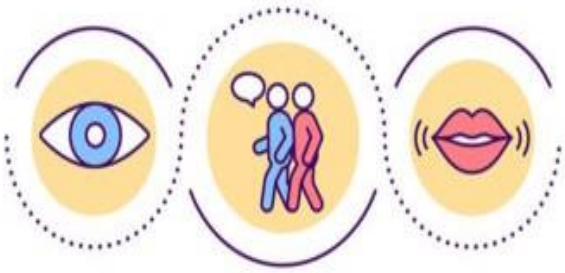
Ain Shams University
Faculty of Computer & Information Sciences
Computer science Department

I Can Hear You

Arabic and English Lip-Reading



July 2023

Ain Shams University
Faculty of Computer & Information Sciences
Computer science Department

I Can Hear You

Arabic and English Lip-Reading

Proposed by:

Mahmoud Salama Mohamed [Computer science]

Mahmoud Saeed Sayed [Computer science]

Eman Ibrahem Yusuf Sam [Computer science]

Noura Ahmad Hafez [Computer science]

Aalaa Ahmad Mohamed [Computer science]

Under Supervision of:

Dr. Mohamed Mabrouk
Computer science Department,
Faculty of Computer, and Information Sciences,
Ain Shams University.

T.A. Hazem Yousef
Scientific Computing Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

July 2023

Acknowledgements

At the beginning we would like to express our gratitude & thanks to Allah and to all who helped us completing this project.

Especially our supervisors,

Dr. Mohamed Mabrouk for his understanding, patience and encouraging us all the time to do our best, he has always been there for us.

TA. Yousef Hazem for his supervision and great effort.

We are proud that we have been surrounded by this amount of care andbeautiful spirt that pushed us to do our best till the last time.

Finally, we would like to thank our family for supporting us and friends for sharing their experience and knowledge.

Abstract

Due to the limited number of people who know sign language or can interpret lip movements, deaf individuals often face difficulties communicating with others. To address this issue, this project aims to develop an application that tracks the lip movements of a person speaking in a video and predicts the corresponding text to make communication easier for the deaf.

This web application extracts the mouth region from each frame of the video as a region of interest and maps each frame to the corresponding character.

The project consists of two different models:

- The first model is for Arabic language input data, which takes 3D input data, applies convolutional and pooling operations to extract spatial features, processes temporal information using bidirectional GRU layers, and produces a probability distribution over the classes using a dense layer followed by SoftMax activation. The accuracy of this model is 80%.
- The second model is for English language input data, which takes 3D input data, applies convolutional and pooling operations to extract spatial features, processes temporal information using bidirectional LSTM layers, and produces a probability distribution over the classes using a dense layer followed by SoftMax activation. The accuracy of this model is 97%.

Table of Content

List of figures

List of Tables

List of abbreviations

AI	Artificial intelligence
ANNs	Artificial neural networks
BI-LSTM	Bidirectional long short-term memory
CNN	Convolutional neural networks
D	Dimensional
DL	Deep learning
FC	Fully connected
GRU	Gated recurrent units
HPC	High performance computing
LRS2	Lipreading Sentence in the wild version 2 (dataset)
LRW	Lipreading sentence in the wild (dataset)
LSTM	Long short-term memory
NLP	Natural language processing
NN	Neural networks
RELU	Rectified linear unit
RNN	Recurrent neural networks
ST	Spatiotemporal
TF	Tensorflow
TM	Transformer model
VGG	Visual Geometry Group (architecture)

Table 1 abbreviations

Chapter One: Introduction

1. Introduction:

This chapter introduces the reader to the author's research project. It explains the problem background and description, research objectives and importance.

1.1. Overview:

Lipreading, also known as speechreading, is a technique of understanding speech by visually interpreting the movements of the lips, face, and tongue when normal sound is not available, also it plays a vital role in human communication and speech understanding. So, visual speech recognition "lipreading" is a field of growing attention. It is a natural complement to audio-based speech recognition that can facilitate dictation in noisy environments and enable silent dictation in offices and public spaces. It is also useful in applications related to improved hearing aids and biometric authentication.

1.2. History:

Deaf individuals utilize two primary methods for communication: lip reading and sign language. Sign language is a visual language widely employed within the deaf community, relying on hand gestures, facial expressions, and body language to convey words and emotions. The origins of sign language can be traced back to Spain, but it was in France where the first formal sign language was established. Charles Michel de l'Eppe, concerned with deaf rights, founded the original public school for deaf children in 1755, marking a significant milestone [1].

Lip reading, on the other hand, is commonly used by deaf individuals who do not know sign language, particularly those who were born with hearing abilities but gradually or suddenly lost their hearing during their lifetime. The World Congress of Educators in Milan made a significant decision in 1880, stating that all deaf children should be taught lip reading [2]. Lip reading instruction focuses on observing the movements of the lips, tongue, and jaw, as well as understanding the stress, rhythm, and expression of spoken language. Students learn the lip readers' alphabet and identify groups of sounds that share similar lip shapes (visemes), such as "ت – ط – د","ز – ص – س ", "ج – ش – ي" "p," "b," "m," or "f," "v." The goal is to grasp the overall message, enabling individuals to confidently engage in conversations and prevent the social isolation that often accompanies hearing loss.

1.3. Problem Definition:

People with hearing problems as deaf people face difficulties in communicating with other people, some of whom may not know the sign language. Lip reading is one communication skill that can help them communicate better and understand what is being said. movements and translating them into text.

1.4. Motivation:

Assisting individuals with hearing difficulties in understanding speech is crucial. Since human lipreading performance is often limited, unlike machine lipreaders that offer significant practical potential, converting videos of people speaking into text can greatly enhance communication for deaf individuals. This technology enables them to easily interact and engage with society.

1.5. Objectives:

- Conduct a survey of different deep learning architectures from previous research papers that have worked on our problem to determine their strengths and weaknesses.
- Build an Arabic language model and train it on an Arabic dataset.
- Build an English language model and train it on the GRID dataset.
- Develop a web application that allows users to upload or capture a video, which will be processed by the model to predict the corresponding text, and the predicted text will be displayed on the application.

1.6. Time plan

1.7. Applications:

Currently, numerous applications are being developed with a focus on "lipreading." One notable example is in the field of **self-driving cars** [4]. In the bustling environment of city streets, where noise levels can be high, passengers may need to provide instructions or indicate their desired destination to the car. In such situations, where sound signals may be compromised by excessive background noise, lipreading technology proves beneficial for achieving more accurate results.

Lipreading plays a vital role in speech processing for face-to-face videos, particularly in the creation of fake videos and animation movies [5]. By capturing the lip movements of a real person and transferring them to a cartoon character, realistic lip movements can be achieved during dialogue, enhancing the authenticity of the animated character's speech.

Lipreading is utilized in password authentication [6] as a secure method of verification. Similar to a unique fingerprint, each individual possesses distinct lip movements that can be considered as their unique signature. While different individuals may pronounce the same word in various ways due to accents and cultural backgrounds, their lip movements remain specific to them. In contrast to the traditional method of typing passwords, which can be susceptible to hacking and visual interception, lipreading provides a more secure approach to authentication.

1.8. Document Outline:

- **Chapter 2:**
  This chapter presents algorithms used in this project. By explaining Neural Networks (NNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), Bidirectional Long Short-Term Memory (BLSTM).

- **Chapter 3:**
  This chapter presents all the previous related works, papers, history of the problem and ways of solving it.

- **Chapter 4:**
  This chapter describes the model's architecture, implementation, android application, pre-processing, and the used dataset.

- **Chapter 5:**
  This chapter states the experiments and results of the trained models.

- **Chapter 6:**
  In this chapter we introduce the user manual and the installation guide and allthe dependencies to run our application.

- **Chapter 7:**
  In this chapter we state our conclusion after all the project phases and the best model and future works

- **References:**
  References for papers, websites we used to come out with this project.

Chapter Two: Background

1. Background:

In this chapter, the authors provide an overview of the background technologies that have been utilized and explored to accomplish the objectives of this project.

1.1. Intro to Artificial Intelligence:

Artificial Intelligence (AI) is an approach that aims to enable computers, robots, and other products to emulate human intelligence. It involves studying how the human brain thinks, learns, makes decisions, and solves problems in order to develop intelligent software systems. The primary goal of AI is to enhance computer functions related to human knowledge, such as reasoning, learning, and problem-solving[8].

The objectives of AI research encompass various areas, including reasoning, knowledge representation, planning, learning, natural language processing, realization, and the ability to manipulate objects. Furthermore, there are long-term goals within the field of general intelligence, which seeks to create AI systems that exhibit a comprehensive range of intelligent behaviors.

2.1.1. Need for Artificial Intelligence:

1. To create expert systems which exhibit intelligent behavior with the capability to learn, demonstrate, explain and advice its users.
2. Helping machines find solutions to complex problems like humans do and applying them as algorithms in a computer-friendly manner.

1.2. Intro to Deep Learning:

Simply, Deep learning is just learning from examples. That's pretty much the deal. At a very basic level, deep learning is a machine learning technique that teaches a computer to filter inputs (observations in the form of images, text, or sound) through layers to learn how to predict and classify information.

Deep learning is inspired by the way that the human brain filters information!

Essentially, deep learning is a part of the machine learning family that's based on learning data representations (rather than task-specific algorithms). Deep learning is closely related to a class of theories about brain development proposed by cognitive neuroscientists in the early '90s. Just like in the brain (or, more accurately, in the theories and models put together by researchers in the 90s regarding the development of the human neocortex), neural networks use a hierarchy of

layered.

filters in which each layer learns from the previous layer and then passes its output to the next layer.

Deep learning attempts to mimic the activity in layers of neurons in the neocortex. In the human brain, there are about 100 billion neurons, and each neuron is connectedto about 100,000 of its neighbors. Essentially, that is what we're trying to create, butin a way and at a level that works for machines. The purpose of deep learning is to mimic how the human brain works to create some real magic.

What does this mean in terms of neurons, axons, dendrites, and so on? Well, the neuron has a body, dendrites, and an axon. The signal from one neuron travels down the axon and is transferred to the dendrites of the next neuron. That connection (not an actual physical connection, but a connection nonetheless) where the signal is passed is Neurons by themselves are kind of useless, but when you have lots of

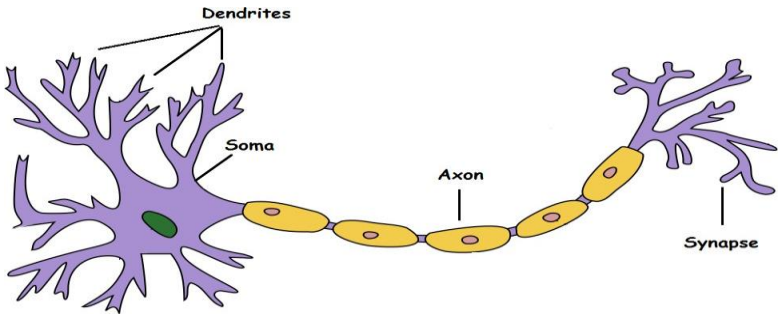


Figure 1 normal nerve cell

them, theywork together to create some serious magic. That's the idea behind a deep learning algorithm! You get input from observation, you put your input into one layer that creates an output which in turn becomes the input for the next layer, and so on. This happens over and over until your final output signal! called synapse[9]

So, the neuron (or **node**) gets a signal or signals (**input values**), which pass through the neuron, and that delivers the **output signal**. Think of the input layer as your senses: the things you see, smell, feel, etc. These are independent variables for one single observation. This information is broken down into numbers and the bits of binary data that a computer can use. (You will need to either standardize or normalize these variables so that they're within the same range).

What can our output value be? It can be **continuous** (for example, price), **binary** (yes or no), or **categorical** (cat, dog, moose, hedgehog, sloth, etc.). If it's categorical you want to remember your output value won't be just one variable, but several output variables.

1.3. Artificial Neural Networks:

What about **synapses**? Each of the synapses gets assigned weights, which are crucial to **Artificial Neural Networks** (ANNs). Weights are how ANNs learn. By adjusting the weights, the ANN decides to what extent signals get passed along. When you're training your network, you're deciding how the weights are adjusted.

What happens inside the **neuron**? First, all the values that it's getting are added up (the **weighted sum** is calculated). Next, it applies an activation function, which is a function that's applied to this neuron. From that, the neuron understands if it needs to pass along a signal or not.
This is repeated thousands or even hundreds of thousands of times!

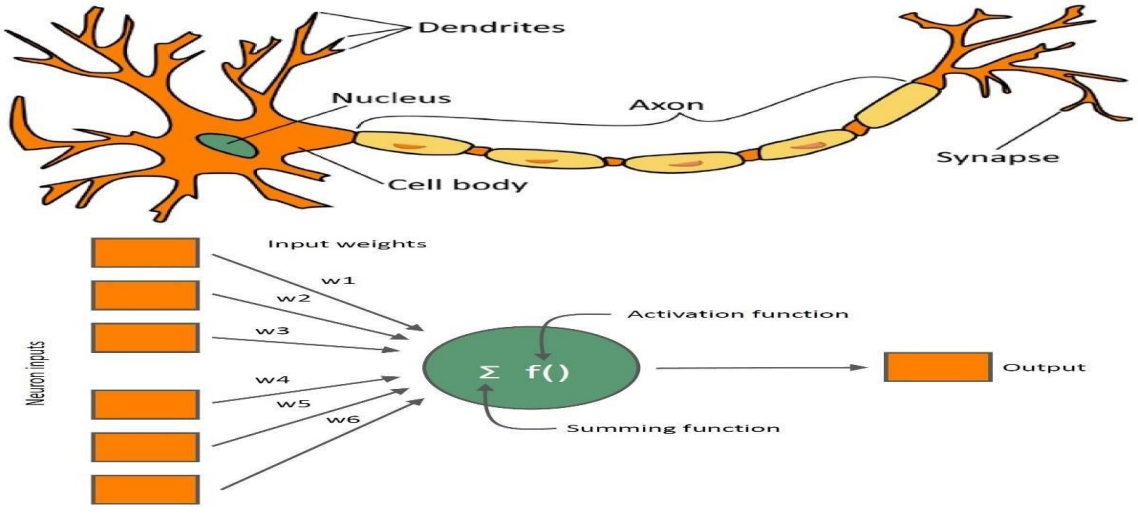


Figure 2 Biological neuron VS Artificial neuro

We create an artificial neural net where we have nodes for input values (what we already know/what we want to predict) and output values (our predictions) and in between those, we have a hidden layer (or layers) where the information travels before it hits the output. This is analogous to the way that the information you see through your eyes is filtered into your understanding, rather than being shot straight into your brain.

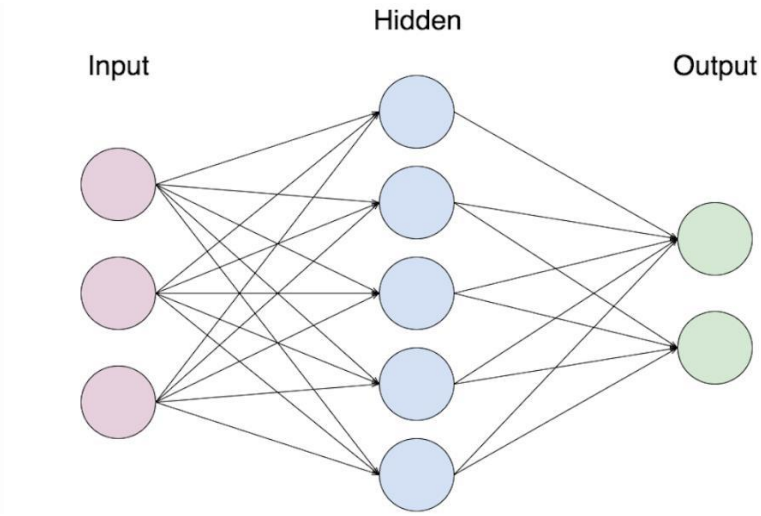


Figure 3 ANN

1.4. Convolution Neural Networks:

In fully connected network (FC) each node in each layer is connected to all nodes in the next layer. For images, if we want to use a fully connected neural network, we can consider each pixel of an image as an input node. So, number of nodes in the input is equal to no. of pixels in the image. we can manage to train the network for small images, but for larger images, the number of nodes of a fully connected network becomes too huge and so correspondingly the no. of parameters (weights and biases) becomes huge, and it becomes impossible to train those networks for good accuracy in reasonable time.

Also, in FC network the order of the input nodes doesn't matter which means that we can get the same result, if we change the order of all inputs. But, for images, there is correlation between input pixels. Pixels close to each other are highly correlated and pixels farther are less correlated. So, we can use this property of images to reduce the no. of connections between layers, by using the same filter, which is a small sliding window representing the learnable weights, to extract a certain property across the all parts of the image, and here comes the Convolution Neural Network.

In deep learning, a convolutional neural network (Conv-Net or CNN) is a kind of deep neural networks applied to visual imagery analysis. It is also known as shift invariant or space invariant artificial neural networks (SIANN). It is applied in computer vision in scene labeling, face recognition, action recognition, image and video classification. CNN is also used in the field of speech recognition and text classification for natural language processing (NLP).

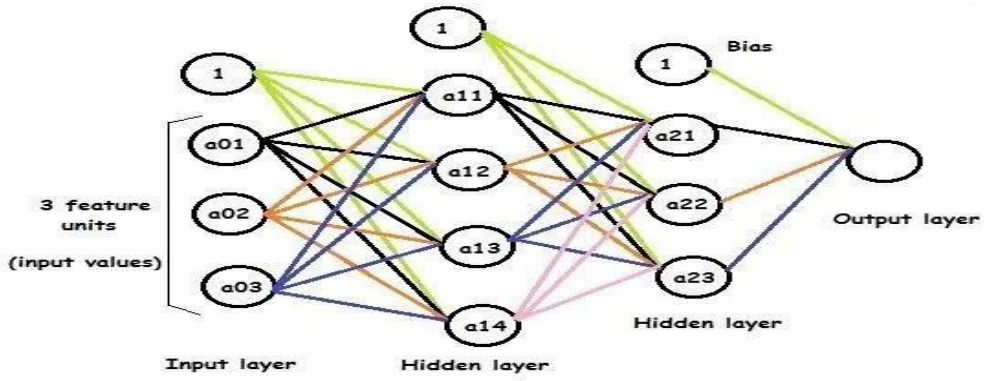


Figure 4 Feed Forward Neural Network with Fully connected layer

1.5. Convolution Neural Network Architecture:

Convolutional networks consist of multiple stages, each stage consists of multiple layers such as convolutional layer, pooling layer, non-linearity layer (activation layer), and fully connected layer at the end of the network. The output of each stage is fed as input to the next stage.

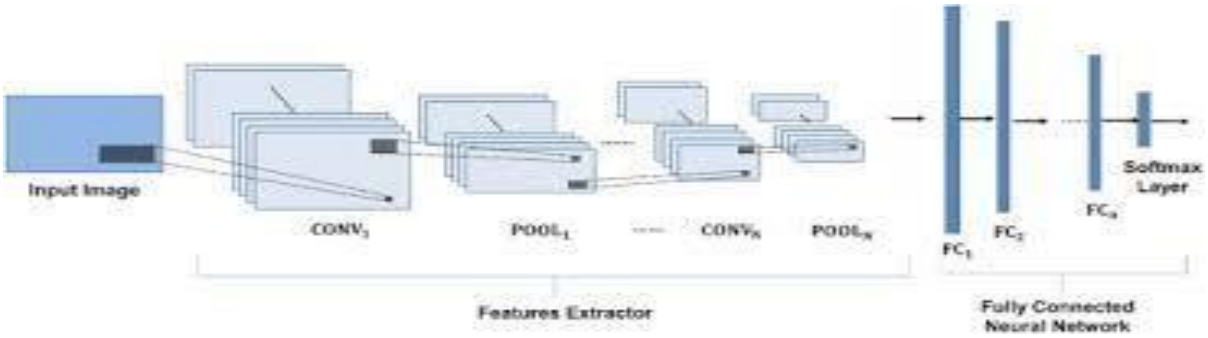


Figure 6 Convolutional Neural Network

2.5.1. Convolutional layer:

This layer is the core of the network. It consists of several filters (which are the learnable parameters, each filter consists of number of kernels) each convolves with the full depth of the input to extract a certain feature which is called a Feature Map. The input of each filter is a small set of neurons from the previous layer's output, this set is called Receptive Field.

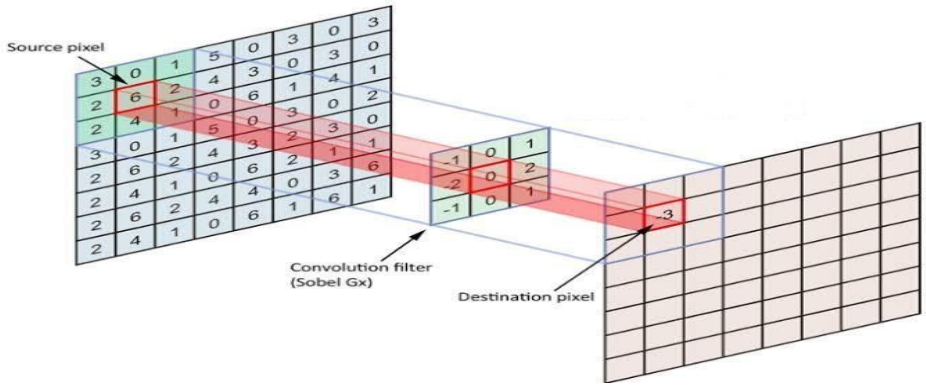


Figure 8 Convolution layer

There are three types of convolution: 1D, 2D, and 3D convolutions. The key difference between them is the dimensionality of the input and how the filter slides across the data.

In 1D convolution (Conv1D), kernel slides along one dimension. It is applied to Time-series data, when the input is in one-dimensional shape such as text or audio signals.

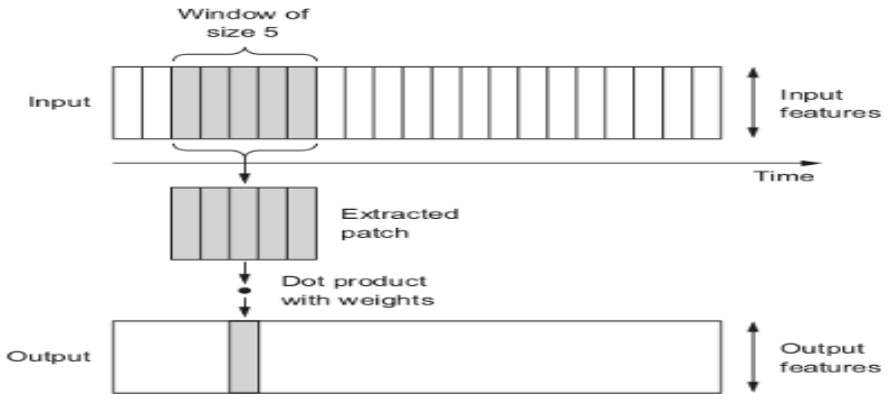


Figure 10 Conv1D

In 2D convolution (Conv2D), kernel slides along 2 dimensions across the data (the spatial dimensions x and y). It is used on image data. The depth of each filter is equal to the depth of the input, and each filter outputs a 2D matrix. It is used to extract the spatial information in images.

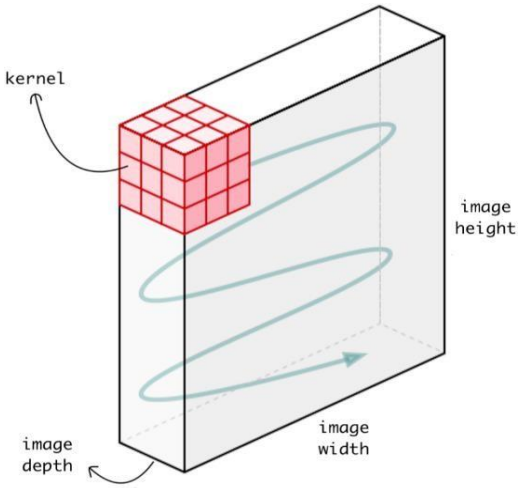


Figure 12 Conv2D

In 3D convolution (Conv3D), it is like Conv2D except the kernel slides along 3 dimensions (x, y, z) across the data. It is used on video data. The depth of each filter is smaller than the depth of the input, so each filter outputs a 3D volume. It takes time in account, so it is used to extract the spatiotemporal features.

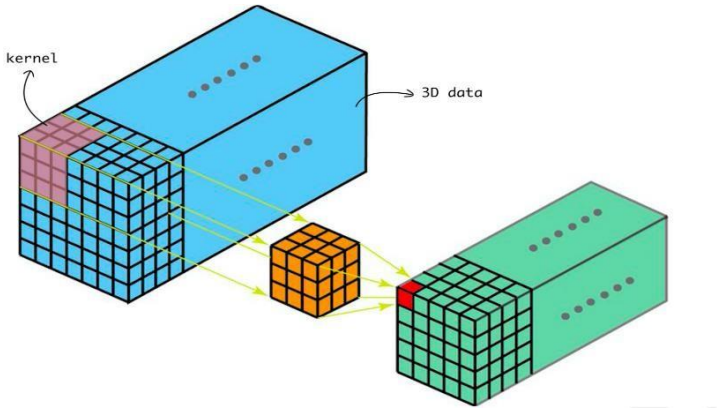


Figure 13 Conv3

2.5.2. Non-linearity layer:

Also, called activation or squashing layer. It decides whether a neuron should be activated or not. It limits the output of the convolution layer to a certain range to map it to the desired output.

There are several types of activation layers such as: rectified linear unit (RELU), leaky RELU, sigmoid, tanh.

2.5.3. Rectified linear unit (RELU):

It is the most widely used. One of its advantages is that it does not activate all the neurons at the same time because it maps all negative inputs to zero, so this makes the network more efficient, but one disadvantage is that the gradients of the negative regions are zero which will not allow these weights to be updated during backpropagation, this is called Dying RELU. To fix this issue, we use Leaky RELU.

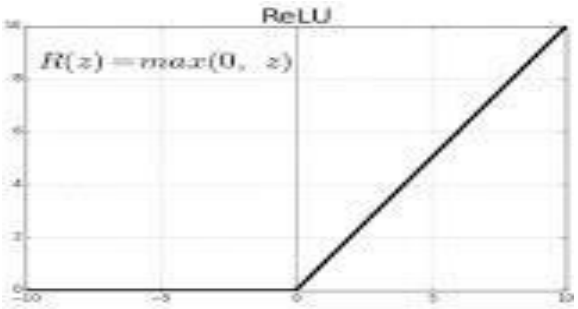
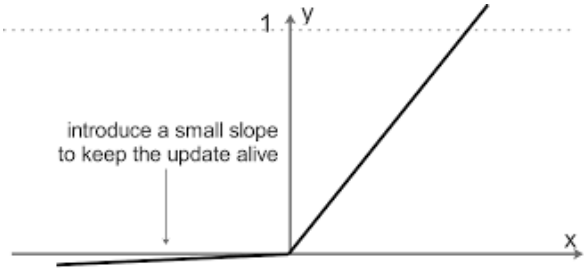


Figure 15 RELU activation function

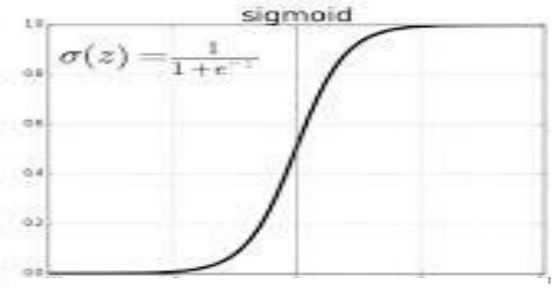
2.5.4. Leaky RELU:

Leaky RELU is like RELU, but it has a slight slope in the negative range, so it doesn't suffer from Dying RELU problem.



Sigmoid

Sigmoid output ranges from zero to one. It assigns probability to each class.



2.5.5. Tanh:

Tanh is like sigmoid activation function, but its output ranges from -1 to +1.

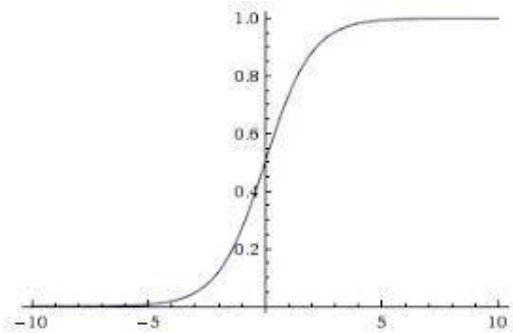


Figure 17 Tanh activation function

2.5.6. Pooling layer:

Pooling layer is used to reduce the spatial size (width and height) of its input to reduce the amount of weights and computations so the network works more efficient and avoids overfitting. Pooling layer operates on each feature map independently.

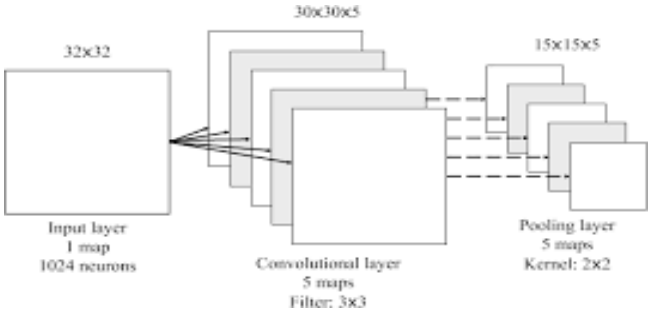


Figure 18 Convolution layer followed by pooling layer

There are types of pooling layers, local pooling and global pooling. Local pooling operates on a small chunk of its input and combines it into one cell. On the other hand, global pooling operates on the whole input at once. Also pooling may compute max value or average value. Max pooling substitutes the input by its maximum value, average pooling substitutes the input by its average value.

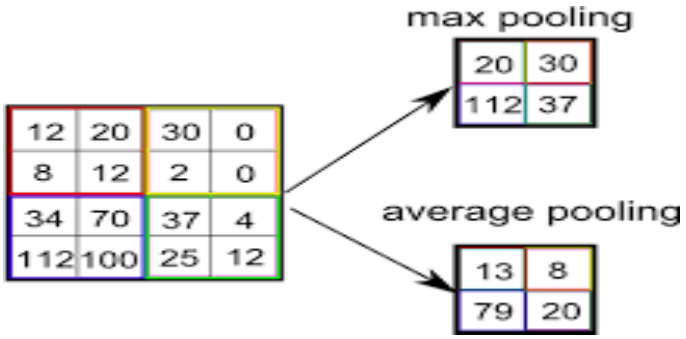


*Figure 19 Pooling with 2*2 filters and stride 2*

2.5.7. Fully connected layer (FC):

This is the last block in the convolutional network. FC layer(s) represents the feature vector for the input. It holds the vital information for the input; therefore, this feature vector is used for classification, regression, or as input for another network such as Recurrent Neural Network (RNN).

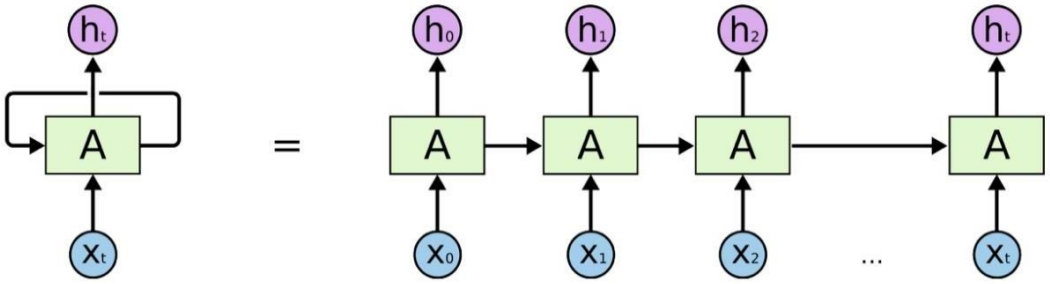
1.6. Recurrent Neural Networks (RNNs):

Recurrent Neural Network is a class of artificial neural networks where it is a generalization of feedforward neural network that has an internal memory. RNN can use this internal memory to process sequence of inputs.

RNN is recurrent in nature as it performs the same function for every input of data while the output of the current input depends on the past one computation. After producing the output, it is copied and sent back into the recurrent network. For deciding it considers the current input and the output that it has learned from the previous input. RNN can deal with any sequential data, including time series, video or audio sequence etc.

2.6.1. How do RNNs work?

RNN is a type of Neural Network where the output from the previous step is fed as input to the current step.



First, it takes the X (0) from the sequence of input and then it outputs h (0) which together with X (1) is the input for the next step. So, the h (0) and X (1) is the input for the next step. Similarly, h (1) from the next is the input with X (2) for the next step and so on. This way, it keeps remembering the context while training.

The formula for the current state is: ht = f (ht-1, xt)

Applying Activation Function: ht = tanh (Whhht-1 + Wxhxt)

W is weight, h is the single hidden vector, Wh is the weight at previous hidden state, Wxh is the weight at current input state, tanh is the Activation function, that implements a Non-linearity that squashes the activations to the range[-1,1].

Output: Yt = Whyht (Yt is the output state) [10]

2.6.2. Difference between CNN and RNN:

CNN is a feed forward neural network that is generally used for Image recognition and object classification. While RNN works on the principle of saving the output of a layer and feeding this back to the input in order to predict the output of the layer.

CNN considers only the current input while RNN considers the current input and the previously received inputs. It can memorize previous inputs due to its internal memory.

CNN has 4 layers: Convolution layer, RELU layer, Pooling and Fully Connected Layer. Every layer has its own functionality and performs feature extractions and finds out hidden patterns.

RNN has 4 types: One to One, One to Many, Many to One and Many to Many. RNN can handle sequential data while CNN cannot.

2.6.3. Problem with RNNs:

RNNs have the problem of vanishing gradients, so the earlier layers can't learn enough and suffer from short term memory. In the conventional feed-forward neural network, the weight updating that is applied on a layer is a multiple of the learning rate, the error term from the previous layer and the input to that layer. Thus, the error term for a layer is somewhere a product of all previous layers' errors. When dealing

with activation functions like the sigmoid function, the small values of its derivatives that occurring in the error function gets multiplied multiple times as we move towards the starting layers. As a result of this, the gradient almost vanishes as we move towards the starting layers, and it becomes difficult to train these layers.

RNNs can remember the characteristics of previous inputs and outputs. For certain cases, the immediate previous output may not just be enough to predict what is coming and the network may have to rely on information from a further previous output.

RNNs does not learn long range dependencies across time. When the gap between the relevant information and the point where it is needed is very large RNNs become unable to learn to connect the information. The idea that they might be able to connect previous information to the present task, such as using previous video frames might inform the understanding of the present frame.

1.7. Long Short-Term Memory (LSTM):

Long Short-Term Memory networks are a special kind of RNN, that are capable to learn long term dependencies. LSTM is well-suited to classify, process and predict time series given time lags of unknown duration. And required in complex problem domains like machine translation, speech recognition, and more. The vanishing gradient problem of RNN is resolved here[11].

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior[12] [12].

2.7.1. How do LSTMs work?

LSTM networks have some internal contextual state cells that act as long-term or short-term memory cells. The output of the LSTM network is modulated by the state of these cells.

LSTMs have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

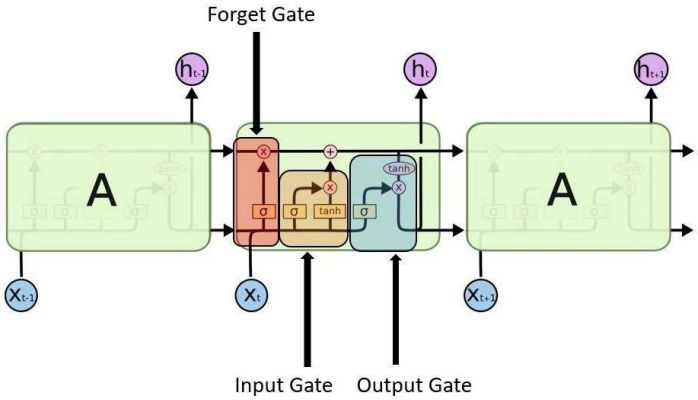


Figure 20 LSTM

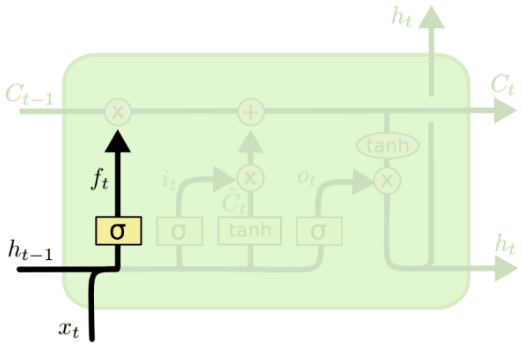
The information flows through a mechanism known as cell states. This way, LSTMs can selectively remember or forget things, The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It is very easy for information to just flow along it unchanged.

LSTM does have the ability to remove or add information to the cell state, by structures called gates.
Gates: are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.

Forget gate: is responsible for removing information from the cell state. The information that is no longer required for the LSTM to understand things or the information that is of less importance is removed via multiplication of a filter. This is required for optimizing the performance of the LSTM network.

This gate takes in two inputs: ht-1 and xt. ht-1 is the hidden state from the previous cell or the output of the previous cell and xt is the input at that time step. The given inputs are multiplied by the weight matrices and a bias is added. Following this, the sigmoid function is applied to this value. The sigmoid function outputs a vector, with values ranging from 0 to 1, corresponding to each number in the cell state Ct−1. The sigmoid function is responsible for deciding which values to keep and which to discard. If a 0 is output for a value in the cell state, it means that the forget gate wants

the cell state to forget that piece of information completely. Similarly, a 1 means that the forget gate wants to remember that entire piece of information. This vector output from the sigmoid function is multiplied to the cell state.

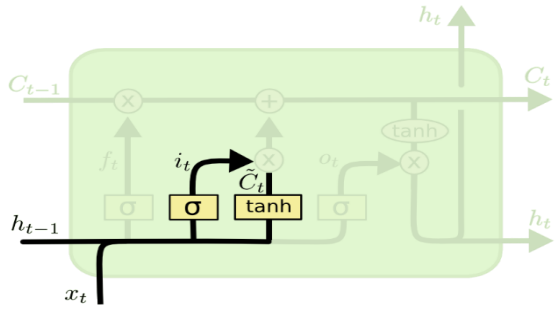


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figure 21 Forget Gate

Input gate:

The input gate is responsible for the addition of information to the cell state. We pass the previous hidden state ht-1 and current input xt into a sigmoid function. That decides which values will be updated by transforming the values to be between 0 and 1. 0 means not important, and 1 means important. Then pass the hidden state and current input into the tanh function to squish values between -1 and 1 to help regulate the network. Then multiply the tanh output with the sigmoid output. The sigmoid output will decide which information is important to keep from the tanh output[13].

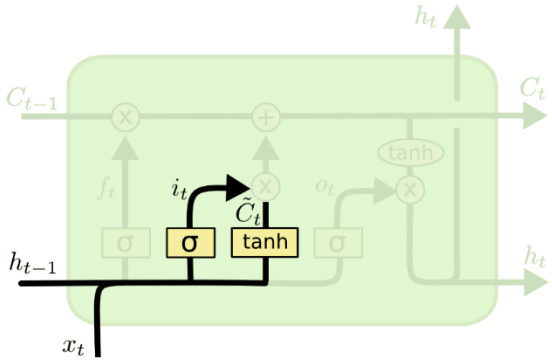


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figure 22 Input Gate

Cell state:

The cell state gets pointwise multiplied by the forget vector. This has a possibility of dropping values in the cell state if it gets multiplied by values near 0. Then we take the output from the input gate and do a pointwise addition which updates the cell state to new values.

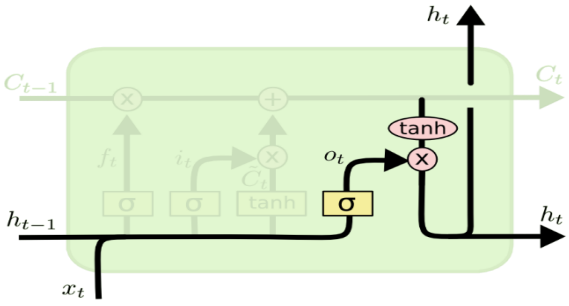


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Output gate:

Selecting useful information from the current cell state and showing it out as an output. we pass the previous hidden state ht-1 and the current input xt into a sigmoid function. Then we pass the newly modified cell state to the tanh function. We multiply the tanh output with the sigmoid output to decide what information the hidden state should carry. The output is the hidden state. The new cell state and the new hidden is then carried over to the next time step[13].



$$o_t = \sigma\left(W_o \; [\, h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 23 Output Gate

1.8. Grated Recurrent Unit (GRU):

Like LSTM the, GRU has gating units that modulate the flow of information inside the unit, however, without having a separate memory cell.
GRU aims to solve vanishing gradient problem.

Update gate: It determines how much of the past knowledge needs to be passed along into the future. It is like to the output gate in an LSTM.

Reset Gate: It determines how much of the past knowledge to forget. It is analogous to the combination of the input gate and the forget gate in an LSTM.

Current Memory Gate ($\overline{h}_t$): It is often overlooked during a typical discussion on Gated Recurrent Unit Network. It is incorporated into the Reset Gate just like the Input Modulation Gate is a sub-part of the Input Gate and is used to introduce some non-linearity into the input and to also make the input Zero-mean. Another reason to make it a sub-part of the Reset gate is to reduce the effect that previous information has on the current information that is being passed into the future[13].

1.9. Bidirectional LSTM:

Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on sequence classification problems. Bidirectional recurrent neural networks are just putting two independent RNNs together. This structure allows the networks to have both backward and forward information about the sequence at every time step.

Using bidirectional will run your inputs in two ways, one from past to future and one from future to past and what differs this approach from unidirectional is that in the LSTM that runs backward you preserve information from the future and using the two hidden states combined you are able in any point in time to preserve information from both past and future [10].

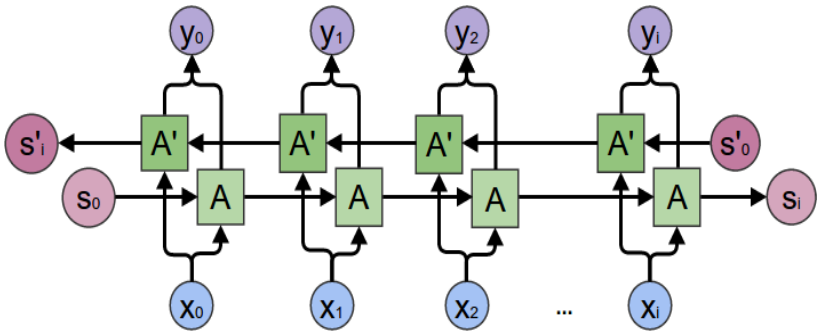


Figure 24 BI-LSTM

In problems where all timesteps of the input sequence are available, Bidirectional LSTMs train two instead of one LSTMs on the input sequence. The first on the input sequence as-is and the second on a reversed copy of the input sequence.

Chapter Three: Related Works

3. Related Works:

This chapter is divided into three sections. Firstly, there is an overview of lip-reading models over the last years. Secondly, the recent papers and methodologies are discussed. Thirdly, overview of the common issues found.

3.1. Overview:

Lip reading is a matter of research that many papers are concerned with. The common point in many research is how to track the lip movements and how each character of the word is pronounced in order to translate it into its corresponding text. Different models and neural networks were used in lip reading, some achieved high accuracy in knowing the exact text that was said and others didn't.

Lip reading problem was raised since 1984[14]. First researchers focused on locating the mouth region in an image of the entire face was done by locating and then tracking the nostrils from frame to frame. The nostrils were located, using region matching against a nostril template composed of the region parameter values for each nostril. The horizontal and vertical distance between the nostrils was also used to confirm the location of the nostril regions. Using a mouse! the nostril template was created semi automatically for each speaker by manually defining a bounding rectangle around them in a single facial image.

Two samples of each spoken alphabetic letter were used as a representative sample of mouth images for a speaker. An iterative clustering routine was then applied to the mouth images given a pre-set mouth image distance threshold to separate the clusters. The number of clusters was then compared to 255.

A successive approximation algorithm was then run to determine the distance which yielded a cluster count closest to but not greater than 255. For each iteration of the successive approximation, the clustering routine was re-run with the new distance threshold. Thus, the process terminated when nearly 255 clusters were created, and a representative from each cluster was entered in the mouth image codebook in preparation for vector quantization of visual speech templates (mouth image sequences).

A new utterance would be classified by finding the closest matching combined rank from the example set which includes images of utterance of each digit and character. Generally, the vector quantization of the visual speech templates to one of only 256 mouth images produced adequate performance and should facilitate the real-time implementation of the lipreading system. The faced issues were arbitrary lighting, range and viewing angle and lack of examples set in the dataset.

In 1991 another methodology was used which is optical flow analysis[15].The velocity of lip motions may be measured from optical flow data which allows muscle action to be estimated. Pauses in muscle action result in zero velocity of the flow and are used to locate word boundaries. The pattern of muscle action is used to recognize the spoken words. However, an overall accuracy of approximately 70% was achieved.

In 2005 another paper[16]focused on lip finding, and lip tracking using feature extraction approaches as ASM (active shape models) and DCT (discrete cosine transformation) and Hidden Markov Models (HMM) can be used to perform the recognition. The DCT outperforms the results obtained by ASM, as it does not require a precise contour localization, but only an approximation of the mouth region. The Word Error Rate using the DCT is 11.8% lower than using the ASM.

In 2010 another methodology was used which is Optical Flow and Support Vector Machine[17]. The reported technique analysis the video data of lip motions by computing the optical flow (OF). The statistical properties of the vertical OF component were used to form the feature vectors for training the support vector machines (SVM) classifier. It succeeded in achieving accuracy of 95%.

Lately many research are directed to applying different neural networks beside CNN.

3.2. Recent Papers:

In 2022:

The proposed Arabic visual speech recognition model is capable of classifying digits and phrases in the Arabic language by examining our collected visual datasets. The keyframe extraction technique and CFIs were utilized to perform the concatenated stretch image, which represents the sequence of the input video. Different approaches were used in the experiments, and the performance has been validated on our collected dataset. The results show that the CNN network with VGG19 networks to extract the bottleneck features by employing the batch normalization provides a high accuracy by stabilizing the training process, as compared to state-of-the-art methods where the input of the models is concatenated frame images. The performance of our model has yielded the best test accuracy of 94% for digit recognition, 97% for phrase recognition, and 93% in the experiments of digit and phrase recognition. Furthermore, we intend to focus on examining the model by working with different locations of the lip landmark localization by increasing the dataset to provide more accurate results in the field of visual speech recognition.

In 2016:

The authors in [18] introduced CNN with LSTM on sentences with and without voice.

The model is based on the recent sequence to-sequence (encoder-decoder with attention) translator architectures that have been developed for speech recognition and machine translation.it outputs at the character level, can learn a language model, and has a novel dual attention mechanism that can operate over visual input only, audio input only, or both.

The model consists of three key components:

1. The image encoder ‘Watch’.
2. The audio encoder ‘Listens’.
3. The character decoder ‘Spell’.

Each encoder transforms the respective input sequence into a fixed dimensional state vector, and sequences of encoder outputs.

- Image encoder ’Watch’:

  The image encoder consists of the convolutional module that generates image features for every input timestep, and the recurrent module that produces the fixed dimensional state vector sv and a set of output vectors ov.

  The convolutional network is based on the VGG-M model, as it is memory-efficient, fast to train and has a decent classification performance on ImageNet.

- Audio encoder ‘Listen’:

  The Listen module is an LSTM encoder like the Watch module, without the convolutional part. The LSTM directly ingests 13-dimensional MFCC features in reverse time order and produces the state vector and the output vectors.

- Character decoder ‘Spell’:

  The Spell module is based on a LSTM transducer, here They added a dual attention mechanism. At every output step k, the decoder LSTM produces the decoder states and output vectors from the previous step context vectors. They used two independent attention mechanisms for the lip and the audio input streams to refer to the asynchronous inputs with different sampling rates. The probability distribution of the output character is generated by an MLP with SoftMax over the output.

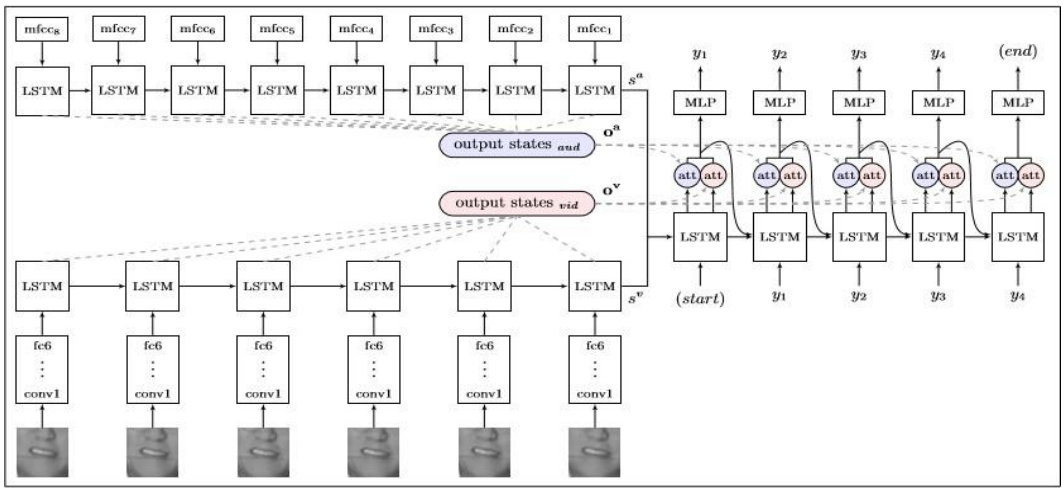


Figure 25 Model’s architecture

For Watch and Listen modules, they used a two-layer LSTM with a cell size of 256. For the Spell module, they used a two-layer LSTM with cell size of 512. The output size of the network is 45.

A new strategy where they started training only on single word examples, and then let the sequence length grow as the network trains. These short sequences are parts of the longer sentences in the dataset. They observed that the rate of convergence on the training set is several times faster, and it also significantly reduces overfitting, presumably because it works as a natural way of augmenting the data. The test performance improved a lot.

Training on the full sentence data was stopped when the validation error did not improve for 5,000 iterations. The model was trained for around 500,000 iterations, which took approximately 10 days. In conclusion, they reported a strong performance of 3.3%(WER), reports a word error rate of 6.6% and the WAS model achieves 5.8%, and approximately accuracy of 96.7% on Grid dataset and 84.5% on LRW dataset.

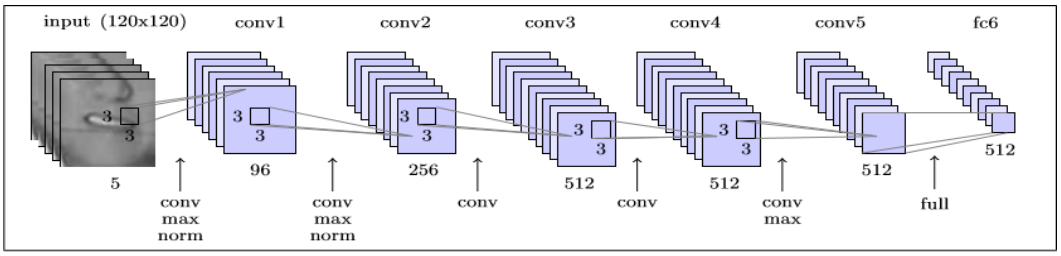


Figure 26 Model's Overview

Also, in 2016 another similar approach was introduced in [19]

LipNet is a neural network architecture for lipreading that maps variable-length sequences of video frames to text sequences.

The model operates at the character-level, using spatiotemporal convolutional neural networks (STCNNs), recurrent neural networks (RNNs), and the connectionist temporal classification loss (CTC).

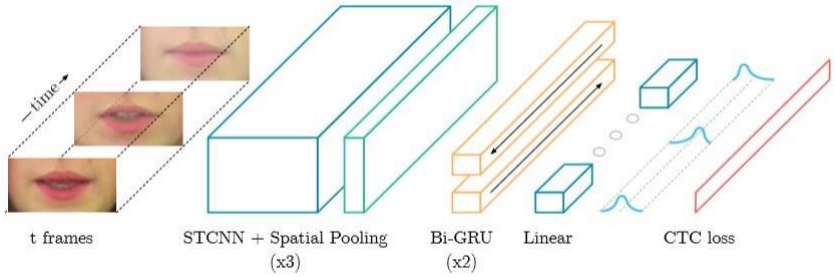


Figure 27 Lip-net model architecture

LipNet architecture is a sequence of T frames is used as input and is processed by 3 layers of STCNN (spatiotemporal convolutions, channel-wise dropout, spatial max-pooling), each followed by a spatial max-pooling layer. The features extracted are processed by 2 Bi-GRUs; each time-step of the GRU output is processed by a linear layer and a SoftMax. This end-to-end model is trained with CTC. All layers use rectified linear unit (ReLU) activation functions.

To measure the performance of LipNet and the baselines, they computed the word error rate (WER) and the character error rate (CER)

LipNet achieved on unseen Speakers 6.4% CER, 11.4%WER, and on overlapped Speakers 1.9% CER and 4.8 % WER. In conclusion, LipNet attains a 95.2% sentence-level word accuracy, and it can generalize across unseen speakers in the GRID corpus with an accuracy of 88.6%.

In 2017:

The authors in[20] introduced the idea of Combining Residual Networks with LSTMs for Lipreading, The system is a combination of spatiotemporal convolutional, residual and bidirectional Long Short-Term Memory networks.

It combined three sub-networks:

- The front-end, which applies spatiotemporal convolution to the frame sequence.
- Residual Network (ResNet) that is applied to each time step.
- The backend, which is a two-layer Bidirectional Long Short-Term Memory (Bi-LSTM) network.

The SoftMax layer is applied to all timesteps and the overall loss is the aggregation of the per time step losses, and the system is trained in an end-to-end fashion.

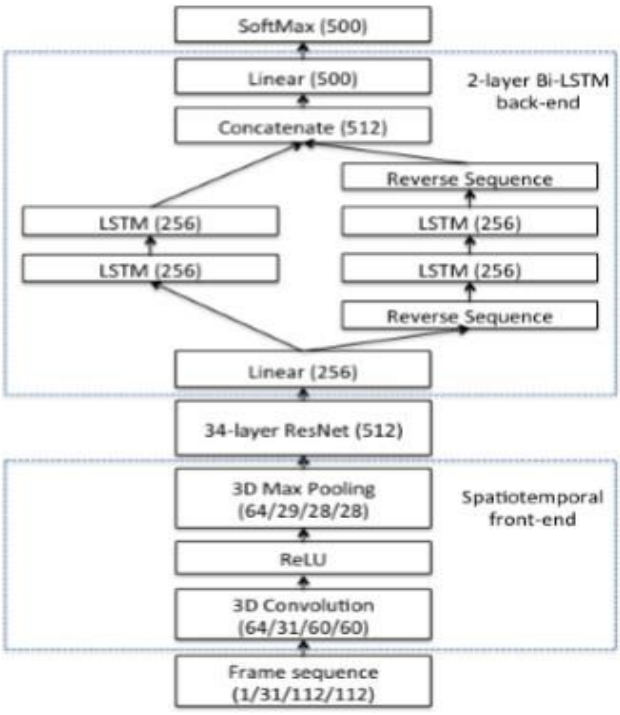


Figure 28 Combining Residual Networks with LSTMs model architecture

- Spatiotemporal front-end:
  The first set of layers applies spatiotemporal convolution to the preprocessed frame stream. Spatiotemporal convolutional layers can capture the short-term dynamics of the mouth region and are proven to be advantageous, even when recurrent networks are deployed for back end.

  They consist of a convolutional layer with 643-dimensional(3D) kernels of 5×7×7 size (time/width/height), followed by Batch Normalization and Rectified Linear Units (RELU). The extracted feature maps are passed through a spatiotemporal max-pooling layer, which drops the spatial size of the 3D feature maps.

- Residual Network:
  The 3D features maps are passed through a residual network (ResNet, [30]), one per time-step. They used the 34-layer identity mapping version, which was proposed for ImageNet.

  The ResNet drops progressively the spatial dimensionality with max pooling layers, until its output becomes a single dimensional tensor per time step.

- Bidirectional LSTM back-end:
  The back end of the model is a Bidirectional LSTM network. For each of the two directions, they stacked two LSTMs, and the outputs of the final LSTMs are concatenated.

The proposed network attains word accuracy equal to 83.0%, yielding 6.8% absolute improvement over the current state-of-the-art, without using information about word boundaries during training or testing.

In 2018:

A comparison of models on deep lip reading [21] is made between three models:

- Recurrent model using LSTMs.
- Fully convolutional model.
- The recently proposed transformer model.

The recurrent and fully convolutional models are trained with a Connectionist Temporal Classification loss and use an explicit language model for decoding, the transformer is a sequence-to-sequence model.

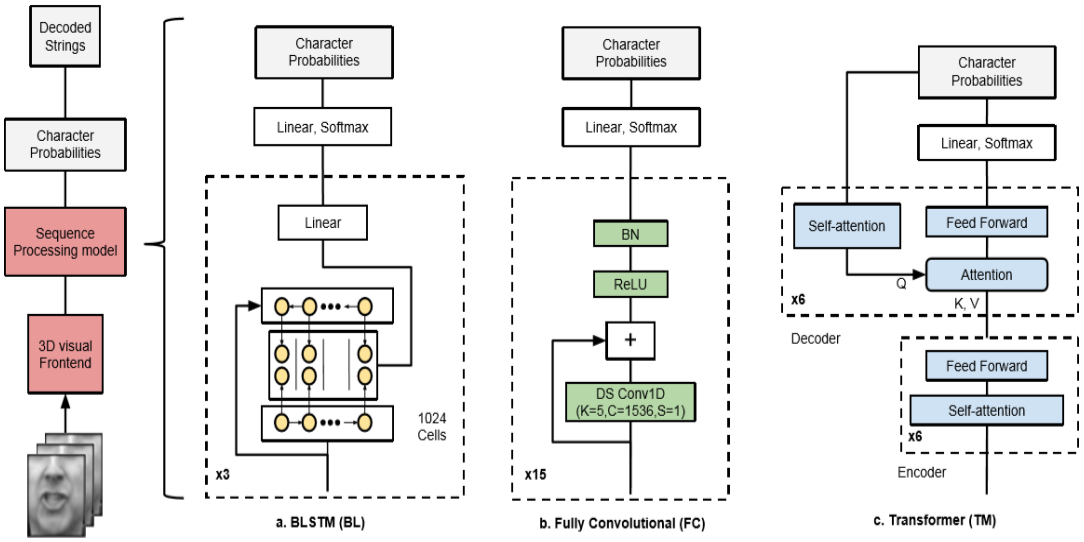


Figure 29 comparison of models

1. Bidirectional LSTM(BL):

It consists of three stacked bidirectional LSTM (BLSTM) recurrent layers. The first BLSTM layer ingests the vision feature vectors, and the final BLSTM layer emits a character probability for every input frame. The network is trained with CTC. The output alphabet is there for augmented with the CTC blank character.

2. Fully Convolutional (FC):

The network consists of several temporal convolutional layers. They used depth-wise separable convolution layers, that consist of a separate convolution along the time dimension for every channel, followed by projection along the channel dimensions (a position-wise convolution with filter width.

After each convolution they added a shortcut connection, followed by Batch Normalization, and RELU.

The FC network is also trained with a CTC loss, with sequences decoded by using a beam search that incorporates the external language model (above). They considered two variants: one with 10 convolutional layers (FC-10), and a deeper one with 15 convolutional layers (FC-15).

3. Transformer model (TM):

The Transformer model has an encoder-decoder structure with multi-head attention layers used as building blocks. The encoder is a stack of self-attention layers, where the input tensor serves as the attention queries, keys and values at the same time. Every decoder layer attends on the embeddings produced by the encoder using common soft attention. The decoder produces character probabilities which are directly matched to the ground truth labels and trained with a cross-entropy loss.

They obtained worse performance with BL compared to FC-10 with accuracy 37%, even though the recurrent model has full context on every decoding timestep compared to the convolutional that only looks at a limited time-window of the input. While FC model obtained 45% and TM model obtained 50% accuracy.

In 2019:

Authors [22] of this work also presents analysis of two different approaches for lipreading on this architecture.

The first approach, 3D-2D-CNN-BLSTM network is trained with CTC loss on characters (ch-CTC). Then BLSTM-HMM model is trained on bottleneck lip features (extracted from 3D-2DCNN-BLSTM ch-CTC network) in a traditional ASR training pipeline.

The second approach, same 3D-2D-CNN-BLSTM network is trained with CTC loss on word labels (w-CTC). This network contains two 3D CNN layers, two 2D CNN layers followed by two BLSTM layers. In 3D convolution kernel moves along time, height and width dimensions of input. Whereas in 2D convolution kernel only moves along height and width dimensions.

Every 3D CNN and 2D CNN layer is followed by batch norm layer. They didn't use any pooling layer after 2D CNN layer. There is 3D MaxPool layer after every 3D convolutional layer. The output of 2D-Conv2 is linearized and fed to two bidirectional LSTM layers. SoftMax activation is applied on final CTC labels to get probabilities.

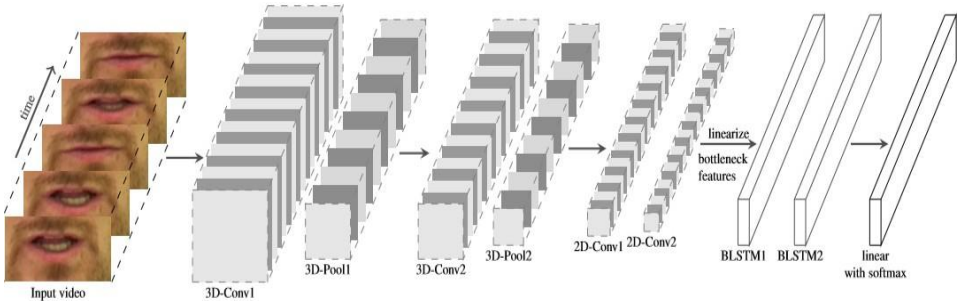


Figure 30 3D-2D-CNN BLSTM-HMM

The first approach CTC loss is computed on sequence of character labels and in second approach they computed CTC loss on sequence of word labels.

Also, Curriculum learning is applied which has been shown to provide better performance and faster convergence. In curriculum learning, the strategy is to train the network with segmented words for initial epochs, then in later epochs the network is trained on full sentences.

It took 45 epochs to converge with curriculum learning whereas without curriculum learning it took 89 epochs to converge.

BLSTM-HMM Hybrid Model:

In this approach, the 48-dimensional bottleneck features are extracted for each input frame from 3D-2D-CNN-BLSTM network trained with CTC loss function. Then each feature is duplicated four times. Thus, input sequence of length l is converted to a sequence of length $4 \times l$. Following the standard hybrid model training pipeline, a GMM-HMM model is trained on these features for context-independent phones (mono-phone GMM-HMM model).

In conclusion an accuracy of 98.7% is achieved regarding the word's classification.

Also, in 2019 another method was introduced in [23]for Lipreading.

This work shows replacing the shallow 3D CNNs + deep 2D CNNs front-end with recent successful deep 3D CNNs — two-stream (i.e., grayscale video and optical flow streams) I3D as The experiments show that, compared to the shallow 3D CNNs + deep 2DCNNs front-end, the deep 3D CNNs front-end with pre-training on the large-scale image and video datasets (e.g., ImageNet and Kinetics) can improve the classification accuracy for visual lipreading.

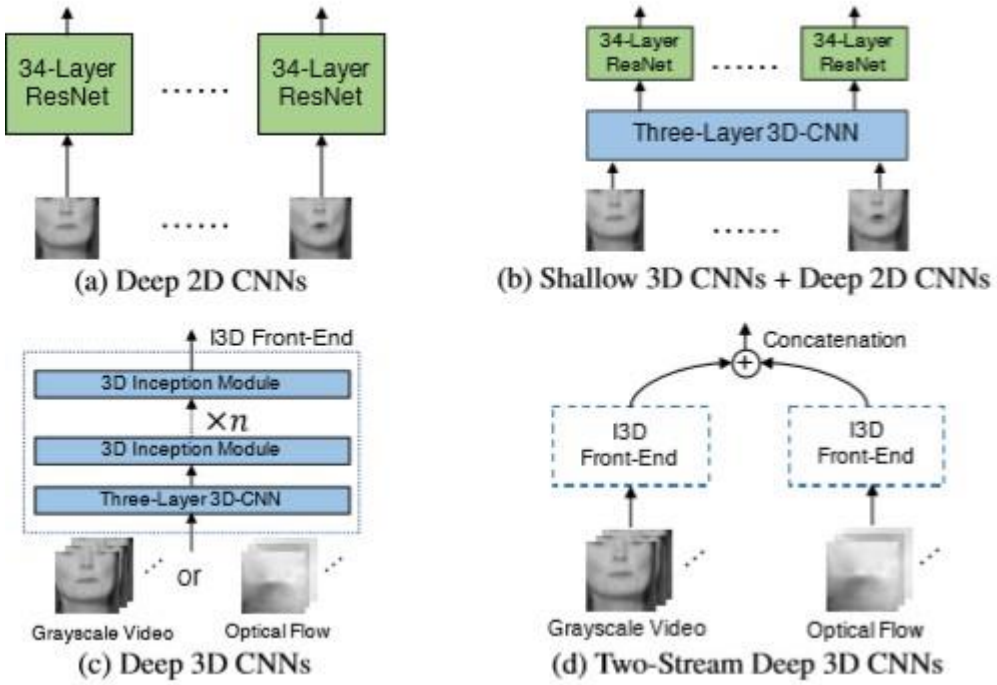


Figure 31 Two-Stream Deep 3D CNNs

- Deep 2D CNNs:
  Although the features extracted from the deep 2D CNNs front-end only contain the image-based information, the following back-end module such as a recurrent neural network can model the temporal dependency to some extent.

- Shallow 3D CNNs + Deep 2D CNNs:

It is commonly known that the 3D convolution can capture the short-term dynamics and is proven to be advantageous in visual lipreading even when the recurrent networks are deployed for the backend. However, due to the difficulty of training a huge number of parameters introduced by the three-dimensional kernels. They considered this shallow 3D CNNs + deep 2D CNNs as another strong baseline front-end module.

They pass the grayscale video through three 3D convolutional layers, each followed by the batch normalization, Rectified Linear Units (RELU) and 3D max-pooling layer. The extracted features are then processed by the 34-layer ResNet on each channel of features.

- Deep 3D CNNs:

  Although the shallow 3D CNNs are proven to be helpful, the effect of only using three-layer 3D convolutional layers might not hit the maximum, it is natural to use the deep 3D CNNs alone to replace the shallow 3D CNNs + deep 2D CNNs front-end.

- Two-Stream Deep 3D CNNs:

  Deep 3D CNNs are expected to extract strong enough spatial-temporal features from the gray-scale video inputs. In practice, fusing an additional optical flow stream with the grayscale video stream is proven to be useful in many video tasks, as the optical flow can explicitly capture the motion of pixels in adjacent images.

  They then use another I3D front-end (employing the same architecture as the I3D front-end in grayscale video stream but not sharing weights) to extract the features from the optical flow and concatenate with the features extracted from the grayscale video stream.

In conclusion it is shown that:

- when the shallow 3D CNNs + deep 2D CNNs front-end in is replaced with the pre-trained I3D front-end, the performance is improved from 78.77% to 81.52% on the test set, meaning that the deep 3D CNNs front-end with the two-round pretraining is more powerful than the shallow 3D CNNs + deep 2D CNNs front-end.

- To demonstrate if the optical flow input is effective, they replaced the grayscale input with the precomputed optical flow. It is found that either the optical flow or the grayscale video alone can predict the spoken word.

- When comparing the two-stream network with its single-stream counterpart (i.e., either the grayscale stream or the optical flow stream), the performance is clearly improved no matter the front-end is the shallow 3D CNNs + deep 2D CNNs or the pre-trained I3D. This confirms that the two-stream network is beneficial for lipreading. Overall, our two-stream pre-trained I3D+Bi-LSTM achieves an absolute improvement of 5.3% over the previous art on the LRW dataset.

Table 2 Table 2 related word and papers results

Paper	Model	Dataset	Accuracy
Lip Reading sentencesin the wild	CNN LSTM	GRID LRS LRW	Grid: 96.7% LRW: 84.5%
Lip net: end-to-endsentence-level lipreading	ST CNN GRU	Grid	95.2%
Combining Residual Networks with LSTMsfor Lipreading	3D Resnet BLSTM	LRW	83%
Deep Lip Reading: a comparison of models and an online application	1- BLSTM 2- FC 3- Transformer model	LRS2	1- 37% 2- 45% 3- 50%
Lipreading with 3D-2D-CNN, BLSTM-HMM and word-CTC models	3D CNN BLST M	Grid	98.7%
Learning Saito - Temporal Features with Two-Stream Deep3D CNNs for Lipreading	3D CNN BLST M	LRW	84%
Deep Learning-Based Approach for Arabic Visual Speech Recognition	VGG-19 (not include the top) + batch normalization layer	Private dataset	83% Classification problem

3.3. Conclusion:

At the end of this chapter and because of our research we concluded some mainways to deal with the idea of lipreading.

Problem Ideas:

1- Multi view representation: This is done by taking shots of the same person saying the same word but in different viewpoints, this maybe not be the best idea as there is no suitable and large enough dataset.

2- Audio and visual representation: in this way model relay on both audio and visual features of the word, that's a better way but it's not sufficient because there are many issues either it is the noisy environment or the problem of multi-talker speech.

3- Visual representation: in this way the model relay on the visual features only to learn the movements of each word and we preferred this way also it's challenging but it meets our objectives and we found that from research that the visual features obtained from the lip movements are more effective. As in [15] it states that the mouth movements provide important cues in speech recognition when the audio signal is noisy; and also give an improvement in performance even when the audio signal is clean, the character error rate is reduced from 16.2% for audio only to 13.3% for audio together with lips [15].

Visual models also have more than one way to deal with:

1- character level: model's input and outputs are in the form of characters, but we found that it's not the best way as characters differs its tone depending on its position in the word.

2- sentence level: model's input are words but in testing it build a full sentence we preferred this method as it meets our objectives and can be applied in real life applications.

Faced Issues:

After conducting research, we have come to the realization that lipreading is a challenging problem due to the variations in people's speech patterns, accents, and speaking speeds. These factors can potentially confuse the model when attempting to accurately interpret lip movements. However, we have made efforts to address this issue by providing the model with diverse and abundant training data, aiming to enhance its robustness against different speech characteristics.

Another significant challenge in lipreading is handling the varying lengths of words. Each word consists of a different number of frames, making it necessary to standardize the input size for training the model. In the upcoming chapter, we will delve into this issue in greater detail and explore potential solutions.

Chapter Four: Implementation & Methodologies

4. Implementation & Methodologies:

Within this chapter, we provide a concise overview of the datasets utilized, the preprocessing steps undertaken, the implemented models, and the tools employed in our study. Additionally, we outline the notable achievements made in the development of the web application.

4.1. Datasets:

In our project we used two different datasets, the first one was our Arabic dataset, and thesecond one was GRID [25].

4.1.1. Our Arabic Dataset:

Initially, our search for comprehensive and extensive datasets yielded limited results, with only a single dataset available. Unfortunately, this dataset was relatively small and encompassed videos of only 14 individuals uttering numbers from 0 to 9, along with four short sentences: "السلام عليكم" "اتصل بالشرطة" " اتصل بالاسعاف," and "احتاج للمساعدة." However, due to the limited size of this dataset, it was deemed insufficient to form the foundation of our project.

Figure 32 Small Arabic Dataset

However, given the insufficient nature of the available dataset, we made the decision to generate our own data following a similar style as the small dataset. In doing so, we carefully selected 24 significant words commonly utilized by Arab citizens in various everyday situations. These words were chosen based on their relevance and frequency:

Table 3 Arabic sentence in our dataset

قائمة الطعام	صداع مزمن	وجدت حقيبة	كيف حالك
تفضل الطلب	ضغط عالي	سرقة حقيبة	صباح الخير
أين المحاضرة	أريد قهوة	رخصة قيادة	مساء الخير
موعد المحاضرة	اريد الحساب	مما تعاني	أين المترو
مبنى الامتحان	اريد الفاتورة	سرطان الحنجرة	كيف اساعدك
موعد النتيجة	اريد المدير	ضعف سمعي	تقديم بلاغ

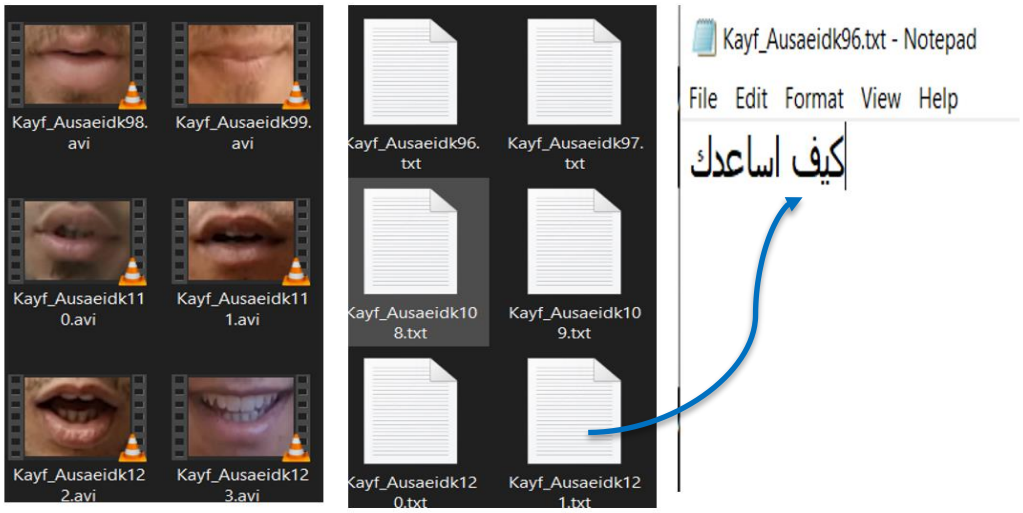


Figure 33 Our Private Arabic Dataset

So, we have dataset which contains from ten digits (0,1,2….,9) and 28 sentences, in total 4400 videos and aligns files.

4.1.2. GRID Dataset [25]:

The second dataset is GRID which contains 1000 sentences spoken by 34 talkers (18 males, 16 females). Each sentence consists of a six-word sequence of the form command, color, preposition, letter, digit and adverb, e.g. “Place green at H 7 now”. A total of 51 different words and 165k word instances are contained in GRID. Videos have a fixed duration of 3 seconds at a frame rate of 25 FPS with 720×576 resolution, resulting in sequences comprising 75 frames.

Each video has a text file with the spoken word and the duration for each word. Figure 4.2 shows a sample of the dataset.

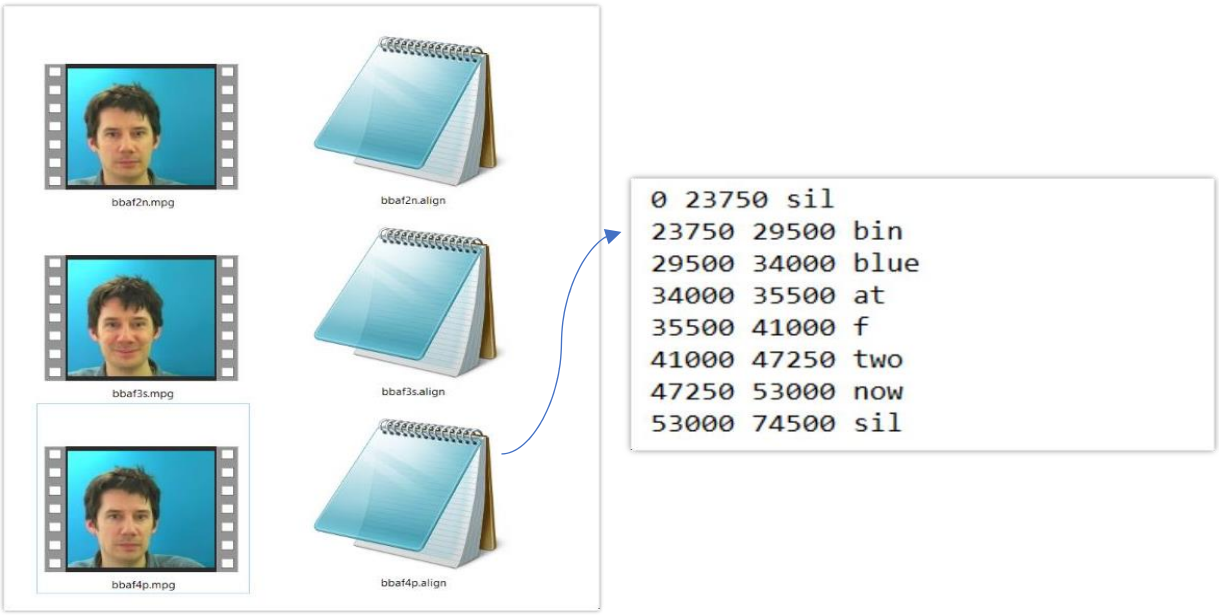


Figure 4.3. Grid dataset sample

4.2. Preprocessing:

The preprocessing stage for Arabic lipreading presented greater challenges compared to English. In Arabic videos, the individuals speaking were often in motion, leading to additional complexities. Additionally, the background in these videos was not stationary, resulting in a blurry environment. These factors made it more difficult to accurately extract and analyze the lip movements for Arabic speech.

Conversely, the preprocessing for English lipreading was relatively easier due to several factors. The background in the English videos was fixed, providing a stable environment for lip movement analysis. Moreover, the individuals speaking in the English videos were relatively static, minimizing the challenges associated with motion blur.

Overall, the differences in video characteristics and the variability in the environments for Arabic and English lipreading necessitated tailored preprocessing approaches to account for the specific challenges posed by each language.

4.3. Labeling:

Following a similar pattern as the English data, each video in the Arabic dataset was accompanied by an align file. This aligns file contained the corresponding text spoken by the individual in the video. Leveraging this pattern, we generated align files for each Arabic video, containing the transcribed Arabic text that was spoken in the respective video.

This alignment process allowed us to establish a connection between the visual cues of the lip movements in the videos and the corresponding textual representation of the spoken words. By aligning the lip movements with the accompanying Arabic text, we facilitated the training and evaluation of the lipreading models for accurate interpretation of Arabic speech.

4.4. Mouth Extraction:

4.4.1. Arabic Mouth extraction

Initially, we experimented with Harkescad for lip extraction, but unfortunately, it did not yield satisfactory accuracy for our lipreading project. As an alternative, we implemented the Landmarks algorithm, which proved to be more successful in achieving accurate lip extraction. The following steps were involved in the process:

1. Input: A video containing the speaker's face.
2. Frame Extraction: The video was converted into individual frames to facilitate lip extraction on a frame-by-frame basis.
3. Lip Extraction: Using the Landmarks algorithm, the specific region of the lips was identified and isolated from each frame. This algorithm relies on identifying key facial landmarks associated with the lips, such as the corners of the mouth, to accurately extract the lip region.
4. Output: The extracted lip regions were then combined to form a new video that solely focused on the lips, removing any extraneous facial features.

By employing the Landmarks algorithm, we achieved improved accuracy in isolating the lips from the speaker's face. This enhanced lip extraction process laid the foundation for subsequent analysis and interpretation of lip movements in our lipreading system

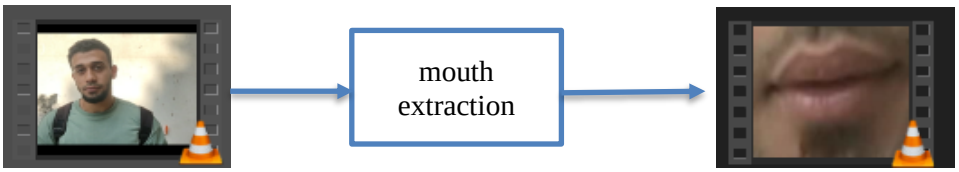


Figure 34 Mouth Extraction

4.4.2. English Mouth extraction

In contrast to the Arabic data preprocessing, the English data presented a comparatively easier task. This was primarily due to the static nature of the speaker in the videos, resulting in minimal changes in the lip area throughout the frames. As a result, we adopted a simpler approach for English preprocessing, specifically utilizing slicing techniques to enhance processing speed.

The process involved dividing each frame of the English videos into smaller segments or slices, focusing solely on the region encompassing the lips. Since the speaker's face remained relatively stationary, this slicing technique enabled us to quickly extract the lip area without the need for complex algorithms or extensive computational resources.

By implementing this streamlined approach, we were able to expedite the preprocessing stage for English lipreading data. This allowed us to allocate more time and resources towards subsequent analysis and modeling, ultimately contributing to the overall efficiency and effectiveness of our lipreading system for English speech.

4.5. Handling video's length:

When dealing with the size of the videos in our lipreading project, we encountered different scenarios for Arabic and English data. In the case of Arabic videos, the sizes varied, posing a challenge for consistent processing. On the other hand, the English videos had a fixed size, simplifying the preprocessing task.

To address the varying sizes of Arabic videos, we explored two approaches: padding and duplication. The padding approach involved adding extra frames or extending the duration of the video to match the length of the longest video in the dataset. This ensured uniformity in size but didn't always yield optimal results.

Alternatively, we experimented with the duplication approach, where we replicated frames within the video to match the length of the longest video. This technique proved to be more effective in maintaining the natural flow and rhythm of the lip movements, resulting in better overall performance.

By selecting the duplication approach for Arabic videos, we achieved improved accuracy and preserved the integrity of the lipreading process. This technique ensured that all videos, regardless of their original size, were transformed into fixed-sized inputs suitable for further analysis and modeling.

4.6. Implemented models:

This section discusses the two implemented models and their architectures. The first model is Lip-net and the second one is Sentences in the wild.

4.3.1. Main Lip-net Model Architecture [19]:

The lipreading model architecture begins with three blocks, each comprising a spatiotemporal convolution layer, a batch normalization layer, a dropout layer, and a spatial max pooling layer. These blocks are designed to extract meaningful features from the input data. The extracted features are then passed through two bidirectional gated recurrent unit (Bi-GRU) layers, which capture the temporal dependencies in the sequence of features.

Following the Bi-GRU layers, a linear transformation is applied to further refine the learned representations. This transformation prepares the features for the final classification step. The output of the linear transformation is passed through a SoftMax activation function, which assigns probabilities to each class label.

To introduce non-linearity and enhance the model's expressive power, the activation function used in all layers is the rectified linear unit (ReLU). This activation function has been widely adopted in deep learning models for its ability to efficiently model complex relationships between features.

The implementation of this lipreading model is carried out using the Keras deep learning library, which provides a high-level API for building and training neural networks. Keras simplifies the development process, allowing for efficient prototyping and experimentation with different architectures and configurations.

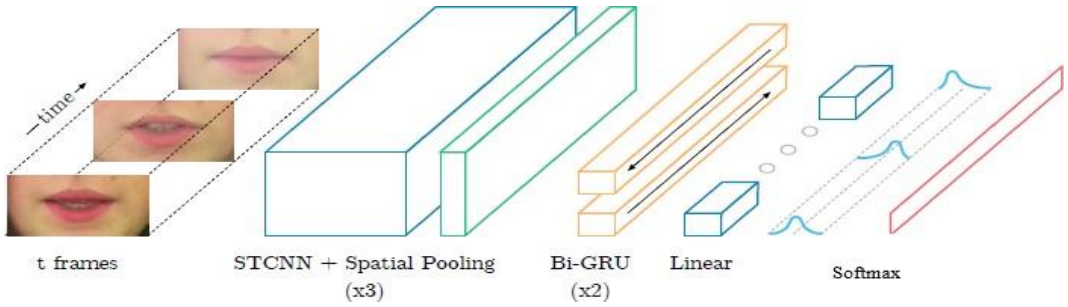


Figure 35 Main Lip-Reading model

4.6.1. Lip-net model for Arabic:

This model combines convolutional and recurrent layers to extract spatiotemporal features from lipreading data. It uses bidirectional GRU layers to capture temporal dependencies and a dense layer with SoftMax activation for classification. The model is trained to predict the class labels associated with lip movements, based on the input lipreading sequences.

It's important to note that the provided code snippet only describes the model architecture and does not include the details of data preprocessing, training, or evaluation.

The given code represents the architecture of a 3D convolutional neural network (CNN) followed by recurrent neural network (RNN) layers for sequence processing. Here's a summary of the model:

1- Input: The model expects input data in the shape of (60, 150, 160, 1), representing a 5D tensor with dimensions (batch_size, time_steps, height, width, channels). The data type is set to float32.

2- Convolutional Layers:

- ZeroPadding3D: Adds zero padding to the input tensor to preserve spatial dimensions. Padding of (1, 2, 2) is applied.
- Conv3D: Performs a 3D convolution operation with 32 filters, each having a kernel size of (3, 5, 5). The convolutional operation uses a stride of (1, 2, 2) to reduce the spatial dimensions.
- BatchNormalization: Normalizes the activations of the previous layer across the batch.
- Activation: Applies the rectified linear unit (ReLU) activation function to introduce non-linearity.
- SpatialDropout3D: Performs dropout regularization by randomly setting elements of the input to 0 during training to prevent overfitting.
- MaxPooling3D: Performs 3D max pooling with a pool size of (1, 2, 2) and a stride of

(1, 2, 2) to downsample the feature maps.

3- Recurrent Layers:

- TimeDistributed: Applies the following layers to each time step of the input tensor independently.
- Flatten: Converts the 3D feature maps into a 2D representation.
- Bidirectional GRU (Gated Recurrent Unit): Processes the input sequence in both forward and backward directions. Two layers of bidirectional GRU with 256 units each are stacked together.
- The output of the second bidirectional GRU layer is a sequence of hidden states for each time step.

4- Output Layers:

- Dense: A fully connected layer with 37 (number of Arabic chars) units (assuming 37 different classes) that produces the final output.
- Activation: Applies the SoftMax activation function to obtain a probability distribution over the classes.

5- Model: Constructs a Keras Model object with the input and output tensors.

Model Explanation

- Input Layer: The model starts with an input layer that expects input data with a shape of (60, 150, 160, 1). This means the model accepts sequences of 60 frames, where each frame has dimensions of 150x160 pixels and a single channel (grayscale).

- Convolutional Blocks: The model contains three convolutional blocks. Each block consists of a series of operations including zero padding, 3D convolution, batch normalization, ReLU activation, spatial dropout, and 3D max pooling. These blocks aim to extract important spatial and temporal features from the input data.

- TimeDistributed Flatten: This layer applies the flatten operation to each time step of the input sequence, converting the 3D feature maps into a 2D representation while preserving the temporal dimension.

- Bidirectional GRU Layers: The flattened features are fed into two bidirectional GRU (Gated Recurrent Unit) layers. These layers capture the temporal dependencies in the sequence and provide context information for lipreading.

- Dense Layer: The output of the last GRU layer is passed through a dense layer with 37 units (corresponding to the number of classes or categories being predicted). The dense layer applies a linear transformation to the GRU outputs.

- Activation Layer: The output of the dense layer is passed through an activation function (SoftMax) to obtain the final predicted probabilities for each class

In summary, this model takes 3D input data, applies convolutional and pooling operations to extract spatial features, processes the temporal information using bidirectional GRU layers, and produces a probability distribution over the classes using a dense layer followed by SoftMax activation.

4.6.2. Lip-net model for English:

Overall, this model combines Conv3D, MaxPool3D, LSTM, and Dense layers to capture spatiotemporal features from lipreading data and perform classification. It is trained to predict the class labels associated with the lip movements in the input sequences.

The model architecture of a 3D convolutional neural network (CNN) followed by recurrent neural network (RNN) layers for sequence processing. Here's a summary of the model:

1- Model Initialization: The model is defined using the Sequential API.

2- Convolutional Layers:

- Conv3D: The first Conv3D layer has 128 filters, a kernel size of 3, and a padding of 'same'. It takes input with a shape of (75, 46, 140, 1), representing a 5D tensor with dimensions (time_steps, height, width, channels).
- Activation: Applies the rectified linear unit (ReLU) activation function to introduce non-linearity.
- MaxPool3D: Performs 3D max pooling with a pool size of (1, 2, 2) to downsample the feature maps.

3- Additional Convolutional Layers:

- Two additional Conv3D layers follow the same pattern as above, with 256 filters and 75 filters respectively. They both use a kernel size of 3 and 'same' padding.
- Activation: Applies the ReLU activation function to introduce non-linearity.
- MaxPool3D: Performs 3D max pooling with a pool size of (1, 2, 2) to downsample the feature maps.

4- Temporal Processing:

- TimeDistributed: Applies the following layers to each time step of the input tensor independently.
- Flatten: Converts the 3D feature maps into a 1D representation.

5- Recurrent Layers:

- Bidirectional LSTM (Long Short-Term Memory): Processes the input sequence in both forward and backward directions. Two layers of bidirectional LSTM with 128 units each are stacked together. The LSTM layers have dropout and recurrent dropout set to 0.2 to prevent overfitting.

6- Output Layer:

- Dense: A fully connected layer with 41 units (assuming 41 different classes) that produces the final output.
- Activation: Applies the softmax activation function to obtain a probability distribution over the classes.

In summary, this model takes 3D input data, applies convolutional and pooling operations to extract spatial features, processes the temporal information using bidirectional LSTM layers, and produces a probability distribution over the classes using a dense layer followed by softmax activation.

Model Explanation

This model is a sequential model designed for lipreading tasks. Here's an explanation of each component in the model:

- Conv3D Layers: The model starts with three Conv3D layers. Each layer applies 3D convolution to the input data. The first Conv3D layer has 128 filters, the second has 256 filters, and the third has 75 filters. The "padding" parameter is set to 'same' to ensure the output has the same spatial dimensions as the input. Each Conv3D layer is followed by a ReLU activation function.

- MaxPool3D Layers: After each Conv3D layer, a MaxPool3D layer is applied. These layers perform 3D max pooling, reducing the spatial dimensions of the feature maps. The pooling size is (1, 2, 2), meaning it performs pooling over the second and third dimensions while keeping the first dimension unchanged.

- TimeDistributed Flatten: This layer applies the flatten operation to each time step of the input sequence, converting the 3D feature maps into a 2D representation while preserving the temporal dimension.

- Bidirectional LSTM Layers: The flattened features are passed into two Bidirectional LSTM layers. Each LSTM layer has 128 units and is bidirectional, meaning it processes the input sequence in both forward and backward directions. The LSTM layers are designed to capture temporal dependencies and extract meaningful representations from the lipreading data.

- Dense Layer: The output of the last LSTM layer is connected to a Dense layer with 41 units. The Dense layer applies a linear transformation to the LSTM outputs and is followed by a SoftMax activation function, which produces the final predicted probabilities for each class.

4.7. Tools:

4.7.1. Google Collaboratory:

Google Colab is like jupyter notebooks that allows you to write and execute Python code without any setup and gives us free GPU.
We tried to use Google Colab to execute our model, but it keeps crashing all the time due to our large dataset and types of layers used in model's architectures like Conv3d and LSTMs which needs some additional resources to train such models.

4.7.2. Kaggle

Kaggle is an online platform that hosts machine learning competitions, provides datasets for practice, and offers a collaborative environment for data scientists and machine learning practitioners. It is widely used by the data science community to develop and showcase their skills, learn from others, and participate in competitions.

Kernels/Notebooks: Kaggle offers a feature called "Kernels" (previously known as notebooks) which allows users to create and share code, analysis, and visualizations. This is a convenient way to develop and run deep learning models using popular frameworks such as TensorFlow, PyTorch, or Keras. Users can also explore and learn from the kernels shared by others.

GPU and TPU Support: Kaggle provides access to powerful hardware accelerators like GPUs (Graphics Processing Units) and TPUs (Tensor Processing Units). Deep learning models can greatly benefit from the parallel computing capabilities of these devices, enabling faster training and inference times.

Overall, Kaggle provides a comprehensive platform for running deep learning models, leveraging its datasets, kernels, GPU/TPU support, competitions, and a vibrant community. It is a valuable resource for both beginners and experienced practitioners in the field of deep learning

Chapter Five: Experiments & Results

5. Experiments & Results:

In this chapter we will go through the experiments we performed on the two implemented models and the two datasets and trace the results while changing the hyper parameters: padding method, batch size.

GRID dataset:

Data is spitted into 80% training data and 20% validation, batch size is 3, 5000 videos with total number of words occurrences 50,000 word and total number of classes 42.

Our Arabic Dataset:

The data is also spitted into 80% training data and 20% validation, batch size 3 withtotal number of videos 4400 and total number of classes 37 (char number).

5.1. Lip-net Model Results for English:

The loss used by this model is CTC Loss and Adams optimizer.

Table 4 the English model accuracy

Dataset	Epochs	Accuracy
Grid	30	30%
Grid	50	53%
Grid	70	69.4%
Grid	100	86.2%
Grid	130	91.3%
Grid	150	97.15%

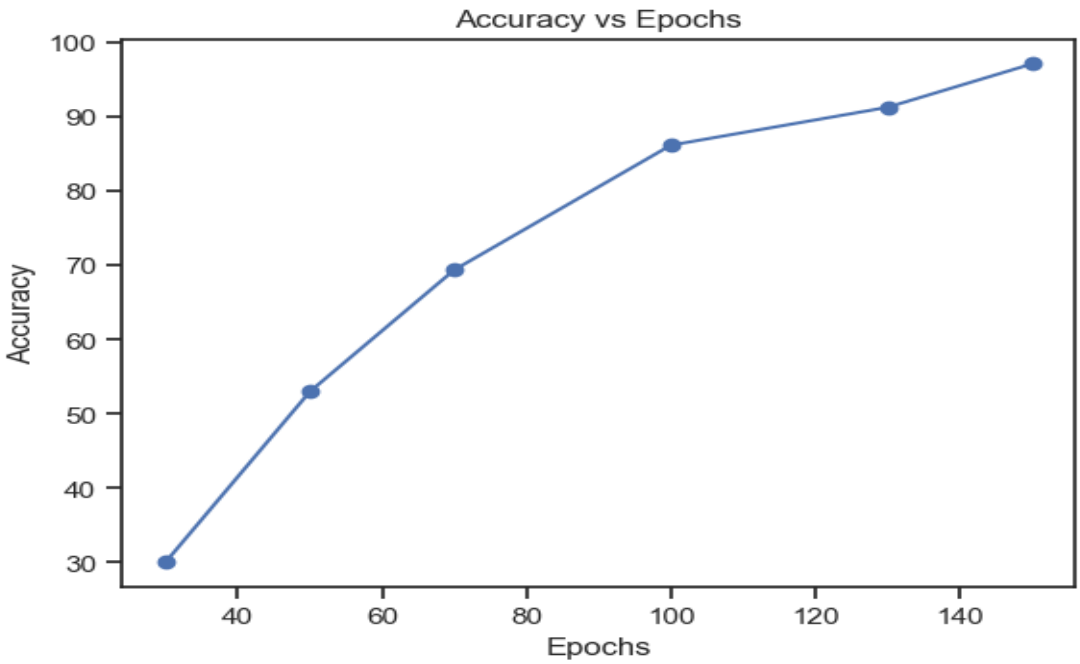


Figure 36 number of epochs and accuracy

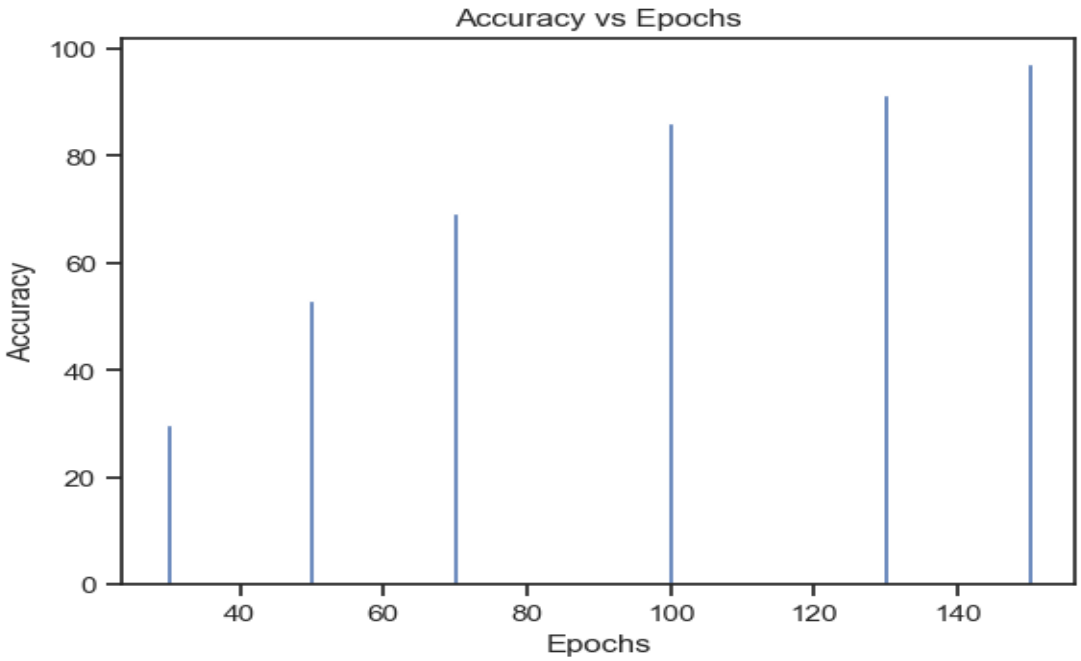


Figure 37 accuracy in epochs

5.2. Lip-net Model Results for Arabic:

The loss used by this model is CTC Loss and Adams optimizer, each checkpoint is 10 epochs

Table 5 Arabic model accuracy

Dataset	Checkpoint	Accuracy
Our private data	5	40%
Our private data	15	43%
Our private data	25	56%
Our private data	35	70.2%
Our private data	45	76.3%
Our private data	65	79.33%
Our private data	75	79.27%
Our private data	80	79.83%
Our private data	83	80.06%

This graph explains the accuracy over the checkpoints in training process

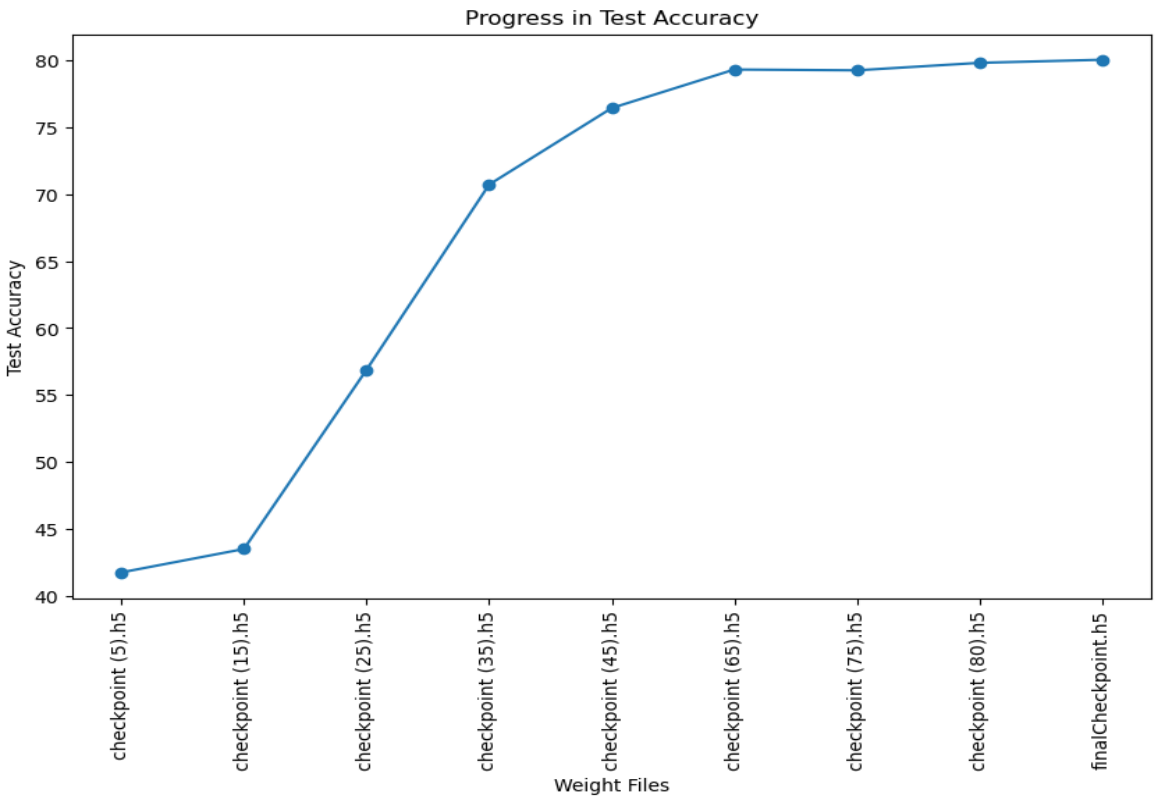


Figure 38 arabic accuracy over the checkpoints

5.3. Result Visualization:

In this chapter, we will discuss the results using visualization techniques and graphs. Visualizations play a crucial role in presenting data in a more understandable and insightful manner. By representing data visually, we can gain a deeper understanding of patterns, trends, and relationships within the data.

One common type of visualization is graphs, which are effective in displaying the relationship between different variables. We can create various types of graphs such as line plots, bar charts, and scatter plots to represent different aspects of the data.

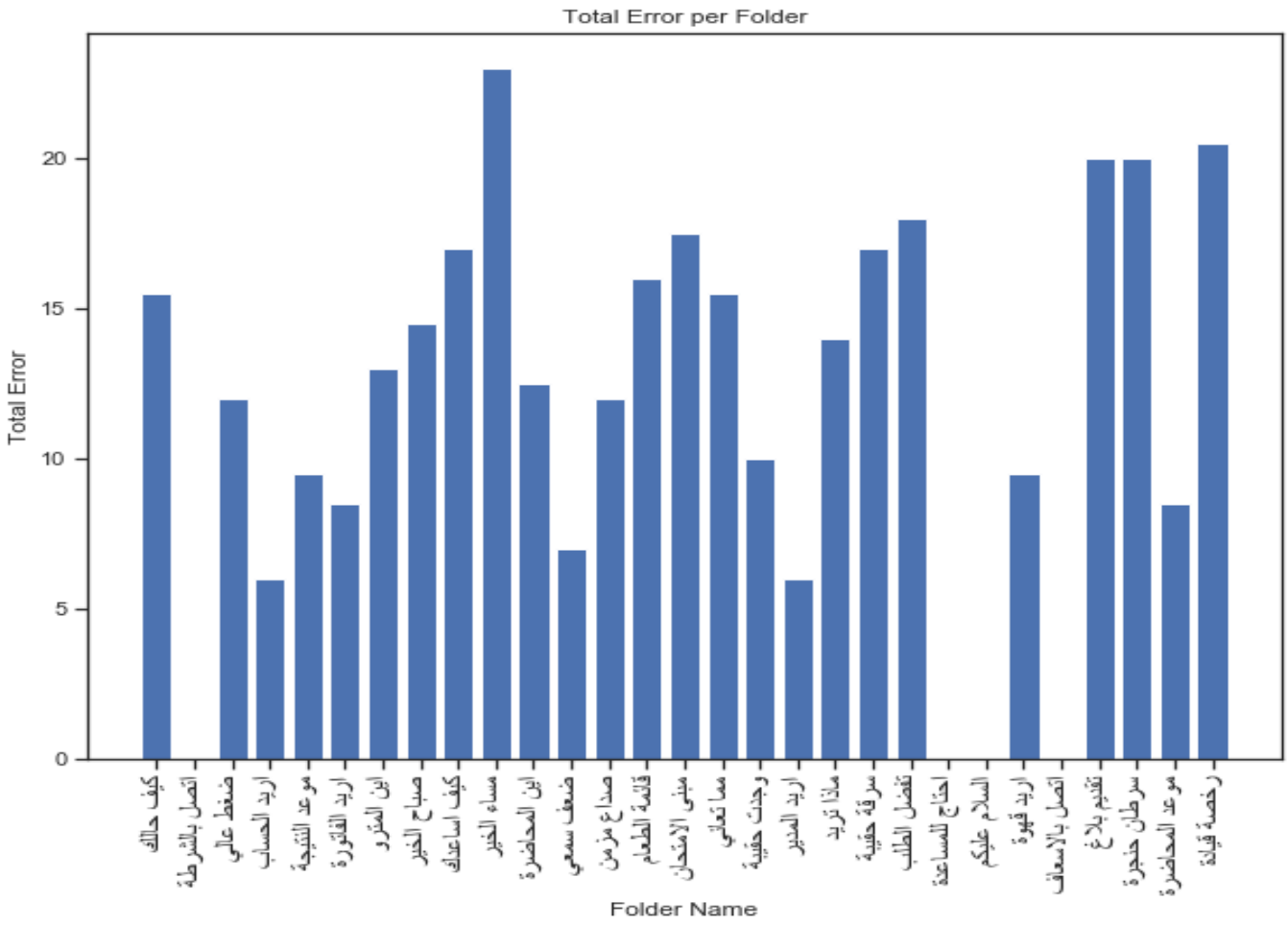


Figure 39 accuracy or each sentence

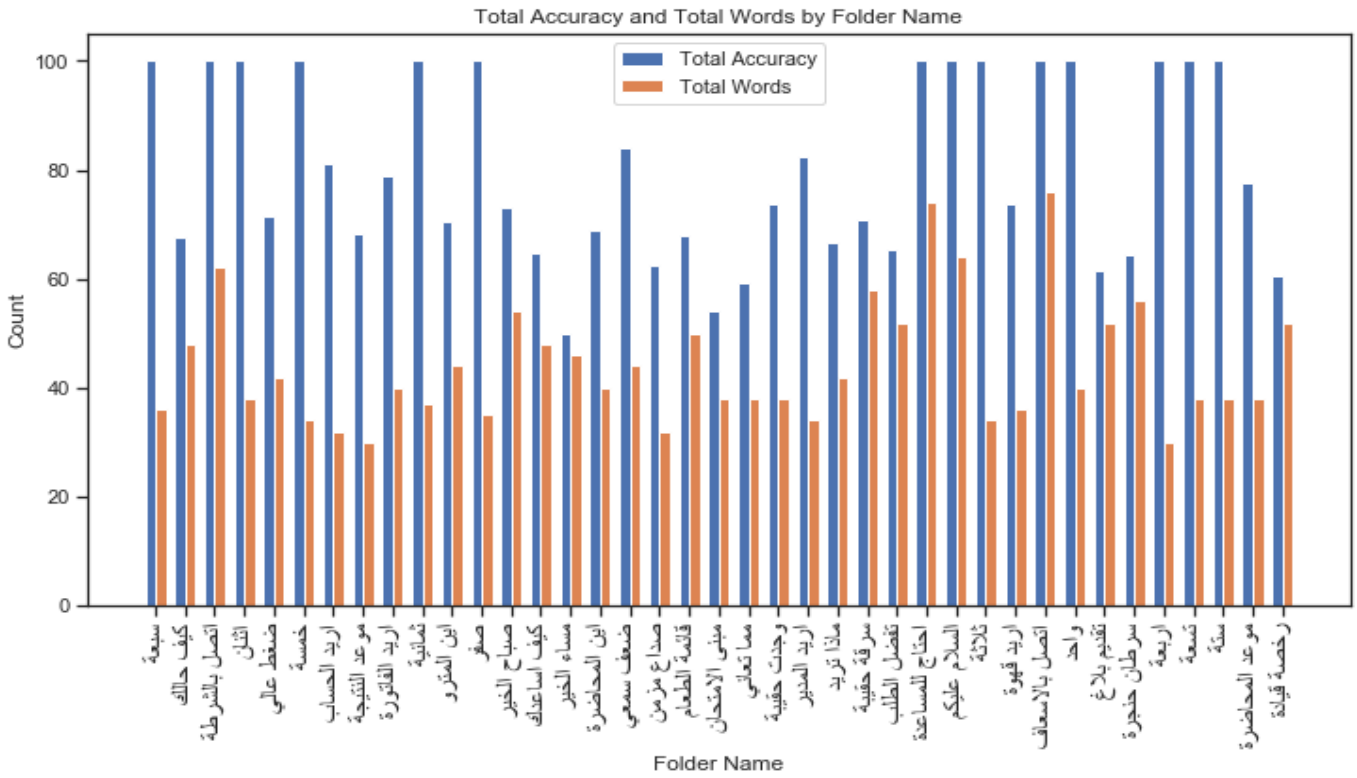


Figure 40 accuracy with the number of words in sentence

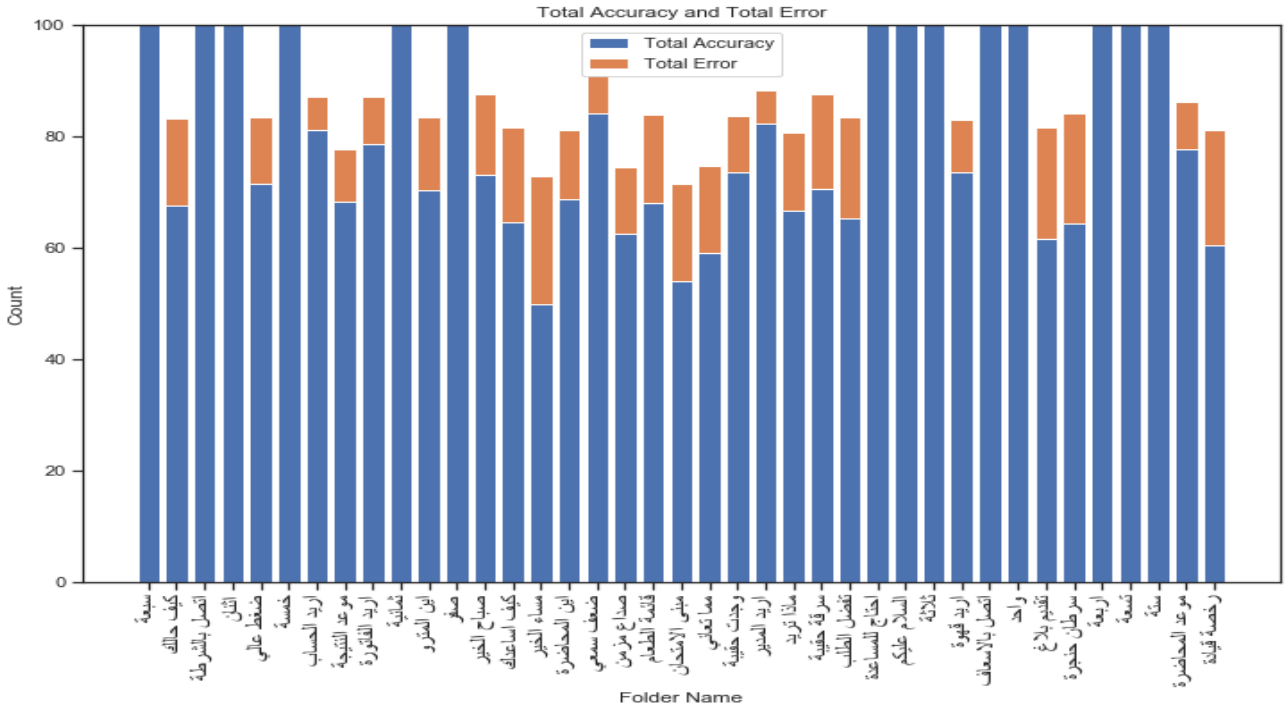


Figure 41 Accuracy against error in each sentence

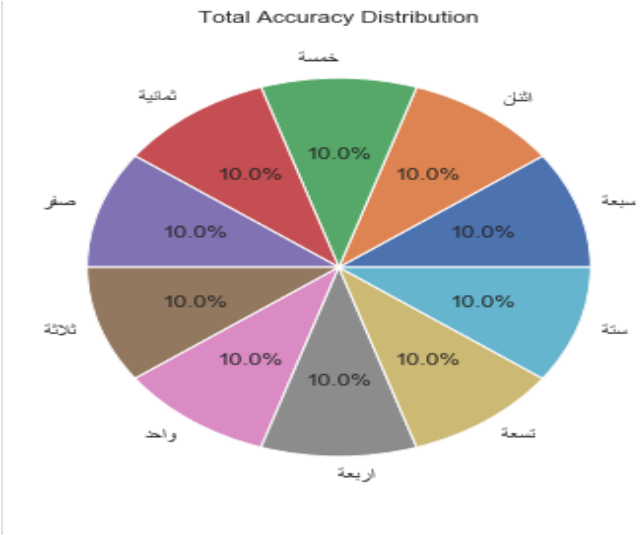


Figure 42 accuracy distribution in numbers [0-9]

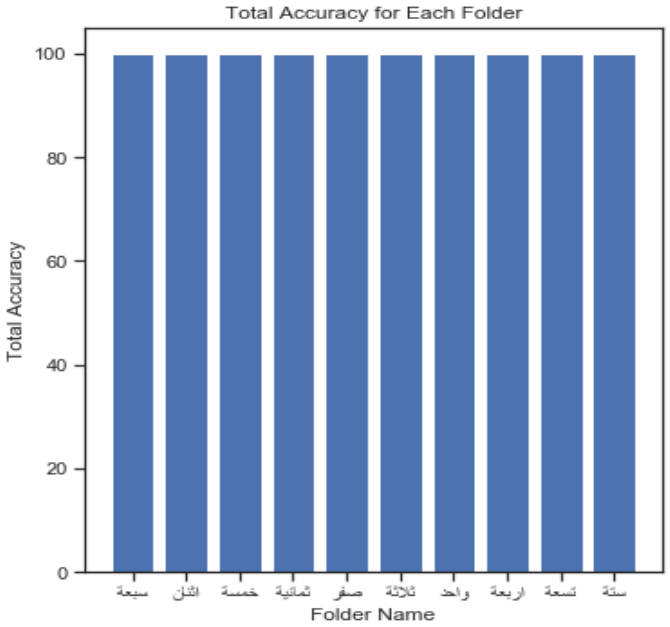


Figure 43 accuracy in numbers dataset

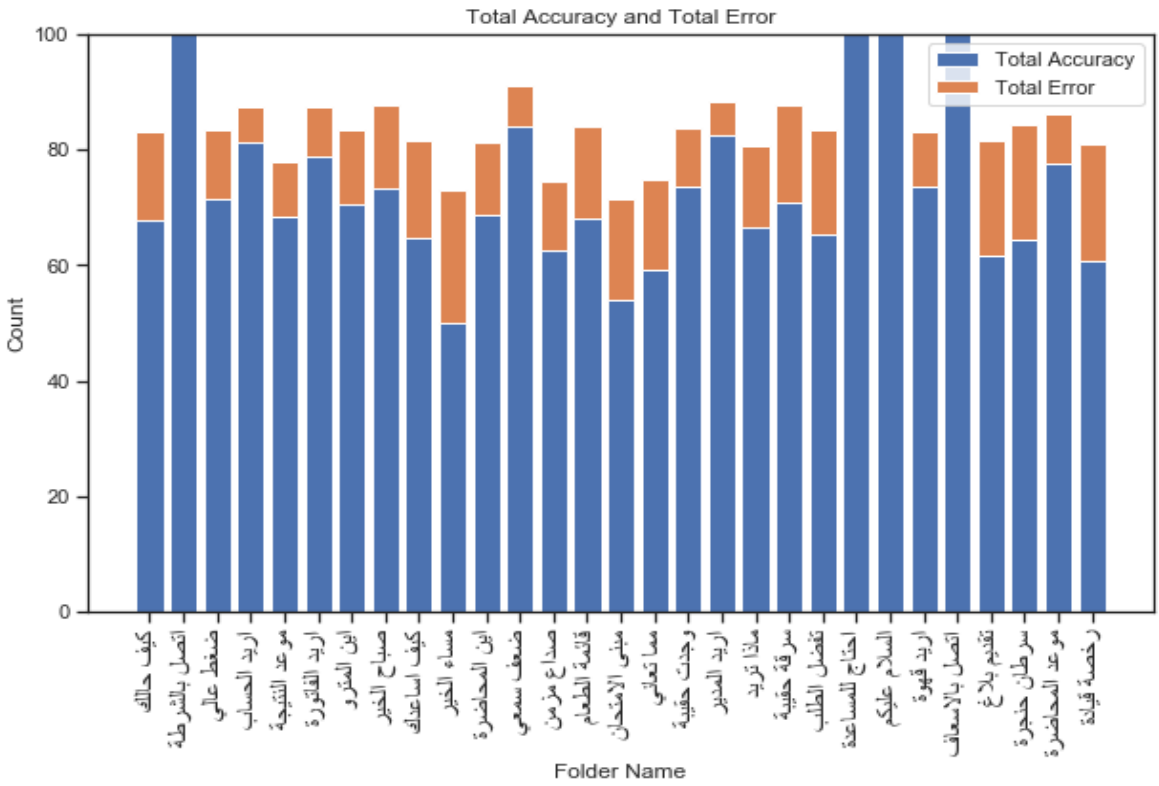


Figure 44 accuracy against error in our data

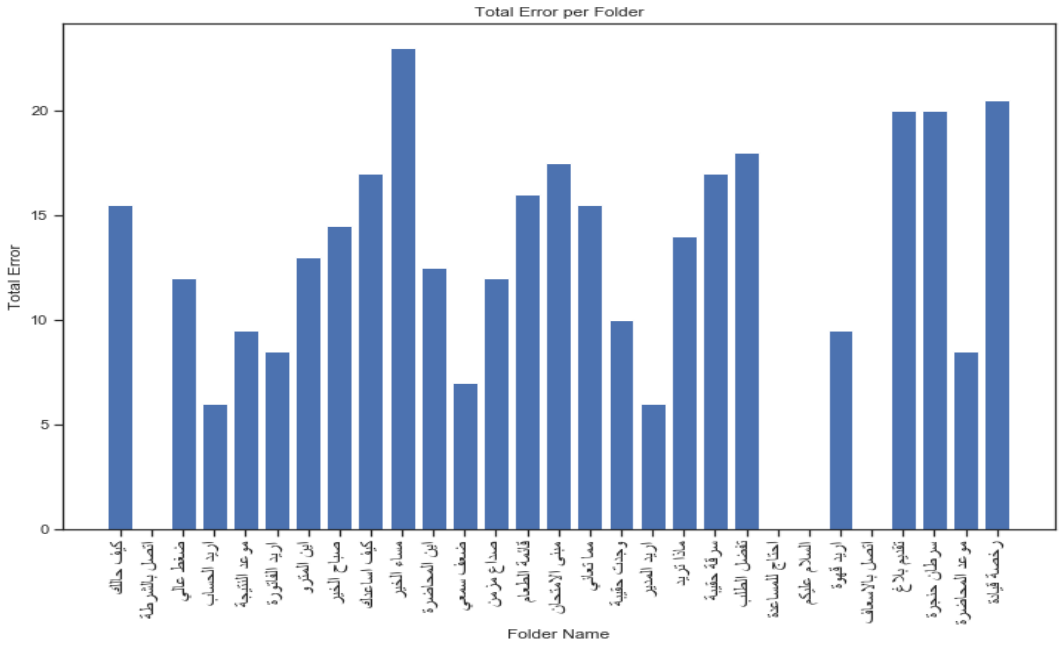


Figure 45 error in each Arabic sentence

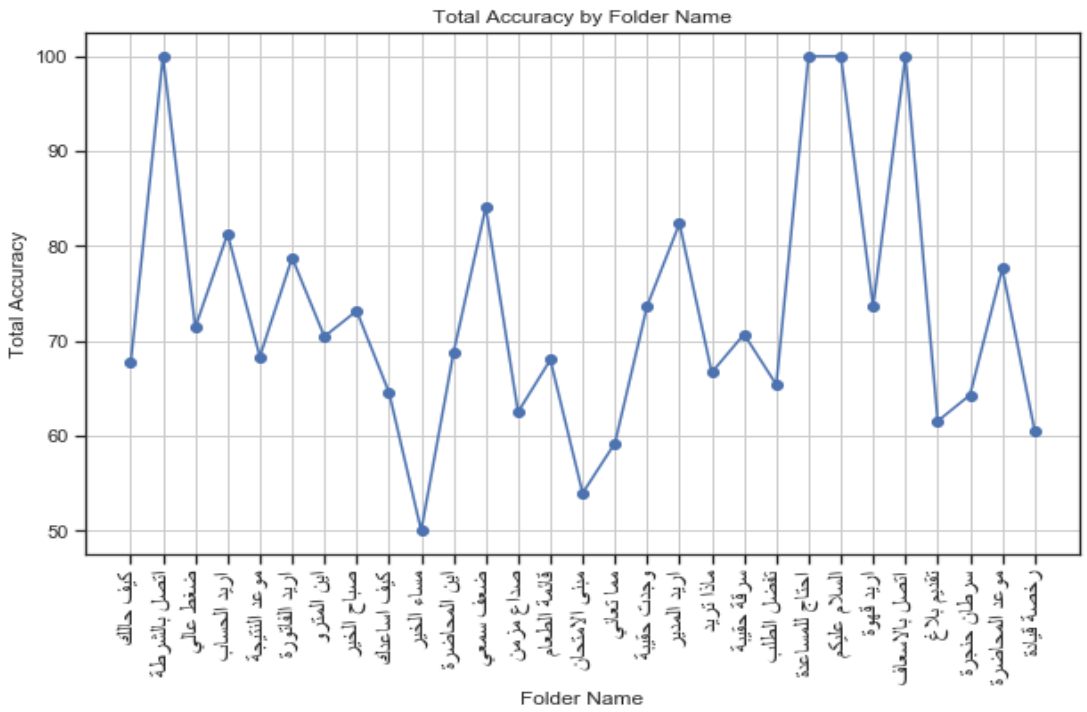


Figure 46 accuracy line

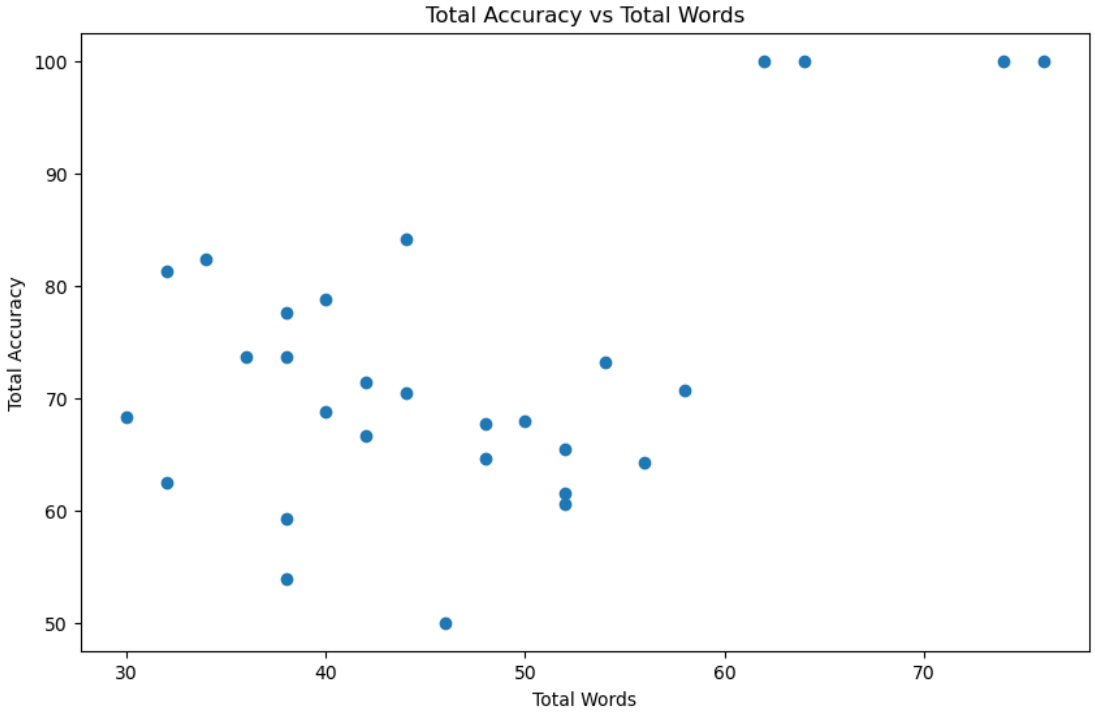


Figure 47 total words and accuracy

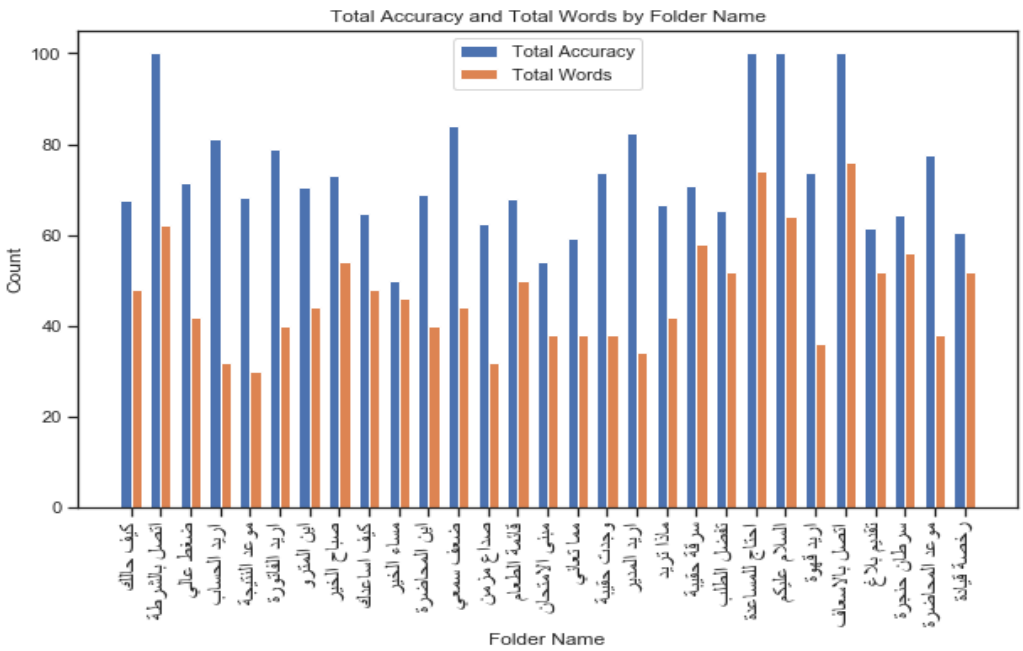


Figure 48 the relationship between the accuracy and total words

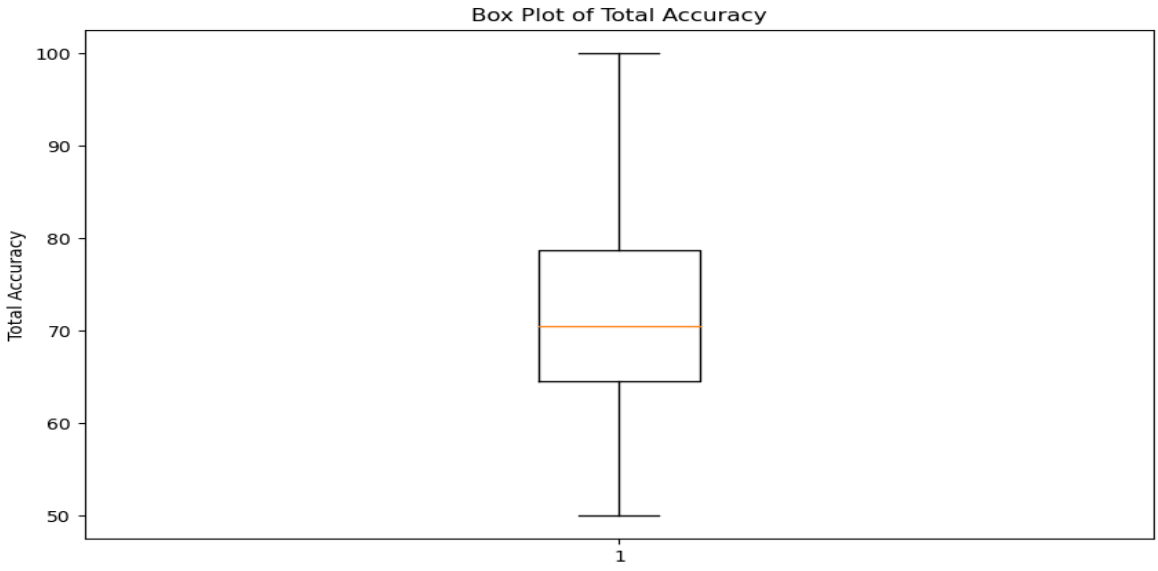


Figure 49 distribution of accuracy in sentence

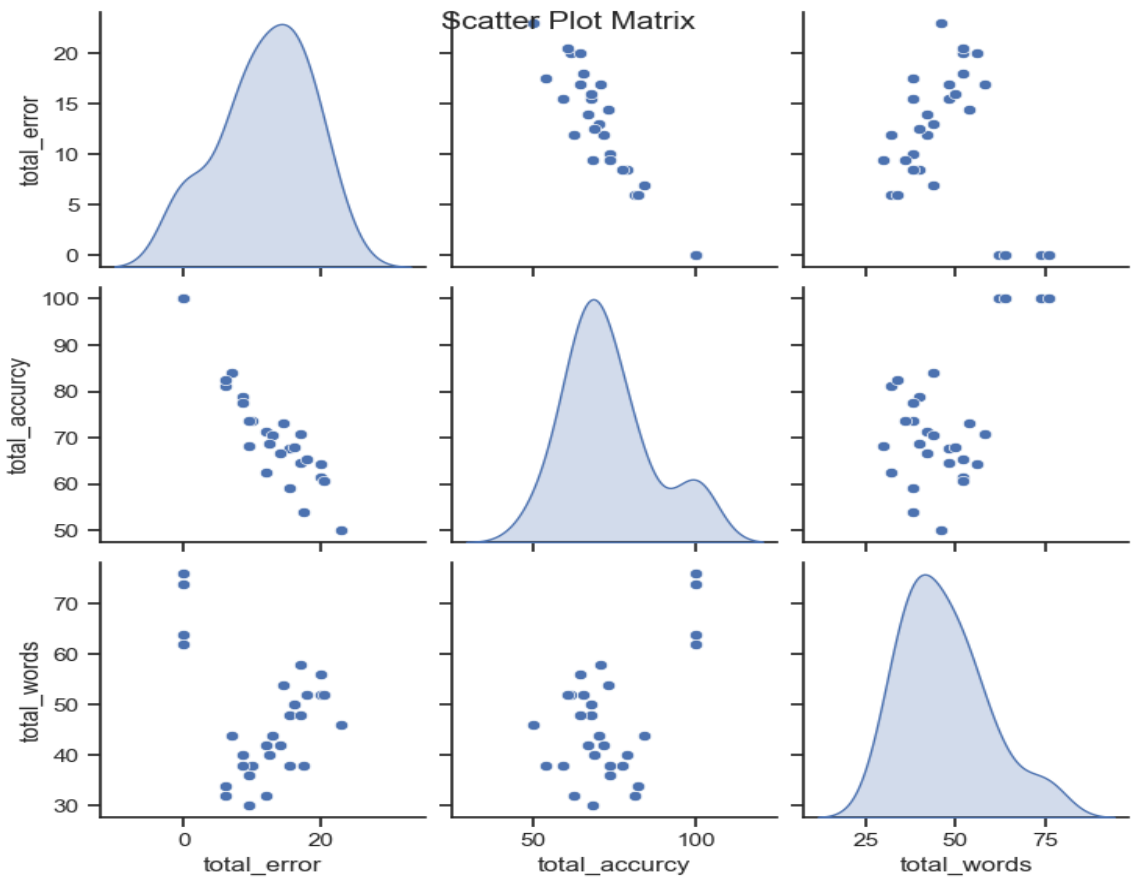


Figure 50 the 3 main relationships between accuracy, error and total words

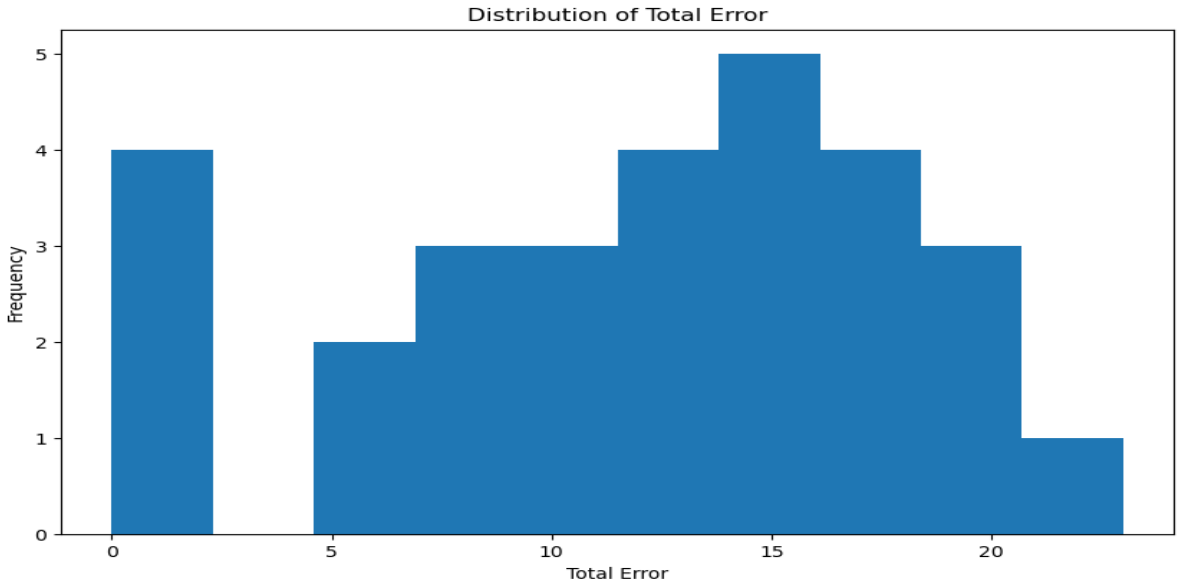


Figure 51 total error

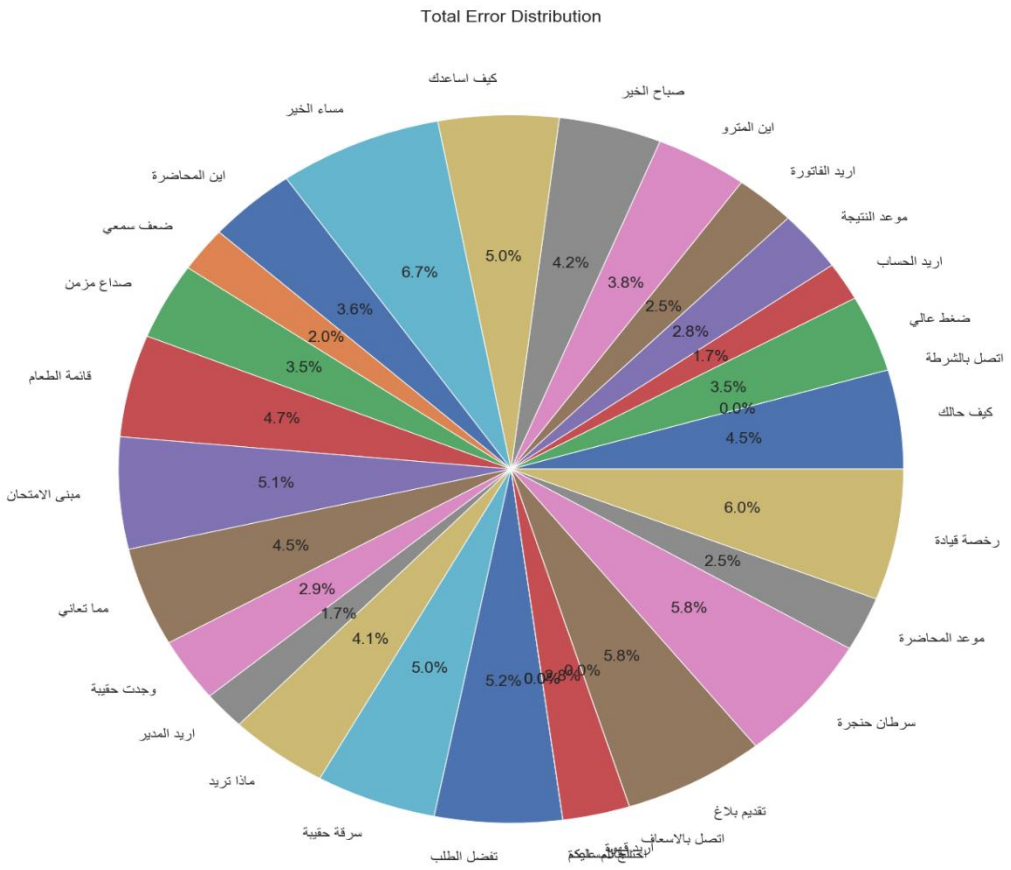


Figure 52 error distribution

The avg time for some sentence in different time and different machines

Table 6 Avg time for prediction

Number	Sentence	Avg Time
1	أين المترو	29.922 s
2	كيف حالك	28.774 s
3	ماذا تريد	28.186 s
4	صباح الخير	26.915 s
5	سرطان الحنجرة_	30.463 s
6	تفضل الطلب	29.231 s

5.4. Observation:

Working with the Arabic language in our graduation project presented us with a significant risk and challenge. Unlike English, there was a lack of previous outputs and resources available for Arabic language processing. However, despite the difficulties, we found the experience to be enjoyable and rewarding. Achieving an accuracy of 80% in Arabic and 97% in English was a remarkable accomplishment for us. We hope that this progress signifies a positive trend for future advancements in Arabic language processing. We remain optimistic about the prospects for Arabic language research and development in the coming years, and we look forward to witnessing further advancements that will benefit the Arab community.

Chapter Six: User Manual

6. User Manual:

This chapter explains the installation guide and all the dependencies required to run our desktop application and how to interact with it.

6.1. Web Application Walkthrough:

For The Application in main screen that is two buttons for Arabic and English

For Arabic

1- Click on Arabic button.

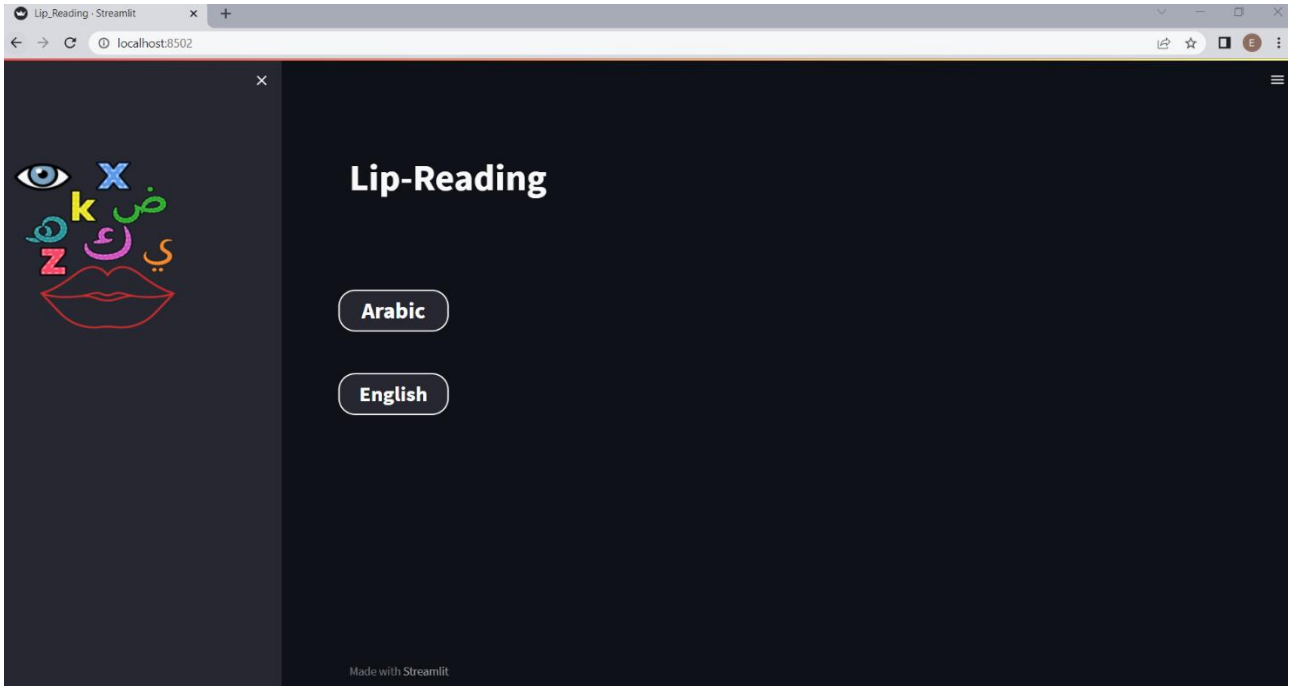


Figure 53 home page

2- Click on the Browse file

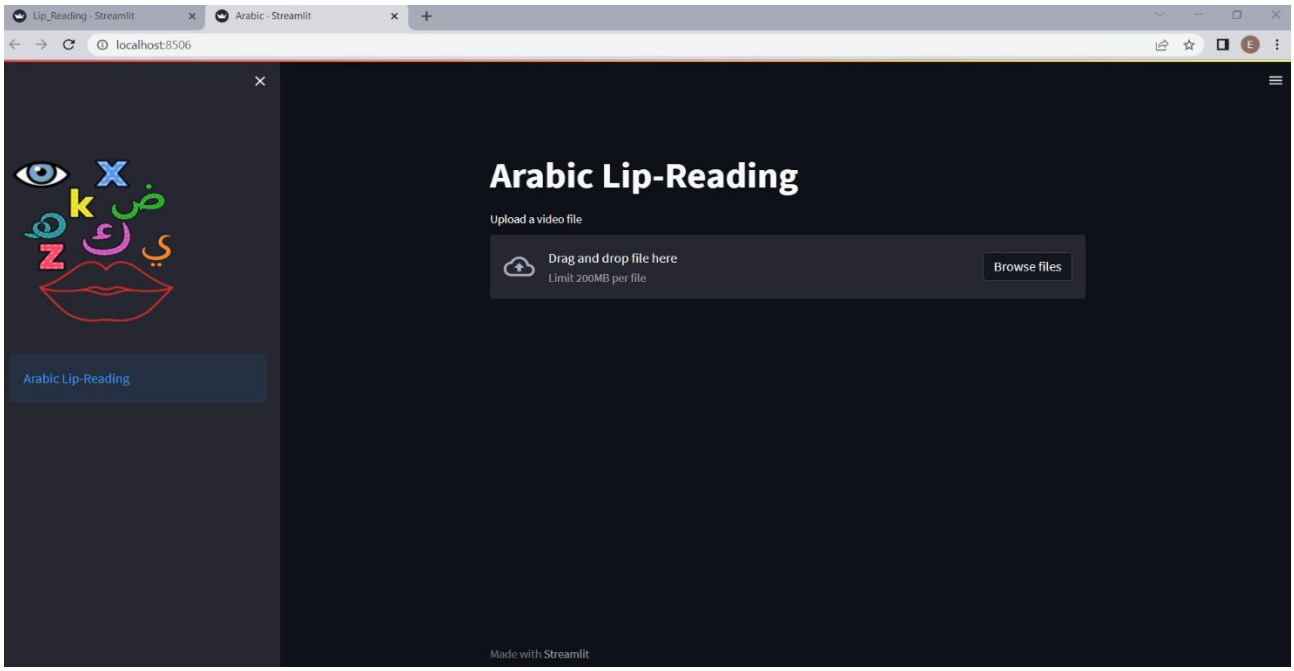


Figure 54 Arabic screen

3- Choose file from the windows dialog

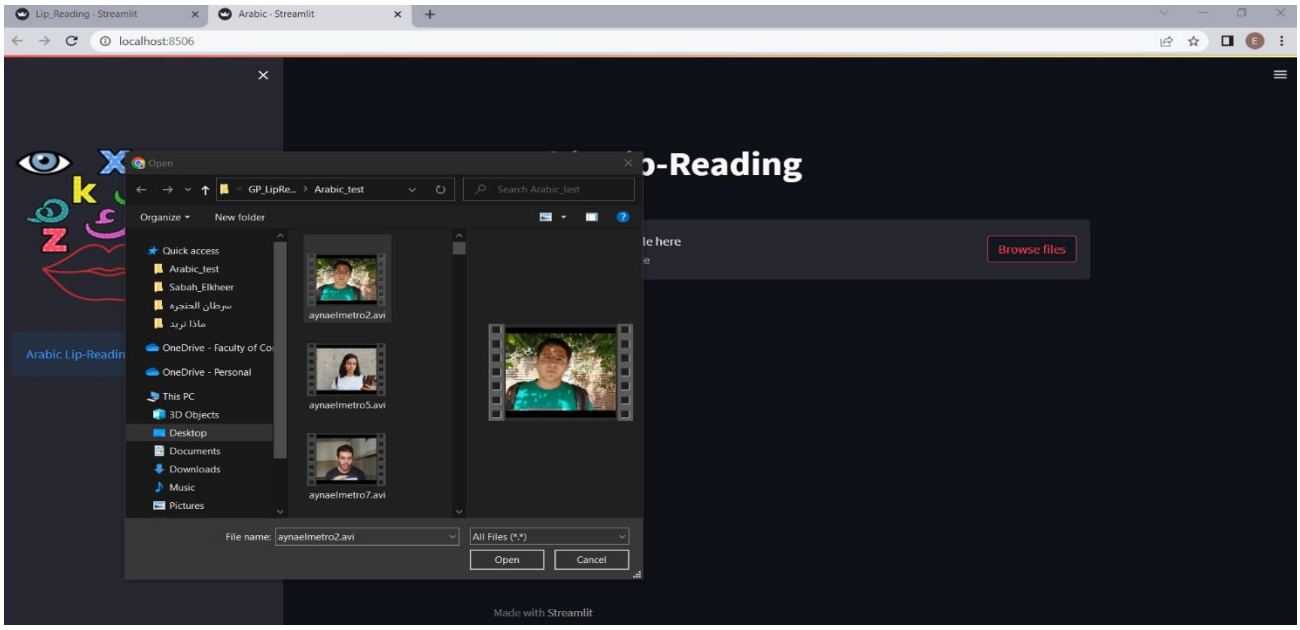


Figure 55 pick video from dialog

4- Wait until the output is displayed.

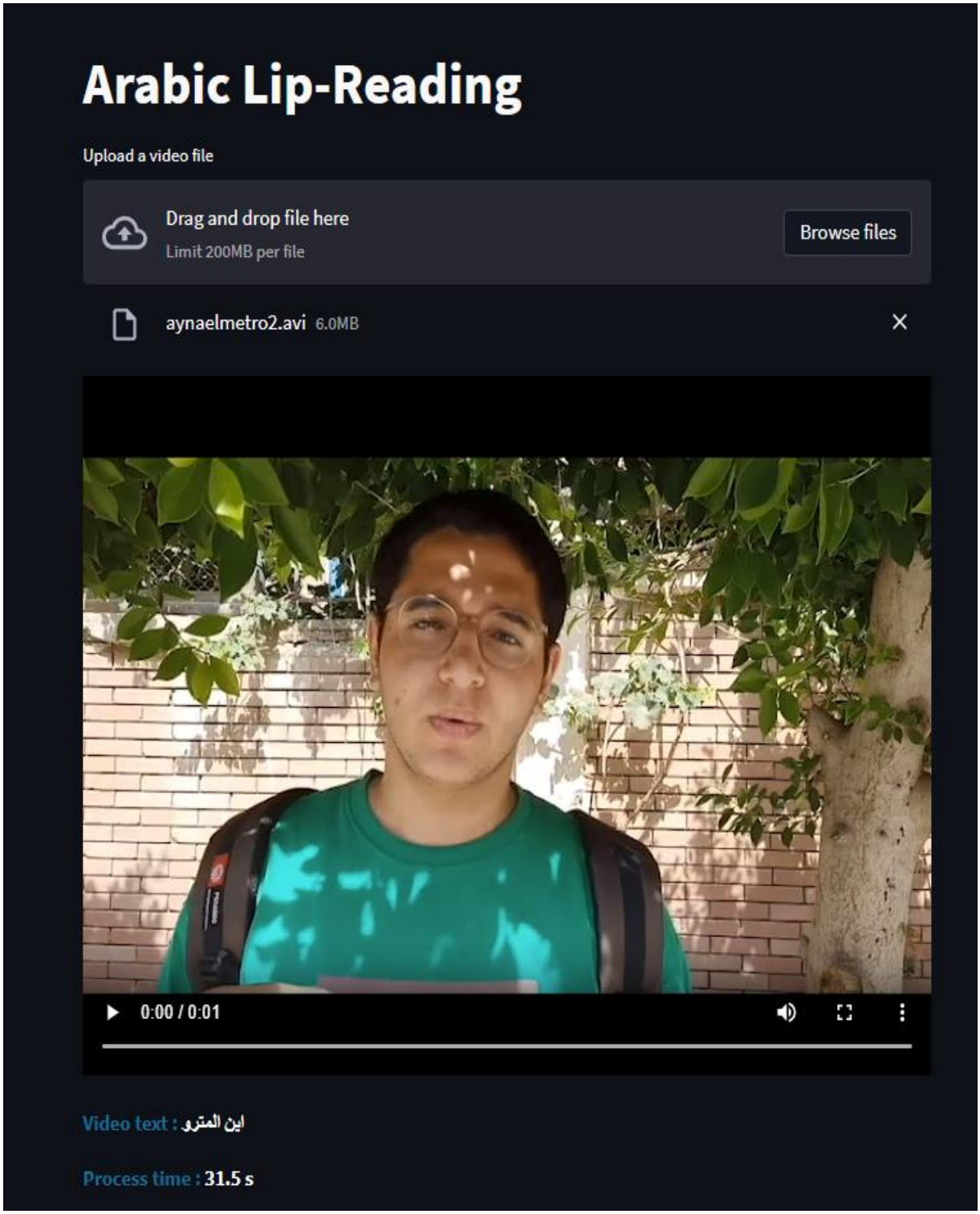


Figure 56 the output

For English

1- Click on English button.

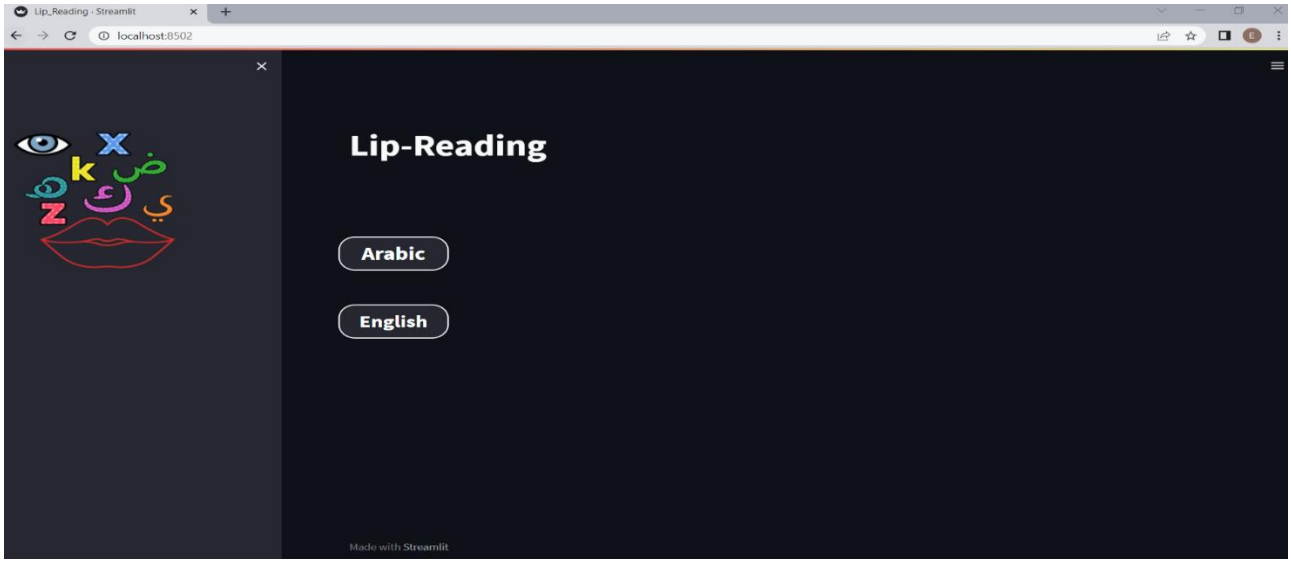


Figure 57 Home page

2- Click on the Browse file.

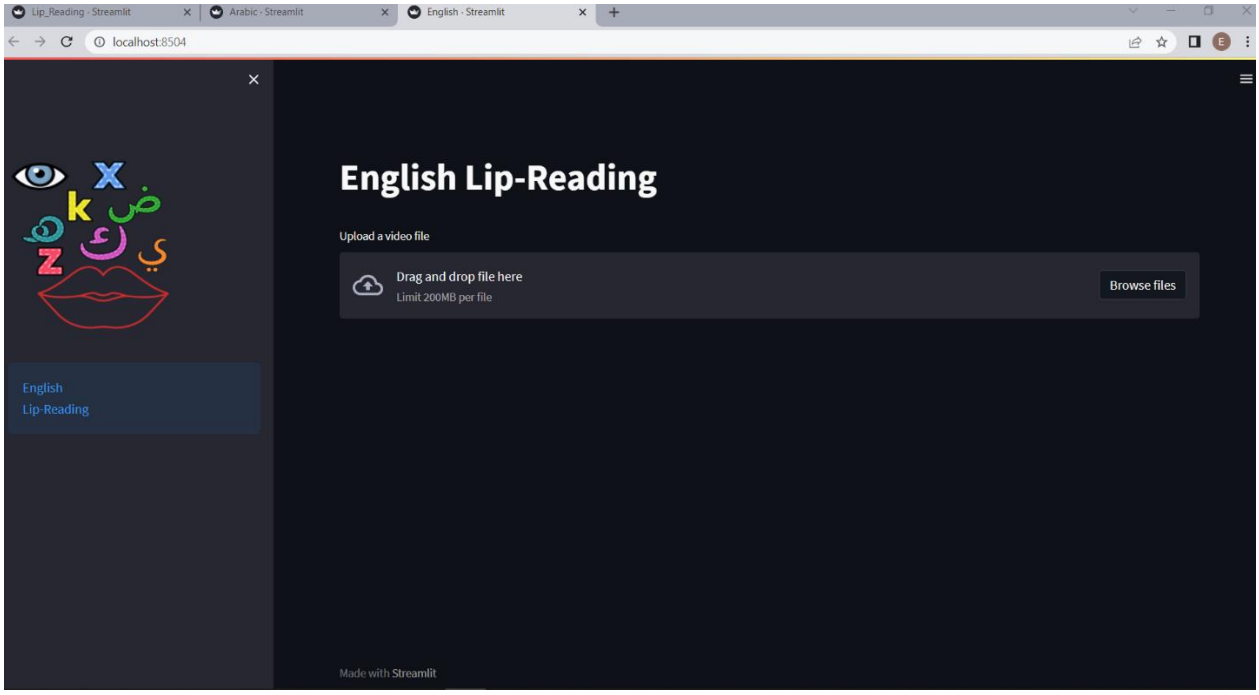


Figure 58 English screen

3- Choose file from the windows dialog.

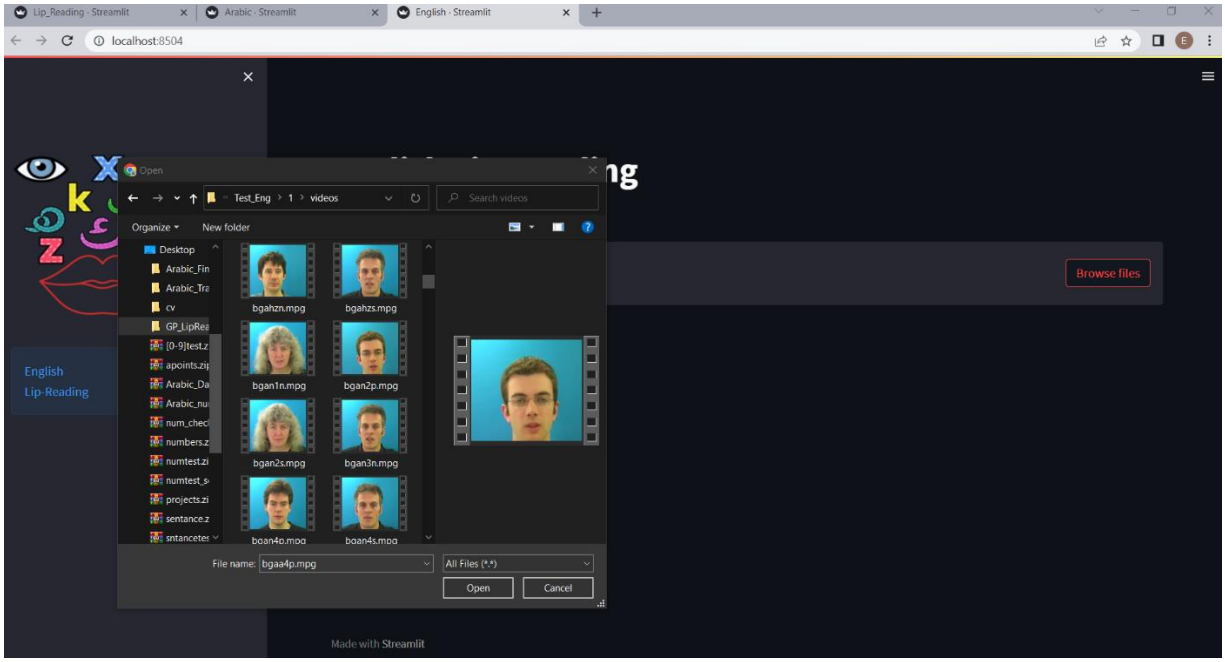


Figure 59 pick video from dialog for English

4- Wait until the output is displayed.

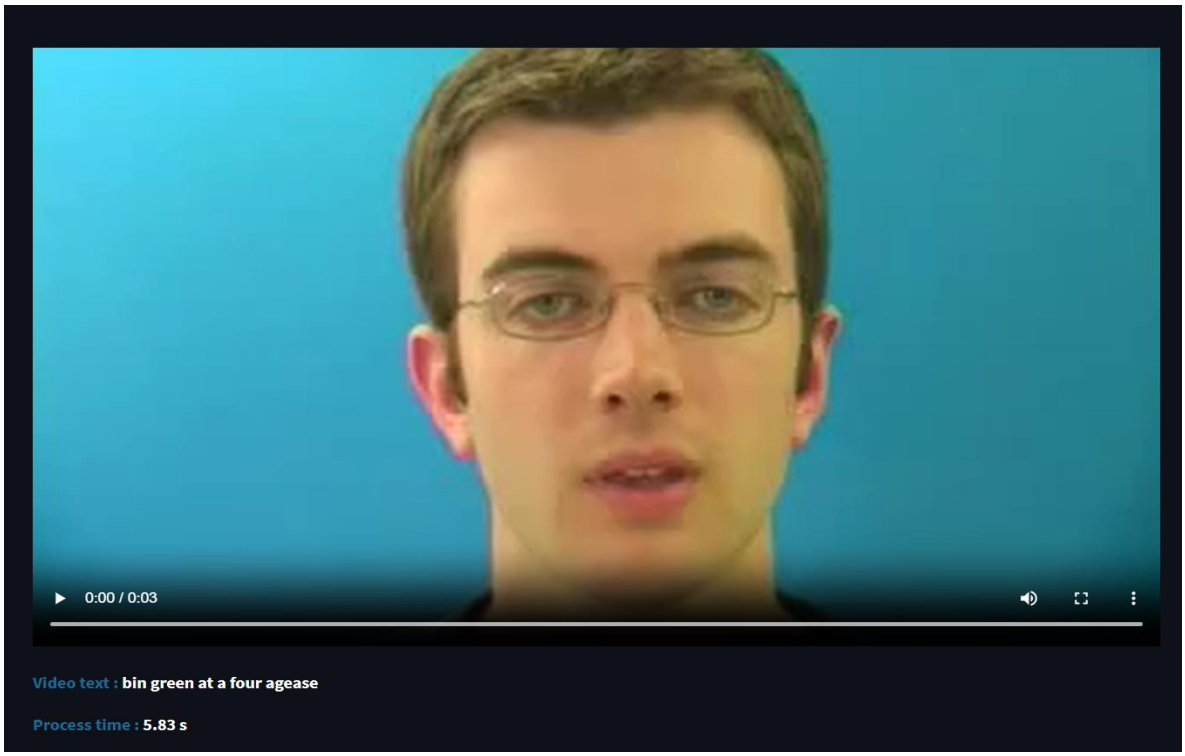


Figure 60 the English output

Chapter Seven: Conclusion & Future Work

7. Conclusion & Future Work:

In this chapter we will explain our conclusion from all phases of the project and what else we can add to this project to come into real life as future work.

7.1. Conclusion:

Working with the Arabic language in our graduation project posed a significant risk and challenge due to several reasons. One of the main challenges was the scarcity of previous outputs and available resources for Arabic language processing compared to English. While there are numerous tools, datasets, and models already developed for the English language, the same cannot be said for Arabic. This lack of existing infrastructure made it much more difficult for us to find relevant references and build upon existing work.

Despite these challenges, we found the experience of working with Arabic to be enjoyable and rewarding. We were motivated by the opportunity to contribute to the advancement of Arabic language processing and address the existing gaps in the field. This project allowed us to explore uncharted territory and push the boundaries of what is possible in Arabic language analysis.

One of the significant achievements of our project was attaining an accuracy of 80% in Arabic and 97% in English. This accomplishment is remarkable considering the limited resources and established frameworks available for Arabic. It demonstrates the effectiveness of our approach and the potential for further improvements in Arabic language processing.

We believe that our progress in this project indicates a positive trend for future advancements in Arabic language research and development. As more researchers and developers recognize the importance of Arabic language processing, we anticipate an increase in efforts to overcome the existing challenges and fill the

gaps in this domain. This progress will contribute to the overall development of Arabic NLP (Natural Language Processing) tools, which will have significant benefits for the Arab community.

Looking ahead, we remain optimistic about the prospects for Arabic language research and development. We hope to witness further advancements in Arabic NLP techniques, including the creation of larger and more diverse datasets, the development of robust Arabic language models, and the improvement of language understanding and generation systems. These advancements will not only benefit academic research but also have practical applications in various fields such as machine translation, sentiment analysis, information retrieval, and voice assistants.

In conclusion, despite the risk and challenges associated with working on Arabic language processing, our graduation project proved to be a rewarding experience. Our achievements in terms of accuracy and the knowledge gained throughout the process give us confidence in the potential for further advancements in Arabic language research. We are excited about the future of Arabic language processing and its positive impact on the Arab community

7.2. Future Work:

In order to further enhance the accuracy of Arabic language processing, we have devised a plan to focus on two key areas: training and data acquisition.

Firstly, we aim to improve accuracy by conducting additional training. This involves refining our existing models and algorithms through iterative training sessions. By analyzing the performance of our models and identifying areas of weakness, we can fine-tune the parameters and optimize the algorithms to achieve better results. This process often involves adjusting hyperparameters, modifying the network architecture, or implementing advanced techniques such as transfer learning or ensemble methods.

Secondly, we recognize the importance of data quality and quantity in improving accuracy. To that end, we plan to enhance our dataset by utilizing better materials in the video recording process. By using high-quality videos and audio recordings, we can capture a more diverse range of Arabic language variations, accents, and contexts. This will provide a richer and more representative dataset for training our models, allowing them to better understand and process the intricacies of the Arabic language.

Expanding the dataset is particularly crucial in Arabic language processing due to the challenges associated with its linguistic diversity and regional variations. By incorporating a wide range of dialects, registers, and topics in our training data, we can improve the models' ability to handle real-world scenarios and accurately capture the nuances of Arabic communication.

Additionally, we will employ rigorous data preprocessing techniques to clean and standardize the dataset. This involves removing noise, correcting errors, and normalizing the data to ensure consistency. By employing robust data preprocessing methods, we can mitigate potential biases and ensure the accuracy of our models is not compromised by faulty or misleading data.

It is important to note that increasing accuracy in Arabic language processing is an ongoing and iterative process. As new data becomes available and the field advances, we will continue to refine and update our models to stay at the forefront of Arabic language processing research.

By focusing on both training optimization and dataset enhancement, we aim to significantly improve the accuracy of Arabic language processing. This, in turn, will enable a wide range of applications and benefits, such as improved machine translation, sentiment analysis, voice recognition, and text understanding for the Arabic-speaking community.

Overall, our commitment to increasing accuracy through targeted training and improved datasets reflects our dedication to advancing Arabic language processing and providing better tools and technologies for the Arab community.

7.3. References:

[1] ”The History of Sign Language” https://study.com/academy/lesson/the-history-of-sign-language.html (accessed Feb. 10, 2020).

[2] “Lip-Reading” https://www.encyclopedia.com/social-sciences-and-law/education/education-terms- and-concepts/lip-reading (accessed Feb. 10, 2020).

[3] “How does a deaf person communicate?” https://www.hearingdogs.org.uk/deafness-and-hearing-loss/how-deaf-people-communicate/. (accessed Feb. 10, 2019).

[4] “LipNet in Autonomous Vehicles | CES 2017.” https://www.youtube.com/watch?v=YTkqA189pzQ (accessed Nov. 20, 2019).

[5] ” New AI research makes it easier to create fake footage of someone speaking” https://www.theverge.com/2017/7/12/15957844/ai-fake-video-audio-speech-obama (accessed Nov. 20, 2019).

[6] “Automated lip reading technique for password authentication.” https://www.youtube.com/watch?v=hZdrudfP7k0 (accessed Nov. 20, 2019).

[7] T. Afouras, J. S. Chung, and A. Zisserman, “The conversation: Deep audio-visual speech enhancement,” *Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH*, vol. 2018-Septe, pp. 3244–3248, 2018, doi: 10.21437/Interspeech.2018-1400.

[8] M. Selvamanikkam, “Introduction to Artificial Intelligence.” https://becominghuman.ai/introduction-to-artificial-intelligence-5fba0148ec99.

[9] A. Bonner, “The Complete Beginners Guide to Deep Learning.” https://towardsdatascience.com/intro-to-deep-learning-c025efd92535.

[10] R. Aggarwal, “Bi-LSTM.” https://medium.com/@raghavaggarwal0089/bi-lstm-bc3d68da8bd0.

[11] A. Mittal, “Understanding RNN and LSTM.” https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e.

[12] “Understanding LSTM Networks.” https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[13] M. Phi, “Illustrated Guide to LSTM’s and GRU’s: A step by step explanation.” https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21.

[14] E. Petajan, B. Bischoff, D. Bodoff, and N. M. Brooke, “An improved automatic lipreading system to enhance speech recognition,” *Conf. Hum. Factors Comput. Syst. - Proc.*, vol. Part F1302, pp. 19–25, 1988, doi: 10.1145/57167.57170.

[15] K. M. A. Pentland, “Automatic lipreading by optical-flow analysis,” 1991.

[16] J. F. Guitarte Pérez, A. F. Frangi, E. L. Solano, and K. Lukas, “Lip Reading for robust speech recognition on embedded devices,” *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. I, pp. 473–476, 2005, doi: 10.1109/ICASSP.2005.1415153.

[17] A. A. Shaikh, D. K. Kumar, W. C. Yau, M. Z. C. Azemin, and J. Gubbi, “Lip reading using optical flow and support vector machines,” *Proc. - 2010 3rd Int. Congr. Image Signal Process. CISP 2010*, vol. 1, no. June

2014, pp. 327–330, 2010, doi: 10.1109/CISP.2010.5646264.

[18] J. S. Chung, A. Senior, O. Vinyals, and A. Zisserman, “Lip reading sentences in the wild,” *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 3444–3450, 2017, doi: 10.1109/CVPR.2017.367.

[19] Y. M. Assael, B. Shillingford, S. Whiteson, and N. de Freitas, “LipNet: End-to-End Sentence-level Lipreading,” pp. 1–13, 2016.

[20] T. Stafylakis and G. Tzimiropoulos, “Combining residual networks with LSTMs for lipreading,” *Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH*, vol. 2017-Augus, pp. 3652–3656, 2017, doi: 10.21437/Interspeech.2017-85.

[21] T. Afouras, J. Son Chung, and A. Zisserman, “Deep lip reading: A comparison of models and an online application,” *Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH*, vol. 2018-Septe, pp. 3514–3518, 2018, doi: 10.21437/Interspeech.2018-1943.

[22] D. K. Margam *et al.*, “LipReading with 3D-2D-CNN BLSTM-HMM and word-CTC models,” 2019.

[23] X. Weng and K. Kitani, “Learning Spatio-Temporal Features with Two-Stream Deep 3D CNNs for Lipreading,” 2019 17.

[24] ” The Oxford-BBC Lip Reading Sentences 2 (LRS2) Dataset” http://www.robots.ox.ac.uk/~vgg/data/lip_reading/lrs2.html#:~:text=Overview,divided%20according%20to%20broadcast%20date. (accessed Feb. 20, 2020).

[25] A. E. and S. Peleg, “VID2SPEECH: SPEECH RECONSTRUCTION FROM SILENT VIDEO Ariel Ephrat and Shmuel Peleg The Hebrew University of Jerusalem Jerusalem, Israel.”