People's Democratic Republic of Algeria

Ministry for Higher Education and Scientific Research

University of Ghardaia

Faculty of Science and Technology

Laboratory of Mathematics and Applied Sciences

Department of Mathematics and Computer Science

Master Thesis

*Presented to obtain the **Master diploma** in Computer Science*

Specialty: Intelligent Systems for Knowledge Extraction

Theme

Sequence Models: Deep Learning Approach

_ Case Speech Recognition _

Presented by:

DJEBRIT Aida

RAHMANI Maroua

Jury members:

M. OULAD NAOUI Slimane	MCB	Univ.Ghardaia	President
M. ADJILA Abderrahmane	MAA	Univ.Ghardaia	Examiner
M. BOUHANI Abdelkader	MAA	Univ.Ghardaia	Examiner
M. BELLAOUAR Slimane	MCB	Univ.Ghardaia	Supervisor

University Year: 2018/2019

Abstract

Our era is characterized by the existence of huge amount of data. Perhaps sequence data is among the important data types. It is used, mainly, in bioinformatics and natural language processing applications. Consequently, a great deal of research has been devoted to sequence data.

Sequence modelling is used to analyze intelligently sequence data. Recent studies use deep learning approach to ameliorate the performance of sequence modelling. The present thesis deals with speech recognition systems. Hence, we process audio data as sequences.

We first study in general artificial neural networks, and in particular recurrent neural networks (RNN). RNN are able to handle audio data in efficient way.

To make the studied theoretical concept in practice. We conduct experimental study on English speech using the deepSpeech2 architecture with LibriSpeech data set.

Although the limited hardware environment, the result (character error rate=27%) reveal that DeepSpeech2 perform well with audio data especially if we use more sophisticated hardware environment and if we tune the hyper parameter of the system.

Key words : Artificial Neural Networks, Deep Learning, Recurrent Neural Networks, Sequence Models, Speech Recognition.

Résumé

Notre époque est caractérisée par l'existence d'une énorme quantité de données. Peut-être que les données de séquence font partie des types de données importants. Ils sont principalement utilisé dans les applications de la bio-informatique et du traitement du langage naturel. En conséquence, Un grand nombre de chercheurs ont été consacrées aux données de séquence.

La modélisation de séquence est utilisée pour analyser intelligemment les données séquentielles. Des études récentes utilisent une approche d'apprentissage en profondeur pour améliorer les performances de la modélisation de séquences. Le présente mémoire traite des systèmes de reconnaissance vocale. Par conséquent, nous traitons les données audio sous forme de séquences.

Nous avons d'abord étudié les Réseaux de Neurone Artificiel en général, et les Réseaux Neurone Récurrents (RNNs) en particulier. Les RNNs sont capables de gérer les données audio de manière efficace. Pour réaliser les concepts de l'étude théorique en pratique. Nous faisons une étude expérimentale sur des discours anglais en utilisant l'architecture deepSpeech2 avec le dataset LibriSpeech.

Malgré que l'environnement matériel est limité, le résultat (taux d'erreur sur les caractères = 27%) révèle que DeepSpeech2 fonctionne bien avec les données audio, en particulier si nous utilisons un environnement matériel plus sophistiqué et si nous ajustons les hyper paramètres du système.

Mots clés : Réseau de Neurones Artificiels, Apprentissage en Profondeur, Réseaux de Neurones Récurrents, Modèles de Séquences, Reconnaissance Vocale.

ملخص

يتميز عصرنا بوجود كمية هائلة من البيانات. ربما تكون بيانات التسلسل من بين أنواع البيانات المهمة. يتم استخدامها بشكل أساسي في المعلوماتية الحيوية و تطبيقات معالجة اللغة الطبيعية. وبالتالي، تم تخصيص قدر كبير من البحث في البيانات المتسلسلة.

يتم استخدام نمذجة التسلسل لتحليل البيانات المتسلسلة بذكاء. تستخدم الدراسات الحديثة نهج التعلم العميق لتحسين أداء نمذجة التسلسل. تتناول المذكرة الحالية أنظمة التعرف على الكلام. وبالتالي نعالج البيانات الصوتية كتسلسلات.

بدأنا الدراسة في الشبكة العصبية الاصطناعية على العموم، وخاصة الشبكة العصبية المتكررة(RNN) .

الشبكة العصبية المتكررة قادرة على التعامل مع البيانات الصوتية بطريقة فعالة.

لتطبيق الدراسة النظرية, نجري دراسة تجريبية على خطاب اللغة الإنجليزية باستخدام بنية DeepSpeech2 مع مجموعة بيانات LibriSpeech .

على الرغم من أن بيئة الأجهزة محدودة ، إلا أن النتيجة (معدل خطأ الحرف = 27٪) تكشف أن DeepSpeech2 يعمل بشكل جيد مع بيانات الصوت خاصةً إذا استخدمنا بيئة أكثر تطوراً.

الكلمات المفتاحية : الشبكات العصبية الاصطناعية، التعلم العميق، الشبكات العصبية المتكررة، نمادج التسلسل، التعرف على الكلام.

Table of Contents

List of Figures

Dedication

To *the special man who spent his life in the cause of my happiness and made of his eyes a lamp that illuminates my path , my father "Ismail".*

T*o the special women who care about me and give me*

everything I asked about, I ask Allah to give her the happiness and the long life, my mother "Mani".

T*o my soul and my cause of happiness who encourage me all the time, I ask Allah to give her the happiness and the long life, my mother "Fatima".*

T*o the man who tired for our comfort, who gives me everything I need it, I ask Allah to give him the long life, my father "Messaoud".*

T*o My dear brothers: Chams eddin, Salim, Omar and Haithem.*

T*o my dear friends.*

T*o all family "DJEBRIT".*

DJEBRIT Aida

Dedication

I dedicate this work to

My Dears parents who help me and give the support whole my life

My brothers Bayane & Salmane

My sisters Kaouthar & Ghofrane and my love Iktimel which I wish them a lot of success in their life

All RAHMANI & FARADJI family

All my friends and my colleagues

All my friends in ELMENIAA

<u>All those who love me & all those I love</u>

<u>RAHMANI Maroua</u>

Thanks

We thank

ALLAH who gives us the help to arrive to this day.

Our parents for their support us throughout our study.

Our supervisor BELAOUAR Slimane for his helps and advices to complete this modest memoir.

Our teacher ADJILA Abderrahmane for his help.

Our jurors who have followed this work.

Special thanks to our teacher OULED NAOUI Slimane.

All our teachers from the primary school to the university.

Who helped us in this work even those who prayer for us.

General Introduction

Recent years have been stamped by the phenomenon of data explosion and the diversity of their nature which makes it difficult for humans to deal with it manually or in traditional ways.
Sequential data is a complex type of data that requires special and expensive processing, which has caused the thought of creating high-performance intelligent models.

Sequence modeling is used to analyze sequence data, it has an attention in recent years, specialty in medical domain and natural language processing.
Speech recognition is handled with the sequence modeling because of its sequential nature.
The speech recognition model has a long history. The Hidden Markov Model (HMM) is the most used model for long time usually with Gaussian Mixture Model (GMM) (Rodríguez et al. [1997]), until the emergence of deep learning which replace the GMM with Deep Neural Network(DNN) as hybrid DNN-HMM (Hinton et al. [2012]), Convolutional Neural Network(CNN) as hybrid CNN-HMM (Abdel-Hamid et al. [2014]) and Recurrent Neural Network (RNN) as hybrid RNN-HMM (Gao and Glowacka [2019]).
Recently, deep learning offers a new pipeline named end-to-end (Graves and Jaitly [2014]) based on deep neural network that eliminate the need of HMM model and ameliorate the performance of speech recognition.
In order to develop an Automatic Speech Recognition (ASR) system, we conduct an experimentation on end to end model (DeepSpeech2,DS2) with the LibriSpeech dataset of read English speech.
The performance of the trained model achieves a character error rate (CER) of 27%.
This thesis is composed of four chapters :
The first chapter introduces Artificial Neural Networks and their development over times and present the way of their work and their architectures.
The second chapter presents the deep learning approach and its important in modeling different kind of data according to its architecture which can be Convolutional Neural Network (CNN) or Recurent Neural Network (RNN).
The third chapter illustrates sequence models in general and speech recognition in particular. A bibliographic study is presented.
The final chapter presents the experiment conducted on a speech recognition system with deep learning using deepSpeech2.

Finally, the conclusion to sum up all what we have seen.

Chapitre 1

Neural Networks

1.1 Introduction

Neural network (NN) is one of the fields of artificial intelligence. This field is used to simulate the way that human brain works. The idea behind NN came from the ability of man to distinguish between things. A small child can identify a group of images after viewing it while the machine is unable to do this simple process, which is complex even for super computer which is able to process huge amount of data. In this sense, scientists thought to try to imitate the way of human mind work, and found that neural cell is the most important part of the formation of the brain and from its collection we have a network of cells. From this perspective, studies began on the mechanism of biological neural networks to simulate their work on the computer to solve complex problems such as pattern recognition which need a complex architecture like neural networks to be solved.

In this chapter we introduce the basic concepts to understand NN. We start by a brief history of NN. Thereafter, we try to describe the link between biological and artificial NN. The rest of chapter is dedicated to some concepts as activation function, learning process and architecture of NN. Finally, we show the strengths and limits of NN.

1.2 History

The neural network has passed by many developments since 1940s. In this period NN encountered phases of ups and downs (Touzet [1992]).

1943 : J. Mc Culloch and W. Pitts modelling a biological neuron, they proof that a simple formal neuron can be realized as logic, arithmetic and symbolic complex functions (McCulloch and Pitts [1943]).

1949 : The American Psychologist D. Hebb wrote The Organization of Behavior book, which refers to the basic concept of human learning stating that neural pathways enhance their connection whenever they are used together (Hebb [1949]).

1958 : F. Rosenblatt developed a perceptron model which is the first artificial system capable of learning from experience (Rosenblatt [1958]). B. Widrow and Hoff also introduced the Adaline (ADAptive LINEAR) model (Widrow and Hoff [1960]), which is the basic model for multilayered networks.

1969-1972 : This period was characterized by a decline in the evolution of neural networks. Due to the criticism presented by M. Minsky and S. Papert in 1969 (Minsky and Papert [1969]) about the characteristics of perceptron, which had a negative influence in this field until T. Kohonen presented his work in 1972 on new associative memories founded on a correlation matrix and proposes applications to pattern recognition (Kohonen [1972]).

1982 : The physician Hopfeld, who relaunched the interest in artificial neural networks, through his article, which provides a theory about the work and possibilities of neural networks (Hopfield [1982b]).

1985 : The backpropagation model (Rumelhart et al. [1985]) was developed for the multilayer perceptron by three groups of researchers, which is a method that allows to learn the parameters of model in very efficient way.

1989 : Development of a model called convolutional neural network which was specialized for image analyzing (LeCun et al. [1995]).

1990-1994 : Start of application of CNN and backpropagation models. However, to data sets in real world but they didn't perform well.

1995 : Appearance of Long short-term memory (LSTM) technique which is a very key technology for analyzing data that is a function of time (Audio, music...).

1998-2005 : After the emergence of backpropagation, CNN and LSTM, the scientists tried to use neural networks in the real world again but the networks did not work well.

2010 : Renamed deep learning, because of the increasing in the number of layers, it was at maximum 3 layers and became 10 or more (Arel et al. [2010]).

2013 : The emergence of the deep neural network is a combination of CNN and GPU (graphics processing unit) which is a developed for game community using ImageNet which is dataset of million images.

2015 : The big companies (Google, Facebook,...) have recall artificial neural network experts to develop new applications with high efficiency. AlphaGo [1] relied on CNN to improve the performance of game Go. It achieved results that exceeded accuracy and speed than humans.

1. https ://deepmind.com/research/alphago/

1.3 Biological Neural Networks

The human brain has a billions of interconnected neurons. Each neuron is composed of three main elements as shown in Figure 1.1 :

— Dendrites : input canals which receive signals from previous neurons.

— Body (Soma) : contains the kernel (Nucleus) which is responsible for the conduct of vital cellular activates.

— Axon : send signals into the synapse (link between two neurons).

The transmission of information in the neural network has unidirectional way from dendrites to axon. Each neuron receives inputs or signals from other neurons that are transmitted by the dendrites. At the level of the cell body (Soma), the neuron analyzes and processes these signals by summing them. If the result obtained is greater than the threshold of activation so the neuron is active and inactive in other side.

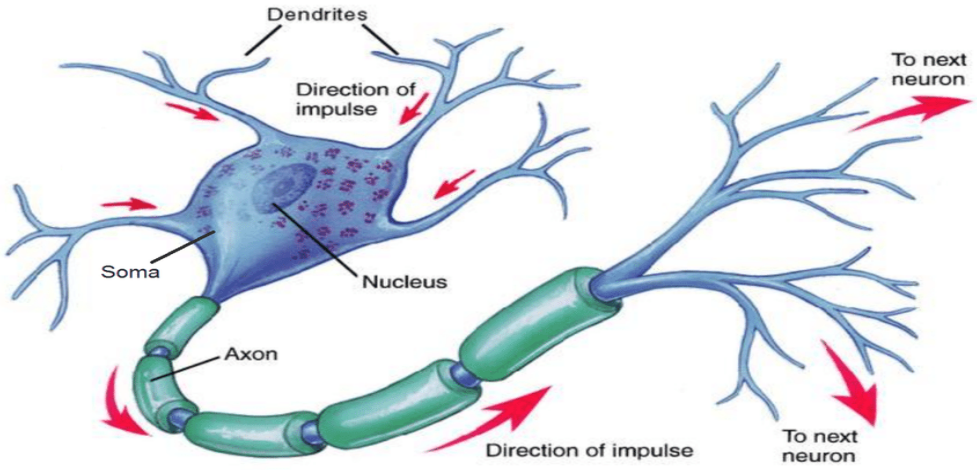


FIGURE 1.1: Biological Neuron (Jimenez Romero [2019]).

From the above described mechanism, scientists (McCulloch and Pitts [1943]) have modeled a mathematical model that artificially performs the same function of biological neuron. This is the basic element of **an artificial neural network**.

1.4 Artificial Neural Network

Artificial neural network (ANN) is an artificial representation of biological neural network. Its architecture can be represented as a directed graph that has a number of neurons interconnected forming many layers (Figure 1.2).

— Input layer : always only one layer, pass the input data to the hidden layers.

— Hidden layers : can have none, one or multiple layers. Process data that come from input layer.

— Output layer : only one layer too, uses the information in hidden layers or directly from input layer, finally it produces a final result.

As illustration, Figure 1.2 represents a simple ANN which is formed by 4 layers interconnected. Input layer with 3 neurons, two hidden layer with 4 neurons in each one, and an output layer with one neuron.

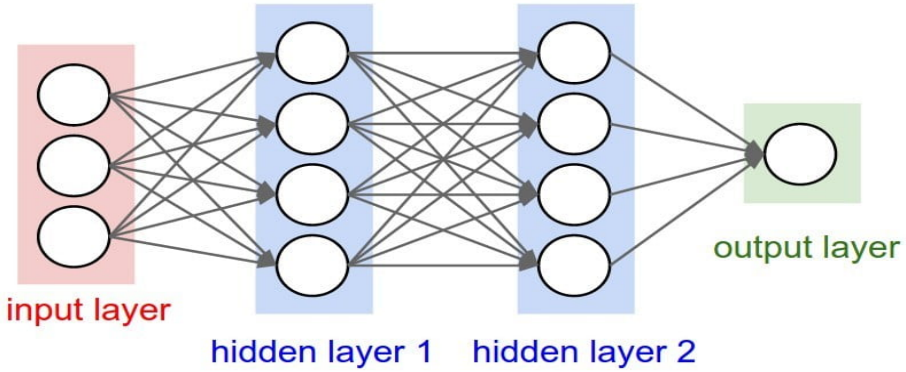


FIGURE 1.2: Artificia thel Neural Networks

A neural network without hidden layers is similar to a logistic regression.

Each artificial neuron (Figure 1.3) is an elementary processor which receives input as vector $X = [x_1, \dots x_n]$ and the correspond weights $W = [w_1, \dots, w_n]$ represent the force of interconnection between neurons (n is numbers of inputs), and then calculates the sum of the weighted inputs added to a bias (b) which is the input that makes the network flexible. Then the result is passed by an activation function (f) with a threshold to limit the response to arrive at desired output y (Equation 1.1).

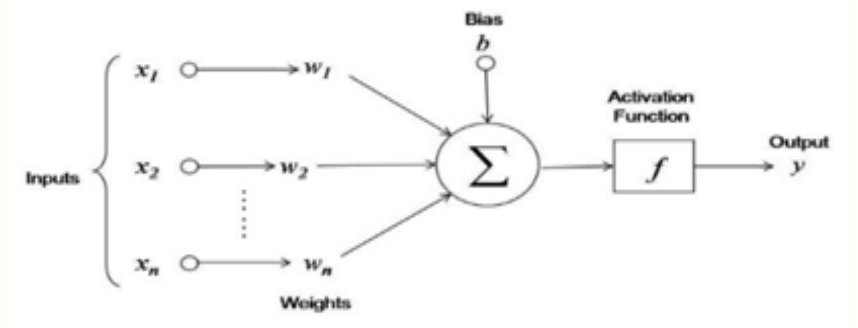


FIGURE 1.3: Artificial Neuron

$$Y_{output} = f(\sum_{i=1}^{n}(x_i * w_i) + b) \tag{1.1}$$

That can be represented in its vectored from as in equation :

$$Y_{output} = f(W^T X + b) \tag{1.2}$$

1.5 Activation Function

The activation function or transfer function is a mathematical function. It is important for the behavior of the neuron. It decides which neuron to fire. This function should be nonlinear. It has two parameters the weighted sum of inputs and the bias. If the activation functions are linear, then the network is equivalent to linear regression model.

In the literature, there are many types of nonlinear activation functions (sigmoid, ReLu, tanh, softmax,...) (Nwankpa et al. [2018]). The choice of the function is according to the given problem.

1.5.1 Sigmoid Function

It is the oldest and the most popular activation function. It limits the values to vary between 0 and 1. If the values are very large positive number returned 1, else if the values are very large negative returned 0. It is commonly used for binary classification. It can be defined as follows :

$$f(x) = \frac{1}{1 + e^{-x}} \tag{1.3}$$

And its derivative is as follow :

$$f^{'}(x) = f(x)(1 - f(x)) \tag{1.4}$$

- Advantages
  - It makes clear predictions.
  - Normalize the output values.
- Disadvantages
  - Vanishing gradient : An increase or decrease in the values does not change the prediction.
  - The values of outputs are not centralized in 0.
  - Computationally expensive because it uses exponential function.

the sigmoid function graph is as follow :

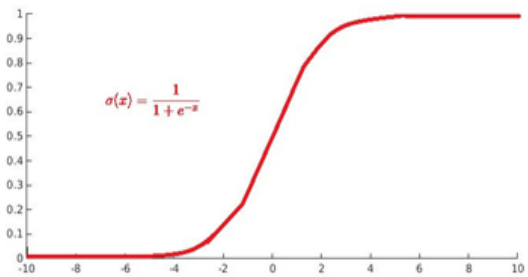


FIGURE 1.4: Sigmoid function

1.5.2 Tangent Hyperbolic (tanh)

Tangent Hyperbolic is similar to the sigmoid function, except that its output vary from -1 to 1 (Figure 1.5). Its symmetry gives better result than the sigmoid function. It is used usually in hidden layers. Tangent Hyperbolic is given by :

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{1.5}$$

Its derivative is as follow :

$$f'(x) = 1 - f(x)^2 \tag{1.6}$$

- Advantage
  - besides the advantages of sigmoid function, the output values are centralized in 0.
- Disadvantages
  - Vanishing gradient : An increase or decrease in the values does not change the prediction.
  - Computationally expensive because it uses exponential function.

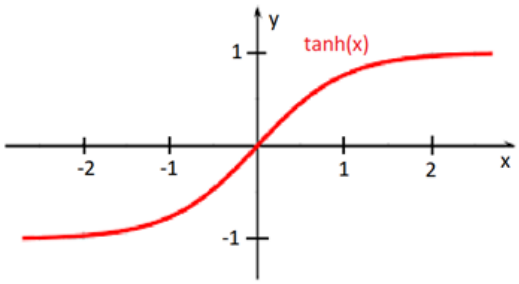


FIGURE 1.5: Tangent Hyperbolic function

1.5.3 Rectified Linear Unit (ReLu)

The ReLu function (Figure 1.6) is the most used activation function. It solves the problems of the two previous functions (sigmoid and tanh). If the input values are negative returns 0, otherwise output remains as input. ReLu is defined as follow :

$$f(x) = \max(0, x) \tag{1.7}$$

or

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \tag{1.8}$$

Its derivative is as follow :

$$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases} \tag{1.9}$$

- Advantage
  - It allows the network to converge very quickly.

— Disadvantages
  — The Dying ReLu problem : when the inputs are negative, the neuron stay inactive and the weighs keep not changed (the gradients become 0) and therefore the network cannot learn and cannot perform the backpropagation

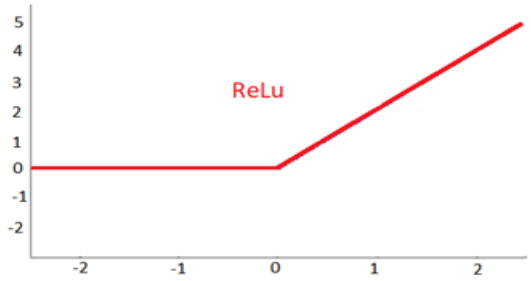


FIGURE 1.6: ReLu function

1.5.4 Softmax

Softmax function is usually used (but not only) for classification tasks. It allows building neural networks with several standardized outputs by given probabilities. Used in multiple classifications, k is the number of classes. Softmax function is given by :

$$f(x_i) = \frac{e^{x_i}}{\sum_{c=1}^{k} e^{x_c}} \tag{1.10}$$

k = number of classes
  — Advantages
    — Able to handle multi classes.
    — Useful for output neurons (output layers).

1.6 ANN Learning

Learning is the most important phase in the construction of an artificial neural network model to acquire knowledge about the environment. There are two main classes for training a neural network which are supervised learning and unsupervised learning.

1.6.1 Supervised learning

The scientists studied the issue of supervised learning in neural networks in the past 30 years. Applied the learning algorithm called “back propagation” .Which was applied in many problems, including converting texts to audios and diagnosing diseases and others.

The training of ANN in supervised learning is presented the input to the networks which will produce an output. This output is compared with the desired output. A line signal is created if the difference is large, for this the weights are adjusted until the difference is

too small or the desired output is reached.It has different rule such as the Error correction learning rule which is a technique that comparing the obtained output to the desired output value, the error used to direct the training.

1.6.2 Unsupervised learning

In this learning framework the network, itself must discover the patterns features from the input data and the relation for the input data over the output. The input vectors of similar type are combined to form clusters. When a new input pattern is applied, the neural network gives an output response by indicating the class to which input pattern belongs. Therefore, there is no feedback and the framework cannot know what should be the desired output and whether it is correct or incorrect.

1.6.3 Learning Process

The learning process is an iterative process of going and return (Forward propagation and backpropagation of the information) with calculation of the loss. as shown in Figure 1.7

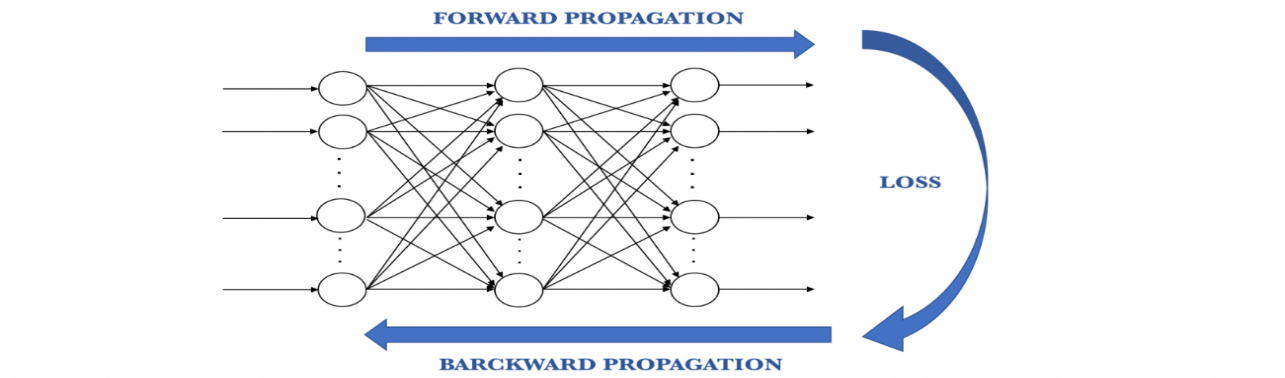


FIGURE 1.7: General view of learning process

In order to illustrate the learning process, we use supervised learning with error correction learning rule. The process is as follows (Figure 1.8) :

1. Random initialization of the network parameters (w_{ij} weights and b_j biases).
2. Pass with a set of input data over the network to obtain their prediction output (forward propagation).
3. Comparison between desired and labeled output then calculate their loss (using the loss function).
4. Propagate the value of the loss to all the parameters (bias and weights) of the model (backpropagation).
5. Using the propagated information to update the parameters of NN (bias, weights) with the gradient descent in order to minimize the loss.
6. Repeat frequently the previous steps until we obtain a model that perform well.

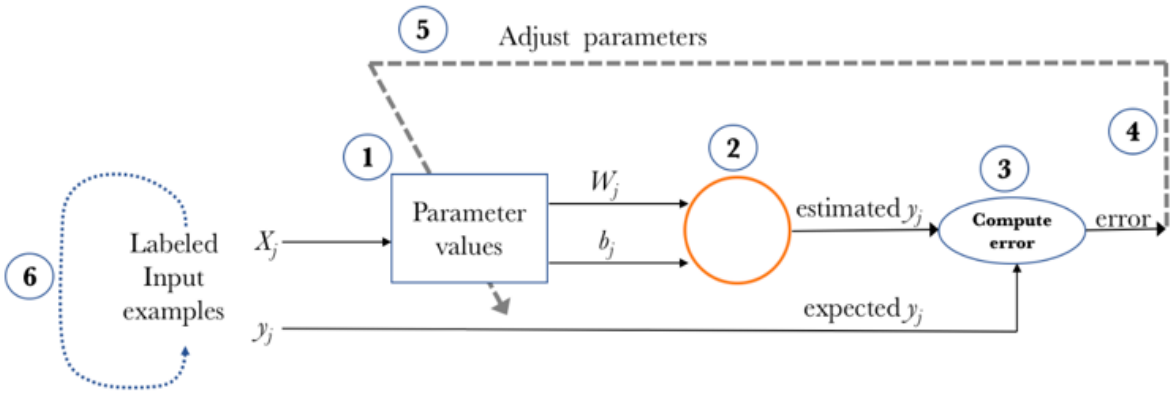


FIGURE 1.8: The steps of learning process

Forward propagation : The input x provides the initial information that is propagate to the hidden units at each layer and finally produce the output y.

Loss function : It applied to a single training example. The choice of a loss function is depending to the given predictive modeling problems (Regression, Binary classification . . .). In general, we are given the training dataset $(x^1; y^1), \ldots, (x^m; y^m)$ where m is the number of examples and the output $\hat{y} = f(W^T X + b)$ then the loss function became :

$$L(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y})) \tag{1.11}$$

The cost function is the cost of the parameters w and b. there is many types of cost function. Each one has a special case of utilization. For example Binary Cross Entropy used for binary classification is given by :

$$J(w, b) = \frac{1}{m} \sum_{i=1}^{m} [L(\hat{y}^{(i)}, y^{(i)})] = \frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})] \tag{1.12}$$

Back propagation : Browse the networks from the outputs to the inputs to modify all the parameters that build the networks. It looks to define the parameters w and b that minimize the cost function by computing the gradient of cost function as follow :

$$w = w - \alpha \frac{\partial J(w, b)}{\partial w} \tag{1.13}$$

$$b = b - \alpha \frac{\partial J(w, b)}{\partial b} \tag{1.14}$$

Repeat this calculation using partial derivatives until update all the parameters that construct the network.

The parameter α is learning rate that controls the size of the step of the gradient descent. It is used to accelerate the learning process.

1.6.4 Variance and bias

Variance and bias are statistical concepts that are important for all types of machine learning and therefore neural networks namely for the evaluation of the performance the model. Before we define the bias and variance, we should introduce some basic concepts.

As we know, to build a model we should have a dataset (examples). Split it into 3 parts. Which are training set, test set and validation set (dev set).

— The training set : a set of data used as input in-order to train our model.
— Test set : is used at the end of the training process to evaluate the performance of our model.
— Validation set : a set of data from the training set, used to validate the model during training, it helps to give information for adjusted the hyper parameters (Number of hidden units, learning rate) of the model.
— Bias : It is a type of error results from difference between the output predicted by our model and the value which we are trying to predict.
— Variance : It is a type of error that results from the difference between validation set error and training set error.

1.6.5 Over-fitting and under-fitting Problem

Over fitting and under fitting are two most popular problems in machine learning method including the neural network (Figure 1.9).

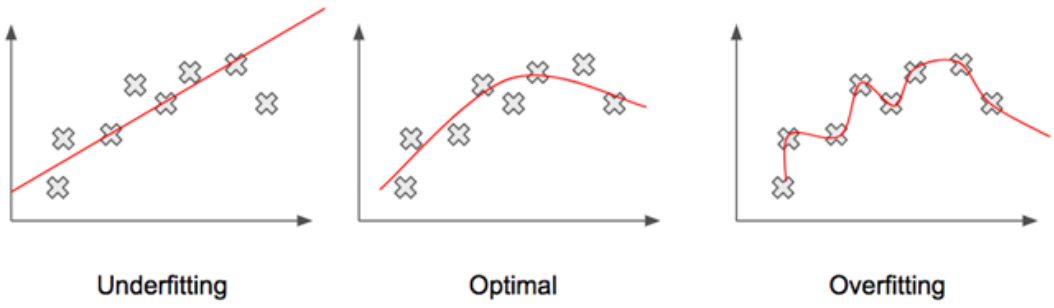


FIGURE 1.9: Over-fitting Problem , Optimal Case and under-fitting Problem

Over-fitting Problem

The over fitting happened when the network model gives a good result with training set, but bad result with the test set (generalization problem).

Its Symptoms are :

— Low bias : good predictions for the training set.
— High variance : bad predictions for the test set.

Reducing Over-fitting

Over-fitting is one of problems encountered the phase of learning neural network. To reduce the impact of this problem we make reference to regularization to produce a more general model (Stutz [2019]).

Regularization

There are many type of regularization such as :

— **Dropout :** It is a regularization technique working on dropping a set of neurons selected randomly during each iteration of gradient descent to avoid the over-fitting problem (Figure 1.10).

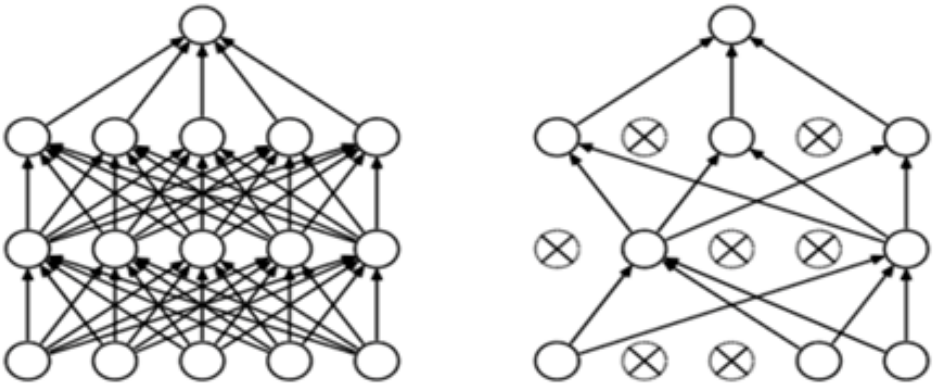


FIGURE 1.10: Dropout

Every neuron is kept with probability p and dropped with probability $1 - p$. the values of p may be different for each layers, usually $p = 0.5$ in hidden layer and $p = 0$ in output layer (no dropout).

— **Early Stopping :** Stop the training as soon as the error of validation set reaches a minimum before the error increases again.

— **Weight Sharing :** The idea consists to the different units in the same layer sharing an identical weight in order to reduce the complexity of the network and may be incorporated the prior knowledge into the network architecture.

— **Batch Normalization :** batch normalization is a technic of regularization that consist of normlizing the input data of the network and between each layer. It used to acclerate training in the deep artificial neural network.

Under-fitting Problem

The network model cannot work well with the training set and even the test set.

Its Symptoms are :

— High Bias : bad predictions for the train set.

— High Variance : bad predictions for The test set.

Reducing Under-fitting

— **Increase the complexity of the model :** It consists to add more neurons in each layer or more layers in the model.

— **Add more feature :** add more training set examples to help the model for better classification.

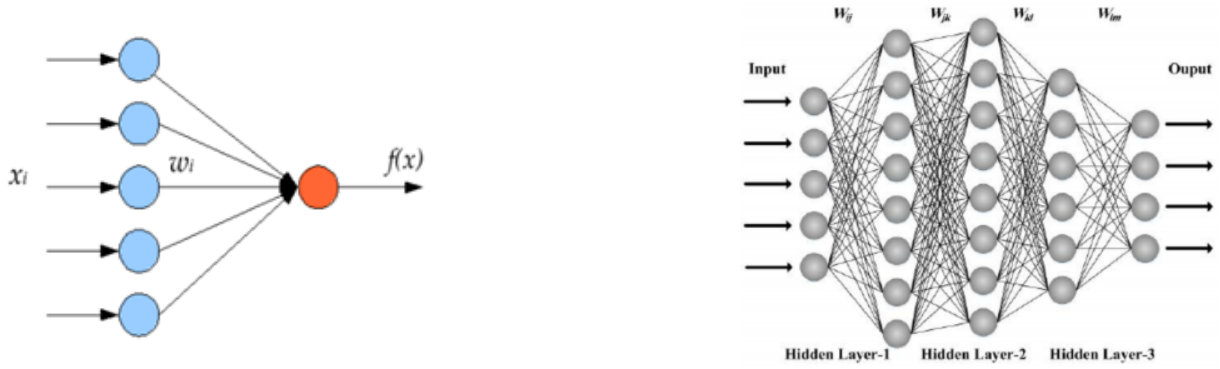
— **Reduce dropout**.

1.7 Architecture of Neural Networks

1.7.1 Feed-Forward Networks

In Feed-forward architecture signals travel one way only, from input to output. Don't allow feedback (loops), i.e. the output of any layer does not affect that same layer. They are extensively used in pattern recognition. This type of architecture is also known as bottom-up or top-down (Figure 1.11) .

Perceptron : is the oldest and the easiest network, contains two types, perceptron with one layer (single layer perceptron) and the second is with many layers (multi layers perceptron MLP). Generally it is used in classification.



(a) simple networks

(b) multi layers perceptron

FIGURE 1.11: Feed-Forward network

1.7.2 Feed-Back Networks

The feedback networks have a signal traveling in both directions by inserting loops in the networks. They are very powerful and can be much complicated. These architecture are dynamic, which is used to model dynamic systems. Also used in process control or filtering. Its

condition changes to the extent of reaching the equilibrium, and remain at equilibrium point until the input changes and then the search for a new equilibrium. Feedback architectures are structure as interactive and recurrent (Figure 1.12).

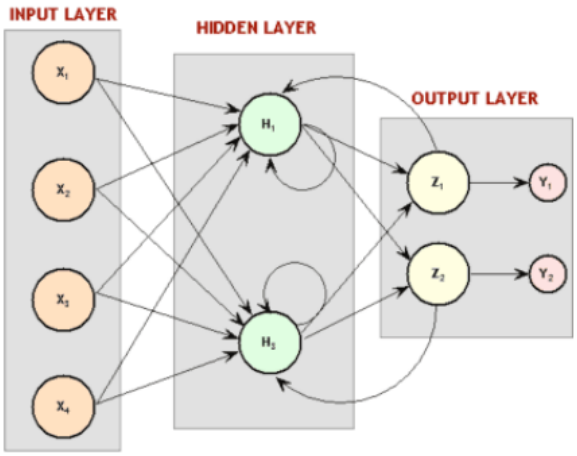


FIGURE 1.12: Feed-back network

1.8 Advantages and Disadvantages of Neural Networks

1.8.1 Advantages

— The main important things in neural networks are that they are able to adapt and solve nonlinear problems well, and the speed of execution, and they are suitable for parallel implementation.
— Robustness : they have good noise immunity.
— Transfer learning : ability to use trained network for solve other similar problems similar (classification of helicopter using trained network for classification of airplane).
— Fault tolerance : The disappearance of some parts of the information does not affect the work of the network. Also, if there is a small error in the network, it does not affect the overall work of the network.
— Learn from examples : Not like the traditional algorithms that are programmed by following an instruction block, neural networks do their work through what they learned from the lot of examples given to them.

1.8.2 Disadvantages

— Neural networks need more powerful machines : in accordance with their structure, their ability to process a large amount of information with parallel processing power.
— Network Architecture : (Black box) cannot do an interpretation for what happens on hidden layers, it is possible to repair the input and output layers but hidden layers do not know how to do this.
— Learning process :

— Can be very long.
— Backpropagation error learning can converge to local minima gives a sub-optimal solution.
— Network paralysis : The weights become so large that their modification does not affect the behavior of the neurons (Exploding Gradients problem).

1.9 Conclusion

In this chapter, we presented the basic concepts related to artificial neural networks ANNs, and we explained how they learn and their different architectures, and we show its advantages and disadvantages.

However, they still have a deficit in processing unstructured data that requires a much deeper architectures. From this point, we present the next chapter dealing with the development and diversity of neural networks in terms of depth and work.

Chapitre 2

Deep Learning

2.1 Introduction

Artificial Intelligence or AI coined by John McCarthy in 1956. AI is a big domain in computer science that looks to enable machine to mimic human behavior using logic, If-else rules. However, machine learning (ML) a sub domain of AI uses statistical methods to allow machines to learn by themselves using the provided data and make accurate predictions. ML includes also many natures inspired techniques. Deep learning (DL) is sub domain of ML based on artificial neural networks (Figure 2.1).

Deep learning architectures made revolution in AI due to their ability to process unstructured data (text, image, audio...) and large amount of data (Minar and Naher [2018]).

DL improves results in many domains such as computer vision, speech recognition and machine translation. Deep learning techniques solve many problems in many areas of the economy, health, transportation, trade, finance and energy (Alain Tapp [2019]).

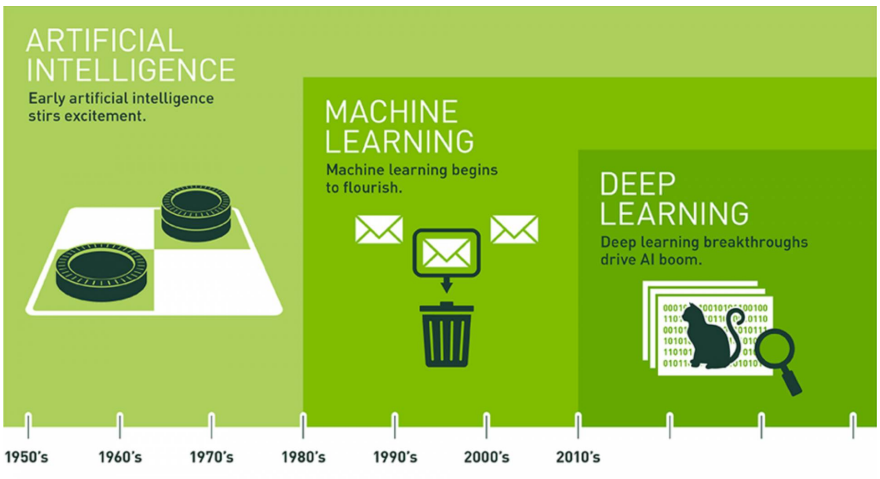


FIGURE 2.1: chronological order of artificial intelligence, machine learning and deep learning (Copeland [2016])

In this chapter, we present deep learning and why it is important their famous architectures Convolutional Neural Network and Recurrent Neural Network and finally its application domain.

2.2 Definition

Deep Learning is a machine learning method based on artificial neural networks. DL methods attempt to model high-level abstractions in data by using model architectures composed of multiple non-linear transformations (Graves et al. [2013]).

DL methods designed to handle large amounts of data by adding layers to the network. Figure 2.2 shows the deference between a simple neural network and deep learning neural network.

A deep learning model has the ability to extract features from raw data through multiple processing layers consisting of multiple linear and nonlinear transformations and learn about these features step by step through each layer with minimal human intervention (Deng and Yu [2014] ; LeCun et al. [2015]).

Deep Learning is a model that perform classification tasks directly from images, text or audio. Deep Learning models can achieve an exceptional level of accuracy, sometimes exceeding human performance. Models are trained through a large set of labeled data and neural network architectures that contain many layers.

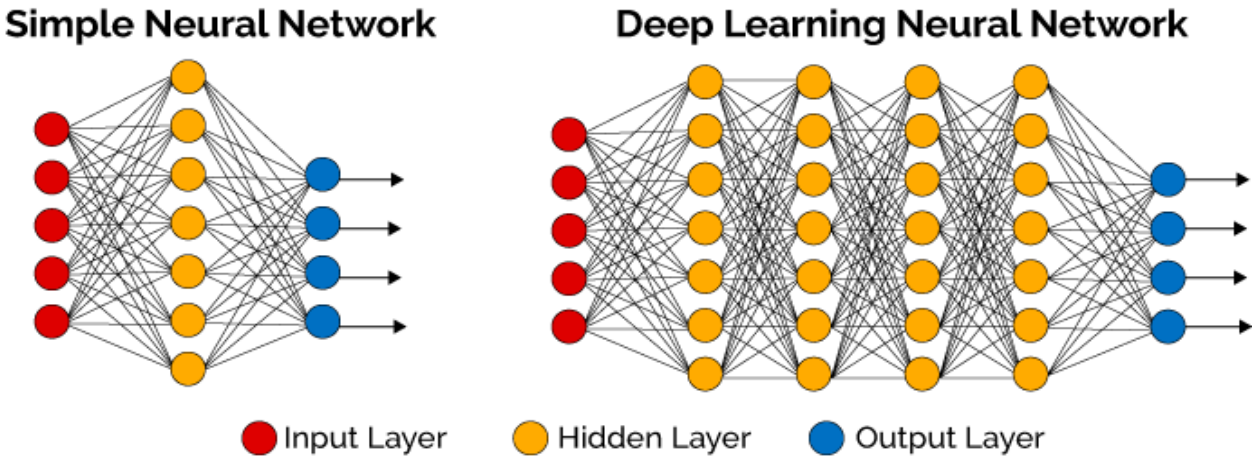


FIGURE 2.2: architecture of a simple neural network and deep neural network

2.3 Why Deep Learning ?

ML algorithms work well for a wide variety of problems. However, they have failed to solve some major AI problems such as speech recognition and object recognition. Deep learning solves those problems.

We can understand the real potential of Deep Learning only after larger amounts of data are available thanks to Big Data and the computing machines have become more powerful.

First, feature extraction in deep learning is performed automatically. Contrary to ML, where the feature extraction is performed, in general, manually which is a difficult and costly in time and it requires a domain expert. This fact is depicted in Figure 2.3.

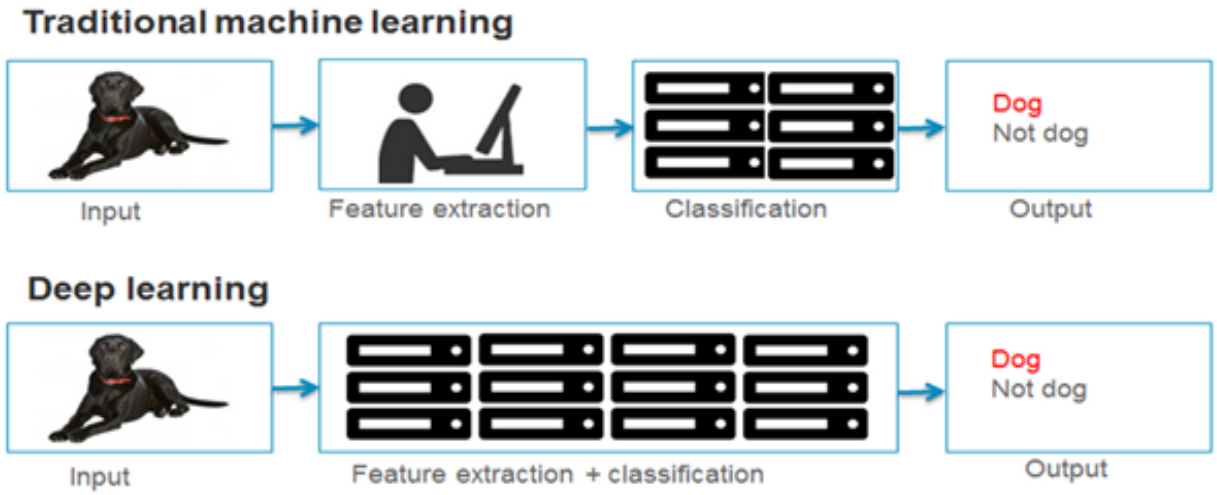


FIGURE 2.3: Difference between machine learning and deep learning

Another deference which is the big advantages that make DL popular is their power to process a big amounts of data with height performance. Contrary, traditional ML algorithms are useful for small amount of data. This fact is depicted in Figure 2.4. the Chinese scientist Andrew Ng, which is the famous researcher of DL said that, *"The analogy to deep learning is that the rocket engine is the deep learning models and the fuel is the huge amounts of data we can feed to these algorithms."*

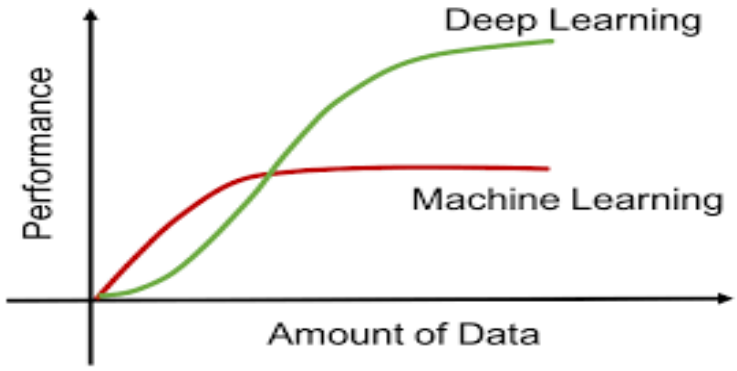


FIGURE 2.4: Performance of deep learning and machine learning by change of amount of data (Di et al. [2018])

2.4 Convolutional Neural Network (CNN)

Convolutional neural network (CNN) is a type of deep neural network inspired by human visual system. It has a feed-forward architecture introduced in 1995 by Yann LeCun and Yoshua Bengio (LeCun et al. [1995]).

CNN is a powerful tool designed to recognize visual patterns directly from pixel images with minimal preprocessing. Its strength emerged in 2012 when Alex Krizhevsky construct CNN named Alex net (Krizhevsky et al. [2012]) gives a good result by reducing error from 26% to 16% in ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [1].

After this success, many various architectures of CNNs was created more performance by increasing the number of layers (deeper) to reduce the error rate ZFNet [2], VGGNet [3], GoogLeNet [4], ResNet [5],...As shows in Figure 2.5. Moreover, CNN also reduces the number of features that's why it used for processing images unlike the traditional Neural Network.

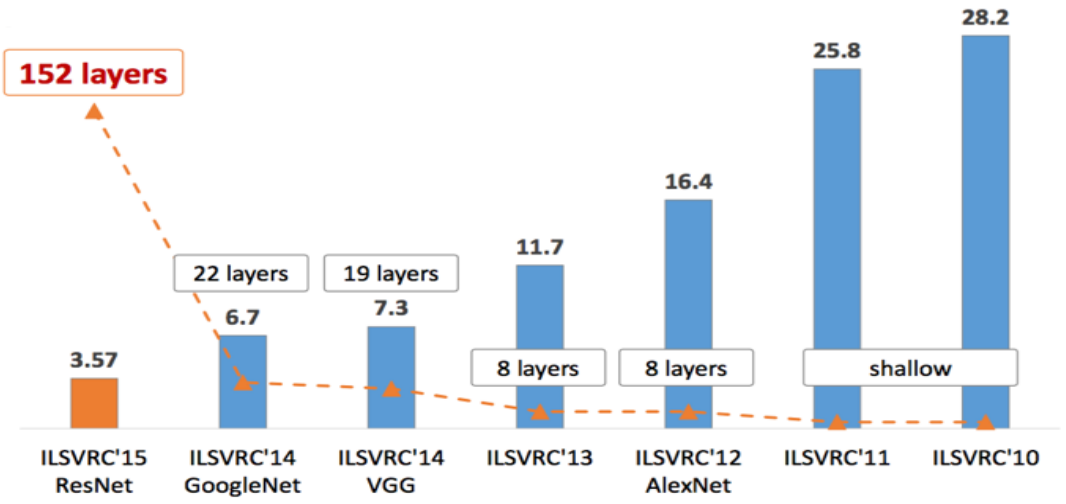


FIGURE 2.5: ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners (Russakovsky et al. [2015])

Layers of CNN

CNN has specific hidden layers which consist of convolutional, Relu, pooling layers for featrues extraction task and Flatting, fully connected layers for classification task, as shown in Figure 2.6. Convolutional and pooling are the essential parts.

1. https ://arxiv.org/abs/1409.0575
2. https ://arxiv.org/abs/1311.2901
3. https ://arxiv.org/pdf/1409.1556v6
4. https ://www.coursehero.com/file/27610887/GoogLeNetpdf/
5. https ://arxiv.org/pdf/1512.03385

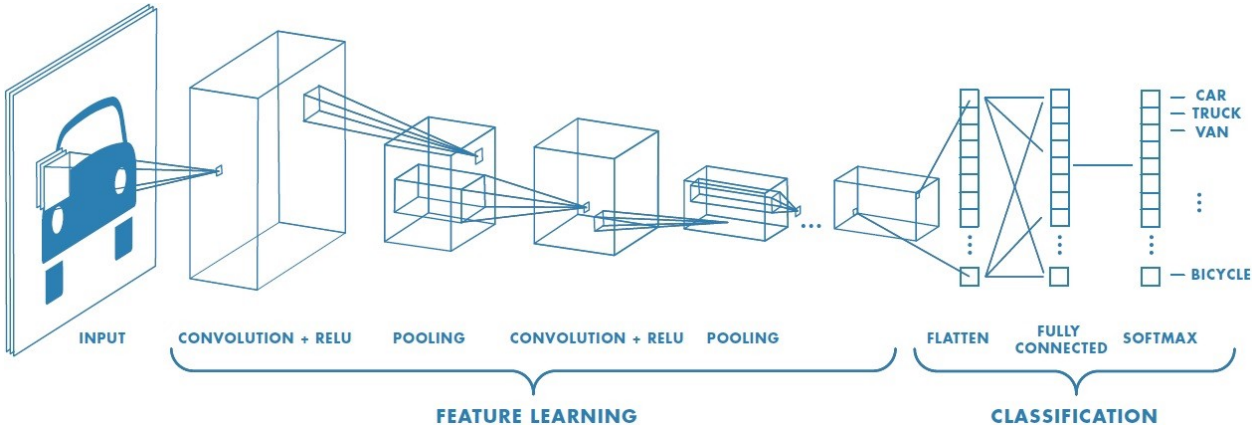


FIGURE 2.6: Convolutional Neural Networks

2.4.1 Convolutional Layer

Convolution layer is the core of CNN. It applied a convolution operation which takes as inputs image represent as a matrix of pixels and learnable kernel (filter) applied a calculation and then produce a result which is an image usually smaller. Leading to reduce the numbers of features. As shows in Figure 2.7.

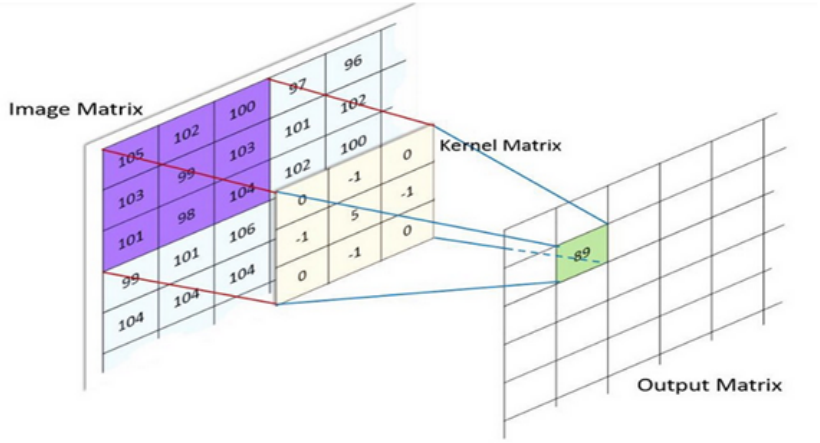


FIGURE 2.7: Convolution Layer

2.4.2 Relu Layer

ReLU layer (or nonlinear layer) applied a nonlinear function usually Relu which is an activation function replaces all negative numbers with 0, as illustrate in Figure 7. Usually it used on the result of the convolution layer to many reasons :

- Through definition of ReLu it is a function that breaks linearity by removing some of the values (all negative).
- ReLU also speeds up calculations by deleting some of the data.
- It keeps the positive values as it and makes them more prominent.

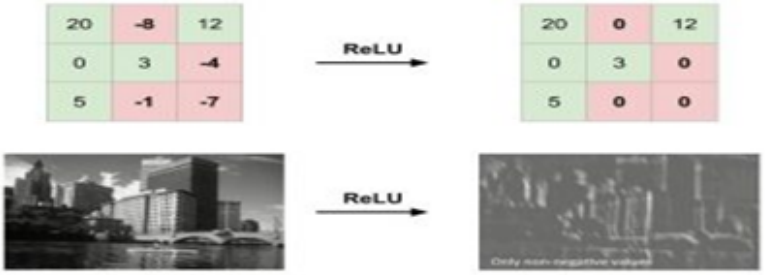


FIGURE 2.8: ReLu Layer

2.4.3 Pooling Layer

Pooling Layer is a simple operation which consist to replace a square of pixel (usually 2*2 or 3*3) by a single value, there are 3 types of pooling (Figure 2.9) :

— Max pooling : take the max value of the pixels in selected square.

— Average pooling : take the average value of the pixels in selected square.

— Sum pooling : take the sum value of the pixels in selected square.

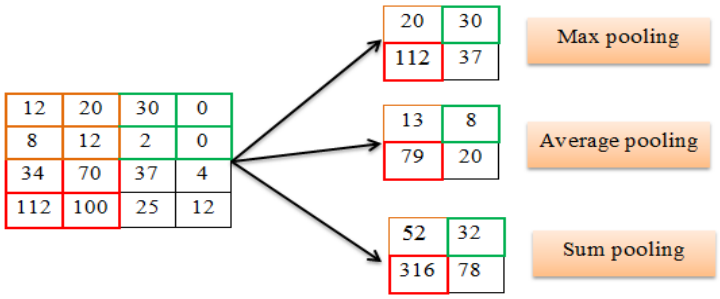


FIGURE 2.9: Pooling Layer

2.4.4 Flattening Layer

Flattening Layer is the Last step in extraction of information part, flattening is simply to put end to end all the images (matrices) that we have to make a (long) vector (Figure 2.10). The final vector is constructed by recover the pixels of images line by line.

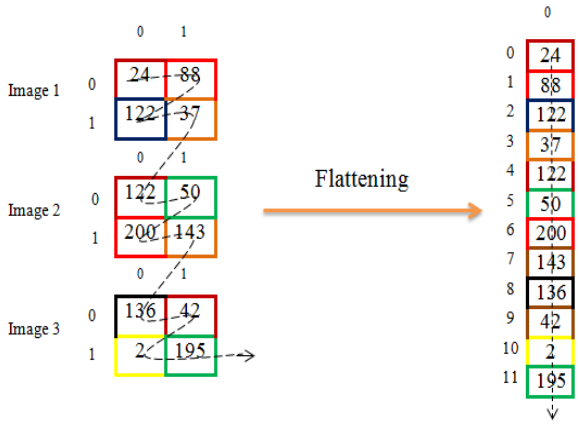


FIGURE 2.10: Flattening Layer

2.4.5 Fully Connected Layer

Fully connected layer is feed forward neural networks that uses a softmax activation function in output layers to normalize the output vector so that it represents probabilities. The Fully Connected Layers form the last few layers in the network. Fully connected means that every neuron in the next layer is connected to every individual neuron in the previous layer (Figure 2.11). The input to the fully connected layer is the output from the final flatting layer

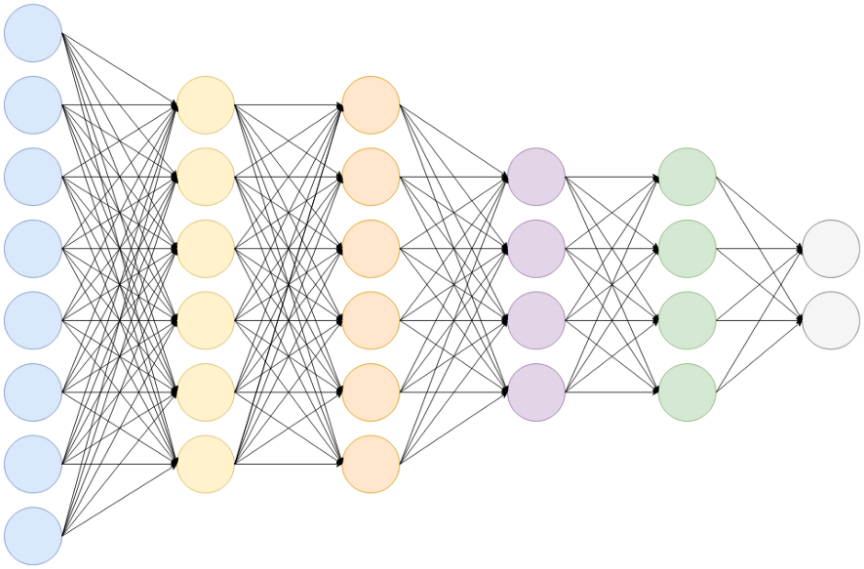


FIGURE 2.11: Fully Connected Layer

2.5 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks or RNNs were proposed by John (Hopfield [1982a]). They are type of neural network, classified from deep neural network architecture used to model sequential data (speech, text...)(Deng [2019]).

In the context of sequential data, RNN can be very powerful for the prediction process compared to other networks because it uses feedback loops in processing data. The advantage of RNN is that it takes into account what is encountered before to process current information (Bouaziz [2017]). To do so, RNN contains a memory that stores all previous information that has been calculated until their current position, but this only in theory, because in practice does not have ability to look back in long sequences, they look back only in few steps, To solve this problem, another type of RNN termed Long-Short Term Memory (LSTM) is appeared (Mdhaffar et al. [2018]).

In feedforward architecture, the network takes their decisions just from current input. But the RNN takes their decision from current and previous input. Hence, RNNs are suitable for temporal series analysis, that allow to exhibit dynamic temporal behavior for a time sequence.

2.5.1 RNN Formalism

In language processing, when we want to read a sentence, must be read it word by word. It is necessary to store information from the first word to last, the next word take in consideration the previous word for complete the meaning. RNN take its formalism from this mechanism.

RNN represented as shown in Figure 2.12

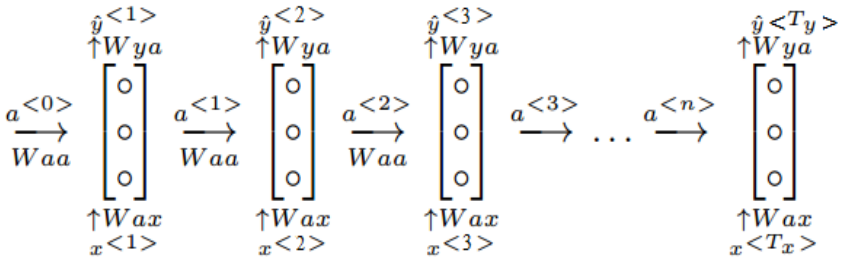


FIGURE 2.12: Recurrent Neural Network

RNNs takes as input a sequence $x = (x_1, x_2, x_3, \ldots, x_T)$ and defines the sequence of hidden states $a = (a_1, a_2, a_3, \ldots, a_T)$ to produce a sequences of output $y = (y_1, y_2, y_3, \ldots, y_T)$ in time $t = T$, where T is total number of inputs, W is weights matrix, $\hat{y}$ is prediction output with

an activation function g (usually the hyperbolic tangent (tanh)) as following equations

$$a^{<t>} = g(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \tag{2.1}$$

$$\hat{y}_{<t>} = g(W_{ya}a^{<t>} + b_y) \tag{2.2}$$

to simplify this notation, we can use vector representation :

$$a^{<t>} = g(W_a[a^{<t-1>}, x^{<t>}] + b) \tag{2.3}$$

With

$$W_a = [W_{aa}|W_{ax}] \tag{2.4}$$

And

$$[a^{<t-1>}, x^{<t>}] = \begin{bmatrix} a^{<t-1>} \\ \\ x^{<t>} \end{bmatrix} \tag{2.5}$$

Then

$$[W_{ax}|W_{ax}] \begin{bmatrix} a^{<t-1>} \\ \\ x^{<t>} \end{bmatrix} = W_{aa}a^{<t-1>} + W_{ax}x^{<t>} \tag{2.6}$$

2.5.2 Types of RNNs

We have seen that artificial neural network, convolution neural network . . .are constrained and limited. They allow fixed size vectors in input and output and fixed number of layers in the model. However, recurrent networks allow variable length sequences of vectors either input, output or both. that's what make it the better and more exciting (Karpathy [2019]).

There are many types of recurrent neural networks (Figure 2.13), each one is used according to the type of the used data.

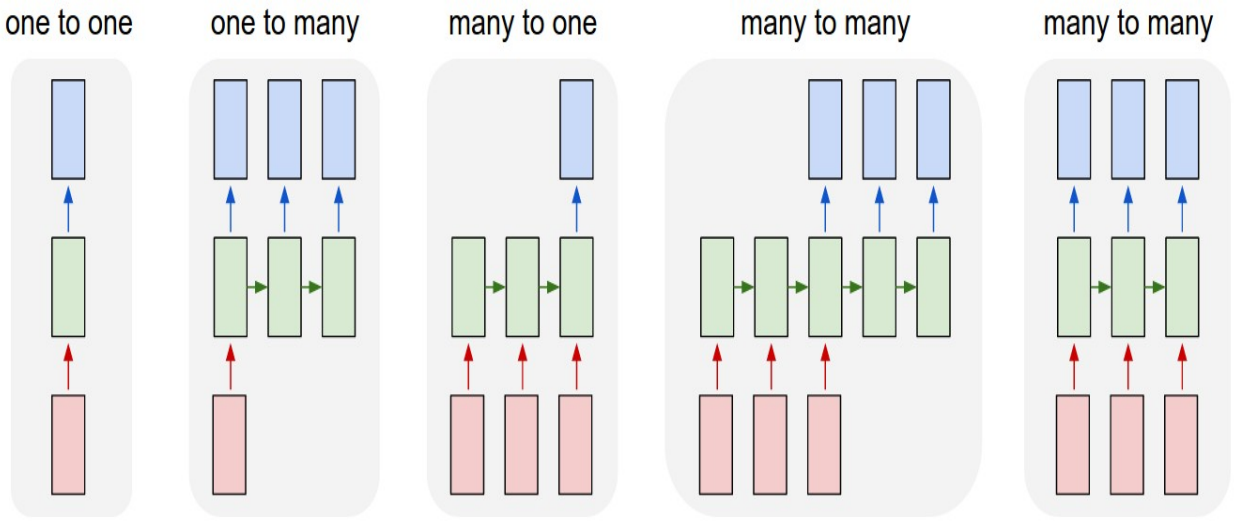


FIGURE 2.13: Types of RNN (Zocca [2017])

The different types of RNN are :

One to One

One to One are used in the processing of non-sequential data because it loses the interest for recurrence, it does not need timestamps. In this type there is almost no difference between RNN and CNN. It accept a single input for designated to output. This type is applied in image classification application.

One to Many

One to Many type is used to process single input for generating sequence of outputs. For example, create the captions to the images where the input is a picture and the outputs are sequence of words as comments annotation of the images. This application applied images captioning.

Many to One

Many to One handles a sequence of inputs in order to single output that is usually used to sentiment classification. For example, it can set text in inputs and the output be a notes about it (positive, negative, or funny...).

Indirect Many to Many

Indirect Many to Many receives a sequences of inputs and produce sequences of outputs when the inputs is finished, which is the most common and recently used in the application at recent days. It uses in machine translation and speech recognition.

Direct Many to Many

Direct Many to Many has the same length of inputs and outputs, that result from each new input, where the current input is built based on the previous. It uses in the video classification (Varma and Das [2019] ; Zocca [2017]).

2.5.3 Problems of RNN

Although the RNNs are widely used in sequence modeling, they encounter some problems in term of their depth.

Exploding Gradient Problem

Exploding Gradient is the significant increase in the level of gradient during training. It leads to an unstable network in learning, especially in long sequences. This problem can be solved if the gradients are cut or canceled (Pascanu et al. [2019b]).

Vanishing Gradients Problem

This was a major problem in the 1990s and much harder to solve than the exploding gradients. The values of a gradient become too small and the model stops learning or takes a long time to reach a solution.

Vanishing Gradients Problem happened in the case of long term dependencies because RNN has difficulties in learning it.
For example, in the sentence of "The man who was carrying a green wallet in his hand went inside". The verb "went" too far from "the man", the normal RNN is unlikely to be able to capture the information of "the man is inside".

2.5.4 Long Short-Term Memory

Long Short-Term Memory or LSTM (Hochreiter and Schmidhuber [1997]) is a type of RNNs developed to solve their problems.

Short-term problem known in RNNs is means that RNNs are not able to memorize at the long term (Mdhaffar et al. [2018]). Therefore came what's named Long short-term memory for expand memory to learn over a long time. LSTM also used for tackle the exploding and vanishing gradient problems (Pascanu et al. [2019b]) it prevents gradients from increase or decrease too much.

The basic units in LSTM called gates (forgot, input, output gates) used for control information flowing over the network, they try to define which information should be updated in the memory cell.

The gates in LSTM are the sigmoid activation functions that mean their output is 0 or 1. In most of the cases it is either 0 or 1.

Sigmoid function because we want to give only positive values, we need to keep a particular feature or we need to discard that feature, where 0 means the gates are blocking everything and 1 means gates are allowing everything to pass through it.

The Forgot gate decide which is relevant information can be keep for the next steps, and input gate or update gate decides which relevant information must to add to the current step and output gate decide which a next hidden state by which will the information pass.

LSTM takes the following path :
We use a cell memory $c^{<t>}$ instead of $a^{<t>}$ then we compute temporally $\tilde{c}^{<t>}$ as follows :

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c) \tag{2.7}$$

The update gate can be computed as a sigmiod function :

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u) \tag{2.8}$$

The forgot gate use also a sigmiod function :

$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f) \tag{2.9}$$

The output gate :

$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o) \tag{2.10}$$

The output of the LSTM unit can be obtained as follows :

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>} \tag{2.11}$$

2.5.5 Gated Recurrent Units (GRU)

Gated Recurrent Units or GRU introduced by (Cho et al. [2019]), It is similar to LSTM in some cases and also used to solve RNNs problems. Like LSTM , GRU have gates, update gate and reset gate are two basic vectors in GRU that decide which information can be passed to the output. The important thing which is special for them is that they train to keep information in long time, without eliminate the information which is irrelevant to the prediction.

The update gate decides to copy all the information in the past or determine the amount of them (the information required in the future) and this can eliminates the possibility of happened the vanishing gradient problem.
The module uses reset gate to determine which information to be forgotten. It's same of update gate but the difference is in the weights of the gates usage.
If set the reset gate to all 1's and update gate to all 0's, it will arrive at the simple RNN model.

GRU takes the following path :
We use a cell memory $c^{<t>}$ instead of $a^{<t>}$ then we compute temporally $\tilde{c}^{<t>}$ as follows :

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * a^{<t-1>}, x^{<t>}] + b_c) \tag{2.12}$$

The update gate can be computed as a sigmiod function :

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u) \tag{2.13}$$

The reset use also a sigmiod function :

$$\Gamma_r = \sigma(W_r[a^{<t-1>}, x^{<t>}] + b_r) \tag{2.14}$$

The output of the GRU unit can be obtained as follows :

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 + \Gamma_f) * c^{<t-1>} \tag{2.15}$$

$$a^{<t>} = c^{<t>} \tag{2.16}$$

2.5.6 GRU vs LSTM

The obvious difference between LSTM and GRU is in number of gates. LSTM have 3 gates (update, forgot and output) and GRU have (update and reset) and both solve vanishing and exploding gradient problems.

There is difference between them in the maintenance of the cell states. LSTM use its three gates to control the information flow from internal memory cell which makes it more flexible, but less efficient wise in the memory and time compared to GRU. GRU use its gates to control the information flow from the previous time steps. GRU is fast in training than LSTM (Gao and Glowacka [2019]).

The Figure 2.14 illustrates the different between LSTM and GRU, their gates, functions and the operations used in them.

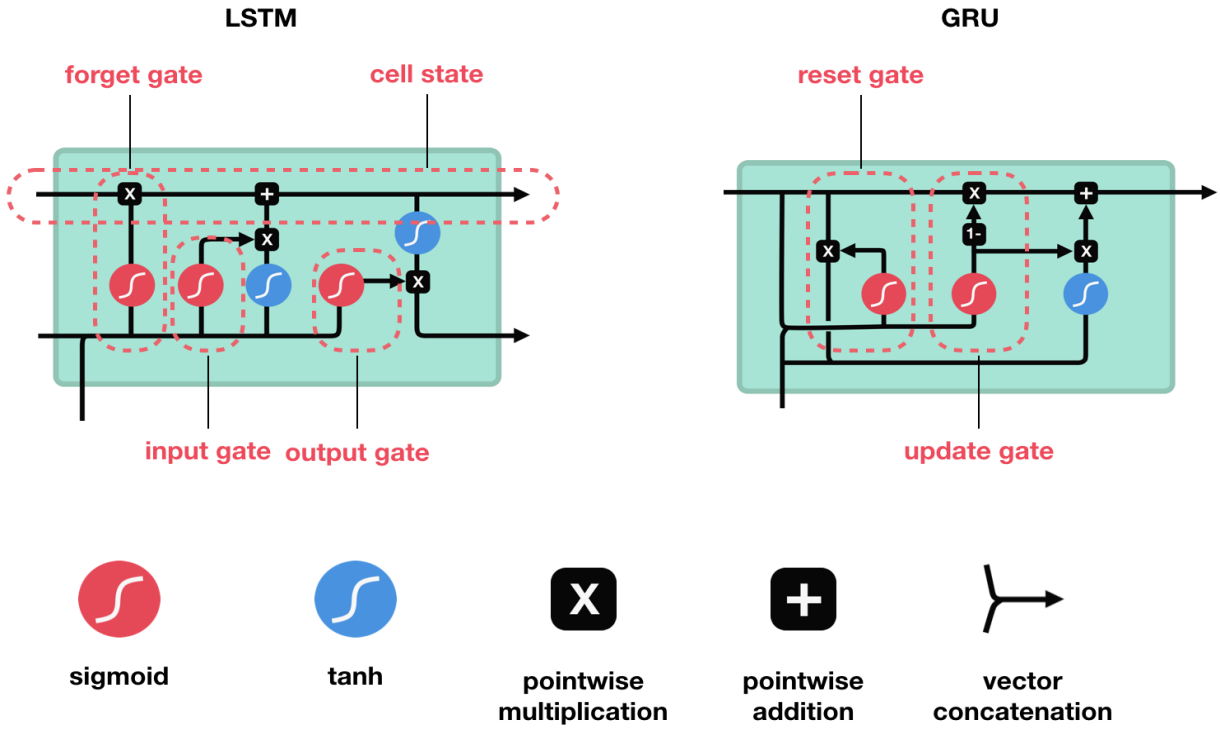


FIGURE 2.14: different between LSTM and GRU

2.5.7 Bidirectional RNN (BRNN)

In some cases, we need to know what's happen in the future to complete the current step. For example, a psychiatrist, must listen to every what the patient says, in order to diagnose the patient's condition well.

In case of sequence classification tasks, sometimes we need to know the past and all of the future or part of it, into the sequences before making a prediction. According to processing the sequences in RNNs, it use the previous information in their treatment of the current step, therefore it don't take the future information in consideration.

In 1997, Schuster and Paliwal (Schuster and Paliwal [1997]) introduced a solution to this situation, called bidirectional recurrent neural network (BRNN). That consists in presenting each sequence to be treated to two RNNs of the same type but with different parameters. One treats the sequence in the natural order and the second treats the sequence in reverse order.

In a bidirectional RNN, the input sequence is traversed in both directions by two distinct RNNs whose output sequences are combined. That can take a decision which is based on the information contained in the entire input sequence, in each time step (Gelly [2017]).

The Figure 2.15 represents BRNN.

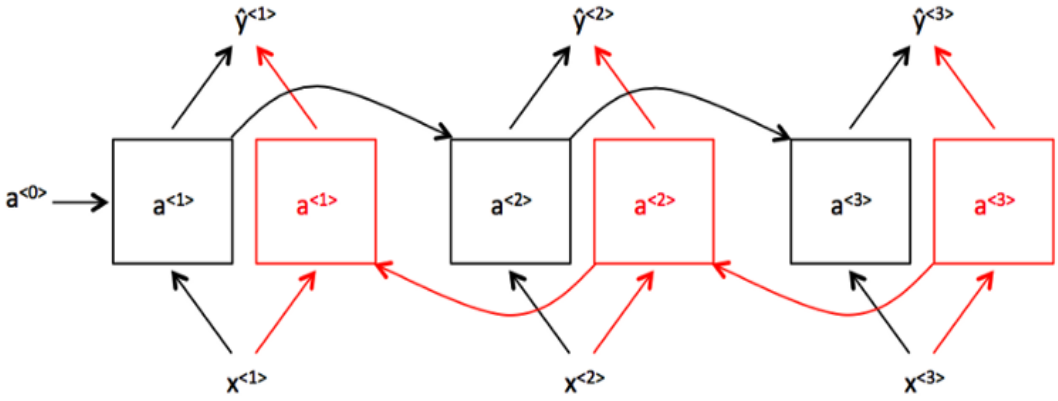


FIGURE 2.15: Bidirectional RNN (Gelly [2017])

BRNN takes the following path :
RNN in natural order :

$$\overrightarrow{a} = g(W_{(x\overrightarrow{a})})x_t + W_{(\overrightarrow{a}\overrightarrow{a})}\overrightarrow{a}_{(t-1)} + b_{\overrightarrow{a}} \quad (2.17)$$

RNN in order inverse :

$$\overleftarrow{a} = g(W_{(x\overleftarrow{a})})x_t + W_{(\overleftarrow{a}\overleftarrow{a})}\overleftarrow{a}_{(t-1)} + b_{\overleftarrow{a}} \quad (2.18)$$

The output of BRNN :

$$y_t = W_{(y\overrightarrow{a})} + W_{y\overleftarrow{a}} + b_y \tag{2.19}$$

However BRNN have a disadvantage that it needs a complete sequence of data, and then begin the prediction. That can be long, time consuming and causing congestion in data processing.

2.5.8 Deep RNNs

Deep RNNs (Pascanu et al. [2019a]) is expansion of RNN terms of hidden layers (Figure 2.16). In fact it uses a multiple hidden layers used to learn complex functions. The equation of Deep RNN (example for the case of $a^{[2]<3>}$) is :

$$a^{[2]<3>} = g(W_a^{[2]}[a^{[2]<2>}, a^{[1]<3>}] + b_a^{[3]}) \tag{2.20}$$

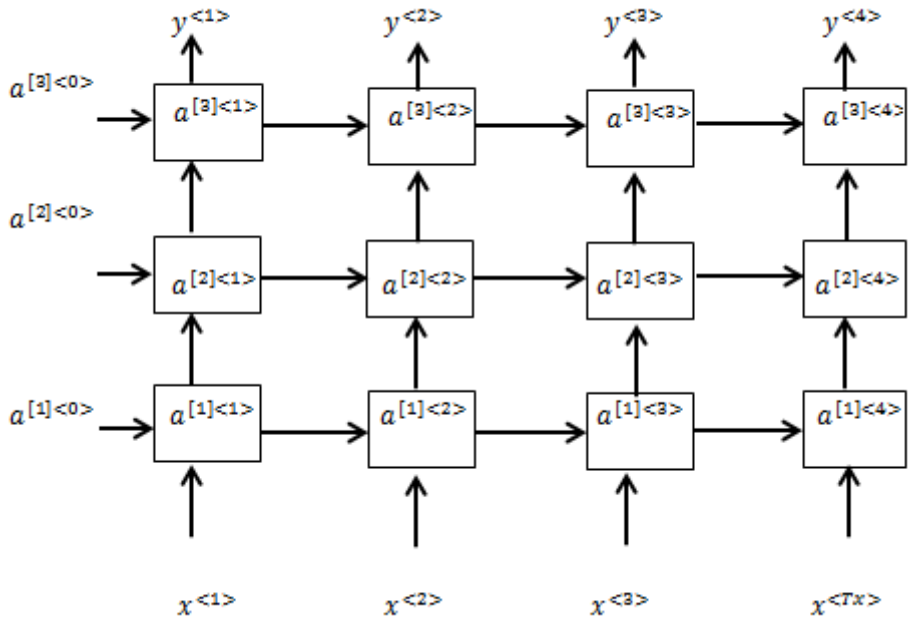


FIGURE 2.16: deep RNN

2.5.9 Backpropagation through Time

Backpropagation through Time or BPTT (Rumelhart et al. [1986]) is an extension of backpropagation (BP) in neural network. The goal of the backpropagation algorithm is to minimize the error of the network outputs, using update weights with derivatives. BP is suitable for fixed size of input and output, and this is not compatible with RNN, for that BPTT is used in training instead of BP.

BPTT is an algorithm used to train RNNs, in each time step the RNNs given an input and predicts an output compatible to it .For that, the BPTT applies its work for each time step by calculating and sum the errors and then returns in the network in order to update the weights (Figure 2.17).

The Loss function of backpropagation through time is :

$$L^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -(y^{<t>} \log \hat{y}^{<t>} + (1 - y^{<t>}) \log(1 - \hat{y}^{<t>})) \tag{2.21}$$

Cost function :

$$L(\hat{y}, y) = \sum_{t=1}^{T_y} L^{<t>}(\hat{y}^{<t>}, y^{<t>}) \tag{2.22}$$

In the BPTT we will calculate the gradient descent of the loss with respect to W, at each time step (Equation 2.23). W is matrix of the weights for the long term dependencies.

$$\frac{\partial L}{\partial W} = \frac{\partial L_t}{\partial W} \tag{2.23}$$

For example, for calculate the gradient of the loss at the time step t=3.

$$\frac{\partial L_3}{\partial W} = \frac{\partial L_3}{\partial y_3} . \frac{\partial y_3}{\partial a_2} . \frac{\partial a_2}{\partial W}$$

$$= \sum_{t=0}^{2} \frac{\partial L_3}{\partial y_3} . \frac{\partial y_3}{\partial a_2} . \frac{\partial a_2}{\partial a_t} . \frac{\partial a_t}{\partial W}$$

$$= \sum_{t=0}^{2} \frac{\partial L_3}{\partial y_3} . \frac{\partial y_3}{\partial a_2} . \left(\prod_{j=t+1}^{2} \frac{\partial a_j}{\partial a_j - 1} \right) . \frac{\partial a_t}{\partial W} \tag{2.24}$$

However, in the case of the increasing in the time steps also makes the increasing calculation leading to a noisy model. The high cost of updating one parameter, turns impossible to use BPTT in large iterations. It's one of the disadvantages of BPTT.

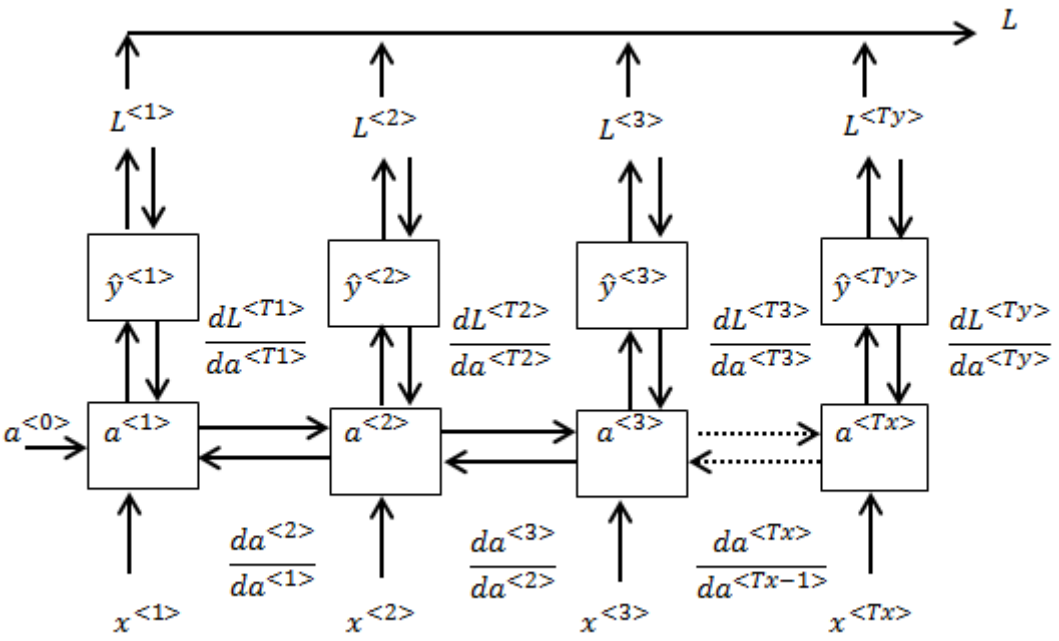


FIGURE 2.17: Backpropagation through Time

2.6 Application domains of deep learning

There are many domains that deep learning has succeeded in developing such as :

— Automated Driving : deep learning has been applied in this area for helping drivers to safely drive, there are methods to identify the road, as well as to identify objects automatically such as traffic signs, and whether there are pedestrians or not, and other instructions needed by the drivers while driving. This can be subscribed in traffic regulation and minimize the accidents (Huval et al. [2015]). Furthermore, the CNN is appropriate deep learning model, which was a successful model in ImageNet challenge, this has been positive effect in autonomous driving (Sallab et al. [2017])

— Medical Research : in the field of medicine doctors use deep learning in the detection of cancer cells automatically, this project was carried out by the University of California at Los Angeles (UCLA) [6]. The project was led by the professor Bahram Jalali and his students Claire Lifan Chen, and Ata Mahjoubfar. They build a microscope that extracts a set of data for deep learning application to identify cancer cells with high accuracy (Mason [2019]).

— Electronic : Deep Learning is used in recent times for the services of electronic devices and their applications, including speech recognition that is used for internet searches and for speech translation. In 2014, Skype introduced a new service called the Skype translator [7] which translates at real time for help Skype users to communicate with

6. https ://medschool.ucla.edu/research-overview
7. https ://www.skype.com/en/features/skype-translator/

any user in the world no matter what language they use.

2.7 Conclusion

In this chapter, we introduced concepts of deep learning and its definition and why came this type of learning. we have describe the two basic types of deep learning named CNN (Convolutional Neural Networks) specialized in image processing and RNNs (Recurrent Neural Networks) that are dedicated for sequences processing. Finally, we have presented briefly some areas in which deep learning had been used successfully.

Our focus was clear on RNN because it is considered as one of the sequence models approaches that deals with several applications, including speech recognition. This is what will be discussed in the next chapter.

Chapitre 3

Sequence Models and Speech Recognition

3.1 Introduction

Sequence Models or SM have attracted a lot of attention because most of the data in the current world is in the form of sequences. This kind of data is difficult to processed manually for that it needs to be treated automatically with specific models such as hidden markov model (HMM) and recurrent neural network (RNN) which handle to manipulate sequential data.

SM are needed for many fields in real world and specially in business practice in terms of quality and productivity. SM has many applications like machine translation, text generation, image captioning, sentiment classification and speech recognition,. . ..

Speech Recognition (SR) is used to recognize spoken words by microphone or telephone, in order to converted into transcription text. SR is a complex model based on two basic models, one for generate the acoustic data (Acoustic Model) and the other for generate the language data (Language Model).

In this chapter, we introduce the definition of sequence data and sequence model. Thereafter, we show some examples of SM. Then we overview the speech recognition. We try to give the state of the art of speech recognition and the methods used for measuring its performance.

3.2 Sequence Models

This section presents, first, the definition of the sequence models.Thereafter we describe some sequence models. Finally, we briefly introduce some sequence models applications.

3.2.1 Definitions

Sequence is a data that could be seen as a series of data point which depend on each other and the order is important for giving the meaning (text or audio...). For example, text is a sequence of words, word is a sequence of character, audio is a sequence of signals and DNA is a sequence of nucleotide (A, C, G, T).

Sequence Model is a model that process sequential data. Where either input or output or both are sequences.

3.2.2 Examples of Sequence Models

Hidden Markov Model (HMM)

Hidden Markov Models (HMM) (Baum and Petrie [1966]) is a statistical sequence model. HMM are widely used in machine learning where the data samples are time dependents, for example in speech recognition, language processing, and video analysis (Blunsom).

The HMM model represents the probability which generate the sequences x that represent the observation states and y that represent hidden states (Figure 3.1) by $P(x, y)$ that composed of two main parameters, the first is the transition probabilities $p(y_i|y_i - 1)$ which determine the degree of connection between y_i and $y_i - 1$ i.e. the probability of pass to different states, this probability depends only on the previous state that mean don't effect from the prior of previous states. and the second is the emission probabilities $p(x|y)$ which define how the observed variables of x are related to those of y.

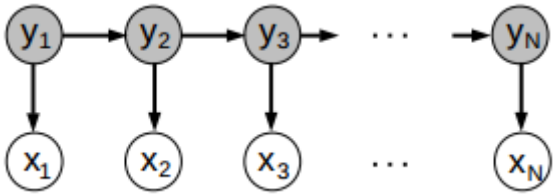


FIGURE 3.1: Diagram of an HMM model (Bouaziz [2017])

Hidden Markov models are very commonly used for acoustic modeling in Speech Recognition Systems (Bouaziz [2017]).

N-gram Model

The n-gram models since their appearance (Damerau [1971]) have been considered as the best approaches in language modeling.

In language modeling the probability of word w_i is determinate by all history h_i of previous words $P(w_i|h_i) = P(w_i|w_1w_2...w_i\_1)$.

Instead of calculating all histories, n-gram model offered an approximation which consist to consider just the last few words. n represent n-last words takes as histories. For example, for n=2 (2-gram or bigram), instead of computing $P(w_{10}|w_1 \dots w_9)$ we compute $P(w_{10}|w_8 w_9)$ (Bouaziz [2017]).

Recurrent Neural Network Model

In deep learning, RNN and their extension (LSTM, BRNN,...) are state of the art and the most powerful of sequence modelling, Thanks to their specific architecture. RNNs are detailed in section 2.5.

3.2.3 Sequences Model Applications

Sequence models were applied in the areas that contain sequential structure. The sequence can be in their input, output, or both. They are processed by a lot of models like RNNs which are one of the best models in sequence processing. Sequence models can be useful in many applications such as :

— Machine translation : like " Google translate " it accepts as input a sentence in a particular language (e.g. French) and the model converts it into another language (e.g. English). In this case both the input and the output are sequences.

— DNA sequence analysis : uses in biological domain, DNA are sequences represented with four alphabets A, C, G, and T. DNA sequence analysis applications accept DNA sequence as input and the model give as output a sequence which is a part of DNA sequence.

— Sentiment classification : is the most popular application of sequence models used in the last years especially in social media that use it for knowing the opinions of users . It accepts sentence as input and the model predict the sentiment of the sentence as notes (positive, negative, angry, elated,...) or ratings or starts.

— Video activity recognition : The model accepts as input sequences of frames of video and tries to recognize as output what is the activity that occurs in the video. Such as subtitle in YouTube.

— Name entity recognition : it accepts sentence as input and the model tries to extract the flag names from it.

— Speech recognition : Like " Google Voice " it accepts as input an audio clip and the model produces the text corresponding to what's spoken.

3.3 Speech Recognition

Speech recognition also known as Automatic Speech Recognition (ASR) and speech-to-text (STT/S2T), has a long history. Since 1952, the first speech recognition system which recognizes only digits spoken by a single person (Davis et al. [1952]).

ASR was developed over times from an ASR that recognize a limited vocabulary and speech with low performance to an ASR that perform with huge vocabulary and speech, even in existence of noises. This development started by traditional AI including hidden markov model in 1980 until recent years with emergence of deep learning that gives speech recognition a new boost in performance.

Many major tech companies of the world have an interest in speech recognition because of the different applications for which it can be used. For example, Voice Search by Google, Siri by Apple, and Alexa by Amazon (Bakker [2017])(Manjutha et al. [2019]).

Speech recognition is a process that allows a machine to convert a speech signal to a sequence of words (Anusuya and Katti [2019]). There are many reasons that makes this domain interesting such as (Manaswi [2018]) :

— Allows Blind or physically challenged people to control different devices using only voice.
— The ability to keeps records of meetings by converting the spoken conversation to text transcripts.
— Get subtitles of the words being spoken from audio in video or an audio files.

3.3.1 Taxonomy of ASR Systems

Speech Recognition Systems can be classified in several classes according to the type of speech utterance, type of speaker model and the type of vocabulary (Figure 3.2). This diversities make speech recognition challenging and more complex task.The diversities are explained below(Radha [2019]) :

Types of speech utterance

Speech recognition can be categorized according to what kind of speech it can recognize. Can be classified into three varieties as follows :

— **Isolated word :** The isolated word recognizer recognizes single utterances i.e. accepts single word at a time. Each spoken word should be quiet.
— **Connected word :** A connected word system is similar to isolated word, however it allows separate utterances to run together with a minimum pause between them.
— **Continuous speech :** Continuous speech recognition system allows the users to speak naturally and in parallel, while the computer will determine the content. This type of system is difficult to develop.

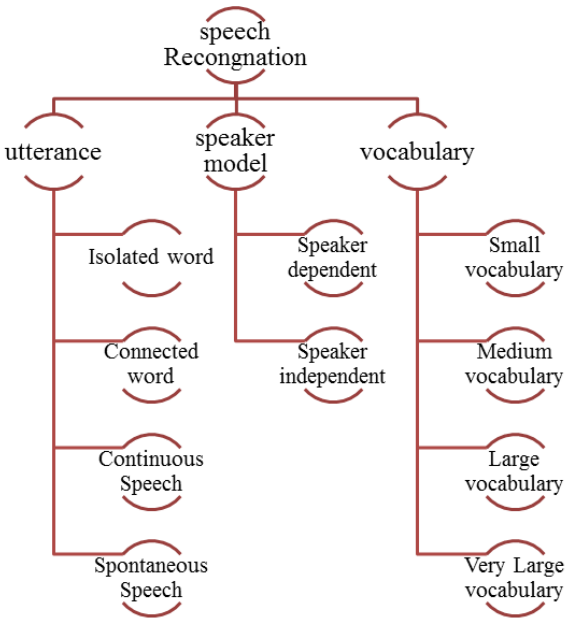


FIGURE 3.2: Taxonomy of Speech Recognition Systems

— **Spontaneous Speech :** Spontaneous speech recognition system recognizes the natural speech. It is the type of speech that comes suddenly through mouth. It can contain mispronunciation, false-starts and non-words.

Types of speaker model

Each speaker has special voice, according to his unique physical body and personality. Speech recognition system can be classified into two main categories as follows :

— **Speaker dependent models :** Speech recognition systems with this model are designed for a specific speaker. They are easier to develop and more accurate but they are not so flexible.

— **Speaker independent models :** Speech recognition system with this model is designed for variety of speaker. It is difficult to develop this kind of system. It has a less accurate but they are very flexible.

Types of vocabulary

The vocabulary size of speech recognition system affects the processing requirements, complexity and the accuracy of the ASR system. The ASR system are classified based on the vocabulary as followings (Saksamudre et al. [2015]) :

— **Small vocabulary :** 1 to 100 words or sentences.

— **Medium vocabulary :** 101 to 1000 words or sentences .

— **Large vocabulary :** 1001 to 10,000 words or sentences.

— **Very large vocabulary :** More than 10,000 words or sentences.

3.3.2 Speech Recognition Pipeline

The basic ASR pipeline contains many components including pre-processing and feature extraction, acoustic model, language model, pronunciation model and finally the decoder phase as shown in Figure 3.3. Each component performs a tasks in order to output the most probable transcription according to the speech utterance; this is represented by the following :

$$Transcription = argmax(P(words|audiofeatures)) \tag{3.1}$$

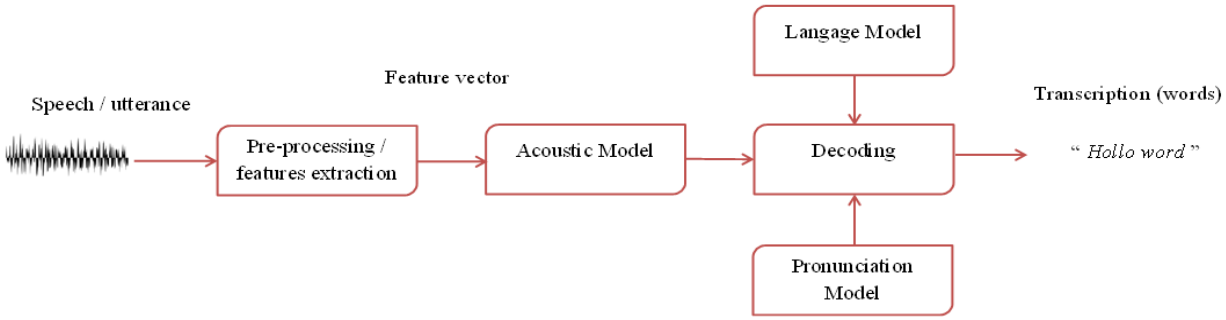


FIGURE 3.3: Automatic Speech Recognition Pipeline

Speech as data

Speech or utterance is the most basic, efficient and common form of communication method for people to interact with each other. It can typically convey information. It is a vibration that propagates through a medium, such as air. The audible vibrations of humans are between 20 Hz and 20 kHz.The Figure3.4 represents a recording to someone saying "hello word" .

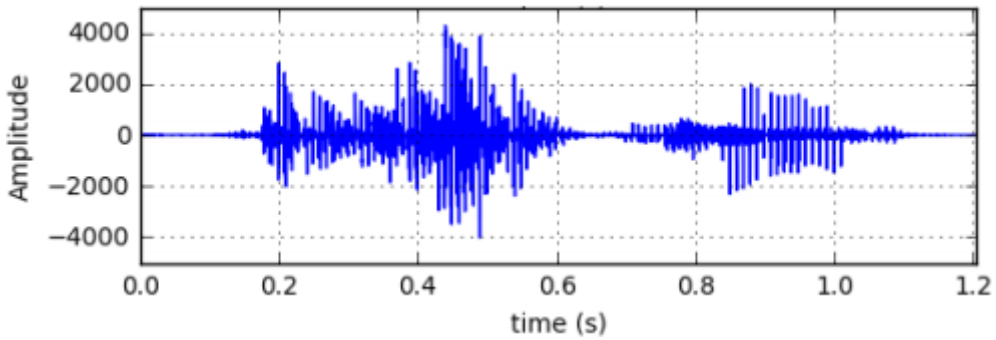


FIGURE 3.4: Speech signal of "hello world" (Zocca [2017])

To manipulate electronic devices comfortably, people would also like to interact with computers via speech, rather than using primitive interfaces such as keyboards and pointing devices. In order to simulate this task to the computer it should capture and convert the vibrations into a digital signal by a microphone that sampled from a continues signal applied an audio signal processing. To feed in ASR model for recognition (Radha [2019]).

Pre-processing and Features Extraction

The input speech signal should be processed in order to extract the relevant features for recognition task. This task is done in training and test phases (Samudravijaya [2004]).
The recorded acoustic signal is an analog signal. An analog signal cannot be directly transferred to the ASR systems. Hence these speech signals need to be transformed in the form of digital signals and then only they can be processed (Stenman [2019]).

Feature extraction is very important step because the performance of recognition in ASR depends heavily to it (Ghai and Singh [2019]).

It looks to extract the feature from an audio wav. This step identifies the component of the audio that are useful for the recognition of the linguistic components and discard the irrelevant one which are a noise. There are several method used to perform this task such as Mel-Frequency Cepstral Coefficient (MFCC), Linear Prediction Cepstral Coefficient (LPCC), Linear Predictive coding (LPC), Principle Component Analysis (PCA), Linear Discriminant Analysis (LDA)...(Gao and Glowacka [2019]).

— **MFCC** (Davis and Mermelstein [1980]) is one of the most popular feature extraction techniques used in speech recognition. It is based on frequency domain using the Mel scale which is based on the human ear scale. The steps to generate an MFCC feature vectors are illustrated in Figure 3.5 The MFCC involves the following steps (Manjutha et al. [2019]) :

1. Pre-emphasis : in this step the speech signal increases the amplitude of high frequency bands and decrease the amplitudes of lower bands which is implemented by FIR (Finite Impulse Response) filter.
2. Framing and windowing : The speech signal is split into number of frames. The frame size considered as 25 ms, hamming windowing then applied to minimize the signal discontinuities at the starting each edge of the frames.
3. Fast Fourier Transformer (FFT) : Each frame of N samples in time domain is converted into frequency domain.
4. Mel Filter Bank : converting the scale of frequency from linear to mel scale.
5. Logarithm : is knowns as log Mel spectrum taken for the mel filter bank.
6. Discrete cosine Transform (DCT) : converting The log mel scale from frequency domain to time domain which produces the feature of MFCC.

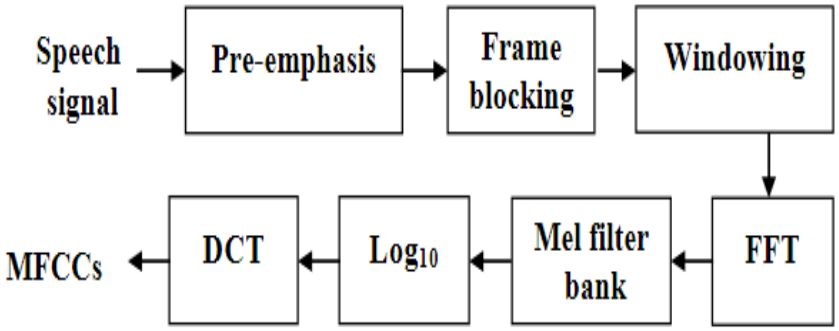


FIGURE 3.5: Steps of MFCC feature extraction

— **Spectrogram**

Spectrogram is a photographic representation of a spectrum computed as a sequence of FFTs (Fast Fourier transformation) of windowed data segments.

The format of spectrogram is a graph with two geometric dimensions, one axis represents time, and another axis represents frequency. A third dimension uses the color or size of point to indicate the amplitude of a particular frequency at a particular time.Figure 3.6 represents the spectrogram of the signal audio of "hello world".

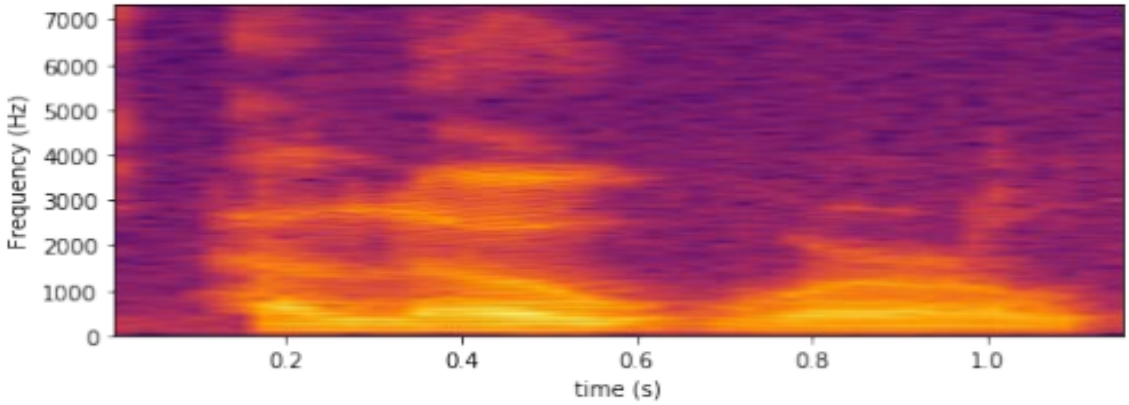


FIGURE 3.6: Spectrogram of "hello world" (Zocca [2017])

Acoustic Model (AM)

Acoustic Model (AM) is the fundamental part of an ASR system. The AM takes the feature extracted and find the most probable phonemes(Equation 3.2)

$$P(audiofeatures|phonemes) \tag{3.2}$$

Phonemes are the basic units of sound that define the pronunciation of words. For example, word "SPEECH" is composed of four phonemes /s/, /p/, /iy/ and /ch/. Each phoneme is tied to a specific sound. Spoken English consists of around 44 phonemes. Different approaches have been used to Acoustic modelling AM based on HMM including, for example GMM, ANN and support vector machines (SVM) (Krüger et al. [2007]).

HMM is the most used acoustic model in several years. In recent years deep learning approach appeared in acoustic model and improved the performance of ASR (Zocca [2017]).

HMM for Acoustic Model

Hidden Markov Model (HMM) analysis founded in the 1970s and 1980s has been the dissemination and development and successfully applied to the modeling of the acoustic signal.

HMMs are simple networks that can treat speech (sequences of cepstral vectors) using a number of states for each model and modeling the short-term spectra associated with each state. Each word or phoneme, will have a different output distribution, a HMM for a sequence of words or phonemes is made by concatenating the individual trained HMM for the separate words and phonemes.

Hidden Markov models was the most used in AM before the emergence of deep learning. HMMs usually used with Gaussian mixture model (GMM) formed an hybrid HMM-GMM model(Rodríguez et al. [1997]).

HMM to deal with the temporal variability of speech and GMM to determine how well each state of each HMM fits a frame or a short window of frames of acoustic features (Rodríguez et al. [1997]).

Deep learning for Acoustic Model

The first use of deep learning in speech recognition is to replace GMMs with deep neural networks (DNN). DNNs take a window of feature vectors as input and output the posterior probabilities of the HMM states :

$$P(HMMstate|audiofeatures) \tag{3.3}$$

The networks used in this step are typically pre-trained as a general model on a window of spectral features DNN-HMM model (Hinton et al. [2012]).

GMMs are less perform than DNNs because DNNs are able to learn non-linear and complicated functions, which can better handle complex acoustic features (Zocca [2017]).

CNN for Acoustic Model

CNN is also used in acoustic modeling. It replaces DNN formed a hybrid CNN-HMM model. Ossama Abdel-Hamid with a group of researchers in their paper (Abdel-Hamid et al. [2014]) replace DNN with CNN. They conclude that the ASR with CNN-HMM acoustic model reduces the error rate by 6% - 10% compared with DNN-HMM.

RNN for Acoustic Model

Although the RNN is perfect in processing the sequential data however it encounter problems. The RNN can't handle the ASR well, because it need a perfect alignment between the labels of training data and the inputs, consequentially the mapping from input to output has a lot of noise that makes a network not learn anything. In order to exploit the force of RNN in modeling appear a model that hybrid the HMM with RNN (Gao and Glowacka [2019]), where the RNN is used for modeling the emission probabilities of HMM model.

LSTM also was used in acoustic modeling to output the posterior probabilities of phonemes at a given frame (Zocca [2017]).

Connectionist Temporal Classification (CTC)

CTC (Graves et al. [2006]) is an objective function to optimize the distance between the output and the target sequences.
The output layer of the CTC network has a softmax function to produce the distribution for each labels plus the extra blank. The blank represents the unexciting of relevant label in a time steps. Then it is predict an output, this output translated in to sequence labeling by eliminating the blank and the repeated labels.
This reduction consists of reducing multiple output sequence to the same output.

The CTC function with LSTM give the state of the art in acoustic model in ASR. This Combination eliminates the need of HMM for modeling the temporal variability (Zocca [2017]).

Attention Based Model

Attention based model (Chorowski et al. [2019]) are sequence to sequence which have the ability to pay attention dynamically to parts of the input sequences. It predicts the correct phoneme by an automatic search about the relevant parts in the input sequences without need to a segmentation of parts.

Attention based model contains an RNN encoder-decoder technique. The decoder network decodes the representation of input to phonemes then the encoder network will generate the representation of inputs by an encoding to a suitable representation (Zocca [2017]).

Pronunciation Model

Pronunciation model compares the sequence of phoneme generated by the acoustic model with the set of words present in a dictionary in order to produce a sequence of words (Saini and Kaur [2013]).

It contains the information about all words that are known to the system and their corresponding phoneme (how this words are pronounced) $P(phonemes|words)$.

Language Model

Language models (LM) is the probability of a sequence of words $P(words)$. There are words that have the same sounding phoneme. The human can distinguish between them by knowing the context. The goal of the LM is to provide this context to the ASR system. The ML specifies the valid words in the language and sequence that can occur. LM helps the recognition process by guiding and limit the search through word hypotheses. There are two level language models which are word- and character-level language models.

Word-based Models

The word-based model determines the probability of distribution through the sequence of word . It computes the probability $P(w_1, ..., w_m)$ for a sequence of words of length m to the full sequence of words. One of the models that perform this type of LM well is n-gram model.

Character based Model

Most LMs are performed at the word level, where the distribution is over a fixed vocabulary of words. However, the word level models are limited when they model text data that contains non-word strings like multi-digit numbers or words that there are not from the training data (out-of-vocabulary words).

The character based model overcome these problems. It models the distribution over sequences of characters instead of words. It allows to compute probabilities over a much smaller vocabulary. The vocabulary should comprise all the possible characters in corpus of texts. however, it needs to model much longer sequences for capturing the same information over time. The model that perform this type of LM well is LSTM RNN because of its ability to generate a long term dependencies.

Decoding

The decoding is the most important step in the speech recognition pipeline. A decoder performs for recognizing most likely transcript for the utterance. The acoustic model and language model with the pronunciation model combine together to get a probability of words over audio features :

$$P(words|audiofeatures) = P(audiofeatures|phonemes) * P(phonemes|words) * P(words) \tag{3.4}$$

This probability doesn't gives the final transcriptions yet. To find the transcription we should search over all the possible distribution of word sequences. The decoder models the search process that looks for the most likely transcription.

To find the most likely transcription, we have to search through all the possible word sequences.
The popular search algorithm is Viterbi algorithm based on dynamic programming (Forney [1973]). This algorithm is mostly associated with finding the most likely sequence of states in an HMM. However, the heuristic search algorithms such as beam search is used for large vocabulary speech recognition. The beam search keeps the n-best solutions during the search and suppose that all the rest does not include to the most likely sequence.

End-to-End Model

End to end model (Graves and Jaitly [2014]) offered by deep learning methods such as CTC, attention model allows to learn a ASR in new pipeline. End-to-End explain the force of DL by combining the Language Model and Acoustic Model in one model (Figure 3.7) that output the distribution of words directly. without need to modelling the phonemes explicitly. This model is becoming conceptually easier to be understood.

The learning in an end to end ASR allows to handle a variety of speech including different accents, noisy environments and different languages (Amodei et al. [2015]).
End-to-End model makes the new ASR pipeline more accurate (Zocca [2017]).

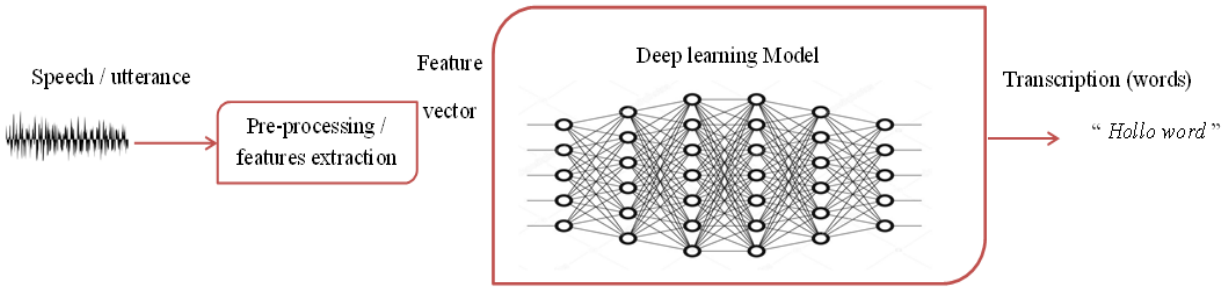


FIGURE 3.7: End-to-End Model

3.3.3 Performance of Speech Recognition

The performance of Speech Recognition system is measured by the accuracy. The accuracy measured by Word Error Rate (WER) or Character Error Rate (CER).

Word Error Rate is related to the error rate in term of word. is calculate the difference between the labeled and predicted transcription. the tool used for computing this error is the Levenshtein distance which count the number of Insertion (I), Substitutions (S) and deletions (D) then divided by total numbers (N) of words existing in the labeled transcription.

— Insertions (I) where the ASR output present a word not present in the labeled.
— Deletions (D) where a word is missed in the ASR output.
— Substitutions (S) where a word is substituted with another one.

The Word Error Rate defined by the following formula :

$$\text{WER} = \frac{S + D + I}{N} \tag{3.5}$$

Character Error Rate is related to the error rate in term of character. It is the same of Word error rate but in this case the measure based on character. The formula of CER is also by counting of the number of I, S and D and was divided by N number of character in the labeled transcription.

$$\text{CER} = \frac{S + D + I}{N} \tag{3.6}$$

3.4 Conclusion

This chapter is divided into two parts. The first part is an overview about sequence models. In the second part, we present speech recognition, then its stat-of-art from the conventional architecture into deep learning architecture which achieve an amelioration in performance in ASR system.

Chapitre 4

Experimentation

4.1 Introduction

In this chapter we attempt to give an experimentation of a speech recognition model based on deep learning methods such as RNNs. This model has a end-to-end architecture seen in section 3.3.2. Namely we use DeepSpeech2 (DS2) model.
The basic code source obtained from "Python Deep Learning Projects" book (Lamons [2018]).
we have not started from the scratch, but we have used the basic code source obtained from `https://github.com/PacktPublishing/Python-Deep-Learning-Projects/tree/master/Chapter07` to train and test the DS2 model, we use the LibriSpeech dataset.

This chapter is organised as followers : we start by describing the programming environment and the data set. Afterword the pre-possessing and the learning tasks are detailed. Finally we measure the performance of the trained model.

4.2 Environment

The hardware environment used for this experimentation is DELL computer that has an Intel Core i5 processor at 2.2 GHZ with 8 GB RAM, the operating system is windows 10 professional with 64 bit.

The implementation environment used in this experimentation is **Python** with Spyder[1] [1] editor in Anaconda Distribution[1] [2]. and Audacity to visualize the audio and spectrogram.

4.2.1 Python

Python is high level, interpreted and object-oriented open source programming language. Python is easiest to build models of machine learning and deep learning. It provides several

1. Spyder is free and powerful python IDE included with Anaconda.
2. Anaconda Distribution is the Data Science Platform, easiest way to perform programming languages Python and R in machine learning and deep learning. It is available in Linux, Windows, and Mac OS X.

libraries such as Tonserflow, numpy,...

— **Tonserflow** is a numerical computation library developed by Google Brain for machine leaning and deep learning. It can be used to build and train neural networks for detection of patterns and correlations.

— **NumPy** is the fundamental scientific computing library. It provides an advanced calcul to python such as basic linear algebra functions, Fourier transforms.

Python also makes the speech recognition application easy because it has packages that allow the extraction of features and to read audio files,...

— **Soundfile** is library used to read and write audio file. It represents audio data as NumPy arrays.

— **Python-features-speech** is library used to extract an MFCC features of speech for ASR.

— **Python-Levenshtein** is library used to calculate the distance and similarity between strings with an unequal length, the distance value describes the minimal number of deletions, insertions, or substitutions that are required to transform one string (source) into another (predicted).

4.3 Data Description

Speech recognition system need a dataset of utterances and their corresponding written text transcription.

4.3.1 LibriSpeech

LibriSpeech (Panayotov et al. [2015]) is a detaset segmented and aligned carefully containing 1000 hours of 16 khz of read English Speech. This dataset is prepared by Vassil Panayotov and Daniel Povey, the data extracted from the read audiobook of LibriVox project. The dataset is available on : `http://www.openslr.org/12/`.

LibriSpeech dataset is already divided into several parts (as folders) of train, test and validation set. Each folder contain several sub-folders, each folder is assigned to a specific speaker. The sub-folder contain 2 components. The first is set of utterances (.flac) [3] and the second is the corresponding transcription of these utterances in one file (.txt) (Figure 4.1).

In this project the dataset choosed is part of the huge librispeech dataset. The train set choosed named "train-clean-100" contain 100 hours of clean speech, The validation set named "dev-clean" and the test set named "test-clean" of few hours of clean speech.

3. flac : (Free Lossless Audio Codec) is open source audio format that allows compression of files without loss of data unlike MP3.

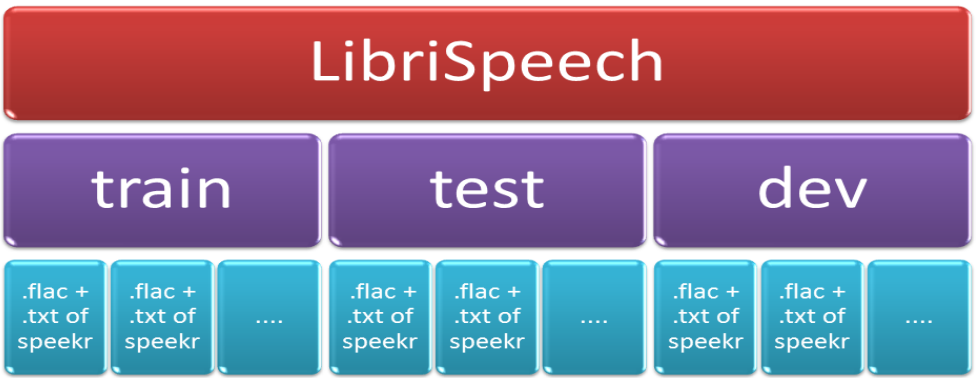


FIGURE 4.1: Architecture of LibriSpeech dataset

The Figure 4.2 is speech signal from the data set of a person saying "THEY WERE RUN OUT OF THEIR VILLAGE", and Figure 4.3 is its spectrogram .

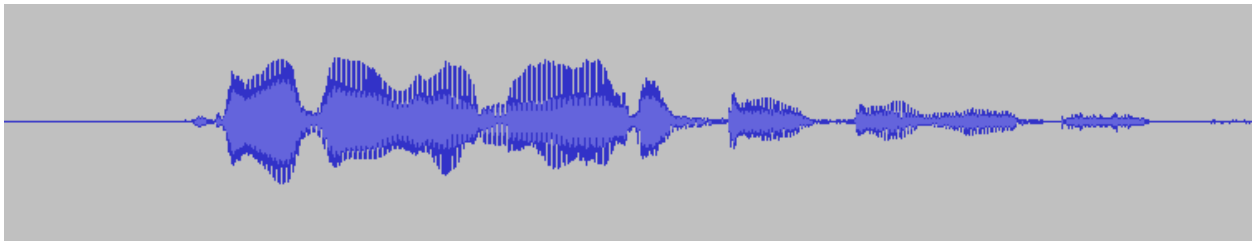


FIGURE 4.2: speech signal of someone saying "THEY WERE RUN OUT OF THEIR VILLAGE"

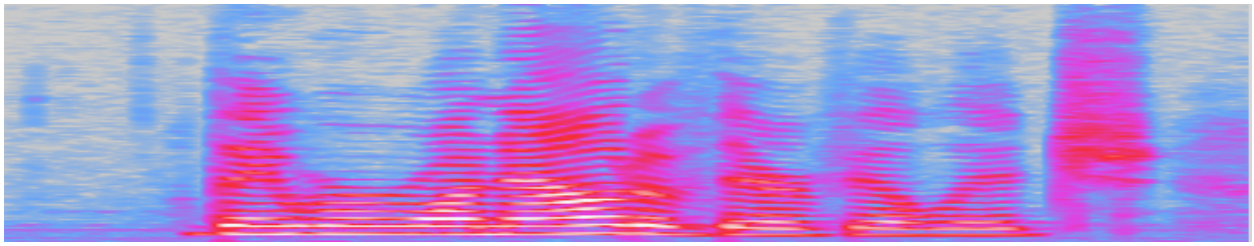


FIGURE 4.3: Spectrogram of "THEY WERE RUN OUT OF THEIR VILLAGE"

4.4 Pre-processing

The pre-processing is the first step before the data feed in the model. It has an feature extraction with MFCC phase to extract the relevant parts of the signal (audio). Then a transformation into specific Tensorflow file format (TFreacord).

4.4.1 Features Extraction

Features Extraction is function to extract MFCC features. In Python, It returns a Numpy array containing features. Each row contains one feature vector :

```
mfcc_feat = mfcc(audio_data, sample_rate, winlen=0.025,
            winstep=0.01,numcep=13, nfilt=26, nfft=512,
            lowfreq=0, highfreq=None,preemph=0.97,
            ceplifter=22, appendEnergy=True)
return mfcc_feat
```

These are the parameters used :
audio-data is the time series of the speech utterance.
sample-rate is the sampling rate.
winlen (by default 0.025 mn) is the length of analysis window.
winstep (by default 0.01 mn) is the successive window step.
numcep (by default 13) is the number of spectrum that should returned by the function.
nfilt (by default 26) is the number of filters in the filter bank.
nfft (by default 512) is the size of (FFT).
lowfreq (by default 0 in hertz) is the lowest band edge.
highfreq : the sample rate divided by 2, it is the highest band edge.
preemph (by default 0.97) applies a pre-emphasis filter with preemph as the coefficient.
ceplifter (by default 22) applies a lifter to the final spectral coefficients.
appendEnergy (by default true) is the zeroth spectral coefficient is replaced with the log of the total frame energy.
Mfcc return matrix of 13 colomn contain the feature of audio signal as shown in Listing 4.1.

Listing 4.1: **MFCC matrix of one signal audio**

```
[[ -7.01220528  -9.1938759   -2.48128829 ...,  18.36870741   9.45484062
   -2.7815733 ]
 [ -7.14785254  -8.73846392  -3.46921152 ...,  17.59393788   5.92525864
   -5.22633879]
 [ -7.16551111  -7.36665283  -1.67388437 ...,  15.42741432   9.57715763
   -1.70454635]
 ...,
 [ -6.87677471 -13.56657302 -12.2033174  ...,  10.76778892  12.59400282
```

```
    1.05247423]
 [ -6.91148682 -10.70038877  -8.22378769 ...,   8.71622965   9.98541256
   -0.34180357]
 [ -7.06626486 -11.47082186  -6.50133507 ...,  11.39820198   9.49033028
    3.10762377]]
```

4.4.2 Transformation of Data

The data transformation is done by transforming the Numpy tensor format resulted by feature extraction step into TFRecords file format.
TFRecord is file that stores the data as binary sequences. It is a fast and powerful way of feeding data to a tensorflow model like a complex Deep Network Architecture. It is useful for the data that are too large to be stored in memory by loaded only the data that are required by time (a batch). It handles a complex training dataset structure in a single record file.

4.5 Learning

In our experimentation we choose DeepSpeech2 based on deep learning end-to-end architecture. We defined set of the hyperparameters for training the model :
The number of RNN layers is `num-rnn-layers = 3`, and the type of RNN is `rnn-type = bi-dir`, and the maximum number of step is `max-steps = 30000`, and the Initial learning rate `initial-lr = 0.0003`.

4.5.1 DeepSpeech2

DeepSpeech2 or (DS2) (Amodei et al. [2016]) is an end-to-end deep learning ASR system. DS2 achieves an high performance according to its architecture that allows to learn from large data set. DeepSpeech2 capacity augments by adding more depth.
DS2 architecture consist of 11 layers of many bidirectional recurrent, convolutionl and fully connected layers. Furthermore an impact of batch normalisation is applied to RNN.

DS2 pipeline (Figure 4.4) begin by the spectrogram as input features. Then applied convolution operation followed by a dropout function to avoid the overfitting problem. The output function of convolutional layer will be reshaped to be suitable with the data into RNN layer (bidirectional or unidirectional) followed by a dropout. After that one or more fully connected layers are applied, and the output is a softmax function that produce a probability distributions over character. thereafter, CTC loss function was performed.

4.5.2 Training

The train phase came after extracting the features. It consists of compute the CTC loss function by given the pair (x,y) and the parameter of the networks and derivatives (compute

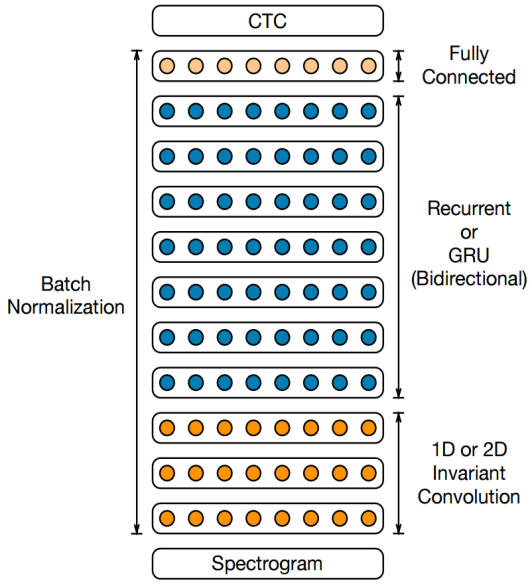


FIGURE 4.4: Architecture of deepSpeech2

gradient descent) used to update parameters through the backpropagation through time. we can summarize the main steps of training as follows :

— The first step is to setup the learning rate from the hyperparameter :

```
learning_rate = set_learning_rate():
```

— Then the creating of an optimizer that performs the gradient descent :

```
optimizer = tf.train.AdamOptimizer(learning_rate)
```

— get the features, labels and sequence lengths from a queue :

```
data = fetch_data()
```

— Apply the gradients to adjust the shared variables.

```
apply_gradient_op = optimizer.apply_gradients(grads,
global_step=global_step)
```

— Group all updates into a single train op :

```
train_op = tf.group(apply_gradient_op, variables_averages_op)
```

— Build summary op which grads are the average gradient :

```
summary_op = add_summaries(summaries, learning_rate, grads)
```

— Initialization of variables :

```
if ARGS.checkpoint is not None:
    global_step = initialize_from_checkpoint(sess, saver)
else:
    sess.run(tf.initialize_all_variables())
```

— Start the queue runners.

```
tf.train.start_queue_runners(sess)
```

— Finally, the function Run training loop (run-train-loop) train the model for required number of steps, which the parameter `saver` used to save and restore variables to and from checkpoints.

```
run_train_loop(sess,(train_op, loss_op, summary_op),saver)
```

. The training phase of our model takes seven days to finish 30000 steps. The loss is changes over the numbers of steps as depicted in Figure 4.5 and according to the adjustment of the learning rate.

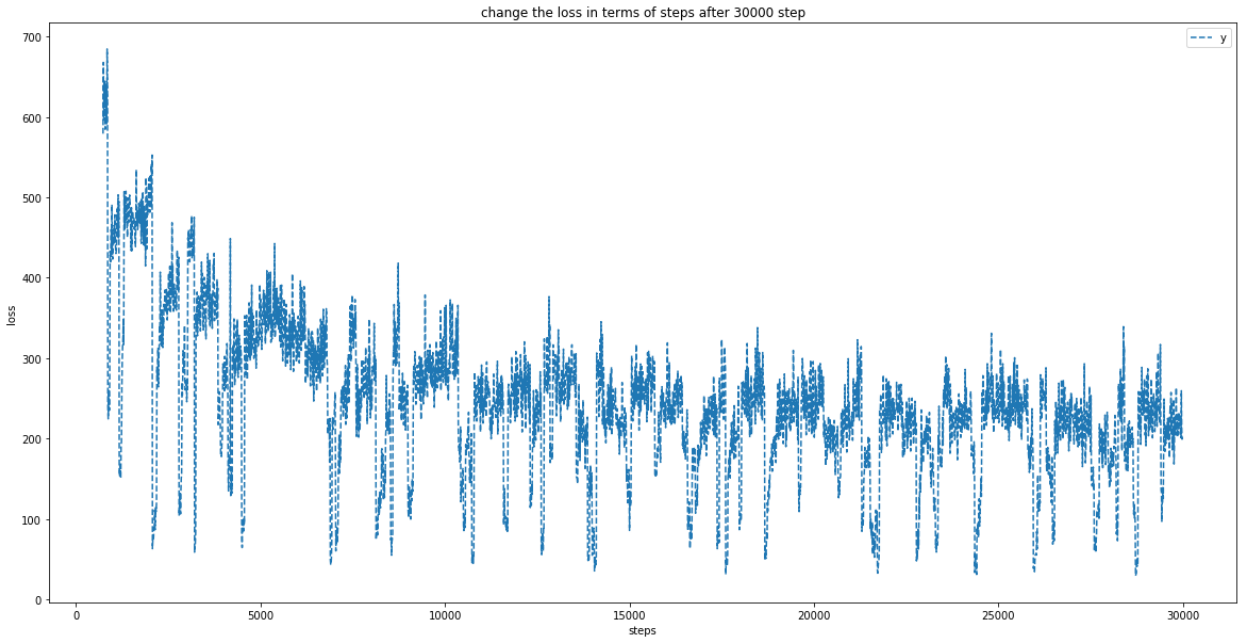


FIGURE 4.5: Change the loss in term of steps after 30000 steps

4.6 Performance

The test phase consists to predict transcription of the unseen utterances inputs from a test set. the thereafter, it evaluated the performance of the model using the measure of character error rate. The model records around 27% of error rate.
The table 4.1 represents the transcription of some speech predicted by the model vs the truth transcription in the test set.

4.7 Discussions

The result achieved by our trained model is not preformed as state of the art models. This can be justified by our limited hardware environment and the limited time accorded to our research. The result can be ameliorated by adjusting some hyperparameters as the training steps and the learning rate,...We can also consider important training sets. Nevertheless, we can mention that our character based model (DS2) is still able to learn implicit languages.

Table 4.1: Truth vs predicted transcription.

Truth transcription	**Predicted transcription**
THEY WERE RUN OUT OF THEIR VILLAGE	THEY WERE ON OUT OF THEIR VELLIGE
I COULD NOT HELP MY FRIEND	I COULD NOT HALD MY FREND
SO HE'S A FRIEND OF YOURS EH	SO HES A FRIEND OF YOURS ANY
I HAVE ONE GREAT PRIVILEGE	IND OR ONE GREAT PRIPLICTS

An other path to follow in order to ameliorate the performance of the model is to incorporate a Language Model.

4.8 Conclusion

In this chapter we have conducted an experimentation of speech recognition system based on DeepSpeech2 architecture. We have first trained the DS2 model using the LibriSpeech data set. Once the model trained, we performed the test step on the test data set. We have obtained around 27% CER that can be ameliorated by considering either hyperparameter adjustment, or more training set or even incorporating language model.

General Conclusions

Our thesis focus on sequences modeling by considering deep learning approach. We have targeted speech recognition systems, as they capture a great of researcher.

To tackle the subject, we have started by investigating artificial neural networks as well as deep learning approach with its Convolutional and recurrent architectures.

We have then presented the state of the art of the speech recognition system, starting by the Hidden Markom Model (HMM) until the emergence of end-to-end system.

In the implementation level we have conducted experiments on an end-to-end model (DeepSpeech2) with LibriSpeech dateset of read English speech.

Once the model trained, the test step leads to character error rate of around 27%.
This not satisfactory result can be argued. to our limited hardware environment and the limited accorded to our research.
In the perspective of ameliorating the developed ASR we propose to adjust some hyperparameter as the training steps and the learning rate. We can also consider more important data set.
Furthermore, we propose also to incorporate language model to augment the performance of our system.

Bibliographie

O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10) :1533–1545, 2014.

N. L. R. Alain Tapp, Yoshua Bengio. Introduction à l'apprentissage profond, 2019.

D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, E. Elsen, J. Engel, L. Fan, C. Fougner, T. Han, A. Y. Hannun, B. Jun, P. LeGresley, L. Lin, S. Narang, A. Y. Ng, S. Ozair, R. Prenger, J. Raiman, S. Satheesh, D. Seetapun, S. Sengupta, Y. Wang, Z. Wang, C. Wang, B. Xiao, D. Yogatama, J. Zhan, and Z. Zhu. Deep speech 2 : End-to-end speech recognition in english and mandarin. *CoRR*, abs/1512.02595, 2015. URL `http://arxiv.org/abs/1512.02595`.

D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, et al. Deep speech 2 : End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182, 2016.

M. A. Anusuya and S. K. Katti. Speech recognition by machine, A review. *CoRR*, abs/1001.2267, 2019.

I. Arel, D. C. Rose, T. P. Karnowski, et al. Deep machine learning-a new frontier in artificial intelligence research. *IEEE computational intelligence magazine*, 5(4) :13–18, 2010.

I. Bakker. *Python deep learning cookbook : over 75 practical recipes on neural network modeling, reinforcement learning, and transfer learning using Python*. Packt Publishing, Birmingham, UK, 2017. ISBN 978-1787125193.

L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *Ann. Math. Statist.*, 37(6) :1554–1563, 12 1966. doi : 10.1214/aoms/1177699147.

P. Blunsom. Hidden markov models. *Lecture notes, August*, 15(18-19) :48.

M. Bouaziz. *Recurrent neural networks for sequence classification in parallel TV streams*. Theses, Université d'Avignon, Dec. 2017.

K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation : Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2019.

J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. Attention-based models for speech recognition. *CoRR*, abs/1506.07503, 2019.

M. Copeland. What's the difference between artificial intelligence, machine learning, and deep learning ? `https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/`, 2016.

F. J. Damerau. *Markov models and linguistic theory*, volume 95. Walter de Gruyter GmbH & Co KG, 1971.

K. Davis, R. Biddulph, and S. Balashek. Automatic recognition of spoken digits. *Journal of the Acoustical Society of America*, 24(6) :637–642, 1952.

S. B. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *ACOUSTICS, SPEECH AND SIGNAL PROCESSING, IEEE TRANSACTIONS ON*, pages 357–366, 1980.

L. Deng. Three classes of deep learning architectures and their applications : A tutorial survey. 2019.

L. Deng and D. Yu. Deep learning : Methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4) :197–387, 2014. ISSN 1932-8346. doi : 10.1561/2000000039.

W. Di, A. Bhardwaj, and J. Wei. *Deep Learning Essentials : Your Hands-on Guide to the Fundamentals of Deep Learning and Neural Network Modeling*. Packt Publishing, 2018. ISBN 1785880365, 9781785880360.

G. D. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3) :268–278, March 1973. ISSN 0018-9219. doi : 10.1109/PROC.1973.9030.

Y. Gao and D. Glowacka. Deep gate recurrent neural network. 04 2019.

G. Gelly. *Speech processing using recurrent neural networks*. Theses, Université Paris-Saclay, Sept. 2017.

W. Ghai and N. Singh. Literature review on automatic speech recognition. *International Journal of Computer Applications*, 41 :42–50, 03 2019. doi : 10.5120/5565-7646.

A. Graves and N. Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International conference on machine learning*, pages 1764–1772, 2014.

A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification : Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 369–376, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi : 10.1145/1143844.1143891.

A. Graves, A. Mohamed, and G. E. Hinton. Speech recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778, 2013.

D. O. Hebb. The organization of behavior ; a neuropsycholocigal theory. *A Wiley Book in Clinical Psychology.*, pages 62–78, 1949.

G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29, 2012.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8) :1735–1780, Nov. 1997. ISSN 0899-7667. doi : 10.1162/neco.1997.9.8.1735.

J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79 :2554–8, 05 1982a. doi : 10.1073/pnas.79.8.2554.

J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8) :2554–2558, 1982b.

B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue, F. A. Mujica, A. Coates, and A. Y. Ng. An empirical evaluation of deep learning on highway driving. *CoRR*, abs/1504.01716, 2015.

C. Jimenez Romero. *A Heterosynaptic Spiking Neural System for the Development of Autonomous Agents.* PhD thesis, 11 2019.

A. Karpathy. The Unreasonable Effectiveness of Recurrent Neural Networks, May 2019.

T. Kohonen. Correlation matrix memories. *IEEE transactions on computers*, 100(4) :353–359, 1972.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

S. E. Krüger, M. Schafföner, M. Katz, E. Andelic, and A. Wendemuth. Support vector machines as acoustic models in speech recognition. In *In 33rd German Annual Conference on Acoustics (DAGA*. Citeseer, 2007.

M. Lamons. *Python deep learning projects : 9 projects demystifying neural network and deep learning models for building intelligent systems.* Packt Publishing, Birmingham, UK, 2018. ISBN 978-1788997096.

Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10) :1995, 1995.

Y. LeCun, Y. Bengio, and G. E. Hinton. Deep learning. *Nature*, 521(7553) :436–444, 2015. doi : 10.1038/nature14539.

N. Manaswi. *Deep learning with applications using Python : chatbots and face, object, and speech recognition with TensorFlow and Keras*. Apress, Berkeley, CA, 2018. ISBN 978-1-4842-3516-4.

Manjutha, Gracy, D. P. Subashini, and D. M. Krishnaveni. Automated speech recognition system – a literature review. 2019.

S. Mason. Microscope uses artificial intelligence to find cancer cells more efficiently, 2019.

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

S. Mdhaffar, A. Laurent, and Y. Estève. Etude de performance des réseaux neuronaux récurrents dans le cadre de la campagne d'évaluation Multi-Genre Broadcast challenge 3 (MGB3). In *Proc. XXXIIe Journées d'Études sur la Parole*, pages 169–177, 2018. doi : 10.21437/JEP.2018-20.

M. R. Minar and J. Naher. Recent advances in deep learning : An overview. *CoRR*, abs/1807.08169, 2018.

M. Minsky and S. Papert. An introduction to computational geometry. *Cambridge tiass., HIT*, 1969.

C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. Activation functions : Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv :1811.03378*, 2018.

V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. Librispeech : an asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio. How to construct deep recurrent neural networks. 12 2019a.

R. Pascanu, T. Mikolov, and Y. Bengio. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063, 2019b.

V. Radha. A review on speech recognition challenges and approaches. 2019.

E. Rodríguez, B. Ruíz, Á. García-Crespo, and F. García. Speech/speaker recognition using a hmm/gmm hybrid model. In J. Bigün, G. Chollet, and G. Borgefors, editors, *Audio- and Video-based Biometric Person Authentication*, pages 227–234, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg. ISBN 978-3-540-68425-1.

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

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. Mcclelland, editors, *Parallel Distributed Processing : Explorations in the Microstructure of Cognition, Volume 1 : Foundations*, pages 318–362. MIT Press, Cambridge, MA, 1986.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3) :211–252, 2015. doi : 10.1007/s11263-015-0816-y.

P. Saini and P. Kaur. Automatic speech recognition : A review. *International Journal of Engineering Trends and Technology*, 4(2) :1–5, 2013.

S. Saksamudre, P. Shrishrimal, and R. Deshmukh. A review on different approaches for speech recognition system. *International Journal of Computer Applications*, 115 :23–28, 04 2015. doi : 10.5120/20284-2839.

A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani. Deep reinforcement learning framework for autonomous driving. *CoRR*, abs/1704.02532, 2017.

K. Samudravijaya. Automatic speech recognition. *Tata Institute of Fundamental Research Archives*, 2004.

M. Schuster and K. Paliwal. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45 (11) :2673–2681, Nov. 1997. ISSN 1053-587X. doi : 10.1109/78.650093.

M. Stenman. Automatic speech recognition an evaluation of google speech, 2019.

D. Stutz. Understanding convolutional neural networks. *InSeminar Report, Fakultät für Mathematik, Informatik und Naturwissenschaften Lehr-und Forschungsgebiet Informatik VIII Computer Vision*, 2019.

C. Touzet. *les réseaux de neurones artificiels, introduction au connexionnisme.* EC2, 1992.

S. Varma and S. Das. *Introduction to Deep Learning.* 2019.

B. Widrow and M. E. Hoff. Adaptive switching circuits. Technical report, Stanford Univ Ca Stanford Electronics Labs, 1960.

V. Zocca. *Python deep learning : next generation techniques to revolutionize computer vision, AI, speech and data analysis.* Packt Publishing, Birmingham, 2017. ISBN 978-1786464453.