

University of “Politehnica” of Bucharest
Faculty de Electronics, Telecommunications and Information Technology

Automated lip reading system using deep

Diploma thesis

Submitted in partial fulfillment of the requirements for the Degree of *Engineer* in the domain of *Electronics and Telecommunications, study program Applied Electronics (ETC- ELAeng)*

Scientific coordinator:
Prof. Dr. Eng. Bogdan Emanuel IONESCU

Student:
Andreea- Mihaela MUNTEANU

University "Politehnica" of Bucharest
Faculty of Electronics, Telecommunications and Information Technology
Department **EAIT**

Anexa 1

DIPLOMA THESIS
of student **MUNTEANU Gh. Andreea-Mihaela , 441F**

1. Thesis title: Automated lip reading using deep learning

2. The student's original contribution will consist of (not including the documentation part):

The project objective is to create an automated lip reading software technology. The original contribution will be represented by the development of techniques for the analysis and processing of video spatio-temporal information and machine learning via deep learning networks. The developed system should be able to automatically detect a person's mouth and lips and learn based on specific movements to translate them into words. The undergoing work will include: the overview of the current state of the art in the field bibliographic synthesis of the achievements that have been done until now in the related field, simulation and experimental validation using real data (e.g using Matlab, Python, C++ or Java environments), analysis of the result, concluding on the performance of the developed techniques and identifying future challenges.

3. Pre-existent materials and resources used for the project's development:

Matlab, Python, C++ or Java environments

4. The project is based on knowledge mainly from the following 3-4 courses:


Computer Programming, Data Structures and Algorithms, Oriented Object Programming, Medical Imagistic

5. The Intellectual Property upon the project belongs to: U.P.B.

6. Thesis registration date: 2017-11-29 11:25:18

Thesis advisor(s),

Prof. dr. ing. Bogdan Emanuel IONESCU

signature: 

Student,

signature: 


Department director,

Prof. dr. ing. Sever PAȘCA

signature: 

Dean,

/ Prof. dr. ing. Cristian NEGRESCU

signature: 

Validation code: **1c887d8a97**

Statement of Academic Honesty


I hereby declare that the thesis “*Automated lip reading using deep learning*”, submitted to the Faculty of Electronics, Telecommunications and Information Technology in partial fulfillment of the requirements for the degree of Engineer in the domain Electronics and Telecommunications, study program Applied Electronics (ETC-ELAeng), is written by myself and was never before submitted to any other faculty or higher learning institution in Romania or any other country.

I declare that all information sources I used, including the ones I found on the Internet, are properly cited in the thesis as bibliographical references. Text fragments cited “as is” or translated from other languages are written between quotes and are referenced to the source. Reformulation using different words of a certain text is also properly referenced. I understand plagiarism constitutes an offence punishable by law.

I declare that all the results I present as coming from simulations or measurements I performed, together with the procedures used to obtain them, are real and indeed come from the respective simulations or measurements. I understand that data faking is an offence punishable according to the University regulations.

Bucharest, 18.06.2018

Andreea- Mihaela Munteanu



(student's signature)

Content

Content.....	7
List of figures	9
List of tables.....	11
List of acronyms	13
Introduction.....	15
Chapter 1. Theoretical body.....	17
1.1 Fundamentals of machine learning	17
1.2 Introduction to deep learning	18
1.3 Overview of neural networks	21
1.4 Convolutional neural networks (CNNs)	23
1.5 Recurrent neural networks (RNNs)	25
1.6 Overview of lip reading systems.....	27
1.7 Description of mouth detection algorithms.....	30
Chapter 2. Proposed method	33
2.1 Necessary tools	34
2.2 Architecture of the system	35
2.3 Initial parameters	38
2.4 Training parameters	39
2.5 Constraints imposed by Romanian language	41
2.6 Sentence level prediction	41
Chapter 3. Experimental results	43
3.1 Description of the datasets	43
3.2 Performance metrics	45
3.3 Results obtained for English	47
3.4 Results obtained for Romanian	57
3.5 Conclusions of the experimental results	68
Chapter 4. Conclusions and future perspectives	73
Bibliography	75
Annexes	77

List of figures

Figure 1.1. Learning system model, pag 17

Figure 1.2. Comparison between Deep learning and older learning algorithms, taking into account the amount of data and the resulted performances, pag. 19

Figure 1.3. Comparison between machine learning and deep learning systems, from the workflow's point of view, pag. 19

Figure 1.4. Structure of an artificial neural networks, including the main three components: weights, thresholds, activation function, pag. 22

Figure 1.5. Structure of an artificial neural network, including input, hidden, output layers, pag. 22

Figure 1.6. Dimensions of CNN, as visualized on one of the layers, pag. 24

Figure 1.7. Fully recurrent neural network, pag. 26

Figure 2.1. Flow chart of a lip reading system, pag. 33

Figure 2.2. Outside approach of LipNet system, pag. 35

Figure 2.3. High level architecture of the system, including the raw data and the 2 main important blocks of LipNet, pag. 35

Figure 2.4. Architecture of the lip segmentation block, including both components DLib face detector and iBug face shape predictor, pag. 36

Figure 2.5. Flow of the Lip segmentation part, pag. 37

Figure 2.6. Architecture of the lip reading processing block. The inputs of the block are images that are processed using neural networks, pag. 37

Figure 3.1. Structure of the GRID, pag. 43

Figure 3.2. Representation of mean CER for unseen speakers protocol, English language. Y axis: value of mean CER, X axis: Epoch number, pag. 48

Figure 3.3. Representation of mean WER for unseen speakers protocol, English language. Y axis: value of mean WER, X axis: Epoch number, pag. 48

Figure 3.4. Representation of mean BLEU for unseen speakers protocol, English language. Y axis: value of mean BLEU, X axis: Epoch number, pag. 48

Figure 3.5. Representation of normalized values of CER, WER, BLEU for unseen speakers protocol, English language. Y axis: value of the normalized parameters, X axis: Epoch number, pag. 48

Figure 3.6. Representation of mean CER for unseen speakers curriculum protocol, English language. Y axis: value of mean CER, X axis: Epoch number, pag. 51

Figure 3.7. Representation of mean WER for unseen speakers curriculum protocol, English language. Y axis: value of mean WER, X axis: Epoch number, pag. 51

Figure 3.8. Representation of mean BLEU for unseen speakers curriculum protocol, English language. Y axis: value of mean BLEU, X axis: Epoch number, pag. 52

Figure 3.9. Representation of normalized values of CER, WER, BLEU for unseen speakers curriculum protocol, English language. Y axis: value of the normalized parameters, X axis: Epoch number, pag. 52

Figure 3.10. Representation of mean CER for random split protocol, English language. Y axis: value of mean CER, X axis: Epoch number, pag. 53

Figure 3.11. Representation of mean WER for random split protocol, English language. Y axis: value of mean WER, X axis: Epoch number, pag. 54

Figure 3.12. Representation of mean BLEU for random split protocol, English language. Y axis: value of mean BLEU, X axis: Epoch number, pag. 55

Figure 3.13. Representation of normalized values of CER, WER, BLEU for random split protocol, English language. Y axis: value of the normalized parameters, X axis: Epoch number, pag. 56

Figure 3.14. Representation of mean CER for unseen speakers protocol, Romanian language. Y axis: value of mean CER, X axis: Epoch number, pag. 56

Figure 3.15. Representation of mean WER for unseen speakers protocol, Romanian language. Y axis: value of mean WER, X axis: Epoch number, pag. 59

Figure 3.16. Representation of mean BLEU for unseen speakers protocol, Romanian language. Y axis: value of mean BLEU, X axis: Epoch number, pag. 60

Figure 3.17. Representation of normalized values of CER, WER, BLEU for unseen speakers protocol, Romanian language. Y axis: value of the normalized parameters, X axis: Epoch number, pag. 60

Figure 3.18. Representation of mean CER for unseen speakers curriculum protocol, Romanian language. Y axis: value of mean CER, X axis: Epoch number, pag. 61

Figure 3.19. Representation of mean WER for unseen speakers curriculum protocol, Romanian language. Y axis: value of mean WER, X axis: Epoch number, pag. 62

Figure 3.20. Representation of mean BLEU for unseen speakers curriculum protocol, Romanian language. Y axis: value of mean BLEU, X axis: Epoch number, pag. 63

Figure 3.21. Representation of mean CER for random split protocol, Romanian language. Y axis: value of mean CER, X axis: Epoch number, pag. 64

Figure 3.22. Representation of mean WER for random split protocol, Romanian language. Y axis: value of mean WER, X axis: Epoch number, pag. 64

Figure 3.23. Representation of mean BLEU for random split protocol, Romanian language. Y axis: value of mean BLEU, X axis: Epoch number, pag. 65

Figure 3.24. Representation of normalized values of CER, WER, BLEU for random split protocol, Romanian language. Y axis: value of the normalized parameters, X axis: Epoch number, pag. 66

Figure 3.25. Representation of normalized values of CER, WER, BLEU for a small dataset, Romanian language. Y axis: value of the normalized parameters, X axis: Epoch number, pag. 66

Figure 3.26. Representation of normalized values of CER, WER, BLEU for a fix number of sentences in the dataset, Romanian language. Y axis: value of the normalized parameters, X axis: Epoch number, pag. 68

List of tables

Table 2.1. Values of mean and standard deviation per channel, pag. 38

Table 2.2. Hyper- parameters used for the initialization of the system, pag. 39

Table 3.1. Words used for building GRID dataset, pag. 44

Table 3.2. Words of the CAMPUS corpus, split into speaking parts, pag. 45

Table 3.3 Real and decoded sequences for unseen speakers protocol, English language, pag. 50

Table 3.4 Real and decoded sequences for unseen speakers curriculum protocol, English language, pag. 53

Table 3.5 Real and decoded sequences for random split protocol, English language, pag. 56

Table 3.6. Real and decoded sequences for unseen speakers protocol, Romanian language, pag. 61

Table 3.7. Real and decoded sequences for unseen speakers curriculum protocol, Romanian language, pag. 63

Table 3.8. Real and decoded sequences for random split protocol, Romanian language, pag. 66

Table 3.9. Real and decoded sequences for small dataset, Romanian language, pag. 67

Table 3.10. Real and decoded sequences for a fix number of sentences in the dataset, Romanian language, pag. 68

List of acronyms

3D - Three dimensional
AAM - Active Appearance Model
ASM - Active Shape Model
Bi-GRU - Bidirectional gated recurrent unit
BLEU - Bilingual evaluation understudy
CER - Character error rate
CPU - Central processing unit
CTC - Connectionist temporal classification
GPU - Graphics processing unit
GRU - Gated recurrent unit
NLP - Natural language processing
NLTK - Natural language toolkit
PCA - Principal component analysis
ReLU - Rectifier linear unit
RGB - Red, green, blue model
ROI - Region of interest
STCNN - Spatio-Temporal CNN
WER - Word error rate

Introduction

To begin with, computer vision is considered an interdisciplinary field that has very diverse approaches in order to develop new systems, with a wide range of applications. Using image processing, it seeks to automate activities that are usually done by human beings, but it is very challenging due to the high accuracy required for predictions.

Lip reading is a technique that is used in order to decode text by visually interpreting the movements of the mouth and face, without hearing speaker's voice. Being considered a complement of audio based speech recognition, it is an area of growing attention, where the diversity of different approaches led to the development of new systems. The investigation of this field started around 1985, when E.D. Pertjan began to analyze the subject for his PhD [1], but it is still a very challenging and ambiguous task, at the word level, due to homophones, that produce exactly the same lip sequence, depending on the language that is taken into account. On the other hand, there is a holistic approach [2] that trains the system using the whole word, instead of the visemes, but it has some constraints due to the large number of words that should be used, in order to make the system reliable.

Lip reading is the field where speech recognition mixes with computer vision and comes with a new point of view, which completes the past experiments in both areas. It deals with visual domains of speech and involves image processing, artificial intelligence, pattern recognition and statistical modelling. Traditional approaches separated the problem into two stages, designing or learning the visual features and prediction, but more recently trials came with an end-to-end trainable system, that uses deep learning. This new possibility, that became recently available, uses deep neural network models and it can access large scale datasets for training.

However, end-to-end lip reading has a more limited background, with few published results. All of them are for English, reason why their reinterpretation for any other language is more difficult, because of the high number of phonemes that it might have. In all cases, there is a mechanism that is applied to the region of interest (eg., mouth region), which pays special attention to it and then the model is trained end-to-end. While the first attempts used simple neural networks, the later trials came with more complex systems, that include convolutional [3] and recurrent networks.

Starting from the idea that humans understand speech not only by hearing, but also by seeing lips' movements [4], a visual only lip-reading system proposes to read the words said by a speaker, only by analyzing his mouth. The information is also relied on the context, knowledge of language or any other person that could appear in the video. Also, the existence of different phonemes that are pronounced the same or special letters that are found only in one language must be taken into consideration.

In this context, the main scope of the project is to create a machine that can be used exclusively for Romanian language. It proposes to be an end-to-end system, that create a model for the sentence-level, analyzing at the same time, spatial-temporal features, because both position and motion are important. The challenge come from the high number of visemes that exists in Romanian, reason why the movements of the lips look almost the same. Moreover, it has six phonemes that cannot be translated in other language, fact that leads to another issue, because the system does not know how to interpret those movements of the lips.

Audio speech recognition has been an area of interest for several years due to the wide area of applications that it has. However, the development of these type of systems got to a saturation,

because of some limitations that exist and cannot be solved, such as noisy environments. Furthermore, there are people that cannot use these machines because they are deaf or hard-hearing. This is one of the reasons that determined researchers to find another approach of the speech recognition, that could work either as a complement of the audio based one, or independently, for other applications.

Visual lip reading systems can play a very important role in the life of human being, as well as in the human-computer interaction. Using them, researchers have already created audio-visual speech recognition machines (AVSR), but they, also, have potential when taking into account speaker recognition, talking heads or sign language. While the first thought for the applications of these machines goes to deaf or hard hearing people, the second one should analyze the possibility of including them into video-surveillance systems, in order to detect any unusual conversation. Moreover, visual lip reading techniques aim to help hearing-impaired people feel more comfortable and integrate them in social activities, without finding any inconvenient. From another point of view, these systems can be used for silent movies, in order to transcribe the actor's speech.

The thesis will present the theoretical background, that had been studied in order to understand better and optimize the system. It has a brief description about deep learning and it presents the role of neural networks and the importance of features extraction in the project. Also, it includes the proposed approach, with all the attempts that I have done in order to get the best result, and all the experimental results that led to the existing system. Not least, there will be presented few perspectives for the future, that could help me improve the model or develop it for another language.

All in a nutshell, lip reading techniques represent an area of growing attention. Even if for Romanian language the development of such systems might include some challenges, the proposed method aims to solve them and create a functional system that can be used for both research and society.

Chapter 1. Theoretical body

1.1 Fundamentals of machine learning

Learning process is a very complex phenomenon that includes the acquisition of new knowledge and the development of new skills either by studying them, or by practicing them. Since computers were invented, researchers have been trying to include these capabilities in them, such that this problem has been described many times as the most challenging task that they ever had. Artificial intelligence came with a solution that is not complete yet, but, however, it is already able to create computers with intelligent behavior.

Machine learning is one of the branches that artificial intelligence has and it aims to understand the structure of data, in order to create models of learning processes that include the multiple manifestations of this complex subject. It has been evolved from the study of pattern recognition, around 50 years ago, and, nowadays, it explores the studying and building of algorithms that are able to learn using datasets and, then, can make predictions. It has as a main purpose to help software solutions be more accurate, without being programmed by human beings.

Even if machine learning is a field that uses computer science, it is different from traditional approaches. More exactly, it lets computers use data to train the system and, then, with the help of statistical analysis, it gets an output value. This is the main reason why machine learning encourages computers to build models that are used later to automatically make decisions, based on the inputs. Figure 1.1 highlights how a learning system looks, such that it is easy to observe that training data is useful, in order to obtain, depending on the learning method, a model that can be used later on for testing. The system is, in fact, an algorithm that has certain features, which depend on both, the task that has to solve, and the performances that are required. The learning method depends on the application, but, also, on the dataset that is used for training. Input samples are usually represented by a large dataset, that includes useful information for the system, that offers support in order to learn different features that can be used later on for testing.

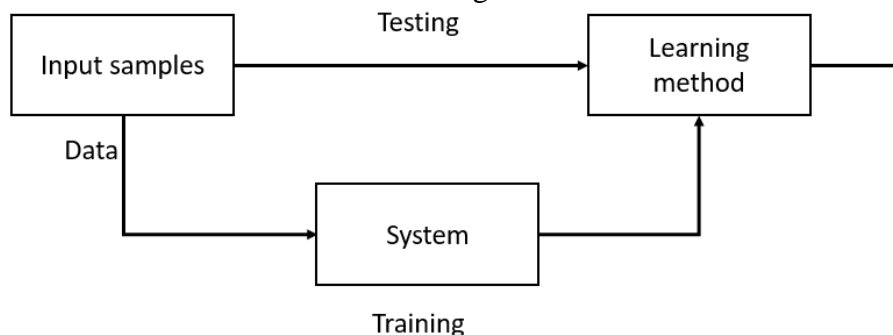


Figure 1.1. Learning system model

Machine learning has been, since its appearance, a field of big interest for researchers. While at the beginning of the development, researchers used to work with everything as a whole, the increase

of challenges that have been approached led to a change. So, nowadays, machine learning is a field organized [5] using three primary research foci:

- Task oriented - studies, known also as the “engineering approach” [5], focus on the development and analysis of any learning system, to improve the performance in a set of tasks that is predetermined;
- Cognitive simulations focus on the investigation and simulation of human hearing processes;
- Theoretical approach focuses on the theoretical analysis of the possible new approaches, including new learning processes and algorithms;

When trying to get a road map of the machine learning area, it is very important to take into account the criteria that researchers used in order to classify these systems. There are different dimensions that were used for the classification, but the most important are based on the knowledge representation, application of the domain and learning fundamentals. More exactly, any system must analyze the amount of information that is used for learning and how it interferes, taking into account what is happening, also, after the training. So, it is essential to see which are the skills that the learner can acquire after a certain number of iterations.

Starting from the idea that machine learning systems have the ability to automatically learn and improve performance without human’s involvement, there are some categories where they can be placed, depending on the way that learning is received and feedback is sent. The basic advantage is represented by the huge amount of data, but, however, it is very important how the data is classified. From this point of view, machine learning systems [6] use different types of learning:

- **Supervised Learning** is used by the systems whose input and desired output data is known. In this case, all data is labeled;
- **Semi-supervised Learning** is used by the systems that got an incomplete set for training, such that only part of the data is labeled;
- **Active Learning** is used by the systems that are able to obtain a limited number of labels from the dataset;
- **Reinforcement Learning** is used by the systems where training data is used as a feedback for the programs’ actions;
- **Unsupervised Learning** is used by the systems that have all input data unlabeled;

All in all, machine learning is a field based on the development of different algorithms, in order to create models that do not need essentially programming for getting a result but are trained using input data. Depending on developed algorithm [7] and the desired application of the system, researchers took different approaches, in order to obtain better results. For example, decision tree learning uses a decision tree in order to create a predictive model. Also, artificial neural networks, which are inspired from the biological neural networks, represent another approach of machine learning, that is based on a structure of interconnected groups of neurons.

1.2 Introduction to deep learning

Deep learning is a machine learning approach that is based on the idea of learning from examples, using big datasets. The constraints that shallow architectures had, because of the reduced dimensions of the databases, led to the development of a new method, that is able to work with huge

amount of data. The attention to this subject have increased a lot lately, due to the large number of applications that can have, in different areas, being able to get systems such as driverless cars.

Even if it was theoretically mentioned since 1980 [8], it became used only in the recent years, because of the large amount of data that is needed and had to be labeled. It aims to make learning algorithms easier to use and, at the same time, to increase the accuracy of the output results. The big advantage of these kind of systems is highlighted in the Figure 1.2, where deep learning algorithms are compared to older algorithms. The second types of algorithms achieve saturation at a certain point, even if the dataset keeps increasing, so the performance does not improve anymore. In opposition to the traditional approaches, deep learning systems have a correlation between the amount of data and the performance of the system. More exactly, the bigger the dataset is, the better the performance become.

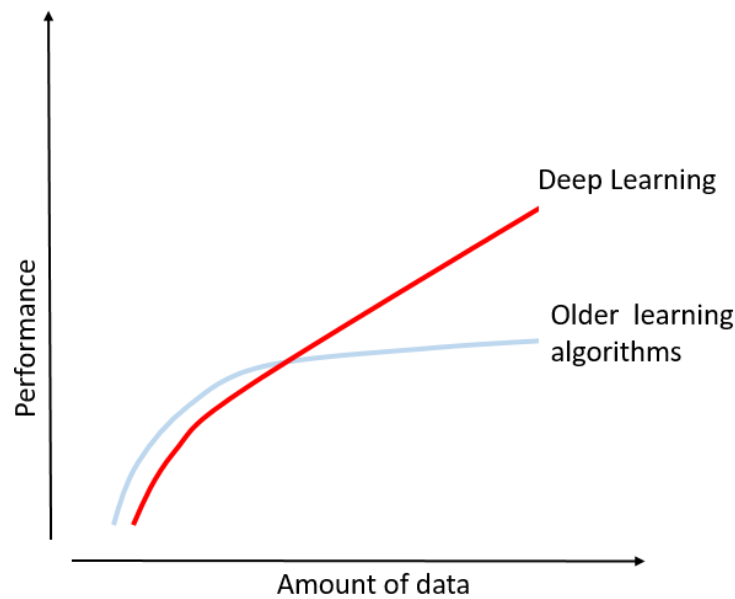


Figure 1.2. Comparison between deep learning and older learning algorithms, taking into account the amount of data and the resulted performances

Deep learning systems use deep neural networks and are, usually, used in various fields such as computer vision, speech and text recognition or natural language processing. The term “deep” does not have anything in common with the idea of a deeper understanding that this approach can offer. It comes from the number of successive hidden layers, that all systems have, which are, usually, more than 100, in comparison to the traditional neural networks that have two-three layers. The number of hidden layers contributes to one of the dimensions that characterize a network, called “depth”. The high number of layers from the neural network determined researchers to create more complex systems, which are able to perform complicated tasks. Because of the structure of any system that uses deep learning, other names of the method are “layered representation learning” or “hierarchical representation learning”.

Models that use deep learning methods are trained and have the ability to learn the features directly from it, without needing any additional extraction. Being a specialized form of machine learning, deep learning performs end-to-end training [9], such that the network is able to process the raw data, classify it and then learn how to do the classification automatically. On the other hand, machine learning algorithms perform two steps: first one extracts manually the features and the second one creates a model using the existing data. This differences can be easily observed in Figure

1.3, where the comparison is done using a real case. The input data is represented by frames, that show different types of automobiles, which are classified by the two methods presented above. It is observable that deep learning systems are more efficient, being able to scale with data.

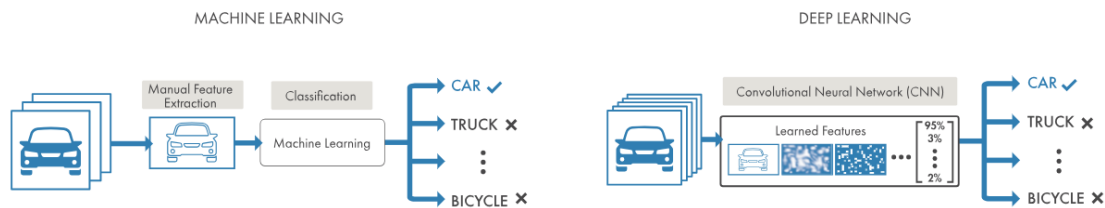


Figure 1.3. Comparison between machine learning and deep learning systems, from the workflow's point of view. Source [10]

The development of deep learning models aims to performs object classification, that can be done in three ways:

- **Training from scratch**, whose purpose is to train the labeled data with the help of a network architectures that is designed to learn features and then create the model. This is usually used for new applications that have a large number of output categories;
- **Transfer learning**, which uses pre-trained models and optimize them. Most of the deep learning applications are developed using this method;
- **Feature extraction**, which is a specialized approach that has a feature extractor. It is less common;

The approach of deep learning that is usually used represents, in fact, a hierarchical learning, where any higher level is formed depending on the lower ones. Each level tends to transfer data into a more abstract representation of it. However, there have been some other attempts to create deep learning systems using the holistic approach, called also the “whole approach”. More exactly, in this case the system has to learn everything as a whole, reason why the amount of needed labeled data is a lot bigger. For example, in the case of text or speech recognition, the text should have had a dataset that included all the words from a certain language.

Deep learning algorithms use more complex models of neurons, that are able to learn. They create neural networks by being structured one on the top of the other, the inspiration of such a structure being, in fact, the brain of the humans. However, deep learning methods are not brain models, they only inspired by the way that we are able to learn as people. It is a subfield of machine learning, which has a different approach about learning representation. It is based on a learning process that uses successive layers, whose meaning is increasing.

Any deep learning system works based on the parametrization of all the layers that the network has. The weights are, in fact, the information that tells what a layer would do to the input, so in this context, learning means finding the right values for all the layers of the network, in order to map correctly the input to the associated target. Therefore, any deep learning system is a sensitive one, because the change on any value influences the behavior of the whole network. However, in order to control the network, developers have to observe and understand it. More exactly, for influencing the output, it is very important to know the distance between the actual output and the desired one. In this case, it is necessary to include a loss function, that would make predictions. The obtained result is used by any deep learning system as a feedback, to adjust the value of the parameters of the network, in order to optimize it.

All in a nutshell, the various fields where deep learning can be applied determine researches keep developing new algorithms. From speech recognition to driverless cars, these machines can find a place and their optimization is essential, such that, people start trusting them more. Even if the concept appeared a long time ago, the usable algorithms are quite new, reason why their optimization is essential. However, in order to make from deep learning the most efficient tool, it is necessary to have a large amount of data, that can be used for training. Also, machines which choose this method require an expensive computational system and more time for training, so they are not recommended for applications that appear on short notice. The performance of deep learning algorithms is highly dependent on the used dataset, reason why another important aspect is represented by the labels, which must be as accurate as possible.

1.3 Overview of neural networks

Artificial neural networks are computing systems that follow the biological neuron model and are able to learn to perform tasks using some examples. They can do this without any prior information about a certain task, but only by being trained using a dataset. The ability to learn of the neural networks represented the most interesting capability that they had. Moreover, artificial neural networks are able to support different pattern recognitions tasks [10], so it is a lot more difficult to derive the logical rules, in order to create a typical software program. In some cases, the models are formed using a learning rule, obtained by varying different parameters. Artificial neural networks are formed by interconnected processing units, that work together for a certain purpose.

Artificial neurons were inspired vaguely from the anatomical model, and they represent the basic element of an artificial neural network. The structure is shown in Figure 1.4 and it consists of three basic elements [11] that are interconnected between them. The first component is represented by the weighting factors, which are associated with each node of the network. Their purpose is to determine how strong an input vector $X = [X_1 \ X_2 \ X_3 \dots, X_n]^T$ is. The product between each input and the associated weight of the neuron connection determines either if the node is excited or inhibited, depending if it is positive or negative. Threshold is the second component of a neuron and it is defined as [11] the magnitude of the offset that affects the activation of the output node. It is subtracted from the sum of the product mentioned before in order to get the result from the output. The last component is the activation function which has different formulas, depending of the type of it. It affects the output signal, by doing a mathematical operation. Some of the most used activation functions are:

- Linear function which satisfies the superposition concept;
- Threshold function which can be either binary, or bipolar;
- Piecewise Linear Function known also as “saturating linear function”;
- Sigmoidal function which is a non-linear function usually used to design neural networks;

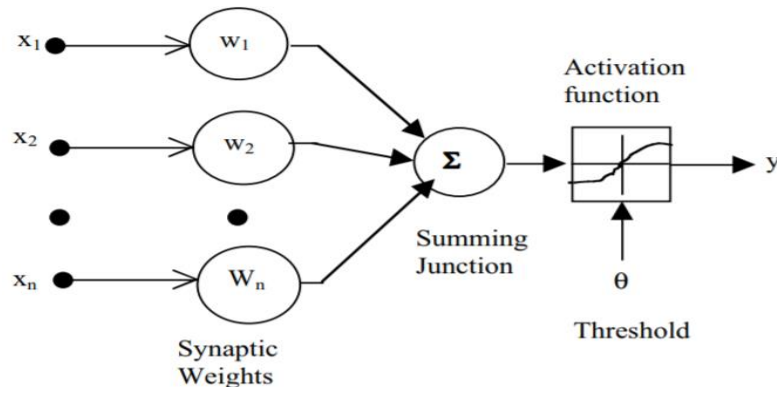


Figure 1.4. Structure of an artificial neural networks, including the main three components: weights, thresholds , activation function. Source [11]

Being, in fact, a mathematical model inspired from the human anathomy, artificial neural networks can be described as [12] a map of input space to the output space. The structure of such a network is shown in Figure 1.5 and it includes three types of layers [11]: input, hidden and output. The input layer can be described as a connection with the environment. Its main purpose is to work only with the input data and transfer it to the hidden layers. Actually, input layer can be considered the condition for which the user does the training of the neural network and the input neurons should represent independent variables that are able to influence the output. Hidden layers are formed by neurons thart have an activation function applied to them. It is an intermediate layer between the input layer and the output one, and its main purpose is to process the input obtained from the previous layer, by extracting the required features. The necessary number of neurons from this layer that determine a good result of the network represented, since the very beginning, a challenge for the researchers, who concluded that it depends on the complexity of the processed data. Moreover, the variable number of hidden layers influences the network, such that the developer has to take into account how many layers are necessary in order to solve the proposed problem. The output layer has the purpose to collect the information given by the previous one and send forward, in the way it has been design from the beginning. Usually, the number of neurons from this layer depends on the applications of the network and it can be considered as a pattern of the input layer.

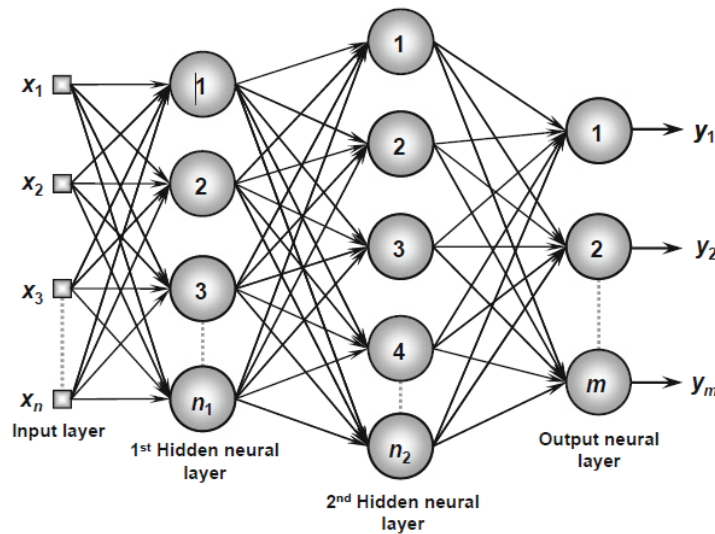


Figure 1.5. Structure of an artificial neural network, including input, hidden and output layers. Source [11]

While the first artificial neural networks were simple and did not have a lot of hidden layers, the development of this field made researchers create more complex structures, which are able to perform more complicated tasks. However, depending on the number of the hidden layers and the complexity of the structure, there have been defined certain types of networks that are now used to build different systems. The ones that have more than three hidden layers are called “deep neural networks” and are used for deep learning applications.

Artificial neural networks (ANN) represented a big step ahead for machine learning field, reason why researchers pay a lot of attention to this area. The multiple applications that could use such a structure were a starting point, but there were other advantages [13] that influenced their positive attitude in regards to it. In traditional programming, information used to be stored in databases, but now, it is kept in the entire network. In consequence, even if part of the data is missed or corrupted, the network is still able to work. Another huge advantage is the ability to work with data which is not complete, such that using an ANN, the output can be produced with incomplete information. However, in this case the performance of the system decreases, depending on the importance of the missing information. Moreover, this ability determines the network to be fault tolerant. ANN is able to learn depending on the examples that are given and the desired output, so the success of such an architecture depends on the selected instances, because in case of showing only parts of the data, it will return false outputs. Usually networks work slower over time and suffer a gradual degradation, but it does not happen immediately. The last important advantage is represented by the power of performing more than one task at a time.

On the other hand, these networks come with some drawbacks [13] that researchers are still trying to solve. For example, they are hardware dependent, by requiring parallel processing power. Furthermore, there is no specific rule in order to determine the structure of an ANN, so it can be optimized only by trying and solving the appeared errors. They have a mathematical explanation and they work with numerical information, such that all data has to be translated into numerical information before being introduced into an ANN. Time efficiency is very important nowadays, but the duration of the training is unknown and it depends on both, size of the dataset and chosen parameters.

All in a nutshell, the main goal of ANN was to solve problems in the way the human brain does. They came with a lot of advantages and few disadvantages, but, however, nowadays they are specialized to perform different specific tasks, that deviated from the initial purpose. ANN can be used for computer vision, speech recognition or medical diagnosis.

1.4 Convolutional neural networks (CNN)

To begin with, deep neural networks are ANNs with a higher number of hidden layers and some different features. They are able to process larger datasets and they are usually used tasks such as image retrieval, action and gesture recognition in image sequences, person re-identification and many more. Convolutional neural networks (CNN) are a class of deep neural networks and can be defined as a variation of multilayer perceptron. The biological process where the connectivity pattern between the neurons reconstruct the animal visual cortex represented the basis of these types of networks. They have been introduced in early 1990's [14] and since the very beginning, they have shown good performances for some tasks such as face detection. CNN demonstrated a high efficiency in understanding image content [15], highlighting the capability to learn image features that were interpretable.

Being very similar with ANNs, they have neurons, with weights and biases. They work almost in the same way, but CNN assume that all the inputs are images, reason why there are certain properties that the architecture has, in order to make the forward function more efficient and to reduce the number of parameters. Unlike ANNs, convolutional neural networks are built in a more sensitive way and they have arranged the neurons following three dimensions: width, height and depth [16], as shown in Figure 1.6. It is important to add that depth does not refer to the whole network, but to the third dimension of the activation function.

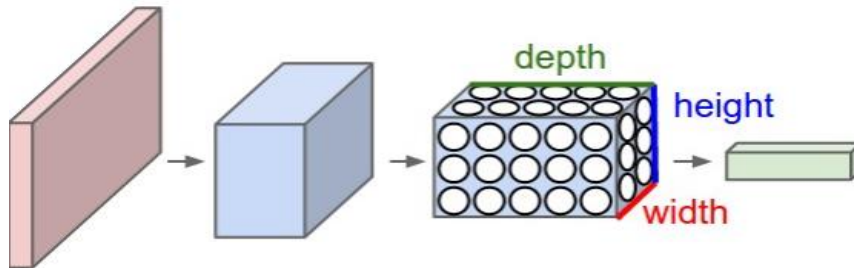


Figure 1.6. Dimensions of CNN, as visualized on one of the layers. Source [16]

CNNs are networks like ANNs, which have input, output and hidden layer. Because of the applications that use deep learning, the number of hidden layers is bigger and usually consists of convolutional layers, pooling layers or fully connected layers. Every image is considered a matrix of pixels, where each pixel can be encoded depending on its bit size. In the case of RGB based images there are three separated channels and CNN introduce an additional data, which constitute the depth, such that the input volume will be three dimensional. Analogous, the output volume will be a 3D network made up of activation neurons.

The convolutional layer has the role to compute the output neurons which are connected to certain regions of the input. It is considered the building block of CNN, doing the most of the computations. The parameters of this layer are represented by [17] the size of the map, the number of maps kernel sizes and the connection table. Each layer has a maps with the same size and the kernel would be shifted over a valid region from the image, such that it will be inside of it. There are also as pune: skipping factors i.e. stride, which etc. which aim to tell to the filter how many pixels would skip in both x and y directions. In order to optimize this layer, researchers started using shared weights, such that the number of unique weights is smaller and, in consequence, the number of performed computations over the matrix.

Pooling layer does a down-sampling along the spatial dimensions and it is usually inserted between successive convolutional layers, in order to reduce the number of parameters and computations in the network, such that overfitting can be controlled. This layer operates on each input slice and it changes the size of it, using different functions, namely MAX or average operation. The result of this operation is given by [16] the maximum activation over non-overlapping rectangular regions of a certain size. It is applied on the spatial dimensions and it will reduce the weight and height of the network, but it will not affect the depth. It works by taking a window that will be moved across the input volume, transforming every value in a representative one. This is an optional layer, but it is useful because the decrease in size will lead to more efficient future computations.

Fully connected layer is usually used in the last stage of the CNN, in order to connect the output layer an build the number of necessary outputs. In this layer, neurons have full connection with the previous layer and their activation can be computed with a matrix multiplication that has afterwards a bias offset. Fully Connected layer uses the features extracted in the previous steps for

classifying the input image into various classes based on the training dataset. It does this using a SoftMax function to transform each feature into a score for each class, after which it takes the max mean score. The main difference when compared to convolutional layer is the fact that number of connected neurons, which is a lot higher for the fully-connect layer. However, both of them calculate a dot product, so their functional form is the same. In consequence, FC layer can be converted to a convolutional layer.

Taking into account the building blocks mentioned before, it is also important to specify the design of the CNN, by mentioning the dimensions of each layer. Spatial dimensions are computed as a function of the size of the input and the parameters mentioned before. Usually input layer should be a multiple of 2 and the convolutional layer uses a small filter, with the stride equal to one, such that it does not affect the spatial dimensions of the input. Stride is chosen to have value one, because in this way the pooling layer can work alone and more efficient, the convolutional layer being responsible only for transforming the input from the depth point of view. Being used for down-sampling, usually pool layer has a 2x2 matrix and perform a max-pooling operation.

Showing outstanding image recognition and classification features, CNNs are even more performant than ANNs, because of their ability to work without needing any pre-processing. While for ANNs there were some hand-engineering actions needed, when using a CNN, the system is able to perform the actions without the help of a human being. They assume the input is an image and all the architecture is built based on that idea. To control their capacity [18], CNN vary the depth and the width and because of the lower number of connections it is easier to train a system using CNNs.

CNNs have a lot of advantages such as automatic learning or sharing weights, but there some drawbacks that determined researchers, at least at the very beginning, to be sceptical in regards to this architecture. So, despite the qualities that were very attractive, it is expensive to apply them on large scale, for images with high resolution. Moreover, they require advanced devices to perform on, with GPUs, whose size constrain the amount of data that can be processed and dictate the time that any training will need.

In conclusion, CNNs are build on an architecture where hidden layers perform specific tasks, assuming that the input is an image. They were considered really helpful since the moment they have appeared, but, they had some drawbacks that researchers tried to solve over time. However, they can be applied in different areas such as computer vision or speech recognition, reason why their improvement still represents a priority.

1.5 Recurrent neural networks (RNN)

Recurrent neural networks are a type of ANN that process data sequentially, having usually variant lengths. They use sequential information, that is temporal modelled, such that they are capable to predict a future value of the input. Once the network is trained, it can work in a generative way. Because of their ability to keep in the internal memory the steps that happened before, RNNs are a real success in NLP tasks, such as handwriting processing or speech recognition.

They offer another approach to correlate data [17] which is not always so closed in a certain sequence, the output of an element being dependent of the previous computations that have been done. They are called “recurrent” because of the fact that each task is performed for every element of the sequence and this is the reason why RNN are considered networks with memory, having the ability to keep the information that have been computed so far. However, even if theoretically the network memorises what have been done before, it is able only to look back for few steps.

At first sight, RNN can be considered identical with some other neural networks. However, in comparison to feedforward networks, they have a feedback loop, which is connected to the decision taken in the past, considering the elements' output as an own input. Considered as networks with memory, the purpose of it is to use the information the sequence itself, in order to perform some tasks. Furthermore, the information from the sequence is kept in the RNN and it spans some steps ahead, in order to influence each new example. With some other words, because of the correlations that are found, it is considered that RNN have "long term dependencies", reason why they have also the capability to share, over time, certain weights.

The structure of a basic regular RNN is shown in Figure 1.7 [19], having, in fact, three layers: input layer, recurrent hidden layer and output layer. Inputs layer has more input units, but they are fed one at a time into the RNN. Input units and hidden units from the hidden layer that are connected, this connection being defined with a weighted matrix. Moreover, hidden units are linked between them, in a recurrent way. It also, has a delay line, whose purpose is to hold the activations until they can be processed at the next step. It is important to mention that time is discrete in this case and activations are updated at each step. The big advantage of such a network is represented by the fact that if an input comes later, it can be still used by delaying the output with a certain number of frames, such that the information can be included.

In case of such a structure, more linear hidden layers act as a single layer. However, in the case of non-linear layers, it is more complicated, because they are more powerful, reason why RNN uses relationships for learning input-target. Some of the most famous activation functions have been presented before, for ANN and they are used also for RNN. The selection of such a function depends on the problem and nature of the data [20], the main goal being to get a higher performance, so a lower loss of the information.

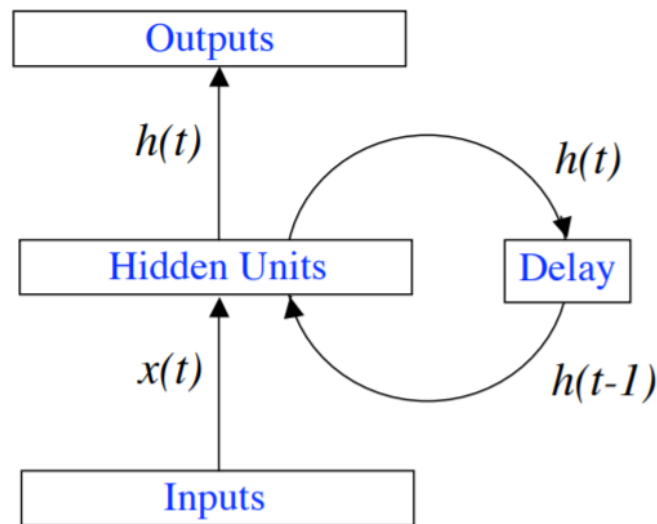


Figure 1.7. Fully recurrent neural network. Source [19]

RNNs are, in fact, based on an extension of the backpropagation architecture, that is performed over time. It is expressed as a series of computations, where the next step is linked with the one before. Starting from the idea that all neural network are actually nested composite function, when adding a time variable, the series of functions are extended, especially when computing derivatives using the chain rule. This is the case of the RNN architecture.

However, RNNs are old architectures of ANNs, reason why, they had to face some issues such as vanishing gradient problem, which influenced the performance of the network. In this case, RNNs try to establish connections between a final output and past inputs, but it is difficult for the network to know exactly the importance of each remote step. Actually, this issue cause problems that determined the network to ignore the long term dependencies and make it complicated for the network to learn the correlations that are distant. The main reasons that led to the appearance of this issues were represented by the standard non-linear functions, whose gradient is almost zero everywhere or by the magnitude of the gradient, which is multiplied many times, such that when the eigenvalues of the recurrent matrix are smaller than one, the gradient would converge rapidly to zero.

Researchers seek to find different solutions for this challenge, so RNN started to have some different architectures [21], with specific features and applications. Independently RNN (IndRNN) is one of the examples of RNN's architecture which solve the issue mentioned above, by creating a structure where each neuron from one layer gets the past state as a representation of the context information. Thus, they are independent one of the other and the exploding problem does not appear anymore. For this type of network, the gradient of backpropagation can be normalized and the memory can be kept as well. This architecture can be used to build very deep networks, having also the ability to learn long dependencies, as the typical RNN.

Another example of the RNN architectures is represented by recursive neural networks, which were created such that the same set of weights can be applied over a structure, by passing it in a transversal order. They are usually trained for automatic differentiation, being usually applied to NLP. On the other hand, another example is Hopfield, whose connections are symmetric. However, this one requires stationary inputs, so it is not a typical RNN, which can process a pattern of sequences.

Furthermore, LSTM are a special type of RNN, being able to learn long dependencies and solve issues such as vanishing or exploding gradient. They have been introduced in 1997 [22] and they became popular not only because of solving the problem mentioned before, but also because of large number of application that they had. They were designed to avoid long dependencies, because they remember the information by default, so there is not need for them to learn it.

All in all, RNNs are a class of artificial neural networks, which have one or more feedback loops. They can be considered cycles over time, whose efficient training represented a major issue, because of the initial values of the weights. Usually, the first values of the weights are small, between 0.001 and 0.01, depending on the task that has to be done and, also, on the properties of the input data. Moreover, researchers use to assign, to the biases the value zero, which might be different for the output bias, whose values can be different, but still very small. Also, the optimization of the network was more difficult, because of the relationships between the parameters of the network. Usually, algorithms that use this network require a large number of iterations, even if the initial goal was to reduce their complexity. There are some approaches that are usually used in order to train RNNs [20], such as multi- grid random search or time-weighted pseudo-newton optimization.

Depending on the type of the recurrent neural network, there are different advantages and disadvantages of using them. However, most of them require complex computations. While typical RNN have to face the vanishing issue, more advanced RNN have other constraints such as knowing the start and the end of the sequence for bidirectional RNN. The main step ahead was anyway represented by their feedback connection and ability to store over time. Also, they were highly appreciated for their capacity to learn and to handle sequential information, ignoring the length of the sequence. On the other hand, it is hard to start the training, because of the initial values of the weights,

whose importance is very high. They are dependent on the performed task and also on the properties of the dataset, so there is no pattern that can be created for them.

All in a nutshell, RNNs are a model of ANNs, that are still used, because of some of their features. The fundamentals of this architecture led to the development of new ones, which are more adaptable, but, however, RNNs still represent an important model of neural networks, for their ability to work over time.

1.6 Overview of lip reading systems

Lip reading is a method through which a text is decoded by using the movements of the mouth. Usually, there are some challenges that a developer had to overcome such as [23] pose, motion or photo resolution. Even if this approach firstly appeared as a complement of the audio based speech recognition, afterwards it has been suggested as an individual system, that can be used whenever audio signal is not available at all. It is important to add that this technology includes not only machine vision, but also, language perception.

Researchers have been observing that this is a very different area of study than ASR, because training results for the same dataset are different when the sound is on or off, including, also, a study where speakers whispered, observing as well a degradation. This task is being considered even harder than the one mentioned before, because of the lack of context, such that the system does not know where and how to place the information. However, the main applications of such a system were represented by AVSR, VSR, human expression recognition. In all cases, the main procedure included three important steps [24]: face detection, localization of ROI and extraction of the visual features. After features are extracted, they are analyzed in order to observe certain patterns.

Furthermore, the development of this area led to a new desire, to get an automatic lip reading system, which can extract both spatial and temporal information, in order to use it later. Usually these types of system are able to perform word classification, but there have been few attempts to create an end-to-end trainable sentence system. From this point of view, the analysis can be problematic, because of two factors: firstly, to include the importance of teeth or tongue in the system, secondly to differentiate some visemes, depending on the language, even if when they are pronounced, they sound almost the same.

When this concept appeared, around 2002 [25], deep learning was not used in order to build this type of systems, reason why they were not so efficient, because of the increased time that had to be allocated for preprocessing. Most of the systems used hand-engineering features, being usually modelled using HMM. In order to extract features from a video, researchers proposed initially spatiotemporal descriptors such as optical-flow or detection of the movement, but they concluded that this not an efficient way to create a lip reading system. However, until this attempt, the first visual-only system was build in 1997 [26] in a dataset which was limited. It was based on a model that included a study of visemes and the way they are conducted, depending on their size. Later on, based on the same model the first AVSR system have appeared, improving in this way the performances in noisy environments.

Recently, the inclusion of deep learning in such a system led to a lot of different attempts, usually performing a word level recognition or a phoneme classification. Usually, when the trial is moved to the sentence-level, their spatio-temporal properties have a lot of constraints, such that the performances are poorer. Moreover, the systems were not build to have the ability to perform if the sequence length is variable, such that they are even more sensitive. Multimodal learning means

fussing and processing information that comes from different sources, being able to find correlations between the available modalities. Modelling them afterwards represented an important task, especially because this was one of the most used approaches when trying to get a visual only lip reading system.

Starting from the idea that in a traditional lip reading system, firstly the information is extracted and then it is classified, a lot of systems were designed. While the first step used different algorithms that were tried and optimized, the second one usually included dynamic classifiers such as HMM or LSTM. Some other researchers, used CNN in order to predict phonemes or visemes, in total opposition to the approaches who tried to recognise directly words, or even sentences.

Even if these attempts, which used deep learning constituted a real success, working on an end-to-end approach has a limited number of trials. One approach was done by M. Wand [26], who created a system made up of a feedforward layer and two LSTMs. It was able to learn from raw ROIs, but the results were not so convincing, by failing to return the ones promised in the paper. Based on the same model, the authors from [27] created an end-to-end system that has the ability to do what Wand proposed in his paper, using two streams: one which encodes static information and one which encodes local temporal dynamics.

More advanced systems created architectures that have the ability to learn end-to-end from the sentence level point of view. It took into account previous drawbacks, such that it tried also to make it adaptable, if the sequence has a variable length. LipNet is the system developed in [28] and the authors described it as the first successful attempt which can be trained to the sentence level. It has a more complicated architecture, but it seems that it is more efficient, taking into account the results presented in the paper.

However, from a different point of view, prior work that has been done on this subject, can be split into two big categories. Firstly, some systems use CTC, so the model usually predicts a correlation between frames and labels and after that, they look at the best approach of the alignment in order to predict the output sequence. On the other hand, some systems use models that read all the input information before starting any task. Usually in this case a lot of tricks are taken into account, such as scheduled sampling, in order to increase the performance of the system.

Beside all these attempts, any lip reading system follows the same important steps. Firstly, it has some pre-processing stages, which usually conclude to the mouth extraction from the frames, which would be later used in order to get an output value. After that, depending on the approach and architecture of the system, the changes that appear, when analyzing the mouth, led to certain conclusions. In this way the system is trained to recognize different words or sentences, depending on the purpose of it. The training is done usually by making correlations between the frames and the alignments that were created from the video. From this perspective, there are different approaches, including alignments that mention only the start time of each word or the start and end time for each sequence. After that, it can be tested, such that the user is able to observe the efficiency of it. Usually the data used for such a system is not the one used for testing, because, otherwise, the system might not recognize how the speaker is moving the lips, but it might recognize it as a whole and return what was used for training directly.

Moreover, it is important to add that any system like this is highly dependent on the language for which it is used. Depending on it, different phonemes might be included, such as “a” for Romanian language and differencers or group of letters might be pronounced differently. From this perspective, it has been observed that vowels are said with in a more clear way, but the way the mouth moves when almost the same. This theory was anyway developed firstly by Alexander Graham Bell who

concluded at that time that multiple phonemes are identical when they are analyzed from a visual point of view. This is the reason why some systems made a phonem-to-visem analysis, clustering them into different categories as follows:

- Lip rounding vowels
- Alveolar semi-vowels
- Alveolar-fricatives
- Palato-alveolar (D)
- Bilabial (E)
- Dental (F)
- Labio-dental (G)
- Velar (H)

Depending on the nationality of the speakers and the chosen languages used for training, there are some confusions that are usually done. For example, in the case of British people, most of them appear between “aa” and “ay”, because probably they are articulated identical as an open unrounded vowel. Also, for the same language, “b”, “p” and “m” and usually confused with “d”, “t” or “n”, because from the front they look the same, even if vocal fold vibration is different.

From a word perspective, it is more difficult to have confusions, even if the pronunciation of certain words might look the same. However, in this case the movements of the tongue should be also taken into account, because they are part of the pronunciation. Usually, in this case, there are two types of words: first which require great articulatory movements, such as “please” for English, and second which require little lip changes, making for the lip reading systems more difficult to identify the word.

All in all, lip reading systems can be considered a mixture between computer vision and speech recognition. Because of their potential, researchers, tried different approaches, in order to get more and more performant systems. The evolution of this field is obvious, starting from a machine that needed a lot of hand- engineering almost 20 years ago, nowadays systems that can be trained at sentence-level become more and more accessible. Even if until now, they have been tried only for English language, some researchers started to make attempts for some other ones. Urdu language has been part of this analysis, but the results were not so encouraging, because of the way the different phonemes are pronounced. Also, this thesis analyses the possibility to implement such a system for Romanian language, taking into account specific constraints such as the letter that exist only for our language.

1.7 Description of mouth detection algorithms

Lip reading systems are trained using a region of interest which is represented by the mouth. Actually they analyze the movements of it, reason why its extraction represents a very important step in the building of such a machine. In order to localize the mouth, usually all algorithms detect firstly the face.

Lip segmentation is still an important topic for researchers, due to the large number of applications that imply this step as well, such as audio recognition, speech recognition or gesture recognition. Different approaches have been tried over time, in order to improve the performance of such an algorithm, but also to decrease the processing time. Usually, any system that does lip recognition can be placed in one of the following three categories [29]: first one uses directly the

information from the image, second one builds models that are used afterwards and the third category is a hybrid between the two ones mentioned before.

The first category uses directly the information from the pixels, reason why it is less expensive from the computational point of view. On the other hand, it is more sensitive to any change, such as variation of light. So, there are several algorithms that try to detect the lips directly from the image, using the difference of color between the skin and the lips. However, in some cases lack of contrast or change of the illumination affect these systems are the results are not that good. In order to optimize them, some researchers suggested color transformations in order to increase the contrast or color clustering.

Another technique proposed for lip segmentation was based on PCA. The flow of such an algorithm is simple and consists of two important steps: firstly, the lips are labelled manually and then the PCA is applied in order to extract the contour of the different shapes that the mouth can take. This second step is called eigencontour.

Technique based on the model building are considered quite robust and expensive from the computational point of view. In this case, the approaches were more efficient, having even the ability to track the lips on real time, using usually different filters such as Kalman filter. These algorithms are also able to work in both situations: frontal and profile. Two of the most used models are ASM (Active Shape Model) and AAM (Active Appearance Model) which have the capacity to learn which the shape of the lips and how they look like. After that a PCA is applied for reducing the dimensions and, because of the existence of cost function, all models are used iteratively in order to test different images.

The hybrid technique is, in fact, a mixture between the two methods described above: image based and model based. While the first ones are cheap from a computational point of view, they are too sensitive to illumination, reason why their performance can be easily affected. On the other hand, model based systems are expensive and have a more complex architecture. Thus, almost all hybrid systems that have been developed use the color technique for a quick estimation and after that introduce the model based one in order to get the contour of the lips.

All these techniques are used to detect the lips, but before getting there, any system must do some steps before. More exactly, in order to find the mouth, any machine must be able to do also face recognition. There are some algorithms that do not use at all deep learning, referred by the literature as “shallow”. They use usually hand-crafting image descriptors, which are finally aggregated in an overall face descriptor. However, their efficiency is not so high and the training time is a lot bigger than the one required by the algorithms which use deep learning, reason why they do not represent a point of interest in building a lip reading machine. On the other hand, a lot of face recognition systems are based on deep learning architectures, which usually include a CNN feature extractor and a function that has the ability to learn.

After face is detected, the next step is to localize the mouth, depending on the used architecture. There are some algorithms that detect directly the mouth using the contrast, but there is also some architecture that find the lips, depending on the position of the eyes or nose. Anyway, in all cases, the system must take into consideration some other aspects. For example, some of the people might have facial hair and in this case the image model might be affected. Also, if people smile, teeth are, also, visible and this might create some confusion when making the lip segmentation. Furthermore, in some situations, the color of the lips can be different such that the color model can face again some issues.

Lip segmentation is done from both videos or images. For images, the technique is more simple, because the mouth is detected with no additional step. However, when this is done in videos, usually there are two approaches. First one is represented by a real time tracking, that is not so useful for lip reading systems, because the movements are not so easily analyzed and the machine cannot learn all of them. The second option is represented by a more complex but accurate option, which split the video in frames and after that extract the mouth. In this case, lip reading systems can clearly see any change of the lips, so it is more probably that such an approach would be usually chosen.

All in all, lip segmentation represents a very interesting topic, reason why researchers keep working in order to optimize the existing techniques. This step that is made up for the pre-processing part is very important for any lip reading system, because without accurate data, the training cannot be done properly. Their movement is usually analyzed, so an accurate extraction is vital, in order to get a high performance.

Chapter 2. Proposed method

The proposed method has been built based on the paper [28], such that it can be used for the sentence level recognition. It has been adapted for Romanian language, considering all the possible constraints that it has, reason why there are some changes that have been done. The code was developed in Python, starting from the one that can be accessed at [30] and it uses an open source library that is able to make complex computations.

LipNet is an end-to-end model, which makes sentence-level predictions, being able, after training, to return, almost on real time, sentences that a speaker says, if the words were recognized before. The system was developed after analyzing a lot of approaches that developers had before and the reasons why they did not obtain great performances. Moreover, past machines were developed only for phoneme or word identification, but this one is able to make sentence predictions, thus, it is both, more advanced and more useful in the real world. Firstly, the system was tested on English, using a big dataset that was after that adjusted, having the same dimension as the one that was created for Romanian language, to see exactly the differences. This comparison is useful because it shows the dependencies between the size of the dataset used for training and the performances that can be obtained.

The flow of the whole process can be observed in Figure 2.1 and it follows the pattern that almost all lip-reading systems have. It has six important stages, that include everything from video acquisition to the feedback that would send. First step that is presented in the flowchart from below is called “Video acquisition” and it represents, in fact, by the process of creating the dataset that will be used for training. In this case, it is made up by short videos, that include sentences that have from four to six words. Same sequences are said by all speakers, such that the training is done having a big diversity of people. The videos include both genders, such that the system can get results when being tested by males or females. The quality of the videos is, also, important, reason why all videos that have been used are clear and include people that are filmed from the shoulders to above.

The second stage of the flowchart is called “Image processing”, which is represented, in fact, by all the changes that the video suffers to transform it into frames, that can be used later to detect the mouth. All videos are split into 75 frames that will be processed to extract the lips used for the training part. The next two stages are done together with this one, such that the pre- processing part is made up by three steps that will finally return the mouth. The system used an algorithm that returns directly the mouth, to use it for training. Even if it might be a little bit more consuming, the efficiency of it worth it. The automated lip-reading system is the fifth stage and it is represented by a set of algorithms that have different training protocols that can be used for training. The purpose of them is to analyze deeper the mouth movements, such that the results can be more accurate. The last part of such a machine is represented by the feedback, which in this case returns different information that will be described later. They offer an overview about the performance of the system and let the user know what could be improved.

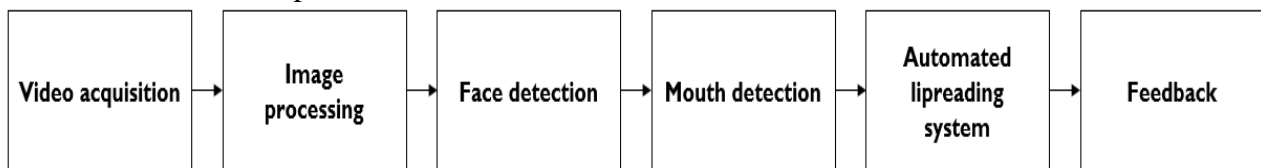


Figure 2.1. Flow chart of a lip-reading system.

This flowchart includes all steps that are done from getting the raw data to obtaining a valuable result. While LipNet used a data base that can be found online, for Romanian language the video acquisition was done by creating a new dataset. The next three stages are done at the same time, with the help of a detector. The actual system is pretty flexible and allowed me to make trainings in different situations, returning as a feedback the same metrics.

2.1 Necessary tools

The chosen programming language was Python, which is an object- oriented language exactly like Java or C#. It has a dynamic semantics and it is built using data structures. Because of the supported packages, it encourages code reuse and modularity. Since the release that happened around 1990 [31], more versions were launched and usually they are not compatible between them. LipNet was developed using Python 2.7 and it includes different open source libraries. The adapted version, that is used for Romanian language uses the same version.

TensorFlow is an open source library that aims to solve numerical computations using data graphs. Its aim was from the very beginning to be used for deep learning and machine learning research, but nowadays it is applied in many other domains. It is a cross platform that can run on CPU and GPU, such that LipNet was trained on both, depending on the availability and the computational power that was needed.

One of the reasons why this framework was chosen upon the others is represented by the documentation, which is well done, offering a good ability to modify anything. Also, from the debugging point of view, TensorFlow is easier to be managed, because it has a special tool which let the developer evaluate any expression. Also, it is a great tool when the visualization part is taken into consideration, because it comes with TensorBoard, which can plot almost everything, offering the possibility to make comparisons in an easy, efficient way. Even more, TensorFlow looks more like a library, than an actual framework, because it offers a lot of features, such that the developer has the freedom to do whatever he wants.

Keras is a high-level API that aims to work with neural networks. It is written in Python and it is able to work on top of TensorFlow. It can run on both, CPU or GPU, and it has a user friendly interface, by reducing the cognitive load. The API is very clean and simple to learn and it was recently called “Lingua Franca”, because it has the simplified frontend of TensorFlow. This correlation lead to a nature choice of Keras as a complement to TensorFlow. One of the main qualities is represented by the modularity. More exactly, it understands a model as an independent graph, that has as few restrictions as possible. In comparison to Caffe, it is more documented and the active community is bigger, such that any issue can be solved easier.

It uses PIP, which is a package manager in order to manage or install any software package that is written in Python. It was a replacement of easy_install, that was preferred due to the operation system which has been used for training and testing LipNet. Besides this, there are some other libraries that have been installed in order to run the algorithms. More exactly, H5PY has been installed in order to have an interface of HDF5 binary data format. It was necessary because all models that the system saved after training were stored using this format. This library let the system store a big amount of data and help it manipulate data from NumPy. Matplotlib was another installed library, useful for publishing figures. However, it was not used, because of TensorBoard, which offered more accurate and clear graphs. A basic installed package was NumPy that can work with large, multi-

dimensional arrays and matrices, such as images. SciPy is another used library, for image and video processing. For the same reason was used Pillow or PIL and SK-Video, that are more flexible and can manage different types and formats of images and videos. In order to detect the facial landmarks (mouth), the algorithm needed DLIB library. Last, but not least, NLTK was necessary for working with human language, such as words and sentences.

LipNet is a set of algorithms written in Python, that uses both Keras and TensorFlow, in order to get accurate data, after complex computations. The system runs on Linux, a software operation that is more adaptable for deep learning applications. For training, both CPU and GPU, were used, depending on the operation that has been done. Figure 2.2 highlights the approach that the system had, including Python, Keras and TensorFlow. In fact, it is an overview of the inside project, then includes the way the information propagated thorough the system.

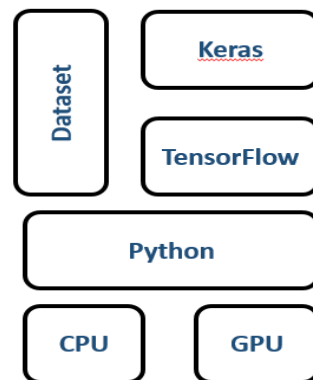


Figure 2.2. Outside approach of LipNet system

Before getting the final frames that were used by the system, there are some other tools that have been used, to obtain the desired data. More exactly, there are some actions that have been taken upon the videos, such that they could be processed by LipNet. Firstly, to get videos with 25fps, there was a need to edit videos, because the used camera had a default fps of 50. Consequently, FFMPEG was used to do this change, because of the high flexibility that the framework has.

The database that is used for Romanian language was created in the laboratory, reason why all the necessary alignments that have been done, mentioned the start and end time of each word, including at the beginning and at the end few moments of silence. These alignments were done manually, using VideoPad and Audacity, which were helpful in order to see both the audio signal and exact moments of time, including seconds and milliseconds.

2.2 Architecture of the network

LipNet was built on an architecture that took into consideration the specific requirements for a sentence-level prediction, but, also, the reasons why previous systems did not get high performances. It is a neural network architecture that is able to map sequences with variable length from a video to a text. It is trained end-to-end and it has two important blocks, that have also some other sub-blocks, that will be described below.

The architecture of LipNet can be observed in Figure 2.3 and it contains two important parts: lip segmentation and lip-reading processing. The first part is made up by the necessary divisions needed to process the videos and the second part is made up by all the required blocks that would be used for processing the obtained frames. Videos are processed using a Dlib face detector and an iBug

face shape predictor. The second big block contains three STCNNs and Spatial Pooling, followed by two Bi-GRUs and a CTC loss. Regularization, as a process that is needed to get a higher performance, is also included in the architecture. In addition, there are some additional functions and transformations that are applied to the raw data, to get a more accurate feedback. The input of such an architecture is represented by N frames, that are taken at different moments in time. Frames have the same dimension and they include only the lips of the speaker. All the blocks from the image and all the functions are applied to them.

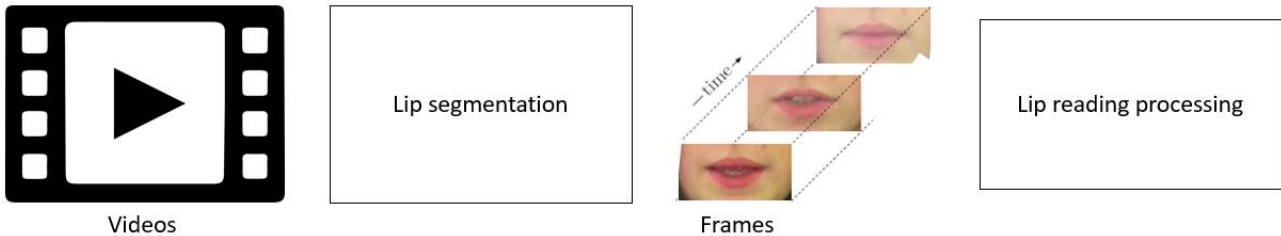


Figure 2.3. High level architecture of the system, including the raw data and the 2 main important blocks of LipNet

The first block that I called “Lip segmentation” can be observed in detail in Figure 2.4 and it has two important components as mentioned before: DLib face detector and iBug shape predictor. DLib face detector is a software that can recognize the face. Based on an open source library, it was developed to work on Python, R or Matlab. Dlib-ml is a platform that was initially written in C++ and it was intended to be used for both research and real world, in opposition to other libraries such as Torch or Shogun which focus only on the research area. The face detector is built using this library, based on a classical approach that uses the feature of Histogram of Oriented Gradients (HOG). It is combined with a classifier which is linear, a multi-scale image representation and a sliding window. The chosen option for the image representation is the pyramid one, because of the repetition of smoothing and subsampling. The sliding window is the one that will fit the face inside of it. However, this algorithm is used for the detection of different objects, reason why, in few cases, it confused the face with some other objects from the background and it extracted something else. This lead to a clear decision that all subjects would be filmed using a white background. The second component of the block is iBug face shape predictor. It uses 68 landmarks which are in fact points having the coordinates (x,y) and map the facial structure on the face as per image from Annex 1.



Figure 2.4. Architecture of the lip segmentation block, including both components DLib face detector and iBug face shape predictor

Kalman filter is the last step applied on the videos, before getting the final output that would be used by the rest of the architecture. It is an algorithm based on more measurements of the statistical noise or any other inaccuracy. Using them, it estimates unknown variables which tend to be more accurate than the one that get their value based on a single measurement. In this case, it is useful to

make correlations between the frames and the alignments, such that they can be used for further training.

After this block processed the videos, it returns, as outputs, frames that include only the mouth of the speaker. In Figure 2.5, can be observed the flow of this process, where firstly the face from the video is recognized and the it is cropped. While the last version would be used for the training, the other ones are not even stored. They are considered intermediary steps that can help the process of lip reading, but they are not that important to let the user observe them more accurately. This happened for any video that has been used by LipNet, without considering for what language it would be used. In the same way, videos that are used for validation would be pre-processed. The only videos that do not follow this flow are the ones used for testing.

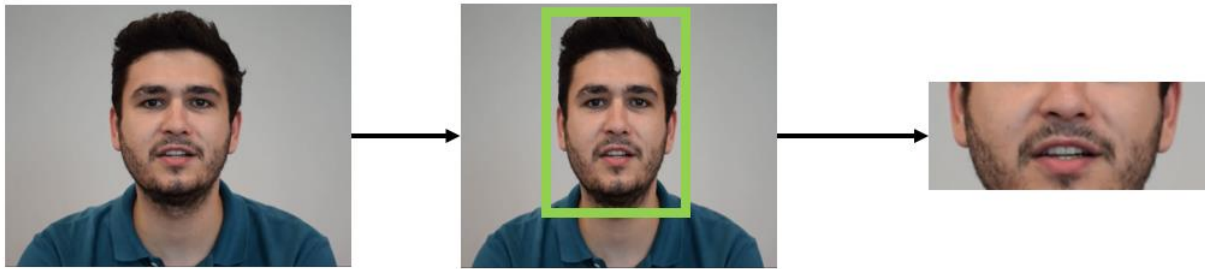


Figure 2.5. Flow of the Lip segmentation part

The second important block of the LipNet’s architecture is called “lip reading processing” and is made up by different neural networks, whose role is to get a higher performance of the system. Figure 2.6 highlights the important components and the number of each of them. The input of this block is represented by the frames which resulted after processing the videos using the previous block. A more detailed and personalized architecture, that includes all nodes can be found in Annex 2.

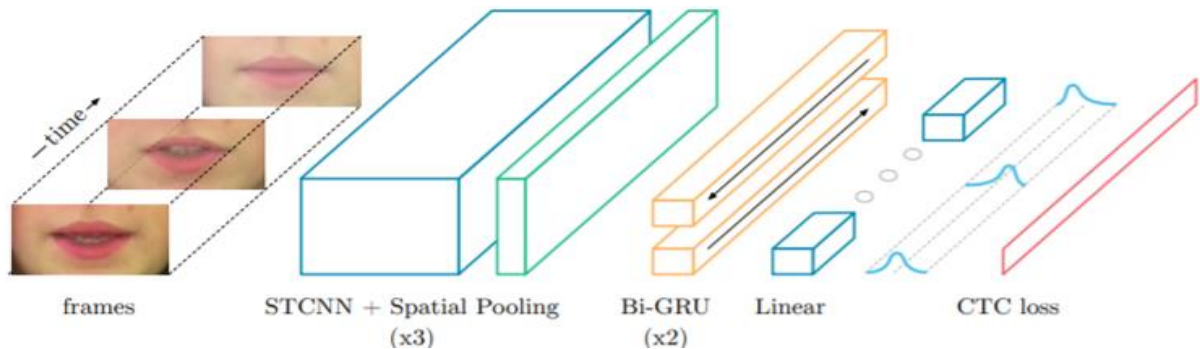


Figure 2.6. Architecture of the lip reading processing block. The inputs of the block are images that are processed using neural networks. Source [29]

The block has three STCNNs that are applied together with spatial pooling. Spatial-temporal CNNs represented a big advance in computer vision, because of their capability to process data over both space and time. So, the role of these three networks is to convolve frames over the time and spatial dimensions. Each STCNN is followed by a spatial pooling, whose role is to group spatially adjacent pixels, to improve the robustness of any deformation that the object might suffer. The chosen function of the spatial pooling is the max one, such that the chosen pixels will have the maximum values. The size, stride and pad of both the STCNN and the pool for all three layers depends on the size of the input volume, which is modified depending on the operations applied to the frames. After that, in the architecture there are two Bi-GRU, whose purposes are to keep information from a long time ago. GRUs are special RNN, which add cells and gates to control the informational flow. It is

like a LSTM, but in this case the information can be propagated over more steps of time. The chosen GRUs are bidirectional because one RNNs map the information on both directions from A to B and B to A, ensuring that any element from B depends on all elements from, at each moment of time. The input of Bi-GRU is, in fact, the output of the STCNNs. Inside this part of the architecture, a linear layer and a softmax function are applied to a feed- forward network, at each time- step, whose role is to obtain real values from arbitrary real values. In fact, output of the GRUs is processed by the linear layer and softmax function and the result is then transferred to CTC for training.

CTC is used in computer vision to get score functions for trainings that use RNNs. Usually, the input is represented by observations and the output is represented by labels. In the case of LipNet, the aim of this neural network is to eliminate the need to have alignments of the input for the target output for the data used for training. Starting from a model whose output is a sequence of discrete distributions over the vocabulary, CTC computes [29] the probability of a sequence by marginalizing over all sequences that are defined as equivalent to this sequence.

CTC trains models that use RNN, to estimate the probabilities at each time step. CTC uses two functions that work together to compute different matrices: CTC loss and CTC blank. It uses the blank spaces to delimitate the words, blank labels serving in fact as transitional state between two classes. The CTC loss function let for training deep neural networks, being able to perform end-to-end iterations.

All presented layers use ReLu activation function, which uses, the positive part of the arguments. It is a classical approach that is usually taken by deep learning applications, and it is considered the simplest choice that is non-linear. One big advantage is represented by the impossibility to get a squeezing effect. Also, it is useful for training big datasets, reason why it is highly recommended for a system like LipNet, whose purpose is to decrypt movements of the lips.

2.3 Initial parameters

The performance of the system is highly dependent on the parameters that it has. Their initialization depends on the type of the network and, also, on the type of the method that it used. The architecture described before has different initial parameters, that are changed and updated, depending on the specific of the network.

Firstly, all frames that are used for training are, obtain after processing the videos. They are, in fact, RGB images, which are normalized per channel and, after that, standard deviations are applied. The mean is considered the value that is expected for the channel to have and it refers in fact to a probability distribution or a random variable from the central tendency. The standard deviation quantifies the amount of dispersion of a certain set of values. The mean of each channel and its standard deviations have the values that can be found in Table 2.1.

Channel	Mean(μ)	Standard deviation(σ)
R	$\mu_R = 0.7136$	$\sigma_R = 0.1138$
G	$\mu_G = 0.4906$	$\sigma_G = 0.1078$
B	$\mu_B = 0.3283$	$\sigma_B = 0.0917$

Table 2.1. Values of mean and standard deviation per channel

Moreover, neural networks that used in the architecture of the system have different parameters, that depends on both, input volume and applied operations. First part of the lip reading processing block has three layers with STCNNs+ Pooling, whose defined parameters are size, stride

and pad. Size of the networks has three dimensions that depends on the number of channel, which are three in all cases, and the dimensions of the kernel. The stride is responsible for controlling the way that filter convolves around the input volume. It can be done by shifting the filter on all three dimensions, such that it also has three values. Padding is the parameter used to determine the number of pixels that would be added on each side of the input. The spatial pooling uses a max function, whose dimensions were chosen such that the dimension of the final frame would be reduced. It has also three parameters that have different values that would be seen below. Bi-GRUs are RNNs with a fix size of 256 neurons, because it replicates two LSTMs with 128 neurons each. Linear transformation support 28 characters: 27 letters or digits and a blank space.

All the parameters mentioned above are used for initialization of the system. They are called hyper-parameters and their value is influenced by the input volume. The initial values that the parameters from the lip-reading processing block take can be observed in Table 2.2 [29]. The first column of the table mentions the layer, the second one the values of the parameters, the third one the dimension of the input and the forth one the order of the dimensions. It is important to add what is the meaning of each symbol used for the last part of the table. More exactly, T is used to define the time, C represents the number of channels, F denotes the feature dimensions, H and W are the height and width of the image and V describe the number of words used in the vocabulary. V includes also the CTC blank symbol.

Layer	Size/Stride/Pad	Input size	Dimension order
STCNN	3×5×5 / 1,2,2 / 1,2,2	75×3×50×100	T×C×H×W
Pool	1×2×2 / 1,2,2	75×32×25×50	T×C×H×W
STCNN	3×5×5 / 1,2,2 / 1,2,2	75×32×12×25	T×C×H×W
Pool	1×2×2 / 1,2,2	75×64×12×25	T×C×H×W
STCNN	1×2×2/ 1,2,2/ 1,1,1	75×64×6×12	T×C×H×W
Pool	1×2×2/ 1,2,2	75×96×6×12	T×C×H×W
Bi-GRU	256	75×(96×3×6)	T×(C×H×W)
Bi-GRU	256	75×512	T×F
Linear	32 = (31+blank)	75×512	T×F
Softmax		75×32	T×V

Table 2.2. Hyper- parameters used for the initialization of the system

As it was mentioned before, all parameters are influenced by the input size. Each video that is processed returns 75 frames, reason T has always this value. However, all other dimensions of the input will change, depending on the applied network. The value that vocabulary can take depends on the used alphabet. In the case of English, there are 27 letters, plus the blank one would be 28. For Romanian, the situation is different, because we have some additional letters such as “ă”, “â”, “î”, “ș”, “ț”, reason why the number of characters is 32, including the blank.

2.4 Training parameters

The training process plays an important role when talking about LipNet’s result. Beside the initial parameters, there are some other factors that must be taken into consideration, to get higher performances. The output that the system would get when tested, depends totally on what it learned during the training.

Training is done after the pre-processing part takes place, such that it works with frames instead of videos. It uses an iterative method, such that images are shown to the network one at a time and the weights associated are adjusted every time. There are multiple iterations of the dataset, such that it can learn it better. Because of the existing of CTC, there is a connectionist learning process, that analyzes also the links between the units.

Network can process data one at a time, by using firstly the initial parameters and then the new ones, including the weights and functions from hidden layers that were presented before. After that, it compares the results to the ones that are desired to get at the output and propagates the errors back through the system, to adjust weights. During the training process same dataset is processed a certain number of times, to get a better feedback, and the connection weights are continuously updated. The partial results and the final ones are stored in CVS files, that can be accessed, also, before training ended, to make an idea if there are any changes or not. Also, during the training process there are some HDF5 files are stored, that represent the models that could be use later for testing. Usually the system stores more than a model and the user can choose any of them for testing.

LipNet is an adaptable system that offers more training protocols, such that the approach that is taken over lip reading is diverse and shows different capabilities, depending on the considered situation. Same dataset is used, such that the results can be compared, but the way that each protocol trains is different. The first proposed method is called “Overlapped speakers” and it has the same approach that the author had in [27]. More exactly, a certain number of random sentences of each speaker are used for validation, while the rest of data is gathered together for training. The second proposed method is called “Unseen speakers” and it was never mentioned before in the literature. It takes an equal number of males and females from the dataset and use them for validation. All the other data is used for training. The third option is called “Random split” and it is the simplest method for training. Considered an unmaintained possibility, it just gathers all the data together and from there a random speaker is chosen for validation.

Also, LipNet offers that can use curriculum learning. It implies algorithms that can learn gradually, starting with small parts that will increase the difficulty level. In this case, at the beginning, the system is quite restrictive, but the resources become bigger as the system learns. This option is used for both overlapped and unseen speakers.

The training process is a complex one and it is done using the frames. There are some parameters that can be edited and might influence the stored model and, consequently, the results that can be obtain when testing it. Firstly, there can be chosen the images that will be used for both, training and validation. Also, it offers the chance to modify the dimensions of the images, such that it can be adapted on any data base or any type of image processing. Furthermore, it has a start and a stop epoch, so there will be a defined number of iterations that will tell the user how many times did the system passed through the database in order to learn the movements of the lips. In addition to these parameters, there can be modified the number of characters from a string or word and the batch size. It influences how many images can be processed at the same time, in one iteration, having three options:

- **Batch mode:** batch size is equal to the total dataset
- **Mini-Batch mode:** batch size is greater than one, but less than the total size of the dataset
- **Stochastic mode:** batch size has value one

Therefore, batch size tells how many samples go through the network, requiring a different amount of memory or training time. However, it does not influence so much the performance of the system, but the accuracy of the gradient.

As mentioned before, the training process gives a model that stores the decoded data, depending on the performances of the systems at that moment. It can be saved after different number of epochs, chosen by the user, and then and it can be used for testing. Usually, the last stored model is chosen for further utilization, but in some cases, an older one is chased.

2.5 Constrains imposed by Romanian language

Each language has its specifications and different aspects that are unique. Some phonemes are said moving the lips in the same way, reason why it is more difficult to identify exactly the sound. In the same situation are some groups of letters, reason why their interpretation might fail sometimes. Also, for each language, the alphabet is pronounced in another way, reason why there cannot exist a pattern of the movements of the lips, depending on letter. However, each language has its visemes, which highly change how mouth's movements are interpreted. In almost all cases, vowels are pronounced with the same movement of the lips, so they are identified usually in groups of phonemes. Also, for languages like Chinese or Arabic, the interpretation includes some characters that cannot be defined as letters.

For Romanian, there are some extra letters that influence the maximum number of characters. More exactly, there are the following phonemes that are also included in the alphabet: "ă", "â", "î", "ș", "ț", reason why the total number of characters is 31. In addition to this, there are some groups of letters that are made up of two or three letters but said from a single movement of the lips. They are "ce", "ci", "che", "chi", "ghe", "ghi", "ge", "gi". Their presence in a word or in a sentence influence the way it is decoded by the system. Also, as in any other language, the vowels are considered in fact visemes, whose pronunciation is almost the same. The only difference is how the tongue is moved.

On the other hand, Romanian has a lot of homophones, so whenever the system has to learn a word that include them, it will have difficulties. For example, the pronoun "ea", which means "she" is pronounced the same with "ia", which means "take". The same situation is for "ele" and "iele", two words with a totally different sense, but with the same mouth movement. Even more, there are some words that can be written both together and separately, and still have a meaning in Romanian. In this case, the system can be confused when to learn them as a single word and when to learn them as two different words, such that, in many cases might return a value instead of two. For example, "numai", which means "only" and "nu mai", which means "not anymore" can be easily mixed. In any of these cases, the meaning of the whole sentence would change and the interpretation would be different.

LipNet is a complex system that can be adapted for any language. However, it has two constraints when trying to use it. More exactly, it has a dictionary that must include all sentences that would be used for training and it has a document that must have all the phonemes from the language. For Romanian, it included five sentences, with four or five words each. On the other hand, the list of phonemes is updated with the letters from Romanian, adding afterwards the group of letters mentioned before.

2.6 Sentence level prediction

Any lip reading system can be trained in two ways, either world level, or sentence level. Depending on the chosen method, there are some constraints or performances that can be obtained. However, in order to predict whole sentences, it is recommended to train the system in the same way. Otherwise, when trying to recognize words in a sentence, the machine might recognize the whole sentence as a word and confuse silence moments with some phonemes that require a movement of the lips which looks almost the same. On the other hand, if the purpose of system is one that needs only words, it is more comfortable and efficient to train it on word-level.

LipNet is a system that has been trained on sentence level, such that it could be able to recognize more words from a movie, not only one. The applications that aim to include this system need a high accuracy, such that it could be helping, for example, deaf people. This is, also, an important reason why the machine must be able to recognize more than a word. In order to implement this aspect for Romanian, the system was trained using sentences, with a variable number of words. All of them have a variable number of letters, such that the system is not obliged to follow any pattern.

Chapter 3. Experimental results

First of all, LipNet is a system that was tested for English and then adapted to Romanian language, such that I could see that the system works and the constraints that it has. The system was trained on more protocols, that were described before and their result was analyzed in order observe which one has higher performances. The English results were better than the ones obtain in Romanian, but this happened from different reason, such as a size of the dataset used for training. However, the system was able to learn also Romanian words and depending on the approach that I took, it gets a certain accuracy. However, for all the attempts that have been done, the metrics that have been used were the same, such that the results can be analyzed from the same perspective. In this way, any increase or decrease have been observed easier.

3.1 Description of the datasets

To begin with, LipNet used two data bases, depending on the language that I used for training. In both cases there some technical aspects that have been kept, in order to make the pre-processing step more efficient. More exactly, both datasets use videos with a length of approximatively 3 seconds, and have females and males speakers, such that the system learns to recognize both of them. The used frame rate is 25fps and the same vides are used for all the training protocols. Each video would return so 75 frames which would be used for training or validation. All videos had alignments, that indicated the start and the stop of each word. At the beginning and at the end, it had few moments of silence, such that the system could know that a new video started.

3.1.1. GRID corpus

For English, I decided to try the same dataset used by the developer at [31], that I accessed from [32]. GRID is an audio-visual corpus, that includes recordings of 1000 sentences. They are said by 34 speakers, both males and females and use the same number of words. It was chosen because it can be trained at sentence-level and because it has a basic and clear structure. It uses a simple grammar, shown in Figure 3.1, which includes different classes of words such as nouns or verbs. It, also, includes simple phonemes, such that the system could decode also very short words.



Figure 3.1. Structure of the GRID

There are a certain number of options available for each of the categories mentioned above. Table 3.1 shows all the choices that would be used for training, yielding a total of 64000 possibilities of making sentences. The words have a variable length, so there is no constraint form this point of view. There are four commands that will be used, four colors and all the letters of the alphabet excepting “W”, which is pronounced as two double words: “Double u”. Also, the dataset has ten digits, four adverbs and four prepositions. These options are mixed between them, such that all words are pronounced for the same number of times. In total, the dataset includes 34000 sentences, out of the 64000 possible ones, having a duration of 28hours if added together.

Command	Color	Preposition	Digit	Adverb
Bin	Blue	At	Zero	Again
Lay	Green	By	One	Now
Place	Red	In	Two	Please
Set	White	With	Three	Soon
			Four	
			Five	
			Six	
			Seven	
			Eight	
			Nine	

Table 3.1. Words used for building GRID dataset

However, speaker number 21 is missing and some videos are corrupted, such that after getting the frames, almost 2000 sentences are lost. It is important to add that in the case of unseen speakers protocol, four videos are used for validation, two males and two females, summing a total of 3971 videos, the reminder being used for training, having a total of 28775 videos. In the case of overlapped speakers protocol, 255 randomly chosen videos are used for evaluation and the rest of them will be used only for training.

3.1.2 CAMPUS corpus

For Romanian, a dataset was built by me and my colleague Cristian Cioflan, to use it for training and testing LipNet. It followed the rules mentioned at the beginning, all videos having an approximate time of three seconds. The alignments of the videos were done manually, following the rules that GRID had, so it had few milliseconds of silence at the beginning and at the end, and all words had a start and a stop time mentions. The dataset included 20 speakers, males and females, that were used for training and validation. Videos were filmed in CAMPUS laboratory, using a Nikon camera, on a white background, in a noiseless environment. Anyway, before getting a big dataset, a small one was created and used for few tests, to observe the behavior of the system. It included only 9 speakers, males and females, that have been used for small trainings.

The corpus includes five sentences, with four or five words each. It did not follow the structure of GRID, but it included nouns, pronouns, verbs and prepositions. Table 3.2 highlights all the words that formed the sentences, where some of them repeated for a certain number of times. More exactly, words “anul”, “acesta”, “electronica”, “imagini”, “procesez” appear two times in the corpus, counting a total of 15 words. The sentences used for by the dataset are presented below, but it is also important to mention that in this case the appearance rate of the words is unequal. Also, for an easier approach, “a” and “a”, “s” and “s”, “t” and “t” were considered the same.

- Anul acesta procesez imagini
- Eu prezint licenta anul acesta
- Electronica este pasiunea mea
- Procesez imagini pentru licenta
- Sunt student la electronica

Noun	Pronoun	Verb	Preposition
Anul	Aceasta	Sunt	Pentru
Electronica	Eu	Este	La
Imagini	Mea	Procesez	
Licenta		Prezint	
Pasiunea			

Table 3.2. Words of the CAMPUS corpus, split into speaking parts

CAMPUS is, also, an audio-visual dataset, so it could be used for other AVSR applications. It has a total of 5 hours of short movies that are processed by LipNet to be trained. However, for speaker 20, there is a video corrupted, such that the total number of frames of it is smaller. Depending on the training protocols, there are some choices that have been done, to be able to observe the performances of the system and compare them to the ones obtained using GRID. This was one of the reasons why there are two versions of the dataset: one including only sentences with four words and one including a corpus with sentences made up of four or five words. For unseen speakers protocol, speaker 1 and 2 were used for validation and the rest of 75 videos were used for training. In the case of overlapped speakers a number of 10 random sentences have been used for validation, while the other ones were used for training.

3.2 Performance metrics

LipNet is a system that can be trained in different conditions, being able to adapt depending on the used protocol or to the chosen language. However, in all situations, it measure the performances using the same metrics that will be described below. It offers information during the training and at the end of it, when all the results are stored in a .CVS file, which contains the performances at each epoch. Also, for a better understanding of the obtained feedback, the system shows also what it decoded and what should have been understood during the training. LipNet computes three metrics for each training protocol, which show the performances of it. They calculate the error rates that the system has, including, also, automatic evaluation metrics used for lexical phrase-based optimization. The mean of the values from each epoch are shown and also a normalized average can be observed. The results are obtained by comparing at different levels the supposed output with the real one, depending on the metrics.

First of all, all error rates are computed using sequences, which are considered strings of characters that are obtained by modifying one or more characters. Consequently, the computation of any metric is not that simple, because it has to determine the frequency of errors in different cases.

Character error rate (CER) is an error rate metric that measures the minimum number of necessary operations that are required to get the desired output from the reference text. The number is known as Levenshtein distance and the larger the it is, the more different the two texts are. In some cases, there also some swaps, reason why there is a generalization of the formula, known as Damerau-Lavenshtein, which include these changes as a fourth basic operation. Even if for lengthy sequences or noisy environments, an optimal value is hard to get, there are some automatic algorithms that compute the value, based on the formula from below. It uses n number of characters that are divided

by the sum of the minimal number of character insertions i , substitutions s and deletions d that are necessary to transform the reference text into the desired output. The typical

$$CER = \frac{i + d + s}{n}$$

Word Error Rate (WER) is one of the most common metrics used to measure the performance of any AVSR system. It is based also on the Levenshtein formula, but it works at word level, in comparison to CER which works at phoneme level. It has the same principle, by observing the differences between the reference text and the output. The first step [33] is to compute the error distance in words between the reference and the hypothesized sequences, using the standard weights of the three typical error that can occur, after that using the formula from below, WER can be computed. In this case, n is the total number of words from the reference text, i represents the total number of words that were inserted in order to get the desired output, s is the total number of words that have been substituted and d gives the total number of deleted words.

$$WER = \frac{i + d + s}{n}$$

In fact, WER represents the number of words that have been incorrectly recognized from the total number of words that have been spoken by the speaker. It is useful, also, to validate the language model that would be used later for testing and possible demos. However, WER does not have the ability to analyze the recognized words from a semantic point of view, reason why there are some situations when a word with the same meaning is said and it is considered error because it does not appear in the stored model.

Bilingual evaluation understudy (BLEU) is another performance metric used for measuring the statistical machine translation. More exactly, it evaluates the quality of a text that have been translated from one natural language to another. The more the match is between the two texts, the better the score of BLEU is. It is based on n -gram method and usually, it takes values between 0 and 1, so few translations will lead to a value equal to 1, whereas a perfect match would get a score of 0. Being a modified form of precision, the translation can vary [33] in word choice or in word order, such that the results are influenced by both aspects. Depending on the value of n , there are different types of precisions that are computed, returning models such as unigram, bi-gram or grouped unigram models.

BLEU was frequently correlated to human's judgement, but there are some drawbacks that have been observed. Even if it is considered capable to evaluate any translation, it cannot deal with languages whose word boundaries are not well defined. Another issue that this performance metric has is represented by its tendency to work in favor of short translations. However, LipNet uses this metric to observe the translations that the machine does from natural language to another.

The system returns, also, the normalized values of the metrics mentioned above, because of the possibility to get outputs larger than 100%. In this case, the resulted accuracy would get a negative number, so the operation is necessary to have a clear overview of the performances that LipNet can get in different situations. In this case, the number of mistakes is divided by the sum of deletions, insertions and substitutions, to get a number which is smaller. In fact, it takes the minimum values that the ratio between the sum of weights of the edit operations and the sum of the operations themselves.

Beside these performances metrics, while doing the training, LipNet returns some other information in regard to the way that system learns. More exactly, it computes the loss for each epoch, showing a summation of the errors that the system does for any example from training or validation.

It uses a custom loss function, done with the help of Lambda layer, that is include in CTC. It highlights, in fact, the quality of the behavior of the algorithms after each iteration, showing how well or poor it acts. It uses in fact a loss function for the training data, whose result is returned by the system with the name “loss”. The value decreases after each epoch, until it gets stable and the performance of the system does not get better anymore. Furthermore, after each iteration, the system returns “val_loss”, which represents the cost function of the cross validation. The difference between these two parameters is represented by the moment when the value is computed, so “loss” is obtained during the training epoch, when the model is still adapting, and “val_loss” is given at the end of each epoch, when the model is constant.

All these parameters are helpful in order to get an overview of the performance that LipNet gets. Depending on the training protocol, language and chosen parameters, the values are different. Also, the feedback depends on the number of iterations that the system does, it can get different effects such as overfitting, that appears when the analysis is too close to a part of the dataset. LipNet has different results, depending on the approach that has been taken, so all of them lead to comparisons that highlight both advantages and drawbacks of the system.

3.3 Results obtained for English

LipNet was firstly developed and then optimized for English, reason why the obtained results are the best ones. I used GRID data base in order to train and test the system, trying three out of five protocols. Overlapped speakers and overlapped speakers curriculum were not tested because of the random way of validation and also because of the long time that was required for training. This protocol need different 33 trainings, for each speaker and after that I would have had to choose the best model. Consequently, even if the developers used a large number of iterations (5000epochs) in order to get the mentioned results from [30], I only trained the system for 200 epochs. Also, the proposed batch size was 50, but I adjusted it to 25, because however, the necessary training time was not influenced that much. Also, in this case, the hardware constraints have been taken into account, because a lower batch size require less memory. Moreover, the system stores a model of the what it learns after a certain number of epochs, so even if initially a model was saved after each iteration, I modified it and, in the experiments, that I did, the model was stored only after 25 epochs.

3.3.1 Unseen speakers protocol

This training protocol uses four speakers for validation and the rest of them for training. Following the advice from paper [28], two males and two females were chosen for validation (speaker 1, speaker 2, speaker 20 and speaker 22) and the rest of them were used for training (speaker 3..19, speaker 23..34). The processed followed the aspects mentioned before, so the training was done using 200 iterations, storing 10 models.

When analyzed from the performance point of view, the metrics show a behavior that tell exactly the evolution of the system. Figure 3.2 show the mean of CER, for the 200 epochs that have been used for training, plotting on the horizontal axis the number of iterations that have been done and on the vertical axis the value of the parameter. It varies between 0 and 17, getting the greatest values at the beginning of the training. After that, it starts decreasing, from CER=15 at epoch 10, to CER=4.1 at epoch 60, where the training process is efficient. From that point, the parameter has only small variances between 1.8 and 5, reason why a shorter training of approximatively 200 epochs would have been enough.

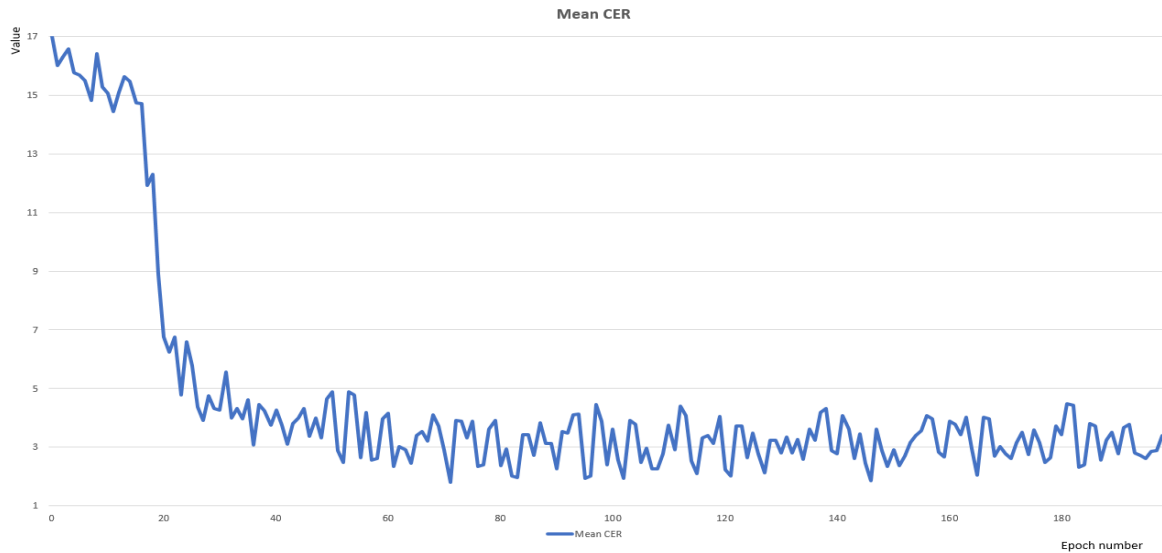


Figure 3.2. Representation of mean CER for unseen speakers protocol, English language. Y axis: value of mean CER, X axis: Epoch number

The second analyzed metric is WER, which shows the error rate when analyzing the words. Figure 3.3 shows the evolution of the parameter, for 200 epochs, where it gets value between 0.8 and 5.8. On the vertical axis, it is represented a mean of WER obtained during the training and on the horizontal axis it is represented the number of epochs that LipNet did. The poorest performances are obtained at the beginning of the training, in the first 12 epochs, but after that the training becomes more efficient and WER keeps decreasing until epoch 60, where it has the value 1.4. From that point, the performances do not improve and LipNet becomes stable, so WER has small variances between 1.4 and 1.6.

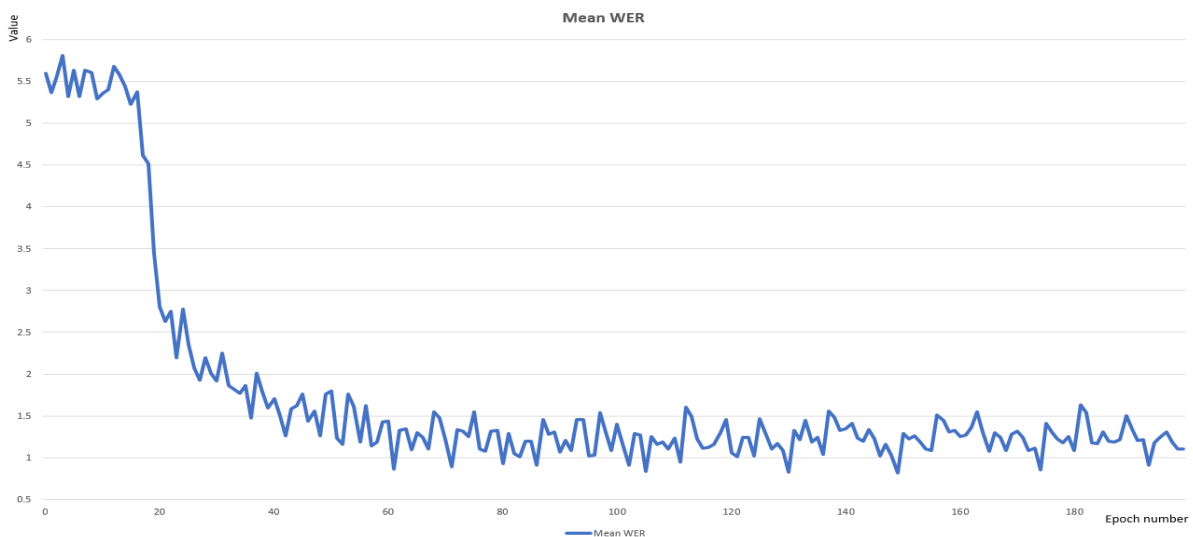


Figure 3.3. Representation of mean WER for unseen speakers protocol, English language. Y axis: value of mean WER, X axis: Epoch number

BLEU is the third performance metric that is analyzed, as shown in Figure 3.4 from below, where on the horizontal axis is plotted the number of epochs and on the vertical one the values of the mean of the parameter. The results are between 0.3 and 1, so the number of translations is small from the very beginning. However, in the first 20 epochs, the results are poor, with results that vary between 0.3 and 0.4. After that, a great increase can be observed, for the next 20 epochs, such that

BLEU is doubled at epoch 40, getting a value of 0.72. From that moment, there are only small variances, getting a maximum BLEU=0.86, reason why there is no big improvement in the performance of the system.

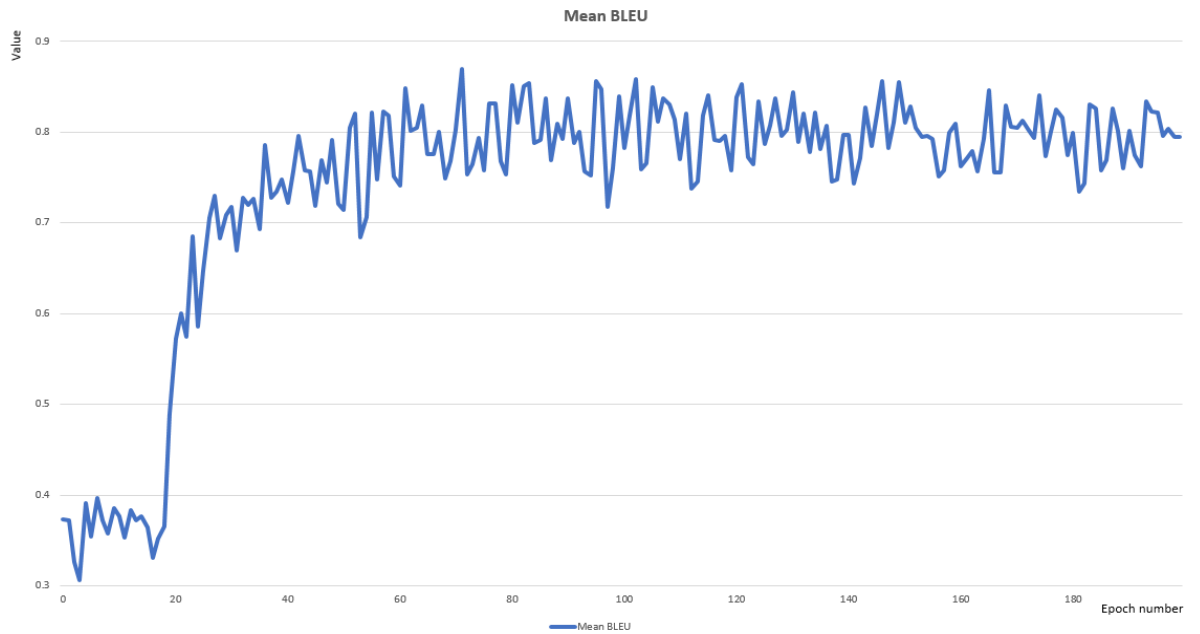


Figure 3.4. Representation of mean BLEU for unseen speakers protocol, English language. Y axis: value of mean BLEU, X axis: Epoch number

Unseen speakers protocol has an efficient training, reason why after a relative reduced number of epochs, the performances do not improve anymore. When analyzing the normalized means of CER, WER and BLEU, shown in Figure 3.5, where on the horizontal, it is represented the number of epochs and on the vertical it is plotted the values, it is easily to observe a certain pattern for all metrics. More exactly, at the beginning of the training, LipNet gets values that are not desired, but after that, the system starts learning, so the interval between epochs 15 and 60 returns values that that vary a lot. When CER and WER decreases, BLEU increases, process that is normal if the learning process is efficient. The training process does not get big improvements afterwards, reason why instead of 200 epochs, it could have been shorter.

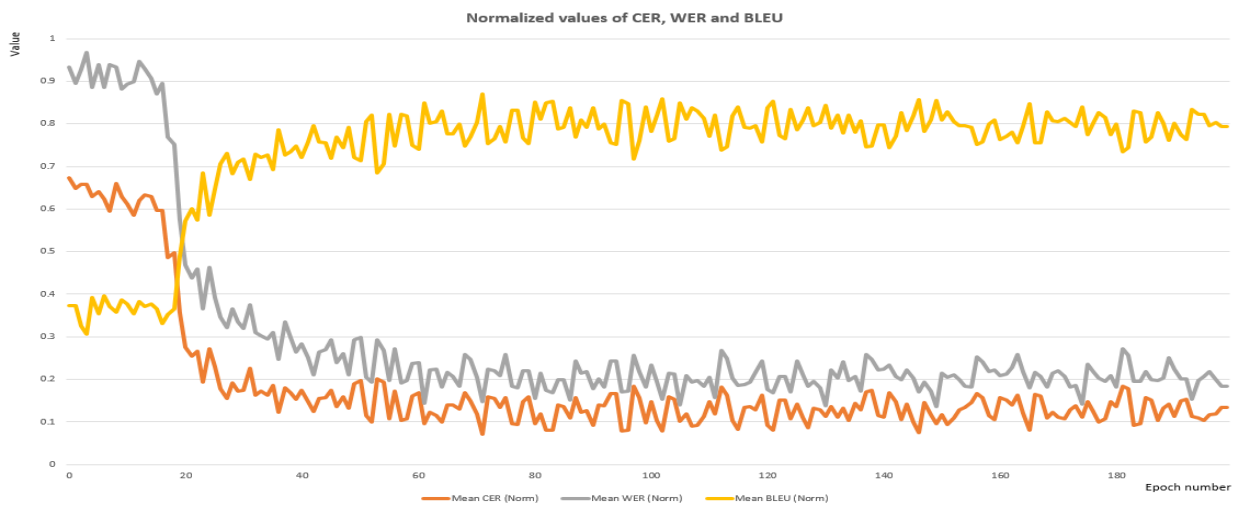


Figure 3.5. Representation of normalized values of CER, WER, BLEU for unseen speakers protocol, English language. Y axis: value of the normalized parameters, X axis: Epoch number

For the validating process, the system used 4 speakers and a number of 25 different sequences. They all have 6 words and follow the structure of GRID. When analyzing the sentences and what the system decoded, it is observed that only 7 of them were perfectly understood. The other ones, which are wrongly decoded, are shown in Table 3.3, where the red color highlights the words. All verbs and colors are correctly recognized, but a total of 24 words are not identified. More exactly, 4 characters, 9 number, 7 adverbs and 4 prepositions are detected wrongly.

Real sequence	Decoded sequence
place white with o nine now	place white with o eight now
place white with p one please	place white with p eight please
place white with p two again	place white with p zero now
place white with v four soon	place white with v one soon
place white with v six again	place white with v six now
set blue at e nine now	set blue in d nine now
set blue at f one please	set blue at f set please
set blue at f two again	set blue at f two now
set blue at f zero soon	set blue at f seven soon
set blue at l three now	set blue i z two now
set blue at l five please	set blue at f eight please
set blue at l six again	set blue at l six now
set blue at r eight soon	set blue at r nine soon
set blue at r nine please	set blue at r eight please
set blue at s zero again	set blue in k zero now
set blue at z two soon	set blue at z two now

Table 3.3 Real and decoded sequences for unseen speakers protocol, English language.

3.3.2 Unseen speakers curriculum protocol

This training protocol is an advanced version of the training protocol that has been mentioned before. The curriculum rules proposed by the developer at [32] were kept, such that the maximum number of decoded words from a sentence was six. In this case, the necessary time for training was greater, due to the way that data was processed. I used the same four speakers for validation (speaker 1, speaker 2, speaker 20, speaker 22) and I used the other ones for training. The system did 200 iterations and stored 10 models.

CER is the first performance metric that is analyzed, showing the error rate that the protocol gets, when analyzing the character recognition. Figure 3.6 shows the evolution of the mean of the parameter, by plotting on the vertical axis its values and on the horizontal axis the number of epochs that the system did. The results vary between 2 and 17, the worst ones being at the beginning of the training. From epoch 12, where CER=15.9 to epoch 40, where CER=2.8, it is a great decrease, but after that the system gets stable and there are only small variances.

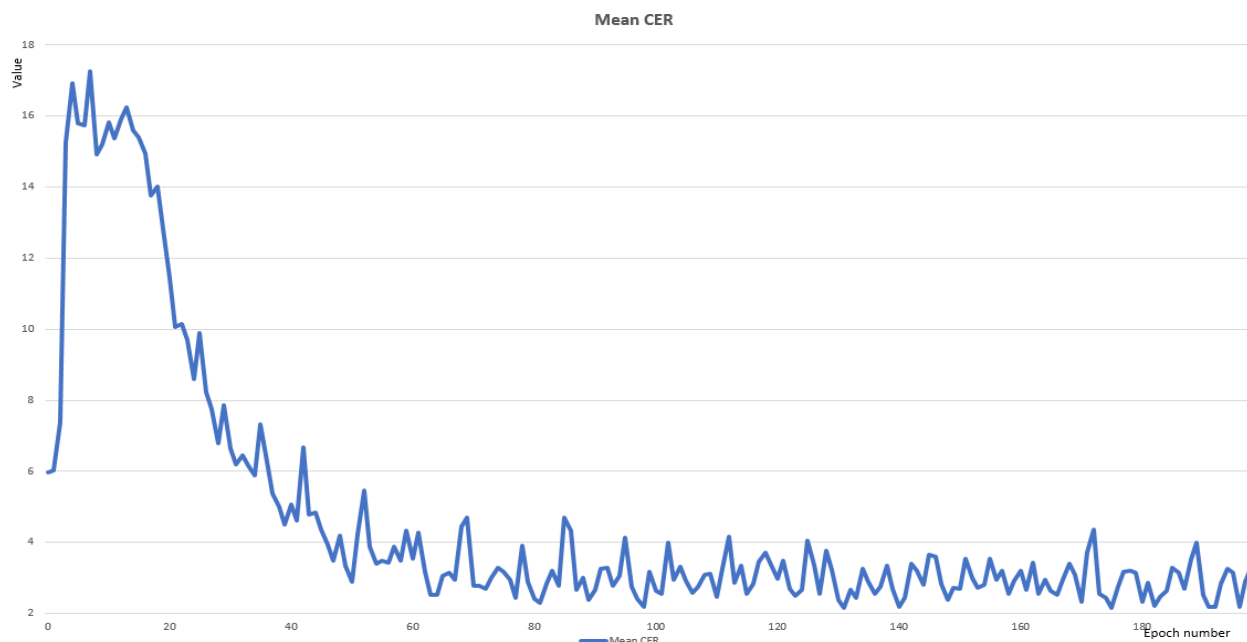


Figure 3.6. Representation of mean CER for unseen speakers curriculum protocol, English language. Y axis: value of mean CER, X axis: Epoch number

The evolution of the mean of WER can be observed in the Figure 3.7, where on the vertical axis, it is represented the values of the parameter and on the horizontal one, it is plotted the number of epochs that the system did. The result varies between 2.96 (at epoch 161) and 5.6 (at epoch 8), having poor performances at the beginning of the training. Between epochs 14 and 101, LipNet learns in an efficient way, but after that it gets stable and the learning process does not get big improvements, WER having small variances 3 and 4. In this case, a shorter training of maximum 160 epochs would have been enough.

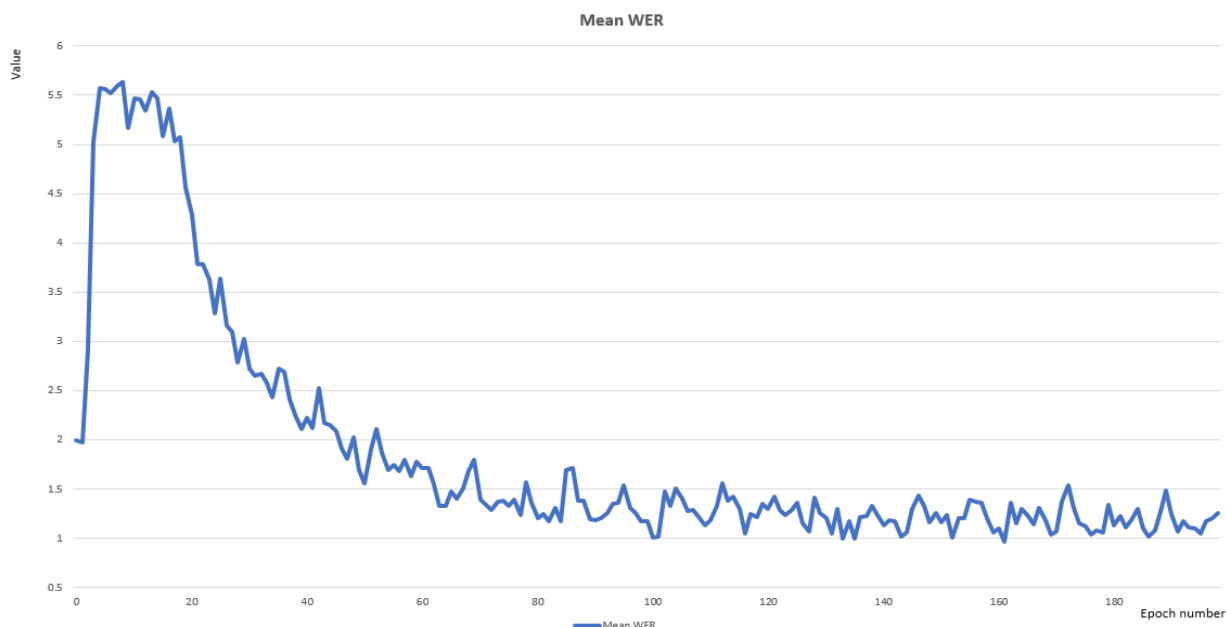


Figure 3.7. Representation of mean WER for unseen speakers curriculum protocol, English language. Y axis: value of mean WER, X axis: Epoch number

For this training protocol, the values that the mean of BLEU has are plotted on the vertical axis of the Figure 3.8, where the horizontal axis show the number of epochs. The parameter varies between 0.25 and 0.85, so it does not have perfect matches while training, but, on the other hand, it is not necessary a big number of translations, not even at the beginning. Between the 9th and 63rd epochs, BLEU gets a good increase of 0.6 points, but after that there are only small variances, so the learning process is not so efficient anymore.

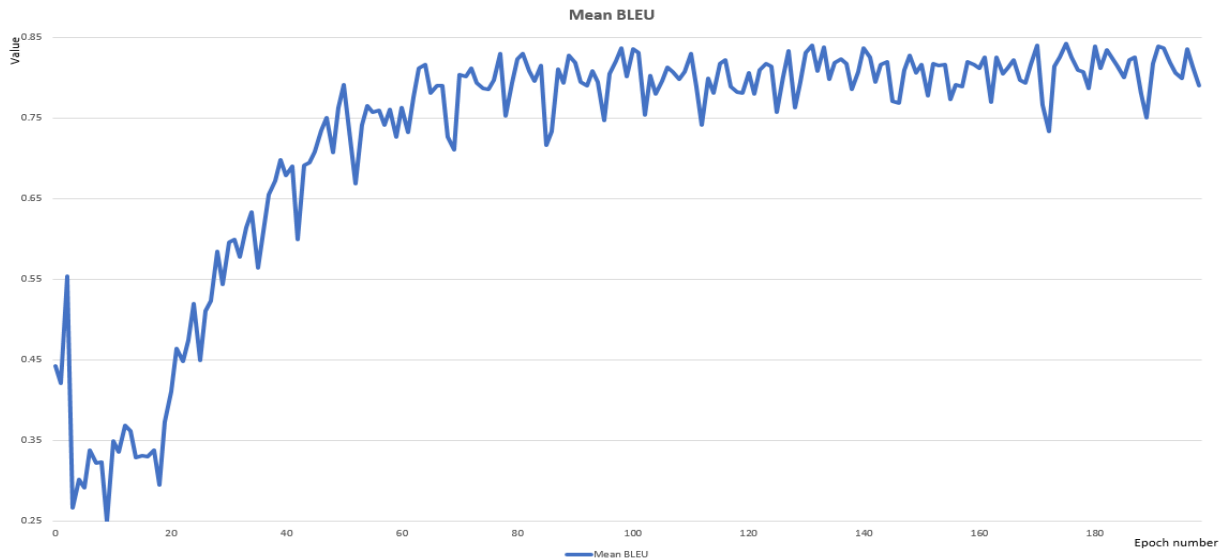


Figure 3.8. Representation of mean BLEU for unseen speakers curriculum protocol, English language. Y axis: value of mean BLEU,X axis: Epoch number

This training protocol is a particularization of unseen speakers protocol, reason why the expected results should be better. Even if the system does not learn very fast, it is desired to get good performances. When analyzing the normalized values, there is a pattern that can be observed in Figure 3.9, where CER and WER decrease almost at the same time with the increasing of BLEU. CER and WER start improving almost at the same time, starting with epoch 16, while results of BLEU get better approximatively 2 epochs later. However, the normalized values of the mean of these three parameters, show that the character error rate is smaller than the word error rate.

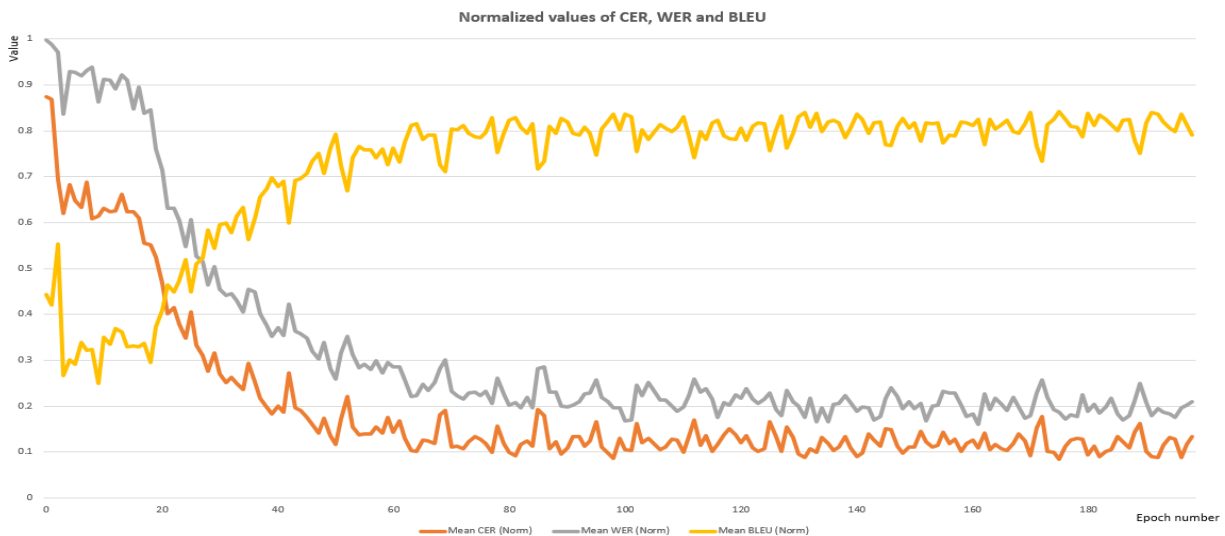


Figure 3.9. Representation of normalized values of CER, WER, BLEU for unseen speakers curriculum protocol, English. Y axis: value of the normalized parameters, X axis: Epoch number

For the validation process, I used for speakers, counting a total of 25 sentences, made up by 6 words each. When analyzing the results, 8 sequences are correctly decoded and the other 17 ones have different mistakes, that can be observed in Table 3.4. More exactly, in the incorrectly detected sentences, there are 25 words that are wrongly recognized, that are colored with red: 2 numbers, 7 prepositions and 16 characters.

Real sequence	Decoded sequence
place white by d eight please	place white by c eight please
place white by d nine again	place white by c nine again
place white by k one soon	place white by d one soon
place white by k three again	place white by c three again
place white by q six please	place white by u six please
place white by x eight now	place white with v two now
place white by x nine soon	place white by h nine soon
place white by y zero please	place white by p four please
place white in c eight now	place white at g eight now
place white in c nine soon	place white at g nine soon
place white in d one again	place white in c one again
place white in d zero please	place white at c zero please
place white in p six now	place white at b six now
place white in p seven soon	place white in b seven soon
place white in p eight please	place white i b eight please
place white in p nine again	place white i b nine again

Table 3.4 Real and decoded sequences for unseen speakers curriculum protocol, English language.

3.3.3 Random split protocol

The simplest protocol of LipNet is called “random split” and it choses one speaker for validation, while the other ones are used for training. Because of the simple approach that it has, it is expected to get the best performances, but the training that the system does when using this protocol is not so accurate, so the obtained metrics do not have so good values. There are 20 models that are stored, the last one having the best results. The protocol is analyzed using the three performances metrics presented above and their normalized results. Figure 3.10 shows the values of the character error rate, after each epoch, for the 200 iterations that the system did. The results are between 0 and 17 and it is easily to observe that that the greatest values are obtained at the beginning of the training, in the first 12 epochs. At that point the system returns less than a half of the characters. After that, CER decreases until it gets stable around the 90th epoch, when the value is around 0.5. After that, the results only have small variances between 0.6 and 0.4, reason why additional training would have been useless.

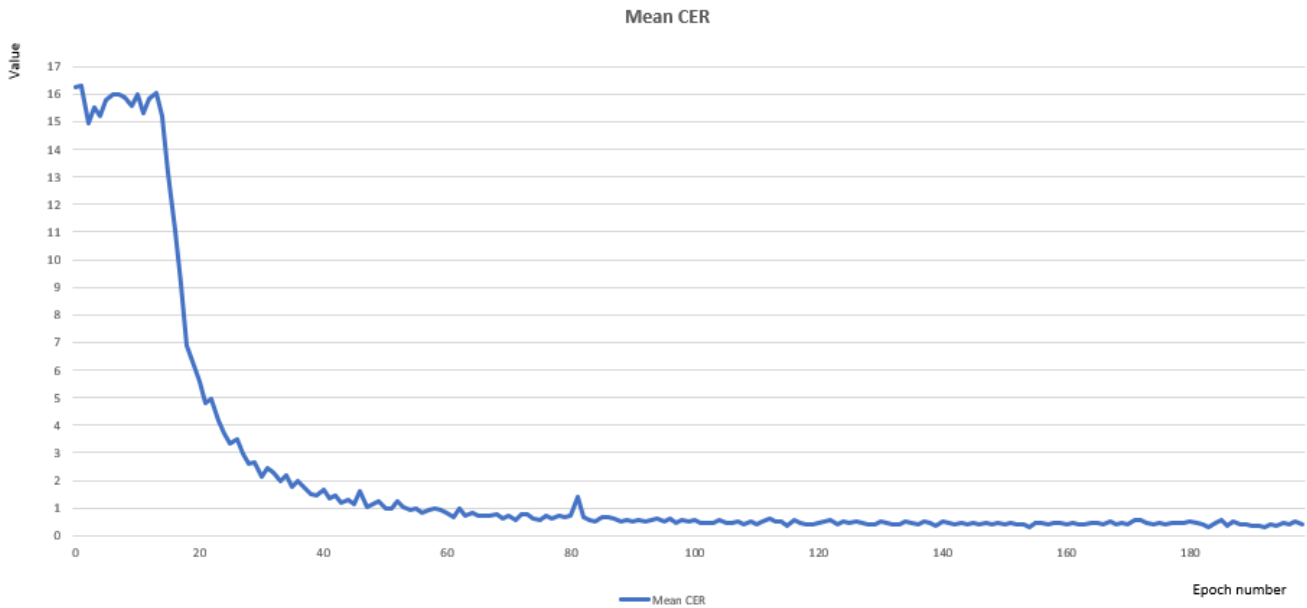


Figure 3.10. Representation of mean CER for random split protocol, English language. Y axis: value of mean CER, X axis: Epoch number

The second analyzed metric is WER, shown in Figure 3.11 from below, whose values vary between 0 and 5.6. On the vertical axis, it is plotted the values of WER and on the horizontal axis it is represented the number of epochs that the system did. Taking into account that the maximum number of words from the trained sentences is 6, at the beginning of the training, LipNet gets poor performances. In the first 12 epochs, WER has results between 5 and 6, but after that it decreases for the next 85 epochs, getting a value of 0.31 at the 90th epoch. From that point, the system becomes stable and there are only small variances, so the performances do not improve a lot.

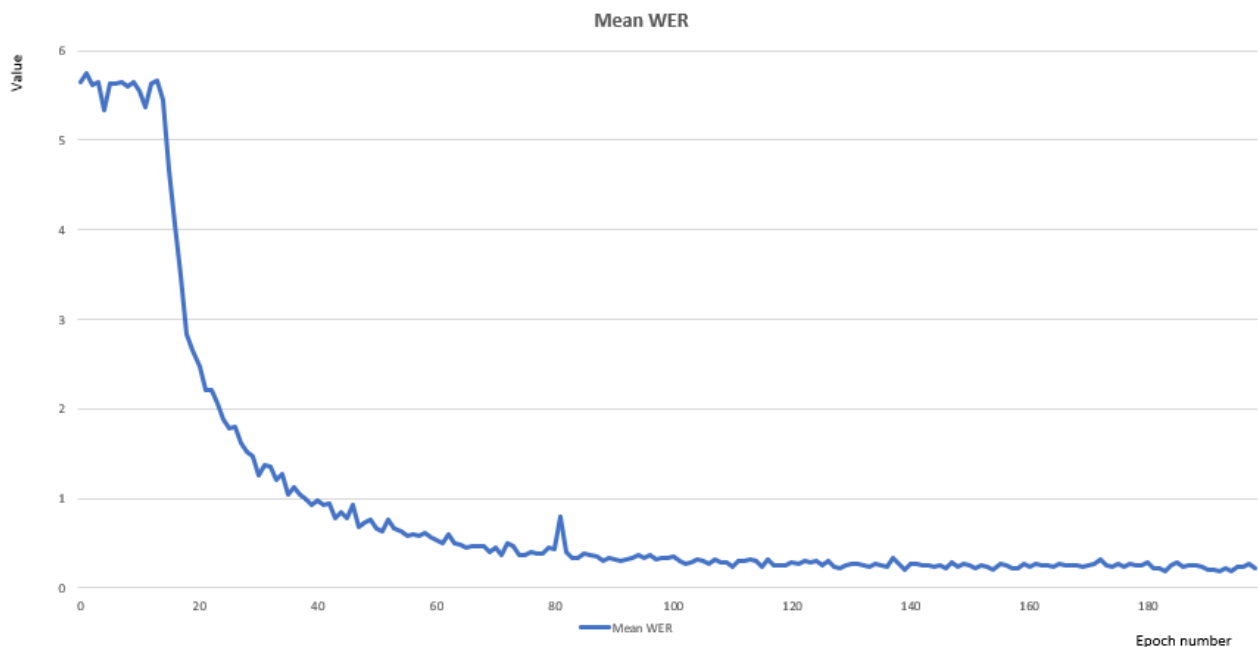


Figure 3.11. Representation of mean WER for random split protocol, English language. Y axis: value of mean WER, X axis: Epoch number

BLEU is the third performance metric, that takes values between 0 and 1, depending on the number of translations that the machine did. Following the pattern described before, Figure 3.12 shows the variation of the parameter, where on the Y axis is represented the value of BLEU and on the X axis the number of epochs. At the beginning of the training, the result is 0 or close to this number, and it start increasing until the 84th epoch. From that point, the metric has very small variances between 0.95 and .096, so the performance of the system does not improve anymore.

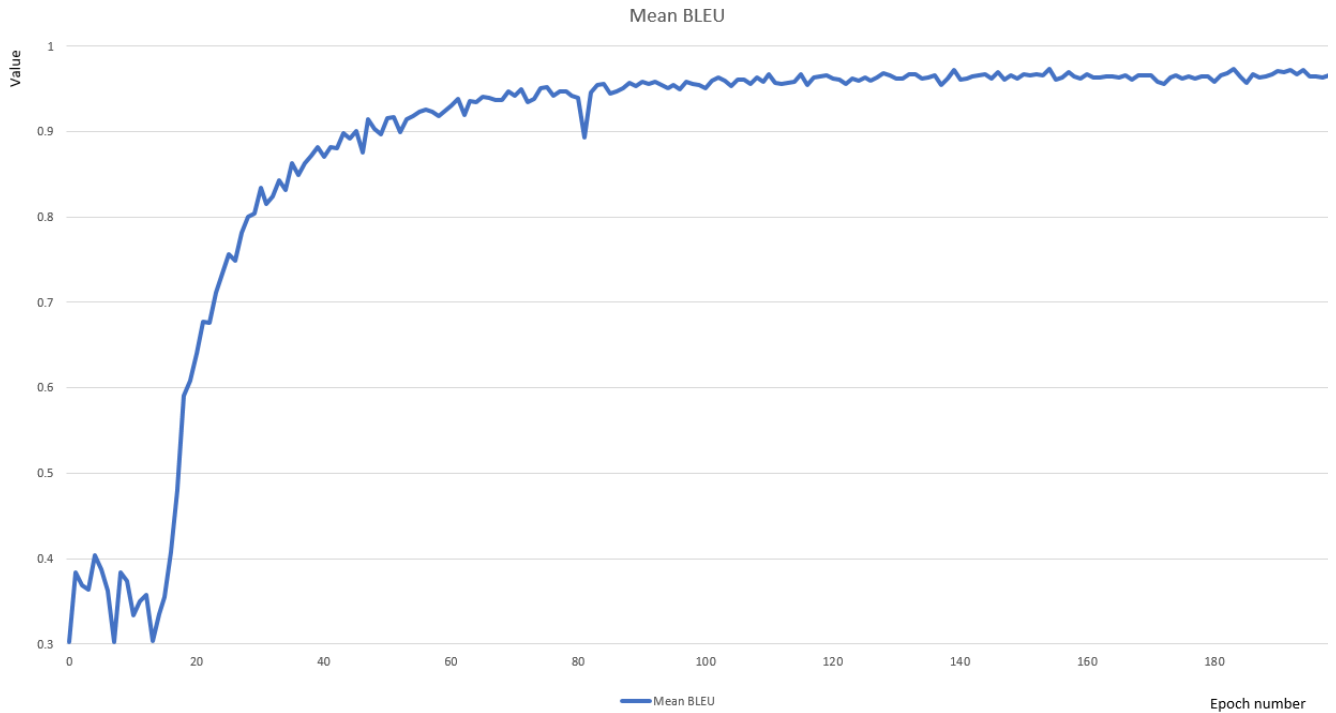


Figure 3.12. Representation of mean BLEU for random split protocol, English language. Y axis: value of mean BLEU, X axis: Epoch number

So, this training protocol gets good performances, but not the best ones. The normalized values of the mean of CER, WER and BLEU are shown in Figure 3.13. Therefore, it has poor results at the beginning of the training, but after approximatively 13 epochs, the values improve. Between the 13rd epoch and the 90th one, the results keep modifying, getting values that are closer to the desired one, such that CER and WER decrease and BLEU increase. After that the system gets stable, reason why a training of maximum 150 epochs would have been enough. Figure 1 from below highlights the normalized values of the performance metrics, where vertical axis shows the obtained values and horizontal axis shows the number of iterations that LipNet did. It is easily to observe that WER and CER decrease almost at the same time, behavior that is normal. While WER has bigger values, CER is smaller from the very beginning, because the necessary characters that must be changed is small. On the other hand, BLEU start increasing approximatively at the same time when the other two metrics decrease, so the system works properly. In all cases, at epoch 80, there was a spike which is shown in all the plotted graphs.

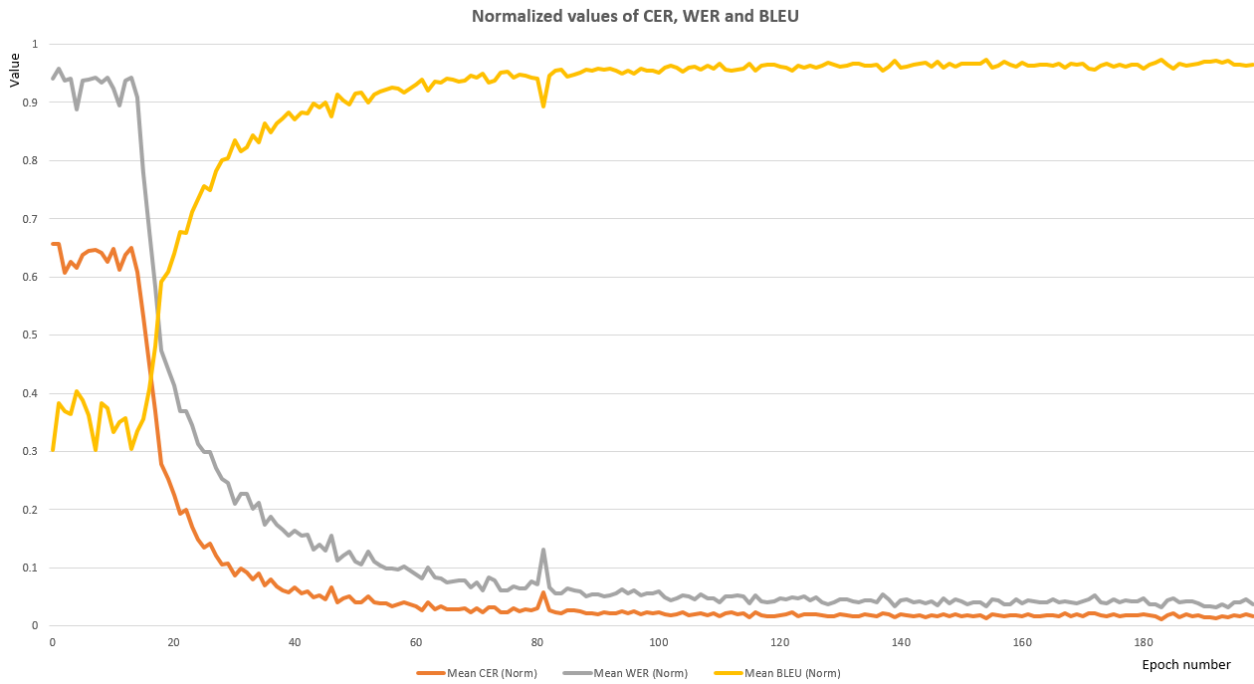


Figure 3.13. Representation of normalized values of CER, WER, BLEU for random split protocol, English language. Y axis: value of the normalized parameters, X axis: Epoch number

For validating this protocol, a number of 25 sentences, all of them having 6 words, were used. From these sequences, 17 of them were perfectly decoded, while the other 9 ones can be observed in Table 3.5. On the left column, it can be observed the real sentences used for validation and on the right column, there are presented the decoded sentences. When analyzing the differences, in all cases the error appears at fourth word, colored with red, which is in fact a character from the alphabet. All the other words from the structure described before are correctly detected and recognized. This shows that the pronunciation of different phonemes is still difficult to identify, even if they are not considered visemes. For a better performance from this point of view, it is necessary a larger dataset that includes these characters and a clearer pronunciation of them. It also detects wrongly some prepositions, confusing “in” and “at” in multiple cases.

Real sequence	Decoded sequence
lay green at z three please	lay green at l three please
place white at p nine again	place white in p nine again
set red in t seven now	set red at t seven now
place red in c three soon	place red in j three soon
place blue by n one please	place blue by a one please
place blue at l four now	place blue i l four now
set white at t six again	set white in g six again
place green at v eight please	place green at d eight please
lay blue at o eight soon	lay blue at u eight soon

Table 3.5. Real and decoded sequences for random split protocol, English language.

3.3.4 Comparison between the obtained results

Depending on the training protocol, the system has a different behavior and obtains different performances. In the three tested situations, there are three main steps of the training process that correspond to different performances. In all cases, at the beginning of the training, the results are poor, as values of CER, WER and BLEU are comparable. After that, there is a part when the learning process starts and the system becomes efficient. Depending on the protocol, it takes a different number of epochs, but for all of them, there are great variances of the values of the performance metrics from one iteration to another. However, when the learning process is finished, the system becomes stable and the variances of the metrics are small, with no great improvements.

All tested protocols obtain different results of the performance parameters and are able to decode more or less from a sequence. More exactly, random split protocol needs more epochs to learn, but after that the system is more stable and the variances are smaller than 0.1. On the other hand, unseen speakers protocol needs less epochs for the learning process, but it is less stable and the results vary more. Also, even the best results of this protocol are poorer than the ones that random split can obtain, such that mean of CER, WER and BLEU show a better improvement of the second mentioned protocol. From another point of view, unseen speakers curriculum protocol has results that are placed between the other two protocols. More exactly, the training process needs more epochs to learn, but however, the final results are not very stable and the performances are not the best ones.

When analyzing the decoded sequences used for validation, random split protocol has the fewest mistakes. It gets only 13 words incorrect detected in comparison to 24 or 25 words for unseen speakers and unseen speakers curriculum protocol. Also, this protocol gives the most correct identified sentences, counting a total of 17, in opposition to unseen speakers protocol that recognizes only 7 sequences.

Therefore, the obtained results highlight that random split protocol is the most efficient one, so it is recommended to use the last model that it stored for any further application. It recognizes most of the words and the performance metrics show the best improvements. It needs a longer training time, so in case of any constraints from this point of view, this choice is not so efficient. From another perspective, this protocol uses less speakers for validation, so this could influence also the results. On the other hand, if the training process must be fast, unseen speakers protocol fits better the requirements, because it needs less than 50 epochs to learn. After that, the system gets stable and there are not important improvements.

3.4 Results obtained for Romanian

LipNet was firstly developed for English, but the researchers stated that it could be adapted for any other language. Starting from this idea, CAMPUS dataset has been created and it has been used for training, trying three out of five protocols. For a better understanding of how the system works and for more useful comparisons, I used the same protocols that I trained for English: unseen speakers, unseen speakers curriculum and random split. In this case the number of iterations was increased from 200 to 400. Moreover, the maximum batch sized that the system could use in this case is a lot smaller, so the optimum chosen value was five. However, because of the changes that have been done, LipNet was tested in more versions that will be described below. Also, because of the alphabet and phonemes differences between the two languages, I did two sets of tests: one using English alphabet and one using Romanian alphabet and groups of letters. The results are different, because the system recognized the words in different ways.

3.4.1. Dataset with variable number of words

The first tests that have been done include a dataset that had sentences of four and five words, aspect which influenced the performances that the system obtained, because it was created for a variable length of the words, not of the sequences. In this case, in addition to this constrain, the phoneme list that has been used was the same that I used for English trainings, reason why some groups of letters have been excluded, such as “ce” or ”ci”. This aspect led to a wrong detection of some words as “licenta”. Even if the dictionary was updated with the sentences from Romanian, when the system tried to validate the decoded text, it got confused because of the available phonemes. For this case, all three training protocols have been tested, in order to observe the evolution of the performance metrics and the ability to learn the sequences.

3.4.1.1 Unseen speakers protocol

CAMPUS dataset has fewer videos, reason why, for unseen protocol, only two speakers were used for validation, one male and one female (speaker 1 and speaker 2). The rest of 18 speakers have been used for training. This time, LipNet did 400 iterations, but stored 40 models, supposing that the changes from one epoch to another are more obvious.

Therefore, the performance metrics show the behavior of the system when adapted to another language. Figure 3.14 shows the values of the mean of CER, plotted on the vertical axis, for 400 epochs, plotted on horizontal axis. In this case, result varies between 16 and 29, such that it is easily to observe the improvement of LipNet. At the beginning of the training, mean of CER has the worst results, between 23 and 29, but after that, the system starts learning. In the first 4 epochs there is a decrease of 3 points, but after that it does not improve for a while, for the next 60 epochs. After there, another decrease of CER is observed, so the system learns for other 100 epochs, until it gets stable around the value of 17. From that point, variation of the parameter is small, and the improvements of the system are not so notable, when analyzing the character error rate.

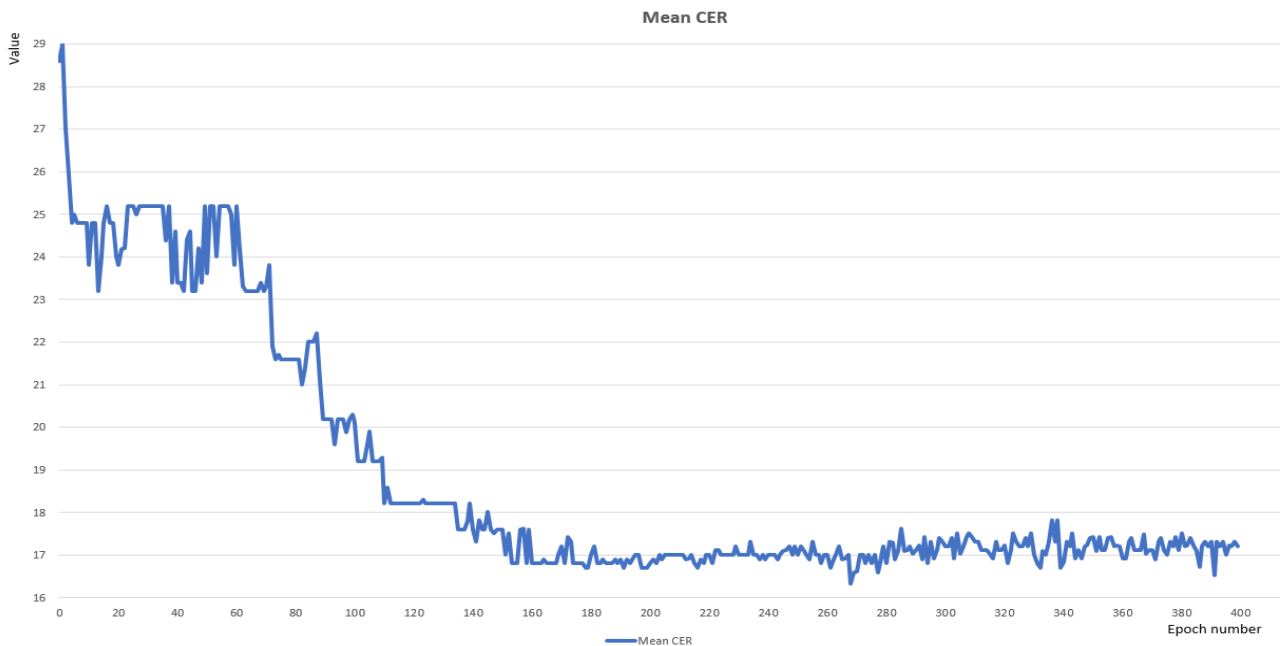


Figure 3.14. Representation of mean CER for unseen speakers protocol, Romanian language. Y axis: value of mean CER, X axis: Epoch number

Another performance metric shows the ability of the system to identify words in the sentence. Figure 3.15 shows the evolution of mean WER, whose values are represented on the vertical axis, when number of epochs increase, which are plotted on the horizontal axis. Starting from the idea that all sequences had maximum five words in the sentences, the variation of WER between 3.7 and 4.8, shows an average of 2 decoded words. The behavior of this parameter is not the typical one, because the results of the first epochs are not the worst one. Even more, the first 140 epochs get the lowest values of WER, but after that the ability of the system to decode the words gets poor.

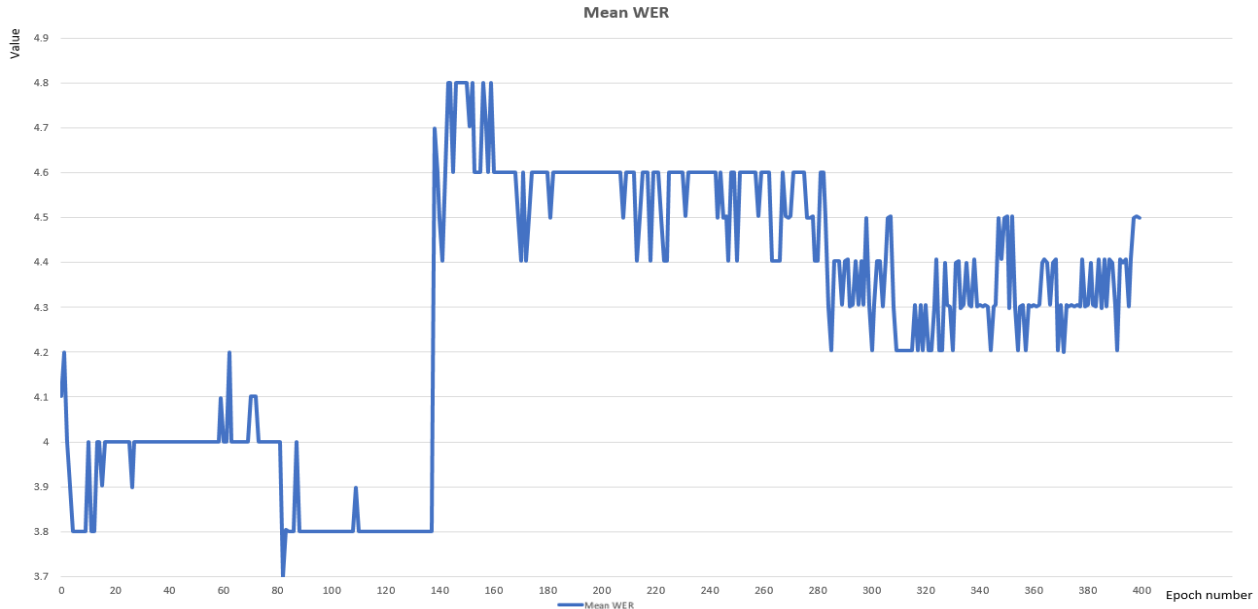


Figure 3.15. Representation of mean WER for unseen speakers protocol, Romanian language. Y axis: value of mean WER, X axis: Epoch number

BLEU is the third metrics that is analyzed and it should get values between 0 and 1, where 1 represents a minimum number of needed translations in order to get the desired output. Figure 3.16 shows the evolution of the mean of BLEU, its values being plotted on the Y axis, when taking into account the number of iterations that the system does, plotted on the X axis. While in the first 5 epochs there a big variation of the parameter, between 0 and 0.45, for the next 100 epochs it keeps decreasing, until it gets a stable value. After that, for almost 200 epochs it gets small variations, between BLEU=0.25 and BLEU=0.3, values that show a poor performance of the system. From that point, mean increases with 0.05 points and from that point, there are only variances that can be observed. Anyway, this parameter does not show a behavior that follows the pattern presented before. The improvements that the system obtain are not the best ones, but however, it is obviously that it becomes able to decode part of the text.

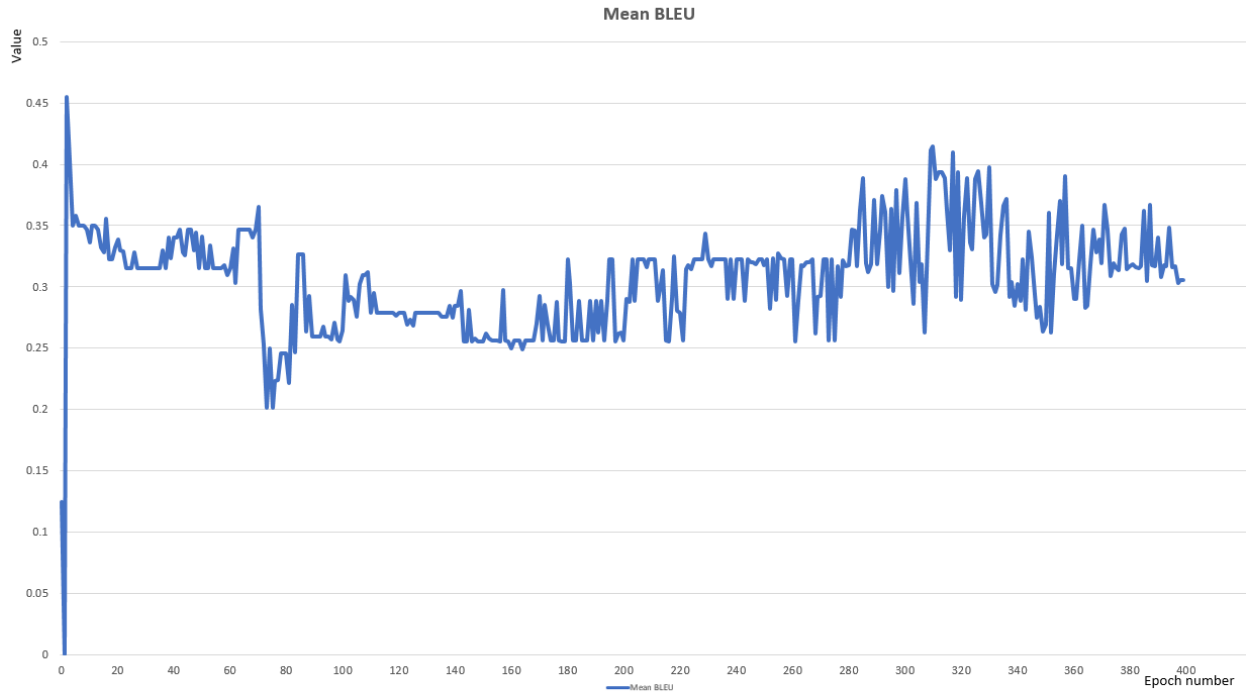


Figure 3.16. Representation of mean BLEU for unseen speakers protocol, Romanian language. Y axis: value of mean BLEU, X axis: Epoch number

Unseen speakers protocol has an unusual behavior when adapted to another language. This is influenced by the size of the dataset, but also, by the variable length of the sequences. When analyzing the normalized means of CER, WER and BLEU, shown in Figure 3.17, where on the horizontal, it is represented the number of epochs and on the vertical it is plotted the values, it is easily to observe a lack of pattern for all metrics. More exactly, CER and WER do not evolve in the same way and the increasing of BLEU is too small. However, CER starts decreasing, almost at the same time with the improvement of BLEU, so the system works properly, but the normalized mean of WER, shows that the system can identify characters, but not whole words.

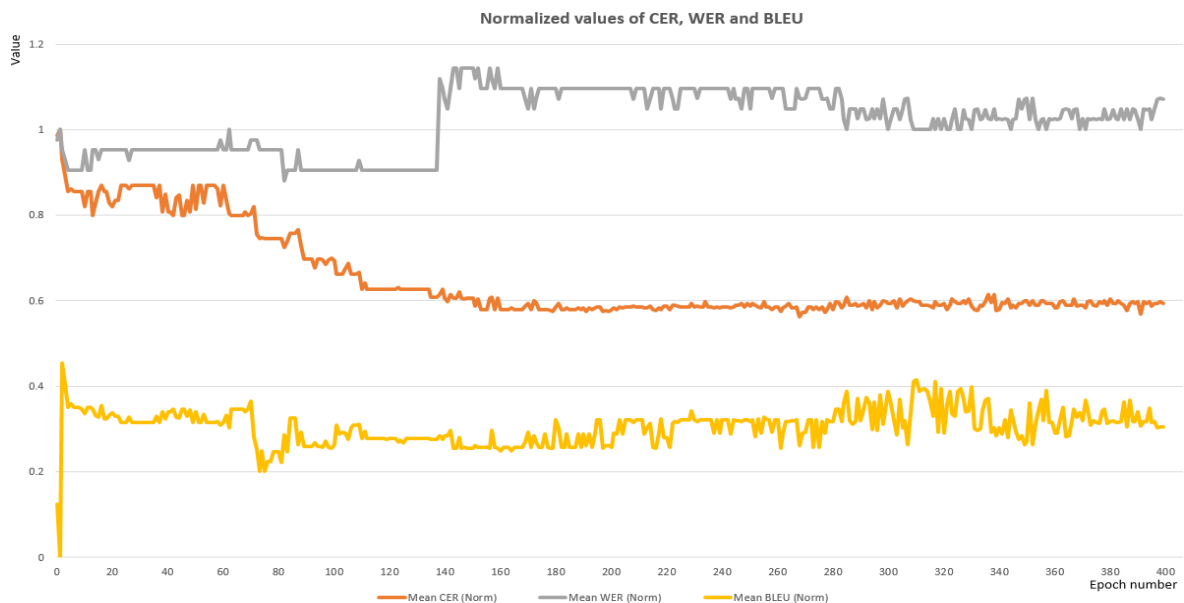


Figure 3.17. Representation of normalized values of CER, WER, BLEU for unseen speakers protocol, Romanian language. Y axis: value of the normalized parameters, X axis: Epoch number

For the validating process, the system used 2 speakers and a total of 5 sentences, counting a total of 21 words. When analyzing the real sequences and what the system decoded, shown in Table 3.6, it can be observed that there is no complete sentence that has been detected. The color red from the table highlights the wrongly recognized words, but, on the other hand, the identified words are randomly and do not necessary form a sentence. Moreover, the system does not have the ability to recognize long words or detects only part of them, such as “rezinta” instead of “prezinta”.

Real sequence	Decoded sequence
anul acesta procesez imagini	anul acesta la este iaeita
electronica este pasiunea mea	eu este iaeita mea
eu prezint licenta anul acesta	eu rezina anul iaeita
procesez imagini pentru licenta	eu rezina pentru iaeita
sunt student la electronica	sunt rezina la iaeita

Table 3.6 Real and decoded sequences for random split protocol, Romanian language.

3.4.1.2 Unseen speakers curriculum protocol

This is an advanced version of the before training protocol, so it is expected to get better performances than it. In this case, there are only 320 iterations that have been done, because I observed that the system does not improve its results anyway. Because of the variable length of the sentences, the obtained values still have an unusual behavior. In order to validate the results, I used two speakers, one male and one female, (speaker 1 and speaker 2), following the pattern from protocol that represents the basis of this one.

Mean CER is plotted in the Figure 3.18, having the values represented on the vertical axis and the number of epochs represented on the horizontal axis. Ignoring the first epoch, at the beginning, until iteration number 80, the results are poor. However, it can be observed a leaning path, because the mean decreases from CER=27.39 at epoch 2, to CER=18.1 at epoch 110. After that, the variances are small and the performance do not improve anymore.

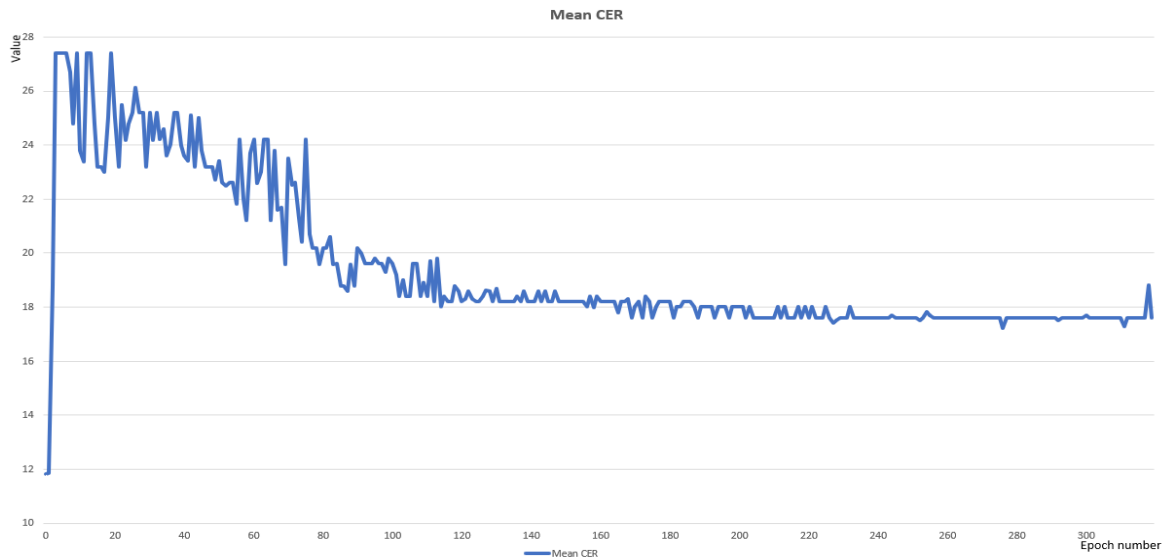


Figure 3.18. Representation of mean CER for unseen speakers curriculum protocol, Romanian language. Y axis: value of mean CER, X axis: Epoch number

WER shows the ability of the system to recognized words in a sequence. The evolution of this metric is plotted in Figure 3.19, where the mean WER is represented on the vertical axis and the number of epochs is represented on the horizontal axis. The values vary between 2 and 5, so it clear that the system does not recognize any complete sentence, but it still recognizes at least one word. The maximum is also influenced by the variable length of the sequence. However, the behavior of this parameter highlight that the learning path of the system is not a usual one, because WER start increasing after epoch 180, instead of decreasing.

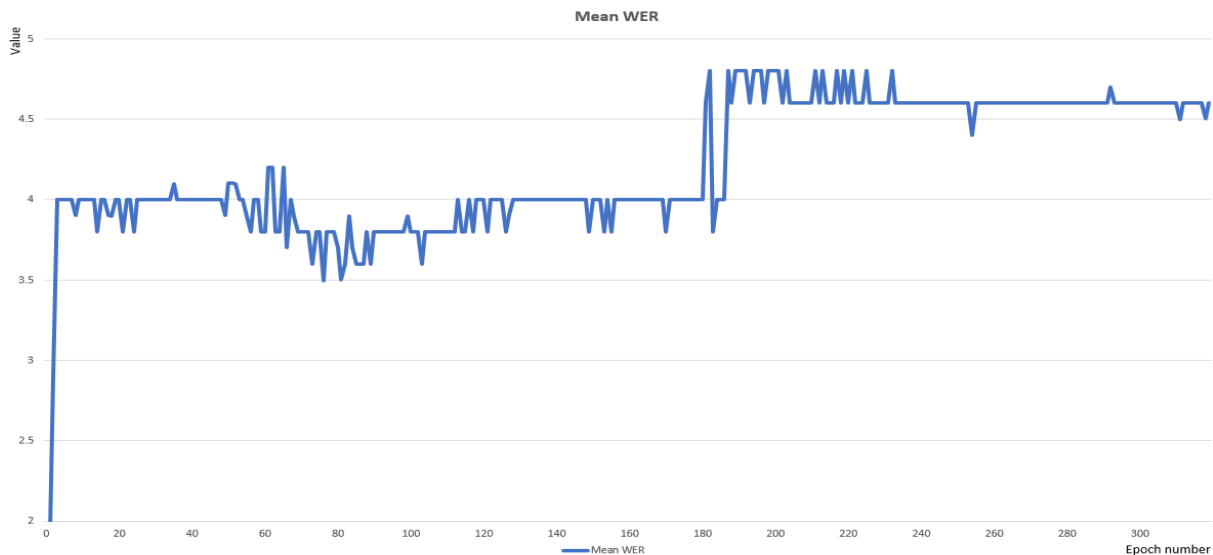


Figure 3.19. Representation of mean WER for unseen speakers curriculum protocol, Romanian language. Y axis: value of mean WER, X axis: Epoch number

BLEU is the third metric that is analyzed to observe the performances of the system for Romanian language. The evolution of it is shown in Figure 3.20, where on the horizontal axis it is represented the number of epochs and on the vertical axis it is plotted the values of mean BLEU. This metric varies between 0.2 and 0.6, but it does not follow a pattern that could be described. At the beginning of the training, the system needs less translations, even if it is unusual, but after that, the number of translations vary from epoch to epoch, such that mean BLEU is between 0.22 and 0.35. These values are correlated with both structure of the dataset and variable length of the sentences.

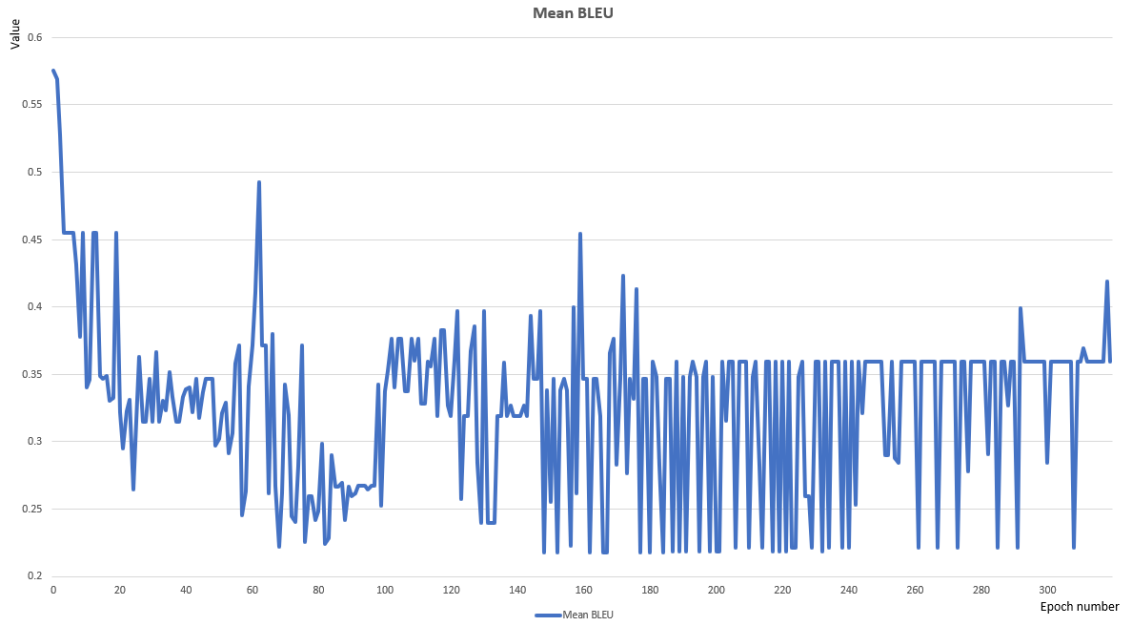


Figure 3.20. Representation of mean BLEU for unseen speakers curriculum protocol, Romanian language. Y axis: value of mean BLEU, X axis: Epoch number

This particular form of the unseen speakers training protocol should have obtain better results. However, after analyzing the performances metrics, is notable that the system has an unusual behavior, that cannot be predict. The improvements are not so great, even if the number of epochs increases, reason why the normalized values of CER, WER and BLEU do not follow any pattern. The evolution of these parameters does not seem correlated, reason why the decoded sentences do not look close to the real ones. Table 3.7 shows the differences between the desired and the real output, using 5 sequences for validation, where red highlights the incorrect words. However, in this case all decoded sentences seem the same, starting with the same word and having only 4 words. It seems that the system learnt some random words that are recognized later.

Real sequence	Decoded sequence
anul acesta procesez imagini	eu la este iaeita
electronica este pasiunea mea	eu este iaeita mea
eu prezint licenta anul acesta	eu rezina anul iaeita
procesez imagini pentru licenta	eu rezina pentru iaeita
sunt student la electronica	eu rezina la iaeita

Table 3.7. Real and decoded sequences for unseen speakers curriculum protocol, Romanian language.

3.4.1.3 Random split protocol

This is considered the simplest training protocol, having a simple approach of the validation process. However, when adapted to Romanian, it gets a behavior that is unusual, because of the evolution of the performance metrics. However, because there is only one speaker used for validation, the results seem better.

Mean of CER is shown in Figure 3.21, where on the vertical axis it is plotted its values and on the horizontal axis it is represented the number of epochs. The result varies between 16 and 29, the worst values being obtained until the epoch 71. From that point until epoch 160, the system starts

learning, reason why CER keeps decreasing, getting to a value of approximatively 17, that has only small variances. So, for this dataset, a training of maximum 200 epochs would have been enough.

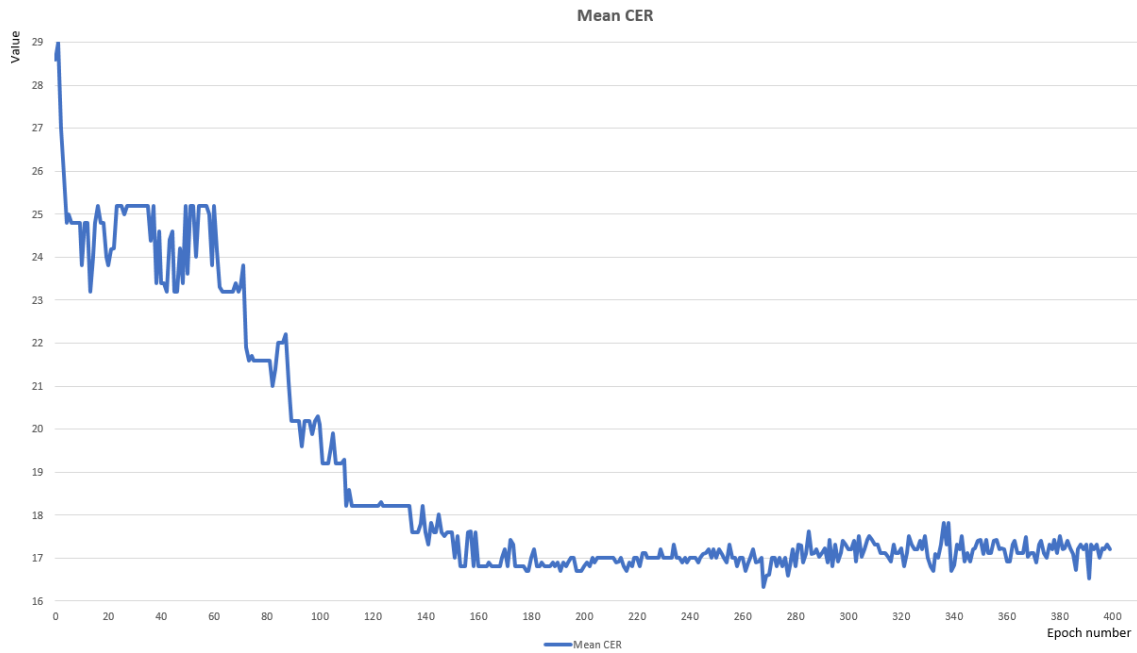


Figure 3.21. Representation of mean CER for random split protocol, Romanian language. Y axis: value of mean CER, X axis: Epoch number

The second performance metric that is analyzed is WER, whose mean is shown in Figure 3.22. On the horizontal axis, it is represented the number of epochs and on the vertical axis, it is plotted the value of the mean WER, getting results that vary between 3.5 and 4.7. However, the variation of this parameter does not depend on anything and it does not increase or decrease, even if the number of iterations that the system does it greater. It shows a behavior that cannot recognize a lot of words from a sentence, but it needs to be taken into account also the variable number of words from a sentence. Anyway, there are no great improvements of LipNet when computing the word error rate and the obtained results are not the desired ones.

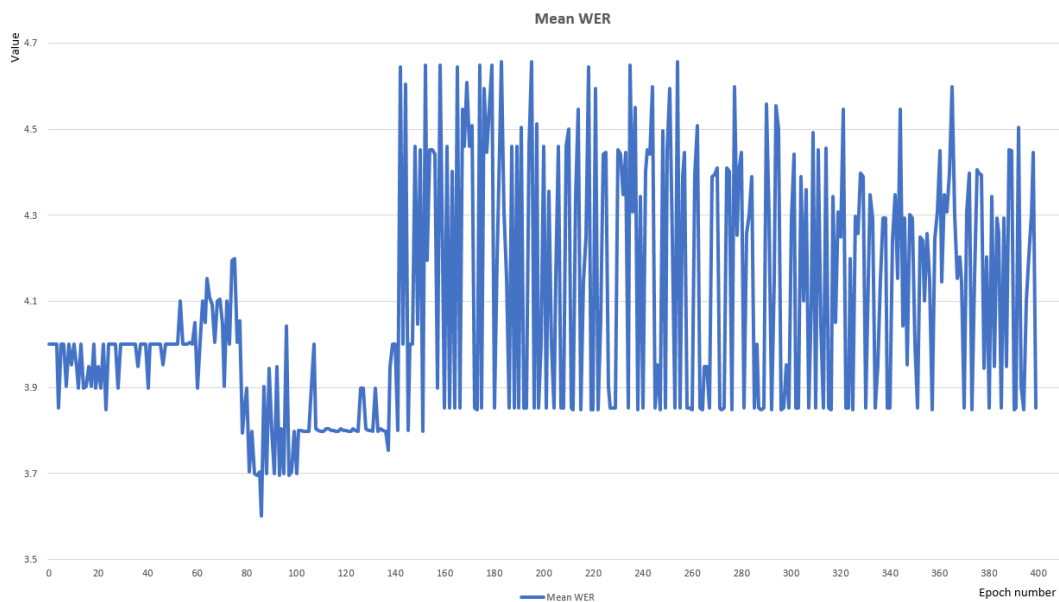


Figure 3.22. Representation of mean WER for random split protocol, Romanian language. Y axis: value of mean WER, X axis: Epoch number

BLEU is the third performance metric that shows how the system behaves when changed to Romanian language. The evolution of it is plotted in Figure 3.23, where on Y axis is represented the mean of BLEU and on the X axis it is represented the number of epochs, such that it get values between 0.19 and 0.49. Most of the translations should be done at the beginning of the training, but in this case they are done between the 60th and 80th epoch. In the rest of the cases, there are variances between 0.25 and 0.45 from iteration to iteration, reason why there is no pattern that can be detected. This is happening because of the different list of phonemes that is used for validation.

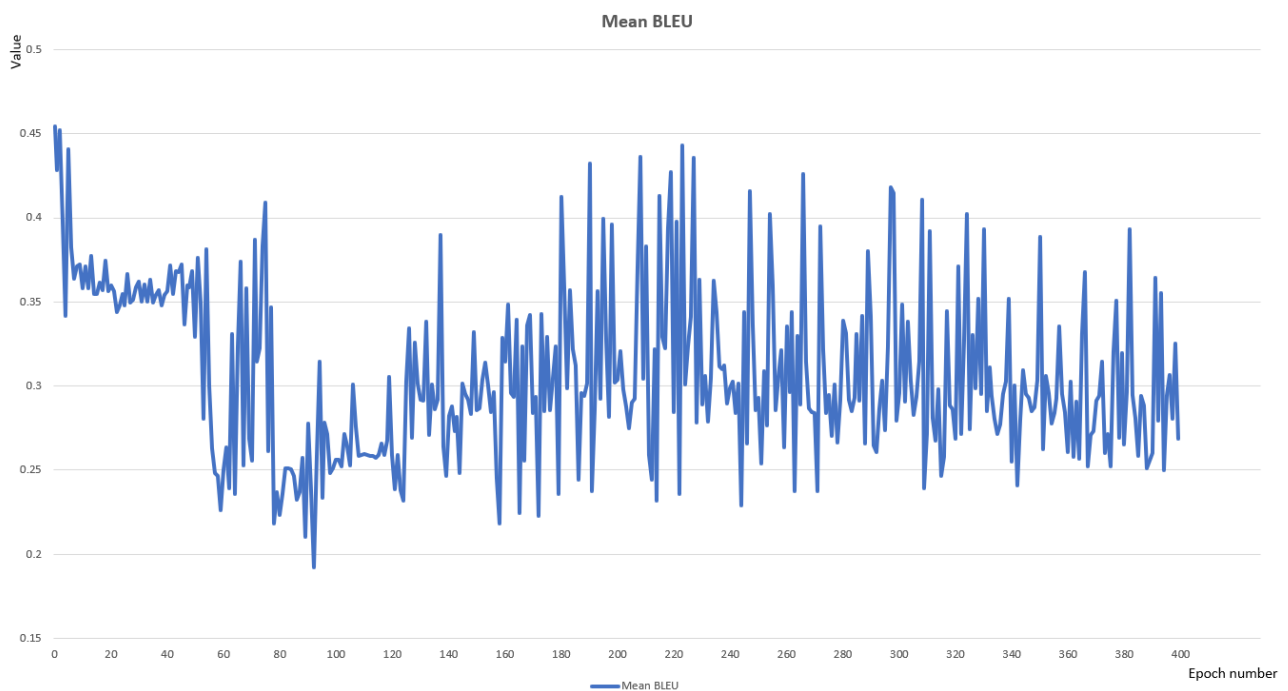


Figure 3.23. Representation of mean BLEU for random split protocol, Romanian language. Y axis: value of mean BLEU, X axis: Epoch number

When changing the system to another language, it is expected to get poorer performances. However, it should follow a pattern that begins with undesired values of the performances metrics, that improve after a certain number of iterations. The normalized values of mean CER, WER and BLEU, plotted in Figure 3.24, highlight an abnormal behavior, with an unusual learning path. On the vertical axis, it is represented the values of the normalized parameters and on the horizontal one the number of epochs, showing that only CER has a real improvement after a certain number of iterations. The other two parameters only vary between certain values, without obtaining any desirable performance.

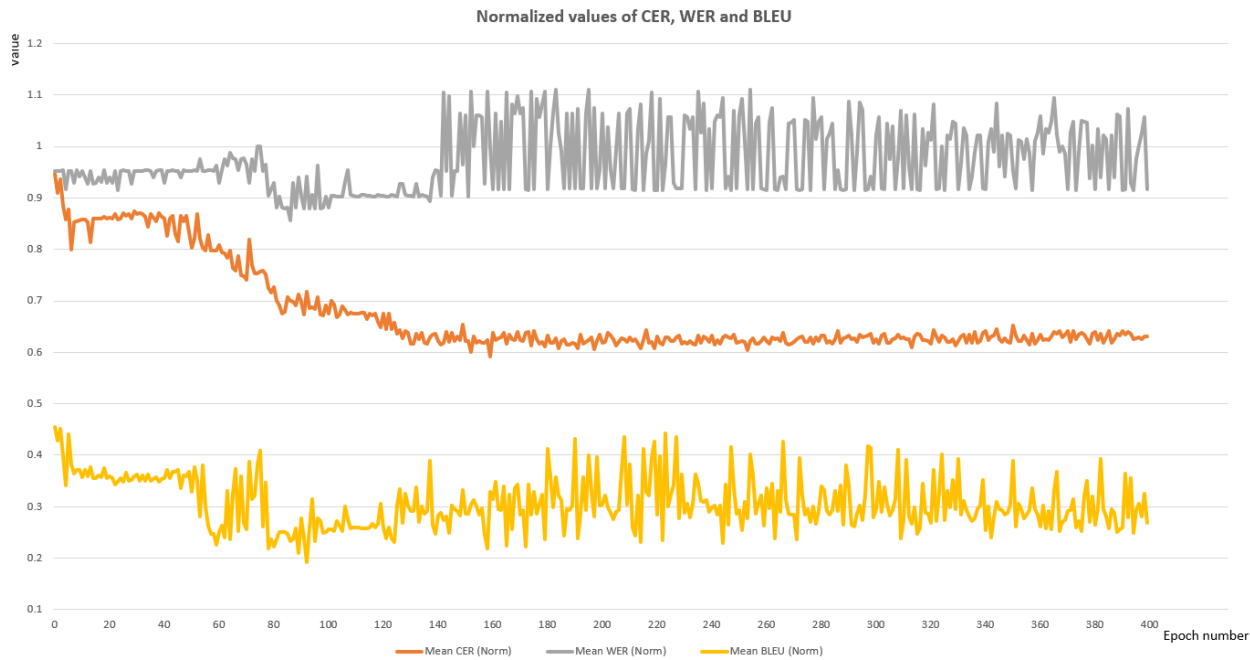


Figure 3.24. Representation of normalized values of CER, WER, BLEU for random split protocol, English language. Y axis: value of the normalized parameters, X axis: Epoch number

For validating, I used the same sentences that were chosen for the other two protocols, with a variable length of four or five words. The real sequences and the ones decoded by the system are plotted in Table 3.8, so it is easily to observe that it did not recognized any complete sentence. However, the number of incorrect words is 7, out of 21 and using this training protocol the system decoded longer words such as “prezinta”. From another point of view, the sequences are somehow randomly identified and they are not necessarily the same with the ones from the real sequences. For example, in the fourth case the system recognized two words from the dictionary, but only one of them is also part of the sentence said initially by the speaker.

Real sequence	Decoded sequence
anul acesta procesez imagini	anul acesta la este iaeita
electronica este pasiunea mea	eu este iaeita mea
eu prezint licenta anul acesta	eu prezinta anul iaeita
procesez imagini pentru licenta	eu rezina pentru iaeita
sunt student la electronica	sunt stuiet la iaeita

Table 3.8. Real and decoded sequences for random split protocol, Romanian language.

3.4.2 Small dataset

Before creating a dataset with 20 speakers, in order to observe if the system is adaptable for another language, a small corpus has been created. It included only 9 speakers, males and females and it was tested using unseen speakers protocol. The sentences used in this case had a variable length and the list of phonemes was the one that LipNet had initially, adapted for English language. In this case, only 200 iterations were done, aiming to observe the behavior of the system. Figure 3.25 shows the normalized values of mean CER, WER and BLEU, represented on Y axis, depending on the

number of epochs plotted on the Y axis. However, in this case the improvements are not so great, because the system has a small dataset. All values vary from epoch to epoch and the performances are really poor, but it is clearly that LipNet has the ability to be adapted for another language.

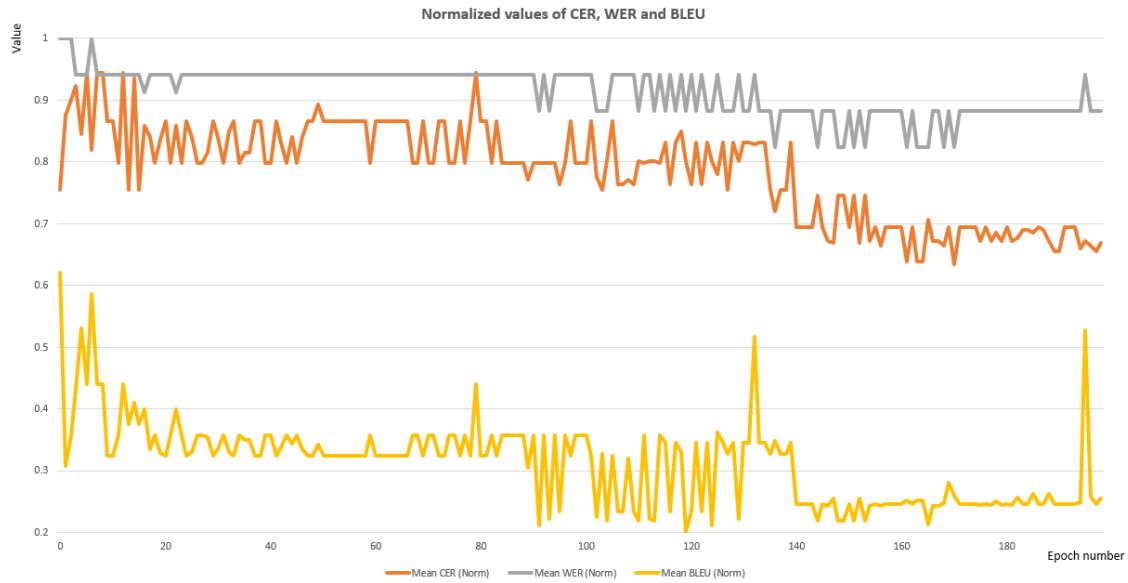


Figure 3.25. Representation of normalized values of CER, WER, BLEU for a small dataset, Romanian language. Y axis: value of the normalized parameters, X axis: Epoch number

Even if there are no great results of the training process when using such a small dataset, the protocol used one male speaker for validation, in order to observe if the system is able to recognize any word. Table 3.9 shows the differences between the real sequences and the decoded one, using 4 sentences. Color red highlights the words that are incorrectly decoded, but anyway, not even the correct words are not part of the sentence. In fact, the system learns some short words that use later for decoding, even if they are not in the desired output.

Real sequence	Decoded sequence
sunt student la electronica	eu reieu acesta
anul acesta procesez imagini	eu reieu acesta
electronica este pasiunea mea	eu reieu acesta
eu prezint licenta anul acesta	eu reieu acesta

Table 3.9. Real and decoded sequences for small dataset, Romanian language.

The small dataset highlights that capability of the system to learn another language, as long as different constraints are taken into account. These tests were not meant to highlight and good performance of the system, but to observe what changes are necessary to make it work for Romanian. Consequently, I observed the necessity to update the dictionary of the system and modify the list of phonemes, depending on the ones that any language has. However, these tests did not show the importance of a fix number of words in all sentences.

3.4.3 Dataset with variable fix number of words

The first two sets of tests have been using a variable length of the sentences, aspect which influenced, from my point of view the performance of the system. Also, they used the list of phonemes that English has, reason why some phonemes were not included, such as “ce” or “ci”. Thus, new tests

were taken, with a modified version of CAMPUS dataset. More exactly, the sentence that had five words have been deleted from all the speakers. On the other hand, the system's dictionary modified and the sentence with five words have been deleted. In addition, the list of phonemes was totally changed and it included only the ones that Romanian has..

For a better overview of LipNet's behavior, all three training protocols have been tried and analyzed, using the metrics described before. However, only random split's result will be presented, because it obtained the best values for CER, WER and BLEU. Furthermore, it decoded most of the words from the real sequences, but it still did not recognize a whole sentence. Figure 3.26 shows the normalized mean of CER, WER and BLEU, representing on the Y axis the values of the metrics and on the X axis the number of epochs. This time, the system did 400 iterations, to observe exactly, if there is any improvement or not. Therefore, it is easily to observe that the values of the performance metrics do not suffer great variances, even if the number of epochs increases. Anyway, from another perspective, this aspect shows a more stable system, whose values do not change a lot from one iteration to another.

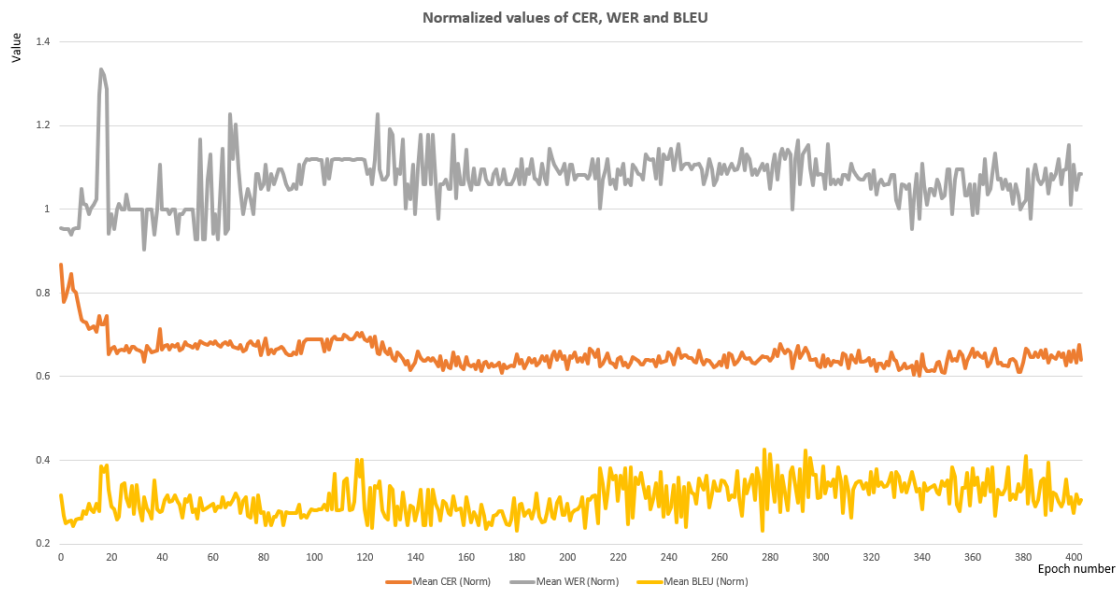


Figure 3.26. Representation of normalized values of CER, WER, BLEU for a fix number of sentences, Romanian. Y axis: value of the normalized parameters, X axis: Epoch number

For validation, a number of 5 sentences have been used, counting a total of 20 words. Table 3.10 shows that decoded sequences, highlights an improvement, caused by the only one word that has been identified and do not appear in the dictionary. Moreover, LipNet decoded a whole sentence and the other words recognized are part of the desired output. Another improvement is represented by the fact that the system decrypts also longer words, not only the ones made up by 2 or 4 letters.

Real sequence	Decoded sequence
procesez imagini pentru licenta	este acesta pentru licenta
anul acesta procesez imagini	anul acesta pentru licenta
procesez imagini pentru licenta	eucr imagini pentru acesta
procesez imagini pentru licenta	procesez la pentru licenta
electronica este pasiunea mea	Electronica la acesta pasiunea mea

Table 3.10. Real and decoded sequences for a fix number of sentences in the dataset, Romanian language.

3.4.4 Comparison between the obtained results

LipNet is a system that has been developed and tested for English, but the researchers stated it was adaptable for any other language. However, they did not mention any constraints that might exist and the way that the performances are influenced by any change. For Romanian, the created dataset did not follow the structure of GRID, reason why the words appeared in the training process for a different number of times. Moreover, the phonemes that the language has and the variable length of the sequences determined to get poorer results.

The first tests were done only for a small dataset, made up by 9 speakers. They aim to observe the capability of the system to learn another language and the possible challenges that might appear. Thus, I modified the dictionary and included only the sentences used for training, in order to let the system learn the correct words. Also, while doing these tests, I observed that the best results are obtained using a batch size equal to five, that would be kept for all other experiments. These tests were done only for a training protocol and the values of the metrics are not even close to the desired one, but when comparing the real and the decoded sentences, it can be observed words that are correctly recognized. So, the system is able to learn another language, but it must be optimized.

The other two sets of tests aimed to observe the performance on the system in different situations, having a variable and a fix length of sentences. Both performed trainings using the same three protocols, but the best results are obtained for random split. However, there is no clear correlation between the metrics, such that their evolution is somehow, independent. When the sentences have either four, or five words, the system is unstable and there are great variances of mean CER, WER and BLEU from one epoch to another, in opposition to the situation when the sequences have a fix number of words, and the metrics do not vary a lot. In addition, a variable dataset determines a random decoding of the words, in comparison to the fix dataset, that leads to a whole sentence decoding. Even more, in the second case, the system can identify longer sequences of characters, reason why longer words like “electronica” have been decoded.

In conclusion, all the performed experiments highlighted two important aspects. Firstly, the system can learn another language, but it needs some optimizations and a dataset with a clear structure. It must have a fix number of words, even if their length can be variable. Secondly, the system needs more iterations to learn a new language, such that the necessary time for training increases dramatically.

3.5 Conclusions of the experimental results

The experiments performed using LipNet included two languages, Romanian and English, and three training protocols: unseen speakers, unseen speakers curriculum and random split. The obtained results analyzed two aspects: performance metrics and differences between the real and the decoded sequences. While for English the system was tested before, for Romanian language there are no results that can be used for comparison, reason why, the experiments did not obtain a great performance.

English dataset had a very clear structure, including 6 different categories of words, so for any training each sequence of characters repeated for the same number of times. On the other hand, CAMPUS dataset had an unclear structure, with sentences that had four or five words. Also, the number of times that each sequence of characters appeared during a training was different, so the ability to recognize them was influenced as well. However, the technical aspects were taken into

account and both corpus had speakers males and females, who appeared in videos with a length of approximatively 3 seconds. Thus, the pre-processing part led to comparable results.

Experiments shown different dependencies and behaviors of the system, depending on the situation. English gets the best results when using random split algorithm, but the required time for training is greater, so in some situations this might not be suitable. The other two protocols showed performances that are comparable, even if unseen protocols fits better when the learning time is short, because it has the ability to get stable in approximatively 50 epochs. Anyway, for all three types of experiments, it has been easily to observe a correlation between the performance metrics. More exactly, at the beginning of the training, all metrics have values that are not desired, but there are a certain number of epochs when they change dramatically. CER and WER decrease almost at the same time, and this process is correlated with the increase of BLEU. After that, the metrics get stable and their variation does not lead to any improvement of the system. For all training protocols, the system is able to decode an entire sentence, but the number of wrongly detected words is higher for the unseen speaker protocol. It mixes different words between them, so the accuracy of it is lower. However, in all situations the necessary training should include a maximum of 150 epochs, a lot less than the number that developer advised. From that point, there is no improvement in the training process and the values of the performance metrics are almost the same.

On the other hand, LipNet was not tested before for another language, so no one presented any possible constraint of the system. So, part of the tests was done using English phonemes and part of them using the Romanian ones, reason why, the recognition of some visemes was more difficult. The first experiments aimed to observe the capability of the system and if it is able for real to return any Romanian word. After that, two sets of experiments have been done, having two different approaches. They returned different behaviors and performances, highlighting also some dependences that the system has. The first set of experiments that have been done used the English alphabet and a dataset with sequences with a variable length: four or five words. This aspect determined both undesired results of the metrics and unstable values, that had great variances from one epoch to another. Even more, in this situation LipNet did not have the ability to recognize longer words and their identification seemed more randomly than correlated to the desired output. The second tests of experiments have been performed using a dataset that had a fix number of words in the sequence, leading to some improvements. More exactly, the system decodes longer words and they are more correlated to the real sentences. Thus, LipNet has been able to recognize a whole sequence of four words and obtains better performances. Even if the correlation of CER, WER and BLEU that has been detected for the English experiments is still not available, a change appear in their variation, such that the system is more stable. The values of the metrics do not modify so much from one epoch to another and the results are closer to the desired ones.

When comparing the experiments that I did for English and for Romanian, it is obviously that the results are almost in opposition. This happened because of different reason, that included the dataset that I used. Therefore, the system needs a clear structure, that lets each word repeat, while training, for the same number of times. Also, the performances are influenced by the list of phonemes that the developer provides to the algorithms. If it does not include all the groups of letters from the language, the decoded words are not even close to the one from the desired output. Also, it is easily to observe that for English, LipNet learns faster than for Romanian, needing less epochs to become stable. The reason that influences this aspect is represented by the size of dataset, which is more than ten times smaller for Romanian and the structure of it, GRID being clearer. The performances obtained for the new language confirm that the system is adaptable, but it needs to be optimized in

order to get great results. The correlation between the metrics is not so obvious for Romanian, but the improvement can be detected when modifying the dataset. It leads to a conclusion that in order to get great results, there are some adjustments that must be done.

All in a nutshell, LipNet is a lip reading system, that can be used for different languages. It offers more training protocols, but the most efficient one is random split. For all experiments that I performed in Romanian and English, it decoded most of the sentences or words, getting less errors. However, it needs more time for training, but it, also, depends on the dataset that is used. Anyway, for another language that the one that was initially tested by the system, it needs at least a double number of epochs, in order to learn. In all situations, the worst results are obtained at the beginning of the training, but they improve for a certain number of epochs, that depends on the protocol. After that the performances of the system do not vary a lot and it does not learn anymore, reason why the training becomes useless.

Chapter 4. Conclusions and future perspectives of the system

Computer vision is described as an interdisciplinary field, that has different approaches, that aim to create automatic system that do activities performed by human beings. Lip reading is a technique that got a lot of attention, because of the wide range of applications that it has. It analyzes the movements of the lips, without taking into account the sounds of the videos, being able to decode the text said by a speaker. It is considered a complement of ASR, but the performances are not obtained that easily, because any system has a certain number of constraints that depend on the architecture of it and on the chosen language used for decoding.

Lip reading is a field that uses deep neural networks, so it involves, also, image processing and artificial intelligence. There are different approaches that have been tried before, but they did not obtain great performances because of the reduced datasets that have been used. Deep learning approach needs a large scale of labeled videos that are used for training. End-to-end systems are even more limited when taking into account past trials and, anyway, have been tested for English. Few other attempts were done for Mandarin [35] or Urdu [36], but they were not successful, because of the special phonemes that both languages have and their impossibility to decode them in a Latin alphabet.

LipNet is a system that has been developed firstly for English, but it can be adapted for any other language. It has a complex architecture, that took into account drawbacks of past systems and tried to solve them. Thus, it used specialized types of CNN, that analyzed both spatial and temporal dimensions, and special types of RNNs, that are bi-directional. Even more, CTC computes the loss in a more efficient way and it helps the system improve based on the past results. This system is developed on the idea that humans understand speech by both, hearing the voice, or viewing the movements of the lips, so it aim to read words on real time.

In order to observe the capabilities of the system, it was tested for both English and Romanian, trying to solve all challenges that appeared later on. While the dataset for English existed already, the one used for Romanian has been created, following the technical indications that GRID had. Anyway, it included only 20 speakers who said 5 sentences, in comparison to the English corpus, which had 34 speakers, who were filmed saying a total of 1000 sentences each. However, before creating a bigger dataset, a small one was used for few trials, in order to observe if the system is really able to learn another language. After that, experiments were performed using a corpus with more information, trying to analyze three training protocols out of five that were proposed by the developers. They had different approaches, using different types of validation, reason why the obtained performances vary a lot. The corpus had also alignments, that indicated the start and stop time of each word, such that the system was able to correlate the lips' movements and their meaning.

Therefore, in all situations, LipNet did the same steps, in order to get the output. Firstly, all videos were processed and they were transformed into frames that have been used later on for training. The images included only the lips of the speaker and each video had a fix number of 75 frames. After that, depending on the training protocol, the crops were processed and the system was trained, returning an output that represented a decoded text.

In order to measure the performances of the system in different situations and to observe any behavior of it, three metrics were used, that measured the character and word error rate and the

necessary number of translations that were necessary in order to obtain the desired output. Also, the system showed the real and the decoded sequence, to offer a better understating of the results or possible improvements. For both, Romanian and English, the worst values are obtained at the beginning of the training, but after that, a learning path can be observed, such that the metrics become better for a certain number of epochs. Later, the system gets stable and it does not learn anymore, so the training is useless. While for English, approximatively 50-80 epochs are enough to obtain the desired values of the metrics, the attempts that I did for Romanian needed more than 120 epochs to observe any improvement. In addition, metrics that show the performance of the experiments that I did using English dataset are correlated, such that CER and WER start decreasing and BLEU increases almost the same time. Trials that used CAMPUS dataset are not so clear, so there is no clear link between the variance of the metrics. However, when using the second version of the corpus, it is observable that the results are more stable, so CER, WER and BLEU do not modify so much from an epoch to another.

Taking into account the comparison between the results obtained using the English and the Romanian dataset, it is clear that the first improvement that should be done include the change of the corpus. More exactly, it must have a fix length of the sentences and a better structure, that let the system learn each word for an equal number of times. Furthermore, it should have a lot more sequences and, if possible, more speakers, both males and females. From another perspective, the list of phonemes that the system uses for Romanian should be updated and must include also all homophones, not only the groups of letters like “ce”, “ci”, etc. Also, the dictionary must be updated to the new sentences. On the other hand, LipNet needs some other optimizations when adapting it to another language. For example, the length of the strings should be updated, depending on the number of characters from the alphabet. Moreover, the batch size can also be changed, in correlation to the size of the dataset and the length of the sequences used for training.

Therefore, LipNet is an adaptable system, that can be trained for different languages. However, another set of experiments would be done, using a mixed dataset, made up by both English and Romanian sentences. In this case, the dictionary must include all sentences and the phonemes' list should have the ones that are available in each country. In this situation, there are some confusions that might appear, because of the different way of pronunciation some letters in each language. From another point of view, the behavior of the system or the performances that it can get, cannot be predicted, but it is expecting to need less epochs for learning.

On the other hand, right now, LipNet is a system that can be trained and tested, but it does not have a friendly interface that can be used by anyone. It is able to decode entire sentences, but it does not have a way of working that can be understood. It is trained and tested using Linux's terminal, so a GUI is absolutely necessary if it aims to become more popular. It must be also updated, depending on the purpose of the application and the requirements that it has. After the training is performed, only one model must be chased for further using, depending on the obtained performances. Even if the system has three training protocols, only one of them would be used, depending on the available time for training and the desired accuracy of the system.

All in a nutshell, lip reading techniques represent an area of growing attention. Even if for Romanian language the development of such a system might include some challenges, the proposed method aims to solve them and create a functional system that can be used for both research and society. LipNet is an alternative that can be adapted for any language, but it must be optimized and tested in more situations. After finishing the research part and obtaining a final model, an interface must be created, in order to offer the possibility to anyone to use such a system.

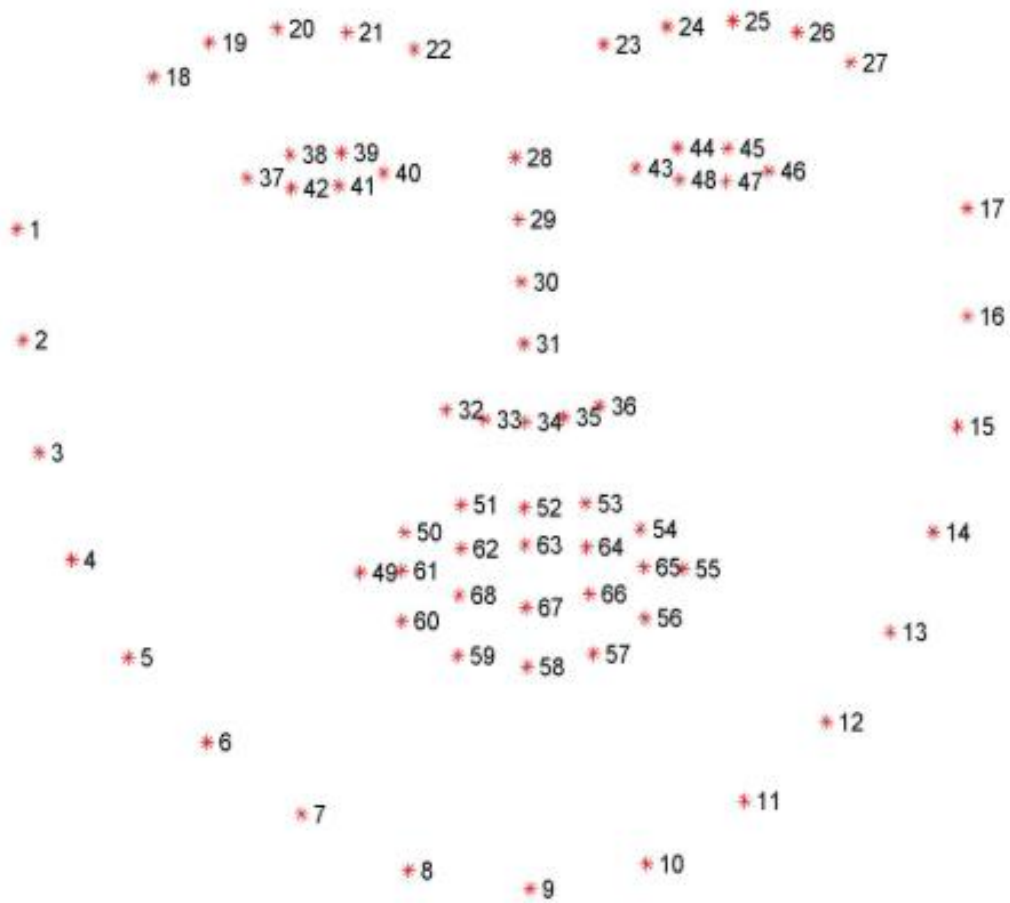
Bibliography

- [1] E. D. Petajan, "Automatic Lip reading to Enhance Speech Recognition (Speech Reading)" Ph.D. dissertation, University of Illinois at Urbana-Champaign, 1985
- [2] Ahmad B. A. Hassanat, "Visual Speech Recognition, Speech and Language Technologies", Prof. Ivo Ipsic (Ed), from: <http://www.intechopen.com/books/speech-and-language-technologies/visual-speech-recognition>, accessed on 10th of June 2018
- [3] K. Noda, Y. Yamaguchi, K. Nakadai, H. G. Okuno, and T. Ogata, "Lip reading using Convolutional Neural Network," in Proc. Interspeech, 2014, pp. 1149 – 1153.
- [4] H. McGurk, J. MacDonald, "Hearing Lips and Seeing Voices", Nature volume 264, no. 5588, pp. 746 – 748, 1976.
- [5] R.S. Michalski, J.G. Carbonell, T.M. Mitchell, "Machine Learning: An Artificial Intelligence Approach", Ed. Srpinge Verlag, 1984
- [6] Sebastian Thrun, Lorien Pratt, "Learning to learn", Ed. Srpinge Science+ Business Media, pp. 108-120, New York, 1998
- [7] Nasser M. Nasrabadi, "Pattern Recognition and Machine Learning," Journal of Electronic Imaging, 2007
- [8] Jürgen Schmidhuber, "Deep learning in neural networks: An overview", Journal "Neural Networks", Vol 61, pp 85-117, 2015
- [9] S. Rajaserkara, G.A. Vijayalakshmi Pai, "Neural networks, fuzzy systems and evolutionary algorithms: synthesis and applications", Ed. PHI Learning Private Limited, 2015
- [10] What is Deep learning?, <https://www.mathworks.com/discovery/deep-learning.html>, accessed on 11st of June, 2018
- [11] Kevin L. Priddy, Paul E. Keller, "Artificial Neural Networks: An Introduction", Ed. Spie Press, USA, 2005
- [12] Ali Zilouchian, Mo Jamshidi, "Intelligent Control Systems Using Soft Computing Methodologies", page 22, Ed. CRC-Press, 2001
- [13] Mijwel, Maad, "Artificial Neural Networks Advantages and Disadvantages", University of Baghdad, <https://www.researchgate.net> accessed on June 2018
- [14] Zeiler M.D., Fergus R., "Visualizing and Understanding Convolutional Networks". In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8689. Springer, Cham
- [15] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, Li Fei-Fei, "Large-scale Video Classification with Convolutional Neural Networks", <https://cs.stanford.edu/people/karpathy/deepvideo/>, accesed on 27th of June, 2018
- [16] Convolutional Neural Networks (CNNs / ConvNets), <http://cs231n.github.io/convolutional-networks/#conv> accesed on 16th of June, 2018
- [17] Mike Schuster, Kuldip K. Paliwal, Bidirectional Recurrent Neural Networks, IEEE Transactions on signal processing, Vol. 45, No. 11, November 1997
- [18] Yu Cheng, Duo Wang, Pan Zhou, Tao Zhang, A Survey of Model Compression and Acceleration for Deep Neural Networks, IEEE Signal processing magazine, Special issue on deep learning of image understanding, 2017

- [19] Recurrent Neural Networks, <http://www.cs.bham.ac.uk/~jxb/INC/I12.pdf> accessed on 16th of June, 2018
- [20] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, Shahrokh Valaee, Recent Advances in Recurrent Neural Networks, <https://arxiv.org/abs/1801.01078>, accessed on 27th of June, 2018
- [21] Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, Ed. MIT Press, 2016
- [22] Sepp Hochreiter, Jurgen Schmidhuber, "Long-short term memory", Neural Computation , <http://www.bioinf.jku.at/publications/older/2604.pdf>, accessed on 27th of June
- [23] Helen L. Bear, Richard Harvey, Decoding visemes: improving machine lip-reading, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2016
- [24] Leticia Ria Aran, Farrah Wong, Lim Pei Yi, "A review on methods and classifiers in lip reading", Automatic Control and Intelligent Systems, pp. 196-201, 2017.
- [25] G. Zhao, M. Barnard, M. Pietikainen, "Lipreading with local spatiotemporal descriptors", IEEE Transactions on Multimedia, Vol. 11, No. 7, 2009
- [26] M. Wand, J. Koutn, and J. Schmidhuber, "Lipreading with long short-term memory", IEEE ICASSP, pp. 6115–6119.2016
- [27] Stavros Petridis, Zuwei Li, Maja Pantic, End-To-End Visual Speech Recognition With LSTMs, <https://arxiv.org/abs/1701.05847>, accessed on 27th of June, 2018
- [28] Yannis M. Assael , Brendan Shillingford , Shimon Whiteson, Nando de Freitas, LipNet: End-to-End Sentence-level Lipreading, <https://arxiv.org/abs/1611.01599>, accessed on 27th of June, 2018
- [29] Saeed U., Dugelay JL. (2010) Combining Edge Detection and Region Segmentation for Lip Contour Extraction. In: Perales F.J., Fisher R.B. (eds) Articulated Motion and Deformable Objects. AMDO 2010. Lecture Notes in Computer Science, vol 6169. Springer, Berlin, Heidelberg
- [30] Keras implementation of 'LipNet: End-to-End Sentence-level Lipreading', <https://github.com/rizkiarm/LipNet>, accessed on 27th of June, 2018
- [31] Python- About, <https://www.python.org/about/>, accessed on 18th of June, 2018
- [32] The GRID audiovisual sentence corpus, <http://spandh.dcs.shef.ac.uk/gridcorpus/>, accsed on 27th of June
- [33] Horia Cucu, "Towards a speaker-independent, large-vocabulary continuous speech recognition system for Romanian", PhD Thesis, University "Politehnica" of Bucharest, 2011, <https://speed.pub.ro/people/horia-cucu/>, accessed on 27th of June, 2018
- [34] Kishore Papineni, Salim Roukos, Todd Ward, Wei-Jing Zhu, BLEU: a Method for Automatic Evaluation of Machine Translation, IBM T. J. Watson Research Center, <https://www.aclweb.org/anthology/P02-1040.pdf>, accessed on 27th of June, 2018
- [35] Aviv Gabbay, Asaph Shamir, Shmuel Peleg, Visual Speech Enhancement, <https://arxiv.org/abs/1711.08789>, accessed on 27th of June, 2018
- [36] M Faisal, Sanaullah Manzoor, Deep Learning for Lip Reading using Audio-Visual Information for Urdu Language, <https://arxiv.org/abs/1802.05521>, ccesed on 27th of June, 2018

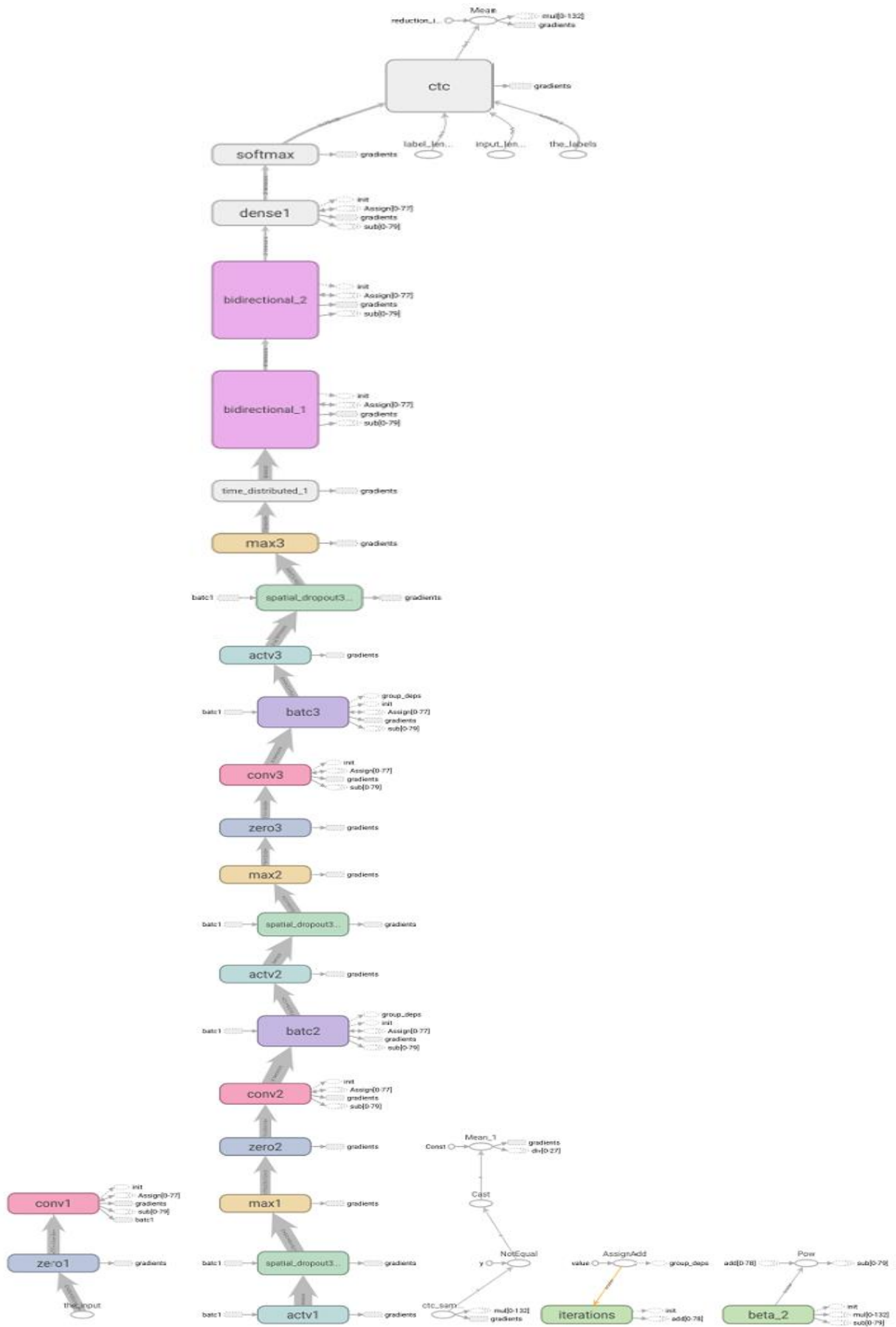
Annex 1

iBug face shape predictor using a landmark of 68 points



Annex 2

Detailed representation of the architecture of LipNet



Annex 3

Code used for random split protocol

```
from keras.optimizers import Adam
from keras.callbacks import TensorBoard, CSVLogger, ModelCheckpoint
from lipnet.lipreading.generators import RandomSplitGenerator
from lipnet.lipreading.callbacks import Statistics, Visualize
from lipnet.lipreading.curriculums import Curriculum
from lipnet.core.decoders import Decoder
from lipnet.lipreading.helpers import labels_to_text
from lipnet.utils.spell import Spell
from lipnet.model2 import LipNet
import numpy as np
import datetime
import os

np.random.seed(55)

CURRENT_PATH = os.path.dirname(os.path.abspath(__file__))
DATASET_DIR = os.path.join(CURRENT_PATH, 'datasets')
OUTPUT_DIR = os.path.join(CURRENT_PATH, 'results')
LOG_DIR = os.path.join(CURRENT_PATH, 'logs')

PREDICT_GREEDY = False
PREDICT_BEAM_WIDTH = 200
PREDICT_DICTIONARY = os.path.join(CURRENT_PATH, '..', '..', 'common', 'dictionaries', 'grid.txt')

def curriculum_rules(epoch):
    return { 'sentence_length': -1, 'flip_probability': 0.5,
            'jitter_probability': 0.05 }

def train(run_name, start_epoch, stop_epoch, img_c, img_w, img_h, frames_n,
          absolute_max_string_len, minibatch_size):
    curriculum = Curriculum(curriculum_rules)
    lip_gen = RandomSplitGenerator(dataset_path=DATASET_DIR,
                                   minibatch_size=minibatch_size,
                                   img_c=img_c, img_w=img_w, img_h=img_h,
                                   frames_n=frames_n,
                                   absolute_max_string_len=absolute_max_string_len,
                                   curriculum=curriculum,
                                   start_epoch=start_epoch).build(val_split=0.2)

    lipnet = LipNet(img_c=img_c, img_w=img_w, img_h=img_h, frames_n=frames_n,
                    absolute_max_string_len=absolute_max_string_len,
                    output_size=lip_gen.get_output_size())
    lipnet.summary()

    adam = Adam(lr=0.0001, beta_1=0.9, beta_2=0.999, epsilon=1e-08)

    # the loss calc occurs elsewhere, so use a dummy lambda func for the loss
    lipnet.model.compile(loss={'ctc': lambda y_true, y_pred: y_pred},
                        optimizer=adam)

    # load weight if necessary
    if start_epoch > 0:
```

```

        weight_file = os.path.join(OUTPUT_DIR, os.path.join(run_name,
'weights%02d.h5' % (start_epoch - 1)))
        lipnet.model.load_weights(weight_file)

    spell = Spell(path=PREDICT_DICTIONARY)
    decoder = Decoder(greedy=PREDICT_GREEDY, beam_width=PREDICT_BEAM_WIDTH,
        postprocessors=[labels_to_text, spell.sentence])

    # define callbacks
    statistics = Statistics(lipnet, lip_gen.next_val(), decoder, 256,
output_dir=os.path.join(OUTPUT_DIR, run_name))
    visualize = Visualize(os.path.join(OUTPUT_DIR, run_name), lipnet,
lip_gen.next_val(), decoder, num_display_sentences=minibatch_size)
    tensorboard = TensorBoard(log_dir=os.path.join(LOG_DIR, run_name))
    csv_logger = CSVLogger(os.path.join(LOG_DIR, "{}-
{}.csv".format('training', run_name)), separator=',', append=True)
    checkpoint = ModelCheckpoint(os.path.join(OUTPUT_DIR, run_name,
'weights{epoch:02d}.h5"), monitor='val_loss', save_weights_only=True,
mode='auto', period=1)

    lipnet.model.fit_generator(generator=lip_gen.next_train(),
        steps_per_epoch=lip_gen.default_training_steps,
epochs=stop_epoch,
        validation_data=lip_gen.next_val(),
validation_steps=lip_gen.default_validation_steps,
        callbacks=[checkpoint, statistics, visualize, lip_gen,
tensorboard, csv_logger],
        initial_epoch=start_epoch,
        verbose=1,
        max_q_size=5,
        workers=2,
        pickle_safe=True)

if __name__ == '__main__':
    run_name = datetime.datetime.now().strftime('%Y:%m:%d:%H:%M:%S')
    train(run_name, 0, 20, 3, 100, 50, 75, 32, 50)

```