Neural Networks for Natural Language Processing

Mohamed Morchid

ACADÉMIE D'AIX-MARSEILLE
AVIGNON UNIVERSITÉ

HDR

Defended at Avignon University
to obtain the Habilitation À Diriger des Recherches diploma

DOMAIN: Computer Science

Laboratoire Informatique d'Avignon (EA 4128)

Neural Networks for Natural Language Processing

by
Mohamed Morchid

Publicly defended the November 26th 2019 front of the jury composed with:

Mrs.	Dilek Z. HAKKANI-TÜR	Senior Principal Scientist, Alexa AI, USA	Reviewer
Mr.	Patrice BELLOT	Full Professor, AMU Polytech', LIS, Marseille	Reviewer
Mr.	Frédéric ALEXANDRE	First class Research Director, INRIA, Bordeaux	Reviewer
Mr.	Yannick ESTÈVE	Full Professor, AU, LIA, Avignon	Examiner
Mr.	Frédéric BÉCHET	Full Professor, AMU, LIS, Marseille	Examiner

Laboratoire Informatique d'Avignon

ACADÉMIE D'AIX-MARSEILLE
AVIGNON UNIVERSITÉ

HABILITATION À DIRIGER DES RECHERCHES

présentée à l'Avignon Université
pour obtenir le diplôme d'Habilitation À Diriger des Recherches

SPÉCIALITÉ : Informatique

Laboratoire Informatique d'Avignon (EA 4128)

Réseaux de Neurones pour le Traitement Automatique du Langage

par
Mohamed Morchid

Soutenue publiquement le 26 Novembre 2019 devant un jury composé de :

M^{me}	Dilek Z. HAKKANI-TÜR	Senior Principal Scientist, Alexa AI, USA	Rapporteur
M.	Patrice BELLOT	Professeur, AMU Polytech', LIS, Marseille	Rapporteur
M.	Frédéric ALEXANDRE	Directeur de Recherche INRIA, Bordeaux	Rapporteur
M.	Yannick ESTÈVE	Professeur, AU, LIA, Avignon	Examinateur
M.	Frédéric BÉCHET	Professeur, AMU, LIS, Marseille	Examinateur

Laboratoire Informatique d'Avignon

To my family and Claire
and in memory of my beloved father,
Belfakih Morchid ...

... you will be eternally present
here and everywhere, now and always ...

Summary

This dissertation presents my recents efforts on both neural networks based methods and algorithms, and their applications on real-world tasks for Natural Language Processing (NLP). This document is organized following my different points of interest from hitherto proposed real-valued neural networks applied to challenging NLP related tasks, to novel complex-valued neural based architectures that seek to address the recent issues related to the learning process. The document presents my main achievements and is organized as follow:

Real-valued Neural Networks. Nowadays, NLP tasks employ massively Neural Networks (NN) based algorithms to solve challenging real-world tasks and the document:

• introduces novel NNs for considering multiple input streams of information during the learning process to better predict the desired output,

• studies the impact of noisy documents from an Automatic Speech Recognition (ASR) system on an encoder-decoder that extracts robust features from spoken documents.

Quaternion-valued Neural Networks. Real-valued NN hardly handle inter- and intra-dependencies between input features during the learning. We propose novel Quaternion-valued neural networks (QNN) that consider inter-dependencies between the input features as well as the dependencies between basic elements composing a given feature. In this document we:

• present a Quaternion-valued Convolutional Neural Network (QCNN) that consider alongside the convolution process (inter-dependencies) and the latent relations between basic elements of each feature (inter-dependencies),

• introduce the Quaternion-valued Recurrent Neural Network (QRNN) to encode multi-dimensional input sequences from speech signal.

Ongoing works and futur directions. Real-valued and Quaternion-valued neural networks are difficult to learn since the amount of data required for learning is potentially large, even huge; moreover, these models are also employed to generate unseen information such as images or documents. Among these models, generative adversarial networks (GAN) have encountered a large success in different natural language processing (NLP) related tasks. Nonetheless, these models are unstable during the learning process. This part:

• introduces novel parsimonious neural networks that better handle large corpora and consider the long-term dependencies,

• stuies the impact of manual transcripts on automatically transcripts of spoken documents during a NLP task with GAN mapping approach.

Contents

Introduction

Context

Few words on my research interests

My initial researches have been mostly based on topic models (Blei et al., 2003) applied to Natural Language Processing (NLP) tasks. Among these high-level representations, the latent Dirichlet allocation (LDA) (Blei et al., 2003) based model allows NLP systems to represent relevant content of documents in a abstract latent topics model. We have employed LDA representations to extract robust features from noisy spoken dialogues. After my Ph.D. thesis, we have been interested to evaluate the benefit of neural networks initialization with latent based features from an LDA model (Morchid et al., 2015b). It was my first investigation of neural networks for real-world NLP applications. Since 2015, my efforts have been focused on applications of hitherto proposed neural networks for real-life problems such as theme identification of spoken dialogues and on complex-valued neural networks for different pattern recognition related tasks. These interests are at the edge of the Natural Language Processing (NLP) and machine learning (ML) domains composing an important part of the Artificial Intelligence (AI) broad spectrum. Indeed, AI covers a large area of NLP throughout different real-world applications. Among AI based paradigms, connectionist models or Parallel Distributed Processing (PDP) (McClelland et al., 1986; Rumelhart et al., 1986) have been introduced in a large number of domains, ranging from different aspects of language processing to cognitive control from perception to memory. Connectionist models take inspiration from the manner in which information processing occurs in the brain. Processing involves the propagation of activation among simple units (artificial neurons) organized in networks, that is, linked to each other through weighted connections representing synapses or groups thereof. Each unit then transmits its activation level to other units in the network by means of its connections to those units. The activation function describes how each unit computes its activation based on its inputs, may be a simple linear function, but is more typically non-linear (for instance, a sigmoid function). These neural networks have encountered a wide success due to their capabilities to code complex abstract structures and patterns in latent sub-spaces. In the context of NLP applications, these neural networks models such as the Multi Layer Perceptron (MLP) have to process large even huge amount of data due to the rapid growing of sharing platforms on Internet as well as the number of available data-sets. Moreover, the datas available are

even more composed with different signals (speech segments) or dimensions (images or multi-channels sequences). Therefore, the models and paradigms proposed to process large amount of multidimensional data have to both be efficient to process quickly this large amount of information and consider multiple dimensions of the document content. Since our work are mostly focused on neural networks based models with applications on NLP, in the last years, we have been interested on other specific applications such as image processing (Parcollet et al., 2019). Nonetheless the last results are still too recent to be mentioned in this document. This thesis reports part of the work done in collaboration with Ph.D. students and colleagues on field of deep learning. In regards to the large number of neural based models and paradigms this chapter focuses on related works and the reader can refer to papers from main AI related conferences and journals for further details.

Brief history of neural networks with some key dates

Figure 1 presents some main steps throughout the last decades of the neural networks history. The contemporary period started with the advent of GPU cards at 1999 from Nvidea and the CUDA librairies in 2007 to develop the algorithms. These cards allows neural networks based models to drastically reduce the processing time required for learning by 70 (Raina et al., 2009). With the large number of academics and industrials that then employ CUDA to develop machine learning systems, NVidea have released a dedicated set of librairies called cuDNN in 2014.

Deep learning is originated from the availability of large-scale training data-sets and the increased computation power from GPU cards and groups feature learning techniques exploiting artificial neural networks (LeCun et al., 2015) among other algorithms. Human being also conducts learning via neural networks based on biological neural networks inside the brain. Neuroscientists have studied how information is processed in human being brain in order to understand it capability of processing information effectively. Neural networks based models and paradigms are able to process data and natural information much better than human being. Deep learning becomes an international strategic issue due to their impressive performances on different domains such as human communications, medicine, law, political analytics or military [1] [2] with unlimited uses. France will be part of these general leaders with recent political efforts [3] [4]. Among AI models, neural networks have encountered a wide success in different real-world applications. These neural networks are composed with basic artificial neurones. This document is focusing on neural networks and their application on different NLP related tasks.

[1] https://www.forbes.com/sites/stratfor/2018/02/07/the-coming-tech-war-with-china/#3a1b6a641cd4

[2] Russia's leader, Vladimir Poutine, have claimed that the nations that will develop effective AI models and applications will lead the world: https://www.cnbc.com/2017/09/04/putin-leader-in-artificial-intelligence-will-rule-world.html

[3] https://www.economie.gouv.fr/France-IA-intelligence-artificielle

[4] https://www.forbes.com/sites/forbestechcouncil/2017/12/05/these-seven-countries-are-in-a-race-to-rule-the-world-with-ai/

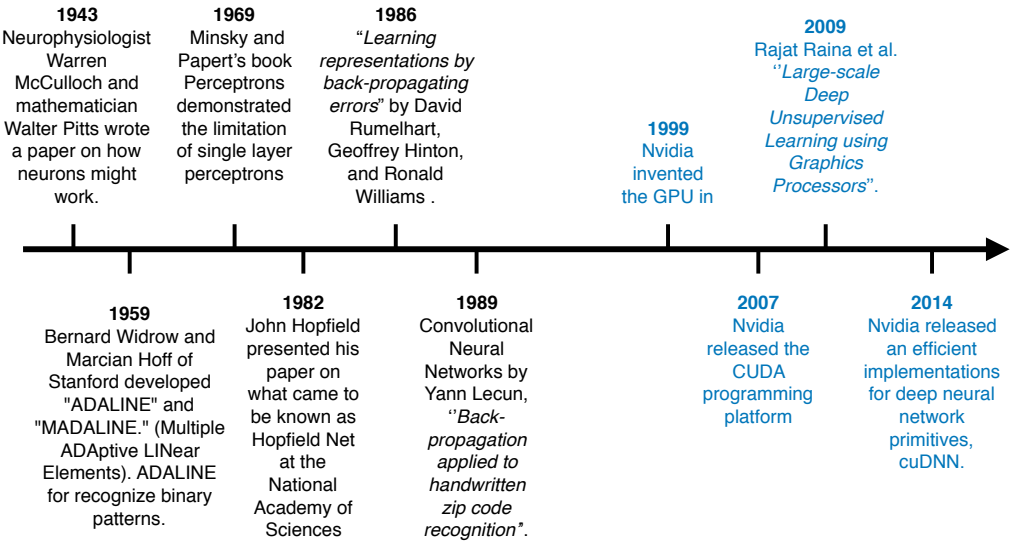


Figure 1: *Neural Networks in some of the main dates.*

Neural Networks

The first definition of artificial neurons or formal neurons has been introduced in (McCulloch et Pitts, 1943). In this definition a neuron has a variable number of weighted inputs simulating the dendrites of biological neurons. All inputs are summed and an activation function is applied to this sum to produce an output value. The first application of the formal neuron is the perceptron (Rosenblatt, 1958) composed of a single neuron and a heavy-side activation function and allows a binary classification. In addition to the neuron a supervised learning method of perceptron is introduced to estimate its weight representing an hyperplane by an iterative process. This hyperplane is optimized to separate two classes. On the other hand, a perceptron is unable to solve a problem that is not linearly separable. This weakness was highlighted by the problem of "Xor": Perceptron cannot model the "exclusive or" function. This limitation has slowed down the development of artificial neural networks for several decades. (Werbos, 1974) propose the gradient backpropagation method for learning artificial neural networks. Since 1982, artificial neural networks are able to extract descriptors through unsupervised learning (Kohonen, 1982). In these networks, the neurons are arranged in a grid. Each area of this neural grid learns to respond to a particular type of data. Once trained, an adaptive map can be used to either discretize or perform a vector quantization or to reduce the dimensionality of input data (Chihi, 1998; Nasrabadi et Feng, 1988). Artificial neural networks whose interneuronal connections form an acyclic graph are

called "feedforward neural networks". The perceptron is an example of a feedforward neural network.
Multilayer Perceptrons (MLPs) (Rumelhart et al., 1985) are supervised networks that can be used for classification, regression or dimensionality reduction. In this architecture the neurons are grouped in layers. There are an input layer that receives the information that the network have to process and an output layer that provides the neural network output prediction. The activation function of the output layer can be chosen according to the task that the network has to perform: the linear activation function is used to learn linear regression models; the sigmoid activation function is employed for logistic regressions or binary classifications; the softmax activation function (Bishop et al., 1995) performs multiclass classifications. A cost function calculates the error between the predicted output and the expected output. A variable number (greater than or equal to 1) of hidden layers with a nonlinear activation function is inserted between input and output layers. Each neuron in a layer is connected to all the neurons of the previous and next layers. On the other hand, there are no connections between neurons from the same layer. The weights of these connections between two successive layers are learned by backpropagating the gradient between the predicted and the expected output. More details on different neural networks throughout the next dedicated chapters to improve the readability and the understanding of each model in a suitable context.

Research directions

We relate in this section different research directions based on ongoing works on different neural networks related models for real world NLP applications. Each of the following sections describes the paradigms and the perspectives of each directions. All these researches are different and close in some regards. Indeed, these directions have a common ground composed of both theoretical (neural networks) and practical (NLP real life applications) components.

Multidimensional models

Available multidimensional data from different mediums are even more large and, in the last decades, scientists from different AI related fields such as language processing or machine learning have given an important efforts to propose efficient algorithms and paradigms to handle large amount of information. Indeed most of the documents available nowadays are composed with different information from different mediums or streams. This is particularly the case for videos that are composed with images, audio and potentially textual content. Sharing platforms such as Youtube [5] collect large even huge amount of videos per days. Indeed this year statistics on Youtube show that 400 hours of video are uploaded to every minute [6] and the number of available videos ex-

[5] www.youtube.com
[6] https://www.brandwatch.com/blog/youtube-stats/

ceed 7 billion. Different approaches have been studied to process such as multi-streams data (Dai et al., 2004; Babcock et al., 2002) such as the Clustering on Demand (CoD) framework (Dai et al., 2004) or the Data Stream Management System (DSMS) (Babcock et al., 2002). These models reach good results but are mostly based on optimization problems that not retain an internal state to memorize sequences. Artificial Intelligence based algorithms are efficient to learn from seqeunce's basic element states such as neural networks.
Deep Neural Networks (DNN) allow systems to efficiently process these big data in terms of systems performances and processing time required during the learning process. Nonetheless, spoken and textual documents processed in Natural Language Processing (NLP) systems have to take into account sequences made of the words composing a sentence or a paragraph. The Recurrent Neural Networks (RNN) (Rumelhart et MacClelland, 1986; Pearlmutter, 1989) is a DNN and is well adapted to sequence data for spoken or textual documents processing in NLP-based systems such as statistical parametric speech synthesis (Fernandez et al., 2014; Fan et al., 2014), speech emotion recognition (Chao et al., 2015; Chen et Jin, 2015), facial landmark detection (Chen et al., 2017) and speech recognition (Gajecki, 2014; Graves, 2012; Siniscalchi et al., 2013). RNNs can be applied to almost any problem with sequential structure. Nevertheless RNNs fail to learn long-term dependencies due to the vanishing gradient problem (Bengio et al., 1994) and gates based units have been introduced to address this drawback like Long Short-Term Memory (Hochreiter et Schmidhuber, 1997) or Gated Recurrent units (Chung et al., 2014). These RNN-based models learn the internal state from a sequence of basic elements and process only uni-dimentionnal data.
Convolutional Neural Networks (CNN) (LeCun et al., 1989) can process multi-dimensional data, such as images (Red, Green and Blue (RGB) pixels) or documents (word/sentence matrix) (Zhang et al., 2015). Nonetheless CNN are limited to 2- or 3-D dimensional feature spaces and the number of available streams for a given information may be larger than 2 or 3. Another drawback of both CNN- and RNN-based models is the manner to compute latent states of multiple parallel inputs during the learning. Indeed recent researches have proposed to learn the latent states in all gates forming the gated RNN (Graves et al., 2007; Graves et Schmidhuber, 2009; Stollenga et al., 2015) and applications such as pixel-wise brain image segmentation (Stollenga et al., 2015). The different inputs corresponding to different streams are added within each gate to code latent dependencies of these streams during the learning process of gates as well as the exposed outputs.
Nonetheless all parts of the model (gates for LSTM/GRU) process all streams to better predict the optimal output while each stream has a particular content. In the case of label prediction of a given stream $\mathbf{s}$, all parallel streams do not contain enough information on $\mathbf{s}$ to propose an efficient candidat as an output of $\mathbf{s}$. We have proposed during the Ph.D. thesis of Mohamed Bouaziz to learn from individual streams an efficient process called Parallel Long Short-Term Memory (see chapter 1) to predict the next label of a given stream knowing the other ones with LSTM-RNN models (Bouaziz et al., 2016). This model allows the system to take into account different streams during the learning process. The experiments show that the proposed RNN-based model reach better performances than n-gram models. Nonetheless, all streams are taken into account in the

same level for all gates and all streams. The streams can be split throughout the gates based on the goal off each gate [7]. Therefore, an interesting perspective for this work is to find paradigms to better split the information from different streams between gates of RNN based models in regards to the application. In the case of label prediction of a given stream **s** the forget and output gates have to process all streams but the input gate have to be considered only with the stream **s**. Indeed, in this particular case, only the stream **s** can predict an efficient candidat for the next label.

Autoencoder neural networks

Encoder-decoder neural networks or Autoencoders are a special architecture of MLP. In regards to the large, even huge, bibliography on neural networks, this section focuses only on autoencoders and their sub models. One can refer to (LeCun et al., 2015; Goodfellow et al., 2016) for more information. The term feedforward refers to a set of neural networks with connections between neurons form an acyclic graph. MLP (Rumelhart et al., 1985) are supervised neural networks that can be used for classification, regression or dimensionality reduction. In this architecture the neurons are grouped in layers with an input layer that receives the information as a set of features that the network process; an hidden layer that represents the input vector in a hidden space; and an output layer that contains the network prediction. The activation function of the output layer can be chosen according to the task that the network has to perform. The linear activation function is used to learn linear regression models. The sigmoid activation function is used for logistic regressions or binary classifications. The Softmax activation function (Bishop et al., 1995) is employed for multi-labels classification related tasks. A cost function computes the error between the output produced by the MLP and the expected output. Each neuron contained in a layer is connected to all the neurons of the next layer with no connections between this neuron and the neurons from the same layer. The weights of these connections are learned by back-propagating the gradient of the error between the predicted output and the expected one throughout the layers. An autoencoder is a MLP with the same features for the input layer and the expected output.

The work presented in Chapter 2 describes the first efforts to process documents on autoencoders to learn the mapping from automatically transcribed spoken documents from an Automatic Speech Recognition (ASR) system to manually transcribed documents. The aim of these studies is to evaluated the potential to neural based systems to learn efficient latent relations between "noisy" documents (from ASR) and "clean" ones (from human transcriptions). These approaches are based on autoencoders neural networks on either the generative, discriminative or both sub-processes. Our work depicted in Chapter 2, reflect my strong interest on latent variables from autoencoders and we plan to follow the study during the next years in regards to the

[7] the input gate of the LSTM evaluate how much from the hidden state candidate or memory has to be considered; the forget gate retain or forget part of the previous hidden state and the output gate decides the proportion of the new hidden state that has to be exposed to the output

promising results obtained by the proposed autoencoders approaches alongside with the enthusiasm generated by GANs. Chapter 6 reports our first effort on GAN for NLP. One can refer to (Creswell et al., 2018) for more information on autoencoders.

Quaternion-valued neural networks

During my Ph.D. thesis, my work has been focused on the extraction process of robust features from latent spaces such as Latent Dirichlet Allocation (LDA) (Blei et al., 2003), from *i*-vectors (Morchid et al., 2015a) or from hyper-complex numbers called Quaternions (Morchid et al., 2013). Quaternions are an extension of complex-numbers and are defined as hypercomplex numbers that contain a real and three separate imaginary components perfectly fitting to three and four dimensional feature vectors such as for image processing, robot kinematics or computer graphics (Sangwine, 1996; Pei et Cheng, 1999; Aspragathos et Dimitros, 1998). Contrary to traditional homogeneous representations, quaternion networks bundle sets of features together. Thereby quaternion numbers allow neural network based models to code latent inter-dependencies between groups of input features during the learning process with fewer parameters than real-valued neural networks by taking advantage of the *Hamilton product* as the equivalent of the ordinary product but between quaternions. Early applications of quaternion-valued back-propagation algorithms (Arena et al., 1994, 1997) have efficiently solved quaternion functions approximation tasks. More recently neural networks of complex and hypercomplex numbers have received an increasing attention (Hirose et Yoshida, 2012; Tygert et al., 2016; Danihelka et al., 2016; Wisdom et al., 2016) and some efforts have shown promising results in different applications. In particular a deep quaternion network (Parcollet et al., 2016; Titouan et al., 2017; Parcollet et al., 2017), a deep quaternion convolutional network (Chase Gaudet, 2017; Titouan et al., 2018) and a quaternion recurrent neural network (Parcollet et al., 2018c) have been developed and employed by out team from the LIA for different challenging tasks such as images classification, compression and reconstruction or speech recognition.

Motivations to use quaternion neural networks over real-valued ones for quaternion-valued tasks are numerous and well documented (Parcollet et al., 2018c; Isokawa et al., 2009). The major advantage of any QNN is the more efficient and natural representation of the input information compared to traditional real-valued NNs. Indeed a good neural network model has to represent, encode and learn the nature of the input data. Many realistic tasks involve multi-dimensional features with input elements composed with multiple internally related components. These applications are image processing with three channels describing a single pixel or speech recognition with three values (Mel filter banks and deltas) for a unique time frequency, or human motion recognition with outputs and inputs to be represented in the 3D space with 3D coordinates. To succeed in such multidimensional tasks the employed model needs a specific representation and processing of the features. Indeed traditional real-valued NNs consider internal relations at the same level than contextual and global dependencies. As an example, relations between different pixels are considered

equivalently to the relation that links the three channels composing each pixel, and it is not clear if both relations are well learned. More precisely, composed entities are processed by traditional real-valued NNs as a bag of independent and smaller components, while a natural solution is to process such entities as multidimensional and internally related elements, such as with quaternion neural networks (QNNs). Consequently QNNs have been shown to suit particularly well to these representations and it has been demonstrated that they are perfectly able to learn internal relations describing the multidimensional features, with way less neural parameters (Matsui et al., 2004; Parcollet et al., 2018d). Indeed a 4-number quaternion weight linking two 4-number quaternion units only has 4 degrees of freedom, whereas a standard neural net parametrization has $4 \times 4 = 16$, i.e., a 4-fold saving in memory.

Parsimonious neural networks

Neural architectures are huge and Quaternions allow the models to require less parameters to learn intra- and inter-dependencies between input features but require a 4-dimensional representation to process Hamilton dot product. Most of the NLP related tasks are based on bag-of-words representations, even the use of embeddings are nowadays the usual input representation. Therefore, real-valued neural networks are well suited to process basic document features such as words occurrences or word embeddings. Moreover, real-valued neural networks require a large amount of data to learn latent dependencies in the features space to represent effectively these basic elements throughout enough related contexts. Therefore, the processing time as well as the large number of documents required for learning efficient document representations is closely related too the network architecture; moreover, these representations have to be robust enough to be effective since unseen documents are processed by the NLP systems by learning different hidden states for different related contexts of sequences for both short- and long-term dependencies. Recurrent Neural Networks (RNN) are a powerful set of architectures that code and store internal hidden states of sequences to learn basic element latent dependencies.

Nonetheless, RNN-based models (Zaremba, 2015) require a large set of patterns to code most of the possible contexts of a given event and are time consuming. It is mostly the case since the task is to reveal latent dependencies between documents event (word occurrences) during a natural language processing (NLP) task. Indeed, a given word has to be encountered in different sentences (contexts) to better understand the latent relations between these basic elements. Nevertheless RNNs fail to learn long-term dependencies due to the vanishing gradient problem (Hochreiter, 1998). RNNs with gates such as Long Short-Term Memory (Wu et al., 2017b; Sundermeyer et al., 2012; Greff et al., 2017) (LSTM) avoid the vanishing gradient problem of the RNN (Hochreiter, 1998) along by taking into account both short- and long-term dependencies of the word properly in different contexts but need to process the information in different memory blocks composed with a set of cell-related gates. Even if the results of LSTM-RNNs are promising the processing time required to treat large documents data sets is quite huge due to the different gates activation sub-processes.

Indeed, LSTM has to decide from a previous state and a candidat state to process dedicated decision processes via gates to update the memory cell. A recent addition to the RNN set of models called the Gated Recurrent Unit (GRU) has been proposed by (Bahdanau et al., 2014; Wu et al., 2017b) to address this issue and has shown good performances in several tasks such as speech recognition (Graves et al., 2013b), machine translation (Sutskever et al., 2014; Cho et al., 2014). Deep internal structures have been proposed to introduce a depth gate (Yao et al., 2015) to better expose internal hidden relations between memories. GRU is very similar to LSTM in that it uses a combination of gates to adjust exposure from input to the hidden states. It does however not use a memory cell instead opting to fully expose its state to the output thereby also doing away with the output gate. Switching between full retention, mixed and forget mode is implemented using a reset gate r and an update gate z. This is the main drawback of the GRU-RNN model in that the role and the management of the GRU gates is not based on the relation between short- and long-term dependencies. For example, the reset of the hidden state is provided by both update and reset gates of the GRU. Moreover, the role of the reset gate of the GRU is to avoid the hidden state candidate but the update gate z can remove this part of the hidden state information without the reset gate r. The minimal gated unit (MGU) (Zhou et al., 2016) is composed with a unique gate to update and reset the cell state. They adapt a GRU with a gate that merges r and z gate of the GRU ($r = z$) onto a unique gate called f. The authors "*prefer an RNN architecture that has the smallest number of gates without losing LSTM's accuracy benefits*" and "*propose the Minimal Gated Unit (MGU), which has the smallest possible number of gates in any gated unit. MGU only has 1 gate, which is the forget gate. MGU is based on GRU, and it further couples the input (reset) gate to the forget (update) gate*". The motivation to merge the update and reset gates is not based on an theoretical study of term dependencies. A unique gate f will both reduce the importance of the candidate hidden state and the memory of long-terms dependencies during the processing of the hidden state candidate. This behaviour is close to the leaky unit (Bengio et al., 2013a) that reduces the hidden state candidate influence when the previous hidden states are considered with a single gate u. The LU is equivalent to the GRU with a single gate and requires less processing time for learning than the GRU and the LSTM. Nonetheless, the LU codes mostly long-term dependencies but reveals little in the way of short-term dependencies. Other papers have studied single gate units such as the MinimalRNN (Chen, 2017). The MinimalRNN is a GRU with a single gate and differs from LU by processing the candidate hidden state with the input vector $s_{h_j m}$ only ($r_j(t) = 0$ in the case of the GRU). Therefore, the authors reduce the number of gate but do not study the behaviour of the gate during the learning process.

These efforts to propose efficient neural based architectures requiring even less processing time is crucial. Indeed, since the large number of documents are available with the grow of Internet and the number of applications are mostly employed on mobile devices such as telephones or tablets, the needs on parsimonious models to process this large amount of information are important with less parameters (for mobile devices for example). Therefore, the neural based models have to process these documents effectively with a small number of parameters and with less processing

time than hitherto proposed neural networks.

Our work started with quaternion neural networks (see chapter 3 and 4) allow NLP systems to require less neural parameters to process effectively big corpus. We have also proposed (see chapter 5) the "Parsimonious Memory Units" (PMU) for RNN that differs from GRU with a reduced number of gates by considering first the behaviour of the gates and the role of the activity of the gates in the learning process of the term dependencies in the hidden state. Moreover, the PMU architecture is based on the assumption that short- and long-term dependencies are related. (Yao et al., 2015) present the depth GRU that allows the internal state to consider latent relations between the memories from different layers and differs from our work which is interested in the best way to manage both short- and long-term dependencies during the learning process. We plan to continue the study of memory behaviour within neural process and provide novel neural based architectures to process large data-sets with less parameters alongside with efficient performances during the AISSPER project (ANR 2019) for wich I am the coordinator that will start on January 2020 in collaboration with LIUM and Orkis. The aim of AISSPER is to provide efficient end-to-end neural based models for Spoken Language Understanding (SLU).

Generative adversarial networks

The obtained results with QNN and autoencoders are promising and generative models will be a main research topic for my future directions. Recently, two promising architectures have been introduced to provide powerful generative models: variational autoencoders (VAE) and Generative Adversarial Networks (GAN). Variational autoencoders are introduced as a link between variational inference (Fox et Roberts, 2012) and deep neural networks (Kingma et Welling, 2013; Gal et Ghahramani, 2015). They consider a set of latent random variables z, to capture the variations of the observed variables x. Their joined distribution are defined by:

$$p(x,z) = p(x|z)p(z) \tag{1}$$

Where the distribution of prior probabilities of $p(z)$ follows a normal distribution and $p(x|z)$ is an observed model whose parameters are calculated by the neural network as a function of z. The non-linear projection of z to x makes it impossible to infer the posterior distribution of $p(z|x)$. For this purpose the VAE uses a variational approximation $q(z|x)$ whose distribution follows a normal distribution. The parameters of this normal distribution (mean and variance) are the output of a non-linear function also learned by the neural network. The generative model $p(x|z)$ and the inference model $q(z|x)$ are learned by backpropagation in order to maximize the lower bound of the likelihood of $p(x)$. They have been used successfully to generate sentences (Bowman et al., 2015), images (Gregor et al., 2015) or to apply rotations on 3D images (Kulkarni et al., 2015).

Generative adversarial networks (Goodfellow et al., 2014) (GAN) combine two

neural networks. The first network G is a generative network that generates a vector x from a latent representation from documents of the training corpus. The second network D is a discriminant network that predicts the probability that the input sample is generated by G instead of coming from the training corpus. Both networks optimize their weight to find a balance. They have been used mainly to generate images, from constraints of shapes or colors (Zhu et al., 2016) or from textual descriptions (Reed et al., 2016).

Nowadays Generative Adversarial Networks (GAN) became one of the dominant approaches for learning latent features from generative models in recent machine learning researches. These models provide a flexible algorithm for learning latent variables describing the data generating processes in which noise is transformed into "clean" data samples on large datasets. A large set of GAN variants are based on autoencoder GANs (AE-GANs). The autoencoders improve GAN training (Nguyen et al., 2017) by combining an autoencoder loss, a GAN loss, and a classification loss defined using a pre-trained classifier. AE-GANs can be broadly classified into three approaches:

- those using an autoencoder as the discriminator, such as energy-based GANs and boundary-equilibrium GANs (Berthelot et al., 2017),
- those using a denoising autoencoder to derive an auxiliary loss for the generator, such as denoising feature matching GANs (Warde-Farley et Bengio, 2016),
- and those combining ideas from VAEs and GANs such as the variational autoencoder GAN (VAE-GAN) (Larsen et al., 2015) that adds an adversarial loss to the variational evidence lower bound objective, the more recent mode-regularized GANs (MRGAN) (Che et al., 2016) or the adversarial generator encoders (AGE) (Ulyanov et al., 2018).

Recently, these generative adversarial networks (GANs) (Goodfellow et al., 2014) received an astonishing interest due to the remarkable results obtained in computer vision (Zhu et al., 2017; Radford et al., 2015; Kim et al., 2017; Berthelot et al., 2017). The ability of GANs to generate samples that are closely-related to targets ones has also been extended to natural language processing (NLP) tasks, such as text and dialogue generation (Rajeswar et al., 2017; Donahue et Rumshisky, 2018; Li et al., 2017), or neural machine translation (Yang et al., 2017; Wu et al., 2017a). To the best of our knowledge, (Wu et al., 2017a) is the most related work to the problem addressed in this paper. Indeed, (Wu et al., 2017a) proposed a model aiming to generate sentences which are hard to be discriminated from human-translated sentences. Consequently, the GAN model is expected to learn a mapping from one language to another one based on human manual translations. GANs became a specific domain in AI and researchers produce a broad spectrum of related models. an conditional version of the GAN has been proposed in (Mirza et Osindero, 2014) to generate unseen elements (not included in the data set) from unlabeled data. (Makhzani et al., 2015) propose a general approach, called an adversarial autoencoder (AAE) that employ hidden states of an autoencoder and random distributions into a generative model

for different tasks. The autoencoder is trained with dual objectives: a traditional reconstruction error criterion, and an adversarial training criterion (Goodfellow et al., 2014) that matches the aggregated posterior distribution of the latent representation of the autoencoder to an arbitrary prior distribution. This model is close to the CycleGan (Zhu et al., 2017) that also employ an intermediate representation from an autoencoder to learn a GAN model. In regards to rapid grow of this sub-domain of GAN and the large number of papers related to GAN, one can easily find out more details and models in the main conferences and journals related to AI [8]. Nonetheless, GAN does not take into consideration the target classes when generating the samples, since *golden-targets* (i.e manual transcriptions) are not available at testing time.

We are interested in such as models since they allow NLP based systems to both generate unseen representations and extract robust features from the document content. We have starting to follow this way and propose novel architectures to process noisy spoken documents (Titouan et al., 2019a). The task considered in this thesis (see chapter 6) replaces the initial language by an automatic transcription of a conversation, and the target language by its manual transcription. Furthermore, we propose to perform classification on top of the generation, as driven by the semi-supervised GAN (SGAN) approach (Odena, 2016).

Structure of the thesis

The document relates different research directions followed since my Ph.D. thesis defended at 2014 November. The different chapters thereafter are initiated with Ph.D. students that I have supervised from 2015 to 2019. The first direction followed was related to my interest on machine learning methods for NLP real-world applications. We have started by evaluating hitherto proposed neural networks based models for Natural Language Processing (NLP) tasks. These models have then been extended with original algorithms to improve the efficiency during the NLP related tasks. Then, we have proposed novel neural networks architectures based on hyper-complex algebra to better handle multidimensional information. These approaches have encountered a wide success and interest from different institute and researchers from heterogeneous AI related fields. Alongside to these works, I have started new directions to allow neural network algorithms to process large amount of documents with less processing time and better performances than mere ones. All these research directions will be followed throughout the ANR AISSPER project that will start at January 2020 in collaboration with the LIUM (Le Mans University) and Orkis (Aix-en-Provence company). Indeed, AISSPER will first employ already proposed neural based architectures and then novel ones for end-to-end SLU tasks.

Part I of the document describes the first works on machine learning for NLP. The aim of this period was to evaluate hitherto proposed neural networks based

[8] For example, this website gives both papers and codes for a large number of GAN models: https://github.com/eriklindernoren/PyTorch-GAN

algorithms for real-world NLP applications. We present a research done during the Mohamed Bouaziz's Ph.D. thesis, in CIFRE convention with the company EDD and co-supervised by Georges Linarès (LIA), Richard Dufour (LIA) and myself between September 2013 and December 2017. Therefore, chapter 1 presents an original Long Short-Term Memory-based (LSTM) architecture, called Parallel LSTM (PLSTM), that carries out multiple parallel synchronized input sequences in order to predict an output. The proposed PLSTM could be used for parallel sequence classification purposes. The PLSTM approach is evaluated during an automatic telecast genre sequences classification task. The results show that the proposed PLSTM method outperforms the baseline and the LSTM model. This work correspond to the first step and direction on machine learning for NLP. This effort is followed by a study of encoder-decoder neural networks for NLP tasks. Chapter 2 corresponds to part of the work done during the Killian Janod's Ph.D. thesis in CIFRE convention with the company Orkis and co-supervised with Georges Lianrès (LIA), Richard Dufour (LIA) from September 2013 to December 2017. This chapter studies different set of features from encoder-decoder neural networks based architectures for the Spoken Language Understanding (SLU) task of DECODA theme identification task of spoken dialogues. These dialogues are transcribed from an Automatic Speech Recognition (ASR) system. These automatic transcripts are affected by errors from automatic transcription process that are especially frequent since speech signal in recorded in noisy conditions. An ASR system induces linguistic errors in features employed for different NLP related applications. This chapter deals with the recovery of corrupted linguistic features in spoken documents by introducing robust features from encoder-decoder neural networks based models evaluated in the real-world application of the DECODA spoken conversation analysis task.

Part II details novel neural networks based architectures partially realized in collaboration with Montréal Institute of Learning Algorithme (MILA) researchers. Among them, Y. Bengio, M. Ravanelli and others have collaborated on Quaternion valued neural networks for speech processing. We have proposed novel research directions to introduce complex-valued neural networks to better manage internal dependencies between features of a basic element. Therefore, chapter 3 presents these promising models based on convolutional neural networks and the Quaternion hypercomplex numbers called "Quaternion Convolutional Neural Network" (QCNN). We report in this thesis the QCNN evaluated on an NLP related task (Theme identification task of spoken dialogues). Chapter 4 introduces the Quaternion Recurrent Neural Networks (QRNN) and an extension of the LSTM to Quaternion algebra called QLSTM. These models are evaluated during the speech recognition task of TIMIT. This work have been realized during the Ph.D. thesis of Titouan Parcollet co-supervised with Georges Linarès (LIA) and during the internship of Titouan Parcollet at the MILA (Canada) under the co-supervision of Yoshua Bengio.

Part III of the thesis corresponds to the ongoing and futur research directions. We first present an ongoing research that will lead to part of my futur research. The previous chapters chart the path of my research throughout different neural networks

based topics and language processing applications. These models compose the first step of my research in the area of machine learning for NLP and require a large even huge processing time during the learning process. Chapter 5 introduces novel promising research perspectives for faster and more efficient neural networks based models and exposes a research direction based on a ongoing work for parsimonious recurrent neural network. Nowadays even more real-life NLP related applications are based on deep learning algorithms. These models require a long processing time for the learning phase. The proposed model allow NLP based systems to achieve better results than other RNN architectures with less processing time. The work started on studying encoder-decoder neural networks is extended in chapter 6 with the first efforts on proposing novel generative models to extract from noisy spoken documents robust features for SLU. Finally chapter 7 presents future short-term directions on both real and quaternion-valued neural networks, and long-term perspectives on parsimonious neural networks and GAN. Overall, this thesis is an overview of my futur directions in terms of neural based architectures (RNN, GAN, ...) and real-world applications (find out more efficient algorithms for NLP tasks) in collaboration with different Ph.D. students, interns and colleagues.

Part I

Real-valued Neural Networks for Natural Language Processing

Chapter 1

Parallel Recurrent Neural Networks

Contents

Abstract

We present in this chapter an original LSTM-based architecture, named Parallel LSTM (PLSTM), that carries out multiple parallel synchronized input sequences in order to predict a common output. The proposed PLSTM could be used for parallel sequence classification purposes. The PLSTM approach is evaluated on the automatic telecast genre sequences classification task. Results show that the proposed PLSTM method outperforms the baseline n-gram models as well as the state-of-the-art LSTM approach. This chapter corresponds to part of the work done during Mohamed Bouaziz's Ph.D. thesis, in CIFRE convention with the company EDD and co-supervised by Georges Lianrès (LIA), Richard Dufour (LIA) and myself between September 2013 and December 2017.

1.1 Introduction

The work presented in this chapter is from the collaboration with Mohamed Bouaziz during his Ph.D. thesis in CIFRE[1] with the EDD (L'Europèenne de Donnèes) company from Paris (France). EDD processes large amount of multimedia documents and streams to provide for the customers a set of speech and natural language related services such as documents retrieval and indexation. In this context, the Ph.D. thesis has focused on the prediction of the genre label of telecasts on Electronic Program Guide (EPG) multi-streams of different channels. Nonetheless, the proposed architecture can easily process different multidimensional data or streams. The context of the Ph.D thesis of EPG prediction, has many concerns related to the difficulty to express an event (telecast genre) given a sequence of previous telecasts that are not taking place at the same time each day. Therefore, the use of different parallel channels gives a more robust context to predict the genre label for a given channel. Moreover, one can find, on the internet, an impressive list of programs available on demand at any time of the day. This forces the original TV channels to diversify their offers and to propose programs that best meet the expectations of viewers regarding the period of the day. However, this large amount of streams needs to be analyzed and organized to allow the consumers and industries to process effectively the mass media content. Scientists have been given a certain level of interest to automatic sequence classification of TV channels' speech in a multi-channel context (Bredin et al., 2014; Bouchekif et al., 2013). Indeed, recently, automatic sequence classification became an ubiquitous problem, having then encountered a high research interest (Gers et al., 2001; Severyn et Moschitti, 2015; Huang et al., 2016). This is due to the need to structure knowledge as a set of dependent localized information alongside with the new computer capabilities to efficiently process large amount of data. Among the recent methods employed to structure these sequences, the machine learning domain provides a set of high-level representations well adapted to automatic sequence classification based on Deep Neural Networks (DNN) such as Convolutional Neural Networks (CNN) (LeCun et al., 1998) or Recurrent Neural Networks (RNN) (Elman, 1990). RNN architectures such as Long Short-Term Memory (LSTM) (Hochreiter et Schmidhuber, 1997) and Bidirectional LSTM (BLSTM) (Graves et Schmidhuber, 2005) have gained a particular attention in different domains and tasks including sentence (Sundermeyer et al., 2012) or successive images (Vinyals et al., 2015) processing. Nonetheless, RNNs or RNN-HMM could not be directly employed for sequence classification using multiple inputs from synchronous streams such as TV shows coming from different channels. Indeed, RNNs can only be trained to make a set of elements labeled in a single stream of input information.

We have introduced an original multi-stream neural network architecture, called Parallel LSTM (PLSTM), that simultaneously takes into account different synchronous streams in order to automatically classify this multi-stream sequence. To evaluate the effectiveness of the proposed PLSTM multi-stream neural network architecture, exper-

[1]The CIFRE convention (Conventions Industrielles de Formation par la REcherche) is a partnership between academics and industrials for granting a Ph.D. thesis with mainly an real world application purpose. More information on the CIFRE convention are available at http://www.anrt.asso.fr/fr/cifre-7843

iments are carried out on the LIA's Electronic Program Guide (EPG) dataset containing 3 years of TV programs from 4 different channels. The PLSTM performance is compared with the LSTM as well as a classic n-gram models considered as the baseline. Our PLSTM approach is an important step further for sequence classification since it can be applied to any set of synchronous sequences. The annexe proposes an overview of a couple of RNN architectures.

1.2 Parallel Long Short-Term Memory (PLSTM)

Sequence models, such as LSTM based RNNs, are well known for their capability to classify sequential data. On the other hand, these models can only take into account homogeneous sequential data from a single stream. In the case of additional sources, these models can not be employed. In order to take into account parallel sequences, a solution is to combine these streams viewed as a single sequence. This combination is carried out by two models with a certain number of limits.

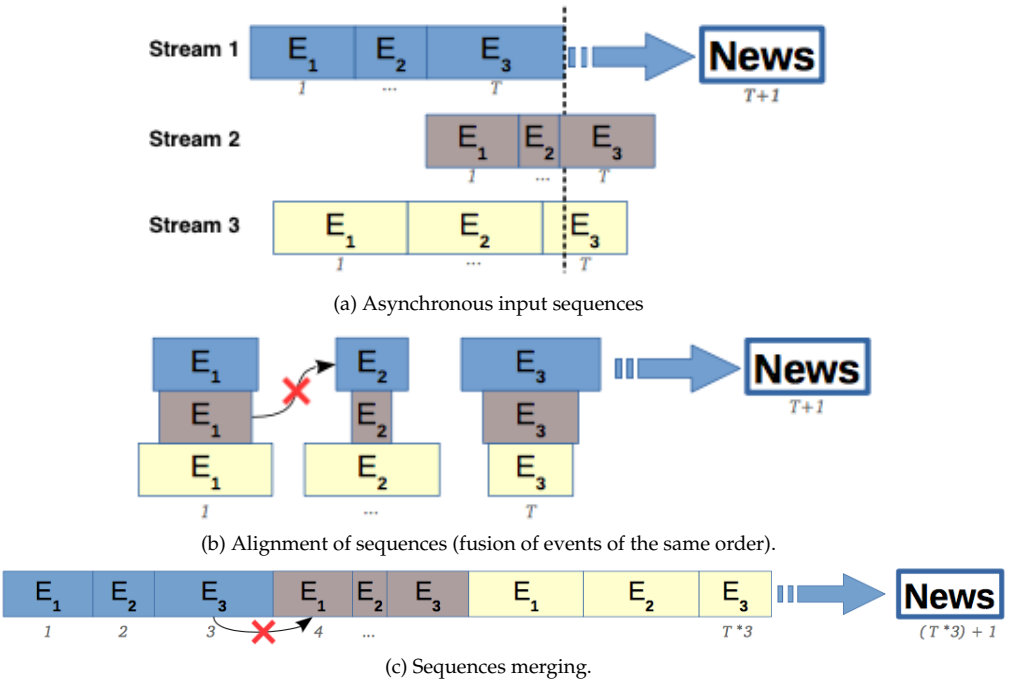


Figure 1.1: *Combination of parallel sequences: an illustrative example with 3 streams on the last 3 events : E_t coressponds to the t^{th} event.*

Multi-streams input features of the PLSTM

The first method of combining the parallel sequences consists of a kind of alignment between the different sequences in order to merge the events of the same order within a single event (see Figure 1.1-b). It is a question of combining the event terms

($1 \leq t \leq T$) of each of the N sequences to form the compound t^{th} event. The sequence of T compound events is then used as a conventional sequence where each event t is represented by a vector of N events. The limitations of this method ere related to, for example, an LSTM layer will consider a recurrent link between the t^{th} and the $(t+1)^{\text{th}}$ composed event. As a consequence, a dependence between the streams will be learned, wrongly, between the t^{th} basic event of the stream n' and the $(t+1)^{\text{th}}$ basic event of another stream n'' ($n' \neq n''$, $1 \leq n' \leq N$ and $1 \leq n'' \leq N$). In Figure 1.1-b, an example of this potentially wrong dependency is between the basic event at $t = 1$ of stream 2 and the elementary event at $t = 2$ of stream 1. The first event appends after the second one The second method concatenates the N parallel sequences, each containing T events, to obtain a single unidimensional "super-vector" of length $T \times N$ (see Figure 1.1-c). the drawback of this method of concatenation is that the vector produced does not represent a sequence but rather a sequence of sequences. Models adapted to sequential data consider any input data as a homogeneous sequence of events. The combination of multi-stream sequential data and their inclusion in sequence models is, in most of the cases, irrelevant. We nevertheless retain the use of the first method of combining the parallel sequences since the latter is more prone to practical order issues.

PLSTM model

The Bidirectionnal-RNN (BRNN) neural architecture, presented in the annexe, uses the same sequence $\mathbf{x}$ as an input for both forward and backward directions, which is useful for information from a single stream. We propose in this chapter an original neural recurrent neural network, called Parallel RNN (PRNN) presented in Figure 1.2, that takes advantage from the BRNN structure in a multistream context. The original PLSTM architecture corresponds to the PRNN description by replacing the $\mathcal{H}$ function with the LSTM composite function. PLSTM differs from the classical BLSTM by feeding forward, not a shared sequence, but different input vectors through a dedicated hidden layer $\mathbf{h}^n$ for each input vector $\mathbf{x}^n$. Moreover, BLSTM employs only 2 hidden layers due to its bidirectional architecture while PLSTM is potentially composed with multiple hidden layers. The input sequences are considered independent and require to be mapped in homogeneous separate subspaces ($\mathbf{W}$ matrix from input $\mathbf{x}$ to hidden $\mathbf{h}$ spaces). Therefore, a single LSTM containing concatenated inputs from different independent sequences is not theoretically suitable for finding out a common homogeneous subspace to map heterogeneous input representation of parallel sequences. Thus, for each n^{th} stream ($1 \leq n \leq N$), the PLSTM takes the input sequence $\mathbf{x}^n = (x_1^n, x_2^n, \ldots, x_T^N)$ and computes the hidden sequence $\mathbf{h}^n = (h_1^n, h_2^n, \ldots, h_T^N)$ and the output vector $\mathbf{y}$ by iterating from $t = 1$ to T.

$$h_t^N = \mathcal{H}(\mathbf{W}_{x^N h^N} x_t^N + \mathbf{W}_{h^N h^N} h_{t-1}^N + b_h^N) \tag{1.1}$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \tag{1.2}$$

$$h_t^2 = \mathcal{H}(\mathbf{W}_{x^2 h^2} x_t^2 + \mathbf{W}_{h^2 h^2} h_{t-1}^2 + b_h^2) \tag{1.3}$$

$$h_t^1 = \mathcal{H}(\mathbf{W}_{x^1 h^1} x_t^1 + \mathbf{W}_{h^1 h^1} h_{t-1}^1 + b_h^1) \tag{1.4}$$

$$y_t = \sum_{n=1}^{N} \mathbf{W}_{h^n y} h_t^n + b_y \tag{1.5}$$

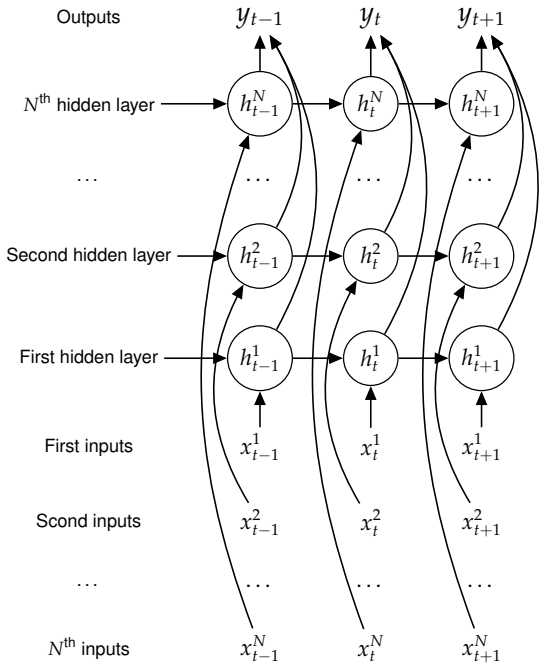


Figure 1.2: *Parallel Long Short-Term (PLSTM) neural network.*

where N is the number of streams. In our experiments, the output vector $\mathbf{y}$ takes advantage of the N channels to predict the telecast's genre for one given channel n ($1 \leq n \leq N$).

Therefore, PLSTM feeds forward separate sequences in order to predict a label and codes internal hidden structures between the parallel hidden sequences. (Graves et Schmidhuber, 2005) introduces the BLSTM with Back Propagation Trough Time (BPTT) algorithm (Schuster, 1999) for training. For our proposed PLSTM architecture, the training takes place over N input sequences:
Forward Pass: feeds all input data for the sequences into the PLSTM and determines the predicted outputs.

- For 1 to T: Forward pass for each of the N hidden layers.
- For all $t \in [1 \dots T]$: perform the forward pass for the output layer using the activation state of the N hidden layers.

Backward Pass: processes the error function derivative for the sequences used in the forward pass.

- For all $t \in [1 \dots T]$: perform the backward pass for the output neurons.
- For t from T to 1: perform the backward pass for each hidden layers using the error observed on the output layer.

Updating Weights The update weight for all matrices from each hidden state related to each stream is computed in the same manner than for the mere LSTM (Hochreiter et

Schmidhuber, 1997).

1.3 Experiments

Multistream sequence classification is evaluated with the proposed PLSTM architecture (2 and 4 parallel sequences) as well as the classic LSTM network on an automatic TV show genre labeling task. Two n-gram based models (baseline) are also considered for fair comparison. Next sections describe the dataset, the genre sequence classification as well as the neural networks settings.

Multichannel EPG dataset
The Electronic Program Guide (EPG) dataset is extracted from 4 French TV channels (M6, TF1, France 5 and TV5 Monde) for 3 years, from January 2013 to December 2015. M6 channel is used in our experiments as the output stream. Data from *2013* and *2014* are merged and split into the *training* (70%) and *validation* (30%) datasets using a *stratified shuffle split* (Pedregosa et al., 2011) in order to preserve the same percentage of samples of each class in the output of both folds, while the *2015* dataset is kept for testing. In order to guarantee a clean experimental environment, labels (*i.e.* genres) that are absent at least in one of the three folds were removed. Doing so allows us to have equivalent datasets in terms of labels vocabulary. Table 1.1 shows the genres distribution for M6, the chosen output channel.

Genres	Training	Validation	Test
Weather	2,691	1,153	1,712
Fiction	1,890	810	1,478
News	913	392	679
Other magazine	981	421	466
Music	461	197	340
Teleshopping	421	180	308
TV game show	476	204	287
Cartoon	361	155	205
Other	277	119	133
Reality TV	83	36	76
Documentary	29	13	14
Total	8,107	3,680	5,698

Table 1.1: *Genres Distribution for train, validation and test sets in M6 channel output.*

Genre Prediction Experiments
For a given input history sequence (composed of the n previous telecast genres), a genre label representing the next M6's telecast is output. The size of the genre sequences (n) varies from 1 to 4. Then, three input configurations are employed.

Mono-channel input: only M6 history sequences for a baseline *n-gram* experiment (with a *statistical language model* from the SRILM toolkit (Stolcke et al., 2002)) and a straightforward *LSTM* model. **Bi-channel input:** both M6 and TF1 channel histories are employed as input for *P2LSTM* (PLSTM with two parallel streams as a BLSTM with forward-forward directions and separate inputs). The aim of this experiment is to move up the context's information using a similar and rival channel, the two being generalist channels. **Multichannel input:** History of each of the 4 streams (*i.e.* channels) is used as input for *4n-gram* and P4LSTM experiments (PLSTM with 4 parallel streams).

Neural Networks Setup
The mere LSTM, and the proposed P2LSTM and P4LSTM, are composed with 3 layers: input layer **x** of size varying from 1 to 4, a hidden layer **h** of size 80 for all LSTM-based models and an output layer **y** with a size equals to the number of different possible TV genres (11). The Keras library (Chollet, 2015), based on Theano (Bastien et al., 2012) for fast tensor manipulation and CUDA-based GPU acceleration, has been employed to train neural networks on an Nvidia GeForce GTX TITAN X GPU card. The training times, detailed in Table 1.2 for all models, match with the sequence size of all models. Indeed, even with the most time-consuming configuration, namely P4LSTM with 4 elements history, the training does not last more than 25 minutes.

Sequence size	**1**	**2**	**3**	**4**
n-gram	1	1	1	1
4n-gram	2	5	17	51
LSTM	51	146	319	362
P2LSTM	259	473	485	439
P4LSTM	536	923	844	1,386

Table 1.2: *Training times (in seconds) of models employed during the experiments for different telecast genres sequence sizes.*

1.4 Results and Discussion

Table 1.3 shows the overall results, in terms of the standard F1 metric related to the genre prediction task outputs, using each method and for different stream sequence sizes from 1 to 4.

Seq. size	**n-gram**	**4n-gram**	**LSTM**	**P2LSTM**	**P4LSTM**
1	19.07	59.39	11.60	47.46	47.24
2	51.38	58.35	34.64	54.17	59.54
3	57.41	57.10	50.74	58.69	59.92
4	56.87	57.26	56.47	58.67	**60.81**

Table 1.3: *F1-score (%) of each n-gram and LSTM models.*

N-gram based models
The multi-channel 4n-gram model outperforms the simple n-gram one for each of the different 4 genre sequence configurations except for 3 sized history. *4n-gram* method reaches around 59% of F-score using 1 sized sequences against near 57% for mono-channel n-gram using its best history configuration. The observed results confirm the interest of using multiple streams to predict the next telecast's genre for a specified channel.

LSTM and PLSTM
One can figure out from Table 1.3 that mono-channel LSTM does not even outperform the mono-channel n-gram experiment. P4LSTM obtains the best result with an F1 score close to 61% using a sequence of size 4. In order to analyze these results, the Error Rates (ER) are also presented in Table 1.4. The overall F1 scores are different from those related to the ER. For example, at its best configuration of a 4 sized sequence, P4LSTM error rate reaches about 23.5%, which corresponds to a correct rate of 76.5% against an F1-measure of only 61%. Moreover, although the range between the lowest and the highest values is almost 12 points for ER, it is only near 4 points for F1-measure. The reason of this is that the F1-metric may be not suitable for the task due to the labels imbalance with different numbers of genre occurrences varying from 14 to 1,712 in the test set.

Seq. size	n-gram	4n-gram	LSTM	P2LSTM	P4LSTM
1	51.36	30.29	60.13	36.68	34.35
2	39.06	28.99	48.99	29.38	25.08
3	31.71	29.64	35.73	28.85	24.76
4	35.43	30.5	29.59	28.27	**23.52**

Table 1.4: *Error rates (ER) observed for each n-gram and LSTM models for different sequence sizes.*

Discussion
Confusion matrices of 4n-gram and P4LSTM methods are shown in Tables 1.5 and 1.6 to point out benefits of the proposed PLSTM model.

It is worth emphasizing that most of the missed instances in all systems are wrongly labelled as one of the two most frequent classes, *Weather* and *Fiction*, as well as the *Other Magazine* genre (some examples are in green cells). *False positives* are more recurrent in some small classes such as *Other Magazine* than in the relatively more frequent class *News*. The reason is that *News* is a well defined genre occurring mostly at the same time each day unlike *Other Magazine* genre that encompasses various telecasts that are broadcast at several and irregular daytimes. *Teleshopping* shows are often broadcast at more nearly the same time of morning than *Cartoons* and affect dramatically the performance of the 4n-gram model in this context (cf. underlined italic cell in Table 1.5). Finally, the confusion matrix of P4LSTM experiment shows also that this system fails to predict the least frequent genres *Others*, *Reality TV*, and *Documentary*. For example, for the two least frequent genres, *Reality TV* and *Documentary*, respectively none of the 76 and the

1282	**65**	0	**323**	11	4	17	0	5	2	3
292	**904**	0	**177**	35	9	15	26	16	0	4
2	12	**636**	1	27	0	0	0	1	0	0
69	**40**	3	**296**	10	2	6	0	29	6	5
16	**35**	17	5	**219**	0	3	2	43	0	0
45	1	0	1	0	**246**	0	0	15	0	0
2	7	0	19	0	0	**245**	0	6	8	0
0	32	6	0	9	***102***	0	**56**	0	0	0
10	10	0	**25**	1	2	15	0	**66**	2	2
20	14	0	9	0	0	4	0	19	**9**	1
4	3	0	**4**	0	0	0	0	2	0	**1**

Table 1.5: *Confusion matrix for the 4n-gram output: labels are shown according to their decreasing frequency as in Table 1.1.*

1624	**38**	0	**46**	0	2	0	0	2	0	0
375	**861**	0	**188**	18	0	18	16	2	0	0
4	0	**669**	1	5	0	0	0	0	0	0
114	**34**	0	**306**	6	0	3	0	3	0	0
15	18	0	1	**297**	0	7	2	0	0	0
49	15	0	0	0	**244**	0	0	0	0	0
49	14	0	14	11	0	**199**	0	0	0	0
25	**28**	1	0	0	4	0	**147**	0	0	0
70	3	0	**48**	0	1	0	0	**11**	0	0
35	0	0	**23**	0	0	14	0	4	**0**	0
8	0	0	**5**	0	0	1	0	0	0	**0**

Table 1.6: *Confusion matrix for the P4LSTM output: labels are shown according to their decreasing frequency as in Table 1.1.*

14 instances were correctly found. This leads to a *precision* of 0 which penalizes the average precision and then the overall F-score. In order to evaluate the impact of the

Seq. size	**n-gram**	**4n-gram**	**LSTM**	**P2LSTM**	**P4LSTM**
1	23.30	70.32	14.18	58.00	57.74
2	59.47	68.04	42.34	66.21	72.77
3	66.71	66.55	62.01	71.74	73.23
4	65.67	66.77	69.01	71.71	**74.32**

Table 1.7: *F1 score (%) of n-gram and LSTM models, the two least frequent genres Reality TV and Documentary not being included.*

least frequent genres on the F1 measure, especially on the three LSTM based systems, we also reported on Table 1.7 the F1 results on the same outputs of the experiments of Table 1.3 by excluding the two least frequent genres from the averages of precision and recall (*Reality TV* and *Documentary*). The state-of-the-art mono-channel LSTM per-

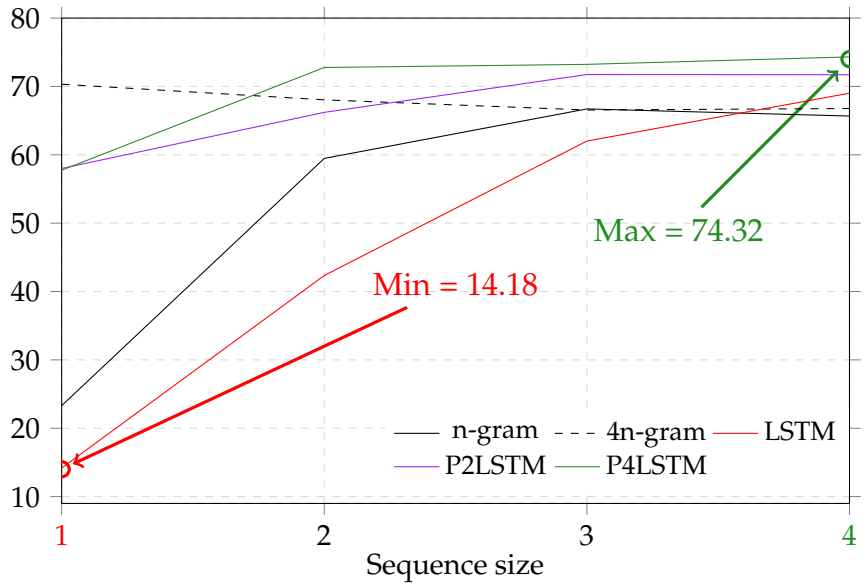


Figure 1.3: *F1 score for n-gram and LSTM models, the two least frequent genres Reality TV and Documentary not being included.*

formances gradually become closer and closer to the multichannel n-gram model ones (cf. Figure 1.3) when the size of sequences moves up with an F1 score of 69%. Therefore, LSTM-based models require longer sequences to learn long term dependencies than the 4n-gram methods. Overall, the results of the PLSTM detailed in Table 1.4 and Figure 1.3, demonstrate the benefits obtained at least for history sequences longer than 2 genres with an F1 score greater than 71%. Regarding multichannel P4LSTM approach, the highest performance reaches an F1-measure of about 74% using 4 sized sequences with a gain of about 3 and 4 points compared respectively to P2LSTM and 4n-gram model best performances. With the rapid grow of multimedia documents such as videos, the results observed with the PLSTM are promising and will allow NLP systems to process multidimensional data. Indeed, researchers across pattern recognition domains propose more and more neural networks based systems to process effectively multi streams for image processing (Singh et al., 2016), video classification (Wu et al., 2016) or speech processing (Ma et al., 2019). This work will be followed to propose novel neural based architectures to better consider each stream in an appropriate condition. For example, LSTM cells are composed with gates that manage in different level the knowledge stream. Indeed, some gates are devoted to expose the output or forget the hidden state. Nonetheless, the input gate has to propose a suitable hidden state candidate for a given stream but all streams have not enough information on this given stream to effectively provide part of the hidden state candidate.

Chapter 2

Encoder-Decoder Neural Networks

Contents

Abstract

This chapter corresponds to part of the work done during the Killian Janod's Ph.D. thesis, in CIFRE (Orkis) where we study different set of features from different encoder-decoder neural networks for the DECODA theme identification task of spoken dialogues. These dialogues are transcribed with an Automatic Speech Recognition (ASR) system and are affected by errors that are especially frequent when speech signal is recorded in noisy conditions.

2.1 Introduction

The other research direction related to hitherto proposed real-valued neural networks for language processing is presented in this chapter. This work has been made during the Ph.D. thesis of Killian Janod. The aim of this work is to find robust representations of spoken documents automatically transcribed by an Automatic Speech Recognition System (ASR). These transcripts are error prone and NLP applications require robust features representing the document content to process effectively these documents. These features for characterizing a document or a conversation can be words, their relations in sentences or discourse structures. Selecting and extracting relevant features of these types depend on the analysis task and is, in general, a very difficult and not completely solved research problem. The problem is even more complex for spoken conversations since words are hypothesized by an ASR component with transcription errors partly due to speech signal variability caused by speaker characteristics, environment and channel noises, or acquisition devices, as well as a highly conversational language register (hesitations, repetitions, grammatical errors, disfluencies...). Nevertheless, important and applicable results have been obtained on specific tasks as speech analytics (Melamed et Gilbert, 2011), topic identification (Lagus et Kuusisto, 2002; Hazen, 2011) and segmentation (Eisenstein et Barzilay, 2008; Purver, 2011), and other spoken conversation analysis tasks reviewed in (Tur et De Mori, 2011). We have proposed a set of neural networks based models to reduce the impact of ASR errors on the identification of a predetermined set of topics assuming that a spoken document can be classified as belonging to one and only one topic as discussed in (Hazen, 2011). This assumption is acceptable if the task is to classify a single topic corresponding to the theme of a conversation. An example of conversation theme is the type of customer concern/request/complaint expressed during a customer care service (CCS) conversation. A fragment of a conversation of this type in the application domain is shown in Figure 2.1. The basic problem is to reduce the effect of ASR errors. For this purpose, two different objectives have been identified, inspired by two deep learning approaches: to consider ASR errors as noise to be reduced with denoising autoencoders (DAE); to make abstractions of ASR features with stacked autoencoders to obtain distributed representations that tend to capture features of semantic contents characterizing conversation themes. These features are expected to produce an effective generalization useful for training classifiers in specific domains (Hinton et Salakhutdinov, 2006; Bengio et al., 2007). The conversation is represented (input features) by a product of word term frequency (TF), inverse document frequency (IDF) as suggested in (Hazen, 2011), and posterior probabilities of a theme given a term. The input is then processed by a stack of autoencoders and a DAE to obtain robust latent features sets to be used for hypothesizing conversation themes with a multi-layer perceptron (MLP) classifier. When DAEs are used, the input variability caused by ASR errors is considered and reduced in the attempt to recover the same type of features obtained with manual transcripts. When stacked autoencoders (SAE) are used the objective is to reduce variability of ASR features without imposing the reconstruction of clean features. The performances obtained with DAEs and SAEs features are compared in a theme identification task. Moreover, effective encoder/decoder architectures are proposed for

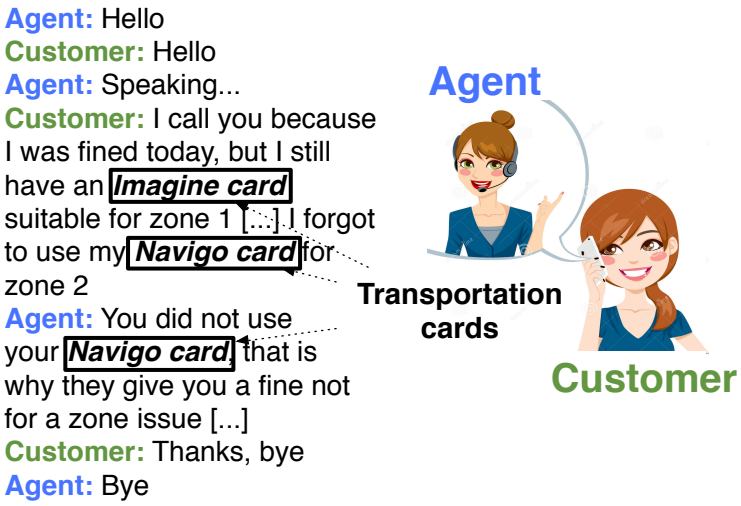


Figure 2.1: *Example of a dialogue from the DECODA corpus labelled by the agent as a transportation card issue but also containing the infraction theme.*

reconstructing bottleneck features obtained with manual transcripts using bottleneck features obtained from ASR transcripts. These architectures are inspired by the consideration that bottleneck features can be further processed to encode a level of abstraction eliciting relevant semantic contents encoded in the input features. A further investigation is proposed for adapting the reconstruction process to a specific task. Experimental results confirm that the proposed novel autoencoders that integrate corrupted and clean feature abstractions provide better evidence of semantic contents relevant for a topic identification task.

2.2 Autoencoder neural networks based systems

We have proposed to consider ASR transcripts as noisy data and regards transcription correcting as denoising. A discussion in (Turian et al., 2010) suggests that neural word representations may capture word semantic information. These representations can be obtained with unsupervised learning of autoencoder parameters. As a first step, the possibility is investigated that hidden features obtained with simple and stacked autoencoders embed sufficient semantic information for characterizing themes of noisy conversations. As a second step, hidden features obtained with denoising autoecoders are considered to evaluate the impact of training for reconstructing an input artificially perturbed with a noise model. As manual transcriptions are available for a train set, in a third step, features are extracted with semi-supervised reconstruction of manual transcription features from ASR noisy transcription features. Notice that manual transcriptions features can be perturbed by a variability induced by the fact that CCS users may mix information relevant for a theme with personal or factual irrelevant information. As a fourth step, the possibility of stacking hidden layers of features estimated with the previously mentioned autoencoders is considered with or without

global fine-tuning.

Document representation
The task considered in this chapter is the reconstruction of a feature distribution corrupted by the imprecision of the feature extraction component. Features are obtained from a set of content words exhibiting high mutual information with the application domain themes. Given a document of a corpus $\mathbb{D}$, an input feature $x_d \in \mathbb{R}^n$ called TF-IDF-Gini is defined with its i-th element computed as follows:

$$x_i = tf(i) \times df_{\mathbb{D}}(i) \times G(i) \tag{2.1}$$

where $tf(i)$ the number of occurrences of terms i in the document, $df_{\mathbb{D}}(i)$ is the inverse document frequency (IDF) and $G(i)$ the word purity. The purity of a feature f is scored with the Gini criterion defined as follows:

$$G(f) = \sum_{t \in \mathbb{T}} \mathbb{P}^2(t|f) = \sum_{t \in \mathbb{T}} \left(\frac{df_t(f)}{df_{\mathbb{D}}(f)} \right)^2 \tag{2.2}$$

where $df_t(f)$ is the number of document containing the term f annotated with label t.

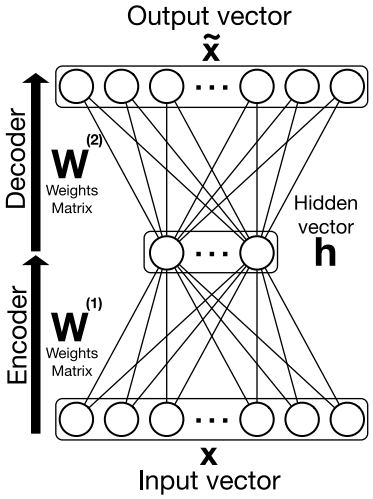


Figure 2.2: *Autoencoder model. Biases are omitted for the sake of simplicity.*

Basic Autoencoder Concepts
An Autoencoder (AE) is a three-layer feed-forward neural network made of an encoder and a decoder as shown in Figure 2.2. The encoder computes a hidden representation of x made of a vector h of size m (number of hidden units) as follows:

$$h = \sigma(W^{(1)}x + b^{(1)}) \,, \tag{2.3}$$

where W^1 is a $m \times n$ weight matrix and $b^{(1)}$ is a m-dimensional bias vector. $\sigma(.)$ is a non-linear activation function. In the approach proposed in this chapter the function is

the hyperbolic tangent defined as follows:

$$\sigma(y) = \frac{e^{y} - e^{-y}}{e^{y} + e^{-y}} \tag{2.4}$$

The decoder attempts to reconstruct the input vector x from the hidden vector h to obtain the output vector $\tilde{x}$:

$$\tilde{x} = \sigma(W^{(2)}h + b^{(2)}) \,, \tag{2.5}$$

where the reconstructed vector $\tilde{x}$ is a n-dimensional vector, $W^{(2)}$ is a $n \times m$ weight matrix and $b^{(2)}$ is a n-dimensional bias vector. AE parameter estimation is performed by minimizing the following reconstruction Mean Square Error (MSE) L_{MSE} with respect to the set of parameters $\theta = \{W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}\}$ using the MSE (Bengio, 2009) :

$$L_{\text{MSE}}(\theta) = \frac{1}{d}\sum_{x \in D} l_{\text{MSE}}(x, \tilde{x}) = \frac{1}{d}\sum_{x \in D} ||x - \tilde{x}||^2 \tag{2.6}$$

Two AEs are trained for the task considered in this chapter: one for reconstructing features ($x^{(\text{ASR})}$) of automatic transcriptions from an ASR system, and the other one to recount features ($x^{(\text{TRS})}$) of manually transcribed documents. The parameters estimated for these autoencoders ($W^{(\text{ASR})}$ and $W^{(\text{TRS})}$) are pre-trained for further use to estimate the bottleneck features (hidden layer) of the Supervised deep autoencoder.

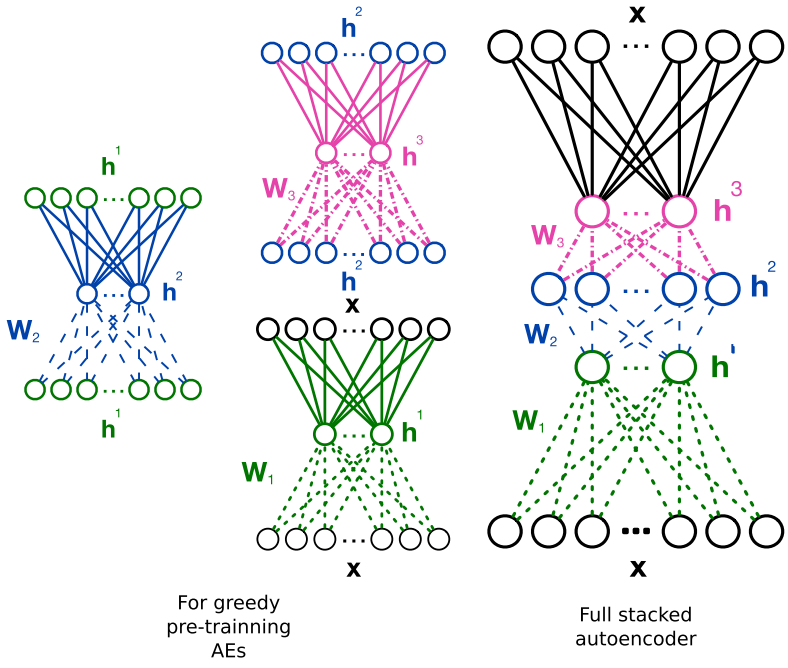


Figure 2.3: *Stacked autoencoder (SAE) architecture (Janod et al., 2016). The parameters of each layer are estimated as for a shallow autoencoder (left). After a pre-training step, the hidden layers are stacked to form a denoising autoencoder (right) whose parameters are further fine-tuned by global error back propagation.*

Stacked Autoencoders (SAE)
It has been argued in (Bengio et al., 2007; Hinton et al., 2006) that DNNs may encode input data at progressively deeper levels of abstraction in successive hidden layers of stacked autoencoders (SAE). In this type of DNN with k hidden layers, the latent features at the i-th intermediate hidden layer, for an input vector x, are computed as with equation 2.3 by setting $h^{(0)} = x$, and $h^{(i)} = \sigma(W^{(i)}h^{(i-1)} + b^{(i)})\forall i \in \{1, 2, \ldots, k\}$. Therefore, each layer is pre-trained as a shallow autoencoder for a fixed number of iterations. The learnt hidden layer vector $h^{(i)}$ is stored and used to learn the next layer $h^{(i+1)}$. Greedy pre-training is progressively performed in this way starting with $h^{(i+1)}$ as shown by Figure 2.3. After pre-training the last layer, a fine-tuning is performed on the entire stack of hidden layers to obtain a generative model providing different levels of abstractions for the input vector x.

Denoising Autoencoder (DAE)
The aim of an autoencoder is to build a robust generative model to encode and decode a given input vector x in a latent space h. During the learning process, the autoencoder fails to completely separate relevant information from residual noise for a given input distribution (Vincent et al., 2010). For this reason, (Vincent et al., 2010) propose to corrupt the inputs before the encoding process and then decode this noisy representation to a clean one with a Denoising Autoencoder (DAE). In this way, the DAE is expected to recover a clean representation from a noisy input by learning a robust generative model. In this model, the input vectors x are considered as "clean" representations. The aim of this DAE is to obtain a robust reconstruction from an input vector to a clean output one. Therefore, x is artificially corrupted via a function that can be:

- An Additive isotropic *Gaussian Noise* (GS) $x^{(\text{corrupted})}|x \sim \mathcal{N}\left(x, \sigma^2 I\right)$
- A *Masking Noise* (MN) which forces a fraction of the element of x to 0.
- A *Salt-and-pepper Noise* (SN) which forces the values of some elements of x to their minimum or maximum possible value (0 or 1) according to a fair coin flip (Vincent et al., 2010).

This corrupted input $x^{(\text{corrupted})}$ is then mapped to a hidden layer with the (3) by replacing x with $x^{(\text{corrupted})}$. During the learning process, the denoising autoencoder learns the parameters $\theta = \{W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}\}$ to minimize the reconstruction error $L(x, \tilde{x})$. The motivation for using this type of DAE is that a good representation h of a corrupted or partially destroyed representation of an input vector x is informative of x and invariant to a perturbation $x^{(\text{corrupted})}$ of x due to noise. The conditional distribution used to generate the corrupted version $x^{(\text{corrupted})}$ is artificially induced. The problem considered in this chapter is different since input features are extracted from already noisy real-life data. In fact, input features originate from conversations recorded with real-life background noise. Such a noise is unpredictable and a noise model cannot be identified as the corruption of a clean input. This motivates the proposal of a new approach based on hidden bottleneck features and the incorporation

of both noisy and clean documents in the learning process.

Basics Concepts on Supervised Bottleneck Features
Multi-Layer Perceptron (MLP) are feed-forward neural network made of an input layer, an output layer and at least one hidden layer. MLP networks are known to be a universal function approximator (Haykin et Network, 2004). In the work described in this chapter, the MLPs are used for document classification. For each hidden layer, a latent representation of the input is computed using the (3) with the same setting as for the stacked autoencoders. Usually the class is predicted by output layers using a non-linear Softmax function defined by:

$$\sigma(y) = \frac{e^{y_j}}{\sum_{k=0}^{K} e^{y_k}} \text{ for } j \in 1, \ldots, K \tag{2.7}$$

Where K is the number of elements in y. The learning process attempts to reduce the parameters $\theta = W_0, ..., W_j, b_0, ..., b_j$ to minimize the prediction error L(x,$\tilde{x}$). The state-of-the-art loss function used in an MLP is the Cross Entropy Error (Nasr et al., 2002). If the MLP parameters are estimated with output supervision, the latent representation encoded in each hidden layer is more abstract and more meaningful for the task than the one computed just for reconstructing the previous layer (Haykin et Network, 2004). Bottleneck features are those of the hidden layer that provides the more meaningful, abstract and if possible compressed latent representation preserving at the same time enough information of the original distribution useful for higher level processing. Bottleneck features have been intensively used in neural acoustic models (Grèzl et al., 2007) (Dong Yu, 2011). The MLP based classification is used in this chapter for comparing the performance a variety of autoencoder latent features. This choice is considered to be acceptable, at least for the considered application type, since the best classification results obtained in this way are superior to those reported in (Esteve et al., 2015) using a deep neural network and the addition of semantic and LDA input features.

Proposed Features from a Supervised Deep Autoencoder
The goal of this chapter is to obtain a robust representation of a malformed and noisy document transcribed by an ASR system. This robust representation is based on latent features of a Supervised Deep Autoencoder (SDAE), using both automatic and manual transcriptions. For this reason, a new architecture shown in Figure 2.4-(c) is proposed in this chapter. The parameter estimation of some components of this architecture is initialized using the estimations obtained with the networks shown in Figures 2.4-(a) and 2.4-(b). The architecture in Figure 2.4-(c) uses, as input, a feature vector from imperfect automatic transcripts (ASR) (considered as a corrupted input), and an output feature vector from clean manual transcripts (TRS).
In this architecture, ASR bottleneck features represented by vector h^{ASR} are mapped into TRS bottleneck features represented by vector h^{TRS} by introducing a mapping layer of hidden features represented by vector h .
Initialization: As emphasized in (Hinton et al., 2006), the parameters initialization in the architecture of Figure 2.4-(c) is critical. For this reason, the estimation should

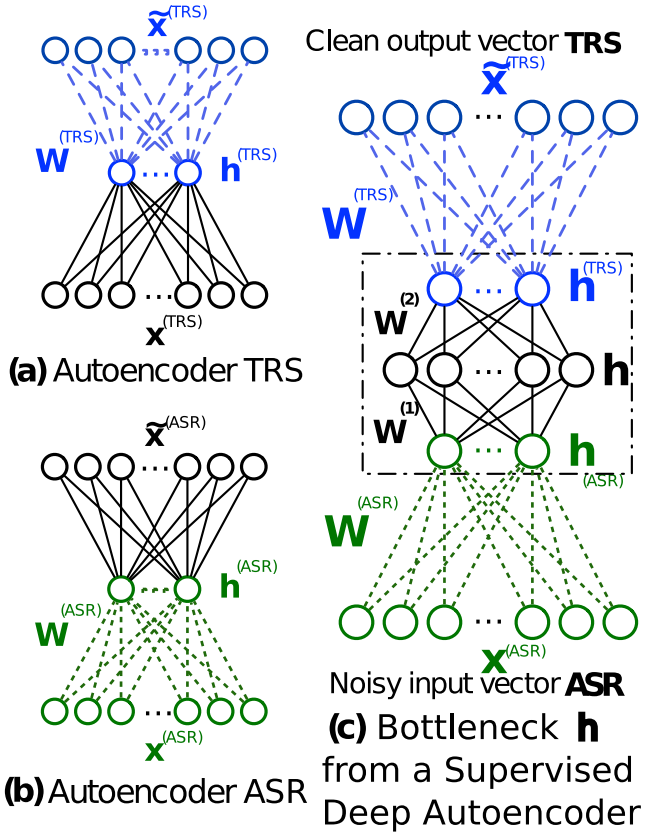


Figure 2.4: *Illustration of the proposed Supervised Deep Autoencoder (c) (SDAE) initialized with weight matrices from autoencoders (ASR (a) and TRS (b)).*

include appropriately pre-trained weight matrices to reach optimal performance. For this purpose, the SDAE weight matrices $W^{(\text{ASR})}$ and $W^{(\text{TRS})}$ are initialized with straightforward autoencoders as shown in Figures 2.4-(a) and 2.4-(b).

Learning process: A shallow autoencoder (dashed square in Figure 2.4 (c)) is then learnt to define the non-linear transformation from the corrupted or noisy hidden space $h^{(\text{ASR})}$ (green) to the clean one $h^{(\text{TRS})}$ (blue) through a bottleneck hidden layer h that represents an encoding of the corrupted input $x^{(\text{ASR})}$. The total reconstruction error L_{MSE} defined in eq. 2.6 is computed with an error l_{MSE} evaluated between the output vector $\tilde{x}^{(\text{TRS})}$ and the "clean" corresponding TRS vector $x^{(\text{TRS})}$:

$$l_{\text{MSE}}(x^{(\text{TRS})}, \tilde{x}^{(\text{TRS})}) = ||x^{(\text{TRS})} - \tilde{x}^{(\text{TRS})}||^2 \tag{2.8}$$

The Heterogeneous bottleneck features extraction process for a given noisy document $x^{(\text{ASR})}$ from the architecture presented in Figure 2.4-(c) with an encoding and a decoding phase is described as follows. **Encoding phase:** an input ASR vector $x^{(\text{ASR})}$ is encoded to obtain a vector h in the bottleneck hidden layer through the hidden space $h^{(\text{ASR})}$; **Decoding phase:** the vector h is then mapped in the "clean" latent space to obtain the vector $h^{(\text{TRS})}$ to reconstruct $\tilde{x}^{(\text{TRS})}$ close to the clean vector representation of the document manually transcribed $x^{(\text{TRS})}$. The Supervised Deep Autoencoder (SDAE)

is not fine-tuned to preserve both transformations by keeping unchanged $W^{(\text{ASR})}$ and $W^{(\text{TRS})}$. This is consistent with the initial intuition that reconstruction error between the observed output $\tilde{x}^{(\text{TRS})}$ and the desired one $x^{(\text{TRS})}$, should not modify the meaningful weight matrices $W^{(\text{ASR})}$ and $W^{(\text{TRS})}$ from the dedicated Autoencoders. For the sake of comparison and to evaluate the relevance of this assumption an equivalent fine-tuned supervised autoencoder (FSDAE) which learnt $W^{(1)}$, $W^{(2)}$, $W^{(\text{ASR})}$ and $W^{(\text{TRS})}$ during the last training step, is proposed.

Task-specific Denoising Autoencoder
The second new architecture proposed has the purpose of obtaining a denoised representation of a document which is best suited for the classification task. As TRS and ASR embeddings are obtained with a theme classifier, these components are estimated with the train data of the application domain corpus. This representation

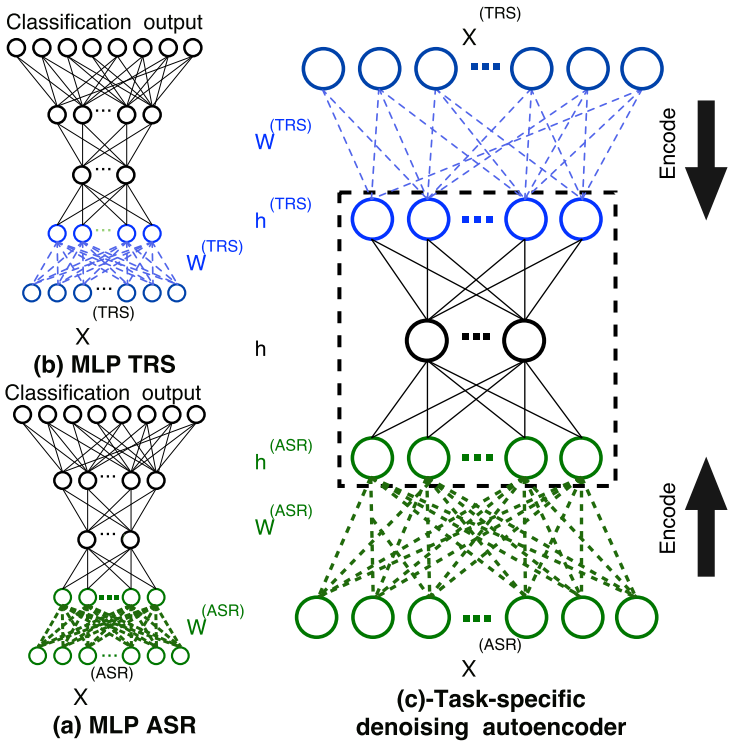


Figure 2.5: *Illustration of the proposed Task-specific Denoising Autoencoder (TDAE) with bottleneck features from MLP trained on ASR (a) and TRS (b).*

is close to the SDAE using both features from manual documents and features from noisy documents. This structure of the proposed architecture is shown in the Figure 2.5-(c). Its input is a noisy representation and its output is a clean one for the same document. Instead of relying on the denoising capacity of two DAEs, this network uses latent task meaningful features estimated by training the task classification. The architecture in Figure 2.5-(c) uses as input both the corrupted transcriptions (ASR) and the clean manual transcriptions (TRS).
First learning step: The TDAE is based on task specific features learnt with supervi-

sion. These features are computed by two MLPs one trained in noisy condition and one trained in clean condition. Bottleneck features are extracted from both MLPs.
Second learning step: During this step, the first and last layers are locked using the representation extracted from the two previously trained MLPs. A shallow autoencoder is then trained to obtain the non-linear transformation from the corrupted supervised hidden space to the clean supervised hidden space through a hidden layer that represents the robust encoding of the corrupted input x^{ASR}. The total reconstruction error L_{MSE} defined in equation (2.6) is evaluated between the shallow AE output $h^{\tilde{TRS}}$ and the MLP encoded version of the clean document h^{TRS}.

2.3 Experiments

The effectiveness of the semantic content encoded in the proposed bottleneck features is evaluated for the theme identification task of the DECODA corpus (Bechet et al., 2012). Theme identification is casted as a classification task performed by an MLP. The MLP has one hidden layer with 256 neurons. The parameters are estimated by Adam gradient descent (Kingma et Ba, 2014) with an initial learning rate of 10^{-3}. The number of iterations is determined by early stopping on a development corpus.

2.3.1 DECODA framework

The DECODA corpus, labeled by human annotators and made available as described in (Bechet et al., 2012), is a set of human-human telephone conversations between agents and customers of the customer care service (CCS) of the Paris transportation system RATP. This corpus is composed of 1,242 telephone conversations, corresponding to about 74 hours of signal, split as described in Table 2.1. Conversations have been manually transcribed and labelled with one theme corresponding to the principal concern mentioned by the customer. The train set is used to compose the subset of discriminative wordswith the TF-IDF-Gini method. A "stop list" of 126 words[1] is removed since they are considered as unnecessary words for the task. The train corpus contains 7,920 words, while the test corpus contains 3,806 words, only 70.8% of which occurring in the train corpus. This is due to the fact that users may mix the expression of their concerns with application domain irrelevant facts while the agents follow a protocol using words expressing domain facts. Following the protocol, agents tend to influence the users to mention, among other things, application domain related facts. This suggests focusing of words that express domain semantic facts for estimating probability distributions of pertinent conversation features. Relevant word selection using the train set is discussed in (Morchid et al., 2014b) showing that there is no advantage in theme identification with manual transcriptions of the development set if more than 800 words are used for the DECODA corpus. Based on the above findings, a set of 100 theme specific words is selected for the experiments described in the following. These words are then

[1] http://code.google.com/p/stop-words/

merged to form a vocabulary of 707 words with which the classifier input features are obtained. For each word, an input TF-IDF-Gini features is computed. Hidden topic LDA features, computed starting with probabilities of 800 words, have been used in (Morchid et al., 2014b). As large variations in classification accuracies were observed for different topic space dimensions in the development set, the topic space for which the best classification results were observed with the development set was also used for the test set. It was also observed that the optimal hidden space for the development set was not optimal for the test set. This suggested considering in (Morchid et al., 2015a) a large number (300) of hidden LDA spaces and integrate the features obtained with all these spaces into a single *c*-vector of reduced dimension. Results obtained with the development set showed that only 116 theme specific words should be considered to obtain the best results. In (Esteve et al., 2015) the same set of 116 words was used with the same ASR system together with 58 semantic features automatically extracted to express domain relevant facts. These features were used for classification with a simple MLP. For each theme, a set of 100 specific words is then selected. These words are then merged to form a vocabulary of 707 words with which the classifier input features are obtained.

Table 2.1: *Composition of the DECODA corpus.*

Class label	*Number of samples*		
	training	**dev.**	**text**
problems of itinerary	145	44	67
lost and found	143	33	63
time schedules	47	7	18
transportation cards	106	24	47
state of the traffic	202	45	90
fares	19	9	11
infractions	47	4	18
special offers	31	9	13
Total	**740**	**175**	**327**

2.3.2 Automatic Speech Recognition System

Experiments are performed with a large vocabulary continuous speech recognition (LVCSR) system with 230,000 Gaussians used in triphone acoustic models. Model parameters are estimated with maximum a-posteriori probability (MAP) adaptation from 150 hours of speech in telephone condition. The vocabulary contains 5,782 words. A 3-gram language model (LM) was obtained by adapting a basic LM with the transcriptions of the DECODA train set. The ASR word error rate (WER) is 33.8% on the train, 45.2% on the development, and 49.5% on the test subset of the corpus. These high error rates are mainly due to speech disfluencies and to adverse acoustic environments for some dialogues (calls from train stations, noisy and crowded streets with mobile phones...). The same corpus was recently processed with a novel ASR system

(Rousseau et al., 2014) with advanced neural components that performed very well in international competitions. WER of 33.8% for the development and 34,5% for the test sets were obtained, still showing rather frequent errors due to above mentioned unpredictable real-life environments and situations.Nevertheless, the impact on the selected topic specific words does not seem to prevent obtaining classification results close with the ones observed with manual transcriptions. In fact, as reported in (Morchid et al., 2014b), a maximum 89.7% accuracy was obtained with the same word set and ASR system as in this chapter. Such an accuracy is close to the reported best 92.2% accuracy obtained with manual transcriptions. Even if the optimal choice on the development set does not lead to the best accuracy with the test set, it is evident that the features used in this chapter are adequate to evaluate and compare the effects of the different feature denoising methods considered in this chapter. The manual transcriptions (TRS) are also used to show the maximum classification performance that could be achieved (*i.e.* best condition if no transcription error has been made by an ASR system). It is worth noticing that a similar WER has been reported on a similar corpus type (conversations from call-center) (Garnier-Rizet et al., 2008).

2.4 Results and Discussion

Table 2.2 compares the best accuracies of the different architectures and features proposed in this chapter. Although AE_{ASR} and SAE are simpler compared to others and are trained in a completely unsupervised way, they are among the more robust evaluated approaches. This can be explained by the fact that these neural networks are both able to remove an important portion of the erratic noise contained in the automatically transcribed documents. However, both methods cannot achieve the performance obtained with the original clean corpus. This means that there is still another kind of noise introduced by the automatic transcription step that cannot be compensated with noisy examples only. The smaller relative improvement obtained by an AE on clean data (0.83%) compared to the improvement on the data from ASR (3.76%), as shown in Table 2.2, pointed out that there is multiple factors for the noise present in ASR data and one of them is the automatic transcription process. The accuracy of the SDAE approaching 83.2% suggests the following observations. Table 2.2 shows that SDAE accuracy is just 0.9 point under the accuracy obtained with clean documents (TRS). This result shows that reconstructed feature vectors are still perturbed by a residual noise resulting in a decrease of classification performance with respect to using TRS features in AE_{TRS}.
The second is that introducing a form of supervision with manual transcriptions in the denoising process improves the quality of the learnt representation and lets the model remove a large portion of the type of noise that is the only one affecting manual transcriptions. Thirdly, the relatively poor results reported in Table 2.2) for DDAE, DAE, FSDAE show that it is difficult to remove general- and task-dependent ASR noise at the same time. In the proposed SDAE, the first and last layers capitalize on the capacity of AE_{ASR} and AE_{TRS} to remove residual noise by reducing the more complex or abstract form of noise thanks to a cleaner latent representation. The fixed matrix during the

second learning phase forces the network to denoise only the latent representations.
Overall, the best results are obtained with TDAE. With this solution, most of the noise affecting manual and ASR features is removed from the latent representations. In this way, the accuracy obtained with the ASR features equals the one achieved with TRS features, *i.e.* 84.1%. This is due to a pre-training supervision that makes it possible to remove most of the noise harmful for the classification task.
The latent representations from different environments let both TDAE and SDAE being able to construct a robust mapping representation for two reasons. First, the latent information carried by a document is related to the semantics of this document. Both clean and noisy versions of documents should share the same semantic independently of the actual words used. Secondly, the latent representation compresses the documents and force them to keep only the relevant information. In these latent spaces, noisy and clean versions of the same document have more in common than their visible form which makes them easier to remove the noise. The improvement obtained by the these networks compared to the very similar networks (in an unsupervised way) SAE and MLP-AE respectively shows that the supervision brought by the manual transcription is favorable for the posterior denoising process. We plan to continue the study of robust

Table 2.2: *Best theme classification accuracy (%) observed for each set of features from ASR.*

Method employed	**Feature vector**	**Test Accuracy**
MLP bottleneck	h^2	71.3
DDAE	$h^{(1)}$	72.5
DAE	h	74.3
FSDAE	h	76.5
TF.IDF	–	77.1
AE_{ASR}	h	81
MLP-AE	h^2	81.3
SAE	$h^{(2)}$	82.0
Proposed **SDAE**	h	83.2
Proposed **TDAE**	h^2	**84.1**

features from neural networks of spoken documents with novel generative adversarial networks (GAN), and a first investigation is reported in chapter 6. GAN models will be a main part of my future direction due to their capability to learn complex mapping from random inputs and require few supervision.

Part II

Quaternion Neural Networks for Natural Language Processing

Chapter 3

Quaternion Convolutional Neural Networks

Contents

Abstract

This chapter presents a novel model based on convolutional neural networks and the Quaternion hyper-complex numbers called "Quaternion Convolutional Neural Network" (QCNN). The proposed QCNN is evaluated on NLP related tasks (Theme identification task of spoken dialogues). This work has been realized during the Ph.D. thesis of Titouan Parcollet co-supervised with Georges Linarès (LIA) while Titouan Parcollet was an intern at the MILA (Canada) under the co-supervision of Yoshua Bengio.

3.1 Introduction

The previous chapters have presented the first research directions on Machine Learning (ML) models for NLP related applications. However, ML models, such as neural networks, employ document or signal representations based on basic low level features. Therefore, these basic representations reveal little in way of document statistical structure by only considering these features as a "bag-of-words", ignoring relations between them; moreover, the number of parameters are often large, even huge. During the last 5 years, we have proposed to remedying this weaknesses by extending the complex features based on Quaternion algebra presented in (Morchid et al., 2013) to neural networks. The effectiveness of such as algebra to represent multidimensional objects have been studied for speech recognition (Parcollet et al., 2018c,d,a, 2019; Titouan et al., 2019b), spoken language understanding (Parcollet et al., 2017; Titouan et al., 2017; Parcollet et al., 2016) and image processing (Parcollet et al., 2019); moreover, different quaternion based neural networks have been studied such as multi-layer perceptron (Parcollet et al., 2016, 2017), encoder-decoders (Titouan et al., 2017), recurrent neural networks (Parcollet et al., 2018c, 2019) and convolutional neural networks called QCNN (Parcollet et al., 2018d, 2019). One can refer to these papers for further details. The two newt chapters focus on the studies evaluating the effectiveness and efficiency of QCNN and QRNN architectures. QCNN are compared in this chapter to the real-valued CNN in a speech recognition (TIMIT) task. CNNs employ document representations based on features basic level. Therefore, these basic representations reveal little in way of document statistical structure by only considering signal, words or topics contained in the document ignoring relations between them. We propose in this chapter, to remedy this weakness by extending the complex features based on Quaternion algebra presented in (Morchid et al., 2013) to neural networks called QCNN. This original QCNN approach is based on hyper-complex algebra to take into account features dependencies in documents. Moreover, such models rely on unidimensional representations of the input information based on real numbers. Many realistic tasks require an adapted representation to fit the multidimensionality of the input features, such as pixels of an image, acoustic features, 3D models, or the different speech turns in a conversation. Therefore, traditional NNs process each component independently while a more natural way is to process each group of components as a single entity to learn both internal and contextual dependencies. Indeed, it is known that human-human conversations about specific items contain contextual relations between mentions of different speakers. In order to capture a part of these relations, it has been proposed to model a conversation with hyper-complex numbers (Parcollet et al., 2016; Morchid et al., 2013), the Quaternions, that integrate specific features for each speaker.
Quaternions are hypercomplex numbers that contain a real and three separate imaginary components, fitting perfectly to 3 and 4 dimensional input feature vectors, such as for image processing and robot kinematics (Sangwine, 1996; Pei et Cheng, 1999; Aspragathos et Dimitros, 1998). The idea of bundling groups of numbers into separate entities is also exploited by the recent capsule network (Sabour et al., 2017). Conversely to traditional homogeneous representations, capsule and quaternion networks bundle sets of features together. Thereby, quaternion neural network based models are able to code

latent inter-dependencies between groups of input features during the learning process with less parameters than traditional NNs, by taking advantage of the *Hamilton product* as the equivalent of the ordinary product, but between quaternions. Quaternion neural networks (Isokawa et al., 2003; Arena et al., 1994, 1997) have been proposed to solve different tasks that involve composed entities as input features (Arena et al., 1994, 1997). More precisely, good results have been obtained in the past for theme identification of telephone conversations (Parcollet et al., 2016) using a quaternion-based multilayer perceptron (QMLP) with adapted features for each speaker. However, the QMLP used as a solution to this task does not take into consideration the external and contextual informations that can exist between different turns of a dialogue.

3.2 Motivations

A major challenge of current machine learning models is to obtain efficient representations of relevant information for solving a specific task. Consequently, a good model has to efficiently code both the relations that occur at the feature level, such as between the Mel filter energies, the first, and second order derivatives values of a single time-frame, and at a global level, such as a phonemes or words described by a group of time-frames. Moreover, to avoid overfitting, better generalize, and to be more efficient, such models also have to be as small as possible. Nonetheless, real-valued neural networks usually require a huge set of parameters to well-perform on speech recognition tasks, and hardly code internal dependencies within the features, since they are considered at the same level as global dependencies during the learning. In the following, we detail the motivations to employ quaternion-valued neural networks instead of real-valued ones to code inter and intra features dependencies with less parameters.

First, a better representation of multidimensional data has to be explored to naturally capture internal relations within the input features. For example, an efficient way to represent the information composing an acoustic signal sequence is to consider each time-frame as being a whole entity of three strongly related elements, instead of a group of unidimensional elements that *could* be related to each others, as in traditional real-valued neural networks. Indeed, with a real-valued NN, the latent relations between the Mel filter banks energies, and the first and second order derivatives of a given time-frame are hardly coded in the latent space since the weight has to find out these relations among all the time-frames composing the sequence. Quaternions are fourth dimensional entities and allow one to build and process elements made of up to four elements, mitigating the above described problem. Indeed, the quaternion algebra and more precisely the *Hamilton product* allows quaternion neural network to capture these internal latent relations within the features of a quaternion. It has been shown that QNNs are able to restore the spatial relations within 3D coordinates (Matsui et al., 2004), and within color pixels (Isokawa et al., 2003), while real-valued NNs failed. In fact, the quaternion-weight components are shared through multiple quaternion input parts during the *Hamilton product* , creating relations within the elements. Indeed,

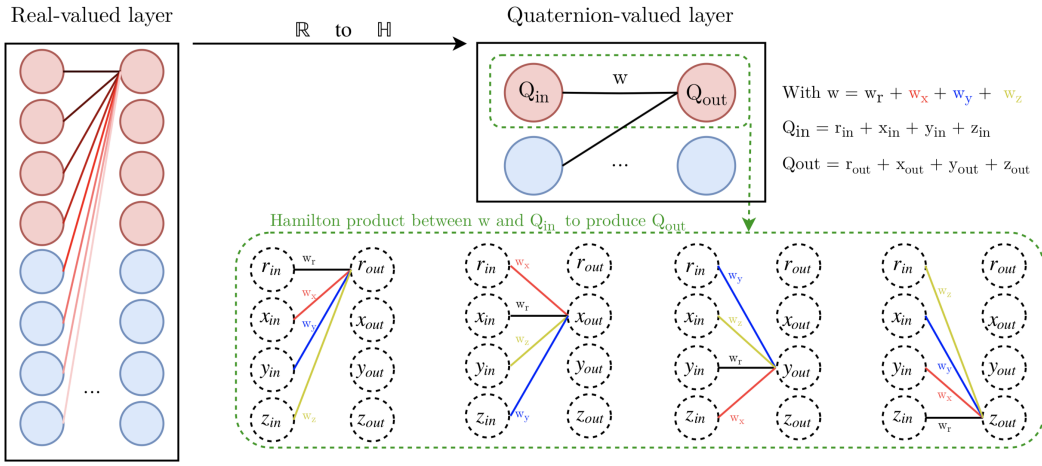


Figure 3.1: *Illustration of the input features (Q_{in}) latent relations learning ability of a quaternion-valued layer (right) due to the quaternion weight sharing of the* Hamilton product *(Eq. 3.5), compared to a standard real-valued layer (left).*

Figure 3.1 shows that the multiple weights required to code latent relations within a feature are considered at the same level as for learning global relations between different features, while the quaternion weight w codes these internal relations within a unique quaternion Q_{out} during the *Hamilton product* (right).

Second, quaternion neural networks make it possible to deal with the same signal dimension than real-valued NN, but with four times less neural parameters. Indeed, a 4-number quaternion weight linking two 4-number quaternion units only has 4 degrees of freedom, whereas a standard neural net parametrization have $4 \times 4 = 16$, i.e., a 4-fold saving in memory. Therefore, the natural multidimensional representation of quaternions alongside with their ability to drastically reduce the number of parameters indicate that hyper-complex numbers are a better fit than real numbers to create more efficient models in multidimensional spaces such as speech recognition. Indeed, modern automatic speech recognition systems usually employ input sequences composed of multidimensional acoustic features, such as log Mel features, that are often enriched with their first, second and third time derivatives (Davis et Mermelstein, 1990; Furui, 1986), to integrate contextual information. In standard NNs, static features are simply concatenated with their derivatives to form a large input vector, without effectively considering that signal derivatives represent different views of the same input. Nonetheless, it is crucial to consider that these three descriptors represent a special state of a time-frame, and are thus correlated. Following the above motivations and the results observed on previous works about quaternion neural networks, we hypothesize that for acoustic data, quaternion NNs naturally provide a more suitable representation of the input sequence, since these multiple views can be directly embedded in the multiple dimensions space of the quaternion, leading to smaller and more accurate models.

3.3 Quaternion algebra

The quaternion algebra **H** defines operations between quaternion numbers. A quaternion Q is an extension of a complex number to the hyper-complex plane defined in a four dimensional space as:

$$Q = r1 + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}, \tag{3.1}$$

where r, x, y, and z are real numbers, and 1, **i**, **j**, and **k** are the quaternion unit basis. In a quaternion, r is the real part, while $x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ with $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$ is the imaginary part, or the vector part. Such a definition can be used to describe spatial rotations. A quaternion Q can also be summarized into the following matrix of real numbers, that turns out to be more suitable for computations:

$$Q_{mat} = \begin{bmatrix} r & -x & -y & -z \\ x & r & -z & y \\ y & z & r & -x \\ z & -y & x & r \end{bmatrix}. \tag{3.2}$$

The conjugate Q^* of Q is defined as:

$$Q^* = r1 - x\mathbf{i} - y\mathbf{j} - z\mathbf{k}. \tag{3.3}$$

Then, a normalized or unit quaternion $Q^\triangleleft$ is expressed as:

$$Q^\triangleleft = \frac{Q}{\sqrt{r^2 + x^2 + y^2 + z^2}}. \tag{3.4}$$

Finally, the Hamilton product $\otimes$ between two quaternions Q_1 and Q_2 is computed as follows:

$$\begin{aligned} Q_1 \otimes Q_2 =& (r_1 r_2 - x_1 x_2 - y_1 y_2 - z_1 z_2) + \\ & (r_1 x_2 + x_1 r_2 + y_1 z_2 - z_1 y_2)\boldsymbol{i} + \\ & (r_1 y_2 - x_1 z_2 + y_1 r_2 + z_1 x_2)\boldsymbol{j} + \\ & (r_1 z_2 + x_1 y_2 - y_1 x_2 + z_1 r_2)\boldsymbol{k}. \end{aligned} \tag{3.5}$$

The Hamilton product is used in QCNNs to perform transformations of vectors representing quaternions, as well as scaling and interpolation between two rotations following a geodesic over a sphere in the $\mathbf{R}^3$ space as shown in (Minemoto et al., 2017).

3.4 Quaternion Convolutional Neural Networks

This section describes the quaternion features employed as well as the convolution process.

Quaternion internal representation. The QCNN is a quaternion extension of well-known real-valued and complex-valued convolutional neural networks (CNN) (He et al., 2016; Trabelsi et al., 2017). The quaternion algebra is ensured by manipulating matrices of real numbers. Consequently, a traditional $2D$ convolutional layer, with a kernel that contains N feature maps, is split into 4 parts: the first part equal to r, the second one to $x\mathbf{i}$, the third one to $y\mathbf{j}$ and the last one to $z\mathbf{k}$ of a quaternion $Q = r1 + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$. Nonetheless, an important condition to perform backpropagation in either real, complex or quaternion neural networks is to have cost and activation functions that are differentiable with respect to each part of the real, complex or quaternion number. Many activation functions for quaternion have been investigated (Xu et al., 2017) and a quaternion backpropagation algorithm have been proposed in (Nitta, 1995). Consequently, the split activation (Arena et al., 1994; Parcollet et al., 2016) function is applied to every layer and is defined as follows:

$$\alpha(Q) = \alpha(r) + \alpha(x)\mathbf{i} + \alpha(y)\mathbf{j} + \alpha(z)\mathbf{k}, \tag{3.6}$$

with α corresponding to any standard activation function.

Quaternion-valued convolution. Following a recent proposition for convolution of complex numbers(Trabelsi et al., 2017) and quaternions (Chase Gaudet, 2017), this chapter presents basic neural networks convolution operations using quaternion algebra. The convolution process is defined in the real-valued space by convolving a filter matrix with a vector. In a QCNN, the convolution of a quaternion filter matrix with a quaternion vector is performed. For this computation, the Hamilton product is computed using the real-valued matrices representation of quaternions. Let $W = R + X\mathbf{i} + Y\mathbf{j} + Z\mathbf{k}$ be a quaternion weight filter matrix, and $X_p = r + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ the quaternion input vector. The quaternion convolution w.r.t the Hamilton product $W \otimes X_p$ is defined as follows:

$$\begin{aligned} W \otimes X_p = &(Rr - Xx - Yy - Zz) + \\ &(Rx + Xr + Yz - Zy)\mathbf{i} + \\ &(Ry - Xz + Yr + Zx)\mathbf{j} + \\ &(Rz + Xy - Yx + Zr)\mathbf{k}, \end{aligned} \tag{3.7}$$

and can thus be expressed in a matrix form:

$$W \otimes X_p = \begin{bmatrix} R & -X & -Y & -Z \\ X & R & -Z & Y \\ Y & Z & R & -X \\ Z & -Y & X & R \end{bmatrix} * \begin{bmatrix} r \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r' \\ x'\mathbf{i} \\ y'\mathbf{j} \\ z'\mathbf{k} \end{bmatrix}, \tag{3.8}$$

An illustration of such operation is depicted in Figure 3.2.

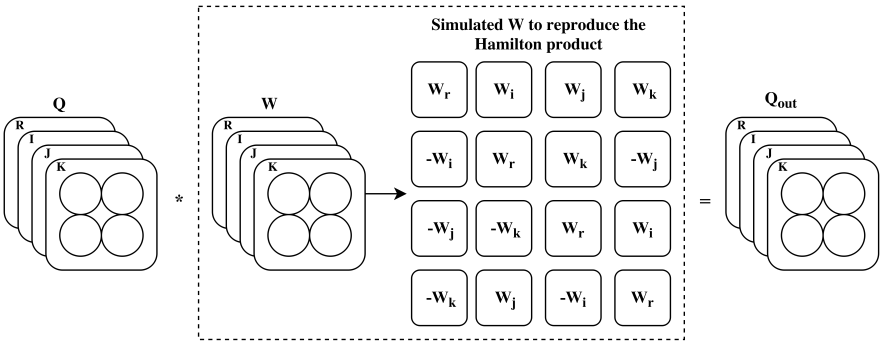


Figure 3.2: *Illustration of the quaternion convolution*

3.5 Experiments

The performance and efficiency of the QCNNs are evaluated on a phoneme recognition task on the TIMIT framework.

Data-set and acoustic features of quaternions. The TIMIT (Garofolo et al., 1993) dataset is composed of a standard 462-speaker training dataset, a 50-speakers development dataset and a core test dataset of 192 sentences. During the experiments, the SA records of the training set are removed and the development set is used for early stopping. The raw audio is transformed into 40-dimensional log mel-filter-bank coefficients with deltas, delta-deltas, and energy terms, resulting in a one dimensional vector of length 123. An acoustic quaternion $Q(f,t)$ associated with a frequency f and a time frame t is defined as follows:

$$Q(f,t) = 0 + e(f,t)\mathbf{i} + \frac{\partial e(f,t)}{\partial t}\mathbf{j} + \frac{\partial^2 e(f,t)}{\partial^2 t}\mathbf{k}. \tag{3.9}$$

It represents multiple views of a frequency f at time frame t, consisting of the energy $e(f,t)$ in the filter band corresponding to f, its first time derivative describing a slope view, and its second time derivative describing a concavity view. Finally, a unique quaternion is composed with the three corresponding energy terms. Thus, the quaternion input vector length is 41 ($\frac{123}{3}$).

Connectionist Temporal Classification (CTC). In the acoustic modeling part of ASR systems, the task of sequence-to-sequence mapping from an input acoustic signal $X = [x_1, ..., x_n]$ to a sequence of symbols $T = [t_1, ..., t_m]$ is complex due to:

- X and T could be in arbitrary length.
- The alignment between X and T is unknown in most cases.

Specially, T is usually shorter than X in terms of phoneme symbols.

To alleviate these problems, connectionist temporal classification (CTC) has been proposed (Graves et al., 2006). First, a softmax is applied at each timestep, or frame,

providing a probability of emitting each symbol X at that timestep. This probability results in a symbol sequences representation $P(O|X)$, with $O = [o_1, ..., o_n]$ in the latent space O. A blank symbol $'-'$ is introduced as an extra label to allow the classifier to deal with the unknown alignment. Then, O is transformed to the final output sequence with a many-to-one function $g(O)$ defined as follows:

$$\left.\begin{array}{l} g(z_1, z_2, -, z_3, -) \\ g(z_1, z_2, z_3, z_3, -) \\ g(z_1, -, z_2, z_3, z_3) \end{array}\right\} = (z_1, z_2, z_3). \tag{3.10}$$

Consequently, the output sequence is a summation over the probability of all possible alignments between X and T after applying the function $g(O)$. Accordingly to (Graves et al., 2006) the parameters of the models are learned, based on the cross entropy loss function. During the inference, a best path decoding algorithm is performed. Therefore, the latent sequence with the highest probability is obtained by performing argmax of the softmax output at each timestep. The final sequence is obtained by applying the function $g(.)$ to the latent sequence.
Weight initialization is crucial to efficiently train neural networks. An appropriate initialization improves training speed and reduces the risk of exploding or vanishing gradient. A quaternion initialization is composed of two steps. First, for each weight to be initialized, a purely imaginary quaternion q_{imag} is generated following an uniform distribution in the interval $[0, 1]$. The imaginary unit is then normalized to obtain $q^{\triangleleft}_{imag}$ following the quaternion normalization equation. The later is used alongside to other well known initializing criterion such as (Glorot et Bengio, 2010) or (He et al., 2015) to complete the initialization process of a given quaternion weight named w. More information on weight initialization are available in (Parcollet et al., 2018c).

3.6 Results and Discussion

Results on the phoneme recognition task on the TIMIT dataset are reported in Table 3.1. It is worth noticing the important difference in terms of the number of learning parameters between real and quaternion valued CNNs. It is easily explained by the quaternion algebra. In the case of a dense layer with $1,024$ input values and $1,024$ hidden units, a real-valued model will have $1,024^2 \approx 1$M parameters, while to maintain equal input and output nodes $(1,024)$ the quaternion equivalent has 256 quaternions inputs and 256 quaternion-valued hidden units. Therefore, the number of parameters for the quaternion model is $256^2 \times 4 \approx 0.26$M. Such a complexity reduction turns out to produce better results and may have other advantages such as a smallest memory footprint while saving NN models. Moreover, the reduction of the number of parameters does not result in poor performance in the QCNN. Indeed, the best PER reported is 19.64% from a QCNN with 256 feature maps and 10 layers, compared to a PER of 20.57% for a real-valued CNN with 64 feature maps and 10 layers. It is worth underlying that both model accuracies are increasing with the size and the depth of the neural network. However, bigger real-valued feature maps leads to overfitting. In fact, as shown in Table 3.1, the best PER for a real-valued model is reached with 64 (20.57) feature maps

and decreasing at 128 (20.62%) and 256 (21.23). The QCNN does not suffer from such weaknesses due to the smaller density of the neural network and achieved a constant PER improvement alongside with the increasing number of feature maps. Furthermore, QCNNs always performed better than CNNs independently of the model topologies.

Table 3.1: *Experiment results expressed in term of phoneme error rate (PER) percentage of both QCNN and CNN based models on the TIMIT phoneme recognition task. The results are from a* 3 *folds average. 'L' stands for number of Layers, 'FM' for number of feature maps, and 'Params' for number of learning parameters. The latter is expressed in order to be equivalent for both models. Therefore,* 32*FM is equal to* 32*FM for real numbers and* 8 *quaternion-valued FM*

Models	**Dev PER %**	**Test PER %**	**Params**
$\mathbb{R}$-CNN-6L-32FM	22.18	23.54	3.3M
$\mathbb{H}$-QCNN-6L-32FM	22.16	23.20	0.87M
$\mathbb{R}$-CNN-10L-32FM	21.77	23.43	3.4M
$\mathbb{H}$-QCNN-10L-32FM	22.25	23.23	0.9M
$\mathbb{R}$-CNN-6L-64FM	21.19	22.12	4.8M
$\mathbb{H}$-QCNN-6L-64FM	21.44	21.99	1.2M
$\mathbb{R}$-CNN-10L-64FM	19.53	**20,57**	5.4M
$\mathbb{H}$-QCNN-10L-64FM	19.78	20.44	1.4M
$\mathbb{R}$-CNN-6L-128FM	20.33	22.14	9M
$\mathbb{H}$-QCNN-6L-128FM	20.12	21.33	2.3M
$\mathbb{R}$-CNN-10L-128FM	19.37	20.62	11.5M
$\mathbb{H}$-QCNN-10L-128FM	19.02	19.87	2.9M
$\mathbb{R}$-CNN-6L-256FM	20.43	22.25	22.3M
$\mathbb{H}$-QCNN-6L-256FM	19.94	20.54	5.6M
$\mathbb{R}$-CNN-10L-256FM	18.89	21.23	32.1M
$\mathbb{H}$-QCNN-10L-256FM	18.33	**19.64**	8.1M

With much fewer learning parameters for a given architecture, the QCNN performs always better than the real-valued one on the reported task. In terms of PER, an average relative gain of 3.25% (w.r.t CNNs result) is obtained on the testing set. It is also worth recalling that the best PER of 19.64% is obtained with just a QCNN without HMMs, RNNs, attention mechanisms, batch normalization, phoneme language model, acoustic data normalization or adaptation. Further improvements can be obtained with exactly the same QCNN by just introducing a new acoustic feature in the real part of the quaternions.

Chapter 4

Quaternion Recurrent Neural Networks

Contents

Abstract

This chapter introduces the Quaternion Recurrent Neural Networks (QRNN) and an extension of the Long Short-Term Memory to Quaternion algebra called QLSTM. These models are evaluated during the speech recognition task of TIMIT. This work has been realized during the Ph.D. thesis of Titouan Parcollet co-supervised with Georges Linarès (LIA).

4.1 Introduction

The work presented in this chapter is another extension to quaternion algebra to a real-valued neural network. The Quaternion recurrent neural network and different extensions have been proposed during the Ph.D. thesis of Titouan Parcollet and presented at different international conferences (Parcollet et al., 2018b,a, 2019). In this chapter we limit ourselves to the QRNN/QLSTM (Parcollet et al., 2018b) and an comparaison to RNN during an application of speech recognition (TIMIT). This chapter proposes then to integrate local features in a novel model called quaternion recurrent neural network (QRNN) and its gated extension called quaternion long-short term memory neural network (QLSTM). The model learns both inter- and intra-dependencies between multidimensional input features and the basic elements of a sequence with drastically fewer parameters, making the approach more suitable for low-resource applications. The effectiveness of the proposed QRNN and QLSTM is evaluated on the realistic TIMIT phoneme recognition task that shows that both QRNN and QLSTM obtain better performances than RNNs and LSTMs with a best observed phoneme error rate (PER) of 18.5% and 15.1% for QRNN and QLSTM, compared to 19.0% and 15.3% for RNN and LSTM. Moreover, these results are obtained alongside with a reduction of 3.3 times of the number of free parameters. Similar results are observed with the larger Wall Street Journal (WSJ) dataset.

4.2 Quaternion Recurrent Neural Networks

The QRNN is an extension of the real-valued (Medsker et Jain, 2001) and complex-valued (Hu et Wang, 2012; Song et Yam, 1998) recurrent neural networks to hypercomplex numbers. The quaternion internal representation is the same that defined in section 3.4. The QRNN differs from the real-valued RNN in each learning sub-processes. Therefore, let x_t be the input vector at timestep t, h_t the hidden state, W_{hx}, W_{hy} and W_{hh} the input, output and hidden states weight matrices respectively. The vector b_h is the bias of the hidden state and p_t, y_t are the output and the expected target vectors. More details of the learning process and the parametrization are available on (Parcollet et al., 2018b).

4.3 Experiments

This Section details the acoustic features extraction and the experimental setups.

Quaternion acoustic features. The raw audio is first split every 10ms with a window of 25ms. Then 40-dimensional log Mel-filter-bank coefficients with first, second, and third order derivatives are extracted using the *pytorch-kaldi*[1] (Ravanelli

[1] pytorch-kaldi is available at https://github.com/mravanelli/pytorch-kaldi

et al., 2018b) toolkit and the Kaldi s5 recipes (Povey et al., 2011). An acoustic quaternion $Q(f,t)$ associated with a frequency f and a time-frame t is formed as follows:

$$Q(f,t) = e(f,t) + \frac{\partial e(f,t)}{\partial t}\mathbf{i} + \frac{\partial^2 e(f,t)}{\partial^2 t}\mathbf{j} + \frac{\partial^3 e(f,t)}{\partial^3 t}\mathbf{k}. \tag{4.1}$$

$Q(f,t)$ represents multiple views of a frequency f at time frame t, consisting of the energy $e(f,t)$ in the filter band at frequency f, its first time derivative describing a slope view, its second time derivative describing a concavity view and the third derivative describing the rate of change of the second derivative. Quaternions are used to learn the spatial relations that exist between the 3 described different views that characterize a same frequency (Tokuda et al., 2003). Thus, the quaternion input vector length is $160/4 = 40$. Decoding is based on Kaldi (Povey et al., 2011) and weighted finite state transducers (WFST) (Mohri et al., 2002) that integrate acoustic, lexicon and language model probabilities into a single HMM-based search graph.

The TIMIT corpus and neural networks configurations. The training process is based on the standard $3,696$ sentences uttered by 462 speakers, while testing is conducted on 192 sentences uttered by 24 speakers of the TIMIT (Garofolo et al., 1993) dataset. A validation set composed of 400 sentences uttered by 50 speakers is used for hyper-parameter tuning. The models are compared on a fixed number of layers $M = 4$ and by varying the number of neurons N from 256 to $2,048$, and 64 to 512 for the RNN and QRNN respectively. Indeed, it is worth underlining that the number of hidden neurons in the quaternion and real spaces do not handle the same amount of real-number values. Indeed, 256 quaternion neurons output are $256 \times 4 = 1,024$ real values. Tanh activations are used across all the layers except for the output layer that is based on a softmax function. Models are optimized with RMSPROP with vanilla hyper-parameters and an initial learning rate of $8 \cdot 10^{-4}$. The learning rate is progressively annealed using a halving factor of 0.5 that is applied when no performance improvement on the validation set is observed. The models are trained during 25 epochs. All the models converged to a minimum loss due to the annealed learning rate. A dropout rate of 0.2 is applied over all the hidden layers (Srivastava et al., 2014) except the output one. The negative log-likelihood loss function is used as an objective function. All the experiments are repeated 5 times (5-folds) with different seeds and are averaged to limit any variation due to the random initialization.

4.4 Results and Discussion

We discuss in this section the results obtained with QRNNs, QLSTMs, RNNs and LSTMs on the TIMIT speech recognition tasks. The results reported in bold on tables are obtained with the best configurations of the neural networks observed with the validation set.

Quaternion recurrent neural networks (QRNN). The results on the TIMIT task

Table 4.1: *Phoneme error rate (PER%) of QRNN and RNN models on the development and test sets of the TIMIT dataset. "Params" stands for the total number of trainable parameters.*

Models	Neurons	Dev.	Test	Params
RNN	256	22.4	23.4	1M
	512	19.6	20.4	2.8M
	1,024	**17.9**	**19.0**	**9.4M**
	2,048	20.0	20.7	33.4M
QRNN	64	23.6	23.9	0.6M
	128	19.2	20.1	1.4M
	256	**17.4**	**18.5**	**3.8M**
	512	17.5	18.7	11.2M

are reported in Table 4.1. The best PER in realistic conditions (w.r.t to the best validation PER) is 18.5% and 19.0% on the test set for QRNN and RNN models respectively, highlighting an absolute improvement of 0.5% obtained with QRNN. These results compare favourably with the best results obtained so far with architectures that do not integrate access control in multiple memory layers (Ravanelli et al., 2018a). In the latter, a PER of 18.3% is reported on the TIMIT test set with batch-normalized RNNs. Moreover, a remarkable advantage of QRNNs is a drastic reduction (with a factor of 2.5×) of the parameters needed to achieve these results. Indeed, such PERs are obtained with models that employ the same internal dimensionality corresponding to $1,024$ real-valued neurons and 256 quaternion-valued ones, resulting in a number of parameters of 3.8M for QRNN against the 9.4M used in the real-valued RNN. It is also worth noting that QRNNs consistently need fewer parameters than equivalently sized RNNs with an average reduction factor of 2.26 times. This is easily explained by considering the content of the quaternion algebra. Indeed, for a fully-connected layer with $2,048$ input values and $2,048$ hidden units, a real-valued RNN has $2,048^2 \approx 4.2$M parameters, while to maintain equal input and output dimensions the quaternion equivalent has 512 quaternions inputs and 512 quaternion hidden units. Therefore, the number of parameters for the quaternion-valued model is $512^2 \times 4 \approx 1$M. Such a complexity reduction turns out to produce better results and has other advantages such as a smaller memory footprint while saving models on budget memory systems. This characteristic makes our QRNN model particularly suitable for speech recognition conducted on low computational power devices like smartphones (Chen et al., 2014). QRNNs and RNNs accuracies vary accordingly to the architecture with better PER on bigger and wider topologies. Therefore, while good PER are observed with a higher number of parameters, smaller architectures performed at 23.9% and 23.4%, with 1M and 0.6M parameters for the RNN and the QRNN respectively. Such PER are due to a too small number of parameters to solve the task.

Quaternion long-short term memory (QLSTM). We propose to extend the QRNN to state-of-the-art models such as long-short term memory neural networks (LSTM), to support and improve the results already observed with the QRNN compared to the RNN in more realistic conditions. LSTM (Hochreiter et Schmidhuber, 1997) neural

Table 4.2: *Phoneme error rate (PER%) of QLSTM and LSTM models on the development and test sets of the TIMIT dataset. "Params" stands for the total number of trainable parameters.*

Models	Neurons	Dev.	Test	Params
LSTM	256	14.9	16.5	3.6M
	512	14.2	16.1	12.6M
	1,024	**14.4**	**15.3**	**46.2M**
	2,048	14.0	15.9	176.3M
QLSTM	64	15.5	17.0	1.6M
	128	14.1	16.0	4.6M
	256	**14.0**	**15.1**	**14.4M**
	512	14.2	15.1	49.9M

networks were introduced to solve the problems of long-term dependencies learning and vanishing or exploding gradient observed with long sequences. Based on the equations of the forward propagation and back propagation through time of QRNN described in Appendix (Parcollet et al., 2018b), one can easily derive the equations of a quaternion-valued LSTM. Gates are defined with quaternion numbers following the proposal of (Danihelka et al., 2016). Therefore, the gate action is characterized by an independent modification of each component of the quaternion-valued signal following a component-wise product with the quaternion-valued gate potential. Let f_t,i_t, o_t, c_t, and h_t be the forget, input, output gates, cell states and the hidden state of a LSTM cell at time-step t:

$$f_t = \alpha(W_f \otimes x_t + R_f \otimes h_{t-1} + b_f), \tag{4.2}$$
$$i_t = \alpha(W_i \otimes x_t + R_i \otimes h_{t-1} + b_i), \tag{4.3}$$
$$c_t = f_t \times c_{t-1} + i_t \times tanh(W_c \otimes x_t + R_c \otimes h_{t-1} + b_c), \tag{4.4}$$
$$o_t = \alpha(W_o \otimes x_t + R_o \otimes h_{t-1} + b_o), \tag{4.5}$$
$$h_t = o_t \times tanh(c_t), \tag{4.6}$$

where W are rectangular input weight matrices, R are square recurrent weight matrices and b are bias vectors. α is the split activation function and $\times$ denotes a component-wise product between two quaternions. Both QLSTM and LSTM are bidirectional and trained on the same conditions than for the QRNN and RNN experiments.

The results on the TIMIT corpus reported on Table 4.2 support the initial intuitions and the previously established trends. We first point out that the best PER observed is 15.1% and 15.3% on the test set for QLSTMs and LSTM models respectively with an absolute improvement of 0.2% obtained with QLSTM using 3.3 times fewer parameters compared to LSTM. These results are among the top of the line results (Graves et al., 2013b; Ravanelli et al., 2018a) and prove that the proposed quaternion approach can be used in state-of-the-art models. We have also evaluated both QLSTMs and LSTMs with a larger and more realistic corpus to validate the scaling of the observed TIMIT results (Section 6.4). Acoustic input features are described in Section

6.3, and extracted on both the 14 hours subset "train-si84" and the full 81 hours dataset "train-si284" of the Wall Street Journal (WSJ) corpus. The "test-dev93" development set is employed for validation while "test-eval92" composes the testing set. Models architectures are fixed with respect to the best results observed with the TIMIT corpus (Section 6.4). Therefore both QLSTMs and LSTMs contain four bidirectional layers of internal dimension of size $1,024$. Then an additional layer of internal size $1,024$ is added before the output layer. The only modification in the training process compared to the TIMIT experiments reported in Section 4.4, concerns the model optimizer which is set to Adam (Kingma et Ba, 2014) instead of RMSPROP. Results are from a 3-folds average.

Table 4.3: *Word error rates (WER %) for WSJ14h and WSJ81h. "test-dev93" and "test-eval92" are used as validation and testing sets respectively. L stands for the number of recurrent layers.*

Models	**WSJ14 Dev.**	**WSJ14 Test**	**WSJ81 Dev.**	**WSJ81 Test**	**Params**
LSTM	11.2	7.2	7.4	4.5	53.7M
QLSTM	10.9	6.9	7.2	4.3	18.7M

It is worth noticing that the results on Table 4.3 are promising (Graves et al., 2013a) (WER of 11.7% on "test-dev93") and competitive with state-of-the-art but more complex models based on better engineered features (Chan et Lane, 2015)(WER of 3.8% with the 81h on "test-eval92"). Table 4.3 shows that QLSTMs outperform LSTMs in all training conditions (14h or 81h) and with respect to both the validation and the testing sets. Moreover, QLSTMs need 2.9 times less parameters than LSTMs to achieve equivalent performances. These experiments demonstrate that QLSTMs scale well to larger and more realistic speech datasets and are still more efficient than real-valued LSTMs.

Part III

Ongoing Research, Future Directions & General Perspectives

Chapter 5

Parsimonious Neural Networks

Contents

Abstract

The previous chapters chart the path of my research throughout different neural networks based topics and language processing tasks. These models compose the first step of my research in the area of machine learning for NLP applications and require a large, even huge processing time during the learning process. In this chapter I give novel promising research perspectives for faster and more efficient neural networks based models. This work is an overview of part of my futur direction on terms of domain (deep learning) and interest (find out more efficient algorithms).

5.1 Introduction

My activities in the field of recurrent neural networks started with the Ph.D. thesis of Mohamed Bouaziz and compose an important part of my ongoing research and this chapter presents a parsimonious RNN suitable to process small corpuses with less processing time. The processing time required to learn the models with large amount of data is huge; moreover, these applications are inclined to be executed on the client side on mobile devices for example. Therefore, ML based systems have to be efficient enough in terms of processing time and obtained performances to be acceptable for mobile devices users. This research direction aims to propose novel neural network architectures and paradigms to better handle this large amount of uncontrolled datasets alongside to maintain even improve the performances. The chapter proposes a recurrent neural network architecture that manages short- and long-term dependencies differently than an Long Short-Term Memory (Sundermeyer et al., 2012; Greff et al., 2017) (LSTM) or Gated Recurrent Unit (GRU) (Cho et al., 2014) with a single gate called "Parsimonious Memory Unit Recurrent Neural Network" (PMU). Indeed, LSTM takes into account both short- and long-term dependencies to the word properly in different contexts, but needs to process the information in different memory blocks composed of a set of cell-related gates. Even if the results of LSTM-RNNs are promising, the processing time required to treat large documents data sets is quite huge due to the different gates activation sub-processes. A recent addition to the RNN set of models called the GRU has been proposed by (Wu et al., 2017b) to address this issue and has shown good performances in several tasks such as speech recognition (Graves et al., 2013b) or machine translation (Sutskever et al., 2014). GRU is very similar to LSTM, in that it uses a combination of gates to adjust exposure from input to the hidden states. It does however not use a memory cell but opting to fully expose its state to the output and doing away with the output gate. Switching between full retention, mixed and forget mode is implemented using a reset gate r and an update gate z. This is the main drawback of the GRU in that the role and the management of the gates are not based on the relation between short- and long-term dependencies. For example, the reset of the hidden state is provided by both update and reset gates of the GRU; moreover, the role of the reset gate is to avoid the hidden state candidate but the update gate z can remove this part of the hidden state information without the reset gate r. Therefore, all hidden states share the same distribution between short- and long-term dependencies. The PMU in which every cell is an efficient integrator with a single gate u that makes a strong assumption to reduce the processing time and better manage the latent relations between short- and long-term dependencies: short and long-term dependencies are related, and the more the RNN-based model learns from short dependencies, the less it learns from long ones.

5.2 Parsimonious Memory Unit (PMU)

The proposed PMU is related to RNN and more precisely to the Gated Recurrent Unit (GRU) introduced by (Cho et al., 2014). GRU is a LSTM-like adaptive "reset" and "up-

date" memory unit. More information about LSTM and GRU are available in the annexe. The LSTM input gate $i_j(t)$ is replaced in the GRU by an update gate $z_j(t)$, and the forget gate $f_j(t)$ by a reset gate $r_j(t)$ for the GRU. The main idea behind the GRU unit is that the GRU exposes the memory content at each time step and balances the previous and new memory content strictly using leaky integration. This is particularly shown during the update process of the weight matrices. For brevity and clarity we detail here only the update of the weight matrix $w_{h_j m}$ of the hidden state h_j for the state x_{hj} (more details are available in the appendix:

$$\begin{aligned}\Delta w_{h_j m} &= \alpha\, e_{h_j}(t)\, \frac{\partial h_j(t)}{\partial w_{h_j m}} \\ \frac{\partial h_j(t)}{\partial w_{h_j m}} &= \frac{\partial h_j(t-1)}{\partial w_{h_j m}} \left(1 - z_j(t)\right) \\ &\quad + z_j(t) r_j(t) \sigma'(x_{hj}(t)) h_m(t-1)\,. \end{aligned} \tag{5.1}$$

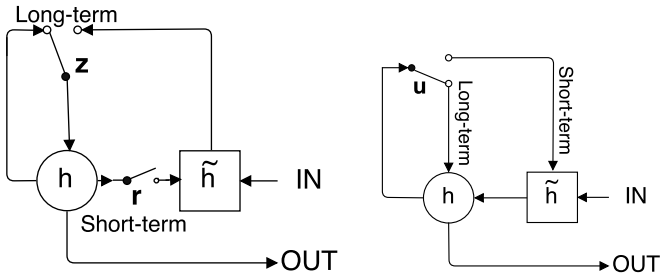
Short and long-term dependencies for the GRU

One can note that the update of the weight matrix depends on both short-term dependencies ($\frac{\partial h_j(t-1)}{\partial w_{h_j m}}$) controlled by the inverse of the update gate $(1 - z_j(t))$, and long-term dependencies ($\sigma'(x_{hj})\ h_m(t-1)$) balanced by the reset $r_j(t)$ and the update $z_j(t)$ gates (if active). The derivative of the hidden state has a restricted behaviour. Indeed, for example, to learn from only long-term dependencies, equation (5.1) shows that the conditions $z_j = 1$ and $r_j = 1$ have to be jointly satisfied. The GRU makes the assumption that long and short-term dependencies are not related and the short-term dependencies are controlled only by the reset gate $r_j(t)$. Indeed, the reset gate $r_j(t)$ controls directly the long-term dependencies and $(1 - z_j(t))$ controls also the long-term dependencies throughout a non-linear transformation. Thus, the weight matrix update does not reflect this balance between $r_j(t)$ and $(1 - z_j(t))$. This model is a variant of the leaky-integration unit (LIU) proposed by (Bengio et al., 2013b) and, in the case of $r_j(t) = 1\ \ \forall t \in T$, the GRU is a straightforward LIU. Therefore, the reset gate $r_j(t)$ and $(1 - z_j(t))$ play the same role since short and long-term dependencies are correlated: the more the memory unit has to learn from short-term dependencies, the less the long-term dependencies are required to evaluate a relevant current hidden state. Based on this assumption, we have:

$$z_j(t) \simeq r_j(t) \tag{5.2}$$

$$\begin{aligned}\frac{\partial h_j(t)}{\partial w_{h_j m}} &= \frac{\partial h_j(t-1)}{\partial w_{h_j m}} \left(1 - z_j(t)\right) \\ &\quad + z_j(t)^2 \sigma'(x_{hj}(t)) h_m(t-1)\,. \end{aligned} \tag{5.3}$$

Equation (5.3) allows the GRU to maintain a balance between short (r_j in σ') and long (z_j and $(1 - z_j)$) term dependencies. We propose a novel memory unit called "Parsimonious Memory Unit" (PMU) that addresses these three main issues of the GRU alongside reducing the processing time (Parsimonious):



(a) GRU (b) PMU

Figure 5.1: *The graphical illustration of the (a) GRU and the proposed (b) PMU. The u represents the self-balanced PMU gate and h and $\tilde{h}$ are the activation and the new activation state for the PMU. Long and short-term dependencies are also represented in the graphics with a single gate u for the PMU and both r and z gates for the GRU.*

- controlling uniformly for forward and update phases, the short-term dependencies by evaluating these term dependencies with a same gate (r_j for previous hidden state and $z_j(t)$ for the hidden state candidate),
- taking into consideration the relation between short- and long-term dependencies ($z_j(t) \simeq r_j(t)$),
- managing short and long-term dependencies with dedicated hidden neurons.

The proposed PMU depicted in Figure 5.1-(b) is based on the assumption that the more the memory unit learns from short-term dependencies r_j and the less it needs to learn from long-term dependencies z_j. Let us describe how the activation of the j-th hidden unit h_j of the PMU is computed. First, the PMU gate u_j is computed as follows:

$$\begin{aligned} u_j(t) &= \sigma(x_{u_j}(t)) \\ x_{u_j}(t) &= \sum_m s_{u_j m} v_m(t) + w_{u_j m} h_m(t-1) \ . \end{aligned}$$

The activation of the proposed unit h_j is defined as:

$$h_j(t) = (1 - u_j(t)) h_j(t-1) + \tilde{h}_j(t) \tag{5.4}$$

$$\begin{aligned} \tilde{h}_j(t) &= \tanh(x_{h_j}) \\ x_{h_j} &= \sum_m s_{h_j m} v_m(t) + w_{h_j m} h_m(t-1) u_j(t) \ . \end{aligned} \tag{5.5}$$

Finally, the output unit y_k is computed by:

$$\begin{aligned} y_k(t) &= \tanh(x_k(t)) \\ x_k(t) &= \sum_m w_{km} h_m(t-1) \ . \end{aligned}$$

The hidden state matrix $w_{h_j m}$ is update as follows:

$$\begin{aligned}\Delta w_{h_j m} &= \alpha\, e_{h_j}(t)\, \frac{\partial h_j(t)}{\partial w_{h_j m}} \\ \frac{\partial h_j(t)}{\partial w_{h_j m}} &= \frac{\partial h_j(t-1)}{\partial w_{h_j m}} \left(1 - u_j(t)\right) \\ &\quad + \tanh'(x_{hj}(t)) u_j(t) h_m(t-1)\,. \end{aligned} \tag{5.6}$$

More details about PMU are available in (Morchid, 2018).

Short and long-term dependencies for the PMU
Equations (5.4), (5.5) and (5.6) are homogeneous with regards to the representation of short and long-term dependencies and the update process described in equation (5.6) of the weight matrix of the hidden state is detailed thereafter:

$$u_j(t) \to 1 \Rightarrow \frac{\partial h_j(t)}{\partial w_{h_j m}} \to h_m(t-1)\sigma'(x_{hj}) \quad \text{(long-term)} \tag{5.7}$$

$$u_j(t) \to 0 \Rightarrow \frac{\partial h_j(t)}{\partial w_{h_j m}} \to \frac{\partial h_j(t-1)}{\partial w_{h_j m}} \quad \text{(short-term)} \tag{5.8}$$

In a same manner, the hidden state of the proposed PMU described in equations (5.4), (5.5), behaves during the forward pass as follows:

$$u_j(t) \to 1 \Rightarrow h_j(t) \to \tilde{h}_j(t) \quad \text{(long-term)} \tag{5.9}$$

$$u_j(t) \to 0 \Rightarrow h_j(t) \to h_j(t-1) + \beta_j \quad \text{(short-term)} \tag{5.10}$$

Table 5.1 presents how the short or long-term dependencies are considered with respect to the u gate of the PMU ($\beta_j = \sigma(\sum_m s_{h_j m} v_m(t))$). All intermediate-term dependencies ($1 < u_j < 0$) give a balance between short and long-term memory.

Table 5.1: *Hidden state* **derivative** *behaviour for exclusive short or long-term dependencies for the* **PMU**.

Dependencies	**Pass**	u_j	**Value**
Long-term (5.4)-(5.5)	Forward	1	$\tilde{h}_j(t)$
Short-term (5.4)-(5.5)	Forward	0	$h_j(t-1) + \beta_j$
Long-term (5.6)	Backward	1	$\sigma'(x_{hj}) h_m(t-1)$
Short-term (5.6)	Backward	0	$\frac{\partial h_j(t-1)}{\partial w_{h_j m}}$

The difference between the classical GRU and the proposed PMU is observed when short-term dependencies are required for the hidden state during the forward pass (second column in these tables) with an additional value (β_j). This value is not related to the hidden state and is evaluated with the input vector ($v_m(t)$) and the input-to-hidden matrix ($s_{h_j m}$). Therefore, the short-term dependencies are not affected by β_j.

The PMU requires only one gate viewed as a balance between short and long term dependencies. Moreover, the PMU manages short and long-term dependencies with dedicated neurons.

5.3 Experiments

Data-sets and metrics. The DECODA corpus as well as on the Automatic Speech Recognition (ASR) system employed are available in sections 2.3.1 and 2.3.2. The categorization task of the 20-Newsgroups (Xu et al., 2013) dataset is employed to exhibit long-term dependencies. This corpus is described in the official website[1]. The same metric is employed to make the comparison between the results obtained by the proposed PMU-RNN model in "long-term" (20Newsgroups) and "short-term" (DECODA) dependencies easier. To evaluate the effectiveness of the proposed method, the authors in (Albishre et al., 2015; Srivastava et al., 2013; Bouallegue et al., 2014; Morchid et al., 2014a) used the accuracy. More information are available in (Van Asch, 2013).
Recurrent Neural Networks Setup. The RNN, LSTM, GRU and the proposed PMU are composed of 3 layers: input layer s of the size of the discriminative vocabulary (Morchid et al., 2014c) (50 words for each class gives 166 words for DECODA and 908 for the 20-News group data set), a hidden layer h of size varying from 10 to 300 for all RNN and an output layer y_k with a size equal to the number of classes (8 for DECODA and 20 for the 20-Newsgroups respectively).

5.4 Results and Discussion

The short and long-term dependencies are measured in the gates activity (update and reset gates for GRU and the gate of PMU) and depicted in Section 5.4.1. Section 5.4.2 and Section 5.4.3 compare RNN, LSTM, GRU and the proposed PMU in terms of accuracy and time processing.

5.4.1 Gates activity of GRU and PMU

This section presents both GRU and PMU gates activity to demonstrate that the role of GRU's gates is similar for all hidden neurons while the PMU gates behave differently across its hidden neurons. The optimal point corresponds to the number of hidden neurons required to reach the best performances for short-term (Table 5.2) and long-term (Table 5.3) dependencies.

Short-term gates activity from a small corpus of spoken dialogues
Figure 5.2 a) presents the gates activity of a GRU-RNN with 8 hidden states [2] in the

[1] http://qwone.com/~jason/20Newsgroups/

[2] The term "hidden state" is employed even if the GRU and PMU have a hidden state rather than a hidden state as the LSTM.

hidden layer. The dashed curves represent the activity of the update gate z and the plain black curves the reset gate r of the GRU-RNN.

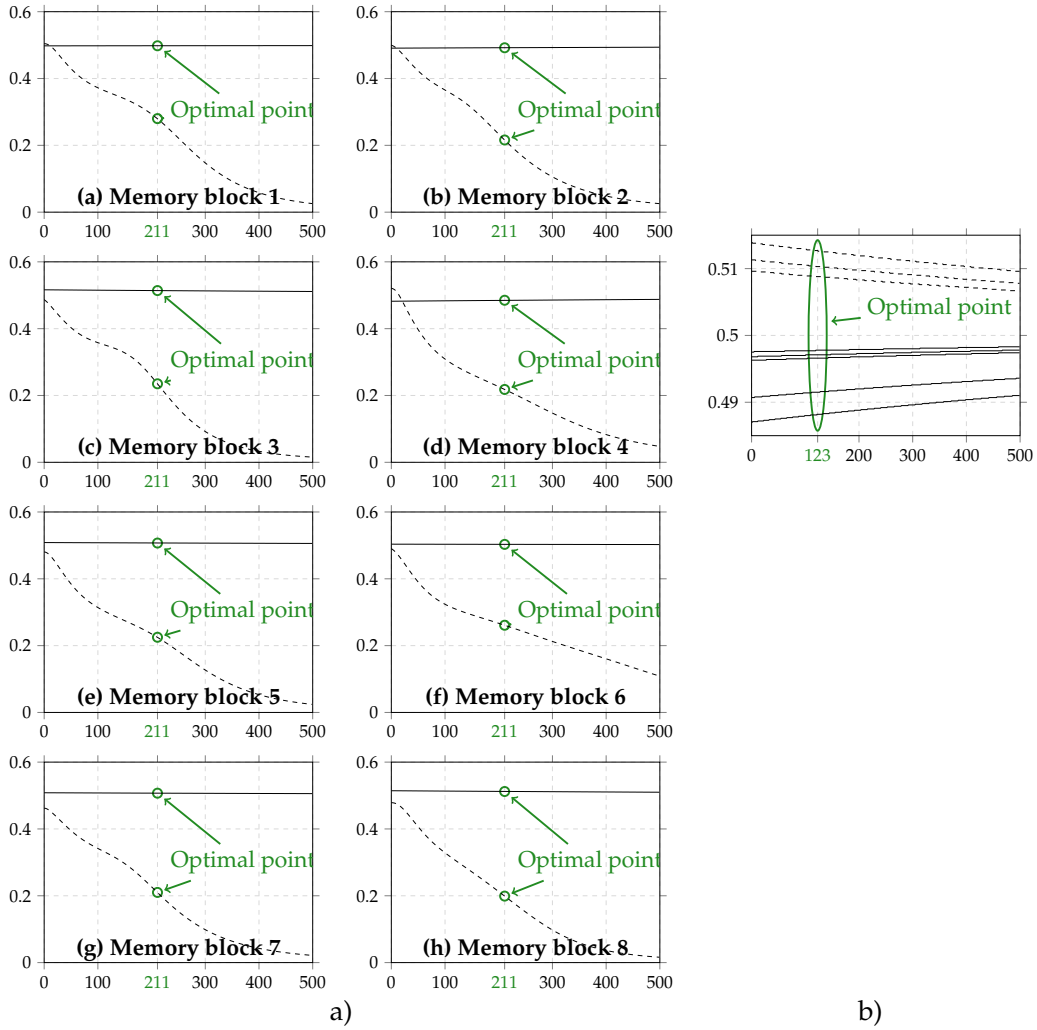


Figure 5.2: *An example a) of the update (z black dashed curves) and reset (r black curves)* **gates activity** *and b) of the* **PMU gate** *u* **activity** *for* **short-term-dependencies** *for* **short-term** *dependencies from the* **Decoda** *corpus for each iteration (from 0 to 500 X-axis) for the 8 memory blocks (hidden units).*

One can easily point out that the update gate z converges to 0 and then let the reset r gate to manage both long- and short-term dependencies. Therefore, the reset gate r controls the short-term dependencies and the long-term dependencies are avoided. This is also mainly due to the small size of the training corpus that contains 740 documents alongside with the small number of themes (or classes) 8. Thus the prior for a given sequence to be met is quite high. Figure 5.2 a) underlines that the update gate $z \simeq 0$ after 500 iterations and $z \simeq 0.2$ for the optimal point (real test accuracy). The reset gate $r \simeq 0.5$ regardless of the number of iterations. Figure 5.2 b) shows the u gate activity of the proposed PMU-RNN for the 8 hidden neurons. For convenience

and readability, part of the neurons (3) are in dashed curves and the others in plain curves. It is worth emphasizing that the role of short and long-term dependencies is split between the 8 gates. Indeed, the Figure 5.2 b) shows that 3 dashed curves of the gates u of the PMU-RNN hidden layer move down and are devoted to short-term dependencies (see Table 5.1) while the 5 others move up and code the long-term dependencies. The proposed PMU allows the RNN to separate well the learning process of short- and long-term sequences in different distinct hidden states (3 for short-term and 5 for long-term).

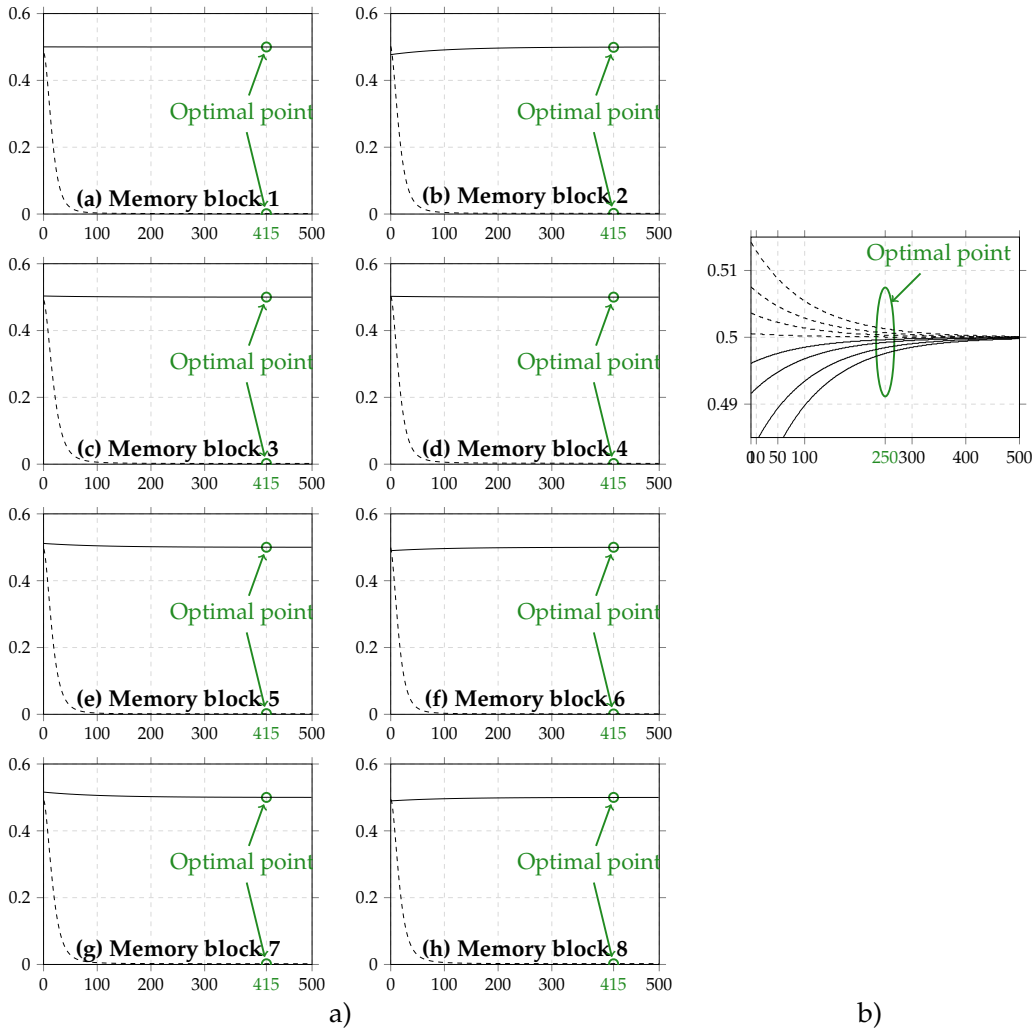


Figure 5.3: *An example a) of the update (z black dashed curves) and reset (r black curves)* **gates activity** *for* **long-term** *dependencies and b) of the* **PMU gate** *u* **activity** *for* **long-term-dependencies** *from the* **20-Newsgroups** *corpus for each iteration (from 0 to 500 X-axis) for the 8 memory blocks (hidden units).*

Long-term gates activity from a large microblogging data set

The activity for the 8 hidden states of the two z and r gates of a GRU-RNN during the

learning process for a given (randomly selected) document is detailed in Figure 5.3 a). The dashed curves represent the activity of the update gate z and the plain black curves the reset gate r of the GRU-RNN. We can first underline that the curves follow the same trend as those for short-term dependencies of spoken dialogues corpus. Nevertheless, the curves are steeper and z converges such as in the case of short-term dependencies from the DECODA corpus, to 0. The reset r gate also manages as in the case of DECODA corpus both long- and short term dependencies. Consequently, the reset gate r controls the short-term dependencies and the long term dependencies are avoided. Figure 5.3 b) shows the u gate activity of the proposed PMU-RNN for the 8 hidden neurons. For convenience and readability part of the neurons (4) is in dashed curves and the others in plain ones. It is worth emphasizing that the role of short and long-term dependencies are well-split between the 8 PMU gates. Indeed, figure 5.3 b) shows that 4 dashed gates u of the PMU-RNN hidden layer move down and are devoted to short-term dependencies (see Table 5.1) while the 4 others move up and code long-term dependencies. The PMU allows the RNN to split the learning process of short and long-term sequences in different distincts hidden states better (4 for short-term and 4 for long-term).

5.4.2 Short-term dependencies from spoken dialogues

This section studies the performances of RNN-based models during a theme identification task of spoken dialogues from the small DECODA corpus with a small training data set that tends to mostly reveal short-term dependencies; moreover this corpus contains a small number of close themes (8) (Morchid et al., 2014c, 2016). We compare RNN, LSTM, GRU to the proposed PMU in terms of accuracy and processing time required for learning. Table 5.2 sums up the accuracies and processing times with the confidence interval (CI).

Accuracies on a theme identification of spoken dialogues
We underlines first that the best accuracies observed for RNN, LSTM, GRU and PMU are equal (88.6%) but the RNN reaches the lowest accuracy (84.6%) with the development data set. Nonetheless, Table 5.2 shows that the accuracy obtained in **real conditions**[3] on the test data set is obtained with the PMU (82.3%) which is better than the performance of the RNN (81.9%) and the LSTM (81.7%). The GRU which is the most related model to the proposed PMU-RNN, real accuracy on the test data set 80.4% is 1.9 point under the PMU one. The best configuration point is close to the maximum for the test set in the case of the RNN. Indeed, the gap for the PMU between the maximum and optimal configuration points is 1 (36 for the maximum and 37 for the optimal) and is lower than the gap for the RNN (gap=2).

PMU vs. other RNNs in terms of processing time
We have noticed during the experiments that the processing time observed for the

[3]The best operating point (number of neurones in the hidden layer) estimated on the development set is applied to the test set.

RNNs (RNN, LSTM, GRU and PMU) follows the same trend as the number of iterations. Nevertheless, the high gap between the maximum of the RNN (500 iterations) and the others (roughly 250 iterations) is not observed for the processing time curves (about 200 seconds for RNN, LSTM and GRU). Moreover, the maximum PT required for the proposed PMU is 174 with a gain of more than 20% compared to the GRU maximum PT. An interesting remark regarding these curves is that the PMU is well adapted to small data sets (requiring a low number of neurons in the hidden layer and a small number of basic operations). These observations are confirmed in Table 5.2. Moreover, the best accuracy reached after the PMU one is 81.9% with the mere RNN. The PMU require only 23 sec. (for the RNN) to reach this accuracy, that represents only one quater of the PT of the RNN (83 sec.).

Table 5.2: **Sum up** *of accuracies (± confidence interval), number of iterations and processing time (seconds) with the* **DECODA** *corpus.*

Neural model	hidden size	Test acc. %	Number of iterations	Proc. time in seconds
RNN	38	81.9 (±4.16)	500	83
LSTM	25	81.7 (±4.18)	162	69
GRU	16	80.4 (±4.29)	211	40
PMU	37	**82.3 (**±4.13**)**	**123**	**37**

5.4.3 Long-term dependencies from 20-Newsgroups documents

The high number of documents (20,000) alongside with the small number of subjects (20 classes) discussed lead to long-term dependencies between words and topics in the 20-Newsgroups corpus. The next sections evaluate the RNNs by varying the number of neurons in the hidden layer from 10 to 300 with a step of 5 in terms of accuracy and time processing. Table 5.3 sums up the performances in terms of accuracy and processing time.

Classification accuracies on a large data set of documents
The curves of gated-RNNs are more stable with a gap between the lowest and highest accuracies equal to about 3 points for both LSTM, GRU and PMU. The number of hidden neurons required to reach the best accuracy on the development data set is small for the PMU (155) compared to those for the RNN (290), LSTM (230) and GRU (215). Nonetheless the real accuracy on the test data set is obtained with the LSTM and GRU (75.0) but is close to the one from the PMU (74.9%) which is better than the performance of the RNN (74.4%). The gain observed for the gated RNNs is more than 5 points compared to the RNN. The large data set allows the RNNs to not observe the drawback related to the gap between the real and maximum accuracies reached on the test data set. Table 5.3 sums up the real accuracies for each RNN. The second column of Table 5.3 depicts the number of required neurons from the hidden layer to reach the optimal configuration on the development data set. During the experiments, the

number of neurons varies from 10 to 300 with a step of 5 and Table 5.3 shows that a large number of neurons are needed to reach this optimal point (higher of 200 neurons for RNN, LSTM and GRU). Nonetheless, the proposed PMU reaches its best accuracy on the development data set with fewer neurones (155) and shows its capability to better code long-term dependencies in fewer hidden states.

PMU vs. others RNNs in terms of processing time for the classification task of 20-Newsgroups

It is worth pointing out that the maximum processing time (PT) of the RNN, LSTM and GRU (more than 5,000 sec.) is 20% higher than the maximum PT of the PMU (3,980 sec.). We can remark that the minimum PT of the PMU is 156 seconds and quite low compared to the minimum of the RNN (272 sec.), GRU (410 sec.) and LSTM (545 sec.) with a gain of more than 43%. Moreover, the PMU has a flat curve compared to the other RNNS with a small gap between the minimum and maximum PT of 3,824 in regards with those of the RNN (4,815), LSTM (5,215) and the GRU (5,165). Therefore, the proposed PMU is more stable in relation to the variation of the hidden size, and thus, to the size of the corpus (more documents requires more neurons in the hidden layer to well-code the latent relations between the input features). The last column of Table 5.3 shows the PT needed to obtain the real test accuracies. We can easily remark that the best PT is observed in real conditions for the test data set since the PMU-RNN is employed (2,282 sec.). The RNN, LSTM and GRU obtain worse performances (more than 5,100 sec.) than the PMU. The proposed Parsimonious Memory Unit allows

Table 5.3: **Sum up** *of the classification task of* **20-Newsgroups** *corpus performances in terms of accuracies and processing time.*

Neural model	hidden size	Test acc. %	Number of iterations	Proc. time in seconds
RNN	290	74.4 ($\pm$0.98)	500	5,053
LSTM	230	**75.0** ($\pm$0.98)	480	5,554
GRU	215	**75.0** ($\pm$0.98)	415	5,127
PMU	**155**	74.9 ($\pm$0.98)	**250**	**2,282**

the RNN to reach equivalent even better accuracies than state-of-the-art GRU- or LSTM-RNN with less processing time and a smaller hidden layer size. Moreover, the PMU better-manage short and long-term dependencies. Indeed, each hidden neuron deals with a particular aspect of the term dependencies (short or long) which is not the case for the GRU (same gate activity for all hidden units). The experiments underline that the number of hidden units required for learning is correlated with the size of the corpus and a huge number of hidden units lead to model over-fitting for small data-sets (DECODA). This phenomenon is not observed since the data-set is large enough to require a large number of hidden units to code internal statistical dependencies (20Newsgroups). Overall, the proposed PMU-RNN obtains better performances on the language modeling task than the other RNNs with less processing time; moreover, the PMU requires the same period for training than the straightforward RNN while the PMU manages better both short- and long-term dependencies.

Chapter 6

Generative Adversarial Networks

Contents

Abstract

This chapter presents the first effort on studying Generative Adversarial Networks (GAN) for NLP. GAN is a dedicated sud domain of deep learning and attracts a wide range of AI researchers. We present in this chapter how GAN based models can allow the NLP system to extract from ASR transcripts robust features based on human transcribed documents (TRS) to better predict the most related theme of a given spoken dialogue.

6.1 Introduction

The work presented in this chapter reflect my recent interest in generative models to extract robust representations of noisy documents. The models proposed here is the logical continuation to autoencoders studied in chapter 2 and more precisely with the TSDAE et SDAE for NLP. But NLP neural based systems are powerful to express relevant content when the spoken documents are recorded in controlled conditions. These systems are also limited by the quality of transcripts from the automatic speech recognition system (ASR). In chapter 2, we have proposed to use the knowledge available at training time through the TRS transcriptions to enhance the input representation of the ASR versions. This enhancement is made possible with the use of stacked and deep stacked auto-encoders to learn a static mapping that projects the ASR representation to the TRS latent space. Based on the promising results observed with this approach we have proposed to further investigate the distillation of the TRS knowledge to the ASR representation with the recent generative adversarial networks (GAN). GANs are an very active field of research and offer an interesting approach that focuses on a game-theoretic method to train a generative model (Goodfellow et al., 2014). Numerous architectures have been proposed to address various tasks (Radford et al., 2015; Wu et al., 2017a). From a simplified perspective GANs are commonly used to learn a mapping from a random noise space to a target one, making it possible to generate new unseen samples. In NLP tasks the noise space is commonly replaced with a well defined input representation such as text written in a specific language for neural machine translation (Yang et al., 2017). Then GANs project this latent representation to a different target language. In the task of theme identification of telephone conversations investigated, we consider the latent ASR transcription as the noise space and the TRS versions as the target one. After training the model is expected to enhance the ASR latent representation with TRS knowledge to further improve the results when classifying the documents. We propose a task adapted model called Machine-to-Human GAN (M2H-GAN) by merging the GAN with a semi-supervised GAN (SGAN) to better represent and classify telephone conversations. This chapter Introduces first a new GAN architecture called M2H-GAN to efficiently map the automatically transcribed representation of conversations, to a latent representation of their manually transcribed version and then compares the classification accuracy obtained with this new representation to other methods on a theme identification of telephone conversations task (Section 6.3).

6.2 Generative neural models

We propose a GAN merged with a semi-supervised SGAN to allow a projection of an automatically transcribed document, to its manual transcription representation with the Machine-to-Human GAN (M2H-GAN).

Generative Adversarial Networks
In a generative adversarial network (Goodfellow et al., 2014), two neural networks are

trained in opposition. First, a generator G outputs a fake object named $\tilde{x}$ from an input random noise vector z:

$$\tilde{x} = G(z) \tag{6.1}$$

Then a discriminator D receives alternatively a true sample x or a fake one $\tilde{x}$ from G, and outputs a probability distribution of the input being a fake or not. During training, D tries to maximize the log-likelihood of the correctly assigned source:

$$L = \mathbb{E}[\log p(\text{real}|x)] + \mathbb{E}[\log p(\text{fake}|\tilde{x})] \tag{6.2}$$

In the same manner G is trained to fool D by minimizing the second term of Eq. 6.2. Indeed, reducing the probability of correct classification of fake inputs increases the generating capability of G.

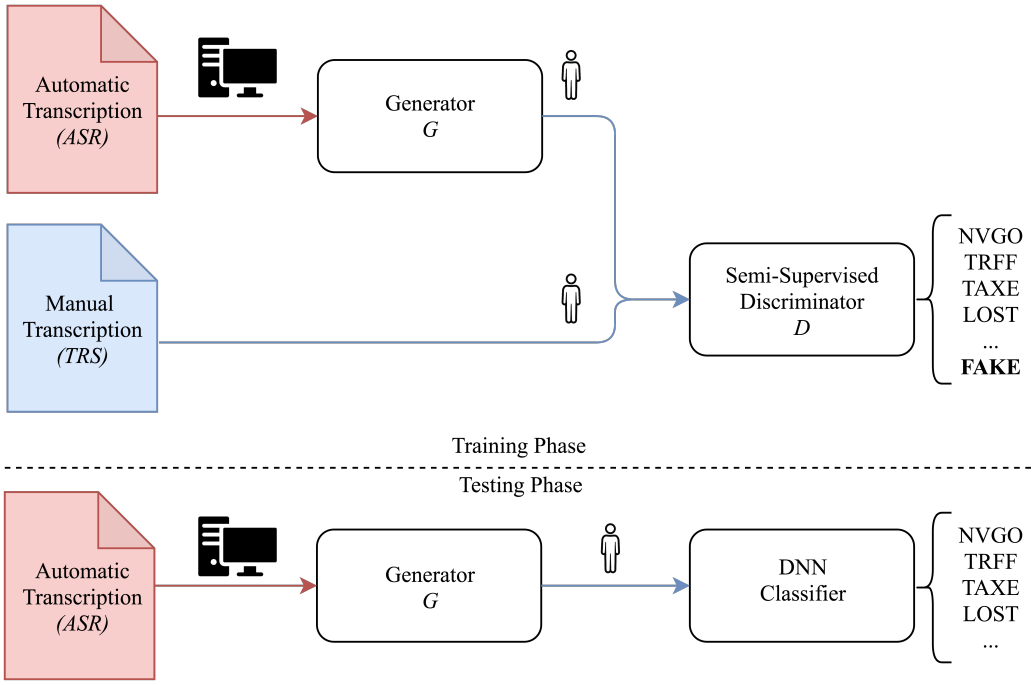


Figure 6.1: *Illustration of the M2H-GAN architecture at training (top) and testing (bottom) time. Red and blue lines show the ASR and TRS representation signal. Note that the output of the generator G goes from red to blue during the training phase.*

Auxiliary and semi-supervised GANs (Mirza et Osindero, 2014; Odena, 2016) have been proposed to take into consideration the labels in both the generator and the discriminator to drive the generation process toward a specific class. In an SGAN, D is trained to determine if the input signal is fake and select the most suitable label. Consequently the output dimension of D is of size $N + 1$ with N being the number of classes and $+1$ representing the *fake* case. The loss function remains unchanged. SGANs use labels to add a condition on the generation process, making it possible to generate samples of a specific class, such as *car* or *bird* for image generation.

Machine-to-Human representation with generative models
We propose to merge the initial GAN with its semi-supervised version SGAN in a model named Machine-to-Human GAN(M2H-GAN). An overview of the M2H-GAN architecture is depicted in Figure 6.1. In M2H-GAN, $\tilde{x}$ is the generated representation of an automatically transcribed document (ASR) from G and x is the "clean" TRS version of the same sample. D is trained to determine if the input has been generated, or belongs to a certain class (SGAN), and thus contains $N + 1$ output neurons. Consequently G is jointly trained to map the ASR transcripts to a latent TRS and "clean" space in order to fool the discriminator. Unlike for SGAN, the generator of M2H-GAN does not have access to the label due to the fact that conversations classes are unknown during the testing phase. This modification allows the discriminator to have more room to discover if an input is fake or not, making it more powerful. As a consequence the generator must create a more convincing representation of the ASR signal, and receives gradients according to the label, without any conditioning on the input. An overview of M2H-GAN is depicted in Figure 6.1.

6.3 Experiments

This section introduces the theme classification of telephone conversations task with the DECODA dataset along with the proposed representation of the document.

Spoken conversations dataset
The corpus of spoken conversations is a set of automatically transcribed and annotated human-human telephone conversations of the Paris transportation system CCS (RATP). This corpus comes from the first version of the DECODA project detailed in section 2.3.1 and is employed to evaluate the effectiveness of the proposed M2H-GAN on a conversation theme identification task. The DECODA corpus is composed of $1,242$ telephone conversations recorded during high traffic days in the capital, which is equivalent to about 74 hours of signal. The dataset was split into 8 categories or dominant themes that are detailed in Table 2.1. An example of a manually transcribed conversation of DECODA.

Abstract document representation with LDA
The latent Dirichlet allocation or LDA is an effective method to represent documents in an unsupervised manner, as probability distributions of hidden topics (Blei et al., 2003) in a document, and have shown their efficiency in many previous related works (Morchid et al., 2013; Parcollet et al., 2016). For the experiments described in this section LDA models are trained over the training set of DECODA following the standard hyper-parameters heuristic (Blei et al., 2003). It is important to note that two LDA models are trained with either the ASR or TRS conversations from the training sub-set of the DECODA data-set. Consequently $\alpha = \frac{50}{T}$, with T the number of topics, and $\beta = 0.01$. The number T has been previously investigated for this task in (Parcollet et al., 2016, 2017), and is set to 25. More precisely, 10 runs of the $T = 25$ LDA

model are concatenated to obtain a final vector of size $25 \times 10 = 250$ to alleviate any variations. Then every conversation is projected into the corresponding LDA space, and is embedded in a vector of size 250.

Experimental protocol
To evaluate the effectiveness of the M2H-GAN to generate TRS-like representations of ASR transcripts we compare M2H-GAN to a GAN model on the theme classification of telephone conversations. Deep feed-forward NNs trained on TRS and ASR transcripts are used as baselines. We also compare M2H-GAN to previously investigated generative models (Janod et al., 2016). Training and testing steps are detailed in Algorithm 1, and can be summarized as follows: 1) Train GAN or M2H-GAN models; 2) Freeze the generator and train a DNN classifier from the generated features. Finally, Figure 6.1 represents the global architecture of the model.

Algorithm 1 Training procedures.

procedure TRAIN GANS(X_{trs},X_{asr})
    Project X_{trs}, X_{asr} in LDA to obtain Z_{trs}, Z_{asr}.
    Generate $\tilde{X}_{asr}$ with G from Z_{asr}.
    Train D and G based on $\tilde{X}_{asr}$, Z_{trs}. (Goodfellow et al., 2014).
procedure TRAIN DNNS(X_{asr})
    Project X_{asr} in LDA to obtain Z_{asr}.
    Generate $\tilde{X}_{asr}$ with frozen G from Z_{asr}.
    Train a DNN to classify $\tilde{X}_{asr}$.

DNNs. Classifiers rely on 2 hidden layers of size 256 with tanh activations, and a final *softmax* layer corresponding to the 8 themes of the DECODA dataset (Bechet et al., 2012). They are trained during 40 epochs based on the Adam optimizer (Kingma et Ba, 2014) with vanilla hyper-parameters and no regularization techniques. After training the maximum accuracy obtained on the test, along with the best result *w.r.t* to the best validation performances are saved.

GAN. The generator is made of 2 hidden layers of size 512 and 250 (corresponding to the size of the LDA vector) with layer-wise normalization (Ba et al., 2016) and tanh activations, while the discriminator is composed of 2 layers of 128 and 8 neurons with tanh and *sigmoid* activation functions. The discriminating labels are smoothed by being sampled from a uniform distribution bounded by $[0.0, 0.7]$ for the valid ones and by $[0.7, 1.0]$ for the fake ones as proposed in (Salimans et al., 2016).

M2H-GAN. The generator is identical to the GAN baseline. The discriminator also includes a semi-supervised classification task. Consequently the output layer is made of 9 neurons for the 8 themes of the DECODA framework plus the *FAKE* label.
Both GAN and M2H-GAN generators are trained to minimize the binary cross-entropy loss observed with the discriminator predictions on their fake generated features while their discriminators maximize the binary and traditional cross-entropy loss functions of correctly classified sources. Finally, models are trained in an adversarial manner as proposed in (Goodfellow et al., 2014) during 25 epochs with SGD, no momentum, and

with a learning rate set to 0.02.

6.4 Results and Discussion

Two baselines DNN classifiers are tested on both the ASR and TRS versions of the DECODA corpus. Then, GAN-based approaches are trained following Algorithm 1. All the experiments are performed 10 times and averaged to alleviate variations due to the random initialization of the parameters.
Table 6.1 reports the average accuracies observed with the GAN, and the more adapted M2H-GAN approaches compared to simple DNN classifiers on the DECODA task. It is firstly important to note the difference in term of accuracy between the two baselines during the theme identification on both ASR and TRS transcripts. Indeed, while the standard deviation remains almost equal both real (*w.r.t* to the validation set) and max test accuracies are different. More precisely, the ASR-based DNN obtains a real test accuracy of 83.4% compared to 88.0% for the TRS-based DNN, representing a drop of 4.6%. This is easily explained by the high WER observed on the ASR transcriptions that alter significantly the LDA representation and the final classification performances. These results support the initial intuition that a translation of ASR documents to TRS-like representations allow us to better identify the most related theme of a spoken dialogue.
As a first step to reduce this gap ASR transcripts inputs are mapped to the TRS ones with a GAN. This approach obtains a best test accuracy of 84.1% for ASR inputs, reducing the absolute difference with TRS performances to 3.9%. The standard deviation is also lowered to 0.012 resulting in a slightly more stable model. Validation performances are altered with an average accuracy of 87.0% compared to 89.5% and 92.5% for the DNNs trained on the ASR and TRS respectively.

Table 6.1: *Accuracies obtained by various models on the DECODA corpus. "Real Test" stands for the performances observed on the test set w.r.t to the validation set, while "Max Test" are the best results obtained. The 'Data" column gives information on the data used for training. Results are averaged over* 10 *runs. The standard deviation is computed over these runs and concern the "Real Test" performances.*

Models	**Data**	**Dev.**	**Real Test**	**Max Test**	**Std. Dev.**
DNN	TRS	92.5	88.0	88.5	0.016
DNN	ASR	89.5	83.4	84.6	0.017
GAN	ASR	87.0	84.1	85.2	0.012
M2H-GAN	**ASR**	**90.0**	**85.5**	**85.8**	**0.007**

The Machine-to-Human mapping is then performed with the M2H-GAN. The real test accuracy is increased to 85.5% representing a absolute gain of 1.4% and 2.1% compared to the simpler GAN and DNN classifier respectively. The gap between the ASR classification performances and the TRS ones is also reduced to 2.5%. It is also worth

underlying that the standard deviation is halved (0.007) in comparison to all the other models, resulting in a more robust representation of the spoken document content.

Table 6.2: *Accuracies obtained by proposed generative models compared to previous works on the DECODA corpus. "Real Test" stands for the performances observed on the test set w.r.t to the validation set while "Max Test" are the best results obtained. The 'Data" column informs on the data used for training. Results are averaged over* 10 *runs. The standard deviation is computed over these runs and concerns the "Real Test".*

Models	**Data**	**Dev.**	**Real Test**	**Max Test**	**Std. Dev.**
AE(Janod et al., 2016)	ASR	-	81	-	-
DAE(Janod et al., 2016)	ASR	-	-	74.3	-
DSAE(Janod et al., 2016)	ASR	88.0	82.0	83.0	-
QDAE(Titouan et al., 2017)	ASR	90.0	85.2	85.2	-
GAN	ASR	87.0	84.1	85.2	0.012
M2H-GAN	**ASR**	**90.0**	**85.5**	**85.8**	**0.007**

Table 6.2 shows the results observed with GAN and M2H-GAN models compared to previously investigated generative models. Both GAN and M2H outperform the auto-encoders (AE), denoising auto-encoders (DAE), and deep stacked auto-encoders (DSAE) proposed in (Janod et al., 2016, 2017). Indeed, encoder and decoder are trained jointly to minimize the reconstruction error while the generator and discriminator are trained on different objectives impacting on each other. M2H-GAN also give better results than the recent quaternion-valued denoising auto-encoder (QDAE) despite the fact that the QDAE is based on a better document representation and a specific segmentation with the quaternion algebra.

Chapter 7

General Perspectives

The previous chapters depict novel research directions. Nonetheless, the efforts for real-valued neural networks (Part I) and for complex-valued neural networks (Part II) will continu throughout different Ph.D. thesis and projects. All these research directions will also be followed during the AISSPER project (ANR AAPG 2019) in collaboration with the LIUM and Orkis (from 2020 to 2023). Indeed, AISSSPER aims to provide robust NLP systems that extract relevant information from spoken documents based on neural networks based models such as QNNs and autoencoders. As scientific coordinator of the project, my expertise will be at the hart of the proposed neural networks based models. The will lead me to explore all these promising researches with different M.Sc. and Ph.D. students and PostDocs.

Real-valued Neural Networks

Chapter 1 presents the parallel long short-term memory recurrent neural network. This model aims to code latent dependencies between parallel sequences. This model consider that an information from a given sequence (or stream) $\mathbf{s}$ is able to be predicted with the effort of other sequences $\mathbf{s}'$ and $\mathbf{s}$. Indeed, the LSTM takes the decision to retain or forget the previous hidden state and decides how much from the hidden state candidate is required to predict the next element of the considered sequence. Nonetheless, the process of forgetting the previous hidden state and the part of the hidden state exposed to the output can be made with the collaboration of all sequences but the decision to select the best hidden state candidate (input gate) have to be made by the concerned sequence (sequence to predict). Indeed, the other sequences have not enough information to select the best candidate for the next element composing the sequence. Therefore this process has to be made by $\mathbf{s}$.

The works on encoder-decoder in collaboration with Killian Janod during his Ph.D. thesis presented in chapter 2 are based on mere neural networks. Nowadays novel architectures allow the neural networks based systems to better handle latent relations between spoken/transcribed features from spoken documents. Among these models Generative Adversarial Networks (GAN) and conditional GAN learn the mapping from a document to another. Therefore a perspective of this work is to employ automatically transcribed from an ASR and human transcriptions of spoken docu-

ments as "true" and "fake" version of the spoken documents content. The important interest from researchers for GAN is mostly due to their capability to learn from noise documents how to build robust representations.

Quaternion-valued Neural Networks

Chapters 3 and 4 presents the Quaternion-valued neural networks (QNN). QNNs have encountered a great interest from researchers. This study have been made in collaboration with Titouan Parcollet during his Ph.D. thesis. His application for an internship at the MILA [1] has been selected to start an collaboration to develop the quaternion neural networks for different NLP elated tasks. The promise of QNNs has been shown in different NLP domains (speech recognition, theme identification, image processing) and different configurations (MLP, DNN, denoising auto-encoders, recurrent and convolutional neural networks). I plan to continue these works to extend these models to state-of-the-art algorithms and to evaluate these models in other domains (medical, economics, etc.).

The main issue related to Quaternion-based neural networks, is the representation of documents. In chapter 3 and 4 we have proposed to representing the document content with LDA based features for the user, the agent and the whole document. Therefore we obtain three sub-features for each topic from LDA. This representation reveals little in the way of interdependence between these sub-features; moreover, the dependance between a given topic for the user and agent is not trivial and difficult to understand. For these reasons we plan to discover novel features of document content to better code the information in the quaternion. We can base this research on image features such as the Red, Green, Blue (RGB) that code a pixel.

Long-term Perspectives for Efficient Neural Networks

My ongoing projects are centered on recurrent neural networks (RNN). The aim of this direction is to better understood how RNNs learn to learn and how the memory is retained throughout the time. The study presented in Chapter 5 proposes a novel RNN-based model that better manages short- and long-term dependencies with less processing time required during the learning phase than GRU. The results obtained show that the PMU model is able to compute models from huge data-sets with less parameters. Therefore such algorithms may have a strong impact on artificial intelligence on mobile devices. Since the number of documents available on Internet is exponentially growing the needs for such models and algorithms become crucial for academics and industrials. Therefore I plan to extend these works during different projects involving industry. Chapter 6 shows that GANs learn from manually transcribed spoken documents robust mapping during a SLU task. The expressive power of GAN will be studied during future investigations. For example this study has underlined the instability during the learning process of both generator and discriminator networks. We plan to propose neural based architectures that focus only on maintaining the optimal balance between generator and discriminator during the learning process of a GAN.

[1] https://mila.quebec

Part IV

Appendix

Recurrent Neural Networks

Basic of Recurrent Neural Networks

Figure 1 presents the conventional RNN folded (a) and unrolled (b). The hidden state h_t is processed with different cells.

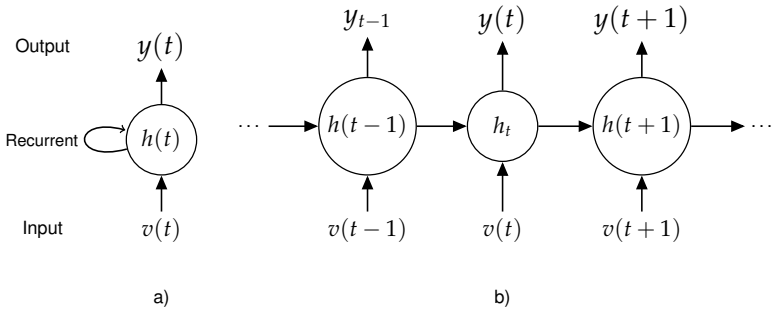


Figure 1: *Illustration of (a) an Recurrent Neural Network (RNN) with loops and (b) an unrolled RNN for input x_t and its associated output y_t presented as a chain-like that reveals the link between sequences throughout the time.*

The Recurrent Neural Network (RNN) is an extension of a straightforward feed-forward neural network that handles a variable-length sequence as input by receiving a recurrent hidden state. The activation of this hidden state depends on the previous hidden state at each time-step. Therefore the activation of the j-th hidden state h_j at time-step t is computed as a state-to-state transition function σ_h of the current input vector v_t and the previous hidden states:

$$h_j(t) = \sigma(x_{hj}(t)) \tag{1}$$

$$x_{hj}(t) = \sum_m s_{h_j m} v_m(t) + w_{h_j m} h_m(t-1) \tag{2}$$

where $x_{hj}(t)$ is the network sum at time step t, $w_{h_j m}$ (from unit m to h_j) and s_{lm} are the state-to-state recurrent and input-to-hidden weight connection respectively; σ_h is the logistic sigmoid function. For brevity the biases are not considered. RNNs are able to output a probability distribution over the next element of a sequence given its current

hidden state h_t. The RNNs factorize the probability of a sequence of length T into:

$$p(x_1, \ldots, x_T) = p(x_1)p(x_2|x_1) \ldots p(x_T|x_1, \ldots, x_{T-1})$$

where the last element is a special end-of-sequence value. We model each conditional probability distribution with

$$p(x_t|x_1, \ldots, x_{t-1}) = \phi(h_j(t))$$

where $h_j(t)$ is evaluated with Eq.(1). (Bengio et al., 2003; Mikolov, 2012) have already proposed neural networks to model a probabilistic distribution over sequences. (Bengio et al., 1994; Hochreiter, 1998) expose the difficulty for a RNN to capture long-term dependencies due to the vanishing and exploding gradient problems. The first attempt to address this drawback of RNNs was to introduce an affine transformation followed by a simple element-wise non-linearty from gates units called Long Short-Term Memory (Hochreiter et Schmidhuber, 1997). (Cho et al., 2014) propose a new recurrent unit called Gated Recurrent Unit close to the LSTM. The next sections detail the LSTM and GRU to better underline the benefits of the proposed Parsimonious Memory Unit.

Long Short-Term Memory (LSTM)

The Long Short-Term Memory (LSTM) (Hochreiter et Schmidhuber, 1997) network depicted in Figure 2, is a special case of RNNs (Elman, 1990). The goal of this architecture is to create an internal cell state of the network which allows it to exhibit dynamic temporal behaviour and to process arbitrary sequences of inputs, such as sequences of words (Sundermeyer et al., 2012) for language modeling, time series (Gers et al., 2001), etc. The basic unit in the hidden layer of the LSTM is a memory block that replaces the hidden unit in the RNN. The memory block allows cells to share the same gates to reduce the number of adaptive parameters.

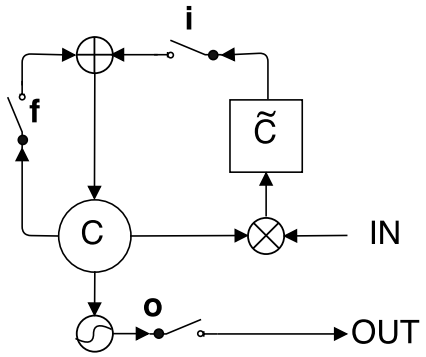


Figure 2: *The graphical illustration from (Chung et al., 2014) of a LTSM. The i, f, and o are the input, forget and output gates respectively.*

This memory cell has in its core a recurrently self-connected gate called forget gate that learns to forget the previous cell state c. The cell state c is updated based on both

the three previously defined gates and its current state. For brevity, the biases are not considered. The input $i_j(t)$, output $o_j(t)$ and forget $f_j(t)$ gates depend on the current input (v_m) and the previous hidden state $h_m(t-1)$) and are computed as follows for the j-th memory block:

$$i_j(t) = \sigma(x_{ij}(t)) \tag{3}$$
$$\text{with } x_{ij}(t) = \sum_m s_{i_j m} v_m(t) + w_{i_j m} h_m(t-1) \tag{4}$$
$$o_j(t) = \sigma(x_{oj}(t)) \tag{5}$$
$$\text{with } x_{oj}(t) = \sum_m s_{o_j m} v_m(t) + w_{o_j m} h_m(t-1) \tag{6}$$
$$f_j(t) = \sigma(x_{fj}(t)) \tag{7}$$
$$\text{with } x_{fj}(t) = \sum_m s_{f_j m} v_m(t) + w_{f_j m} h_m(t-1) \tag{8}$$

where σ is the logistic function; w_{lm} is the weight connection from unit m to unit l ($w_{i_j m}$ connect unit m to gate i_j); $s_{l_j m}$ is the state-to-state recurrent weight connection and σ is a logistic sigmoid function and j indexes memory blocks and the cell c of the j-th memory block is:

$$c_j(t) = f_j(t) c_j(t-1) + i_j(t) \tilde{c}_j(t) \tag{9}$$
$$\tilde{c}_j(t) = \tanh(\sum_m s_{c_j m} v_m(t) + w_{c_j m} h_m(t-1)) \, . \tag{10}$$

Finally the hidden state h_j is computed as follows:

$$h_j(t) = o_j(t) \tanh(c_j(t)) \tag{11}$$

The output gate o_j controls how the memory content (c_j) is exposed. Finally, assuming that the neural network topology contains an input, a hidden and an output layers, the output units y_k of all gates of a given unit are computed by:

$$y_k(t) = \tanh(x_k(t)) \tag{12}$$
$$x_k(t) = \sum_m s_{y_k m} v_m(t) + w_{y_k m} h_m(t-1) \, . \tag{13}$$

The backward pahse is detailed in (Hochreiter et Schmidhuber, 1997).

Bidirectional Long Short-Term Memory (BLSTM)

LSTM networks use only the previous context to predict the next segment for a given sequence. Bidirectional RNN (BRNN) (Schuster et Paliwal, 1997), presented in Figure 3 can process both directions with two separate hidden layers (one for each direction). This type of RNN feeds to a same output layer fed forwarded inputs through the two hidden layers. Therefore the BRNN computes both *forward* hidden sequence $\overrightarrow{\mathbf{h}}$ and

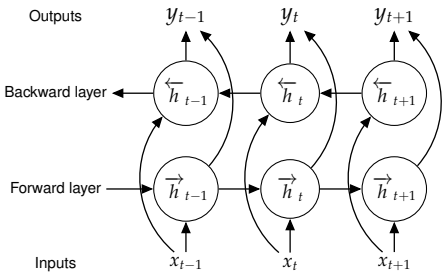


Figure 3: *Bidirectional Recurrent Neural Network (BRNN).*

backward sequence $\overleftarrow{\mathbf{h}}$ as well as the output vector $\mathbf{y}$ by iterating $\overrightarrow{\mathbf{h}}$ from $t = 1$ to T, and $\overleftarrow{\mathbf{h}}$ from $t = T$ to 1:

$$\overrightarrow{h}_j(t) = \mathcal{H}(s_{\overrightarrow{h}_j m} v_m(t) + w_{\overrightarrow{h}_j m} \overrightarrow{h}_m(t-1)) \tag{14}$$

$$\overleftarrow{h}_j(t) = \mathcal{H}(s_{\overleftarrow{h}_j m} v_m(t) + w_{\overleftarrow{h}_j m} \overleftarrow{h}_m(t-1)) \tag{15}$$

$$y_j(t) = s_{\overrightarrow{h}\, y} \overrightarrow{h}_j(t) + w_{\overleftarrow{h}\, y} \overleftarrow{h}_j(t) \tag{16}$$

By replacing the BRNN cells with LSTM cells the Bidirectionnal LSTM (BLSTM) (Graves et Schmidhuber, 2005) is obtained. The BLSTM allows to exhibit long range context dependencies and takes advantage from the two directions structure. The output vector $\mathbf{y}$ is processed by evaluating simultaneously the two directions hidden sequences by computing the composite function $\mathcal{H}$ in the forward ($\overrightarrow{\mathbf{h}}$) and backward ($\overleftarrow{\mathbf{h}}$) directions.

Gated Recurrent Unit

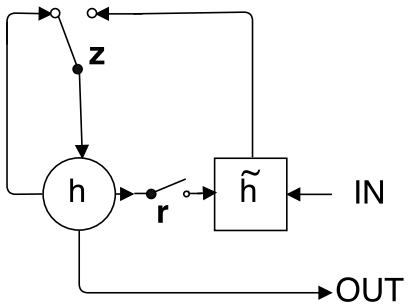


Figure 4: *The graphical illustration from (Chung et al., 2014) of a GRU. r and z represent the reset and update gates; h and $\tilde{h}$ are the activation and the new activation state for the GRU.*

The basic unit in the GRU depicted in Figure 4 is a hidden state learned during the forward and backward processes.

Forward pass: Reset r and update z gates are computed as:

$$r_j(t) = \sigma_r(x_{rj}(t)) \tag{17}$$
$$x_{rj}(t) = \sum_m w_{r_j m} h_m(t-1) \text{ and} \tag{18}$$
$$z_j(t) = \sigma_z(x_{zj}(t)) \tag{19}$$
$$x_{zj}(t) = \sum_m w_{z_j m} h_m(t-1)\ . \tag{20}$$

The activation of the proposed unit h_j is defined as:

$$h_j(t) = (1 - z_j(t))h_j(t-1) + z_j(t)\tilde{h}_j(t) \tag{21}$$
$$\tilde{h}_j(t) = \sigma_h(x_{hj}) \tag{22}$$
$$x_{hj} = \sum_m w_{h_j m} h_m(t-1) r_j(t). \tag{23}$$

Finally, the output units y_k are computed by:

$$y_k(t) = \sigma_k(x_k(t)) \tag{24}$$
$$x_k(t) = \sum_m w_{km} h_m(t-1). \tag{25}$$

Backward pass: The standard error objective function based on the target output t_k is defined as for the PMU. In the same way as for the PMU unit, the weight changes $\Delta w_{y_k m}$ for output units are processed as:

$$\Delta w_{y_k m} = \alpha\ \sigma'_k(x_k(t))\ e_k(t)\ h_m(t-1)\ . \tag{26}$$

The weight changes for the the reset $(l = r)$ and the update $(l = z)$ gates and the hidden state $(l = h)$ are:

$$\Delta w_{lm} = \alpha\ e_{h_j}(t)\ \frac{\partial h_j(t)}{\partial w_{lm}} \tag{27}$$

and its internal error state $e_{h_j}(t)$ is:

$$e_{h_j}(t) = \sum_k \sigma'_k(x_k(t))\ w_{h_j l}\ e_k(t)\ . \tag{28}$$

To calculate the partial $\dfrac{\partial h_j(t)}{\partial w_{lm}}$ from Eq.(27), we differentiate Eq.(21) and obtain a sum of three terms:

$$\frac{\partial h_j(t)}{\partial w_{lm}} = \frac{\partial h_j(t-1)}{\partial w_{lm}}(1 - z_j(t)) + z_j(t)\frac{\partial \tilde{h}_j(t)}{\partial w_{lm}} + \frac{\partial z_j(t)}{\partial w_{lm}}(\tilde{h}_j(t) - h_j(t-1)). \tag{29}$$

Differentiating the forward pass equations (17), (18), (19), (20), (21) and (23) for r, z and h we can substitute the unresolved partials and split the expression on the right hand

of equation (29) into three separate equations for the reset ($l = r$), update ($l = z$) gates and for the hidden state ($l = h$):

$$\frac{\partial h_j(t)}{\partial w_{r_{jm}}} = \frac{\partial h_j(t-1)}{\partial w_{r_{jm}}}(1 - z_j(t)) + z_j(t)\frac{\partial \tilde{h}_j(t)}{\partial w_{r_{jm}}} + \frac{\partial z_j(t)}{\partial w_{r_{jm}}}(\tilde{h}_j(t) - h_j(t-1))$$

with $\frac{\partial z_j(t)}{\partial w_{r_{jm}}} = 0$ and $\frac{\partial \tilde{h}_j(t)}{\partial w_{r_{jm}}} = \frac{\partial \sigma_h(x_{hj}(t))}{\partial w_{r_{jm}}}$, Therefore

$$\frac{\partial h_j(t)}{\partial w_{r_{jm}}} = \frac{\partial h_j(t-1)}{\partial w_{r_{jm}}}\left(1 - z_j(t)\right) + z_j(t)\frac{\partial \sigma_h(x_{hj}(t))}{\partial x_{hj}}\frac{\partial x_{hj}}{\partial r_j(t)}\frac{\partial r_j(t))}{\partial x_{rj}(t))}\frac{\partial x_{rj}(t))}{\partial w_{r_{jm}}}$$

finally for the reset gate r:

$$\frac{\partial h_j(t)}{\partial w_{r_{jm}}} = \left(1 - z_j(t)\right)\frac{\partial h_j(t-1)}{\partial w_{r_{jm}}} + z_j(t)\sigma_h'(x_{hj}(t))w_{h_jm}h_m(t-1)^2\sigma_r'(x_{rt}(t))$$

in the same way with $\frac{\partial \tilde{h}_j(t)}{\partial w_{z_{jm}}} = 0$, we obtain for gate z:

$$\frac{\partial h_j(t)}{\partial w_{z_{jm}}} = \frac{\partial h_j(t-1)}{\partial w_{z_{jm}}}\left(1 - z_j(t)\right) + + \left(\sigma_h(x_{hj}(t)) - h_j(t-1)\right)\sigma_z'(x_{zj}(t))h_m(t-1)\,. \tag{30}$$

In a same manner with $\frac{\partial z_j(t)}{\partial w_{h_jm}} = 0$, we calculate the partial for the hidden state ($l = h_j$):

$$\frac{\partial h_j(t)}{\partial w_{h_jm}} = \frac{\partial h_j(t-1)}{\partial w_{h_jm}}\left(1 - z_j(t)\right) + z_j(t)r_j(t)\sigma_h'(x_{hj})h_m(t-1). \tag{31}$$

Based on equation (27) and the partials in equations (30) and (31) we can calculate the corresponding weight updates with the internal state error $e_{h_j}(t)$ defined in equation (28):

$$\Delta w_{r_{jm}} = \alpha\, e_{h_j}(t)\,\frac{\partial h_j(t)}{\partial w_{r_{jm}}}$$

$$\Delta w_{z_{jm}} = \alpha\, e_{h_j}(t)\,\frac{\partial h_j(t)}{\partial w_{z_{jm}}}$$

$$\Delta w_{h_jm} = \alpha\, e_{h_j}(t)\,\frac{\partial h_j(t)}{\partial w_{h_jm}}$$

finally the weight matrices are updated for the reset r, update z gates, the output unit y_k and for the hidden state h with:

$$
\begin{aligned}
\Delta w_{r_j m} &= \alpha \, e_{h_j}(t) \left(\frac{\partial h_j(t-1)}{\partial w_{r_j m}} \left(1 - z_j(t)\right) \right. \\
&\qquad \left. + z_j(t) \sigma_h'(x_{hj}(t)) w_{h_j m} h_m(t-1)^2 \sigma_r'(x_{rt}(t)) \right) \\
\Delta w_{z_j m} &= \alpha \, e_{h_j}(t) \left(\frac{\partial h_j(t-1)}{\partial w_{z_j m}} \left(1 - z_j(t)\right) \right. \\
&\qquad \left. + \left(\sigma_h(x_{hj}(t)) - h_j(t-1)\right) \sigma_z'(x_{zj}(t)) h_m(t-1) \right) \\
\Delta w_{h_j m} &= \alpha \, e_{h_j}(t) \left(\frac{\partial h_j(t-1)}{\partial w_{h_j m}} \left(1 - z_j(t)\right) \right. \\
&\qquad \left. + z_j(t) r_j(t) \sigma_h'(x_{hj}(t)) h_m(t-1) \right) \\
\Delta w_{y_k m} &= \alpha \, e_k(t) \, \sigma_k'(x_k(t)) h_m(t-1) \, .
\end{aligned}
$$

List of Figures

List of Tables

Bibliography

(Albishre et al., 2015) K. Albishre, M. Albathan, & Y. Li, 2015. Effective 20 newsgroups dataset cleaning. Dans les actes de *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, Volume 3, 98–101. IEEE.

(Arena et al., 1997) P. Arena, L. Fortuna, G. Muscato, & M. G. Xibilia, 1997. Multilayer perceptrons to approximate quaternion valued functions. *Neural Networks 10*(2), 335–342.

(Arena et al., 1994) P. Arena, L. Fortuna, L. Occhipinti, & M. G. Xibilia, 1994. Neural networks for quaternion-valued function approximation. Dans les actes de *Circuits and Systems, 1994. ISCAS'94., 1994 IEEE International Symposium on*, Volume 6, 307–310. IEEE.

(Aspragathos et Dimitros, 1998) N. A. Aspragathos & J. K. Dimitros, 1998. A comparative study of three methods for robot kinematics. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 28*(2), 135–145.

(Ba et al., 2016) J. L. Ba, J. R. Kiros, & G. E. Hinton, 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

(Babcock et al., 2002) B. Babcock, S. Babu, M. Datar, R. Motwani, & J. Widom, 2002. Models and issues in data stream systems. Dans les actes de *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 1–16. ACM.

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

(Bastien et al., 2012) F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, & Y. Bengio, 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

(Bechet et al., 2012) F. Bechet, B. Maza, N. Bigouroux, T. Bazillon, M. El-Beze, R. De Mori, & E. Arbillot, 2012. Decoda: a call-centre human-human spoken conversation corpus. LREC'12.

(Bengio, 2009) Y. Bengio, 2009. Learning deep architectures for ai. *Foundations and trends® in Machine Learning 2*(1), 1–127.

(Bengio et al., 2013a) Y. Bengio, N. Boulanger-Lewandowski, & R. Pascanu, 2013a. Advances in optimizing recurrent networks. Dans les actes de *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 8624–8628. IEEE.

(Bengio et al., 2013b) Y. Bengio, N. Boulanger-Lewandowski, & R. Pascanu, 2013b. Advances in optimizing recurrent networks. Dans les actes de *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 8624–8628. IEEE.

(Bengio et al., 2003) Y. Bengio, R. Ducharme, & P. Vincent, 2003. A neural probabilistic language model. *Journal of Machine Learning Research 3*, 1137–1155.

(Bengio et al., 2007) Y. Bengio, Y. LeCun, et al., 2007. Scaling learning algorithms towards ai. *Large-scale kernel machines,L. Bottou, O. Chapelle, D. DeCoste, J. Weston (eds), MIT Press 34*(5).

(Bengio et al., 1994) Y. Bengio, P. Simard, & P. Frasconi, 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks 5*(2), 157–166.

(Berthelot et al., 2017) D. Berthelot, T. Schumm, & L. Metz, 2017. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*.

(Bishop et al., 1995) C. M. Bishop et al., 1995. *Neural networks for pattern recognition*. Oxford university press.

(Blei et al., 2003) D. Blei, A. Ng, & M. Jordan, 2003. Latent dirichlet allocation. *The Journal of Machine Learning Research 3*, 993–1022.

(Bouallegue et al., 2014) M. Bouallegue, M. Morchid, R. Dufour, M. Driss, G. Linarès, & R. De Mori, 2014. Subspace gaussian mixture models for dialogues classification. Dans les actes de *Conference of the International Speech Communication Association (INTERSPEECH) 2014*. ISCA.

(Bouaziz et al., 2016) M. Bouaziz, M. Morchid, R. Dufour, G. Linarès, & R. De Mori, 2016. Parallel long short-term memory for multi-stream classification. Dans les actes de *2016 IEEE Spoken Language Technology Workshop (SLT)*, 218–223. IEEE.

(Bouchekif et al., 2013) A. Bouchekif, G. Damnati, & D. Charlet, 2013. Complementarity of lexical cohesion and speaker role information for story segmentation of french tv broadcast news. Dans les actes de *Statistical Language and Speech Processing*, 51–61. Springer.

(Bowman et al., 2015) S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, & S. Bengio, 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

(Bredin et al., 2014) H. Bredin, A. Laurent, A. Sarkar, V.-B. Le, S. Rosset, & C. Barras, 2014. Person instance graphs for named speaker identification in tv broadcast. Dans les actes de *Proceedings of Odyssey*.

(Chan et Lane, 2015) W. Chan & I. Lane, 2015. Deep recurrent neural networks for acoustic modelling. *arXiv preprint arXiv:1504.01482*.

(Chao et al., 2015) L. Chao, J. Tao, M. Yang, Y. Li, & Z. Wen, 2015. Long short term memory recurrent neural network based multimodal dimensional emotion recognition. Dans les actes de *Proceedings of the 5th International Workshop on Audio/Visual Emotion Challenge*, 65–72. ACM.

(Chase Gaudet, 2017) A. M. Chase Gaudet, 2017. Deep quaternion networks. *arXiv preprint arXiv:1712.04604v2*.

(Che et al., 2016) T. Che, Y. Li, A. P. Jacob, Y. Bengio, & W. Li, 2016. Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136*.

(Chen et al., 2014) G. Chen, C. Parada, & G. Heigold, 2014. Small-footprint keyword spotting using deep neural networks. Dans les actes de *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4087–4091.

(Chen, 2017) M. Chen, 2017. Minimalrnn: Toward more interpretable and trainable recurrent neural networks. *arXiv preprint arXiv:1711.06788*.

(Chen et Jin, 2015) S. Chen & Q. Jin, 2015. Multi-modal dimensional emotion recognition using recurrent neural networks. Dans les actes de *Proceedings of the 5th International Workshop on Audio/Visual Emotion Challenge*, 49–56. ACM.

(Chen et al., 2017) Y. Chen, J. Yang, & J. Qian, 2017. Recurrent neural network for facial landmark detection. *Neurocomputing 219*, 26–38.

(Chihi, 1998) I. Chihi, 1998. Understanding kohonen networks. *ENSI, National School for Computer Sciences*, 1–10.

(Cho et al., 2014) K. Cho, B. Van Merriënboer, D. Bahdanau, & Y. Bengio, 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

(Chollet, 2015) F. Chollet, 2015. keras. https://github.com/fchollet/keras.

(Chung et al., 2014) J. Chung, C. Gulcehre, K. Cho, & Y. Bengio, 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

(Creswell et al., 2018) A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, & A. A. Bharath, 2018. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine 35*(1), 53–65.

(Dai et al., 2004) B.-R. Dai, J.-W. Huang, M.-Y. Yeh, & M.-S. Chen, 2004. Clustering on demand for multiple data streams. Dans les actes de *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*, 367–370. IEEE.

(Danihelka et al., 2016) I. Danihelka, G. Wayne, B. Uria, N. Kalchbrenner, & A. Graves, 2016. Associative long short-term memory. *arXiv preprint arXiv:1602.03032*.

(Davis et Mermelstein, 1990) S. B. Davis & P. Mermelstein, 1990. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. Dans les actes de *Readings in speech recognition*, 65–74. Elsevier.

(Donahue et Rumshisky, 2018) D. Donahue & A. Rumshisky, 2018. Adversarial text generation without reinforcement learning. *arXiv preprint arXiv:1810.06640*.

(Dong Yu, 2011) M. S. Dong Yu, 2011. Improved bottleneck features using pretrained deep neural networks. Dans les actes de *Conference of the International Speech Communication Association (INTERSPEECH)*. International Speech Communication Association.

(Eisenstein et Barzilay, 2008) J. Eisenstein & R. Barzilay, 2008. Bayesian unsupervised topic segmentation. Dans les actes de *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 334–343. ACL.

(Elman, 1990) J. L. Elman, 1990. Finding structure in time. *Cognitive science 14*(2), 179–211.

(Esteve et al., 2015) Y. Esteve, M. Bouallegue, C. Lailler, M. Morchid, R. Dufour, G. Linarès, D. Matrouf, & R. De Mori, 2015. Integration of word and semantic features for theme identification in telephone conversations. Dans les actes de *Natural Language Dialog Systems and Intelligent Assistants*, 223–231. Springer.

(Fan et al., 2014) Y. Fan, Y. Qian, F.-L. Xie, & F. K. Soong, 2014. Tts synthesis with bidirectional lstm based recurrent neural networks. Dans les actes de *Interspeech*, 1964–1968.

(Fernandez et al., 2014) R. Fernandez, A. Rendel, B. Ramabhadran, & R. Hoory, 2014. Prosody contour prediction with long short-term memory, bi-directional, deep recurrent neural networks. Dans les actes de *Interspeech*, 2268–2272.

(Fox et Roberts, 2012) C. W. Fox & S. J. Roberts, 2012. A tutorial on variational bayesian inference. *Artificial intelligence review 38*(2), 85–95.

(Furui, 1986) S. Furui, 1986. Speaker-independent isolated word recognition based on emphasized spectral dynamics. Dans les actes de *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'86.*, Volume 11, 1991–1994. IEEE.

(Gajecki, 2014) L. Gajecki, 2014. Architectures of neural networks applied for lvcsr language modeling. *Neurocomputing 133*, 46–53.

(Gal et Ghahramani, 2015) Y. Gal & Z. Ghahramani, 2015. On modern deep learning and variational inference. Dans les actes de *Advances in Approximate Bayesian Inference workshop, NIPS*, Volume 2.

(Garnier-Rizet et al., 2008) M. Garnier-Rizet, G. Adda, F. Cailliau, J. Gauvain, S. Guillemin-Lanne, L. Lamel, S. Vanni, & C. Waast-Richard, 2008. Callsurf-automatic transcription, indexing and structuration of call center conversational speech for knowledge extraction and query by content. Dans les actes de *Proceedings of LREC*.

(Garofolo et al., 1993) J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, & D. S. Pallett, 1993. Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1. *NASA STI/Recon technical report n 93*.

(Gers et al., 2001) F. A. Gers, D. Eck, & J. Schmidhuber, 2001. Applying lstm to time series predictable through time-window approaches. Dans les actes de *Artificial Neural Networks at ICANN 2001*, 669–676. Springer.

(Glorot et Bengio, 2010) X. Glorot & Y. Bengio, 2010. Understanding the difficulty of training deep feedforward neural networks. Dans les actes de *International conference on artificial intelligence and statistics*, 249–256.

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

(Goodfellow et al., 2014) I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, & Y. Bengio, 2014. Generative adversarial nets. Dans les actes de *Advances in neural information processing systems*, 2672–2680.

(Graves, 2012) A. Graves, 2012. Neural networks. Dans les actes de *Supervised Sequence Labelling with Recurrent Neural Networks*, 15–35. Springer.

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

(Graves et al., 2007) A. Graves, S. Fernández, & J. Schmidhuber, 2007. Multi-dimensional recurrent neural networks. Dans les actes de *Artificial Neural Networks - ICANN 2007, 17th International Conference, Porto, Portugal, September 9-13, 2007, Proceedings, Part I*, 549–558.

(Graves et al., 2013a) A. Graves, N. Jaitly, & A.-r. Mohamed, 2013a. Hybrid speech recognition with deep bidirectional lstm. Dans les actes de *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, 273–278. IEEE.

(Graves et al., 2013b) A. Graves, A.-r. Mohamed, & G. Hinton, 2013b. Speech recognition with deep recurrent neural networks. Dans les actes de *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, 6645–6649. IEEE.

(Graves et Schmidhuber, 2005) A. Graves & J. Schmidhuber, 2005. Framewise phoneme classification with bidirectional lstm networks. Dans les actes de *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, Volume 4, 2047–2052. IEEE.

(Graves et Schmidhuber, 2009) A. Graves & J. Schmidhuber, 2009. Offline handwriting recognition with multidimensional recurrent neural networks. Dans D. Koller, D. Schuurmans, Y. Bengio, & L. Bottou (Eds.), *Advances in Neural Information Processing Systems 21*, 545–552. Curran Associates, Inc.

(Greff et al., 2017) K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, & J. Schmidhuber, 2017. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*.

(Gregor et al., 2015) K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, & D. Wierstra, 2015. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*.

(Grèzl et al., 2007) F. Grèzl, M. Karafiàt, S. Kontàr, & J. Cernocky, 2007. Probabilistic and bottle-neck features for lvcsr of meetings. Dans les actes de *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IV–757. IEEE.

(Haykin et Network, 2004) S. Haykin & N. Network, 2004. A comprehensive foundation. *Neural Networks 2*.

(Hazen, 2011) T. Hazen, 2011. Topic identification. *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*, 319–356.

(He et al., 2015) K. He, X. Zhang, S. Ren, & J. Sun, 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. Dans les actes de *Proceedings of the IEEE international conference on computer vision*, 1026–1034.

(He et al., 2016) K. He, X. Zhang, S. Ren, & J. Sun, 2016. Deep residual learning for image recognition. Dans les actes de *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

(Hinton et al., 2006) G. E. Hinton, S. Osindero, & Y.-W. Teh, 2006. A fast learning algorithm for deep belief nets. *Neural computation 18*(7), 1527–1554.

(Hinton et Salakhutdinov, 2006) G. E. Hinton & R. R. Salakhutdinov, 2006. Reducing the dimensionality of data with neural networks. *Science 313*(5786), 504–507.

(Hirose et Yoshida, 2012) A. Hirose & S. Yoshida, 2012. Generalization characteristics of complex-valued feedforward neural networks in relation to signal coherence. *IEEE Transactions on Neural Networks and learning systems 23*(4), 541–551.

(Hochreiter, 1998) S. Hochreiter, 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6*(02), 107–116.

(Hochreiter et Schmidhuber, 1997) S. Hochreiter & J. Schmidhuber, 1997. Long short-term memory. *Neural computation 9*(8), 1735–1780.

(Hu et Wang, 2012) J. Hu & J. Wang, 2012. Global stability of complex-valued recurrent neural networks with time-delays. *IEEE Transactions on Neural Networks and Learning Systems 23*(6), 853–865.

(Huang et al., 2016) M. Huang, Y. Cao, & C. Dong, 2016. Modeling rich contexts for sentiment classification with lstm. *CoRR abs/1605.01478*.

(Isokawa et al., 2003) T. Isokawa, T. Kusakabe, N. Matsui, & F. Peper, 2003. Quaternion neural network and its application. Dans les actes de *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, 318–324. Springer.

(Isokawa et al., 2009) T. Isokawa, N. Matsui, & H. Nishimura, 2009. Quaternionic neural networks: Fundamental properties and applications. *Complex-Valued Neural Networks: Utilizing High-Dimensional Parameters*, 411–439.

(Janod et al., 2016) K. Janod, M. Morchid, R. Dufour, G. Linarès, & R. De Mori, 2016. Deep stacked autoencoders for spoken language understanding. Dans les actes de *Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA.

(Janod et al., 2017) K. Janod, M. Morchid, R. Dufour, G. Linares, & R. De Mori, 2017. Denoised bottleneck features from deep autoencoders for telephone conversation analysis. *IEEE/ACM Transactions on Audio, Speech, and Language Processing 25*(9), 1809–1820.

(Janod et al., 2016) K. Janod, M. Morchid, R. Dufour, G. Linarès, & R. D. Mori, 2016. Deep stacked autoencoders for spoken language understanding. Dans les actes de *Interspeech 2016*, 720–724.

(Kim et al., 2017) T. Kim, M. Cha, H. Kim, J. K. Lee, & J. Kim, 2017. Learning to discover cross-domain relations with generative adversarial networks. Dans les actes de *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1857–1865. JMLR. org.

(Kingma et Ba, 2014) D. Kingma & J. Ba, 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

(Kingma et Welling, 2013) D. P. Kingma & M. Welling, 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

(Kohonen, 1982) T. Kohonen, 1982. Self-organized formation of topologically correct feature maps. *Biological cybernetics 43*(1), 59–69.

(Kulkarni et al., 2015) T. D. Kulkarni, W. F. Whitney, P. Kohli, & J. Tenenbaum, 2015. Deep convolutional inverse graphics network. Dans les actes de *Advances in neural information processing systems*, 2539–2547.

(Lagus et Kuusisto, 2002) K. Lagus & J. Kuusisto, 2002. Topic identification in natural language dialogues using neural networks. Dans les actes de *Proceedings of the Third SIGdial Workshop on Discourse and Dialogue*, Philadelphia, Pennsylvania, USA, 95–102. Association for Computational Linguistics.

(Larsen et al., 2015) A. B. L. Larsen, S. K. Sønderby, H. Larochelle, & O. Winther, 2015. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*.

(LeCun et al., 2015) Y. LeCun, Y. Bengio, & G. Hinton, 2015. Deep learning. *Nature 521*(7553), 436–444.

(LeCun et al., 1989) Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, & L. D. Jackel, 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation 1*(4), 541–551.

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

(Li et al., 2017) J. Li, W. Monroe, T. Shi, S. Jean, A. Ritter, & D. Jurafsky, 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

(Ma et al., 2019) C.-Y. Ma, M.-H. Chen, Z. Kira, & G. AlRegib, 2019. Ts-lstm and temporal-inception: Exploiting spatiotemporal dynamics for activity recognition. *Signal Processing: Image Communication 71*, 76–87.

(Makhzani et al., 2015) A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, & B. Frey, 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

(Matsui et al., 2004) N. Matsui, T. Isokawa, H. Kusamichi, F. Peper, & H. Nishimura, 2004. Quaternion neural network with geometrical operators. *Journal of Intelligent & Fuzzy Systems 15*(3, 4), 149–164.

(McClelland et al., 1986) J. L. McClelland, D. E. Rumelhart, P. R. Group, et al., 1986. Parallel distributed processing. *Explorations in the Microstructure of Cognition 2*, 216–271.

(McCulloch et Pitts, 1943) W. S. McCulloch & W. Pitts, 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics 5*(4), 115–133.

(Medsker et Jain, 2001) L. R. Medsker & L. J. Jain, 2001. Recurrent neural networks. *Design and Applications 5*.

(Melamed et Gilbert, 2011) I. Melamed & M. Gilbert, 2011. Speech analytics. *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*, 397–416.

(Mikolov, 2012) T. Mikolov, 2012. Statistical language models based on neural networks. *Presentation at Google, Mountain View, 2nd April*.

(Minemoto et al., 2017) T. Minemoto, T. Isokawa, H. Nishimura, & N. Matsui, 2017. Feed forward neural network with random quaternionic neurons. *Signal Processing 136*, 59–68.

(Mirza et Osindero, 2014) M. Mirza & S. Osindero, 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

(Mohri et al., 2002) M. Mohri, F. Pereira, & M. Riley, 2002. Weighted finite-state transducers in speech recognition. *Computer Speech and Language 16*(1), 69 – 88.

(Morchid, 2018) M. Morchid, 2018. Parsimonious memory unit for recurrent neural networks with application to natural language processing. *Neurocomputing 314*, 48–64.

(Morchid et al., 2015a) M. Morchid, M. Bouallegue, R. Dufour, G. Linarès, D. Matrouf, & R. De Mori, 2015a. Compact multiview representation of documents based on the total variability space. *IEEE/ACM Transactions on Audio, Speech, and Language Processing 23*(8), 1295–1308.

(Morchid et al., 2014a) M. Morchid, R. Dufour, M. Bouallegue, G. Linarès, & R. De Mori, 2014a. Theme identification in human-human conversations with features from specific speaker type hidden spaces. Dans les actes de *Conference of the International Speech Communication Association (INTERSPEECH) 2014*. ISCA.

(Morchid et al., 2014b) M. Morchid, R. Dufour, P.-M. Bousquet, M. Bouallegue, G. Linarès, & R. De Mori, 2014b. Improving dialogue classification using a topic space representation and a gaussian classifier based on the decision rule. Dans les actes de *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 126–130. IEEE.

(Morchid et al., 2014c) M. Morchid, R. Dufour, P.-M. Bousquet, M. Bouallegue, G. Linarès, & R. De Mori, 2014c. Improving dialogue classification using a topic space representation and a gaussian classifier based on the decision rule. Dans les actes de *International Conference on Acoustic, Speech and Signal Processing (ICASSP) 2014*. IEEE.

(Morchid et al., 2015b) M. Morchid, R. Dufour, & G. Linarès, 2015b. Topic-space based setup of a neural network for theme identification of highly imperfect transcriptions. Dans les actes de *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*, 346–352. IEEE.

(Morchid et al., 2016) M. Morchid, R. Dufour, & G. Linarès, 2016. Impact of word error rate on theme identification task of highly imperfect human–human conversations. *Computer Speech & Language 38*, 68–85.

(Morchid et al., 2013) M. Morchid, G. Linarès, M. El-Beze, & R. De Mori, 2013. Theme identification in telephone service conversations using quaternions of speech features. Dans les actes de *Conference of the International Speech Communication Association (INTERSPEECH) 2013*. ISCA.

(Nasr et al., 2002) G. E. Nasr, E. Badr, & C. Joun, 2002. Cross entropy error function in neural networks: Forecasting gasoline demand. Dans les actes de *FLAIRS Conference*, 381–384.

(Nasrabadi et Feng, 1988) N. M. Nasrabadi & Y. Feng, 1988. Vector quantization of images based upon the kohonen self-organizing feature maps. Dans les actes de *Proc. IEEE Int. Conf. Neural Networks*, Volume 1, 101–105.

(Nguyen et al., 2017) A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, & J. Yosinski, 2017. Plug & play generative networks: Conditional iterative generation of images in latent space. Dans les actes de *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4467–4477.

(Nitta, 1995) T. Nitta, 1995. A quaternary version of the back-propagation algorithm. Dans les actes de *Proceedings of the IEEE International Conference on Neural Networks*, Volume 5, 2753–2756.

(Odena, 2016) A. Odena, 2016. Semi-supervised learning with generative adversarial networks. *arXiv preprint arXiv:1606.01583*.

(Parcollet et al., 2016) T. Parcollet, M. Morchid, P.-M. Bousquet, R. Dufour, G. Linarès, & R. De Mori, 2016. Quaternion neural networks for spoken language understanding. Dans les actes de *Spoken Language Technology Workshop (SLT), 2016 IEEE*, 362–368. IEEE.

(Parcollet et al., 2017) T. Parcollet, M. Morchid, & G. Linarès, 2017. Deep quaternion neural networks for spoken language understanding. Dans les actes de *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*, 504–511. IEEE.

(Parcollet et al., 2019) T. Parcollet, M. Morchid, & G. Linarès, 2019. Quaternion convolutional neural networks for heterogeneous image processing. Dans les actes de *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP.*, Volume 44, 1–6. IEEE.

(Parcollet et al., 2019) T. Parcollet, M. Morchid, G. Linarès, & R. De Mori, 2019. Bidirectional quaternion long-short term memory recurrent neural networks for speech recognition. Dans les actes de *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP.*, Volume 44, 1–6. IEEE.

(Parcollet et al., 2018a) T. Parcollet, M. Ravanelli, M. Morchid, G. Linarès, & R. De Mori, 2018a. Speech recognition with quaternion neural networks. *IRASL@NIPS*.

(Parcollet et al., 2018b) T. Parcollet, M. Ravanelli, M. Morchid, G. Linarès, C. Trabelsi, R. De Mori, & Y. Bengio, 2018b. Quaternion recurrent neural networks. *arXiv preprint arXiv:1806.04418*.

(Parcollet et al., 2018c) T. Parcollet, M. Ravanelli, M. Morchid, G. Linarès, C. Trabelsi, R. D. Mori, & Y. Bengio, 2018c. Quaternion recurrent neural networks.

(Parcollet et al., 2018d) T. Parcollet, Y. Zhang, M. Morchid, C. Trabelsi, G. Linarès, R. de Mori, & Y. Bengio, 2018d. Quaternion convolutional neural networks for end-to-end automatic speech recognition. Dans les actes de *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018.*, 22–26.

(Pearlmutter, 1989) B. A. Pearlmutter, 1989. Learning state space trajectories in recurrent neural networks. *Neural Computation 1*(2), 263–269.

(Pedregosa et al., 2011) F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., 2011. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research 12*, 2825–2830.

(Pei et Cheng, 1999) S.-C. Pei & C.-M. Cheng, 1999. Color image processing by using binary quaternion-moment-preserving thresholding technique. *Image Processing, IEEE Transactions on 8*(5), 614–628.

(Povey et al., 2011) D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, & K. Vesely, 2011. The kaldi speech recognition toolkit. Dans les actes de *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society. IEEE Catalog No.: CFP11SRW-USB.

(Purver, 2011) M. Purver, 2011. Topic segmentation. *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*, 291–317.

(Radford et al., 2015) A. Radford, L. Metz, & S. Chintala, 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

(Raina et al., 2009) R. Raina, A. Madhavan, & A. Y. Ng, 2009. Large-scale deep unsupervised learning using graphics processors. Dans les actes de *Proceedings of the 26th annual international conference on machine learning*, 873–880. ACM.

(Rajeswar et al., 2017) S. Rajeswar, S. Subramanian, F. Dutil, C. Pal, & A. Courville, 2017. Adversarial generation of natural language. *arXiv preprint arXiv:1705.10929*.

(Ravanelli et al., 2018a) M. Ravanelli, P. Brakel, M. Omologo, & Y. Bengio, 2018a. Light gated recurrent units for speech recognition. *IEEE Transactions on Emerging Topics in Computational Intelligence 2*(2), 92–102.

(Ravanelli et al., 2018b) M. Ravanelli, T. Parcollet, & Y. Bengio, 2018b. The pytorch-kaldi speech recognition toolkit. *arXiv preprint arXiv:1811.07453*.

(Reed et al., 2016) S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, & H. Lee, 2016. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*.

(Rosenblatt, 1958) F. Rosenblatt, 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review 65*(6), 386.

(Rousseau et al., 2014) A. Rousseau, G. Boulianne, P. Deléglise, Y. Estève, V. Gupta, & S. Meignier, 2014. Lium and crim asr system combination for the repere evaluation campaign. Dans les actes de *International Conference on Text, Speech, and Dialogue*, 441–448. Springer.

(Rumelhart et MacClelland, 1986) D. Rumelhart & J. MacClelland, 1986. Learning internal representations by error backpropagation. chapter 8 from parallel distributed processing. vol. 1: Foundations.

(Rumelhart et al., 1985) D. E. Rumelhart, G. E. Hinton, & R. J. Williams, 1985. Learning internal representations by error propagation. Rapport technique, California Univ San Diego La Jolla Inst for Cognitive Science.

(Rumelhart et al., 1986) D. E. Rumelhart, J. L. McClelland, P. R. Group, et al., 1986. Parallel distributed processing. *Explorations in the Microstructure of Cognition 1*, 216–271.

(Sabour et al., 2017) S. Sabour, N. Frosst, & G. E. Hinton, 2017. Dynamic routing between capsules. *arXiv preprint arXiv:1710.09829v2*.

(Salimans et al., 2016) T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, & X. Chen, 2016. Improved techniques for training gans. Dans les actes de *Advances in neural information processing systems*, 2234–2242.

(Sangwine, 1996) S. J. Sangwine, 1996. Fourier transforms of colour images using quaternion or hypercomplex, numbers. *Electronics letters 32*(21), 1979–1980.

(Schuster, 1999) M. Schuster, 1999. On supervised learning from sequential data with applications for speech recognition. *Daktaro disertacija, Nara Institute of Science and Technology*.

(Schuster et Paliwal, 1997) M. Schuster & K. K. Paliwal, 1997. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on 45*(11), 2673–2681.

(Severyn et Moschitti, 2015) A. Severyn & A. Moschitti, 2015. Twitter sentiment analysis with deep convolutional neural networks. Dans les actes de *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 959–962. ACM.

(Singh et al., 2016) B. Singh, T. K. Marks, M. Jones, O. Tuzel, & M. Shao, 2016. A multi-stream bi-directional recurrent neural network for fine-grained action detection. Dans les actes de *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1961–1970.

(Siniscalchi et al., 2013) S. M. Siniscalchi, D. Yu, L. Deng, & C.-H. Lee, 2013. Exploiting deep neural networks for detection-based speech recognition. *Neurocomputing 106*, 148–157.

(Song et Yam, 1998) J. Song & Y. Yam, 1998. Complex recurrent neural network for computing the inverse and pseudo-inverse of the complex matrix. *Applied mathematics and computation 93*(2-3), 195–205.

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

(Srivastava et al., 2013) N. Srivastava, R. Salakhutdinov, & G. Hinton, 2013. Modeling documents with a deep boltzmann machine. Dans les actes de *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, 616–624. AUAI Press.

(Stolcke et al., 2002) A. Stolcke et al., 2002. Srilm-an extensible language modeling toolkit. Dans les actes de *INTERSPEECH*, Volume 2002, 2002.

(Stollenga et al., 2015) M. F. Stollenga, W. Byeon, M. Liwicki, & J. Schmidhuber, 2015. Parallel multi-dimensional lstm, with application to fast biomedical volumetric image segmentation. Dans les actes de *Advances in neural information processing systems*, 2998–3006.

(Sundermeyer et al., 2012) M. Sundermeyer, R. Schlüter, & H. Ney, 2012. Lstm neural networks for language modeling. Dans les actes de *INTERSPEECH*, 194–197.

(Sutskever et al., 2014) I. Sutskever, O. Vinyals, & Q. V. Le, 2014. Sequence to sequence learning with neural networks. Dans les actes de *Advances in neural information processing systems*, 3104–3112.

(Titouan et al., 2019a) P. Titouan, M. Morchid, X. Bost, & G. Linares, 2019a. M2h-gan: A gan-based mapping from machine to human transcripts for speech understanding. *Proc. Interspeech 2019*, 3325–3328.

(Titouan et al., 2017) P. Titouan, M. Morchid, & G. Linarès, 2017. Quaternion denoising encoder-decoder for theme identification of telephone conversations. *Proc. Interspeech 2017*, 3325–3328.

(Titouan et al., 2019b) P. Titouan, M. Morchid, G. Linares, & R. De Mori, 2019b. Real to h-space encoder for speech recognition. *Proc. Interspeech 2019*, 3325–3328.

(Titouan et al., 2018) P. Titouan, Z. Ying, M. Mohamed, T. Chiheb, L. Georges, D. M. Renato, & B. Yoshua, 2018. Quaternion convolutional neural networks for end-to-end automatic speech recognition. *arXiv preprint arXiv:1806.07789*.

(Tokuda et al., 2003) K. Tokuda, H. Zen, & T. Kitamura, 2003. Trajectory modeling based on hmms with the explicit relationship between static and dynamic features. Dans les actes de *Eighth European Conference on Speech Communication and Technology*.

(Trabelsi et al., 2017) C. Trabelsi, O. Bilaniuk, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, & C. J. Pal, 2017. Deep complex networks. *arXiv preprint arXiv:1705.09792*.

(Tur et De Mori, 2011) G. Tur & R. De Mori, 2011. *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons.

(Turian et al., 2010) J. Turian, L. Ratinov, & Y. Bengio, 2010. Word representations: a simple and general method for semi-supervised learning. Dans les actes de *Proceedings of the 48th annual meeting of the association for computational linguistics*, 384–394. Association for Computational Linguistics.

(Tygert et al., 2016) M. Tygert, J. Bruna, S. Chintala, Y. LeCun, S. Piantino, & A. Szlam, 2016. A mathematical motivation for complex-valued convolutional networks. *Neural computation 28*(5), 815–825.

(Ulyanov et al., 2018) D. Ulyanov, A. Vedaldi, & V. Lempitsky, 2018. It takes (only) two: Adversarial generator-encoder networks. Dans les actes de *Thirty-Second AAAI Conference on Artificial Intelligence*.

(Van Asch, 2013) V. Van Asch, 2013. Macro-and micro-averaged evaluation measures [[basic draft]].

(Vincent et al., 2010) P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, & P.-A. Manzagol, 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research 11*, 3371–3408.

(Vinyals et al., 2015) O. Vinyals, A. Toshev, S. Bengio, & D. Erhan, 2015. Show and tell: A neural image caption generator. Dans les actes de *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3156–3164.

(Warde-Farley et Bengio, 2016) D. Warde-Farley & Y. Bengio, 2016. Improving generative adversarial networks with denoising feature matching.

(Werbos, 1974) P. Werbos, 1974. Beyond regression:" new tools for prediction and analysis in the behavioral sciences. *Ph. D. dissertation, Harvard University*.

(Wisdom et al., 2016) S. Wisdom, T. Powers, J. Hershey, J. Le Roux, & L. Atlas, 2016. Full-capacity unitary recurrent neural networks. Dans les actes de *Advances in Neural Information Processing Systems*, 4880–4888.

(Wu et al., 2017a) L. Wu, Y. Xia, L. Zhao, F. Tian, T. Qin, J. Lai, & T.-Y. Liu, 2017a. Adversarial neural machine translation. *arXiv preprint arXiv:1704.06933*.

(Wu et al., 2017b) Y. Wu, M. Yuan, S. Dong, L. Lin, & Y. Liu, 2017b. Remaining useful life estimation of engineered systems using vanilla lstm neural networks. *Neurocomputing*.

(Wu et al., 2016) Z. Wu, Y.-G. Jiang, X. Wang, H. Ye, & X. Xue, 2016. Multi-stream multi-class fusion of deep networks for video classification. Dans les actes de *Proceedings of the 24th ACM international conference on Multimedia*, 791–800. ACM.

(Xu et al., 2017) D. Xu, L. Zhang, & H. Zhang, 2017. Learning alogrithms in quaternion neural networks using ghr calculus. *Neural Network World 27*(3), 271.

(Xu et al., 2013) X. Xu, L. Lu, P. He, Z. Pan, & L. Chen, 2013. Improving constrained clustering via swarm intelligence. *Neurocomputing 116*, 317–325.

(Yang et al., 2017) Z. Yang, W. Chen, F. Wang, & B. Xu, 2017. Improving neural machine translation with conditional sequence generative adversarial nets. *arXiv preprint arXiv:1703.04887*.

(Yao et al., 2015) K. Yao, T. Cohn, K. Vylomova, K. Duh, & C. Dyer, 2015. Depth-gated recurrent neural networks. *arXiv preprint*.

(Zaremba, 2015) W. Zaremba, 2015. An empirical exploration of recurrent network architectures. *Journal of Machine Learning Research*.

(Zhang et al., 2015) X. Zhang, J. Zhao, & Y. LeCun, 2015. Character-level convolutional networks for text classification. Dans les actes de *Advances in neural information processing systems*, 649–657.

(Zhou et al., 2016) G.-B. Zhou, J. Wu, C.-L. Zhang, & Z.-H. Zhou, 2016. Minimal gated unit for recurrent neural networks. *International Journal of Automation and Computing 13*(3), 226–234.

(Zhu et al., 2016) J.-Y. Zhu, P. Krähenbühl, E. Shechtman, & A. A. Efros, 2016. Generative visual manipulation on the natural image manifold. Dans les actes de *European Conference on Computer Vision*, 597–613. Springer.

(Zhu et al., 2017) J.-Y. Zhu, T. Park, P. Isola, & A. A. Efros, 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. Dans les actes de *Proceedings of the IEEE International Conference on Computer Vision*, 2223–2232.