“ALEXANDRU IOAN CUZA” UNIVERSITY, IAŞI

FACULTY OF COMPUTER SCIENCE

BACHELOR’S DEGREE THESIS

Automated Lip Reading for the Romanian Language

Proposed by:

Aenăşoaei Denis-Claudiu

Session: July, 2021

Scientific coordinator

Lect. dr. Benchea Răzvan

July 2021

“ALEXANDRU IOAN CUZA” UNIVERSITY, IAŞI

FACULTY OF COMPUTER SCIENCE

Automated Lip Reading for the Romanian Language

AENĂȘOAEI DENIS-CLAUDIU

Session: July, 2021

Avizat,

Îndrumător Lucrare de Licență

Titlul, Numele și prenumele
lect. dr. Benchea Mihai - Razvan

Data 22.06.2021 Semnătura

DECLARAȚIE privind originalitatea conținutului lucrării de licență

Subsemntatul(a) Aenășoaei Denis-Claudiu, domiciliul în Buruienești, com. Doljești, jud. Neamț, născut(ă) la data de 01.02.2000, identificat prin CNP 5000201046202, absolvent(a) al(a) Universității „Alexandru Ioan Cuza" din Iași, Facultatea de Informatică, specializarea Engleză, promoția 2021, declar pe propria răspundere, cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licență cu titlul: "Automated lip-reading in the Romanian Language", elaborată sub îndrumarea dl. Lect. Dr. Benchea Răzvan, pe care urmează să o susțină în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului său într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diploma sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Dată azi, 22.06.2021 Semnătură student……

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*Titlul complet al lucrării*”, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași, 22.06.2021

Absolvent *Aenășoaei Denis-Claudiu*

Glossary **6**

Introduction **7**
1.1 Context 7
1.2 Motivation 7
1.3 Personal contributions 8

Datasets for ALR **10**

Related work **11**
3.1 ALR without deep learning techniques 11
3.1.1 Goldschen et al.[3](1997) 11
3.1.2 Gergen et al.[4] (2016) 11
3.2 ALR with deep learning techniques 11
3.2.1 LipNet: End-to-end Sentence-Level Lipreading (2016) [5] 12
3.2.2 LRRo: A Lip Reading Data Set for the Under-resourced Romanian Language (2020) [7] 13
3.3 Conclusions 14

Technologies used **15**
4.1 OpenCV Library 15
4.2 H5py library 15
4.3 Tensorflow library with Keras 16
4.4 Google Colaboratory 16
4.5 Conclusions 17

Theoretical basis **18**
5.1 Neural Networks [9] [10] 18
5.1.1 The gradient descent algorithm 19
5.1.2 Convolutional Neural Networks (CNNs) 20
5.1.3 Recurrent Neural Networks (RNNs) 21
5.2 Conclusions 23

Speech recognition based on lip movement **24**

Implementing a neural network capable of reading romanian words **27**
7.1 Architecture 27
7.2 The datasets [7] 28
7.3 Training 30
7.4 Performance evaluation and results 33

Conclusions **37**

References **39**

Glossary

AI - Artificial Intelligence

ML - Machine Learning

ALR - Automated Lip Reading

ASR - Automated Speech Recognition

VSR - Visual Speech Recognition

NN - Neural Network

ANN - Artificial Neural Network

CNN - Convolutional Neural Network

RNN - Recurrent Neural Network

LSTM - Long Short Term Memory

GRU - Gated Recurrent Unit

BI-GRU/LSTM - Bidirectional GRU/LSTM

HMM - Hidden Markov Model

RGB - Red Green Blue (color scheme)

Lab (environment) - a controlled medium where noise and inconsistency levels are low

Wild (environment) - an unpredictable medium where noise and inconsistency levels are high

1. Introduction

1.1 Context

Artificial neural networks have seen tremendous improvements both in techniques and in applicabilities, given the advancements in computing powers and algorithms. Given this context, they have become more and more usable for problems that, since now, have been thought to be too complicated or not suited for this approach. In this project, we will address the topic of lip-reading as a whole, some implementations already developed and a new implementation that takes some progress made in the english language and tries to raise the bar to the same level in another language, namely the Romanian Language.

One of the newer applications that were found for neutral networks and deep learning in the field of automated lip reading (ALR). Lip reading is understanding the speech by interpreting (only the) lip movement. This task is considered a notoriously hard task even for humans, especially when no context is provided.

Although considered an useful skill only to those with disabilities and we, inherently, are not good at reading lips, humans use some level of lip reading in almost every day of our life, helping us understand messages that we may have not understood otherwise. As an example, if communication is happening through a noisy channel (i.e. on the street, or when loud music is playing), we may understand only a portion of what another person is saying by listening to the sound and we will deduct the rest by analyzing their lip movement, unconsciously.

The presented work will walk the reader through the following elements: introduction in automated lip reading, with common datasets used, examples of work already done with their respective results, a list of already-built technologies that are used, along with the theoretical basis supporting them and what an implementation of a system capable of this task can look like, along with the results of a presented model.

1.2 Motivation

An average hearing impaierd person is able to achieve an accuracy of around 17±12% for a limited subset of 30 monosyllabic words, and 21±11% for 30 compound words [1]. It has been concluded that a human lip reader achieves better results for words that are larger, compared to

monosyllabic or disyllabic ones. ALR systems have shown that the same concept applies to machines, as they achieve far better results on long words. Given this context, it is important to have an automated system capable of surpassing the capabilities of a person.

Applications of such models are many, including helping hearing impaired people to understand vocal communication, helping mute people express themselves, communicating through noisy environments and even vocal commands.

Technology has advanced a lot, but we have to consider the people that are not as lucky and are not able to use it because of a disability. For example, a person who has a hearing disability cannot comprehend the actions happening on a screen and a person that cannot talk will never be able to use a personal assistant, such as Alexa from Amazon or Google Assistant. Much of the current pieces of technology relies on the fact that the user is physically able to interact with it. There is little to no alternative for people without such abilities. If we were to have an accurate way for automatic speech recognition, there could be automated subtitles for any piece of audio-video content existing, without any additional effort, thus helping the growing population that has some kind of hearing-impairment, estimated around 5% of the world's population in 2021 [2].

The presented subject has been chosen because I was fascinated by how a computer, with only the progress done in a few decades, could learn things that are considered too hard for many of us and for the implications this subject could have in the life of many.

1.3 Personal contributions

The idea of ALR is not a new one, being researched for a long time now with various techniques and results, but if we look into the Romanian language, or any other language aside of English, the research is limited to none.

Personal contributions brought to the project include:

1. The study of different automated lip-reading techniques.
2. Proposing a new neural network model capable of taking video frames and deciding what Romanian word is pronounced in them, being one of the first ones doing such a thing
3. Implementing an algorithm capable of taking normal RGB video feed and transform it into a grayscale cut of the mouth, which could be used to have further data fed into the network.
4. Modifying the videos found in a database so that they are more fit to the task at hand.
5. Performance comparison of the presented model with others already developed.

2. Datasets for ALR

By far the most abundant data for ALR is in the English language, being the language in which ALR has been pioneered and where the first steps towards Visual Speech Recognition have been made. They engulf the largest vocabulary size and the largest amount of video hours. The overwhelming majority of research done on the topic of lip-reading is done in this language, also having the most promising results thus far.

In the past few years, there have been some datasets published in other languages, but they are comparatively smaller and less complex. Most notably, there are datasets for the mandarin, urdu and the newly introduced LRRo, for the Romanian language. A comprehensive list of the most popular public datasets, alongside some details about them, can be found in Table 1. Most of the videos used for such datasets are shot in a controlled, laboratory environment, which will lead to better results due to consistency, but limits the applicabilities of methods trained on such data.

The method presented in this document is applied on Wild LRRo dataset and Lab LRRo dataset.

Name	Language	# of speakers	# of words pronounced	Vocabulary size	Number of hours	Year of release
AVICAR	English	100	1300	10	-	2004
OuluVS2	English	52	9100	10	-	2015
GRID	English	34	165000	51	28	2006
LRS2	English	-	~4.6 M	~27800	4900	2017
LRS3	English	~ 6000	-	~64100	475	2018
LRW-1000	Mandarin	>2000	718000	1000	57	2018
Urdu	Urdu	10	10	-	-	2018
Wild LRRo	Romanian	>35	1100	21	21	2019
Lab LRRo	Romanian	19	6400	48	5	2020

Table 1: Overview of the existing ALR Datasets

3. Related work

Automated Lip Reading has been a topic of interest for an extended period of time in Computer Science, with the most early work being done without the help of any neural networks or deep learning algorithms. Great results have been achieved without the use of neural networks, but the latest work seems to surpass the early handcrafted methods.

3.1 ALR without deep learning techniques

It involves either extensive preprocessing of the input frames to extract features from a video, or handcrafted vision methods. This method requires a human professionist in the domain of both linguistics and lip reading in order to perform well, making it extremely hard to scale.

3.1.1 Goldschen et al.[3](1997)

First instance of visual-only lip-reading on a sentence level, using Hidden Markov Models (HMMs), using hand segmented phones. It has achieved an accuracy of 25% on a sentence level prediction on a small dataset.

3.1.2 Gergen et al.[4] (2016)

Built around the same techniques, the team used speaker-dependent linear discriminant analysis transforms of the Discrete Cosine Transforms of the mouth regions in a Hidden Markov Model/Generalized Method of Moments. This has previously held the highest accuracy on the GRID Corpus dataset, with a speaker-dependent accuracy of 86.4%, before RNN methods such as the ones presented in LipNet have surpassed it.

3.2 ALR with deep learning techniques

Most of the work using Machine Learning, including the method presented in this document, is done on a letter or word level, classifying the words or letters that are present in speech, with the exception of more recent research, which tries to take a sentence-level approach, eliminating the need of segmenting a video into words spoken, which didn't always work as intended. Architectures designed for these tasks usually include some kind of recurrency, be it in the form

of Long Short Term Memory (LSTM) or Gated Recurrent Unit (GRU), used to link the mouth movement between multiple frames, resulting in better predictions.

3.2.1 LipNet: End-to-end Sentence-Level Lipreading (2016) [5]

Proposed in 2016 by a group of researchers from University of Oxford, United Kingdom, the work done by Yannis M. Assael, Brendan Shillingford, Shimon Whiteson and Nando de Freitas has introduced a new way of devising a lip-reading Neural Network.

They have developed a Neural Network architecture capable of correctly predicting the speech present in a video without sound in up to 95.2% of videos present in the GRID Corpus dataset. The architecture, presented in Figure 1, consists of an input of t images, corresponding to the analyzed frames of the video, a Spatio-Temporal Convolutional Neural Network each with Spatial Pooling, two layers of Bidirectional Gated Recurrent Unit, the output of which is fed into a number of Dense (fully connected) layers. The loss function used to train the model by LipNet is Connectionist Temporal Classification Loss, which helps to have a view of each time-step (frame) in regards to the final output.

The accuracy is measured in two ways, Word Error Rate (WER) and Character Error Rate (CER). WER (or CER) is defined as the minimum number of word (or character) insertions, substitutions, and deletions required to transform the prediction into the ground truth, divided by the number of words (or characters) in the ground truth. It has been state-of-the-art for lip reading on the GRID Corpus, achieving accuracy of up to 95.2% (4.8% WER). This architecture has been the baseline for designing the model presented in this document. There have been improvements brought upon the LipNet model (LCANet) since the paper got released, achieving up to 2.9% WER [6].

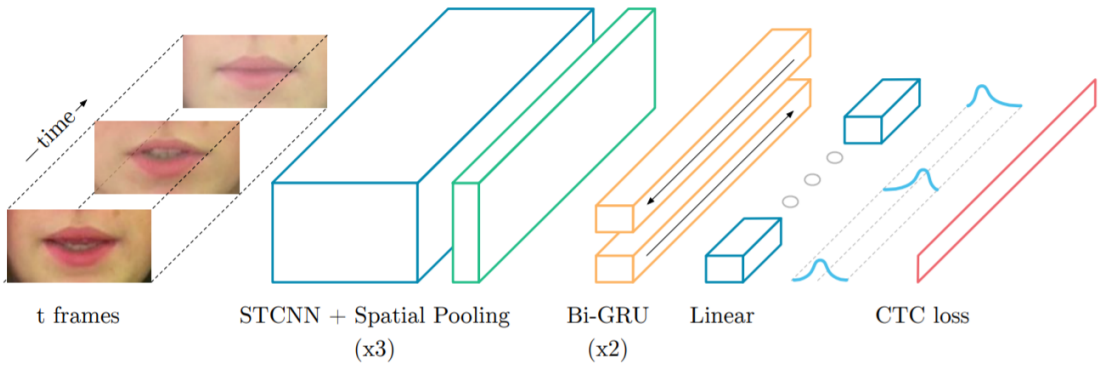

Figure 1: LipNet Architecture

3.2.2 LRRo: A Lip Reading Data Set for the Under-resourced Romanian Language (2020) [7]

Published in June 2020 by Andrei Cosmin Jitaru, Şeila Abdulamit and Bogdan Ionescu from University of Bucharest, it is an attempt to bring less popular languages on par with research in lip-reading on the english language. They have published two public datasets, aiming to aid the development of lip-reading methods for the romanian language. With the published dataset, they have proposed two baseline models in VGG-M via the MT Architecture and Inception-V4 network, both capable of making moderately accurate predictions. Results of the proposed models are presented in Table 2, where "Top-1" means that the model has predicted the correct word and "Top-5" means that the network has put the correct word in his top five predictions for the input.

The main problem with the baseline algorithms is that they are not custom made, therefore not perfectly fit for the problem at hand. Thus, implementing an algorithm that is made for this problem has a high chance of achieving better results.

Overall, this research is concentrated on providing a base for further development in lower interest languages, therefore with fewer resources, in the VSR world, rather than providing an efficient model for the problem.

The newly proposed algorithm will be tested on these datasets, with further explanation about its structure coming in a later section.

Data set structure \ Accuracy		MT				Inception-V4			
		Test		Train-val		Test		Train-val	
		Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
Wild LRRo data set	21 classes	33%	61%	37%	68%	33%	62%	40%	64%
Lab LRRo data set	16 classes	76%	97%	80%	97%	75%	97%	80%	98%
	32 classes	81%	95%	82%	95%	77%	96%	81%	96%
	48 classes	71%	90%	71%	91%	71%	92%	71%	93%

Table 2: Performance of the baseline systems trained on LRRo data sets

3.3 Conclusions

There has been extensive research in the field of ALR in the past decades, understandably so, given that it is a topic of interest that has many real life applications in the domains of medicine and communication. Even if most of the current work is done for the english language, being the most studied by far, it does not mean that there is no progress for other languages. Given that only a relatively small portion of the population speaks english (~20%) [8], there is a need to balance the research in such a way that other languages are taken into consideration.

4. Technologies used

4.1 OpenCV Library

OpenCV, or Open Sourced Computer Vision is an open source (i.e. the source code can be freely used and modified by anyone) library that was first written for the C++ programming language, but later on got also released on Java and Python. Its main focus is, as the name suggests, computer vision, having a large number of algorithms that makes it easy to work with images and videos, from simple Input/Output, to complex object detection.

Its structure is mostly modular; the main modules that are used by this project are:

- **core**: a compact module defining basic data structures and basic functions used by all other modules.
- **imgproc**: an image processing module that includes geometrical image transformations (resize, affine and perspective warping), color space conversion, histograms, and so on.
- **objdetect**: detection of objects and instances of the predefined classes (for example, faces, cars, **mouths**)
- **videoio:** an interface to video capturing and video codecs.

The library was used for reading and preprocessing the input video, detecting the mouths of the speaker, cropping it from the video and transforming the color scheme of a video from RGB to Grayscale, where each pixel from an image is represented in a spectrum with a numbers from 0 to 1, 0 representing black, and 1 representing the color white.

4.2 H5py library

Is a library made for efficiently storing large amounts of data in a compressed, binary format. H5py gives an interface to store huge amounts of numerical data in a relatively small file size.

Among the advantages of using the library there are:

- Efficiently save, store and load large amounts of data
- Easy to use interface

They were used by the program in order to store the network's weights after training so they can be loaded again without the need of re-training the model.

4.3 Tensorflow library with Keras API

Tensorflow is an open source library developed by Google and released in 2015, containing many algorithms for numerical calculus and machine learning. It offers easy to use implementations on a wide range of machine learning algorithms, including neural networks, Naive Bayes or K-Nearest Neighbors.

One of the key advantages the library has is that it is easy to use, even for someone without experience on any machine learning techniques. Creating a neural network for solving a complex problem, such as an object classifier or speech synthesis, can be achieved swiftly. Even if easy to use, performance is not sacrificed, in terms of both speed and scalability, being one of the fastest libraries for machine learning in python.

It is possible to run the library on a wide range of devices, including most operating systems and even mobile devices. It can run both on the CPU or the GPU of a device, but running it on a modern GPU usually yields significantly increased performance compared to a CPU.

There are several high level APIs built to run on top of the tensorflow platform, but Keras will be used in this project, for its popularity among researchers and the ability to further simplify the work without any considerable drawback.

The library will be used in order to create a model, train it on the video data, and make predictions about data that he has never seen before.

4.4 Google Colaboratory

Colaboratory, or “Colab” for short, is a product developed by Google Research. Colab will allow anybody to write and execute any python code within the browser, and is especially well suited to machine learning and data analysis. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use. It has many features and provides free access to resources such as powerful GPUs and even tensor processing units (TPUs). A TPU is an AI accelerator developed by Google, which works best on their ML platform, Tensorflow. They are not available for sale to the public market yet, but there are plans to be in the future. Even if the TPUs are handcrafted to work with Tensorflow, the ones made available through the Colab platform are still thought to have slightly lower performance than the GPUs that can be used.

Colab has its limitations and will not let an user use it for an extended period of time. It will disconnect a user from their runtime if they are idle for more than 90 minutes and has a hard limitation, even with interaction, of 12 hours. Resource allocation is based on the time spent on a connection; being on a GPU connection will expend the “resource budget” faster.These

limitations are put in place so that a limited number of users or parties will not have a monopoly on the resources provided by the platform.

The platform will be used to gain access to high computing power through a GPU, which is necessary when we want to train a large neural network architecture. As a comparison, a training on a CPU would have taken 55 minutes per epoch, and if the same is done with a GPU provided by Colab, the same training will take only about 50 seconds, netting a 60 times faster training speed.

4.5 Conclusions

Given the platform provided by Google and the tools provided through the libraries, development and training of a NN has become as easy as it could be. There are numerous libraries designed to aid with pre-processing of data, devise a NN architecture, train it and verify the result, but the combination of OpenCV for image/video processing and Tensorflow with Keras for an action model is the most widely used by researchers, for their easy-to-use interfaces and high computation speeds.

5. Theoretical basis

5.1 Neural Networks [9] [10]

Popularly known as Universal Function Approximators, Neural networks are artificial programs trying to reflect the behavior of the human brain, allowing computer programs to recognize patterns and solve common problems in the fields of AI, machine learning, and deep learning. Also known as Artificial Neural Networks (ANN) or Simulated Neural Networks (SNN), they are a subset in the field of machine learning and one of the building blocks of deep learning. Each node, or artificial neuron, connects to another and has an associated weight and a bias. Each node of a network, aside of the ones from the input layer, can be seen as a logistic regression on the neurons from the previous layer. Figure 2 shows the basic architecture of an ANN.

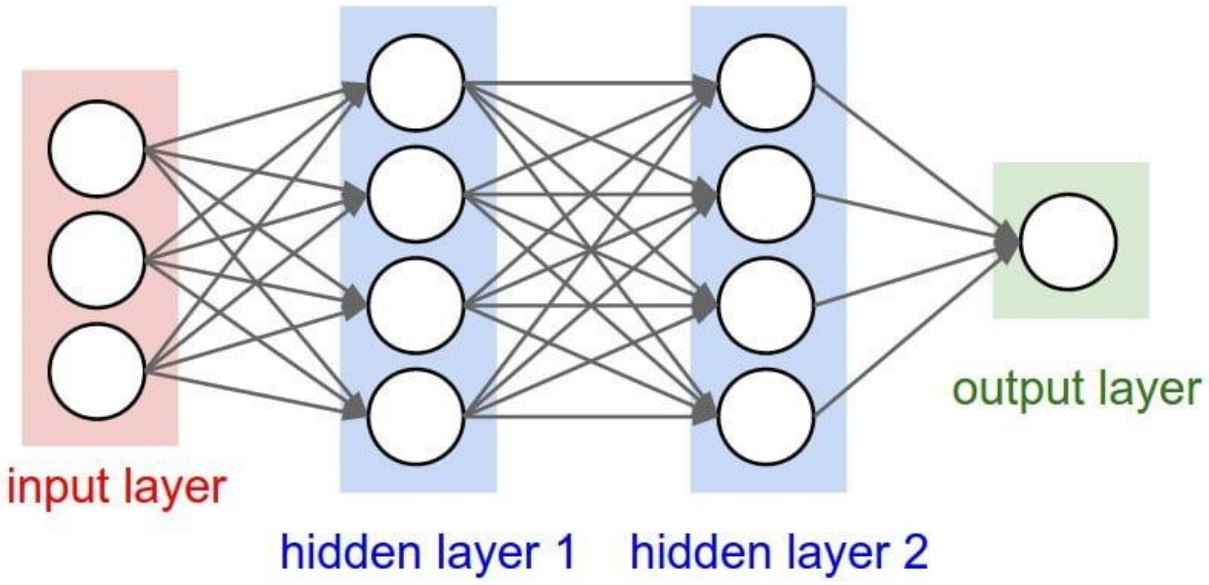

Figure 2: Basic Artificial Neural Network

Since it is known that the value of a given neuron is a logistic regression of previous neurons, we can define it to be:

$$\sum_{i=0}^{n} w_{ij} \cdot x_i + b_i$$

Where w_{ij} is the weight of the i^{th} neuron from the previous layer in the output of the j^{th} neuron from the computed layer and b_i is the bias associated to the i^{th} neuron in the previous layer. Usually, there is an activation function applied to the result, which will be noted with z. The most commonly used activation functions are:

- Step function:

$$f(z) = \begin{cases} 1, \ if \ z \geq \theta \\ 0, \ if \ z < \theta \end{cases}, \ where \ \theta \ is \ an \ arbitrary \ threshold$$

- Sigmoid function:

$$f(z) = \frac{1}{1 + e^{-z}}$$

- Rectified Linear Unit (ReLU)

$$f(z) = max(0, \ z)$$

- Softmax function, predominantly applied to the output layer:

$$f(z) = \frac{e^z}{\sum_{i=0}^{n} e^{z_i}}$$

These networks are trained on input data using the gradient descent algorithm, with its variations (Adam, RMSprop, stochastic gradient descent, etc.).

5.1.1 The gradient descent algorithm

Is an optimization algorithm, used for minimizing a given function by gradually moving in the direction of the steepest, which is defined by the negative of the gradient. The algorithm is used to update the parameters of a linear regression, or more generally, weights and biases of a neural network. Each iteration, a cost is determined by comparing the outputs of a network with the desired output, which will be used to update the network's parameters. The size of the steps taken towards the desired goal is called *learning rate*. A high learning rate means larger steps taken, and theoretically a faster convergence, but we run into the risk of missing the optimum by stepping over it. By contrast, a small learning rate will guarantee this overshoot will not happen, but comes at the cost of slow convergence or even no convergence at all, given that calculating the gradients is a time consuming task.

The cost function is used to tell us how well our model is doing in decision making, giving its parameters. It has its own curve and its own gradients. The slope of this curve tells us how to

update the model so it will be more accurate. The weights of each layer from a network will be updated proportionally to the derivative of the neuron activation function and the weights/biases. The result is multiplied by the learning rate, which usually is a constant between 10^{-2} and 10^{-6}.

There are three main types of gradient descent algorithm:

- Batch Gradient Descent (BGD), which will do the weights update after each epoch (i.e. after each entry in the training data has been used). It has multiple drawbacks, with the most notable being the extremely slow training speed and the memory needed to load all data on it.
- Stochastic Gradient Descent (SGD), which will do the weights update after each entry in the dataset. It performs better in terms of both speed and convergence compared to BGD, but comes with the drawbacks of high variance and the fact that data found sooner in the dataset will dictate a higher change in the weights compared to the ones found at the end.
- Mini-batch Gradient Descent, which takes the best of both worlds by performing an update after each batch training. A batch is a subset of the dataset of arbitrary size that is used for training at a given time. In this way, we reach a more stable convergence, with less variance and with greater speed.

There have been improvements brought upon the original gradient descent algorithm, most notably in the Adaptive Moment Estimation (Adam) and RMSprop and Adadelta algorithms, which deal with the most common problems that classic gradient descent algorithms are facing.

5.1.2 Convolutional Neural Networks (CNNs)

CNNs are used in order to solve problems that were considered too hard for normal NN, especially if the input comes in the form of images or videos. It has been concluded that, in order to have a model capable of solving complicated problems, such as sentiment analysis, image classification or understanding natural language(s), traditional ML methods will need some level of feature extraction. Feature extractions helps the model to make broader generalisation, adding layers of abstractization and reduces the dimensionality of a problem. The feature extraction is usually done through filters that make it so only the essential part of an image or sequence is taken into consideration. The great thing about CNNs is that the filters are not provided manually, they are learned at the same time as other parameters in the model. Figure 3 is a representation of an original image, and the features extracted by applying a filter to it.

Figure 3: Before and after of a convolution feature extraction

An usual architecture of a CNN will include the following parts:

- The convolutional layer(s), where the feature extraction happens;
- The pooling layer(s), where the optimal filter is chosen as the one being used;
- The fully connected layer(s), which are the ones that take the actual decision based on the input data and features extracted;
- The output layer, or the one where we see what decision has been taken based on the input.

Even if they are most suited for input data that involves images, they can be successfully used for other types of inputs, such as sequential data.

5.1.3 Recurrent Neural Networks (RNNs)

Traditional deep neural networks assume an independence between the inputs and the outputs, which is not true when data has a natural succession. RNNs provide a recurrent connection on a hidden state, which ensures sequenced information (i.e. information from the beginning of the sequence is related with information found later on in the sequence) will be captured and the bonds between them will be used. RNNs are usually used for problems involving, but not limited to, time series data, text data, audio data and **video** data. Figure 4 shows a visual comparison between RNNs and a normal feed forward architecture.

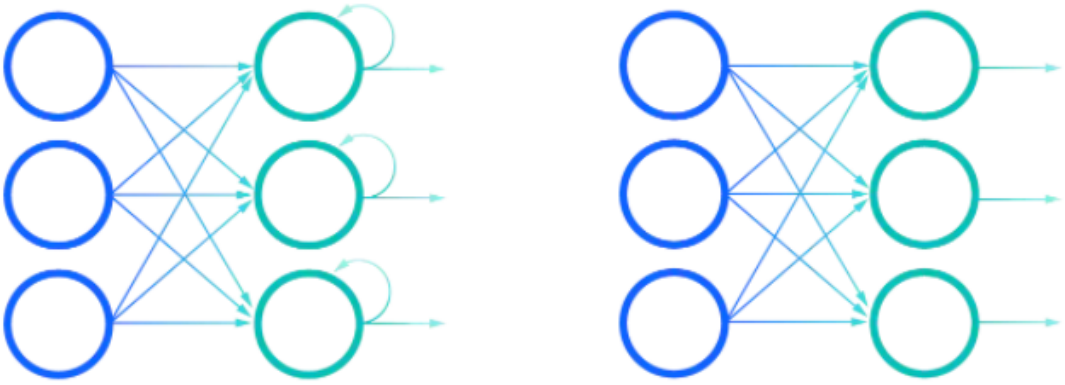

Figure 4: (Left) Representation of a RNN architecture, (Right) Normal Feedforward architecture

The recurrent states could be seen as normal neurons, that on top of input coming from the previous layer, it has an additional input coming from the previous neuron in the sequence. Each node from a RNN will share the same weights, in contrast to ANNs, where each node will have different weights. The usually elected activation function chosen for such layers is tanh:

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

RNNs leverage backpropagation through time (BPTT), which is different from normal backpropagation through summing errors from each time step, in contrast to normal backpropagation which does not sum errors and does not have shared parameters.

In the process, RNNs tend to run into two problems:

- Exploding gradient, which happens when a large gradient will be continuously multiplied by a value greater than one, resulting in large updates of weights, resulting in an unstable network. In the most extreme cases this can lead to overflows.
- Vanishing gradient, which happens when a small gradient will result in the network's parameters updating with small values, or in extreme cases, not be updated at all. It will slow down or even cease the learning process.

These problems are addressed in special cases of RNNs, such as Long Short Term Memory (LSTM) [11] and Gated Recurrent Units (GRUs).

LSTM had been introduced as an attempt to address the vanishing gradient problem and long-term dependencies (i.e. if the previous state that influences the current state is not in the recent past). In order to achieve this, LSTMs have cells, with three gates each, an input gate, an

output gate and a forget gate. These gates help the architecture by controlling the information flow coming and leaving from a cell.

GRUs have been introduced as a variant of LSTM, which instead of three gates, has only two: a reset gate and an update gate, used for the same purpose, to control the information flow. This will usually alleviate the computation power needed for such layers, but have slightly worse accuracy on average.

Normal RNNs assume that the data prior to a node is dependent on the data before that node, but there are cases, as it is with ALR, where the later elements of a sequence will affect the ones that come first. For such cases, Bidirectional RNNs are commonly used, which will make future data have an influence on the current state.

5.2 Conclusions

Although not without drawbacks, neural networks have evolved from the time they were first used, making them more viable for an increasing amount of problems in the real world. Seen first as being useful just as a classifier, they evolved over time to be the most applicable machine learning algorithms. The main negatives of such architectures are the need of large amounts of (annotated) training data, slow training speed and a need of high computation power.

There are many special cases of NNs, with the ones present here being the ones used in this project. A combination of a CNN, a RNN and an ANN are used for predicting speech based on lip movement in the presented model.

6. Speech recognition based on lip movement

The task of visual speech recognition usually involves four main steps, represented in figure 5. The existing lip-reading systems emphasize on face detection, lip localization, followed by feature extraction and speech recognition. After identifying the speaker's face, the lip region has to be found and then information has to be analyzed by the automatic lip reader.

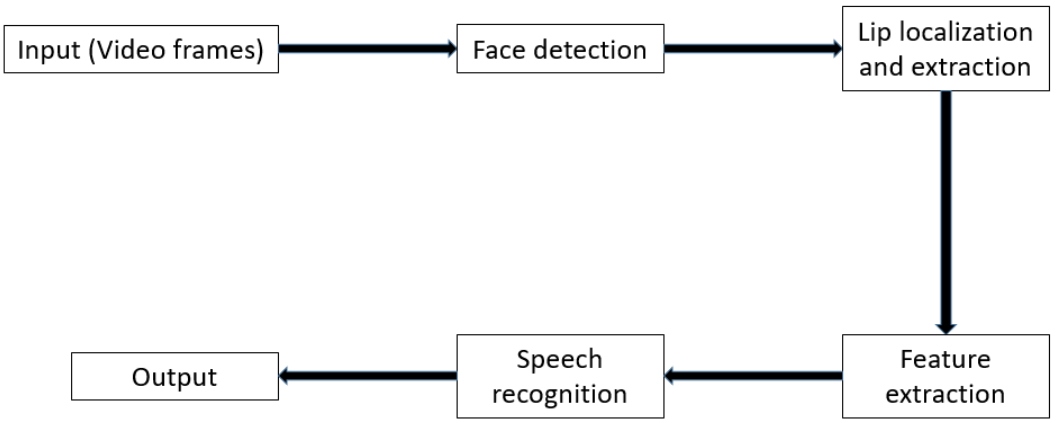

Figure 5: A representation of usual steps taken for current VSR algorithms

Face detection means providing a bounding box from an image that contains a face, cutting on the dimensionality and making a prediction independent of any background. The resulting box will be fed into a landmark detection algorithm, which will detect specific parts of a person's face, giving us a possibility to cut only the lips from a specific person. The most commonly used landmark detection method is the one found in "dlib" library: 68-point facial landmarks. The output of such detection gives 68 pairs of x, y coordinates corresponding to landmarks found in the detected face. Figure 6 shows an usual mapping of a face with the 68-point facial landmarks.

The LRRo datasets, both the ones captured in a lab and the ones captured in a wild environment, will already do these two steps, using OpenCV's facial detection to capture a bounding box of a face and dlib's 68 landmarks of a face to give a bounding box for the lips. The result of these operations are 29 frames of 64 by 64 grayscale images.

Feature extraction can be mainly done in two ways:

- Manually, which will call for an experienced person in the fields of both linguistics and lip reading, usually being done on a per-person basis. Given this, it is hard to scale this solution on a large population.
- Automatically, with a set of (shared) convolutions applied to each frame. Given that the filters applied by a convolution are learned, there is no need for an experienced human

agent to provide any information. Such operations are done with Conv2d layers from tensorflow, or even Conv3d, that takes into account time as well, and not only spatial features.

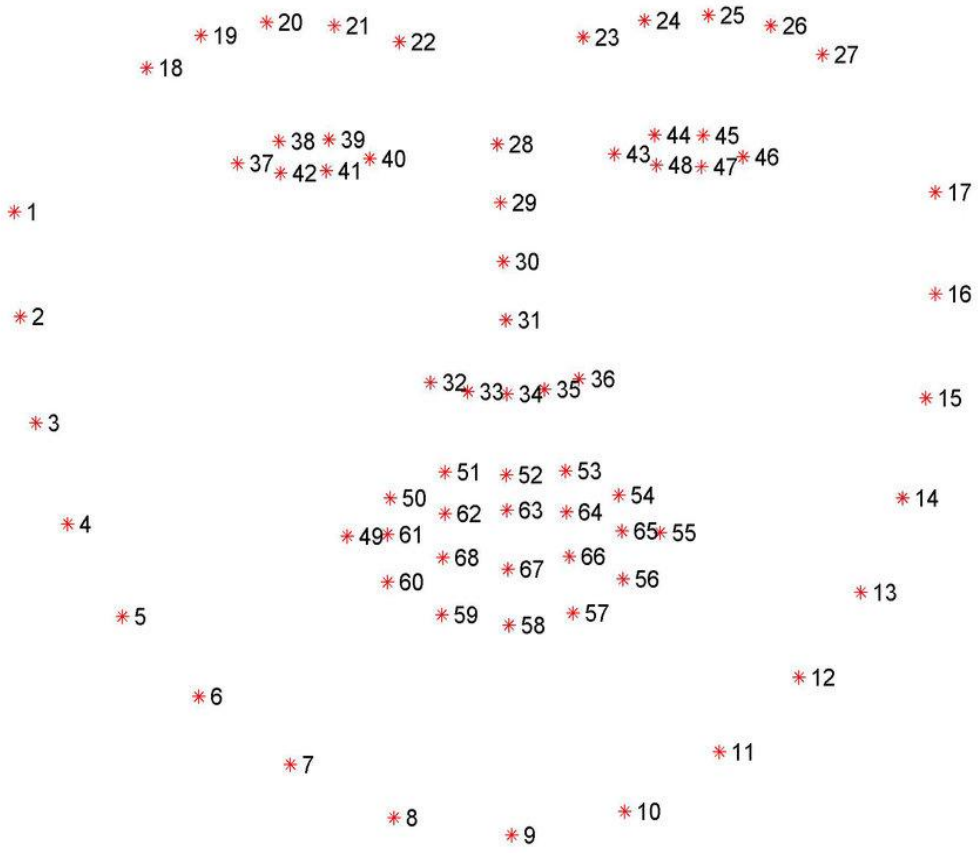

Figure 6: 68 landmarks found on a face

The last step that is needed for a full VSR architecture is input evaluation and decision making. It should be able to give the correct output based on the input (frames of lip movement). The evaluation can be done with multiple methods, including ones that use hand crafted language analysis to ample deep learning architectures. The output can have a form of a category or even a string of characters (the words spoken). This part is the one the most researched in the past years, being the hardest to solve of them all. Numerous approaches were tried with this purpose in mind, most of them resorting to classifying words spoken on a limited vocabulary. Two main problems arise in this case, first being the usability of such architectures, given that rarely the information found in one word is useful in a way. The second problem would be that architectures that are only categorizing the input have a scalability problem, being that the size of

an output has a factorial scaling based on the dictionary size, assuming that each combination of words is possible in a given language. If we were to implement such architectures for the english language, which has around 273.000 words according to the Oxford Dictionary, we would have the following number of neurons on the output layer:

$$273000! - (273000 - n)!, \; where \; n = number \; of \; words \; in \; a \; sentence \; we \; want \; to \; predict$$

Overall, there have been extensive improvements in the past years to ALR with or without deep learning, achieving great results, but only with a limited vocabulary size. There have also been some tries to move the problem away from a classification one, with moderate success.

a. Implementing a neural network capable of reading romanian words

7.1 Architecture

The network(s) trained in this project were inspired by the work done in LipNet[5], preserving the core architecture, but with different implementation details. There are two developed networks, one suited for the LRRo Wild dataset, and one suited for LRRo Lab dataset, which differ slightly, only on the output layer, with the number of classes that are present, lab dataset having 48 different classes, compared to the wild dataset, with only 21.

The input layer expects 29 frames of 64x64 grayscale images with the mouth in the center. For each of these frames the model applies four two dimensional convolutions, with shared weights, with 64, 128, 256 and respectively 512 filters each, with every layer being followed by a maximum pooling. There is an increasing number of filters being used for each layer so that the model is able to make broader and broader generalisations and abstractisations. This is the convolutional part of the architecture, used for feature extraction from input frames.

The aforementioned convolutions, with the features it has extracted, will be transformed in a time distributed sequence and will be fed into two Bidirectional Gated Recurrent Units (Bi-GRUs), each with 64 neurons, used to extract features that are linked in time, with the first one returning sequences that will be used by the next one as input.

The final step of decision making is done with three layers of linear, dense, neurons, which will take the time distributed input provided by the Bi-GRUs determine in which class it belongs. The layers have a number of 1024, 512 and 256 neurons respectively, with the first three layers having a dropout rate $p = 0.5$, needed so the model will avoid overfitting on training data and help it make more generalised decisions, given the relatively small amounts of inputs for each class.

Each of the layers presented above will use the Rectified Linear Unit (ReLU) activation function, with the expectation of GRU layers, which will use its specific tanh function.

The outputs of the linear layers will determine the predicted class, with the output layer having 48 and 21 neurons, for the Lab LRRo dataset and Wild LRRo dataset respectively. A softmax will be applied on the outputs to determine the maximum likelihood predicted by the model.

The resulting model will have around 5.8 million trainable parameters. A simplified version of the model can be found in figure 7.

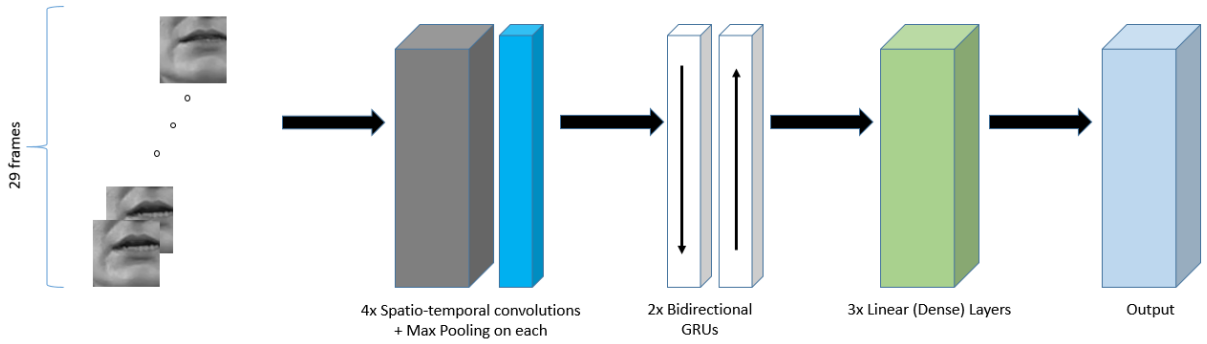

Figure 7: Custom built architecture for the lip reading problem

7.2 The datasets [7]

All training and evaluation will be done on the data provided with the LRRo paper, by researchers from University of Bucharest. The data is composed of over 1,200 minutes of TV shows recording and natural speech recordings. Videos are somewhat gender balanced, having a proportion of around 65% males and 35% females. Each one of them has an exact number of 29 frames and are preprocessed in such a way to provide only a grayscale image of a mouth crop, as an image file, with JPG format. If a word was spoken in less than 29 frames, but not less than 6, they used duplication of the last frame in order to achieve the desired 29. If the word had less than 6 frames when pronouncing it, it was disqualified and deleted from the final dataset. Annotations of each word were done both by multiple manual annotators and two automatic speech recognition systems. The number 29 for frames was chosen because it was the maximum number frames needed for a word to be spoken. The data is separated into three sets: training data, validation data and testing data, with the recommended proportion of each category being 78%, 11% and 11% respectively. Each word class is separated in a folder, with each class having a list of speakers pronouncing that word. The folder structure found in the LRRo dataset can be seen in figure 8.

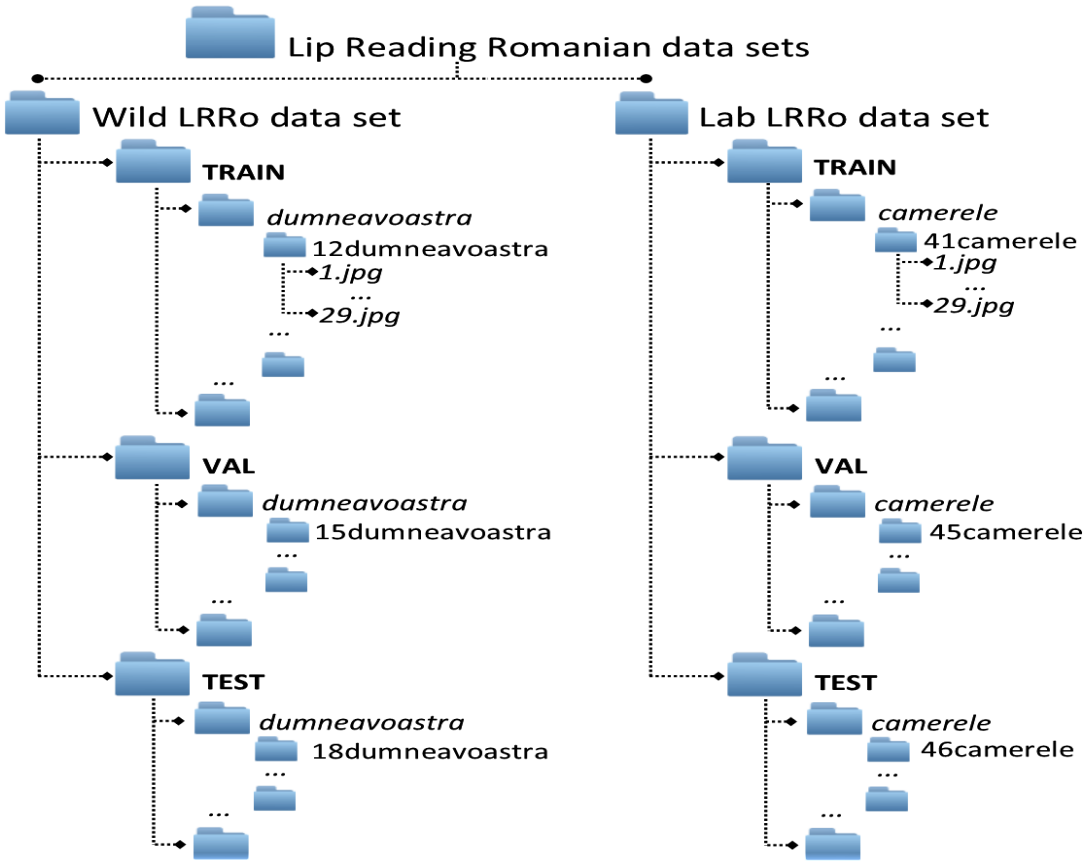

Figure 8: LRRo data folder structure

Each of the two datasets have the same structure, but are different in other aspects. There will be more details on both in the following section:

- Lab_LRRo_data_set, which includes videos that were shot in a controlled environment, with a static background and perfect lighting conditions. Most of the 19 subjects present in these recordings are young adults attending university. Lexicon present in this dataset consists of the 48 classes, representing a word in the romanian language with more than 6 letters, plus another class, with multiple words with less than 6 characters, which will be a distractor. A number of 6505 utterances can be found for training, 860 for validation and 815 for testing. Figure 9 contains images from the videos used to create the Lab LRRo dataset.

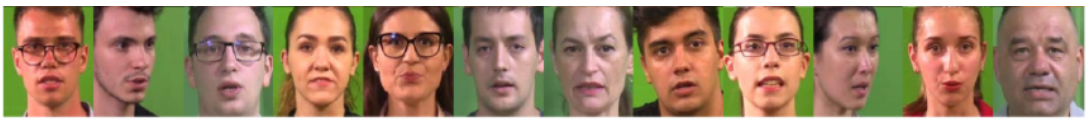

Figure 9: Examples of video frames used for Lab LRRo Dataset

- Wild_LRRo_data_set, includes videos found on the internet, in a wild, ad-hoc environment, downloaded from Youtube. It consists of an extensive number of different viewing angles, backgrounds and lighting conditions. The 35 different speakers found in the dataset were chosen from different TV shows and include various ages, genders and backgrounds. The presented lexicon consists of 21 distinct classes, each representing words in the romanian language with more than 6 letters and the same distractor class specified above. A number of 848 utterances can be found for training, 120 for validation and 121 for testing. Figure 10 contains images from the videos used to create the Wild LRRo dataset.

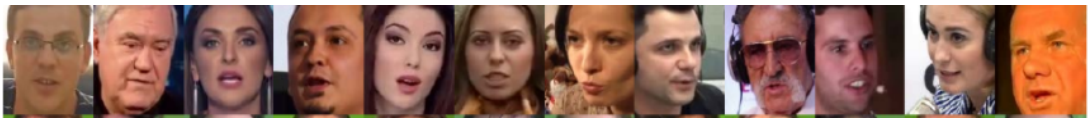

Figure 10: Examples of video frames used for Wild LRRo dataset

There have been a few modifications applied to the given datasets, namely:

- Transforming the data from a series of 29 images in jpg format to a video file of avi format, which helped speed up the input/output process by a significant margin and made the dataset tree clearer.
- Duplicating the last frame in case the utterance did not meet the 29 frames threshold means that a given frame (the one duplicated) will have a larger say in the output than the normal ones. To solve this, instead of copying the last frames multiple times, if a frame is the exact same as the previous one (i.e. is the one copied by the imposed rule), it will be replaced by a totally black frame on the grayscale. This means that neurons associated with that frame will not be activated, therefore rendering the frame useless for the task.
- With the risk of overfitting, the low number of training data found in the datasets has determined a merging of the validation data with the training data, dropping the validation step altogether. This means that instead of 848 videos for training, the number will go up to 968 for the Wild dataset and from 6505 to 7365 for its Lab dataset counterpart.

7.3 Training

Different models were trained for the different datasets. The models were implemented using Tensorflow, with the Keras library. Table 3 contains the value of each hyperparameter used in

training. The chosen optimizer is Adam, being widely considered faster than the classic gradient descent, with its specific first momentum and second momentum values set to 0.9 and 0.999 respectively. The numerical stability number is $\varepsilon = 10^{-7}$. Categorical cross entropy (CE) is used as the loss function, given that we want to distinguish from multiple classes. Its value is computed by the formula:

$$CE = -\sum_{i=0}^{n} y_i \cdot \log(\hat{y}_i)$$

Where n is the number of unique classes on the output, $\hat{y}_i$ is the i-th scalar in the model's output and y_i is its corresponding scalar value in the ground truth. If we have a perfect prediction on the classes (a value of 1 for the correct class, and 0 for the rest) the loss value will be 0. In contrast, if the predicted chance of the correct class is 0.01, the value of the loss is 4.605. Given that each non-fitting category has a value of 0 in the ground truth, the CE loss value will be dependent only on the output likelihood of the correct class.

Normal training procedures will load each datapoint into memory, which is not feasible on datasets that are big in size, because it will surpass the total available memory. For such cases, data generators have been created, which loads into memory only the data points used by the current training batch. This greatly reduces the memory needed for training and is recommended to be used whenever we work with images or videos as training data. There is a built-in generator for images in Tensorflow, but not for videos, therefore there was the need of writing a custom generator. Tensorflow gives a template class that must be implemented if we want it to work with networks created with it. The main methods that must be implemented are dunder methods *__getitem__* and *__len__*, with the first one returning a collection of training data with the length of the used batch size and the second one returning the number of batches per one epoch (total collection size divided by the number of batches).

Parameter name	Parameter value for lab dataset training	Parameter value for wild dataset training
Mini-batch size	32	16
Number of epochs for training	55	100
Learning rate	10^{-4}	10^{-4}

Table 3: Hyperparameters used for training

We have computed the results found over training duration and they will be covered in the following section.

- Lab dataset training, in this case, the model has steadily improved over the training epochs, with the accuracy curve resembling a logarithmic scaling, reaching up to 93% accuracy on training data and 0.2 cross entropy cost, but it has reached a plateau after 40 epochs, a sign of convergence. The graphs containing information on each training step can be found in figure 9. The model might have increased in accuracy if it was trained for longer, but we would run into the risk of overfitting. The training has been going for around 3 hours and 45 minutes, with an average of 245 seconds needed for each epoch.

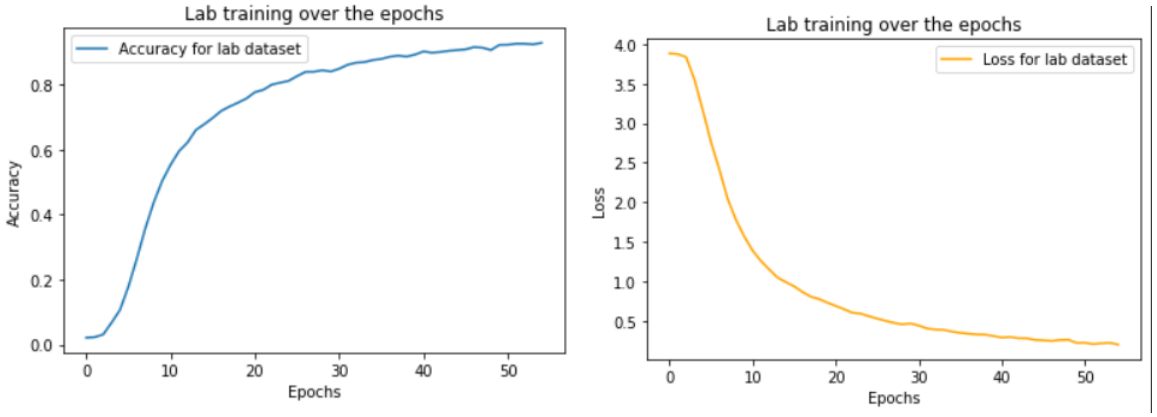

Figure 9: Accuracy increase (Left) and and Loss Decrease (Right) over training time

- Wild dataset training, where we can also see a steady improvement over time, with the accuracy graph resembling more of a linear scaling. The model was trained for longer because the collection size was considerably smaller. Batch size has also been decreased in order to have an increasing amount of parameters updated in a given epoch. It has reached an upward of 80% accuracy and around 0.6 loss on training data, but as we will

see in the next section, it was due to partial overfitting on training collection. This was somewhat expected to happen given the large number of epochs, but if it is trained for less time, it will fare even worse on data it has never seen, on testing. The graphs containing information on each training step can be found in figure 10. The model might have slightly better accuracy if it was trained for longer, but that will come with the cost of largely overfitting it to training data. Compared to the model trained for laboratory-made dataset, we can observe a higher variance and a more linear accuracy curve. It does not seem to reach a plateau, which normally means that it can still improve, but even if the model might have slightly better accuracy if it was trained for longer, it has a high chance of coming with the cost of largely overfitting it to training data. The model has trained for around 50 minutes, with an average of 30 seconds per epoch, which is a much lower time compared to the time needed for the lab dataset, understandably so, given that this collection is about 7.5 times smaller.

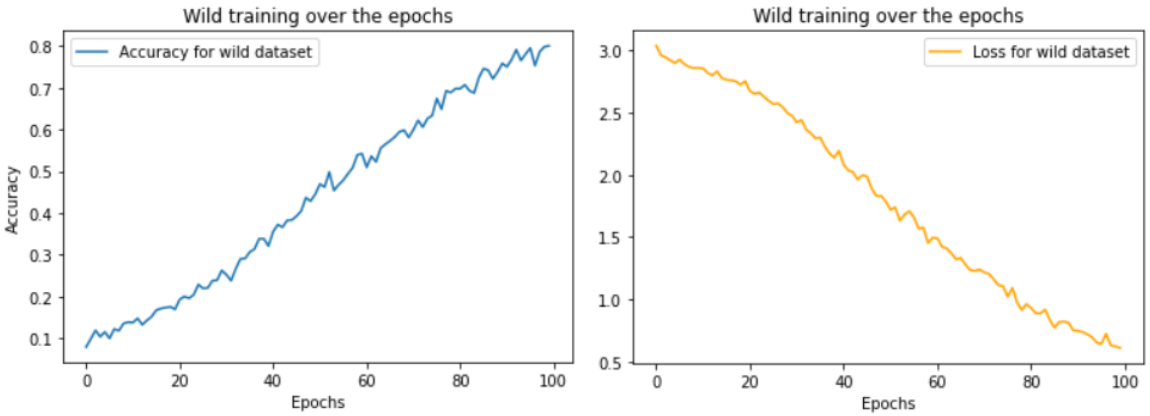

Figure 10: Accuracy increase (Left) and and Loss Decrease (Right) over training time

7.4 Performance evaluation and results

In order to measure how well the model is doing, we will compute accuracy, the word error rate (WER) and character error rate (CER), which is a standard for measuring performance in the field of Automatic Speech Recognition. WER and CER were defined above, but their definition will be mentioned again. WER (or CER) is defined as the minimum number of word (or character) insertions, substitutions, and deletions that are required to transform the model's prediction in the ground truth, divided by the number of words (or characters) in the ground

truth. In our case, WER is equal to 1 - model's accuracy, given that we have only one word in the prediction. In the absence of studies of human lip readers in romanian, we will use data for the English language. The difference should not be too big, given that word level lip reading is easily transferable between languages. According to studies done by Easton & Basala in 1982 [1], the accuracy of human lip readers on polysyllabic english words is around 20%, so we will use that as a baseline for comparison. The results will be compared with the baseline systems proposed by the LRRo datasets developers, which only has WER results. The last comparison will be done with word level ALR systems for the English language, to have a view on how the proposed model fares against the ones found for English.

The results of training, which can be found in table 4, proves that a custom made model will perform better than ones with general purposes (baselines proposed for LRRo), having an accuracy increase on testing data of more than 76% for the wild LRRo dataset and an increase of over 26.7% in the case of lab LRRo dataset. It was expected for accuracy on lab LRRo to not increase as much, given that the number was already at a pretty high point. Compared to english models, it performs approximately at the same level, with the proposed model having an edge over the one mentioned.

Method	Dataset	Dataset Size	WER	CER	Accuracy
Human lip-reader (avg.)	N/A	N/A	78%	-	22%
Papandreou et al (2009) [12]	CUAVE	1800	17%	-	83%
Chung & Zisserman [13]	BBC TV (LRS2)	>400000	34.6%	-	65.4%
LRRo Baselines (Top1)	Wild LRRo	846	67%	-	33%
Proposed model	Wild LRRo	968	41.9%	40%	58.1%
LRRo Baselines (Top1)	Lab LRRo	6505	29%	-	71%
Proposed model	Lab LRRo	7365	10%	9%	90%

Table 4: Results for the proposed model, compared existing ones

As expected, the 80% accuracy present in training of the wild dataset has not translated in the same accuracy on data that the model has never seen, netting an accuracy of only 58.1%, a 22% decrease, which is a lot. The main reason for this is clearly overfitting, which cannot be avoided on longer training periods, but there is also another cause, namely a small collection of data, which is known to lead to such situations. In contrast, with a similar model, proposed for the lab LRRo dataset, such overfitting conditions are not met, with the collection being considerably larger and training period being significantly reduced, which results in only a 2-3% decrease in accuracy when we transfer the model from training to testing.

If we analyze the correlation between word error rate and character error rate, we can see that they are around the same value, this might be because of multiple reasons, most important of which being that words present in the proposed lexicon have just a few common letters on the same position, therefore if the model has got the word as a whole wrong, there is only a small chance that any characters will fit to the correct prediction. An extended confusion matrix can be found in figure 11 for wild dataset and in figure 12 for lab dataset, giving us more insights on which words the model is finding hard to predict. A confusion matrix shows us which words are confused amongst them, giving us an insight about both false positives and false negatives. A high value on the cell (i, j) from this matrix means that the word w_i is usually confused with the word w_j. The sum of each row has to be exactly 1. High resolution images can be found here, and here respectively.

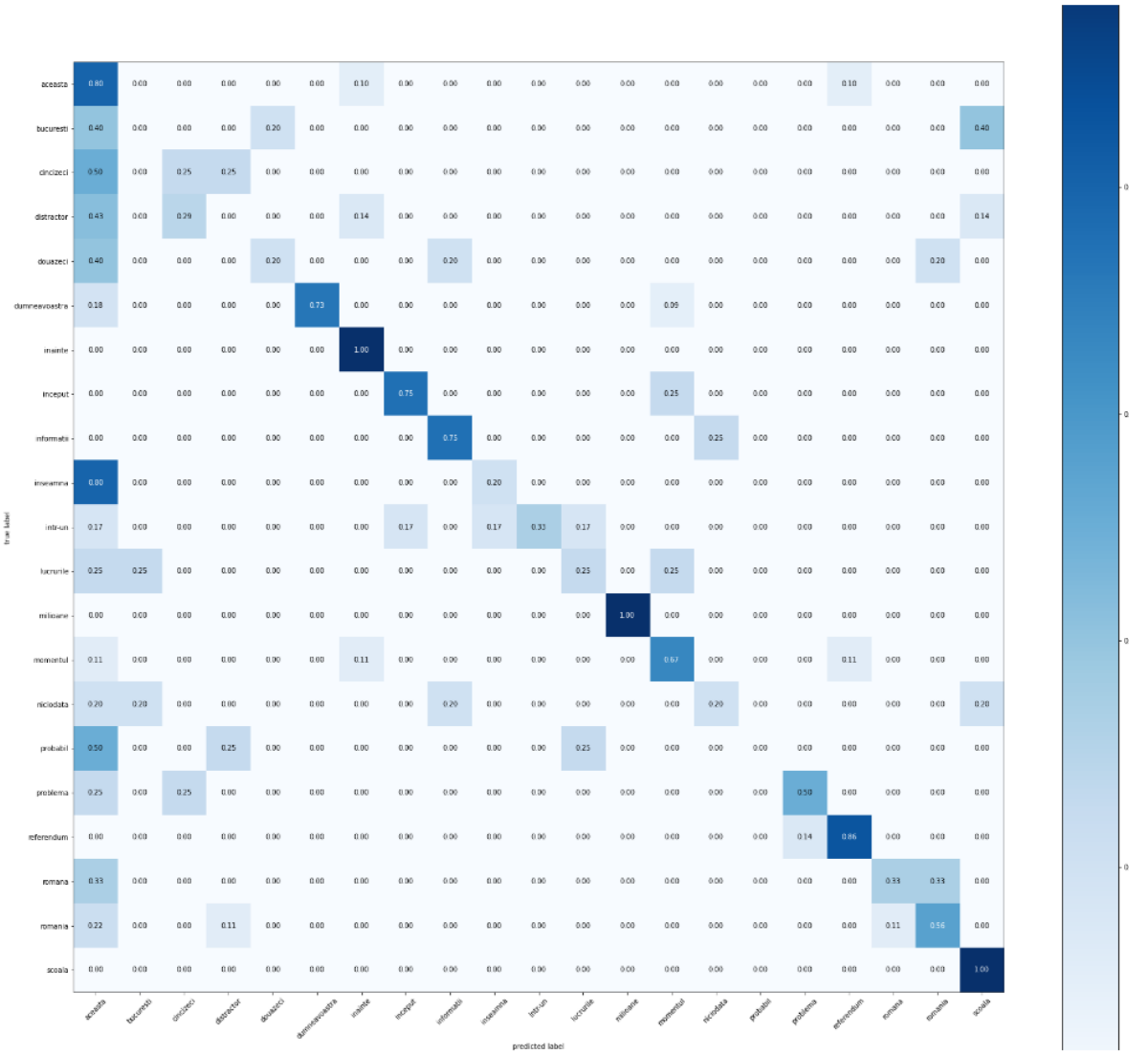

Figure 11: Confusion matrix for wild dataset

In this case, the matrix shows extreme amounts of variance, having values between 0 and 1 on cells (i, i), which means that some words are always guessed right and others are never guessed. Examples of always confused words are: “București”, mostly being mistaken with “aceasta” and “școala”, “probabil”, being mistaken with “aceasta” in half the cases. The last but not least words that were never guessed are the ones found in the distractor class, understandably so, given that there lies a collection of multiple words, most of them with nothing in common, therefore it is extremely hard to construct a model capable of predicting such a class. It is clearly seen that the award of most false positives output goes to the word “aceasta”, which might be caused by the large number of instances for this class compared to others in the collection, or multiple training examples being present in the training’s beginning.

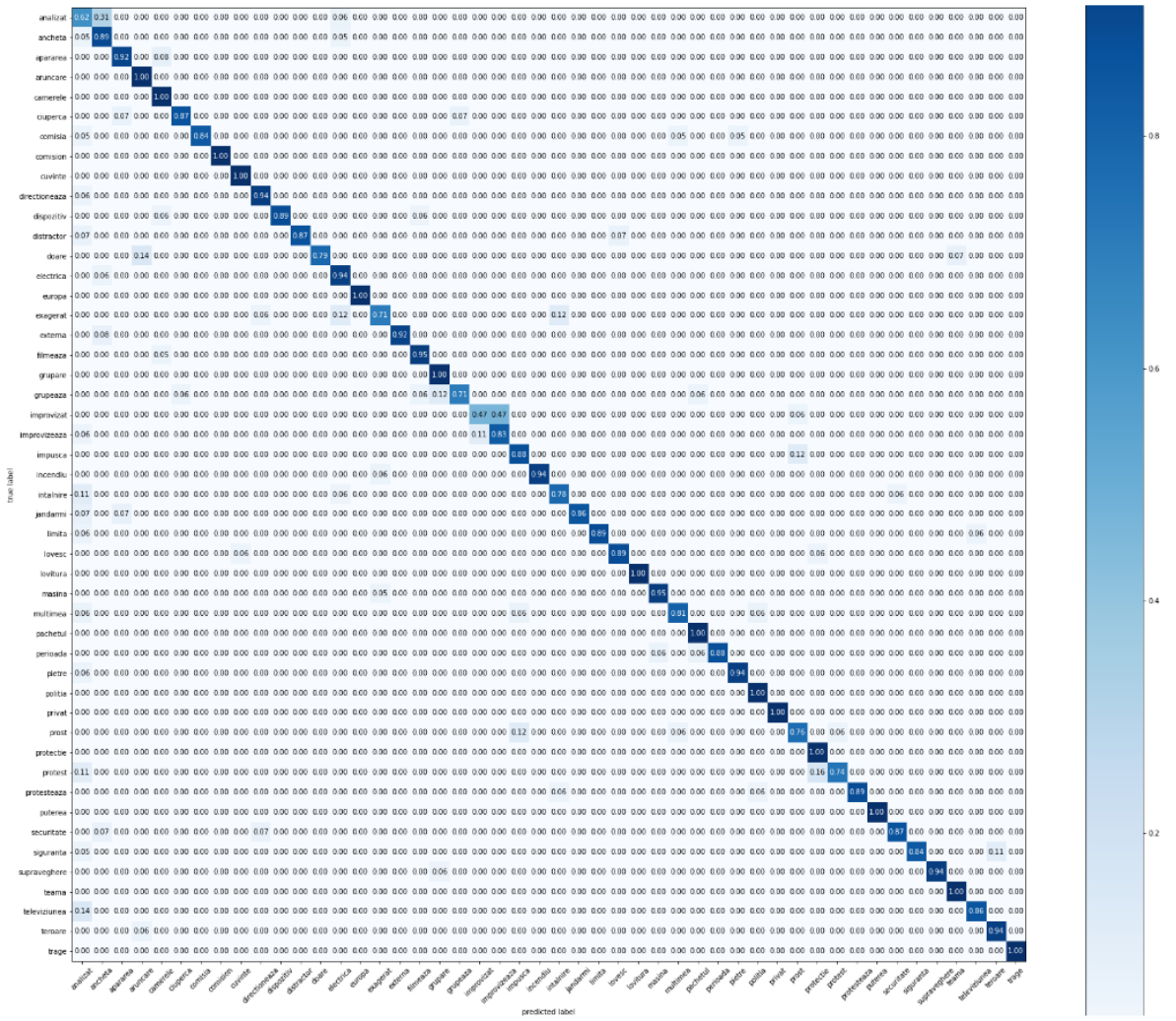

Figure 12: Confusion matrix for lab dataset

When we look into the confusion matrix for the lab dataset, we can clearly see that there is a much lower variance to it, given its 90% base accuracy. Most of the high values will be found on the (i, i) cells, with a good portion of them even reaching the perfect prediction mark, meaning that the word has been correctly classified with every instance of testing data. Most of the words that have a high split of accuracy are the ones having comparable spelling or pronunciation, such as “analizat” with “anchetă” and “improvizat” with “improvizează”. Again, the most false positives will be found on the first column, namely on the word “analizat”, which may be caused by the same reasons stated above.

7. Conclusions

There are multiple approaches if someone wants to take upon the Visual Speech Recognition task, with this work covering ones ranging from methods that will require him to have extensive knowledge of linguistics by building a manual vision pipeline, to methods that will require little to no knowledge in that domain, but extensive knowledge in the field of Computer Science and Machine Learning. As with most tasks that are researched into the field of ML, computers will inevitably surpass the skills of a human doing the same task. This has already happened in the case of ALR even from early development stages, without a need of any Artificial Intelligence, making it a promising prospect for the future.

Tremendous improvements have happened in the past years in this field, with computers reaching the 97'th percentile in terms of accuracy, which is more than enough to transmit a coherent message when audio channels are not available. There are also steps being taken to go from word level to sentence level, which would be a lot more useful in real life scenarios.

The work done in this project builds upon the bases laid by the providers of the LRRo datasets. It is most likely the first model actually developed for the Romanian language, aside from the baselines proposed with the datasets, which were an adaptation of existing models. It has yielded significantly better results than the baselines, and to my knowledge, is the most accurate model for lip-reading in Romanian.

There is a clear contrast between the two collections proposed in LRRo, which will definitely have an impact on the results any model will have on them. The controlled nature of the laboratory dataset will clearly give it an edge compared to the videos that were recorded in a wild environment. In addition, in training, there have been some inconsistencies found with the wild dataset, both with the (maybe) wrong annotations done by the automated annotator and the mismatching of frames. There were some occasions where the face of the speaker changed from one frame to another. On top of that, the added distractor class, which has no common elements in terms of speech, will make it near impossible to determine its features, because the question transforms from "Are the characteristics of this video pointing me towards this word?" to "Are all the other alternatives not fit for this given video?", which is a much harder question to answer, and will most likely lead to a "no" in most cases. With this in mind, it is extremely difficult to devise a network capable of reaching over 95% accuracy for the wild data collection and over 98% for the laboratory-made one.

All in all, all data created for research is great, being the building blocks of almost every problem that can be tackled through machine learning. In addition, it provides data for languages

that are severely under-sourced, therefore raising the bar in the field of ALR in languages that are ignored, even if a significant population is speaking it.

Without a doubt there are improvements that could be added to the proposed model, so it will be both faster and more accurate. It is known that a small difference in a network's architecture could achieve significant improvements or it can render it unusable. Also adjusting the hyperparameters or optimizer could be a key factor of improvement.

Applying the given algorithm to a larger dataset will certainly lead to better results overall, as it is with every neural network.

The biggest improvement that could be done is taking the architecture from a word-level architecture to a sentence level one. Regrettably, there is no dataset for such a thing in the Romanian language, nor in any other language other than English. It would be a great advancement to transfer the learning that was accumulated over the years in the field of ALR for the English vocabulary in other languages.

Even if more limited than a sentence level lip reading, the presented model could still have some applications in the real world, even on a limited vocabulary. For example, short vocal commands, in the absence of audio, be it because the environment was too noisy, or the person is not able to speak. If each word would mean a command, if we are able to predict the speech of a given agent, it would be enough to launch a procedure to execute that command.

8. References

1. D. Easton and M. Basala. Perceptual dominance during lipreading. Perception & Psychophysics, 32(6): 562–570, 1982.
2. Deafness and hearing loss, released by the World Health Organization in April 2021.
3. J. Goldschen, O. N. Garcia, and E. D. Petajan. Continuous automatic speech recognition by lipreading. In Motion-Based recognition, pp. 321–343. Springer, 1997.
4. Gergen, S. Zeiler, A. H. Abdelaziz, R. Nickel, and D. Kolossa. Dynamic stream weighting for turbodecoding-based audiovisual ASR. In Interspeech, pp. 2135–2139, 2016.
5. Y. M. Assael, B. Shillingford, S. Whiteson and N. De Freitas, Lipnet: End-to-end sentence-level lipreading, 2016.
6. Kai Xu, Dawei Li, Nick Cassimatis, Xiaolong Wang, LCANet: End-to-End Lipreading with Cascaded Attention-CTC, 2018.
7. Andrei Cosmin Jitaru, Şeila Abdulamit, and Bogdan Ionescu. 2020. LRRo: A Lip Reading Data Set for the Under-resourced Romanian Language. In 11th ACM Multimedia Systems Conference (MMSys'20), June 8–11, 2020, Istanbul, Turkey.
8. Statistics released by the Statista team, The most spoken languages worldwide in 2021.
9. Neural Networks course, from the Computer Science Faculty from “University Alexandru Ioan Cuza”, 2020
10. Bernhard Mehlig, Machine learning with neural networks, 2019.
11. Hochreiter and J. Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997
12. Papandreou, A. Katsamanis, V. Pitsikalis, and P. Maragos. Multimodal fusion and learning with uncertain features applied to audiovisual speech recognition. In Workshop on Multimedia Signal Processing, pp. 264– 267, 2007.
13. S. Chung and A. Zisserman. Out of time: automated lip sync in the wild. In Workshop on Multi-view Lip-reading, ACCV, 2016b.