prominent

one. As a result, any modification made by the one-shot training step to the model is reset after each episode, otherwise the classes would not be new after their first appearance.

Adopting this testing methodology implies several additional properties. First and foremost, the overall pool of distinct testing labels observed across episodes $\mathcal{V}$ must be large enough so as to thoroughly assess generalization to any new class.

A training procedure prior to the episodic testing phase need not necessarily encompass the notion of episode. In any case, it will most certainly involve not one but a large number of classes to simulate the testing conditions and avoid specializing on these training classes. Moreover, the testing episodes must not involve classes from this training phase. Otherwise, the model would cheat and learn to recognize those categories but fail dramatically on unseen classes, which contradicts the very purpose of one-shot learning.

Since the "training" step that happens inside the episodes is so minimal, virtually all parametric models rely on this prior training phase. The latter's purpose is to acquire preliminary background knowledge and develop the inductive bias which is often crucial to the success of the model. To understand the frontier between this preliminary learning phase and episodic learning, some authors draw a comparison with the long-term memory and working memory from humans [Santoro et al., 2016; Graves et al., 2014; Kirkpatrick et al., 2017]: the working memory helps to memorize recent observations and the method to process the current task, whereas long-term memory outlives individual episodes to accumulate reusable general purpose knowledge. We emphasise the double purpose of memory in this context: it can recall both observations and the method to process them.

To facilitate the comprehension of this episodic testing framework and its consequences for the preceding training procedure, we consider an imaginary task consisting in one-shot learning animals from images. We suppose that users of this model will only ever use it on a handful of classes, each specific to a particular user, and that the model cannot require more than a few samples of these classes to become operational. For example, user 1 works with cat, dogs and horses, user 2 has snakes, eagles and foxes, etc. In this example, each user corresponds to one episode with its own instance of $\mathcal{V}^{ep}$. To train and evaluate the model, a dataset with a *large* set of races $\mathcal{V}$ must be assembled using annotated images of horses, dogs, cats, salmons, goldfishes, dolphins, tortoises, etc.

Assuming a Convolution Neural Network provides a feature extraction module in this case, the few training shots of each episode cannot be relied on in order to fit the filter parameters without over-fitting. Instead, a pre-training step must run prior to the episodes. For example, the CNN layers can be temporarily extracted and topped by a few classification layers, then trained for multi-class discrimination on a subset of randomly chosen classes from the datasets. Once trained, the layers below the classification ones are then transferred back into the one-shot model to serve as general-purpose feature extractors. Obviously, the resulting model is biased toward the animals it has already seen, so these won't be used for testing. Instead, testing will sample episode vocabularies $\mathcal{V}^{ep}$ from the remaining classes in order to assess how well the model generalizes to unseen animal races using episodic training shots only. Moreover, the

model is reset to its pre-trained state after each episode in order to avoid adaptation on these testing samples.

Although pre-training the CNN might be vital to the success of this model, one shall not disregard the importance of an episodic training (step 3 of the episode). If, for example, the model is destined to serve scientists each working on specific species such as reptiles or fishes, then each episode will certainly involve different sets of relevant features for classification. Episodic specialization would therefore have the potential to substantially improve the adequacy of the model for each specific user.

Based on this general description of the one-shot classification paradigm, the following sections introduce the framework used by most one-shot capable Neural Network models, a basis for many subsequent extensions.

4.1.1 Observation Embedding and Memory-based Classification

As mentioned previously, learning from a few episodic training shots and classifying subsequent observations raises two major difficulties: learning quickly and avoiding over-fitting given sometimes as little as one sample for each class. Under such constraints, the k-Nearest Neighbours (kNN) is often one of the first considered options: this model is relatively resilient to over-fitting due to its non-parametric nature and makes an economical use of available samples. For any test sample $\boldsymbol{x}$, and training shots X^{ep}, Y^{ep}, the Nearest Neighbour prediction is given by:

$$\tilde{y} = y^{ep}_{k^*} \quad \text{where} \quad k^* = \arg\min_k \|\boldsymbol{x} - \boldsymbol{x}^{ep}_k\| \tag{4.1}$$

However, the correctness of this prediction lies on the hypothesis that samples with a shared label tend to cluster together, and conversely scatter away for distinct classes. In practice, this assumption is rarely met with raw sample data. For instance, the euclidean metric between image pixels rarely correlates with the semantic proximity of the represented objects as it lacks colour or viewpoint invariance. Nevertheless, the desired property is attainable by first projecting the samples into a suitable embedding space using either crafted transformations or automatically learnt models such as Neural Networks.

Before discussing the multi-class situation and the kNN classifier involved, we focus on the identity verification task with the Siamese Neural Network [Bromley et al., 1993; Chopra et al., 2005] architecture. This small detour via a simpler problem serves a didactic purpose. A verification model is charged to produce a binary classification between authorized users and impostors. When challenged with a pretended identity and a test observation (which might belong to an impostor), the model produces a matching prediction grounded on a set of one or more enrolment observations used as references. Biometric access control at companies entrances are one such example: each company's security database contains reference shots, employees' badges provide the pretended identities and the biometric scanning devices the test observations. Access is granted based on a verification run between these three sources of

information. Each company is equivalent to one episode, with staff registration providing the training shots.

To succeed in this verification task, Siamese models use a Neural Network to generate embeddings for both the enrolment samples (aka training shots) and the testing samples. To avoid ambiguity, the training shots from the registration process will carry the *ep*isode superscript $\boldsymbol{X}^{ep} = (\boldsymbol{x}_k^{ep})$ while the testing shot(s) will be simply written $\boldsymbol{x}$. During testing, an identity is verified by comparing the embeddings for the reference shot(s) of the pretended identity $\boldsymbol{x}^{ep}$ and the challenge $\boldsymbol{x}$. $(\boldsymbol{x}^{ep}, \boldsymbol{x})$ is called a positive pair when $\boldsymbol{x}$ originates from the pretended identity, and a negative pair when the challenge observation originates from an impostor. A prediction $\tilde{y}$ is then made based on the similarity between the representations ($\boldsymbol{z}^{ep}$ and $\boldsymbol{z}$) compared to a threshold τ:

$$\boldsymbol{z}^{ep} = Net(\boldsymbol{x}^{ep}) \tag{4.2}$$

$$\boldsymbol{z} = Net(\boldsymbol{x}) \tag{4.3}$$

$$\tilde{y} = \mathbb{1}_{\cos(\boldsymbol{z}^{ep}, \boldsymbol{z}) > \tau} \tag{4.4}$$

Figure 4.1 provides a graphical representation of the process.

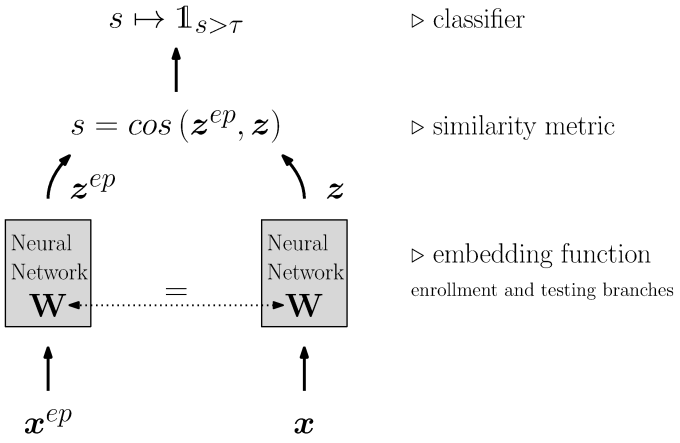


Figure 4.1 – Siamese Neural Networks for binary verification

Training the Neural Network should maximize dissimilarity between the embedding vectors from negative pairs and minimize it for positive ones. This ensures that the similarity measure between the representations from reference shots and challenges reflects authenticity. With a differentiable loss such as the cross entropy, the Neural network can be trained using back-propagation in an end-to-end fashion. Its training samples are composed of generated pairs, either positive or negative, drawn from a pool of observations with known identities. Since there is no reason to change the way embeddings are generated between the enrolment branch and the testing one, each side of the bipartite model uses the same set of shared parameters and are therefore jointly updated, hence the "Siamese" adjective.

Observing that datasets suitable for one-shot models must feature a large pool of classes $\mathcal{V}$, these can be seen as a pool of identities in a verification problem. In our animal classification

example, one can generate positive pairs and negative ones: cat-cat, mouse-mouse, horse-dog. As a consequence, the training procedure from the verification model can still be used to train the embedding function for a one-shot classification model, but the threshold-based classifier must be replaced. The choice usually narrows down to non-parametric methods which tend to be more resilient against over-fitting: k-Nearest Neighbours or "Shepard's Method" [Thrun, 1998] are two such methods. These classifiers take all training shots from an episode as input and infer the class of test observations.

This particular training and inference method noticeably ignores the notion of episode during pre-training. The Neural Network is simply optimized to learn general discriminative features for any pair of observations ignoring all other samples. During episodic testing, the Neural Network remains frozen and the kNN does not have any parameters to train, so the actual episodic learning (step 3 of the episode) remains merely a memorisation exercise of the reference training shots X^{ep}, Y^{ep}, saved for later comparison against the test samples (step 4).

Many contributions in one-shot learning can actually get cast into this fundamental paradigm: training of a general purpose embedding function maximizing between-class variance and minimizing the intra-class one, then followed by a task specific classification using little to no training. Additional refinements are then drawn from various related fields of Machine Learning. The following section presents some of the most prominent ones as of today.

4.1.2 State of the Art and Related Works

Transfer Learning

Transfer learning designates a learning paradigm where training a model involves two consecutive stages. The first one uses a distinct but related dataset while the second one matches the actual task of interest. For Neural Networks, this means training parts or all of the model on a related task and dataset, and then pursuing training on the actual task objective with its dataset. These two stages map to the notions of pre-training and fine-tuning a model respectively. It should be noted that the subset of parameters learnt during pre-training may or may not receive ulterior fine-tuning.

Through Transfer Learning, two related purposes are followed:

Representation learning: When the factors of variation in the samples from each dataset are similar, it is expected that pre-training will learn useful and reusable pattern detectors in the bottom layers of Neural Networks. For a large model with a small target task dataset, training these layers is impossible without the help of this pre-training step.

A successful transfer requires from this pre-training dataset to present inputs of a similar nature to the fine-tuning and testing data. Moreover, if the objective followed during pre-training is different from fine-tuning, it must lead to learn patterns that are relevant nonetheless. For example, one may pre-train an image classifier to distinguish objects and then transfer its lower layers into an animal detection model.

How compatible different tasks and datasets are is mostly based on intuition and expertise. Moreover, one must also decide which parts of the model can be pre-trained and which ones might be adversely affected by the specialization on a different task. To assist in these choices, some of the most influential properties to consider are the size of available datasets, the appreciation of the proximity of pre-training inputs with the final ones and the expected role played by the different modules in the model.

Generalization by hypothesis learning and specialization: By hypotheses learning, we mean learning the nature of features expected to be found in the data, for examples edges and patterns in images.

With Neural Networks, one may assume that most of the hypotheses leading to good generalization are induced by careful architectural choices and regularization. For example, shift invariance in images is emulated by using a combination of convolutional layers and local pooling operations. However, [Zhang et al., 2016] observed that popular deep Neural Network architectures actually (over-)fit easily given totally random data to train on, suggesting that Neural Networks do not enforce assumptions about natural images by design. On the contrary, their experiment suggests that robustness and generalization capabilities are acquired, and that reusable and useful patterns must be learnt from the data samples while optimizing the model for a given task. As a result, Transfer Learning might also bring a form of regularization by imprinting useful assumptions extracted from the pre-training dataset into the model.

A successful pre-training therefore serves two different purposes. Firstly, it optimizes models for the target one-shot task as side effect of optimizing the pre-training objective. Learning these patterns would be hard in a smaller dataset. Secondly, it enforces patterns which are not prone to over-fitting but rather relevant and generalizable to all natural occurrences of input observations.

Multi-task learning

Directly related to the transfer-learning method, Multi-Task learning aims to maintain the ability of the model to perform multiple tasks it has been trained for. This is especially appropriate in a situation more generally known as catastrophic forgetting [McCloskey and Cohen, 1989; Kirkpatrick et al., 2017] where learning a new task or fine-tuning suddenly eliminates knowledge from previous tasks.

For instance, [Shi and Kim, 2017] performs action recognition while regressing the body pose as a side result. Body pose, while not strictly necessary, is known to carry a lot of information about the gestures; adding it as a side target will help the network train on a slightly more accessible objective and improve the action recognition faster as a side effect.

At the edge between Multi-task Learning and Transfer learning comes Incremental or Life-Long Learning, where the model has an ever shifting objective such as learning new classes as learning progresses. In the robotic assistant case we mentioned as an example, this would

translate to preprogrammed instructions to which the user could add its own gestures to detect (see [Hariharan and Girshick, 2017] for an illustration of low-shot incremental learning on image recognition).

Meta Learning

Meta learning has been devised several decades ago in a general paradigm by [Schmidhuber, 1987], where its broad definition is summed up by "learning how to learn". The idea of meta learning is to have an agent working at two separate levels: a more general meta-level running across tasks, and a fine-grained task specific level, for example an episode in one-shot learning. The first level should focus on learning invariances across tasks instead of specificities. By analogy, human intelligence works by learning background knowledge which forges an optimized yet multi-purpose perception of the environment, and by using a working memory which evolves quickly to solve an active task.

An interesting contribution from [Hochreiter et al., 2001] proposes to view the samples within each episode as a sequence like a human would see them in real life. At each iteration, the model must produce a prediction for the current input, but also grounds its output on past observations or a representation thereof. Their paper makes a clever use of a Recurrent Neural Network to process samples with their observations so that episodic training shots are stored inside the recurrent hidden state. In doing so, the model can try to associate an incoming input with previously encountered data-target pairs and infer the appropriate output as shown in Figure 4.2. They argue that fitting the parameters of the Recurrent Neural Network is a form of

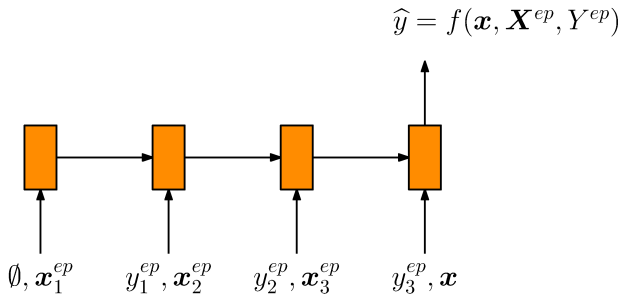


Figure 4.2 – Meta-Learning with sequential episodes [Hochreiter et al., 2001]. During testing, the episode samples $\boldsymbol{X}^{ep}$ and their targets Y^{ep} are both injected sequentially as inputs to the recurrent model. Consequently, the final prediction (potentially) has all the necessary information $(\boldsymbol{x}, \boldsymbol{X}^{ep}, Y^{ep})$ to produce an informed decision $\hat{y}$. It should be noted that labels are injected with a delay (y_i arrives with $\boldsymbol{x}_{i+1}$) to avoid "cheating" by simply copying the provided target.

long-term meta-learning, while the hidden state vector acts as a short-term memory to quickly learn episode samples.

More recently, [Santoro et al., 2016] has expanded this method by introducing an explicit readable and writeable memory bank besides the hidden state vectors to improve memorization. This is achieved again within the Neural Network framework by using a fully differentiable

memory from the Neural Turing Machines model [Graves et al., 2014]. A related work has been produced by [Vinyals et al., 2016b] which also processes episodic training shots sequentially, but manages a short-term memory with the help of an attention model, allowing the model to peek further back at previous hidden state representations instead of being limited to the current one.

More generally, the idea of meta-learning appears to influence most of the recent publications related to one-shot learning [Santoro et al., 2016; Vinyals et al., 2016b; Duan et al., 2017; Woodward and Finn, 2017], with ambitions to improve performances and episodic specialization without sacrificing generalization.

4.2 Objectives and Methodology

With our work, we would like to address few-shots learning of variable length sequences, a domain that has received relatively little attention so far, even though numerous potential applications exist in speech or gesture recognition. Foreseeable use-cases include: adding custom commands in a vocal assistant, teaching new tasks to a robot, and more generally giving the possibility for an end-user to program additional features in a device without resorting to a long and complicated enrolment process.

Our contribution aims to lay the foundations for a general purpose and reusable model to start with one-shot learning over sequential data. Due to the added complexity of learning with few-shots, we restrict the challenge to isolated sequences with a single instance to detect, for example a video containing one gesture, or a recording of one spoken word.

Formally, let $X = (\boldsymbol{x}_k)_{1 \leq k \leq \mathrm{N}}$ and $Y = (y_k)_{1 \leq k \leq \mathrm{N}}$ be a dataset of N labelled sequences, S the number of training shots, V the number of distinct classes in each episode. The evaluation metric is then derived from the following expressions, where the ^{ep} superscript is applied to episode-specific values:

$$\mathcal{V}^{ep} \sim Y^{\mathrm{V}} \text{ so that } \forall i \neq j, v_i^{ep} \neq v_j^{ep} \tag{4.5}$$

draws a vocabulary subset of V distinct classes. Let $K(v) = \{i | y_i = v\}$ index all available samples for each class label, then

$$K^{ep} \sim \underbrace{K(v_1^{ep})}_{\times \mathrm{S}} \times \cdots \times \underbrace{K(v_{\mathrm{V}}^{ep})}_{\times \mathrm{S}} \tag{4.6}$$

gives the indices for the S training shots from each class of this episode, and:

$$k_{test}^{ep} \sim \left(K(v_1^{ep}) \cup \cdots \cup K(v_{\mathrm{V}}^{ep})\right) \setminus K^{ep} \tag{4.7}$$

a testing sample to evaluate the model.

(4.8)

Given the training samples $X^{ep} = (\boldsymbol{x}_j)_{j \in K^{ep}}$, $Y^{ep} = (y_j)_{j \in K^{ep}}$ and the test observation $\boldsymbol{x}_{k^{ep}_{test}}$, the model should predict the true label $y_{k^{ep}_{test}}$. Often, the predictions have a probabilistic interpretation as posterior class probabilities. The model therefore returns $p\left(y = v | \boldsymbol{x}, \boldsymbol{X}^{ep}, Y^{ep}\right)$. In this work, the prediction $\tilde{y}$ will be evaluated by the accuracy metric:

$$\tilde{y} = \arg\max_{v \in L^{ep}} p(y = v | \boldsymbol{x}, X^{ep}, Y^{ep}) \tag{4.9}$$

$$L_{acc} = E_{K^{ep}, k^{ep}_{test}} \left[\tilde{y} = y_{k^{ep}_{test}} \right] \tag{4.10}$$

and the rank metric, which gives the rank of the correct class within the sorted predictions probabilities.

Most papers, and particularly the ones using the Siamese architecture, divide the model into an embedding function $Net : \boldsymbol{x} \mapsto \boldsymbol{z}$ and a classifier $\mathcal{C} : \boldsymbol{z}, Z^{ep}, Y^{ep} \mapsto p\left(y = v | \boldsymbol{z}, Z^{ep}, Y^{ep}\right)$. Often, little to no episodic specialization is required from the embedding function, while this property gets delegated to the second stage which must remain resilient to over-fitting. In the Siamese model, the Neural Network remains completely independent of any particular episode, and the non-parametric kNN takes care of the classification using the training shots. Nevertheless, this first stage must produce reusable discriminative representations over the whole spectrum of observations in the domain.

Given the objectives stated above, we have decided to focus on Neural Network based solutions. To avoid misunderstandings, this decision has not been based primarily on performances, but on the basis of modularity and flexibility with regards to supported input data types which we seek to have in our contribution. In fact, contrary to other topics such as image or speech recognition, their is no clear performance advantage to using Neural Networks once the few-shots constraints are imposed, but we would like to propose a model that is not specific to a particular type of sequences. To illustrate this compromise, a notable example is brought by [Lake et al., 2015] and [Rezende et al., 2016]: the former provides a very compelling use of an interpretable generative probabilistic model which implements a fine understanding of the handwriting process into their model. They decompose character drawings model into an elaborate hierarchy of sub-models starting with elementary primitives deformed into character parts which are themselves tied into strokes[1]. These strokes are in turn assembled according to a relational model to form a complete character. The second paper, by contrast, features a Neural Network offering enough adaptability to process not only handwritten characters from the same dataset, but also videos of facial expressions and house street numbers.

To the best of our knowledge, the closest contribution in one-shot learning that also uses Neural Networks for sequence recognition comes from [Pei et al., 2016], where a vanilla RNN is proposed as an embedding function and trained for verification, following a Siamese set-up as presented in section 4.1.1. Their model is capable of producing a fixed size representation conditioned on the full sequence of inputs using the hidden states of a RNN. They evaluate their

[1] Parts of character drawn without lifting the pen.

model on a series of pair-wise verification tasks: spoken words, speaker identities, signatures and gestures.

After a brief section dedicated to the dataset supporting our experiments, a series of sections present our work and propositions on one-shot sequence recognition. We have decided to build our model by drawing inspiration from the proposition by [Pei et al., 2016], with the difference that our work seeks to achieve episodic multi-label classification. A section is dedicated to the introduction of this model and the modifications that we propose to improve its performances and ensure the compatibility with gesture data. This section also include a description of the training procedure which completes what we view as our baseline system. The subsequent sections present several propositions of improvements to this initial baseline. First, we propose to use the triplet loss [Hoffer and Ailon, 2014] in lieu of a contrastive loss. Then we present the Shepard's method [Thrun, 1998] as a replacement for the kNN classifier. A detailed performance and error analysis is finally provided as part of a comparative study of these different propositions. In the last section, we modify our model to adopt the idea of Matching Networks [Vinyals et al., 2016b] which implements the concept of meta-learning to improve episodic adaptation.

4.2.1 DEVISIGN 2014 Dataset

Contrary to the previous chapter, the predictions are here restricted to a single label per observation sequence, a configuration which is more amenable to episode-based evaluation systems. Moreover, the inherent difficulty of one-shot learning motivates the analysis of this task to validate our models before attempting continuous recognition. Finally, one-shot learning evaluation (and training in most cases) requires to have a large vocabulary of distinct classes to evaluate the generalization capacity of a model, a property rarely found in continuous recognition datasets.

Given the stated requirements, the context of our research and past experiments, we have selected a sign language lexicon to evaluate our models, more precisely the DEVISIGN dataset [Wang et al., 2016] which contains 2000 classes performed by 8 different signers. Such a number of distinct classes is rarely found in gesture datasets and makes this one particularly suitable to evaluate generalization of a one-shot capable model.

The recordings are very similar to the Montalbano v2 dataset used in the previous chapter: frontal recordings with a clean and stable background as shown in Figure 4.3. The same depth camera is used, which therefore provides colour frames, depth frames, and regression of the body pose in 2D and 3D. Thanks to stricter and more advantageous recording conditions, the subject is placed more consistently in the frame, closer to the camera and without distracting or occluding environment. Visual inspection gave us the clear impression that the body pose regression failed less often and was more precise overall. For that reason and to keep the training data and model at a reasonably manageable size, our experiments will only rely on this cue. Our experience on continuous gesture recognition in the previous chapter have already demonstrated that this modality contains a lot of information by itself and should provide a reasonable basis for our study.

Figure 4.3 – Sample crops of the subject from the DEVISIGN dataset

Our tests use a 1000/500/500 random split of the class labels between training, validation and testing respectively. Due to the low number of recordings available for each subject (one or two for each class), we have decided to not split the performers between training, validation and test so that a sufficient variety of samples remains to train and evaluate the models.

4.2.2 Siamese Neural Network and Contrastive Loss Optimisation

Training methodology

As mentioned previously, this method fits a Neural Network embedding model which maximizes the similarity between pairs of latent vectors from genuine sample pairs and minimizes it otherwise. Genuine pairs are characterised by a similarity above a certain threshold τ. The Contrastive Loss [Chopra et al., 2005] translates this objective into a differentiable metric which was found to be both robust and effective:

$$L_{contrastive} = \begin{cases} \|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2^2 & \text{if} \quad y_i = y_j \\ \max(0, M - \|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2^2) & \text{otherwise} \end{cases} \tag{4.11}$$

This loss optimizes two different but related objectives. For positive pairs, the loss simply returns the quadratic error to minimize the dissimilarity between embedding vectors from a same class. For non-matching pairs, it can be viewed as the difference between a lower margin value M and the dissimilarity measure $\|z_i - z_j\|_2^2$, with a minimum value of zero to ignore pairs that already satisfy the margin. Indeed, points beyond this limit are considered to clearly originate from different classes. At the end of a successful optimization, samples from a same class should lie close to each other in the embedding space, while all other data points should lie far away, at a distance greater than M.

The threshold τ from the binary decision function in Equation 4.4 must place a compromise between false rejection of imperfectly matching positive pairs, and false acceptance of data points that are abnormally close to other classes. As such, its optimal value usually lies between 0, which would require perfectly similar positive pairs, and M, assuming perfectly dissimilar negative pairs. Unfortunately, both hyper-parameters τ and M must be cross-validated. Nevertheless,

our experience shows that the value of M does not impact dramatically the outcome of the experiments, and a coarse cross validation thus provides satisfactory results.

Although contrastive loss is frequently used in the one-shot learning community, the more generic binary cross entropy objective is still used sometimes, even in recent works such as [Koch et al., 2015]. Overall, no consensus seems to have emerged on the loss function, but the training objective remains the same anyway: finding an embedding that optimizes pairwise similarity or dissimilarity between samples according to their labels. Once trained, it is hypothesized that a k-Nearest Neighbours (kNN) classifier can exploit the latent projection of input samples to perform episodic multi-label classification:

$$\tilde{y} = y^{ep}_{k^*} \quad \text{with} \quad k^* = \underset{k \in K^{ep}}{\arg\min}\ d(\boldsymbol{z}, \boldsymbol{z}^{ep}_k) \tag{4.12}$$

Algorithm 4.1 summarizes the outline of the training procedure. Line 2 determines whether to train on a genuine or an impostor pair. The proportion of genuine pairs r translates to a compromise between false positive and false negative error rates. As already mentioned in the continuous recognition chapter, extreme class imbalances (r close to 0 or 1) might hurt the ability to learn anything at all in the Neural Network.

1: **repeat**
2: $p \sim \mathcal{B}(r)$
3: $i, j \sim \left\{[1 \,..\, \mathrm{N}]^2 \mid y_i = y_j \text{ if } p \text{ else } y_i \neq y_j\right\}$ ▷ draw samples
4: $\boldsymbol{z}_i, \boldsymbol{z}_j = \mathit{Net}(\boldsymbol{x}_i), \mathit{Net}(\boldsymbol{x}_j)$ ▷ compute embeddings
5: $\mathit{loss} = L_{\mathit{contrastive}}(\boldsymbol{z}_i, \boldsymbol{z}_j)$ ▷ evaluate model
6: update Net ▷ update parameters (gradient descent)
7: **until** termination criterion

Algorithm 4.1 – Detailed training procedure of Siamese Neural Networks $\mathcal{B}$ is the Bernoulli distribution, r the desired proportion of genuine pairs and Net the embedding function.

Preprocessing and Embedding Function

Qualitatively, the recordings of both Montalbano v2 and DEVISIGN are very similar in nature. As a result, the same augmentation and preprocessing techniques that were presented in sections 3.2.3 and 3.2.3 are reused here.

For the embedding function, [Pei et al., 2016] proposes to process the inputs with a single layer of a Recurrent Neural Network equipped with vanilla units. For the representation vector $\boldsymbol{z}$, they use either the final hidden state or the average of all hidden states as illustrated by options 1 and 2 in Figure 4.4. This architecture presents several limitations: firstly, the vanilla RNN cells in a mono-directional setting drastically reduces the maximum sequence duration option 2 can process. The myopic effect of vanilla RNN cells discussed in section 2.2.2 affects this model, and final observations will influence the representation largely more than early ones. By

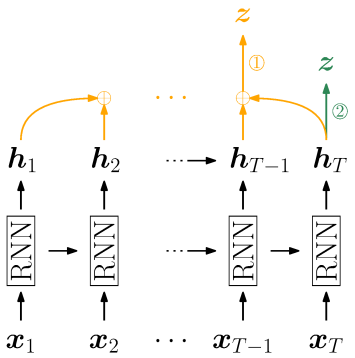


Figure 4.4 – RNN-based sequence embedding function from [Pei et al., 2016]

opposition, option 1 lacks a form of attention system that could take into account the relevance of certain hidden states. Indeed, the simple average gives equal importance to all hidden states, whereas early time-steps clearly lack context. Finally, using a single Neural Network layer as the embedding model limits the representation learning capacity and requires optimized hand-crafted inputs. While this last point does not constitute an issue per se, it conflicts with our objective to propose a generic and reusable model that is amenable to data types with low expertise available.

To address these issues, we propose several improvements over this model:

- replacement of RNN cells with Gated Recurrent Units to improve long-term temporal modelling
- addition of a reversed RNN (bidirectional architecture) to improve temporal modelling and improve modelling of early observations
- introduction of time-step-wise transformation layers to improve representation learning from low-level features

These modifications result in the neural network detailed in Figure 4.5. Its layer design roots its inspiration from the feature extractor used previously for continuous gesture recognition (Section 3.3.1). To summarize briefly the motivations for this architecture:

- At the bottom, three layers operate a set of frame-wise operations to provide learnt representation which are suitable for the targeted one-shot learning task.
- Two RNNs with GRU units process the series of representations in opposing directions using a Bidirectional RNN configuration. These two RNNs return temporally aware representations for each time step $t \in [1 \,..\, T]$. From the forward running RNN, $\vec{\boldsymbol{h}}_t$ represents the current and past observations, whereas $\overleftarrow{\boldsymbol{h}}_t$ respectively carries information backward using current and future observations.
- A final layer combines $\vec{\boldsymbol{h}}_T$ and $\overleftarrow{\boldsymbol{h}}_1$ which both contain representations of the whole sequence. This additional layer helps to control the output dimension without affecting the number of units in the RNNs and therefore their temporal modelling power.
- The final $\tanh$ non-linearity may seem odd considering the widespread use of Rectified linear units in this thesis so far. However, the representation $\boldsymbol{z}$ is meant for embedding

and comparisons with other vectors. The ReLU activation arbitrarily constrains the embedding space to positive only coordinates. On the other hand, $\tanh$ allows an isotropic projection of points in all directions. Consequently, it lends itself better to angle based comparisons between embeddings seen as directional vectors. The superiority of the $\tanh$ was also confirmed by our experiments.

A coarse cross-validation was used to adjust the number of parameters, layers and the output dimension, although we have not invested extensive testing into their optimization. Overall, this process led to a simplified model with fewer parameters than the continuous recognition one. We explain this phenomenon by observing that isolated gesture recognition is a simpler problem, and that the chosen dataset contains less training data of slightly better quality.

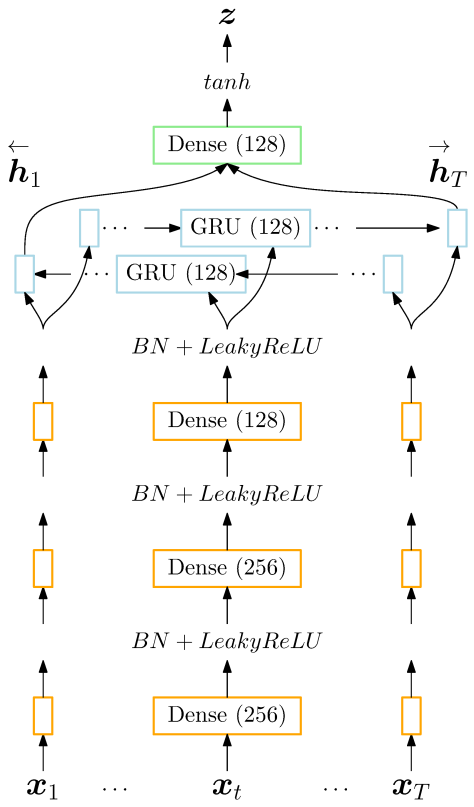


Figure 4.5 – Architecture of the embedding Neural Network

Instead of aggregating the hidden states of the RNN over time (for example by averaging $\overleftarrow{\boldsymbol{h}}_1$ .. $\overleftarrow{\boldsymbol{h}}_T, \overrightarrow{\boldsymbol{h}}_1$ .. $\overrightarrow{\boldsymbol{h}}_T$), only the final hidden states $\overleftarrow{\boldsymbol{h}}_1$ and $\overrightarrow{\boldsymbol{h}}_T$ are used with the assumption that they can hold all the necessary information about the sequence. This option corresponds to the second configuration tested in [Pei et al., 2016] and reportedly performed equally with the aggregation of recurrent hidden vectors. The addition of gated units and a bi-directional RNN configuration into our model should further improve the modelling quality, without giving an unfair advantage to recent observations as it often happens with vanilla RNN.

Regularization

During early experiments, we observed that our model would rapidly diverge and remain stuck issuing an almost constant output. Clipping the gradient values to $[-1, 1]$ proved insufficient, meaning that the gradient descent was not overshooting but naturally converging to this problematic state. We explain this issue by the absence of feedback from the loss about pertinent activity values: the loss function optimizes real vectors without taking into consideration the bounded nature of the $\tanh$ non-linearity on the last layer. To increase the distance between samples from a negative pair, training can try to increase activation values up to the saturation regime of the $\tanh$ function at -1 and $+1$, at which point the gradient becomes very small and training is stuck.

As a result we have decided to apply a regularization on the norm of vectors that would fall past the range $[1, 5]$:

$$L_{norm}(\boldsymbol{z}) = \begin{cases} \|\boldsymbol{z}\|_2 - 5 & \text{if } \|\boldsymbol{z}\|_2 > 5 \\ 1 - \|\boldsymbol{z}\|_2 & \text{if } \|\boldsymbol{z}\|_2 < 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.13}$$

$$loss = L_{contrastive} + L_{norm} \tag{4.14}$$

This regularization solves the degeneration issue while remaining relatively unobtrusive for the training process. Indeed its value is exactly 0 within the authorized range. Figure 4.6 shows the effect of regularization during the first epoch where the distribution goes from two modes at -1 and 1 to a smoother uni-modal distribution centred on 0.

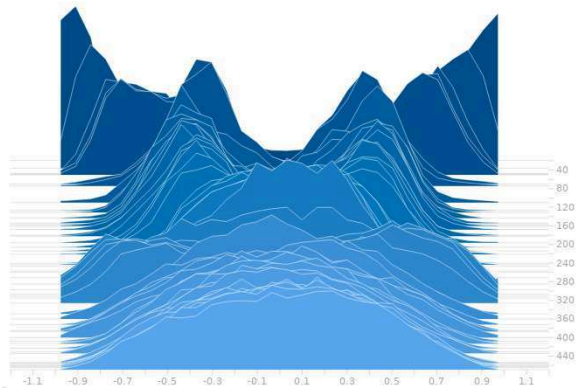


Figure 4.6 – Distributions of embedding neurons activations during the initial training iterations. Histograms are layered in front of each other with iteration 0 at the back.

4.2.3 Triplet Loss Optimization of a Discriminative Latent Space

Notwithstanding the performance aspect which will be covered later in the comparative study, contrastive loss suffers from several theoretical issues.

Its expression involves many hyper-parameters with a non-intuitive impact on performances, all of which require a tedious cross-validation based optimisation. In particular, the pairwise approach requires to set r, the sampling ratio of positive and negative pairs. A rate of genuine pairs too high may rapidly induce false positives considering that for a given observation, any sample from any different class is a potential impostor. On the other hand, a very small r might prevent the Neural Network from learning anything but to scatter points as far as possible from each other.

Moreover, contrastive loss uses fixed similarity thresholds τ and M that are shared by all classes without accounting for variation of cluster densities in the latent space. In effect, this training method imposes a maximal intra-cluster variance and minimal inter-cluster variance. By contrast, the Nearest Neighbour classification can work with variable cluster densities provided that scattered groups are sufficiently isolated from their neighbours as shown in Figure 4.7.

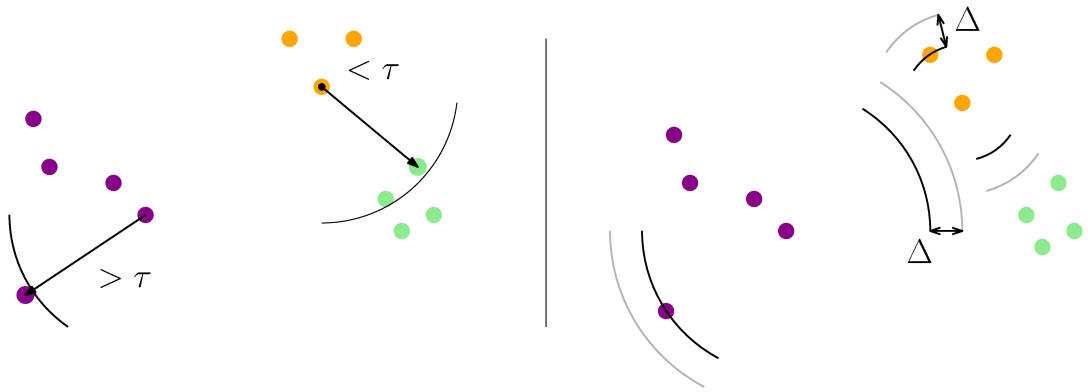


Figure 4.7 – Limitations of contrastive loss (left) compared to kNN classification or Triplet loss (right) for identical point clouds. With contrastive loss, the bottom-left point is too far away from its cluster to qualify as genuine, even-though it is still a closer neighbour than any point from other classes. Similarly, the green and red clusters are dense enough to ensure valid kNN classification in spite of their proximity.

To circumvent the aforementioned issues, we propose to modify the training algorithm of our model with the Triplet Loss function, which has previously been applied to one-shot learning for static data such as images [Hoffer and Ailon, 2014; Hariharan and Girshick, 2017]. Triplet Loss uses three points at once: two from the same class and an additional sample from a different class. The loss ensures that intra-cluster distances remain smaller by a given margin Δ than inter-cluster distances, but dispenses the model from imposing a fixed value (parameter M previously):

$$\begin{aligned} \forall (i,j,k) \in [1 \,..\, \mathrm{N}]^3 \,,\, y_i = y_j,\, y_i \neq y_k, \\ L_{triplet} = \max\Big(0, \|z_i - z_j\| - \|z_i - z_k\| + \Delta\Big) \end{aligned} \tag{4.15}$$

Stated differently: For any sample z_i, the loss become non-zero if a sample from another class z_k lies closer than a sample of the same class z_j (with an additional safety margin Δ). Triplet loss

ensures that the neighbourhood of any point only contains samples of the same class, but leaves the size of this neighbourhood unconstrained.

The norm can change for any given problem as long as it satisfies the properties of a distance, for example L1, L2, squared L2, cosine, etc. Finally, this method naturally balances the contrastive endeavour to minimize the intra-class distances and maximizing inter-class distances without the need for an explicit ratio r.

4.2.4 Shepard's Method

Definition

Both methods presented previously rely on the assumption that a pre-trained embedding function will maximize the classification accuracy of the Nearest Neighbour classifier. However, the notion of episode is absent from the parameter fitting process, as it only uses pairs or triplets of samples. Moreover, kNN classifiers cannot smoothly combine multiple training shots when available. Instead, its parameter k sets a constant number of neighbours to consider. Yet sometimes, weighting some training shots more or less could prove more appropriate to account for intra-class variability and the fact that some samples are more relevant than others.

[Thrun, 1998] proposed the Shepard method by introducing a differentiable decision function that combines the embedding vectors of episodic training shots with a given testing sample to return a class probability vector. The prediction for a test sample to belong to one class is the sum of (normalized) similarities with the training shots of that class:

$$p(\tilde{y} = l \,|\, \boldsymbol{z}, Z^{ep}, Y^{ep}) = \sum_{k \in K^{ep}} a(\boldsymbol{z}, \boldsymbol{z}_k^{ep}) \; \mathbb{1}_{y_k^{ep} = l} \tag{4.16}$$

where a is defined in [Thrun, 1998] by:

$$a(\boldsymbol{z}, \boldsymbol{z}_k^{ep}) = \frac{\frac{1}{\|\boldsymbol{z} - \boldsymbol{z}_k^{ep}\| + \eta}}{\sum_{k' \in K^{ep}} \frac{1}{\|\boldsymbol{z} - \boldsymbol{z}_{k'}^{ep}\| + \eta}} \tag{4.17}$$

with $\eta \ll 1$ to prevent computation issues.
And in [Vinyals et al., 2016a] by:

$$a(\boldsymbol{z}, \boldsymbol{z}_k^{ep}) = \frac{exp\left(cos\left(\boldsymbol{z}, \boldsymbol{z}_k^{ep}\right)\right)}{\sum_{k' \in K^{ep}} exp\left(cos\left(\boldsymbol{z}, \boldsymbol{z}_{k'}^{ep}\right)\right)} \tag{4.18}$$

Both expressions share the same logic with a normalization on similarity measures. We opt for the recent version which resembles attention models and content based memory access from recent works [Xu et al., 2015; Graves et al., 2014]. Swapping the softmax normalization in a by a sharper function can lead to a behaviour similar to the Nearest Neighbours classification, while choosing a kernel function $K(\boldsymbol{z}, \boldsymbol{z}_k^{ep})$ will lead to a kernel density estimator.

This classifier provides a drop-in replacement to the k-Nearest Neighbours, simply taking as input the same training shots and test embeddings Z^{ep} and $\boldsymbol{z}$ as previously. Yet, being differentiable, this classifier opens the possibility for end-to-end training directly over whole episodes of samples. To avoid confusion: this parameter fitting is not the episodic training from the third step of an episode but an optimization based on the predictions for test samples at step 4. This is a form of pre-training similar to that of Siamese Networks with contrastive or triplet losses, except that episodes are used to compute a loss and subsequently optimize the model.

Using episodes brings coherence between the pre-training and testing conditions and allows episodic adaptation down to the embedding model if desired (step 3 of the episodes), although our current model architecture does not exploit this possibility yet. Indeed, the Neural Network can take all training shots into account when generating an embedding: $\boldsymbol{z} = Net(\boldsymbol{x}, X^{ep}, Y^{ep})$. For example, [Vinyals et al., 2016b] runs a Recurrent Neural Network over training shot embeddings to produce episode specific representations. We test and review their method in section 4.4.

Even-though the Shepard's method has been formalized by [Thrun, 1998] back in 1998, most of the later publications in one-shot learning have opted for siamese or triplet based training procedure. However, [Vinyals et al., 2016b] accredited part of the success of their model to the use of a Shepard's classifier, which they used in conjunction with their proposed episode-aware embedding function.

Hinge loss as objective function

Before concluding this presentation, it should also be noted that the convenient notion of margin as offered by the triplet loss is not necessarily lost here. Based on our experiments, we propose swapping the categorical cross-entropy traditionally used on classifiers for a multi-class hinge loss:

$$\begin{aligned} L_{hinge} = \max(0,\ & p\left(\tilde{y} = y \mid \boldsymbol{x}, X^{ep}, Y^{ep}\right) \\ & - \max_{l \neq y} p\left(\tilde{y} = l \mid \boldsymbol{x}, X^{ep}, Y^{ep}\right) \\ & + m) \end{aligned} \tag{4.19}$$

which maximizes the margin between the correct class prediction and the most error-prone one up to a safety margin m.

Not only does this formulation bind more closely to the idea of the triplet loss, but it also seems intuitively more suitable than cross-entropy in the context of classification. Indeed, cross-entropy might allocate a considerable amount of training effort to increase the confidence in predictions when a sufficient margin m would suffice. With hinge loss, the error value beyond this margin becomes zero which lets the training procedure focus on more problematic samples. This idea relates closely to the concept of Curriculum Learning [Bengio et al., 2009] which usually needs to be implemented manually via careful sampling heuristics but is modelled transparently by the hinge loss.

In our experiments, hinge loss provides sensibly better results than cross entropy for identical models, with an absolute difference of around 10 points in accuracy. To begin with, the fact that cross-entropy penalizes correct but under-confident predictions is aggravated here by the presence of a cosinus function with a small support $[-1, 1]$ inside a softmax. Even in an optimal configuration where the cosinus on correct class is 1 and all other -1, the correct class probability is given by (Equations 4.16 and 4.18):

$$\begin{aligned} p(\tilde{y} = l \,|\, \boldsymbol{z}, Z^{ep}) &= \frac{\mathrm{S} \times e^{1}}{\mathrm{S} \times ((\mathrm{V}-1) \times e^{-1} + e^{1})} \\ &= 0.28 \quad \text{for} \quad \mathrm{S} = 1, \mathrm{V} = 20 \quad \text{(one-shot, 20 classes)} \end{aligned} \tag{4.20}$$

which still lags significantly below one, incurring a loss. With cross-entropy as a training objective, the losses from correctly classified samples are not significantly different from the ones of incorrectly classified sequences on which the training procedure should focus, as illustrated in Figure 4.8.

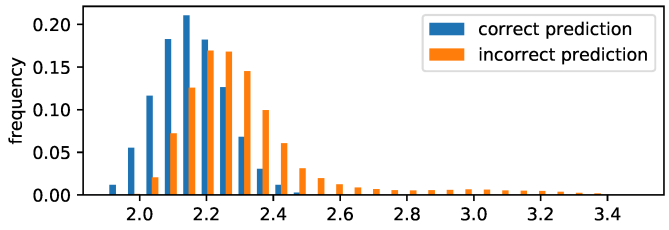


Figure 4.8 – Histogram of cross entropy losses on training sequences.

Regularization Against Uni-modal Embeddings

During early experiments with our model and dataset, we observed a collapsing of embedding vectors $\boldsymbol{z}$ into a single mode. In this configuration, the cosinus in Equation 4.18 hits a plateau at its maximum value 1, leading to small gradient values. For episodes vocabularies below 40, the updates create sufficient perturbations to escape this problematic region of the parameter space and recover the optimization procedure. For larger vocabulary sizes, we found necessary to penalize the lack of variability in the embeddings. To obtain consistent measurement, the new penalty term is based on the empirical variance of neurons activations estimated over the training shots Z^{ep}:

$$L_{diversity} = \frac{1}{\mathrm{D}} \sum_{i=1}^{\mathrm{D}} \max\left(0, 0.1 - VAR\left[z_i^{ep}\right]\right) \tag{4.21}$$

$$loss = L_{hinge} + L_{norm} + L_{diversity} \tag{4.22}$$

where D is the dimension of the embedding vector $\boldsymbol{z}$, 128 in our case.

4.3 Studies and Experimental Validation of our Models

4.3.1 Study 1: Comparison of Training and Classification Methods

Through this study, we train and test the three proposed solutions on the same data samples so that a more definite conclusion can be drawn about their respective advantages. Once the embedding model is trained, the performance is evaluated using both the Shepard's method and a *one*-Nearest Neighbour as classifiers since both can be swapped easily. Using more than one neighbour does not make sense when only one shot is available. Our tests with more training shots showed that using more neighbours is harmful to performances anyway. Finally, the episode size is set to 20 labels in all configurations.

Table 4.1 – Accuracy and target rank for 20 class vocabulary experiments. The distance used by the Nearest Neighbours matches the norm in the loss function. When a model is trained with Shepard's classifier, the cosine distance is used.

Shots	Metric	Objective	Accuracy		Rank	
			Shepard	1-NN	Shepard	1-NN
1	L^2	contrastive ($M=2, r=0.1$)	0.837	0.837	1.32	2.63
	L^2	triplet ($\Delta=1$)	0.851	0.851	1.30	2.49
	cosine	Shepard+hinge ($m=0.5$)	0.867	0.867	1.34	2.33
2	L^2	contrastive ($M=2, r=0.1$)	0.880	0.879	1.20	2.21
	L^2	triplet ($\Delta=1$)	0.901	0.895	1.17	2.06
	cosine	Shepard+hinge ($m=0.5$)	0.908	0.910	1.18	1.91

The results are summarized in Table 4.1. With an accuracy above 80%, all training and testing configurations demonstrate a strong ability to learn previously unseen classes, even in the more difficult one-shot setting. The contrastive and triplet losses produce versatile and reusable embedding functions: though first optimized for binary verification, the embedding functions they produce adapt well to the multi-class problem. The Shepard's method coupled with a multi-class loss function achieves the best results while following a more straightforward optimization scheme. By taking whole episodes at once, it can fit the multi-label classifier end-to-end directly in the episodic setting.

During testing, we observe that the Shepard's method and Nearest Neighbours perform similarly on the accuracy metric. The accuracy in the one-shot setting is mathematically identical because both techniques return the same prediction, but in two-shots, the softmax normalization in equation 4.16 apparently behaves almost like the hard maximum function from the Nearest Neighbours method, hence the similar results. The ranking metric, which measures the rank of the true class in sorted predictions, shows that Nearest Neighbour classification fails more

dramatically when producing an erroneous prediction; whereas the Shepard's method produces "almost correct" predictions. However, some of our early experiments revealed that a weighting between samples smoother than the softmax eventually leads to a lower testing accuracy, so a compromise must be found between averaging and take the closest neighbour.

Among the three training techniques, Shepard's classifier followed by a hinge loss yields the best accuracy overall, while ranks are fairly similar. Part of the explanation lies in the episodic training which tunes the model directly for the evaluation metric. We also suspect this training technique might also be able to move the model into more optimal regions of the parameter space which might not be accessible by other training strategies. Indeed, for an identical model architecture, this is the only experiment where the model would reach very high (>95%) accuracy on the training set: while the validation score hits a plateau, the training performance continued to improve much longer, showing that the model architecture is not the limiting factor on this task.

Error Analysis

Analysing the most frequent error cases here did not uncover any unexpected behaviour. Among the most frequent error cases are look-alike body movements that cannot be distinguished unless additional visual cues such as the hand shape are available. Figure 4.9 illustrates one instance of this error case.

Figure 4.9 – Similar gestures in DEVISIGN, the left arm slides below the right one in both examples, but the hand is flat on the left whereas one finger is pointing on the right.

Some gestures are problematic overall as denoted by the horizontal lines in the pseudo-confusion matrix in Figure 4.10. Visual inspection revealed that they correspond to more complex gestures composed of multiple subparts.

4.3.2 Study 2: Robustness to Variable Testing Difficulty

To evaluate the robustness of the model for more difficult tasks, additional tests have been run over various vocabulary sizes and training shots. To support the experiment, the one-shot model with a Shepard classifier was initially trained on 20-classes episodes as above. Then the trained model was challenged with episodes of varying vocabulary sizes and numbers of training shots. Figure 4.11 summarizes the results.

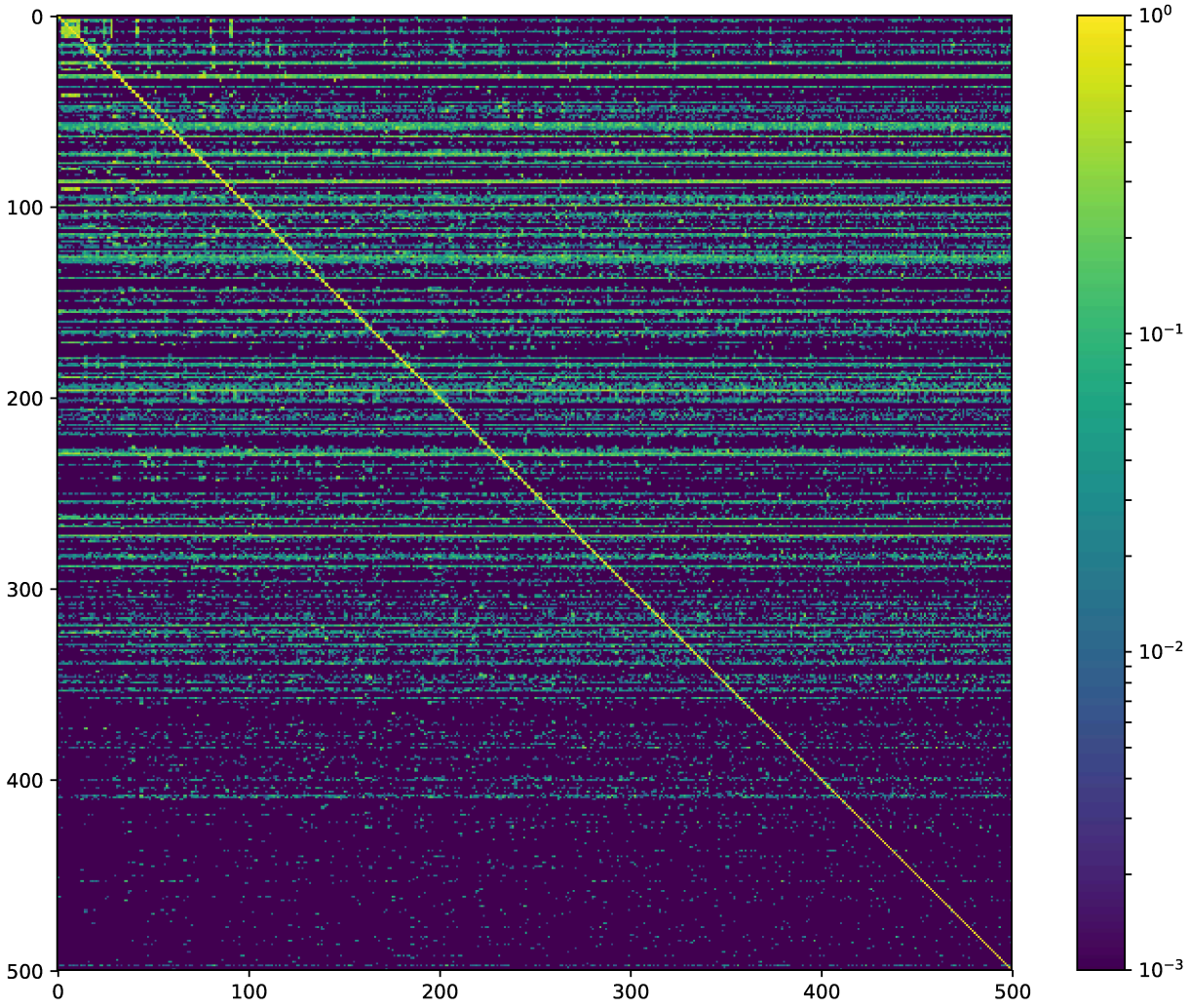


Figure 4.10 – Pseudo confusion matrix for the one-shot and 20 classes experiment with a Shepard classifier. Each element (i, j) contains the expectation of having the class i mistaken as j when both are present in an episode. An hierarchical clustering was used to find an order of the classes that emphasizes some groups of confused gestures, but not many were found.

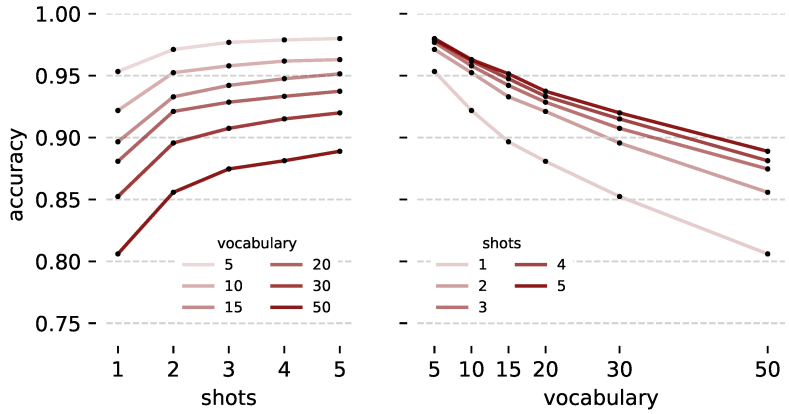


Figure 4.11 – Accuracy under varying episode vocabulary sizes and training shots

We observe that adding a second training shot significantly improves performances (around +5%), but subsequent shots only have decreasing absolute contributions. With respect to the vocabulary size, performances decrease steadily without catastrophic degradation. When considering the classification as a V-ways pairwise classification problem, one might fear that the complexity grows quadratically and quickly overloads the model, but we do not observe such a problem.

4.3.3 Study 3: Influence of Training Task Difficulty

Noting how this last experiment stressed the model outside of its training configuration, we finally compare training configurations with each other. Complexity varies from many shots and small vocabularies to one-shot with large vocabularies. The objectives of this experiment are two-folds: verify whether a model remains tied to its training configuration or not, and investigate the existence of optimal training configurations.

Figure 4.12 compiles the results for all experiments. The inherent difficulty of a testing task is hidden by substracting the score of the best model to all results for the same task. We observe that training on the simple three-shots/10 classes setting gives slightly better results in general, but the overall variability between models remains extremely small. It would seem that all training procedures lead to similarly generic embedding functions with a comparable representational power, discarding the hypothesis that models specialize on their training configuration.

The notion of optimality in the choice of training task remains open to future analysis, where we reckon an approach based on Curriculum Learning[1] [Bengio et al., 2009] might yield benefits and give access to more difficult tasks with many distinct classes.

[1] training with tasks of increasing complexity

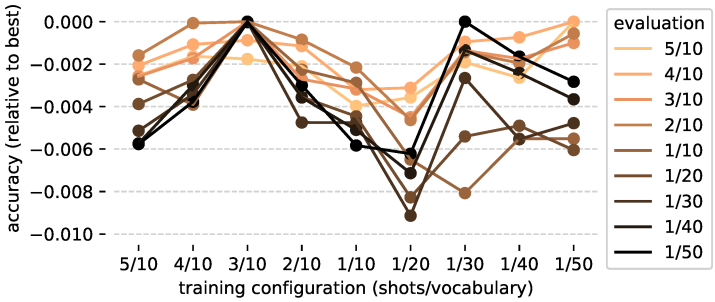


Figure 4.12 – Performances on crossed training/testing difficulty levels. For each testing configuration, the scores are relative to the best model.

4.4 Meta-Learning with Conditional Embeddings

Thus far, the inference model remained static without adaptation to the current episode. The one-shot learning step has not been used to modify the model's behaviour. Generally, the idea of adapting the system for an episode relates to the concept of learning-to-learn or meta-learning introduced earlier in Section 4.1.2. A naive form of episodic specialization would simply consist in fine-tuning the model on the training shots, but a strong counter-argument against this technique stems from the total absence of feed-back about over-fitting, which is bound to happen on such a small training set. As a result we decided to exclude this approach which feature little robustness and re-usability. Instead, we decided to experiment the solution proposed by [Vinyals et al., 2016b] with its Matching Networks. Their main contribution consists in conditioning embedding networks on the set of all training shots from an episode instead of processing samples independently of this source of information.

4.4.1 Matching Networks: Definition

Matching Networks resemble the general model architecture we presented earlier when combined with the Shepard's method: an embedding function that projects the training shots and testing samples followed by a parameter-free and similarity based classifier function (Equation 4.16). This model is still trained end-to-end as a classifier, without fine-tuning. However, given the whole episode as input, it should hopefully learn to adapt to episode specificities as part of the inference process. Following the same notations as the previous section, the embedding function becomes $\tilde{\boldsymbol{z}} = Net'(x, X^{ep}, Y^{ep})$ instead of simply $\boldsymbol{z} = Net(x)$ as we used earlier. More specifically, their modifications actually narrow down to an independent module that goes on top of a regular, sample-wise, embedding function. Two distinct functions g and f are used for the training shots and the testing shots respectively as depicted by Figure 4.13. We review each of them in details below.

From now on, the original embedding Neural Network function will be noted Net and its output $\boldsymbol{z}$ by convention. The fully conditional embedding of the training shots will be noted $\tilde{\boldsymbol{z}}_i^{ep} = g(\boldsymbol{z}_i^{ep}, Z^{ep})$ where $Z^{ep} = Net(X^{ep})$ refers to the original embeddings of the training shots.

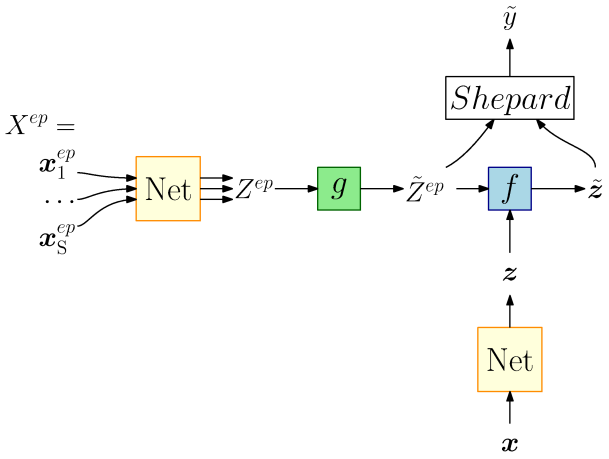


Figure 4.13 – Matching Networks.

The function g shares an interesting similarity with the work from [Hochreiter et al., 2001]: the inputs are chained and read in succession by a Recurrent Neural Network, the outputs of which provide representations conditioned on the previously read inputs. To condition on the whole support X^{ep}, a reversed Recurrent Neural Network is added to provide a conditioning on the following samples:

$$\vec{\boldsymbol{h}}_i, \vec{\boldsymbol{c}}_i = LSTM(\boldsymbol{z}_i^{ep}, \vec{\boldsymbol{h}}_{i-1}, \vec{\boldsymbol{c}}_{i-1}) \tag{4.23}$$

$$\overleftarrow{\boldsymbol{h}}_i, \overleftarrow{\boldsymbol{c}}_i = LSTM(\boldsymbol{z}_i^{ep}, \overleftarrow{\boldsymbol{h}}_{i+1}, \overleftarrow{\boldsymbol{c}}_{i+1}) \tag{4.24}$$

$$\tilde{\boldsymbol{z}}_i^{ep} = \boldsymbol{z}_i^{ep} + \vec{\boldsymbol{h}}_i + \overleftarrow{\boldsymbol{h}}_i \tag{4.25}$$

where $\boldsymbol{h}$ and $\boldsymbol{c}$ are the hidden state vector and cell state respectively. The original embedding is reused in a manner similar to Residual Networks so that the conditional embedding only comes as an additive correction term.

The order in which the samples are read is left unspecified by the authors and therefore assumed random. We also assume that the myopic effect of Recurrent Neural Networks (focusing more on recent inputs), is negligible due to the use of a bidirectional architecture with Long Short-term Memory units. However, this aspect should remain under scrutiny for larger vocabularies and shot numbers.

The conditional embedding for a testing sample $\boldsymbol{x}$ uses a different model. This time, the input is composed of the *conditional* embeddings of the support set $\tilde{Z}^{ep}$ and the embedding of the testing sample $\boldsymbol{z}$. In a series of k refinement iterations, an additive corrective term $\boldsymbol{h}$ is produced

following the set of equations below:

$$\boldsymbol{h}_0 = \vec{0} \tag{4.26}$$

$$a(\boldsymbol{h}_{k'-1}, \tilde{Z}^{ep}) = softmax\left(Z^{ep} \cdot (\boldsymbol{z} + \boldsymbol{h}_{k'-1})^\top\right) \quad \triangleright \text{ attention weight} \tag{4.27}$$

$$\boldsymbol{r}_{k'-1} = a(h_{k'-1}, \tilde{Z}^{ep}) \cdot \tilde{Z}^{ep} \quad \triangleright \text{ attention readout} \tag{4.28}$$

$$\boldsymbol{h}_{k'}, \boldsymbol{c}_{k'} = LSTM(\boldsymbol{r}_{k'-1}, \boldsymbol{h}_{k'-1}, \boldsymbol{c}_{k'-1}) \quad \triangleright \text{ refinement of correction}^1 \tag{4.29}$$

$$\tilde{\boldsymbol{z}} = \boldsymbol{z} + \boldsymbol{h}_k \tag{4.30}$$

Matching Networks only require one meta-parameter k, the number of refinement iterations performed by the function f. The number of LSTM units in each function is otherwise constrained by the size of the embedding vectors.

4.4.2 Experiments and Results

With many widely different classes, the DEVISIGN dataset contains many easily distinguishable signs but also easily confused ones. Consequently, each episode will certainly contain problematic combinations of gestures requiring more classification effort, while others are easily discriminated. With its attention model, Matching Networks could modify the embeddings by focusing specifically on closest training samples which are potentially problematic (unless they belong to the correct class). We therefore consider Matching Networks as good candidates to try to adaptively adjust the model at the episodic level.

Matching Networks are a fairly straightforward modification to add into the models presented previously. Once implemented, it is sufficient to inject the functions g and f into the computation graph prior to the classification step; both the rest of the model and the training procedures remain unaltered by this change. To our knowledge, there is no public implementation of Matching Networks that is both complete and correct. As a result the model has been implemented along with the rest of the experiments presented in this chapter. The code has been specifically modularised to facilitate reuse independently of the embedding Neural Network Network[2].

Since the conditional embedding modifies the model and adds more parameters, the comparison with the previous experiments could be biased, for example due to more over-fitting. As a result we have decided to add a separate baseline model as a more comparable reference. This variation of the Matching Networks uses a residual block with a single Neural Network layer featuring a hyperbolic tangent non-linearity. This layer imitates the functions g and f but processes its inputs sample-wise, in other words it produces *non*-conditional embeddings.

[1] This equation is most likely erroneous in the Matching Networks paper. We base our work on the previous publication from the same authors [Vinyals et al., 2016a] since it is indicated as the authoritative reference.

[2] available at https://gist.github.com/nlgranger/076ad1f7ce3c412a7983b9d1c02bc1b5

Table 4.2 compiles the results and shows that the conditional embedding produced by Matching Networks does not improve the performances over non-conditional embeddings from an equivalent model. The model used in this experiment had $k = 5$ refinement steps in the function f and we were not able to improve the results by varying this hyper-parameter. The same conclusion was reached in [Vinyals et al., 2016b] for a one-shot character recognition experiment, which seems to imply that Matching Networks only provide a benefit in specific situations.

Table 4.2 – One-Shot Learning with Matching Networks

	Accuracy	**Rank**
Matching Networks	0.850	1.42
non-conditional baseline	0.864	1.34
best from Section 4.3.1	0.867	1.34

During our experiments, we observed that the refinement steps operated by g and f (either from the Matching Network or the non-conditional reference model) do modify the initial embedding vectors $\boldsymbol{z}$, but this transformation brings no benefits at all. In fact the predictions based on the raw embeddings $\boldsymbol{z}$ achieve a slightly higher accuracy than when using $\tilde{\boldsymbol{z}}$, even though the loss function optimizes the latter. It remains unclear what influential factors can explain this inefficiency.

4.5 General Discussions and Remarks

In a way, this chapter on one-shot gesture recognition can be viewed as a detour from the previous chapter on continuous recognition. Indeed, we were able to leverage the Neural Network architecture design from the previous chapter and integrate it relatively easily. By choosing a gesture dataset (more precisely sign language), we forced ourselves to rely heavily on the representation learning capabilities of the model to exploit low-level body-pose features. We show that our model based on a bidirectional Recurrent Neural Network is capable of producing small embeddings representing whole sequences. This feat constitutes a testimony to the great modularity and flexibility of Neural Networks.

With our review of different models and training methods, we were able to identify Shepard's method as a superior method to provide an end-to-end differentiable neural network for one-shot learning. When combined with a hinge loss, this method outperforms the most popular techniques used so far in the literature, namely Siamese Networks with contrastive loss or triplet loss. We explain the advantages of the Shepard's method by the adequacy between training and testing conditions. Indeed, contrary to the Siamese Network which optimizes a binary classification objective, Shepard's method allows to keep the same inference algorithm for training and testing. Doubled with our suggestion to use the hinge loss instead of the contrastive loss, the model is trained very specifically for the evaluation task instead of a related

one, an issue that was already raised in the previous chapter when we observed that maximizing frame-wise cross-entropy was not entirely adequate to maximize the Jaccard Index metric.

One-shot learning is a very active topic nowadays, and will probably gain more attraction in the upcoming years considering the ubiquity of these learning conditions in real life. The literature on the topic adopts different paradigms to tackle the problem: multi-task learning, transfer learning, meta learning, etc. The latter seems very promising since it focuses on specializing a model at the episodic level without loosing sight of general prior knowledge, and in particular without over-fitting. Matching Networks are one such instance of a meta-learning model, but failed to improve the results on our dataset. We hope that research in the field will look for solutions with the same ambitions as the Matching Networks, but with improved generality and re-usability.

Chapter 5

Summary, Perspectives and Future Work

As a user, I personally envision many applications for human-machine interfaces based on more natural vectors of communication rather than the existing methods which necessitate keyboards, touch screens or other devices requiring physical interaction ruled by unnatural protocols. Typically, a robotic assistant should be able to interpret vocal instructions or non-verbal communication based on signs or gestures. It should process information from its environment as-is without relying on an expert user to pre-process the data in a more optimal representation.

A large number of these naturally occurring sources of information share a common property: they take the form of a temporal sequence; for example speech, visual scene observations, handwriting... Having adopted this perspective on the long-term applications of this work, this thesis has been organised to adopt a series of objectives and orientations. We have tried to invest more effort on modularity, re-usability and generality rather than optimizing the models for expertly designed hand-crafted features that are specific for one particular type of input modality. To comply with the idea of seamless interaction, our choice of models has been oriented toward solutions that could realistically run in real-time (more precisely with a small delay at a manageable computational cost).

The decision of using gestures was motivated by three reasons: first, the originality and relatively low coverage by the scientific community matched our idea of reducing feature engineering in favour of representation learning. A second related reason was later found when we observed the great variability of execution in gestures, which largely reduces the worth of efforts invested in feature engineering and motivates the development of robust and adaptable models for representation learning and recognition. Finally, gestures are multi-modal with a complementarity between low-dimensional body pose information and high dimensional video frame representations so that the flexibility of the model can be tested within a unique framework corresponding to a single task, which is convenient for comparative studies.

Parallel to our research objectives, the field of Sequential Data Recognition has witnessed over the course of a few years a swift transition from Probabilistic Graphical Models toward

Neural Networks, and the latter have without any doubt grabbed a large amount of attention from the research community. Intrigued by the reasons behind this change more than by the raw performance metrics, a large portion of the thesis has been invested on grounding the detailed properties that explain and justify this sudden change. A second part of the work has targeted Few-shots learning which aims to address learning in situations where only few training examples are available. In terms of real-world application, one or few-shots learning can benefit systems in isolation that need to learn to recognize new classes, execute new tasks or adapt to a new user or environment. In such cases, it seems unrealistic to expect an end-user to provide more than a few training instances, and even more to supply a representative variety of samples that covers the runtime test conditions.

The following paragraphs summarize the different contributions of the thesis.

chapter 1 This chapter focuses on Continuous Gesture recognition in sequences of body poses and video frames. We introduce an experimental setup that combines two major temporal models, namely Hybrid Neural Networks and Bidirectional Recurrent Neural Networks into a shared framework which is more amenable to comparative studies. Through the elaboration of this pair of models, we were able to pinpoint influential properties and parameters, notably how to deal with imbalanced class distribution in term of number of samples.

To asses the robustness of the two temporal models, we trained the models on a variety of inputs of different nature and complexity: body-pose data, hand-crop images, or a combination of both. In all configurations, our models demonstrate close to state-of-the-art performances which proves that our models are robust to the type of input, efficient in training, and that the constraints introduced for the comparative work have not denatured the qualities demonstrated by purely performance-oriented experiments reported in the literature.

The stress testing methodology is extended with a series of experiments where the representation learning layers that precede the temporal models are progressively deprived of local temporal context to the point that all of the temporal modelling lies on the following HMM transition model or Bidirectional Recurrent Neural Networks. The amount of temporal context is regulated with Temporal Convolution Layers. Similarly to low-level 2D filters in Convolutional Neural Networks, we show that these 1D temporal filters are interpretable and learn intuitively sound patterns. We verified experimentally that these Temporal convolution layers effectively learn short-term patterns and simplify the modelling work of the subsequent temporal modules. The Hybrid Neural Network-Hidden Markov Model was found to be much more reliant on this prior short-term context modelling, whereas the RNN-based model managed to compensate when necessary and demonstrated a high level of robustness.

In a final study, we analysed the relationship between the lower part of the model, the "representation learning" layers, and the upper layers; in particular, we tested whether the

temporal model learnt highly specific features that were necessary to achieve the maximum performances or if the representations were mostly generic and interchangeable. Our testing procedure implemented exactly this idea with a series of transfer learning experiments where the representation learnt by one model is submitted to the other. Although both of the HMM transition model and the BDRNN suffered a slight regression from not having tailored inputs optimized via an end-to-end training procedure, the difference was found small enough to conclude that both models generate a similarly generic and reusable representation learning model.

Notwithstanding the performance advantage which we find relatively small for common usage outside benchmarks, the RNN-based model demonstrated more robustness and proved easier to implement due to its homogeneous structure with a single objective governing the end-to-end training procedure. This monolithic nature and black-box training aspect raised several issues nevertheless: difficulties in optimizing the architecture, difficulties in tracing the source of errors, inadequacy between the training loss function and the true objective.

The Hybrid NN-HMM presented a greater challenge with its two stage architecture and the alternating training steps. We have already mentioned how class imbalance and local temporal modelling constitute a difficulty for this model, but we found countermeasures to alleviate their effects in our experiments, namely loss function re-weighting and Temporal Convolution to embedded local context in the state posterior model.

chapter 2 With the one-shot setting, Machine Learning makes one more step toward the way humans themselves rapidly acquire knowledge.

In our experiments, we started with an established method, Siamese Networks with contrastive loss, and improved it using the triplet loss, the Shepard's method and the hinge loss. Our presentation of these methods emphasizes the striking similarities between them, as they all gravitate around the idea of embedding input sequences into a latent euclidean space of fixed dimension, with a semantic interpretation of the distance between samples. In fact, our experiments all reuse the same neural network architecture to implement their embedding function. Model centred around an embedding function have a widespread usage in Machine Learning, for example in content retrieval problems [Taigman et al., 2014; Cao et al., 2017] where it is used explicitly, but was also in our representation learning module from Chapter 3, and in many models where intermediate results may also serve as latent representations. Obviously, this class of methods has largely benefited from the advances in Deep Learning, which provides powerful embedding functions trained more efficiently thanks to new models and techniques. Among the three analysed methods, the Shepard's one trained with hinge loss performs the best, a success we attribute to the adequacy between training and testing configurations which leads to optimize the model precisely for the intended purpose.

However, the training techniques used by these models cannot scale down to the episodic training level, which provide too few training shots to allow adaptation without risk-

ing over-fitting. A workaround is found by anticipating most of the training prior to the episodes in a form of general background knowledge acquisition, leaving little to no training work inside episodes. This concept in fact outreaches the one-shot learning setting, since one can view unsupervised pre-training of early Deep Learning models as an instance of that idea. Yet the datasets used for fine-tuning these models afterwards still counted sufficiently many samples so that regular training methods remained applicable. One-shot learning shines by formalizing training conditions which render the examination of this aspect central to solving the problem. It drives attentions to a different class of models that can leverage previously learnt concepts and quickly adapt to various local problems defined as small data episodes. We analysed and tested Matching Networks [Vinyals et al., 2016b], one example of the meta-learning principle, but observed that it did not improve performances for our combination of model and dataset.

Combined with the modular design and the demonstrated re-usability of the different parts of our model, the works presented in this thesis should be easily reproducible, reusable, and adaptable for other tasks and purposes. The code for our experiments has been made available in order to assist in either objectives. In particular, it should ease considerably the experimentation on our suggested improvements.

Although we did not test other training methods, we are aware of work that jointly trains the posterior and state transition models of Hybrid NN-HMMs, where the posterior model must maximize a Mutual Information objective in a global sequence-wise fashion [Bahl et al., 1986; Hinton et al., 2012]. Another axis of improvement is the scaling issue that comes with the phoneme model used in most speech and gesture recognition works. Indeed the number of states grows linearly with the size of the vocabulary. As a result, tied states and state clustering are promising options to consider in order to improve our model.

Future work may also consider automating the construction of the state graph in order to lower the amount of expertise needed to design it appropriately. Such knowledge might not exist in all domains. This contribution would be two-sided though, since this graph is precisely how a lot of knowledge is gained from the model or injected into it.

We consider the one-shot learning paradigm as a very motivating objective for future work on Neural Networks for gesture recognition. Besides transfer learning, multi-task learning, and meta-learning, the topic also draws inspiration from very recent axes of research, notably algorithm learning where working memory is used to store short-term task-related information [Graves et al., 2014]. Another possible source of inspiration is attention models, where focusing is used to adaptively select relevant features and discard others. We reckon these works will accelerate the pace of one-shot learning research.

As a more general perspective, one particular finding we observed through our experiments is the absence of tangible relationship between the architectural design of our Neural Networks and the properties or results they demonstrated after training. Besides the personal dissatisfaction of having to work partially blindly to find new architectures, we believe this problem should be more explicitly stated as part of the reasoning behind new Neural Network designs, new training

techniques and algorithms. Furthermore, some advances in Neural Network are difficult to justify or quantify due to the complex interaction they might exert on the whole Neural Network.

To conclude, our intuition about promising research in the field would begin with a convergence of techniques that lead to more efficiency and control in the training and inference of large modular models. Gesture recognition is one of these subjects where the computational cost of managing videos inputs in real time justifies investing efforts on these specific points in the short term. We also believe the work on one-shot learning we presented in the second chapter is still in its infancy. One-shot learning defines a specific learning situation, and a particularly challenging one, but the outcome of research on the subject may have far-reaching consequences in terms of real-life applications with more seamless interfaces toward non-expert users.

Bibliography

Ahad, M. A. R. (2013). Motion History Image. In *Motion History Images for Action Recognition and Understanding*, pages 31–76. Springer London, London. DOI: 10.1007/978-1-4471-4730-5_3.

Bahl, L., Brown, P., Souza, P. d., and Mercer, R. (1986). Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *ICASSP '86. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 11, pages 49–52.

Baum, L. E. and Petrie, T. (1966). Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48, New York, NY, USA. ACM.

Bhuyan, M. K., Ghosh, D., and Bora, P. K. (2004). Finite state representation of hand gesture using key video object plane. In *2004 IEEE Region 10 Conference TENCON 2004.*, volume A, pages 579–582 Vol. 1.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

Bluche, T. (2015). *Deep Neural Networks for Large Vocabulary Handwritten Text Recognition*. Theses, Université Paris Sud - Paris XI.

Bourlard, H. and Morgan, N. (1990). A continuous speech recognition system embedding MLP into HMM. In *Advances in neural information processing systems*, pages 186–193.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Braffort, A., Benchiheub, M., and Berret, B. (2015). Aplus: A 3d corpus of french sign language. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility*, ASSETS '15, pages 381–382, New York, NY, USA. ACM.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. (1993). Signature Verification Using a "Siamese" Time Delay Neural Network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, pages 737–744, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Cao, Z., Long, M., Wang, J., and Yu, P. S. (2017). HashNet: Deep Learning to Hash by Continuation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5609–5618.

Caruana, R. A. (1993). Multitask Learning: A Knowledge-Based Source of Inductive Bias. In *Machine Learning Proceedings 1993*, pages 41 – 48. Morgan Kaufmann, San Francisco (CA).

Chai, X., Liu, Z., Yin, F., Liu, Z., and Chen, X. (2016). Two streams Recurrent Neural Networks for Large-Scale Continuous Gesture Recognition. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 31–36.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546 vol. 1.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314.

Dieleman, S., Schlüter, J., Raffel, C., Olson, E., Sønderby, S. K., Nouri, D., Maturana, D., Thoma, M., Battenberg, E., Kelly, J., Fauw, J. D., Heilman, M., Almeida, D. M. d., McFee, B., Weideman, H., Takács, G., Rivaz, P. d., Crall, J., Sanders, G., Rasul, K., Liu, C., French, G., and Degrave, J. (2015). *Lasagne: First release.*

Donahue, J., Hendricks, L. A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. (2014). Long-term Recurrent Convolutional Networks for Visual Recognition and Description. *CoRR*, abs/1411.4389.

Duan, Y., Andrychowicz, M., Stadie, B., Ho, O. J., Schneider, J., Sutskever, I., Abbeel, P., and Zaremba, W. (2017). One-shot imitation learning. In *Advances in neural information processing systems*, pages 1087–1098.

D'Orazio, T., Marani, R., Renò, V., and Cicirelli, G. (2016). Recent trends in gesture recognition: how depth data has improved classical approaches. *Image and Vision Computing*, pages –.

Escalera, S., Baró, X., Gonzalez, J., Bautista, M. A., Madadi, M., Reyes, M., Ponce-López, V., Escalante, H. J., Shotton, J., and Guyon, I. (2014). Chalearn looking at people challenge 2014: Dataset and results. In *Computer Vision-ECCV 2014 Workshops*, pages 459–473. Springer.

Fei-Fei, L., Fergus, R., and Perona, P. (2006). One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611.

Franco, H., Weintraub, M., and Cohen, M. (1997). Context modeling in a hybrid HMM-neural net speech recognition system. In *Proceedings of International Conference on Neural Networks (ICNN'97)*, volume 4, pages 2089–2092 vol.4.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202.

Gers, F. A. and Schmidhuber, E. (2001). LSTM recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340.

Gers, F. A., Schmidhuber, J., and Cummins, F. (2000). Learning to Forget: Continual Prediction with LSTM. *Neural Computation*, 12(10):2451–2471.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. (2006). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM.

Graves, A. and Jaitly, N. (2014). Towards End-to-end Speech Recognition with Recurrent Neural Networks. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1764–II–1772, Beijing, China. JMLR.org.

Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE.

Graves, A. and Schmidhuber, J. (2009). Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems*, pages 545–552.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Hahnloser, R. H. R., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J., and Seung, H. S. (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405:947.

Hariharan, B. and Girshick, R. (2017). Low-Shot Visual Recognition by Shrinking and Hallucinating Features. In *The IEEE International Conference on Computer Vision (ICCV)*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., and Kingsbury, B. (2012). Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006a). A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 18(7):1527–1554.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006b). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554. PMID: 16764513.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507.

Hinton, G. E. and Zemel, R. S. (1994). Autoencoders, Minimum Description Length and Helmholtz Free Energy. In Cowan, J. D., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems 6*, pages 3–10. Morgan-Kaufmann.

Hochreiter, S. (1998). The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 6(2):107–116.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Hochreiter, S., Younger, A. S., and Conwell, P. R. (2001). Learning to Learn Using Gradient Descent. In Dorffner, G., Bischof, H., and Hornik, K., editors, *Artificial Neural Networks — ICANN 2001*, pages 87–94, Berlin, Heidelberg. Springer Berlin Heidelberg.

Hodgkin, A. L. and Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500–544.

Hoffer, E. and Ailon, N. (2014). Deep metric learning using Triplet network. *arXiv:1412.6622 [cs, stat]*. arXiv: 1412.6622.

Huang, G., Liu, Z., Maaten, L. v. d., and Weinberger, K. Q. (2017). Densely Connected Convolutional Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269.

Ibañez, R., Soria, A., Teyseyre, A., and Campo, M. (2014). Easy gesture recognition for Kinect. *Advances in Engineering Software*, 76:171–180.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Jaccard, P. (1902). Lois de distribution florale dans la zone alpine. *Bulletin de la SociéteVaudoise des Sciences Naturelles*, 38:69–130.

Jacob, M. G. and Wachs, J. P. (2014). Context-based hand gesture recognition for the operating room. *Pattern Recognition Letters*, 36:196–203.

Jaderberg, M., Simonyan, K., Zisserman, A., and kavukcuoglu, k. (2015). Spatial Transformer Networks. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 2017–2025. Curran Associates, Inc.

Kelly, D., Donald, J. M., and Markham, C. (2009). Evaluation of threshold model HMMS and Conditional Random Fields for recognition of spatiotemporal gestures in sign language. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 490–497. IEEE.

Kim, J. H., Thang, N. D., and Kim, T. S. (2009). 3-D hand motion tracking and gesture recognition using a data glove. In *2009 IEEE International Symposium on Industrial Electronics*, pages 1013–1018.

Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*. arXiv: 1412.6980.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Kläser, A., Marszalek, M., and Schmid, C. (2008). A Spatio-Temporal Descriptor Based on 3d-Gradients. In *Proceedings of the British Machine Vision Conference 2008, Leeds, UK, September 2008*, pages 1–10.

Koch, G., Zemel, R., and Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2.

Koller, O., Forster, J., and Ney, H. (2015). Continuous sign language recognition: Towards large vocabulary statistical recognition systems handling multiple signers. *Computer Vision and Image Understanding*, 141:108–125.

Koller, O., Ney, H., and Bowden, R. (2016a). Deep Hand: How to Train a CNN on 1 Million Hand Images When Your Data is Continuous and Weakly Labelled. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3793–3802.

Koller, O., Zargaran, S., and Ney, H. (2017). Re-sign: Re-aligned end-to-end sequence modelling with deep recurrent cnn-hmms. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3416–3424, Honolulu, HI, USA.

Koller, O., Zargaran, S., Ney, H., and Bowden, R. (2016b). Deep Sign: Hybrid CNN-HMM for Continuous Sign Language Recognition. In *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*.

Konecny, J. and Hagara, M. (2014). One-Shot-Learning Gesture Recognition using HOG-HOF Features. *Journal of Machine Learning Research*, 15:2513–2532.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kurakin, A., Zhang, Z., and Liu, Z. (2012). A real time system for dynamic hand gesture recognition with a depth sensor. In *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, pages 1975–1979. IEEE.

Lake, B. M., Lee, C.-y., Glass, J. R., and Tenenbaum, J. B. (2014). One-shot learning of generative speech concepts. In *Proceedings of the 36th Annual Meeting of the Cognitive Science Society, CogSci 2014, Quebec City, Canada, July 23-26, 2014*.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Lamere, P., Kwok, P., Gouvêa, R., Raj, B., Singh, R., Walker, W., Warmuth, M., Wolf, P., and Laboratories, S. M. (2003). *The CMU SPHINX-4 Speech Recognition System. Unpublished manuscript. Retrieved from www.cs.cmu.edu/Èrsingh/homepage/papers/icassp03sphinx4_2.pdf*.

Laptev, I. (2005). On Space-Time Interest Points. *International Journal of Computer Vision*, 64(2-3):107–123.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, H.-K. and Kim, J. H. (1999). An HMM-based threshold model approach for gesture recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(10):961–973.

Lee, J., Choi, H., Park, D., Chung, Y., Kim, H.-Y., and Yoon, S. (2016). Fault Detection and Diagnosis of Railway Point Machines by Sound Analysis. *Sensors*, 16(4).

Lehtinen, J., Munkberg, J., Hasselgren, J., Laine, S., Karras, T., Aittala, M., and Aila, T. (2018). Noise2noise: Learning Image Restoration without Clean Data. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2971–2980, Stockholmsmässan, Stockholm Sweden. PMLR.

Levenshtein, V. I. (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707.

Li, H. and Greenspan, M. (2011). Model-based segmentation and recognition of dynamic gestures in continuous video streams. *Pattern Recognition*, 44(8):1614–1628.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the 30th International Conference on Machine Learning (workshop)*, volume 30, page 3.

Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605.

McCallum, A., Freitag, D., and Pereira, F. C. N. (2000). Maximum Entropy Markov Models for Information Extraction and Segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, pages 591–598.

McCloskey, M. and Cohen, N. J. (1989). Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. volume 24 of *Psychology of Learning and Motivation*, pages 109 – 165. Academic Press.

Miranda, L., Vieira, T., Martínez, D., Lewiner, T., Vieira, A. W., and M. Campos, M. F. (2014). Online gesture recognition from pose kernel learning and decision forests. *Pattern Recognition Letters*, 39:65–73.

Mitchell, T. M. (1980). The need for biases in learning generalizations. Technical report.

Morency, L.-P., Quattoni, A., and Darrell, T. (2007). Latent-dynamic discriminative models for continuous gesture recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE.

Morgan, N. and Bourlard, H. (1995). An Introduction to the Hybrid HMM/Connectionist Approach. *Signal Processing Magazine, IEEE*, 12(3):24–42.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA. Omnipress.

Neverova, N., Wolf, C., Taylor, G., and Nebout, F. (2016). ModDrop: Adaptive Multi-Modal Gesture Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(8):1692–1706.

Neverova, N., Wolf, C., Taylor, G. W., and Nebout, F. (2014). Multi-scale deep learning for gesture detection and localization. In *Computer Vision-ECCV 2014 Workshops*, pages 474–490. Springer.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318, Atlanta, Georgia, USA. PMLR.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pei, W., Tax, D. M., and van der Maaten, L. (2016). Modeling time series similarity with siamese recurrent networks. *arXiv preprint arXiv:1603.04713*.

Pigou, L., van den Oord, A., Dieleman, S., Van Herreweghe, M., and Dambre, J. (2016). Beyond Temporal Pooling: Recurrence and Temporal Convolutions for Gesture Recognition in Video. *International Journal of Computer Vision*, pages 1–10.

Pisoni, D. B. (1988). Speech and Speaker Recognition edited by M. R. Schroeder. *The Journal of the Acoustical Society of America*, 84(1):457–458.

Pizer, S. M., Amburn, E. P., Austin, J. D., Cromartie, R., Geselowitz, A., Greer, T., Romeny, B. T. H., and Zimmerman, J. B. (1987). Adaptive Histogram Equalization and Its Variations. *Comput. Vision Graph. Image Process.*, 39(3):355–368.

Plappert, M., Mandery, C., and Asfour, T. (2017). Learning a bidirectional mapping between human whole-body motion and natural language using deep recurrent neural networks. *arXiv preprint arXiv:1705.06400*.

Rabiner, L. and Juang, B. (1986). An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3(1):4–16.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Ravanelli, M., Brakel, P., Omologo, M., and Bengio, Y. (2017). Improving Speech Recognition by Revising Gated Recurrent Units. In *Proc. Interspeech 2017*, pages 1308–1312.

Rezende, D., Shakir, Danihelka, I., Gregor, K., and Wierstra, D. (2016). One-Shot Generalization in Deep Generative Models. In Balcan, M. F. and Weinberger, K. Q., editors,

Proceedings of The 33rd International Conference on Machine Learning, volume 48, pages 1521–1529, New York, New York, USA. PMLR.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham. Springer International Publishing.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. (2016). Meta-Learning with Memory-Augmented Neural Networks. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pages 1842–1850, New York, New York, USA. PMLR.

Schmidhuber, J. (1987). Evolutionary principles in self-referential learning. on learning now to learn: The meta-meta-meta...-hook. Diploma thesis, Technische Universitat Munchen, Germany.

Schreiber, J. (2018). pomegranate: Fast and Flexible Probabilistic Modeling in Python. *Journal of Machine Learning Research*, 18(164):1–6.

Seddik, B., Gazzah, S., Chateau, T., and Amara, N. E. B. (2014). Augmented skeletal joints for temporal segmentation of sign language actions. In *Image Processing, Applications and Systems Conference (IPAS), 2014 First International*, pages 1–6. IEEE.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. (2017). Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *ICLR*.

Shi, Z. and Kim, T. K. (2017). Learning and Refining of Privileged Information-Based RNNs for Action Recognition from Depth Sequences. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4684–4693.

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. (2011). Real-time human pose recognition in parts from single depth images. In *CVPR 2011*, pages 1297–1304.

Simonyan, K. and Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs]*. arXiv: 1409.1556.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Training Very Deep Networks. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 2377–2385. Curran Associates, Inc.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Sutton, C. (2012). An Introduction to Conditional Random Fields. *Foundations and Trends® in Machine Learning*, 4(4):267–373.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.

Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708.

Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Thrun, S. (1998). Lifelong learning algorithms. In Thrun, S. and Pratt, L., editors, *Learning to Learn*, pages 181–209. Springer US, Boston, MA.

Toshev, A. and Szegedy, C. (2014). DeepPose: Human Pose Estimation via Deep Neural Networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1653–1660, Columbus, OH, USA. IEEE.

Tripathi, K. and Nandi, N. B. G. (2015). Continuous Indian Sign Language Gesture Recognition and Sentence Formation. *Procedia Computer Science*, 54:523–531.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *J. Mach. Learn. Res.*, 11:3371–3408.

Vintsyuk, T. K. (1972). Speech discrimination by dynamic programming. *Cybernetics*, 4(1):52–57.

Vinyals, O., Bengio, S., and Kudlur, M. (2016a). Order Matters: Sequence to sequence for sets. In *Proceedings of the International Conference on Learning Representations (ICLR)*. arXiv: 1511.06391.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., and others (2016b). Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638.

Wang, H., Chai, X., Hong, X., Zhao, G., and Chen, X. (2016). Isolated Sign Language Recognition with Grassmann Covariance Matrices. *ACM Trans. Access. Comput.*, 8(4):14:1–14:21.

Wang, H., Kläser, A., Schmid, C., and Liu, C.-L. (2013). Dense Trajectories and Motion Boundary Descriptors for Action Recognition. *International Journal of Computer Vision*, 103(1):60–79.

Wanqing Li, Zhengyou Zhang, and Zicheng Liu (2008). Expandable Data-Driven Graphical Modeling of Human Actions Based on Salient Postures. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(11):1499–1510.

Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339 – 356.

Willmore, B., Watters, P. A., and Tolhurst, D. J. (2000). A Comparison of Natural-Image-Based Models of Simple-Cell Coding. *Perception*, 29(9):1017–1040.

Woodward, M. and Finn, C. (2017). Active one-shot learning. *arXiv preprint arXiv:1702.06559*.

Wu, D., Pigou, L., Kindermans, P. J., Le, N. D. H., Shao, L., Dambre, J., and Odobez, J. M. (2016). Deep Dynamic Neural Networks for Multimodal Gesture Segmentation and Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(8):1583–1597.

Wu, D. and Shao, L. (2014). Multimodal dynamic networks for gesture recognition. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 945–948. ACM.

Wu, D., Zhu, F., and Shao, L. (2012). One shot learning gesture recognition from RGBD images. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 7–12.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A. C., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *CoRR*, abs/1502.03044.

Yang, H.-D. and Lee, S.-W. (2011). Combination of manual and non-manual features for sign language recognition based on conditional random field and active appearance model. In *Machine Learning and Cybernetics (ICMLC), 2011 International Conference on*, volume 4, pages 1726–1731. IEEE.

Yin, Y. and Davis, R. (2014). Real-time continuous gesture recognition for natural human-computer interaction. In *Visual Languages and Human-Centric Computing (VL/HCC), 2014 IEEE Symposium on*, pages 113–120. IEEE.

Young, S. J. and Young, S. (1994). The HTK Hidden Markov Model Toolkit: Design and Philosophy. *Entropic Cambridge Research Laboratory, Ltd*, 2:2–44.

Zagoruyko, S. and Komodakis, N. (2016). Wide Residual Networks. In *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *CoRR*, abs/1611.03530.

Zhu, F., Shao, L., Xie, J., and Fang, Y. (2016). From handcrafted to learned representations for human action recognition: A survey. *Image and Vision Computing*.

Appendix A

DeviSign 2014 dataset splits

The following randomly generated training/validation/testing split was used for all experiments:

training classes: 0, 3, 4, 5, 6, 9, 10, 11, 13, 15, 16, 17, 19, 26, 29, 30, 31, 32, 33, 34, 35, 37, 38, 42, 43, 45, 46, 52, 53, 55, 61, 64, 66, 67, 70, 71, 74, 75, 79, 83, 84, 85, 94, 96, 100, 101, 103, 104, 108, 113, 114, 117, 118, 121, 129, 130, 132, 136, 139, 141, 142, 143, 148, 151, 152, 155, 158, 159, 160, 163, 166, 167, 168, 170, 172, 174, 175, 177, 178, 180, 181, 183, 184, 185, 186, 191, 193, 197, 199, 200, 205, 212, 215, 216, 217, 218, 222, 223, 224, 226, 227, 228, 231, 234, 236, 238, 239, 242, 243, 245, 247, 248, 250, 251, 252, 253, 254, 255, 257, 258, 259, 260, 263, 266, 268, 269, 270, 273, 277, 278, 279, 280, 281, 282, 284, 285, 286, 287, 289, 290, 292, 293, 297, 299, 300, 302, 303, 304, 305, 306, 308, 311, 312, 313, 314, 315, 319, 321, 327, 329, 330, 331, 332, 333, 334, 335, 336, 338, 339, 340, 343, 347, 348, 350, 351, 352, 356, 361, 362, 363, 364, 365, 366, 368, 370, 372, 373, 379, 380, 381, 383, 385, 387, 389, 390, 391, 392, 395, 397, 401, 402, 405, 412, 414, 416, 418, 419, 420, 422, 424, 425, 428, 430, 432, 434, 435, 437, 439, 440, 441, 442, 443, 444, 446, 447, 448, 449, 450, 456, 457, 460, 463, 467, 469, 470, 472, 475, 478, 480, 483, 484, 485, 486, 492, 493, 498, 502, 503, 507, 508, 510, 512, 513, 515, 517, 518, 520, 524, 526, 527, 529, 533, 536, 542, 546, 547, 548, 550, 551, 552, 554, 555, 556, 557, 559, 560, 562, 563, 564, 568, 574, 575, 577, 578, 579, 580, 581, 583, 584, 585, 586, 588, 590, 592, 596, 597, 599, 601, 603, 604, 605, 609, 610, 615, 618, 621, 622, 625, 626, 628, 629, 632, 634, 635, 638, 639, 640, 641, 642, 643, 644, 646, 648, 651, 654, 657, 658, 662, 664, 667, 669, 676, 677, 678, 684, 685, 688, 690, 692, 693, 695, 696, 704, 706, 707, 708, 710, 712, 713, 717, 718, 722, 723, 729, 739, 743, 745, 746, 749, 755, 756, 757, 758, 759, 762, 768, 771, 772, 774, 776, 777, 778, 779, 782, 783, 784, 785, 786, 792, 793, 796, 797, 801, 804, 805, 806, 808, 809, 810, 814, 815, 816, 818, 826, 833, 834, 839, 840, 841, 844, 845, 846, 847, 848, 849, 850, 853, 855, 857, 859, 862, 863, 864, 865, 869, 872, 874, 877, 878, 881, 883, 884, 885, 886, 888, 890, 892, 893, 894, 898, 900, 902, 903, 906, 908, 910, 911, 912, 914, 916, 917, 918, 919, 920, 923, 925, 926, 931, 932, 934, 936, 937, 940, 943, 944, 945, 946, 948, 950, 951, 954, 955, 956, 957, 959, 961, 963, 964, 968, 972, 973, 974, 975, 976, 978, 979, 980, 983, 984, 986, 987, 988, 989, 990, 993, 995, 997, 998, 999, 1001, 1003, 1012, 1013, 1014, 1017, 1018, 1020, 1021, 1024, 1028, 1031, 1033, 1034, 1035, 1037, 1038, 1040, 1044, 1045, 1046, 1048, 1050, 1051, 1053, 1056, 1059, 1060, 1064, 1068, 1069, 1070, 1072, 1074, 1075, 1076, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1087, 1088, 1090, 1091, 1094, 1097, 1098, 1102, 1107, 1111, 1113, 1114, 1116, 1118, 1119, 1121, 1122, 1124, 1125, 1126, 1128, 1129, 1131, 1132, 1133, 1137, 1138, 1140, 1144, 1146, 1150, 1152, 1154, 1155, 1156, 1157, 1162, 1163, 1166, 1167, 1168, 1169, 1170, 1172, 1173, 1178, 1179, 1182, 1183, 1184, 1188, 1189, 1190, 1192, 1195, 1196, 1197, 1199, 1200, 1204, 1205, 1206, 1207, 1208, 1209, 1214, 1215, 1218, 1220, 1222, 1224, 1225, 1226, 1228, 1230, 1231, 1233, 1234, 1237, 1238, 1239, 1240, 1242, 1243, 1244, 1245, 1247, 1248, 1250, 1251, 1253, 1261, 1262, 1264, 1265, 1267, 1269, 1270, 1271, 1272, 1274, 1277, 1281, 1284, 1287, 1288, 1291, 1293, 1294, 1296, 1298, 1299, 1301, 1302, 1303, 1305, 1306, 1312, 1315, 1317, 1319, 1321, 1322, 1323, 1324, 1325, 1326, 1327, 1330, 1331, 1336, 1337, 1341, 1347, 1348, 1349, 1354, 1355, 1361, 1364, 1365, 1373, 1375, 1376, 1379, 1381, 1382, 1383, 1388, 1389, 1390, 1391, 1392, 1395, 1396, 1398, 1403, 1407, 1411, 1419, 1420, 1422, 1424, 1426, 1427, 1428, 1429, 1432, 1435, 1437, 1438, 1440, 1441, 1445, 1447, 1450, 1451, 1452, 1453, 1454, 1459, 1460, 1461, 1462, 1464, 1465, 1467, 1468, 1469, 1473, 1474, 1477, 1480, 1481, 1483, 1485, 1488, 1489, 1491, 1492, 1496, 1497, 1498, 1502, 1503, 1505, 1506, 1507, 1508, 1516, 1519, 1520, 1524, 1526, 1527, 1528, 1530, 1532, 1533, 1534, 1535, 1542, 1543, 1544, 1545, 1547, 1548, 1549, 1550, 1551, 1554, 1555, 1562, 1563, 1564, 1566, 1567, 1568, 1569, 1571, 1575, 1577, 1578, 1582, 1583, 1584, 1585, 1586, 1588, 1589, 1591, 1593, 1594, 1595, 1596, 1598, 1599, 1600, 1602, 1603, 1605, 1606, 1607, 1611, 1617, 1618, 1619, 1621, 1622, 1623, 1624, 1625, 1629, 1631, 1633, 1634, 1637, 1639, 1640, 1643, 1646, 1647, 1650, 1652, 1653, 1655, 1657, 1658, 1660, 1662, 1663, 1666, 1668, 1669, 1670, 1671, 1672, 1675, 1676, 1677, 1678, 1679, 1680, 1682, 1683, 1685, 1687, 1688, 1692, 1693, 1694, 1695, 1698, 1702, 1704, 1707, 1708, 1710, 1711, 1712, 1717, 1718, 1720, 1722, 1723, 1724, 1725, 1726, 1730, 1734, 1736, 1737, 1738, 1739, 1741, 1745, 1746, 1756, 1759, 1760, 1762, 1763, 1764, 1767, 1770, 1771, 1772, 1775, 1776, 1777, 1781, 1782, 1783, 1785, 1786, 1787, 1788, 1789, 1791, 1793, 1795, 1797, 1799, 1800, 1802, 1807, 1808, 1809, 1810, 1816, 1818, 1819, 1820, 1821, 1823, 1824, 1825, 1826, 1827, 1828, 1830, 1831, 1833, 1841, 1842, 1843, 1844, 1846, 1848, 1849, 1851, 1854, 1855, 1857, 1858, 1859, 1862, 1863, 1865, 1867, 1868, 1870, 1872, 1874, 1875, 1876, 1884, 1885, 1886, 1887, 1892, 1893, 1895, 1897, 1898, 1899, 1900, 1902, 1904, 1905, 1907, 1908, 1910, 1914, 1917, 1921, 1924, 1925, 1926, 1927, 1928, 1930, 1931, 1932, 1934, 1935, 1936, 1943, 1946, 1947, 1948, 1949, 1950, 1955, 1956, 1961, 1962, 1966, 1970, 1972, 1974, 1975, 1976, 1977, 1978, 1980, 1981, 1983, 1985, 1986, 1987, 1990, 1991, 1992, 1993, 1995, 1999

validation classes: 1, 2, 7, 14, 20, 21, 22, 25, 36, 39, 44, 50, 54, 57, 62, 63, 65, 68, 80, 82, 86, 87, 88, 90, 92, 93, 95, 97, 99, 105, 106, 109, 110, 111, 112, 115, 116, 120, 122, 126, 127, 131, 134, 137, 138, 140, 144, 147, 150, 153, 156, 157, 164, 165, 169, 171, 187, 188, 194, 196, 201, 202, 204, 214, 220, 225, 229, 230, 235, 244, 256, 261, 275, 276, 283, 288, 296, 298, 301, 307, 309, 317, 318, 320, 323, 326, 342, 344, 346, 349, 353, 354, 357, 358, 367, 371, 375, 376, 378, 386, 393, 396, 400, 408, 411, 415, 417, 421, 423, 427, 429, 431, 436, 445, 451, 453, 454, 458, 461, 464, 465, 466, 468, 471, 473, 482, 487, 488, 489, 491, 495, 500, 501, 504, 505, 506, 509, 511, 514, 516, 521, 522, 530, 531, 532, 534, 537, 543, 544, 558, 561, 565, 566, 570, 571, 573, 576, 582, 587, 602, 607, 611, 614, 616, 617, 624, 630, 633, 647, 650, 652, 653, 656, 660, 668, 671, 675, 680, 681, 682, 683, 686, 687, 689, 699, 700, 702, 705, 714, 715, 716, 719, 720, 721, 725, 727, 728, 730, 731, 733, 735, 737, 738, 741, 744, 752, 753, 761, 766, 767, 770, 775, 788, 789, 791, 794, 798, 799, 800, 802, 807, 811, 813, 817, 820, 821, 824, 825, 828, 830, 831, 832, 837, 842, 843, 851, 856, 860, 866, 867, 870, 871, 875, 880, 882, 887, 889, 895, 909, 915, 921, 924, 927, 930, 938, 941, 947, 952, 960, 962, 965, 966, 967, 969, 971, 977, 982, 996, 1000, 1002, 1004, 1006, 1007, 1008, 1009, 1015, 1016, 1019, 1023, 1026, 1027, 1029, 1030, 1032, 1036, 1039, 1042, 1043, 1052, 1057, 1058, 1065, 1066, 1067, 1071, 1073, 1077, 1089, 1103, 1104, 1110, 1112, 1115, 1117, 1120, 1134, 1135, 1139, 1141, 1142, 1143, 1148, 1161, 1164, 1175, 1180, 1185, 1186, 1191, 1193, 1198, 1202, 1211, 1212, 1213, 1216, 1221, 1223, 1232, 1249, 1252, 1273, 1275, 1276, 1280, 1285, 1289, 1292, 1295, 1304, 1307, 1310, 1311, 1314, 1316, 1318, 1320, 1329, 1333, 1342, 1343, 1344, 1346, 1350, 1357, 1358, 1360, 1363, 1366, 1368, 1369, 1371, 1372, 1374, 1377, 1378, 1380, 1386, 1393, 1399, 1404, 1405, 1406, 1410, 1414, 1415, 1416, 1418, 1421, 1423, 1431, 1444, 1449, 1456, 1458, 1471, 1472, 1476, 1478, 1486, 1504, 1514, 1517, 1521, 1531, 1538, 1559, 1570, 1572, 1576, 1580, 1590, 1608, 1609, 1610, 1614, 1616, 1620, 1626, 1627, 1628, 1644, 1648, 1649, 1651, 1654, 1659, 1661, 1664, 1667, 1674, 1681, 1686, 1689, 1690, 1699, 1701, 1703, 1705, 1709, 1719, 1728, 1729, 1740, 1742, 1744, 1750, 1751, 1752, 1758, 1766, 1768, 1769, 1773, 1779, 1780, 1796, 1798, 1801, 1804, 1805, 1812, 1814, 1817, 1822, 1829, 1832, 1836, 1837, 1839, 1845, 1850, 1861, 1864, 1866, 1873, 1877, 1878, 1879, 1882, 1883, 1891, 1903, 1906, 1909, 1911, 1912, 1913, 1919, 1920, 1922, 1923, 1937, 1939, 1941, 1942, 1951, 1953, 1954, 1957, 1960, 1964, 1967, 1969, 1984, 1988, 1989, 1994, 1996, 1997

testing classes: 8, 12, 18, 23, 24, 27, 28, 40, 41, 47, 48, 49, 51, 56, 58, 59, 60, 69, 72, 73, 76, 77, 78, 81, 89, 91, 98, 102, 107, 119, 123, 124, 125, 128, 133, 135, 145, 146, 149, 154, 161, 162, 173, 176, 179, 182, 189, 190, 192, 195, 198, 203, 206, 207, 208, 209, 210, 211, 213, 219, 221, 232, 233, 237, 240, 241, 246, 249, 262, 264, 265, 267, 271, 272, 274, 291, 294, 295, 310, 316, 322, 324, 325, 328, 337, 341, 345, 355, 359, 360, 369, 374, 377, 382, 384, 388, 394, 398, 399, 403, 404, 406, 407, 409, 410, 413, 426, 433, 438, 452, 455, 459, 462, 474, 476, 477, 479, 481, 490, 494, 496, 497, 499, 519, 523, 525, 528, 535, 538, 539, 540, 541, 545, 549, 553, 567, 569, 572, 589, 591, 593, 594, 595, 598, 600, 606, 608, 612, 613, 619, 620, 623, 627, 631, 636, 637, 645, 649, 655, 659, 661, 663, 665, 666, 670, 672, 673, 674, 679, 691, 694, 697, 698, 701, 703, 709, 711, 724, 726, 732, 734, 736, 740, 742, 747, 748, 750, 751, 754, 760, 763, 764, 765, 769, 773, 780, 781, 787, 790, 795, 803, 812, 819, 822, 823, 827, 829, 835, 836, 838, 852, 854, 858, 861, 868, 873, 876, 879, 891, 896, 897, 899, 901, 904, 905, 907, 913, 922, 928, 929, 933, 935, 939, 942, 949, 953, 958, 970, 981, 985, 991, 992, 994, 1005, 1010, 1011, 1022, 1025, 1041, 1047, 1049, 1054, 1055, 1061, 1062, 1063, 1086, 1092, 1093, 1095, 1096, 1099, 1100, 1101, 1105, 1106, 1108, 1109, 1123, 1127, 1130, 1136, 1145, 1147, 1149, 1151, 1153, 1158, 1159, 1160, 1165, 1171, 1174, 1176, 1177, 1181, 1187, 1194, 1201, 1203, 1210, 1217, 1219, 1227, 1229, 1235, 1236, 1241, 1246, 1254, 1255, 1256, 1257, 1258, 1259, 1260, 1263, 1266, 1268, 1278, 1279, 1282, 1283, 1286, 1290, 1297, 1300, 1308, 1309, 1313, 1328, 1332, 1334, 1335, 1338, 1339, 1340, 1345, 1351, 1352, 1353, 1356, 1359, 1362, 1367, 1370, 1384, 1385, 1387, 1394, 1397, 1400, 1401, 1402, 1408, 1409, 1412, 1413, 1417, 1425, 1430, 1433, 1434, 1436, 1439, 1442, 1443, 1446, 1448, 1455, 1457, 1463, 1466, 1470, 1475, 1479, 1482, 1484, 1487, 1490, 1493, 1494, 1495, 1499, 1500, 1501, 1509, 1510, 1511, 1512, 1513, 1515, 1518, 1522, 1523, 1525, 1529, 1536, 1537, 1539, 1540, 1541, 1546, 1552, 1553, 1556, 1557, 1558, 1560, 1561, 1565, 1573, 1574, 1579, 1581, 1587, 1592, 1597, 1601, 1604, 1612, 1613, 1615, 1630, 1632, 1635, 1636, 1638, 1641, 1642, 1645, 1656, 1665, 1673, 1684, 1691, 1696, 1697, 1700, 1706, 1713, 1714, 1715, 1716, 1721, 1727, 1731, 1732, 1733, 1735, 1743, 1747, 1748, 1749, 1753, 1754, 1755, 1757, 1761, 1765, 1774, 1778, 1784, 1790, 1792, 1794, 1803, 1806, 1811, 1813, 1815, 1834, 1835, 1838, 1840, 1847, 1852, 1853, 1856, 1860, 1869, 1871, 1880, 1881, 1888, 1889, 1890, 1894, 1896, 1901, 1915, 1916, 1918, 1929, 1933, 1938, 1940, 1944, 1945, 1952, 1958, 1959, 1963, 1965, 1968, 1971, 1973, 1979, 1982, 1998

Annexe B

Résumé

En l'espace de 50 ans, le nombre d'interactions entre les humain, les ordinateurs et le reste du monde s'est accru considérablement. L'époque des terminaux physiques est un lointain souvenir, lorsqu'une simple interface textuelle servait à envoyer des commandes et recevoir une réponse d'un appareil qui n'était guère plus qu'un automate. Avec la multiplication des capteurs, des interfaces, et l'augmentation considérable de la puissance de calcul, les ordinateur ont progressivement intégré la majorité des activité humaines. Une branche de cette expansion nous semble particulièrement fascinante : l'apprentissage statistique qui vise à équiper les ordinateurs de modèles d'apprendre et de généraliser des concepts à partir d'exemples. L'apprentissage statistique permets de supplanter l'ajout de fonctionnalités par la programmation en imitant l'acquisition de facultés nouvelles chez les humains.

Diverses applications de ce domaine sont désormais accessibles aux grand public : reconnaissance vocale pour la dictée ou pour des assistant vocaux, reconnaissance d'écriture, etc. L'apprentissage statistique traite aussi des tâches d'expertise dans une variété de domaines, par exemple : la médecine avec la segmentation d'images radio-graphiques ou microscopiques [Ronneberger et al., 2015], la détection non-invasive de failles mécaniques dans des rails de trains [Lee et al., 2016], le débruitage d'images [Lehtinen et al., 2018] ou encore l'identification biométrique [Taigman et al., 2014].

Cette thèse se concentre principalement sur des données séquentielles, en pratiques des flux d'observations temporelles qui sont annotés d'un labels, ou éventuellement d'une série de labels. Les données séquentielles sont omniprésentes dans nos activités quotidiennes, ce qui motive d'autant l'étude de l'apprentissage statistique sur ces modalités : la parole, l'écriture, les vidéos, des mesures physiques, etc. Afin de définir plus concrètement l'objectif de cette thèse, nous avons sélectionné comme exemple applicatif la reconnaissance de gestes dans des enregistrement de vidéos et de poses corporelles. Le choix de cette tâche est notamment motivé par le caractère multi-modal de ces données d'entrée, avec des représentations vectorielles de taille et de nature très différentes. D'autre part, une reconnaissance de geste efficace pourrait démocratiser un nouveau support de communication dans les interfaces homme-machine.

La prédiction sur des séquences peut adopter différentes formes : dans le cas de la reconnaissance isolée, une séquence d'observations contant un unique geste est soumise au modèle de classification, tandis que pour la reconnaissance continue, le modèle doit détecter les instances de gestes successive éventuellement entrecoupées de blancs depuis un flux d'entrées. Cette seconde tâche peut éventuellement se combiner au découpage et à l'alignement temporel des gestes comme objectif secondaire.

Un second axe de recherche exploré par cette thèse s'intéresse au paradigme d'apprentissage one-shot, c'est à dire "en un coup"[1]. Dans ce paradigme, la base d'apprentissage est réduite à un seul exemples pour de nouvelles classes encore jamais observées, ce qui oblige le modèle à apprendre très rapidement et à généraliser de manière très robuste. Pour illustrer ce mode de fonctionnement, on peut considérer un robot destiné commandé par des gestes spécifiques pour accomplir sa tâche dans un environnement industriel, où l'utilisation de commandes vocale ou bien d'un terminal de contrôle physique n'est pas toujours possible. L'apprentissage one-shot permettrait ici de laisser l'utilisateur décider lui-même des gestes et d'entrainer rapidement le robot à l'aide d'un unique exemple pour chaque commande.

B.1 Aperçu de la Reconnaissance de Gestes

Entre les gestes courants comme pointer du doigt et les directives militaires, la communication gestuelle offre un vecteur de communication particulièrement flexible et universellement utilisé dans la vie de tous les jours. Par conséquent, il semble naturel d'essayer d'agrémenter aussi les interface homme-machine d'un système à base de geste afin de compléter les solutions existante pour la voix ou l'écriture. La littérature dans ce domaine propose à titre d'exemples les jeux-vidéos [Ibañez et al., 2014], le contrôle sans-contacte d'un ordinateur dans une salle d'opération, ou encore le contrôle d'un robot industriel. D'un point de vue scientifique, la reconnaissance de geste constitue un sujet particulièrement attractif. En effet, la niveau d'expertise disponible sur l'analyse est moins étendu que pour d'autres exemples de reconnaissance séquentielle comme la parole ou l'écriture, ce qui rend plus prépondérant l'apprentissage automatique de représentations par le modèle pour compenser. En outre, l'apparence la forme et le mouvement jouent un rôle combiné pour donner une signification aux gestes, ce qui nécessite d'extraire des facteurs de variations complexes depuis des source multimodales.

La reconnaissance en continue de gestes vise à détecter et reconnaitre des segments du flux d'observations portant une valeur sémantique particulière. La signification du geste découle d'une combinaison de poses statiques ou de mouvements du corps, des bras, des mains ou même une expression faciale. Les gestes sont produits par individu dont on suppose qu'il cherche activement à transmettre un message, par opposition à des formes d'interprétations passives comme la reconnaissance d'activité où le sujet n'est à priori pas coopératif. Par conséquent, il est raisonnable de considérer ici un sujet centré face à la caméra et visible sans occlusions majeures, dans des conditions d'enregistrement stables.

[1] le domaine étant relativement méconnu, aucune traduction de référence n'existe pour cette expression

La reconnaissance de gestes s'appuie typiquement sur trois type de modalités : l'image, la profondeur spatiale ou la pose corporelle repérée dans l'espace (Figure B.1). La démocratisation des

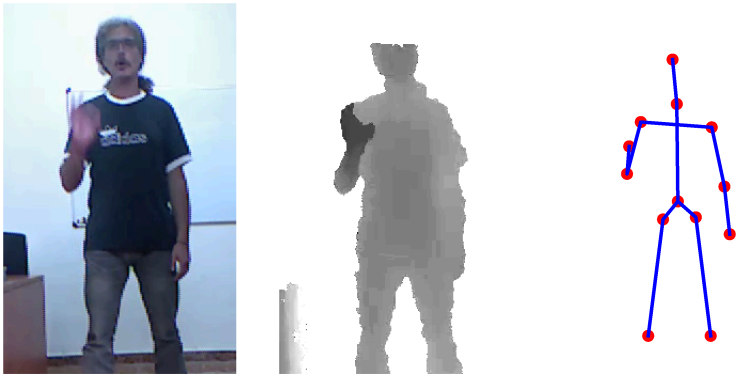


Figure B.1 – Exemples d'observations en reconnaissance de pose : vidéo, carte de profondeur et régression de la pose corporelles sur la base de données Montalbano v2 [Escalera et al., 2014]

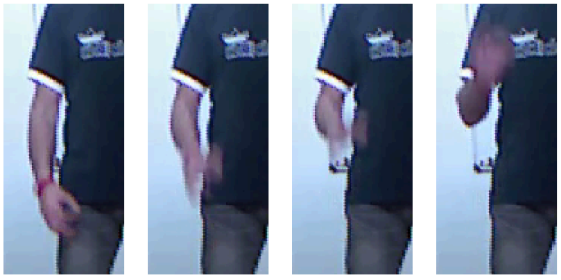
caméras avec capteur de profondeur (en particulier la Kinect) ont grandement facilité l'acquisition de jeux de données sur les gestes. La profondeur permet de lever l'ambiguïté sur la connectivité réelle entre deux points d'un objet sur des pixels adjacents. Par exemple, les effets d'ombres ou les nuances de couleurs peuvent rendre difficile la segmentation entre le torse et un bras situé devant. Les capteurs de profondeur sont en outre insensibles aux variations d'éclairages et au flou cinétique ce qui rend leur données plus fiables. Ces dernière sont en particulier utilisée pour faire la régression de la position du corps dans l'espace [D'Orazio et al., 2016], une information cruciale pour reconnaitre des gestes. Des travaux plus anciens ont eu recours à des gants colorés pour faciliter la localisation des la main, ou encore au port d'accéléromètres et de gyroscopes par le sujet mais ces solutions ne se prêtent pas à un usage hors laboratoire. Toujours au prix de la facilité d'utilisation, la technique de motion capture a été utilisée plus récemment [Braffort et al., 2015] pour constituer une base de langage des signes, augmentant ainsi significativement la qualité des données de position.

Les données vidéo sont généralement de qualité moyenne : d'une résolution de 320×240 à 720×576 avec 10 à 20 images par seconde. Lorsqu'elle sont disponibles, les cartes de profondeur et la régression de la pose sont toutes deux synchronisées avec les images en couleur. Le haut du corps du sujet est souvent centré dans la vue de manière à rester parfaitement visible. Néanmoins, quelques variables demeurent : la position et la nature de l'éclairage, la distance entre le sujet et la caméra, l'arrière-plan et les objets environnants, etc. Les sujets eux-mêmes apportent des variations supplémentaires avec notamment la couleur de la peau et des vêtements, la taille du corps, les variations dans l'exécution des gestes comme la vitesse ou l'amplitude du mouvement. La Figure B.2 illustre ces sources d'erreurs.

Pour évaluer les modèles et les entrainer sur la reconnaissance de gestes, les bases de données doivent aussi inclure des annotations sur les gestes présent dans les enregistrements. À minima,

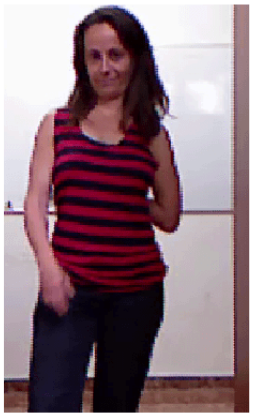
(a) Environnement chargé

(b) Illumination de faible qualité



(c) Flou cinétique

(d) Variété d'apparence et de couleurs (faible contraste, couleur de peau)



(e) Occlusions

Figure B.2 – Sources de variété et d'erreurs dans les observations. Ces échantillons sont issus de la base Montalbano v2 [Escalera et al., 2014]

ces annotations doivent spécifier le ou les labels de gestes tels qu'il apparaissent dans la séquence. Pour évaluer la qualité de la segmentation en plus de la reconnaissance, les annotations peuvent aussi spécifier le découpage temporel du début et de la fin de chaque geste. Il est à noter que ces frontières souffrent généralement d'un certain niveau d'incertitude lié aux mouvements de transition qui délimitent les geste en eux-même.

B.2 Contexte Scientifique et Motivations

B.2.1 Reconnaissance en Continu sur des Séries Temporelles

Le reconnaissance sur des séquences est un cas particulier des problèmes de classification, c'est à dire que l'on cherche à prédire une quantité discrète y étant donnée une observation x. Le caractère continue découle de la structure séquentielle de $x = (x_t)_{1 \leq t \leq T}$ où T représente la longueur de cette séquence. Toutes les observations x_t contiennent le même type de données, en général un vecteur réel pour chaque modalité d'entrée. Dans la majorité des cas, l'échantillonnage temporel est uniforme. Pour les annotations y, on observe trois cas principaux :

- La tâche isolée où chaque séquence d'observation s'associe à un unique label.
- Dans un cas plus général, la séquence comporte M instances de classes successives $(y_k)_{1 \leq k \leq < K}$, par exemple des mots dans un enregistrement audio-phonique. Il s'agit de la formulation la plus couramment utilisé en reconnaissance de la parole et de l'écriture car elle n'implique pas de délimiter précisément l'étalement temporel des instances à détecter. En effet, cet alignement n'est pas nécessairement possible en cas de recouvrements entre les instances.
- Lorsqu'un alignement est néanmoins disponible, la *reconnaissance continue de séquences* défini une classe à chaque instant : $y = (y_t)_{1 \leq t \leq T}$. Cette approche accroit d'une part le niveau de supervision dans l'apprentissage d'un modèle, et permet d'autre part d'évaluer la qualité de l'alignement prédit. Produire ces annotations supplémentaires induit néanmoins un surcout considérable lors de la fabrication d'un jeu de données, et génère parfois des ambiguïtés en présence de phénomènes transitoires.

Cette thèse étudie le troisième cas, qui permet d'analyser plus finement les propriété temporelles d'un modèle. Cette configuration nous permets aussi de comparer plus facilement notre étude et nos résultats à des travaux existants sur la reconnaissance de geste.

Étant donnée les objectifs précisé ci-dessus et les principaux défis à surmonter pour les atteindre, différentes classes de modèles ont émergé au fil des années en particulier dans le cadre de la reconnaissance de la parole et de l'écriture. Ces modèles adoptent approximativement la même logique générale fondée sur une succession d'étapes logiques, tandis que les spécificités sont dues aux variations de méthodes à l'intérieur de ces modules :

Acquisition des données : Cette étape regroupe l'acquisition des échantillons de données d'entrée à chaque instant.

Extraction de facteurs de variation : Ce module comprend généralement deux étape successives. Tout d'abord, une série de transformation globales est appliquée pour obtenir une représentation des données plus commode à manipuler que sous leur forme brute et éventuellement pour augmenter l'invariance à des phénomène dont on sait qu'ils n'affecteront pas la tâche ciblée. Par exemple la normalisation du contraste dans des images, la réduction de dimension par ACP sur de larges vecteurs réels.

Viennent ensuite des extractions de caractéristiques plus pertinentes et spécifiquement conçue pour accomplir la reconnaissance en continu, par exemple les données HOG sur les images, des mesures de vitesses, d'accélération et d'orientation pour des objets dans l'espace, etc. Plus récemment, les avancées du deep learning ont permis d'utiliser des modèles paramétriques (souvent des réseaux de neurones) pour apprendre automatiquement les transformations adaptées pour la tâche via un processus dit d'*apprentissage de représentations*. Bien que ces modèles soient souvent difficile à interpréter, il parviennent souvent à produire dans l'espace de représentation euclidien des notion sémantiques de haut niveau présentes dans les entrée, par exemple la similarité ou la compositionnalité.

Modélisation temporelle : Lorsqu'elle n'est pas fusionnée dans l'étape précédente, un modèle séparé peut servir à capturer les structures temporelles présentent dans les données. Cette thèse se concentre principalement sur les chaînes de Markov Cachés (HMM) et les Réseaux de Neurones Récurrents (RNN), mais de nombreuses autres approches existent dans la littérature.

Il est à noter que cette étape peut être précédée d'une segmentation temporelle explicite produisant des segment de données monolithiques à modéliser, ce qui permet d'utiliser des approches statistiques comme le Bag-of-Words pour étudier les propriété de ces segments. Il n'est cependant pas toujours possible ou même pertinent de procéder à ce découpage auquel cas on préfèrera un traitement en continu des séquences de données.

Classification : Un vaste panel de modèle est disponible pour assurer cette opération. Ce choix dépendra notamment de la nature des représentations ; en effet, une représentation séquentielle demandera un traitement spécifique par rapport à un vecteur de taille fixe. La sélection devra aussi prendre en compte le type de prédictions demandée, c'est à dire uniquement les classe détectées ou aussi leur étendues temporelles.

On notera que la séparation avec le module précédent n'est pas toujours explicite, notamment dans le cas d'un réseau de neurone multi-couches fonctionnant de bout-en-bout ou d'un HMM.

Dans cette thèse, nous étudions plus particulièrement deux modèles temporels de l'état de l'art an reconnaissance continue sur des séquence : Le modèle de chaîne de Markov Cachée (HMM) et les réseaux de neurones récurrents (RNN). Ces deux approches s'articulent autour d'une idée commune : modéliser des phénomènes séquentiels (temporels dans notre étude) par des processus récursifs. Ces deux modèles appartiennent néanmoins à deux école de pensée différentes dans la manière d'implémenter ce modèle.

La HMM s'appuie sur un modèle probabiliste graphique dont la construction s'articule principalement autour des données d'entrée : un ensemble d'hypothèse expertes est formulé pour expliquer le processus sous-jacent qui provoque les phénomènes observés, et résulte en un modèle génératif capable de produire de nouvelle observation artificielles. Ces hypothèses simplifient le modèles et rendent les calculs abordables. Dans le cas de notre étude, ce sont les calculs liés à la détections et la classifications qui sont primordiaux. La HMM est subdivisé entre un modèle des observations instantanée et un modèle de transitions, tous deux entraînés conjointement pour maximiser la vraisemblance des données d'apprentissage. Elle offre un modèle à la fois flexible, robuste et facile à interpréter, ce qui facilite la mise au point et permet éventuellement de mieux comprendre le phénomène étudié. L'apprentissage et l'inférence avec la HMM sont très efficace en raison de l'utilisation de la programmation dynamique qui limite la complexité des calculs. Pour profiter de modèles d'observations plus complexes, [Bourlard and Morgan, 1990] suggère de remplacer ce modèle probabiliste par un modèle discriminatif, en pratique un réseau de neurones. Cette architecture, connu sous le nom d'hybride NN-HMM, dispense le modèle d'apprendre le processus de génération des observations qui n'est pas nécessaire à l'usage de la HMM pour la reconnaissance continue, et qui limite le choix de modèles.

À l'opposé de l'approche probabiliste, les réseaux de Neurones Récurrents suivent une approche largement axée autour de la tâche et des prédictions à produire. L'aspect modélisation, bien que primordial pour la réussite du modèle, n'intervient que pour faciliter l'accomplissement de la reconnaissance continue. L'entraînement des paramètre du réseau de Neurone repose sur l'utilisation d'une fonction objectif dont la maximisation traduit le succès avec lequel un modèle accomplit son objectif. Si l'objectif et la logique diffèrent, l'architecture reste assez comparable à la solution d'hybride NN-HMM. Suite au succès des modèles HMM, et en particulier la version hybride qui fait intervenir les réseaux de neurones, il est donc raisonnable de se demander pourquoi l'approche RNN a tant tardé à rencontrer la réussite qu'on lui connait aujourd'hui. L'une des raisons principale provient de la manière d'appliquer la récurrence dans les RNN standards, qui provoque une atténuation rapide de l'information au cours des itérations, et limite donc la capacité de modélisation à des phénomènes à court-terme. Pour surmonter ce problème, des unités pontées (en référence aux ponts électroniques qui court-circuitent une portion d'un système) ont été proposées. Ces dernière permettent de réduire la profondeur apparente d'un réseau récurrent par rapport au temps et facilite donc la propagation des signaux d'apprentissage ou de prédiction. Cette amélioration se combine à d'autres contributions plus générales dans le domaine des réseaux de neurones et à l'accroissement substantiel de la puissance de calcul disponible pour les faire fonctionner.

B.2.2 Apprentissage "en un coup"

En dépit du nom, l'apprentissage one-shot (en un coup) formalise surtout une méthodologie d'inférence et de test, à l'intérieur de laquelle une étape d'apprentissage occupe une place centrale. L'apprentissage one-shot stipule qu'à l'aide d'uniques exemples issues de classes jamais observées

dans le passé, un modèle doit être capable d'apprendre à classer de nouvelles occurrences des ces catégories.

Le défi central de cette forme de classification réside de toute évidence dans la manière d'apprendre à partir d'un unique exemple. En effet, la plupart des techniques d'apprentissage nécessitant un large ensemble de données statistiquement représentatives de la réalité ne peuvent s'adapter à cette situation extrême. De nouveaux modèles sont dont nécessaire pour exploiter au mieux les exemples d'apprentissage. Des travaux essaie notamment de tirer parti de la robustesse et de la flexibilité des réseaux de neurones pour cette tâche, tout essayant d'éviter les problème de sur-apprentissage. Une partie de cet effort intervient en amont de la phase one-shot, où une large partie de l'apprentissage des paramètres du modèle est effectuée sur une tâche annexe qui réduit considérablement la dépendance sur les exemples observés en one-shot. Cette approche conforte aussi le point de vue qui consiste à présenter l'apprentissage one-shot principalement comme une méthodologie d'évaluation et non d'apprentissage.

L'apprentissage one-shot présente des perspective d'utilisations très intéressante pour la reconnaissance de gestes, et c'est la raison pour laquelle nous avons décidé de l'étudier. En effet, un utilisateur pourrait par exemple programmer un robot assistant pour réagir à ses commandes de manière similaire aux systèmes vocaux actuellement. Cette thèse étudie ce paradigme d'apprentissage et de test dans le cas de gestes isolés, c'est à dire que le modèle doit prédire la classe pour des séquences contenant un seul geste à la fois.

Comparé aux paradigmes d'apprentissage et de classification traditionnels, le one-shot reste jusqu'à aujourd'hui un domaine assez peu étudié, avec néanmoins une activité de recherche croissante. Par exemple, il n'existe à notre connaissance qu'une seule contribution sur l'utilisation de réseaux de neurones pour des données séquentielles [Pei et al., 2016]. Cette situation s'explique notamment par la difficulté de la tâche qui requiert des techniques non-conventionnelles. Les publications récentes [Santoro et al., 2016 ; Vinyals et al., 2016b] ont cependant définit un cadre d'étude et une méthodologie plus précise pour étudier le sujet, ce qui permet de définir des objectifs tangibles à accomplir et de canaliser l'effort de recherche sur ce domaine. Parallèlement, des sujets de recherche connexes comme le apprentissage incrémental, multi-tâche ou le méta-apprentissage [Schmidhuber, 1987] contribuent aussi à ce développement.

B.3 Expériences et Contributions

B.3.1 Reconnaissance de gestes en continu

La reconnaissance en continu sur des séquence possède un état de l'art très fourni avec de nombreuses applications déjà disponibles auprès du grand public. Le domaine reste cependant très actif et continu de s'améliorer ou de se diversifier. Plutôt que de s'investir sur l'amélioration des performances, notre travail porte l'emphase sur l'étude des propriétés de deux modèles de l'état de l'art, l'hybride NN-HMM et les RNN. Avec leur amélioration notables ces dernières années

et leurs performances à l'état-de-l'art, les second ont supplanté leur prédécesseur comme le modèle de facto pour aborder la reconnaissance en continu sur les séquences. Pour autant, les deux modèle partagent un certain nombres de similitudes, notamment un réseau de neurone pour assurer l'apprentissage de représentations à partir des observations. Ils s'appuient tous deux sur un mécanisme de représentation interne latente l'inférence repose dans chaque cas sur une expression récurrente qui alterne entre des transformations linéaires et non-linéaires. Notre travail vise donc à mettre en lumière les différences, similarités, avantages et inconvénients de chaque modèle. Cette étude s'appuie sur un cas d'utilisation concret : la reconnaissance de gestes ; un support d'étude particulièrement intéressant puisque les données associés mélangent différents types de représentations et niveaux de complexité.

Dans un premier temps, le travail de cette thèse a été investi sur la fabrication d'un modèle et d'un environnement de test avec un design modulaire qui assure une comparaison équitable entre les deux modèle. Cette objectif doit s'accomplir sans avoir à faire de compromis sur les capacité d'apprentissage ou d'inférence observées pour chaque modèle optimisé de manière isolée. Pour ce faire, notre étude s'appuie sur un jeu de données de gestes issu d'une compétition [Escalera et al., 2014]. Celle-ci bénéficie d'une méthode d'évaluation rigoureuse et d'un nombre conséquent de publications illustrant notamment l'utilisation des Hybrides NN-HMM et des RNN, ce qui nous permet de valider le fonctionnement optimal de ces modèles dans notre implémentation.

Comme mentionné dans la présentation du domaine, les modèles de reconnaissance séquentielle (et en particulier de gestes), sont découpés en plusieurs modules conceptuels, notamment un dédié a la modélisation temporelle. Notre approche pour une comparaison équitable a commencé par un travail de fusion entre la solution à base d'hybride NN-HMM et de RNN pour tous les autres modules. En limitant les différences et donc les sources variations dans les résultats, il devient plus facile d'analyser les propriétés liées spécifiquement aux modèles temporels. Par ailleurs, le côté historique de l'approche avec les HMM implique que ses implémentations n'ont pas systématiquement bénéficié des dernières évolutions en apprentissage de représentations. La figure B.3 résume l'architecture globale et met en évidence le partage d'un même modèle d'apprentissage de représentations dans les deux approches. On notera que l'architecture utilisée pour le RNN est un modèle bidirectionnel utilisant des unités GRU [Cho et al., 2014]. De son côté, l'architecture HMM repose sur un modèle de type phonème à 5 états semblable à ceux utilisés couramment pour la reconnaissance de paroles. Pour optimiser les méta-paramètres comme le nombre de couches, leur taille, le pas de gradient... nous avons suivi une stratégie opportuniste en optimisant successivement les paramètres qui semblent influencer le plus fortement les performances. La figure n'inclue pas l'acquisition et le pré-traitement des observations ou encore l'extraction de caractéristiques expertes. Là encore, nous appliquons les mêmes opérations pour les deux approches dans un souci d'équité.

Lors de nos premiers essais pour entraîner le modèle de reconnaissance, nous avons observé de multiples problèmes liés au déséquilibre entre la quantité d'observations de gestes et de non-gestes. Ces dernière sont en effet regroupées au sein d'une classe de rejet largement omniprésente dans les enregistrements. Pour les deux approches de modèle temporels, nous utilisons le fait

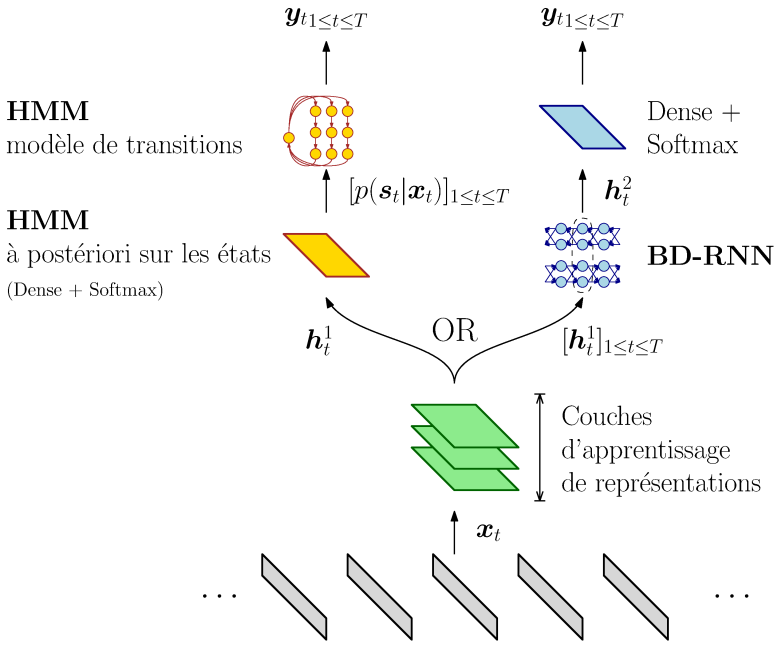


Figure B.3 – Architecture générale des modèles. L'extraction de représentations utilise une couche de convolution temporelle [Pigou et al., 2016] qui permet d'ajouter du contexte à court terme dans la représentation $\boldsymbol{h}_t$, qui ne dépend donc pas uniquement de $\boldsymbol{x}_t$.

que la fonction de coût se décompose en termes pour chaque instant et chaque label pour ajouter un coefficient correcteur :

$$L(y,\tilde{y}) = \sum_{t=1}^{\mathrm{T}} L(y_t,\tilde{y}_t) \tag{B.1}$$

devient ainsi :

$$L'(y,\tilde{y}) = \sum_{t=1}^{\mathrm{T}} \frac{1}{\mathrm{V}\cdot f_{y_t}} L(y_t,\tilde{y}_t) \tag{B.2}$$

où V désigne le nombre de classes[1], et f_k la fréquence d'apparition de la classe k.

Sur des exemples isolés avec un label unique, cette technique revient à ré-échantillonner les exemples uniformément selon les classes. Dans le cas de séquences annotées continument par une série hétérogène de labels, un tel ré-échantillonnage est beaucoup plus délicat à mettre en place car il faudrait découper les séquences de manière à équilibrer les classes, contrairement à la méthode proposée. Aussi bien dans le cas de la méthode hybride NN-HMM que pour le RNN, la solution proposée corrige les problèmes d'apprentissages dûs au déséquilibre des données.

Un second enseignement des tests préliminaires concerne la distribution des erreurs en fonction de la longueur du geste prédit. En effet, les prédictions très courtes du modèle RNN sont presque systématiquement fausse, ce qui signifie qu'une élimination pure et simple en dessous

[1]Nombre de gestes et un non-geste pour le RNN, nombre d'états en comptant l'état de rejet pour la HMM.

d'un seuil de durée est préférable en terme de score. Le phénomène existe aussi pour le modèle HMM, mais de manière moins prononcée. Cet exemple illustre le côté ambivalent de la flexibilité des réseaux de neurones, toute inadéquation entre la fonction de coût et l'objectif réel risque de mener à des prédiction erronées, et ce en dépit des contraindre architecturales imposées au modèle. Ici, la présence de modèles temporels RNN et de convolutions temporelles dans l'apprentissage de représentations n'empêchent pas la prédictions de gestes anormalement courts.

Les performances obtenues par nos modèles sont compilées dans la Table B.1 et montrent que notre implémentation est en accord avec les résultats de l'état de l'art.

Modèle	**Modalité**	**Précision**		**JI**	
		validation	test	validation	test
[Wu et al., 2016] Hybrid NN-HMM	P			0.783	0.779
Hybride NN-HMM	P	0.911	0.912	0.789	0.788
BD-RNN	P	0.921	0.922	0.814	0.811
[Neverova et al., 2016] DNN	P				0.831
Hybride NN-HMM	(P)V	0.890	0.890	0.748	0.745
BDRNN	(P)V	0.909	0.909	0.791	0.789
[Neverova et al., 2016] CNN	(P)VD				0.836
[Pigou et al., 2016] BD-RNN	VD				0.906
Hybride NN-HMM	PV	0.926	0.927	0.829	0.826
BDRNN	PV	0.935	0.934	0.852	0.846
[Neverova et al., 2016] DNN + CNN	PVD				0.868

Table B.1 – Précision et Indice de Jaccard sur différentes modalités : Pose P, pose utilisée indirectement pour la localisation (P), images V et cartes de profondeur D.

Une analyse plus approfondie des prédictions révèle que le modèle à base de HMM commet principalement des erreurs entre les gestes et les non-gestes. De son côté, le modèle RNN commet plus d'erreurs entre les classes, et rencontre plus de difficultés avec les gestes de courte durée.

La seconde phase de notre étude s'intéresse à la robustesse de la modélisation temporelle vis-à-vis de ses entrées. Pour se faire, nous modifions la largeur du contexte temporel présent dans l'extraction de représentation qui précède la modélisation temporelle à proprement parler. Ce contexte est modélisé par une convolution mono-dimensionnelle le long du temps sur les vecteurs de représentation. La largeur de la fenêtre de convolution détermine ainsi la taille du contexte. Dans le cas extrême d'une convolution assez large pour observer un geste entier, on peut supposer que la convolution temporelle seule peut assurer la modélisation temporelle, c'est d'ailleurs la stratégie adoptée dans [Neverova et al., 2014, 2016]. Inversement, une fenêtre courte limite l'apprentissage de représentation aux facteurs de variation à court terme, obligeant ainsi la HMM ou le RNN à modéliser les structures temporelle qui déterminent la signification d'un geste. Dans le cas extrême sans convolution, toute la modélisation temporelle repose sur ces modules.

La Figure B.4 démontre une différence de comportement évidente entre l'hybride NN-HMM et le RNN. Ce dernier présente un comportement invariant en fonction de la quantité de contexte présente dans ses entrées, tandis que le modèle HMM est plus largement dépendant et voit ses performances améliorées à mesure que les représentations prennent en charge une partie de la modélisation temporelle. On notera cependant que le modèle de transition de la HMM apporte toujours un gain de performance substantiel en filtrant les prédictions très bruitées du modèle à postériori des états cachés. D'autre part, le RNN n'"ignore" pas la présence de la convolution temporelle lorsqu'elle est disponible, on observe en effet des motifs très caractéristiques dans les filtres de la convolution temporelle qui prouvent son utilisation. Nous concluons donc que l'ajout de contexte est facultatif pour accompagner le RNN, car ce dernier est suffisamment flexible pour compenser son absence.

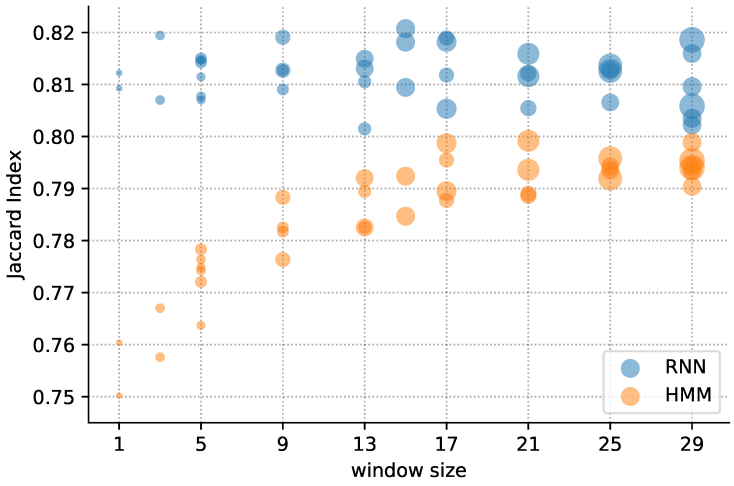


Figure B.4 – Indice de Jaccard sur l'ensemble de validation pour différentes largeurs de convolution temporelle. La taille des marqueurs est proportionnelle au nombre de paramètres présents dans les filtres.

La troisième phase de l'étude de nos modèles temporels s'intéresse à l'interaction en l'apprentissage de représentations et le modèle temporel. Dans toutes les expériences précédentes, l'ensemble des paramètres était entrainé de bout en bout pour chaque approche, y compris pour les couches d'apprentissage de représentation pour lesquelles seule l'architecture et les hyperparamètres sont partagée. Cependant, il est légitime de s'interroger sur le degré de co-adaptation entre ces deux étage du processus de reconnaissance. Une grande dépendance signifierait qu'il est indispensable d'apprendre des représentation spécifique pour un modèle et une base de donnée spécifiques, ce qui limite par conséquent le champs d'applications en conditions réelles. Pour ce faire, nous proposons d'effectuer deux tests de transfert d'apprentissage croisés entre les deux modèles temporels étudiés. Les représentations apprises et extraites pour la MHH seront injectée en entrée du RNN et réciproquement comme illustré sur la figure

Les résultats de la Table B.2 confirment la dépendance du modèle de transition de la HMM sur le travail du modèle à posteriori des observations. Sur les deux types de modalités, pose

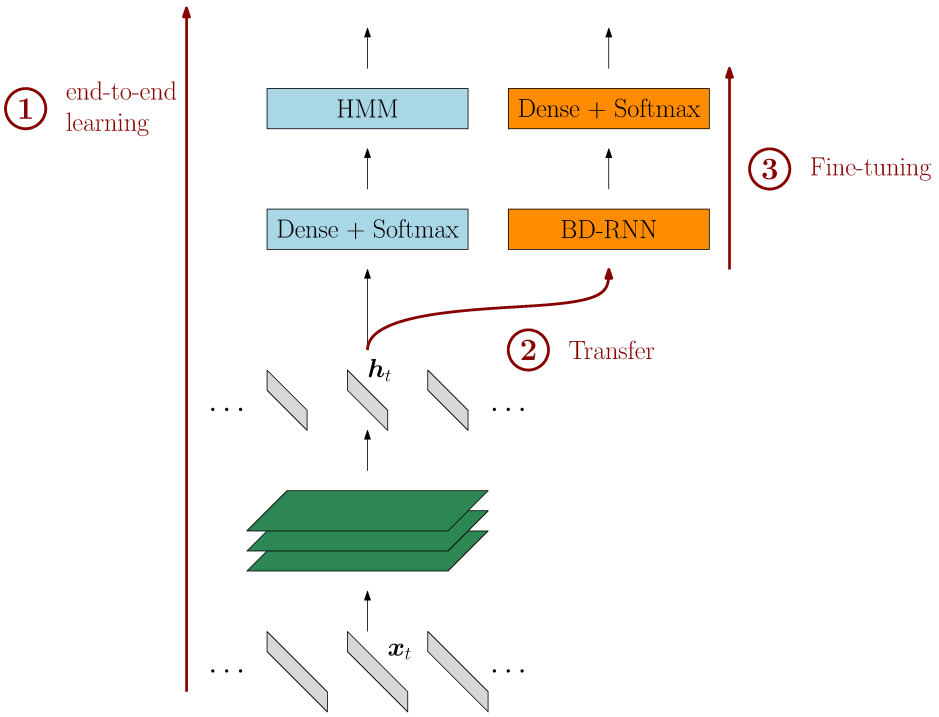


Figure B.5 – Principales étapes et architecture de l'expérience de transfert d'apprentissage Hybride NN-HMM vers RNN : ① Apprentissage bout-en-bout ; ② verrouillage des valeurs des paramètres, transfert dans l'autre modèle ; ③ entrainement des paramètres de couches supérieures selon la méthodologie usuelle.

corporelle dans l'espace et image des mains, le transfert cause une petite perte de performances. Au contraire, le RNN démontre sa versatilité en améliorant même légèrement les performance par rapport au modèle HMM qui lui fournit les représentations. En revanche le RNN accuse lui aussi une régression par rapport à la situation l'apprentissage bout-en-bout, ce qui signifie qu'il est préférable de ré-entrainer le modèle de représentation afin de maximiser les performances.

Modèle	Source	Précision	JI	Δ_{source}	$\Delta_{end\text{-}to\text{-}end}$
HMM	RNN (repr. de pose)	0.899	0.752	−0.059	−0.036
RNN	HMM (repr. de pose)	0.918	0.794	0.007	−0.016
RNN	HMM à posteriori (pose)	0.910	0.782	−0.006	−0.029
HMM	RNN repr. des mains	0.876	0.707	−0.081	−0.038
RNN	HMM repr. des mains	0.900	0.758	0.013	−0.031
RNN	HMM à posteriori (mains)	0.892	0.747	0.002	−0.042

Table B.2 – Performances pour les expériences de transfert d'apprentissage. Δ_{source} indique la différence de performance avec le modèle qui a fourni les représentations sur l'indice de Jaccard, $\Delta_{end-to-end}$ la différence avec le même modèle lorsqu'il est entrainé de bout-en-bout.

B.3.2 Apprentissage "en un coup"

Dans son mode de fonctionnement one-shot, la reconnaissance a lieu dans un contexte épisodique caractérisé par la succession d'étapes suivantes :

1. tirage d'un vocabulaire $\mathcal{V}^{ep}$ (typiquement une dizaine de classes) issu d'un large ensemble de catégories disponibles $\mathcal{V}^{ep}$
2. tirage d'un (éventuellement plus) exemple d'apprentissage pour chaque classe de $\mathcal{V}^{ep}$, donnant ainsi lieu à un ensemble d'apprentissage (X^{ep}, Y^{ep})
3. entrainement d'un classifieur sur (X^{ep}, Y^{ep})
4. évaluation ou utilisation du modèle sur d'autres exemples

On notera que l'étape 3 d'apprentissage one-shot n'est qu'un étape dans l'utilisation du modèle. Elle a donc aussi lieu lors de a phase de test pour évaluer les performance d'un modèle. L'objectif n'est donc pas de maximiser les performances pour un épisode particulier mais pour n'importe quelle combinaison de classes $\mathcal{V}^{ep}$ à distinguer les unes des autres. En d'autres termes, on n'évalue pas le modèle entrainé mais sa capacité à apprendre (rapidement).

Il est à noter que cette procédure de test épisodique est souvent précédée d'une autre phase d'apprentissage en amont qui prépare le modèle et facilite le travail de l'étape 3 dans laquelle le faible nombre d'exemple limite la quantité d'information disponible. Les classes d'exemples utilisées en test ne doivent toutefois pas apparaitre lors de ce pré-apprentissage, afin d'éviter la spécialisation du modèle et les fuites entre l'ensemble de pré-apprentissage et de test. Pour la majorité des modèles, cette étape de préparation est incontournable pour le succès de la reconnaissance, tandis que les données épisodiques ne servent qu'à spécialiser le modèle en pratique. La littérature sur le sujet apporte plusieurs implémentations possibles, comme le transfert d'apprentissage depuis un modèle entraîné sur tâche différente d'un domaine connexe. Un exemple de transfert est l'adaptation des travaux de vérification pour la biométrie. Pour toute base de données compatible avec la classification en apprentissage one-shot, il est possible de constituer un sous-problème de vérification en tirant des paires d'exemples issues de classes tirée au hasard. Lorsque la taille de l'ensemble d'apprentissage (nombre d'exemples et de classes) est suffisante, on peut aussi entrainer directement le modèle sur des épisodes en veillant toujours à isoler les catégories utilisée de celles réservée au tests.

Nous expérimentons ces deux approches ainsi qu'une troisième solution dérivée de la vérification, avec pour objectif d'apprendre des gestes issus du langage des signes. Un lexique de signes enregistrés dans des vidéos nous offre en effet un nombre de classes suffisant (2000 pour la base DEVISIGN [Wang et al., 2016] utilisée) pour tester la généralisation sur les classes, contrairement aux bases de gestes usuelles avec au plus une centaine de catégorie. Pour des raisons de temps et de ressources, nous n'avons exploité que les données de poses dans nos expériences, mais cette base comprend les même modalités que dans les expériences de reconnaissance continue.

La plupart des publications sur l'apprentissage one-shot faisant appel à des réseaux de neurones utilisent une stratégie commune : un réseau de neurone est entrainé pour générer un

vecteur de représentation réel pour chaque observation, et cette dernière est traité par un second modèle. Le premier étage n'est pas nécessairement spécifique à la tâche one-shot, on peut l'optimiser sur une tâche plus adaptée à l'entrainement d'un réseau de neurones, par exemple de la classification multi-labels ou de la vérification sur une large base de données. La seconde partie du modèle est directement liée au contexte one-shot avec les difficulté que cela implique, on y trouve donc souvent des modèles non-paramétriques comme les k-plus-proches-voisins qui sont naturellement robuste face au sur-apprentissage et à la faible quantité d'exemples d'apprentissage.

Nos travaux appliquent aussi cette méthodologie. Pour générer la représentation à partir des séquences d'observations de signes, nous appliquons un réseau de neurones récurrent bidirectionnel dont les états cachés finaux servent de représentation. L'architecture que nous proposons (Figure B.6) modernise largement celle proposée par [Pei et al., 2016] pour la vérification, qui est à notre connaissance la contribution la plus proche de notre travail.

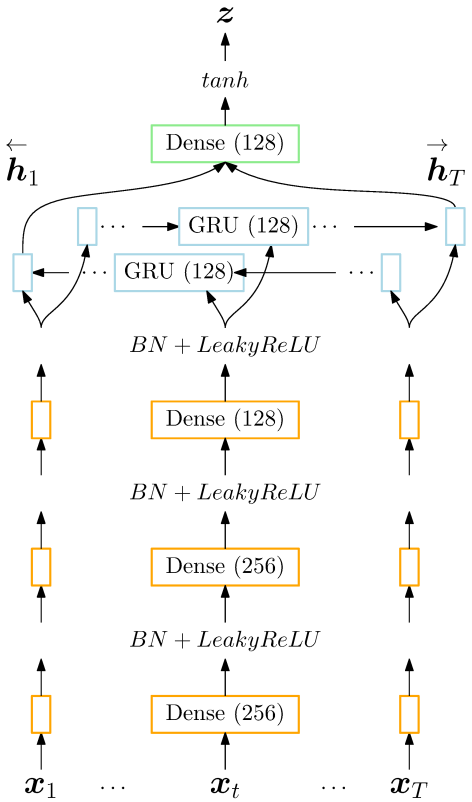


Figure B.6 – Architecture de l'extracteur de représentations pour l'apprentissage one-shot.

Nous utilisons ce réseau de Neurones que nous pré-entrainons selon trois approches différentes pour le même objectif final : l'apprentissage one-shot. La solution "historique" connue sous le nom de réseau siamois consiste à pré-entrainer le modèle pour la vérification de paires d'exemple. Le modèle est donc entrainé à générer des paires de représentations proches ou distante selon qu'on lui soumet deux enregistrement d'un même geste ou de gestes différents. Nous

utilisons une mesure de distance non-paramétrique suivie d'un critère de seuil pour effectuer la classification. Pour les tests sur des épisodes, qui impliquent de la classification multi-labels, la distance et le seuillage sont replacés par la méthode du plus-proche-voisin. Cet exemple illustre la méthodologie générale précédemment évoquée : un pré-entrainement sur une tâche connexe sert à optimiser la majorité voire la totalité des paramètres d'un modèle de représentation des entrée, ce dernier est ensuite coiffé d'un modèle non-paramétrique plus adapté au conditions du one-shot.

Une variation de cette approche consiste à travailler sur des triplets d'observations [Hoffer and Ailon, 2014 ; Hariharan and Girshick, 2017] et non des paires, avec dans chaque triplet deux exemples d'une même classe et un intrus. Cette méthode dispense de déterminer la proportion de paires positives ou négatives et de fixer un seuil de décision pour la tâche de vérification.

La troisième méthode implémente la méthode de Shepard proposée par **??**, dont l'usage n'a pas été très répandu jusqu'à récemment [Vinyals et al., 2016b]. Pour simplifier, la formule proposée offre une variation dérivable de la méthode du plus proche voisin, ce qui permet donc d'entrainer le réseau de neurone de bout-en-bout directement sur la tâche d'apprentissage one-shot en générant des épisodes. Nos essais montrent que l'optimisation du réseau bénéficie sensiblement de l'utilisation d'une fonction de coût hinge en lieu et place de l'entropie croisée couramment utilisée. On notera que la méthode de Shepard peut servir non seulement au pre-apprentissage, mais aussi en test pour toutes les méthodes présentées puisqu'elle remplace le k-plus-proches-voisins.

La table B.3 résume les résultats obtenus pour les trois solutions étudiées. Avec un précision

Table B.3 – Précision et rang de la cible sur des épisodes de 20 classes. Notations : shots → nombre d'exemples d'apprentissage par classe, 1-NN → plus proche voisin.

Shots	Métrique	Pré-apprentissage	Précision		Rang	
			Shepard	1-NN	Shepard	1-NN
1	L^2	siamois	0.837	0.837	1.32	2.63
	L^2	triplets	0.851	0.851	1.30	2.49
	cosine	Shepard+hinge	0.867	0.867	1.34	2.33
2	L^2	siamois	0.880	0.879	1.20	2.21
	L^2	triplet	0.901	0.895	1.17	2.06
	cosine	Shepard+hinge	0.908	0.910	1.18	1.91

supérieure à 80%, toutes les configurations testées présentent de solide performances, validant ainsi l'architecture du réseau de neurone. La méthode de Shepard, dont on rappelle qu'elle entraîne le modèle directement sur la tâche finale à accomplir, présente un léger avantage sur la mesure de précision, mais l'écart général entre les différentes méthodes reste assez faible. L'analyse des cas d'erreurs ne révèle pas de phénomènes particuliers, et la confusion entre certaines classes est souvent expliquée par la similarité du mouvement associé.

Des expériences complémentaires démontrent que ce modèle reste robuste, comme démontré sur des épisodes allant jusqu'à 50 classes, où la précision atteint toujours 80%. Nous avons aussi expérimenté une forme de méta-apprentissage avec les Matching Networks [Vinyals et al., 2016b], mais les résultat n'ont pas été concluants. L'intention du méta-apprentissage est d'augmenter l'impact de la phase d'apprentissage avec les exemples à l'intérieur des épisodes, mais au lieu d'avoir recours à des méthodes d'apprentissage traditionnelles, l'optimisation est confiée à un modèle parent qui prédit les paramètres ajustés. Cette approche est le pendant de la mémoire à long terme qui nous aide à trouver des solutions à de nouveaux problèmes au quotidien en utilisant des concepts génériques et réutilisables.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre:Modélisation par réseaux de neurones profonds pour l'apprentissage continu d'objets et de gestes par un robot

Mots clés : apprentissage profond, réseaux de neurones, reconnaissance de gestes, vision en temps réel

Résumé : Cette thèse a pour but de contribuer à améliorer les interfaces Homme-machine. En particulier, nos appareils devraient répliquer notre capacité à traiter continûment des flux d'information. Cependant, le domaine de l'apprentissage statistique dédié à la reconnaissance de séries temporelles pose de multiples défis. Nos travaux utilisent la reconnaissance de gestes comme exemple applicatif, ces données offrent un mélange complexe de poses corporelles et de mouvements, encodées sous des modalités très variées.
La première partie de notre travail compare deux modèles temporels de l'état de l'art pour la reconnaissance continue sur des séquences, plus précisément l'hybride réseau de neurones – modèle de Markov caché (NN-HMM) et les réseaux de neurones récurrents bidirectionnels (BD-RNN) avec des unités commandées par des portes. Pour ce faire, nous avons implémenté un environnement de test partagé qui est plus favorable à une étude comparative équitable. Nous proposons des ajustements sur les fonctions de coût utilisées pour entraîner les réseaux de neurones et sur les expressions du modèle hybride afin de gérer un large déséquilibre des classes de notre base d'apprentissage. Bien que les publications récentes semblent privilégier l'architecture BD-RNN, nous démontrons que l'hybride NN-HMM demeure compétitif. Cependant, ce dernier est plus dépendant de son modèle d'entrées pour modéliser les phénomènes temporels à court terme. Enfin, nous montrons que les facteurs de variations appris sur les entrées par les deux modèles sont inter-compatibles.
Dans un second temps, nous présentons une étude de l'apprentissage dit «en un coup» appliqué aux gestes. Ce paradigme d'apprentissage gagne en attention mais demeure peu abordé dans le cas de séries temporelles. Nous proposons une architecture construite autour d'un réseau de neurones bidirectionnel. Son efficacité est démontrée par la reconnaissance de gestes isolés issus d'un dictionnaire de langage des signes. À partir de ce modèle de référence, nous proposons de multiples améliorations inspirées par des travaux dans des domaines connexes, et nous étudions les avantages ou inconvénients de chacun.

Title : Deep learning-based continuing robot learning of visual objects and gestures

Keywords : deep learning, neural networks, gesture recognition, real-time vision

Abstract : This thesis aims to improve the intuitiveness of human–computer interfaces. In particular, machines should try to replicate human's ability to process streams of information continuously. However, the subdomain of Machine Learning dedicated to recognition on time series remains barred by numerous challenges. Our studies use gesture recognition as an exemplar application, gestures intermix static body poses and movements in a complex manner using widely different modalities.
The first part of our work compares two state-of-the-art temporal models for continuous sequence recognition, namely Hybrid Neural Network–Hidden Markov Models (NN-HMM) and Bidirectional Recurrent Neural Networks (BDRNN) with gated units. To do so, we reimplemented the two within a shared test-bed which is more amenable to a fair comparative work. We propose adjustments to Neural Network training losses and the Hybrid NN-HMM expressions to accommodate for highly imbalanced data classes. Although recent publications tend to prefer BDRNNs, we demonstrate that Hybrid NN-HMM remain competitive. However, the latter rely significantly on their input layers to model short-term patterns. Finally, we show that input representations learned via both approaches are largely inter-compatible.
The second part of our work studies one-shot learning, which has received relatively little attention so far, in particular for sequential inputs such as gestures. We propose a model built around a Bidirectional Recurrent Neural Network. Its effectiveness is demonstrated on the recognition of isolated gestures from a sign language lexicon. We propose several improvements over this baseline by drawing inspiration from related works and evaluate their performances, exhibiting different advantages and disadvantages for each.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France