جامعة الملك سعود
King Saud University

كلية علوم الحاسب والمعلومات

King Saud University
College of Computer and Information Sciences
Computer Science Department

شـــفه
SHEFAH

Shefah: Lip Reading for the Arabic Language

Prepared by:

Majed Alkhraiji	438104708
Obadah Arabi	438170177
Tariq Aljaber	438101357

Supervised by:

Dr. Sultan Alfarhood

Software project for the degree of Bachelor in Computer Science
Second Semester of 1442 (Spring 2021)

Acknowledgement

We would like to express our deepest gratitude to Dr. Sultan Alfarhood for his valuable and constructive suggestions during the planning and development of this research work. His willingness to give his time so generously has been greatly appreciated.

We would like to extend our gratitude to whomever made this project possible by their willingness to share their time and effort.

English Abstract

Technology nowadays has evolved to the point where devices can recognize speech from voice and convert it to text. However, voice-based speech recognition fails in some situations, loud environments, multiple speakers, or lack of sound. Lip reading can be used in the previous cases. Not only that, but it also can increase the accuracy of voice-based speech recognition.

The purpose of this project is to design a system able to recognize spoken Arabic words from a person's lips using only video input and without relying on audio. Even though this idea is not completely new, no similar systems were built with Arabic pronunciation and vocabulary in mind. The system developed can take a video input of a person pronouncing an Arabic number and output the spoken number relying only on visual input. This is achieved through the use of deep learning and applying multiple steps of preprocessing to the video.

Arabic Abstract

في ضوء تطور التقنية في هذا العصر، أصبح من الممكن أن تتعرف الأجهزة على الكلمات المنطوقة من خلال الصوت وتحويلها لنص مكتوب. لكن هذه التقنية تقل جودة مخرجاتها أو تفشل في بعض الحالات، مثل: البيئات الصخبة أو تواجد أكثر من متحدث أو ضعف الصوت أو عدم وجوده. إحدى الحلول الممكنة التي تعمل في هذه الحالات ومن الممكن استخدامها في تحسين جودة التعرف على الكلمات المنطوقة هي قراءة الشفاه.

نهدف من خلال هذا المشروع إلى تطوير نظام قادر على التعرف على الكلمات العربية المنطوقة من المتحدث دون السماع للمتحدث وهو يتكلم والاعتماد على قراءة الشفاه من مقطع فيديو. هذه الفكرة غير جديدة كلياً، بل يوجد أنظمة مطورة للغات أخرى مثل اللغة الإنجليزية، لكن لا يوجد أي أنظمة مشابهة مصممة لنطق مفردات اللغة العربية. لذلك، تم تصميم برنامج يستطيع استقبال مقطع فيديو لشخص وهو ينطق رقم باللغة العربية واستخراج الرقم المنطوق باستخدام المعلومات البصرية فقط. يقوم البرنامج بالعمل على مبدأ التعلم العميق وعلى عدة خطوات لمعالجة المقطع المدخل.

Table of Contents

Table of Figures

Table of Tables

Chapter 1: Introduction

Speech recognition refers to the computer's ability to recognize human speech. It can be implemented either based on visual data (video input), vocal data (audio input), or even both. The most common approach uses audio input for its ease of use.

In many cases, voice-based speech recognition fails to achieve accurate results or any results at all. This failure can be caused by recording in noisy environments, having more than one speaker speak at the same time, lack of voice in the video, or the speaker is very distant. In these cases, lipreading will prove to be much more beneficial than the more traditional way of speech recognition.

Many systems exist to solve the speech recognition problem. However, none of these systems combined the Arabic language, visual speech recognition, and deep learning. And this paper aims to build one that satisfies these conditions.

1.1 Problem Statement

The main issue facing voice-based speech recognition is the low accuracy in some environments. In most cases, voice-based speech recognition is designed to take audio input. This may result in bad accuracy if the audio is distorted.

Visual speech recognition aims to solve these issues by relying on video instead of audio. However, no deployable visual speech recognition systems were built for the Arabic language.

1.2 Goals and Objectives

The goal of this project is to be able to lip-read the Arabic language from videos of a person speaking without relying on audio and output the spoken words as text.
The following objectives are proposed to achieve this goal:

1. Create a video dataset that will facilitate training the proposed model
2. Research and choose the appropriate machine learning model for the lipreading problem
3. Design and implement a machine learning model to identify spoken words
4. Design an integrated system that accepts video input and output the spoken words

1.3 Solution

To avert the problems that face voice-based speech recognition, the proposed system will not rely on audio input; instead, it will rely on video input. This will allow it to work even in environments where there is no audio, or the audio is distorted. The system will be designed to take video input of a person or more speaking in Arabic, and through the use of machine learning, extract what was spoken as a text output.

As this system will be implemented utilizing machine learning approaches, the system needs data to train on. After an extensive search, no dataset for Arabic speakers was found. Consequently, the dataset has to be created from scratch. This will be achieved by recording Arabic-speaking volunteers. All of the created data will be labeled, cropped, rotated and any other form of preprocessing needed will also be applied, giving a structured dataset. Through the use of this structured dataset, different models will be designed, tested, and optimized to choose the highest accurate model. After that, the model will be encapsulated with a user-friendly interface to allow the system to take video input from the user, and output comprehensible text to the user.

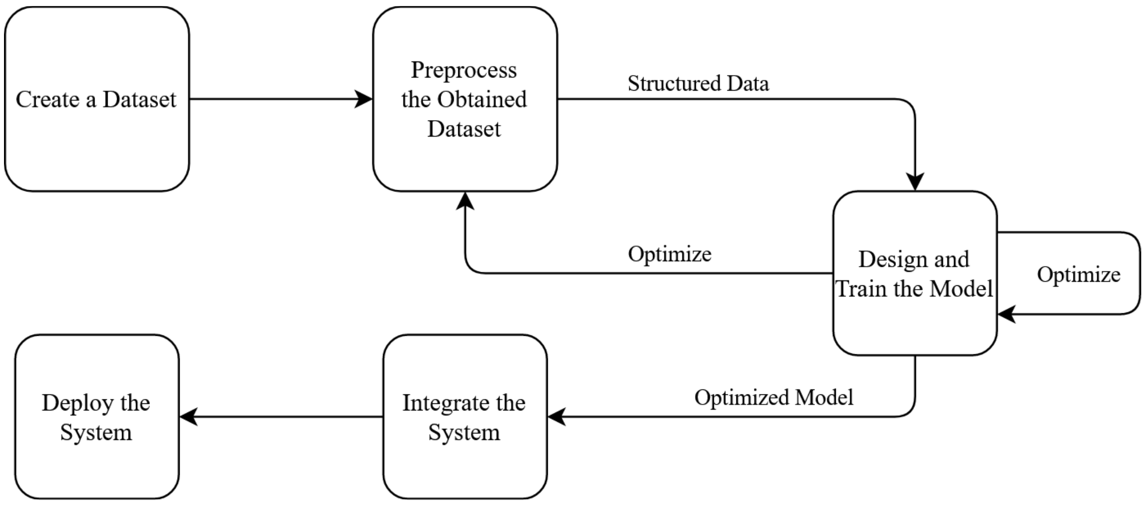


Figure 1: Process of the proposed system.

1.4 Research Scope

This system will be able to extract spoken words from a video input of a single speaker, and due to the limited data availability and the lack of processing power, the vocabulary will be limited to numbers from 0 to 9.

Chapter 2: Background

This chapter provides context to the information that is discussed in this document.

2.1 Speech recognition

Speech recognition is the ability of computers to recognize spoken language. Speech recognition can be achieved by either audio, visual, or both. Voice-based speech recognition is the most commonly used form because of its ease of implementation and its relatively high word correct rate (WCR). However, voice-based speech recognition accuracy declines if the audio quality is bad or the environment is noisy. On the other hand, Visual based speech recognition relies on video input, making it function in environments where audio will not suffice, at the price of higher computation power and lower accuracy. These two approaches can be combined and provide the benefits of both forms and avoid their drawbacks.

This paper focuses mainly on video-based speech recognition, which is also referred to as speech reading, visual speech recognition, or lipreading. Lipreading refers to the ability to distinguish speech from watching the movement of a person’s lips and facial expressions [1].

2.2 Natural Language processing

Natural language processing is a field of machine learning focusing on the ability of computers to process and analyze human language [2]. Natural language processing is widely used in day-to-day programs and especially in mobile phones. Some uses of natural language processing are voice-to-text programs, keyboard auto-completion, or picture-to-text programs.

2.3 Computer Vision

Computer vision refers to the computer’s ability to analyze and understand images and videos [3]. It aims to mimic the abilities of the visual system in human beings such as identifying faces, objects, movement, and more. These functionalities can be utilized to automate or ease tasks that usually require a human eye; for example, reading X-ray images, detecting pedestrians, and lipreading.

2.4 Machine learning

Machine learning is a field of computer science that solves problems by studying data from the problem and constructing a statistical model based on the data [4].

Regarding this paper, there are two main types of machine learning, supervised learning and unsupervised learning. Supervised learning is used when the training data is labeled and creates a model that labels the given data. On the other hand, unsupervised learning is used with unlabeled data and creates a model that can detect find patterns on its own [4].

There are three machine models mentioned in the paper which are k-Nearest Neighbor (kNN) is a model that classifies the input based on the most similar inputs to a given category as an output [4], Support Vector Machine (SVM) is a model used to solve categorization problems by constructing a plane or barrier between the different categories [4] and Hidden Markov Model is a statistical model that gives the probabilities of the output based on the input.

2.4.1 Training

Training refers to processing the input data and manipulating the model to reflect the input data as accurately as possible. During training, the model constantly changes to achieve the best result from the given data. However, sometimes the model adheres to the training data too closely, this is called overfitting. On the other side, when not enough data is given, the model underfits. Both overfitting and underfitting cause the model's performance to decline [4].

One way to avoid overfitting is by using cutout training. It refers to randomly blocking parts of the input data, which forces the model to make use of other parts of the picture more efficiently and blocks other parts making it harder to rely on a single collection of data, thus decreasing the chance of overfitting [5].

Training the model from scratch requires time and processing power that can be very hard to provide. This can be cut down by allowing the model to mimic an already trained similar model; this is called transfer training. Another approach is to use principal component analysis, which is an algorithm for dimensionality reduction. This means that if the dataset features are very large, it can reduce them to a smaller number of features and accelerate the process [4].

2.4.2 Evaluation

After training the model, it is necessary to find out how well the model generalizes to unseen data. There are many different tools to measure the model's performance. Each tool is better used for a specific kind of problem. For example, confusion matrices are used to represent the accuracy of each class of inputs based on the corresponding output, root mean squared error (RMSE) is used to represent the error of the estimation from the actual value [4].

2.5 Deep Learning

Deep Learning is a form of machine learning that aims to imitate the human brain in performing complex tasks such as self-driving cars using neural networks. Neural networks are a set of connected layers; each neural network has at least one input layer and one output layer. It also is possible to add hidden layers in between. Each layer receives input data from the previous layer and outputs the data after processing it to the next layer. Each type of layer processes the data differently. For example, the SoftMax classification layer is used to squash the raw class scores into normalized probabilities for each class.

Stacking different layers will create different neural networks. Each of these combinations serves a different goal. For instance:

1. Convolutional Neural Network (CNN) is a neural network that reduces the number of parameters without losing too much quality by reducing neighboring pixels in an image. And it is possible to expand this principle to videos by using Spatiotemporal CNN (STCNN) which also reduces neighboring frames.
2. Recurrent Neural Network (RNN) is a type of neural network that aims to generate, label, or classify a sequence. However, the main problem in RNN when dealing with long sequences is that the features in the beginning tend to be forgotten. The solution to this problem is to store the features for future use. This can be achieved by using gated RNN [4]. These include long short-term memory (LSTM) and networks based on the gated recurrent unit (GRU). It is also possible to make the network remember in both directions by making it bi-directional.

After building the neural network, loss functions come to use by calculating the gradients. Gradients are used to update the weights in the neural network. And that's basically how the networks learn [4]. There are many loss functions in neural networks, the one mentioned in this paper is Connectionist Temporal Classification (CTC) loss is a loss function to train neural networks where having aligned data is an issue, like in speech recognition.

Chapter 3: Literature Review

There is an increasing interest in visual speech recognition lately since it contributes greatly to the speech recognition problem. Fernandez-Lopez and Sukno [6] introduced an architecture that combines Convolutional Neural Networks (CNNs) and Long-Short Term Memory (LSTM) which can be used for training with a small-scale dataset. In the training stage, they had around 15 million parameters and 1200 sequence samples for training. The ratio between parameters and data is very large which makes it time-consuming to train the network. Therefore, they split the training into modules: visual and temporal. They also relied on weak labeling because CNN only needs to distinguish among visually separable classes, so the labels can be generated automatically in a way that can still be informative about the mouth appearance. The Classification rate (CR) obtained by the CNN module was 47.67% in the visual module. However, in the temporal module, the average CR was 91.38% because the temporal module outputs the spoken phrase it shows the performance of the whole system. The proposed approach would be beneficial for small datasets with large parameters.

Noda et al [7] followed a similar approach by utilizing a convolutional neural network to extract features for VSR for recognizing phonemes, then with a hidden Markov model (HMM) recognizes the word that was spoken. Their dataset was made of 300 Japanese words collected from six different speakers. After the evaluation, the result was their system recognizes vowels better than consonants, the mean for the vowel recognition is 60-100%, whereas for the other phonemes mean recognition is 20-80%. The differentiation between phonemes and consonants is hard because of the bilabial consonants such as b, p, or m. To differentiate between them is to calculate the time for the spoken phoneme because consonants take less time than vowels. This approach supports differentiating vowels and consonants to get more accurate recognition.

A different model was also tested by Garg et al in [8] explored lip reading for words and phrases utilizing machine learning using multiple models. These models were based upon VGGNet, which is a model that identifies objects from a picture by calculating the depth of each object. These models include concatenating first *k* images and then classify it using VGGNet (they tried multiple structures using 2D arrays, 2x2 array, 3x3 array, 4x4 array, 5x5 array, and 5x5 stretched array. The difference between the 5x5 array and 5x5 stretched array is the latter tries to fill the missing images with the nearest image instead of zeros). The models also include training a small model from scratch using the concatenated images, and handling sequence-length using LSTM layers which take the feature vectors from the VGGNet. They collected their data from low-frame-rate videos. They faced an issue with the older models because of the sequence length of the collected data. After they evaluated the models, the best performing model was the concatenated model with interpolation (5x5 stretched array), and it scored a validation accuracy of 76% on both words and phrases surpassing all the other models because of the filled images that increased its accuracy. This proposed work helps identify the speaker’s face and lips, then using the tested method and check if it works with our dataset.

Yet another different model was discussed by Chung et al [9]. They explored lipreading by utilizing RNN in their model which was based on the recent sequence-to-sequence (encoder-decoder with attention) which was developed for speech recognition and machine translation. They collected their data from the LRS corpus which when evaluated surpassed a professional lip reader on videos from the BBC. They divided their approach into 3 parts. First, an image encoder using CNN to generate the features and LSTM to produce the output. Second, an audio encoder using LSTM to ingest 13-dimensional MFCC features in reverse and produce the states and the output. Finally, a character decoder using LSTM to translate what was spoken. Their training strategy was training single words examples only, and then and then let the sequence length grow with the training. This strategy has reduced overfitting because it works naturally to augment the data. The evaluation of their strategy was divided into three metrics, CER (character error rate), WER (word error rate), and BLEU (bilingual evaluation understudy score, where 1 is a perfect match and 0 is a mismatch). The result for the image encoder was 59% for CER, 67.5% for WER, and a BLEU score of 0.238. The proposed work can be utilized in experimenting with more models to check for the highest accuracy model.

Similarly, Wand et al [10] explored lip reading by utilizing a compact neural network that consists of chained feed-forward layers and LSTM layers. They used the GRID (consists of video and audio recordings of 34 speakers saying 1000 sentences each) dataset for the experiment. Their method was creating eigenlips features from raw frames by computing the PCA decomposition on the training data, then transforming the images obtained by multiplication with the PCA matrix, resulting in several dimensions sorted by maximal variance. They evaluated multiple methods, eigenlips with SVM, histogram of a gradient (image depth from color gradients), and LSTM. With an accuracy of 69.5%, 71.2% and 79.5% respectively. Their proposed work explores multiple methods to compare them and determine which one results in the highest accuracy.

Afouras et al [11] compared two models for lipreading, one using Connectionist Temporal Classification (CTC) loss and the other using a sequence-to-sequence (seq2seq) loss. And investigate to what extent lip reading is complementary to audio speech recognition. In the LRS2-BBC dataset, TM-CTC was achieved in a normal environment with 65.0% word error rate (WER) in a video only, 15.3% WER audio-only, and 13.7% in video and audio. In a noisy environment achieved 64.7% WER in audio-only and 33.5% in video and audio. On the other hand, TM-seq2seq achieved in a normal environment 51.2% WER in a video only, 10.5% WER audio-only, and 9.4% in video and audio. In a noisy environment achieved 58.5% WER in audio-only and 35.9% in video and audio. They proved that using lipreading with audio speech recognition improves the text generated. Also, TM-CTC is useful in a noisy environment but TM-seq2seq surpasses TM-CTC in a normal environment.

In another comparison, Afouras et al [12] studied different architectures' accuracy and training time. The first model was a recurrent model using LSTMs, the second was a fully convolutional (FC) model, the third model was a transformer model. The recurrent and FC models are trained with CTC, the transformer is trained with the seq2seq model. As the result stated, Bidirectional LSTM (BL) model achieved 62.2% WER in 4.5 days, the FC model with 15 layers achieved 55% WER in 3.4 days but with 10 layers achieved 57.1% WER in 2.4 days, the transformer model achieved 50% WER in 13 days. The results in this work can be helpful to choose the approach of the desired model.

While [9] studied time challenges while training, Shin et al [13] tried to solve two design issues in real-time lipreading. The first one is it is hard to keep track of a moving lip. The second one is a robust word classifier. Solving the first issue by using the active appearance model (AAM) where the input face is simplified to dot and line for locating the lip, for the outer lip is tracked by using the model-based Lucas Kanade method, and the inner lip is analyzed by the fast block matching method. For robust word classifiers, they used kNN classifiers because the other method, hidden Markov models (HMM) or artificial neural networks (ANN), require a huge amount of training data and require extensive retraining for each new word class added to the database. The kNN word classifier with a speed of 15fps achieved a 92.67% word correct rate (WCR) for person-dependent tests, and 46.50% for person-independent tests. However, during the person-dependent testing, the ANN-based outperformed the HMM-based classifier by 7.09% and kNN-based by 0.49%. However, the person-independent ANN-based classifier outperformed the HMM-based and kNN-based word classifiers by 17.90% and 7.57%, respectively. This work is beneficial in advancing the software to lipreading in real-time.

LipNet [14] is an end-to-end sentence level that uses deep learning for speech recognition. The system uses multiple stacked layers consisting of three layers of Spatiotemporal Convolutional neural networks, followed by two bi-gated recurrent units, and the final layer is a connectionist temporal classification layer. LipNet was trained on the GRID corpus dataset giving a total of 32746 videos to train on, these videos were cropped to show only the mouth of the speaker and were also mirrored to give more training data and to avoid overfitting. To deal with similar-looking phonemes, LipNet uses nine confusion tables that show which phonemes are most likely to get confused with each other. LipNet proved to be very accurate compared to professional lip-readers because it was able to train on a big dataset and utilize confusion tables that were developed by professional linguists. However, both of these are going to be harder to get for the Arabic language. The goals of LipNet match the goals of this paper very closely. Thus, building on the system architecture used in it will improve the results of this paper greatly.

In this paper [15], the authors built an audio-visual sentence-level speech recognition system. The system is split into two different machine learning networks: audio and visual. The visual speech recognition system is completely based on the approach described in LipNet [14], except it was retrained on a smaller Urdu dataset. The audio speech recognition software consists of Long Short-Term Memory Network and a SoftMax Classification Layer. The visual system was retrained using an Urdu dataset. The dataset contained a total of 2000 videos from 10 speakers each repeating 10 words 10 times and doing the same for 10 phrases. The mouth portion of the videos was cut using external tools and given to the NN. They also used an Urdu digits dataset. The authors were unable to train their model to function on the phrases because the data was insufficient. However, they managed to train their audio system to function with an accuracy ranging between 56% to 72% depending on the dataset and the model used. This research work shows how the availability of data is very important when developing machine learning systems. Lacking data can cause a network to fail even if the network proved to work before. It also gives a lower limit of data to take into consideration when collecting the data.

Furthermore, [16] studies the difference in using different regions of interest (RoI) in visual speech recognition's accuracy. These regions are not limited to lips only, but they also include different areas from the face. Such as the cheeks, eyebrows, and even the nose. The authors also tested to see how the size of the region impacts the accuracy, and how using the cutout training technique affected accuracy. The authors used LipNet [14] to test the impact on sentence-level accuracy and used 3D-ResNet18 to test the impact on word-level accuracy. Three different datasets were used in this paper, the LRW, LRW-1000, and GRID. Each video was preprocessed, cropped, and aligned using its landmarks. The most successful RoI was using the whole face and aligning its rotation using nose and eye landmarks. The testing showed that the network automatically focused on the region of the lips and almost completely ignoring the rest of the face, to counter this, the authors used the cutout technique forcing the network to focus on other parts of the face. This gave an increase in accuracy in shots where the face's rotation was less than 40◦. This study gave insight into the importance of using a correct RoI in visual speech recognition. The results of this paper will help improve the selection of an RoI and increase the projected accuracy of the built model.

Additionally, [17] proposes using an audio speech recognition model to help train a lip-reading system using knowledge distillation. By using this technique, the authors aim to increase the accuracy of visual speech recognition by using audio, which has more data available than visual, to help in training. The lip-reading system uses the same architecture proposed by LipNet [14]. Three audio speech recognition systems were used. These systems share the architecture with the lip reader, but the second and the third differ in some layer resolutions and count. The visual system was initially trained only on CMLR and LRS2 datasets, while the audio systems were also trained on aiShell and LibriSpeech datasets. The main problem facing the authors was the lack of synchronization of the audio points and the video frames. This was solved by using multiple synchronization points based on the frames and the projected character by the audio system. As part of training the visual system, knowledge distillation was applied once for each different audio system. Even though the second system was more accurate on its own, the student of the first system, which had an identical architecture, was more accurate. The proposed model can help in training the model by relying on different data sources. This will help solve the lack of data for the training of the model.

Another approach to lipreading is to identify phonemes instead of trying to identify whole words or sentences. These phonemes are then grouped to form words and sentences. The grouping was achieved by comparing the phonemes predicted by the model to a known database and choosing the best-matching word. [18] attempted to implement this approach. However, the implementation lacked accuracy for many reasons; the model was trained on one speaker only and the model outputs phonemes, but the accuracy is measured for words. Resulting in compounding the model's error rate. On the other hand, this approach is simpler than identifying words or sentences. Consequently, less data is required to train the model.

Chapter 4: System Analysis

4.1 Functional Requirements

1. The system should be able to predict the spoken words from the input video without relying on audio.
2. The user should be able to select a video file from a storage space.
3. The user should be able to record a video.
4. The system should be able to display the preprocessing procedure to the screen as videos. The videos represent the detected face and the cropped region of interest.
5. The system should be able to display the output text to the screen.
6. The user should be able to upload another video after finishing processing the current video.
7. The system should be able to display the processing progress bar.

4.2 Non-Functional Requirements

1. The system shall be implemented using TensorFlow.
2. The input video must contain a single speaker only and is no longer than 75 frames at 30 frames per second.
3. The system shall be able to find a face in the input videos.
4. The system shall be able to preprocess the received video and reshape it to the desired region of interest.
5. The system shall be able to recognize only spoken Arabic numbers.

Chapter 5: System Design

5.1 System use-cases

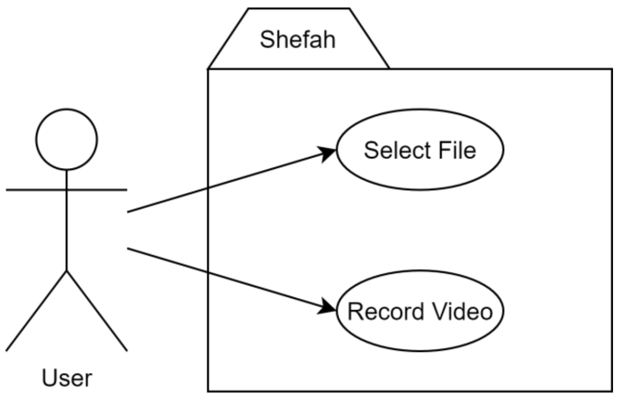


Figure 2: Use-case diagram

The following use-case diagram represents the main functions provided by the system.

<u>Use case name: Select a file</u>
Actor: User
Precondition: No video is under processing, and the user is not recording.
Postcondition: Input video, detected face, region of interest, and predicted number are displayed on the interface.

User’s actions	System’s responses
1. The user presses the “Select file” button.	1. The system responds by displaying the file selection dialog.
2. The user selects a file. **Alt1**	2. The system responds by displaying the input file and starts preprocessing the video and displaying the detected face and facial landmarks. When the preprocessing is done, the system will start predicting the word and displaying the result on the interface. Two progress bars will be displayed. One for the current process, the other for the total process. And the progress bars will be updated accordingly. **Exp1 Exp2**

Alternative1: The user directs the system to cancel the transaction. The use case ends.
Exception1: The chosen file is not a video or not supported. The use case ends.
Exception2: No face detected. The use case ends.

Use case name: Record a video
Actor: User
Precondition: No video is under processing, and the user is not recording
Postcondition: Recorded video, detected face, region of interest, and predicted number are displayed on the interface.

User's actions	System's responses
1. The user presses the "Record a video" button.	1. The system responds by opening the default camera in the device, displaying the camera's video stream on the screen, and hiding the "Record a video" and show the "Start recording" button with the "Stop Recording" button being disabled. **Exp1**
2. The user presses the "Start recording" button.	2. The system responds by starting the recording, disabling the "Select File", and "Start Recording" buttons, and enables the "Stop Recording" button.
3. The user presses the "Stop recording" button.	3. The system responds by stopping the recording, displaying the recorded video, turning the camera off, hiding the "Start Recording" and "Stop Recording" buttons, showing the "Record a video" button, and enabling the "Select File", and starts preprocessing the video and displaying the detected face and facial landmarks. When the preprocessing is done, the system will start predicting the word and displaying the result on the interface. Two progress bars will be displayed. One for the current process, the other for the total process. And the progress bars will be updated accordingly. **Exp2**

Exception1: The camera cannot be reached. The use case ends.
Exception2: No face detected. The use case ends.

5.2 Interaction Diagrams

The following diagram shows how the user and the components of Shefah interact with each other.

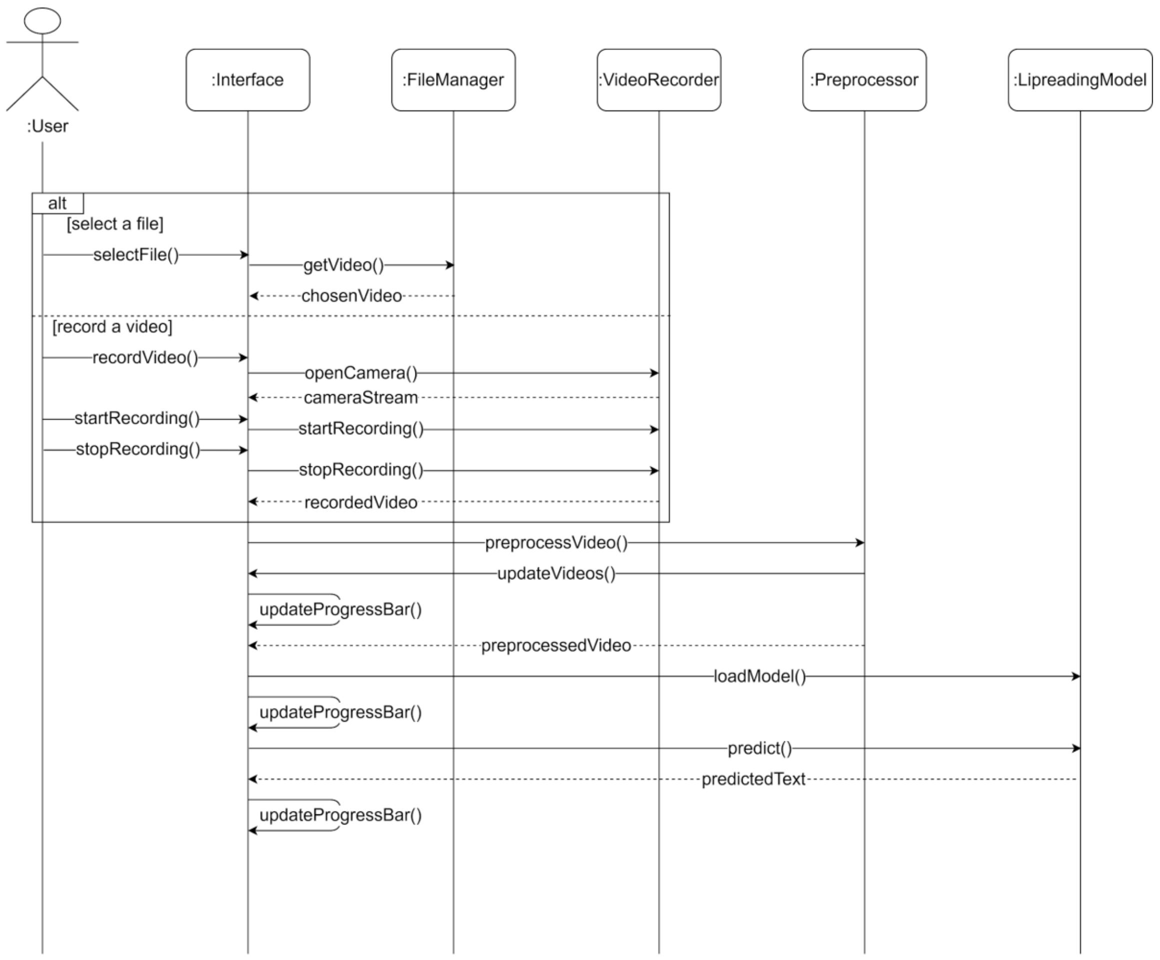


Figure 3: Interaction diagram

5.3 Class Diagram

The following figure shows each component's functions and how they connect with each other.

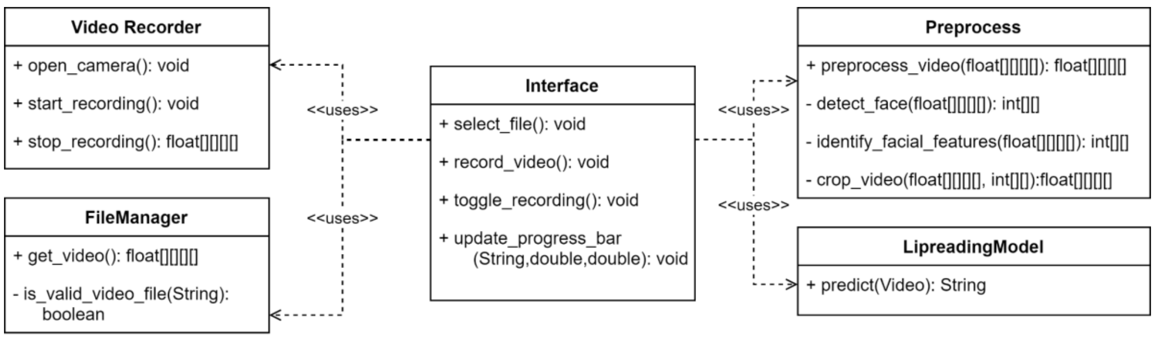

Video Recorder
+ open_camera(): void
+ start_recording(): void
+ stop_recording(): float[][][][]
FileManager
+ get_video(): float[][][][]
- is_valid_video_file(String): boolean
<<uses>>
<<uses>>
Interface
+ select_file(): void
+ record_video(): void
+ toggle_recording(): void
+ update_progress_bar (String,double,double): void
<<uses>>
<<uses>>
Preprocess
+ preprocess_video(float[][][][]): float[][][][]
- detect_face(float[][][][]): int[][]
- identify_facial_features(float[][][][]): int[][]
- crop_video(float[][][][], int[][]):float[][][][]
LipreadingModel
+ predict(Video): String

Figure 4: Class diagram

Interface:

1. select_file(): Prompts the user to select a video file through a file selection dialog. Display a pop-up If an error occurs during the selection.
2. record_video(): If the system is currently not recording and the camera is not enabled, the system camera is enabled and displayed on the screen, the "Record a video" button is hidden, and the "Start recording" and "Stop Recording" buttons are shown with the latter button being disabled.
3. toggle_recording(): If the system is not recording, the system starts recording from the camera, disables the "Start Recording" and "Select a file" buttons, and enables the "Stop Recording" button. However, If the system is recording a video, the system will stop recording, turn the camera off, display the recorded video on the interface, re-enable the "Select File" button, hide the "Start Recording" and "Stop Recording" buttons, and show the "Record a Video" button.
4. update_progress_bar(String, double, double): Receives three parameters: A string representing the name of the current step of processing, a double representing the progress of the current task's progress bar, and a double representing the total progress. The progress bars are updated according to the parameters.

FileManager:

1. get_video(): Prompts the user to select a video file through a file selection dialog and returns the result. Throws an exception if the user cancels or selects a file that is invalid or cannot be read.
2. is_valid_video_file(String): Receives a file path and returns whether the given path represents a readable path and if the path points to a supported video file.

VideoRecorder:

1. open_camera(): Turns on the default camera of the system and returns the camera data stream. Throws an exception if no camera is found, or if the camera is already running.
2. start_recording(): Starts saving the data from the default camera's stream to a temporary folder. Throws an exception if no camera is running or if the system is already recording.
3. stop_recording(): Stops saving the data, turns off the default camera, and returns the recorded data. Throws an exception if the system is not recording.

Preprocess:

1. preprocess_video(float[][][][]): Receives a video as input, detects any faces in the video, crops the video to match the region of interest for the face, and returns the cropped video. Throws an exception if the video does not contain a face, or if it contains more than one face.
2. detect_face(float[][][][]): Receives a video, and for each frame in the video, detect any faces in the given frame, and store the largest detected face position and size into a two-dimensional array using the frame number as a first index and return the array.
3. identify_facial_features(float[][][][]): Receives a video cropped to the size of a single face, and for each frame in the video detect the facial landmarks and store the location in a two-dimensional array using the frame number as the first index and return the array.
4. crop_video(float[][][][],int[][]): Receives a video, and a two-dimensional array representing the location of the desired area. Crop the video to a specified location and return the result.

Model:

1. predict(float[][][][]): Receives a video cropped to the region of interest. Predicts the spoken words and returns the predicted text.

5.4 System Architecture

As in Figure 5, this is how the system architecture is.

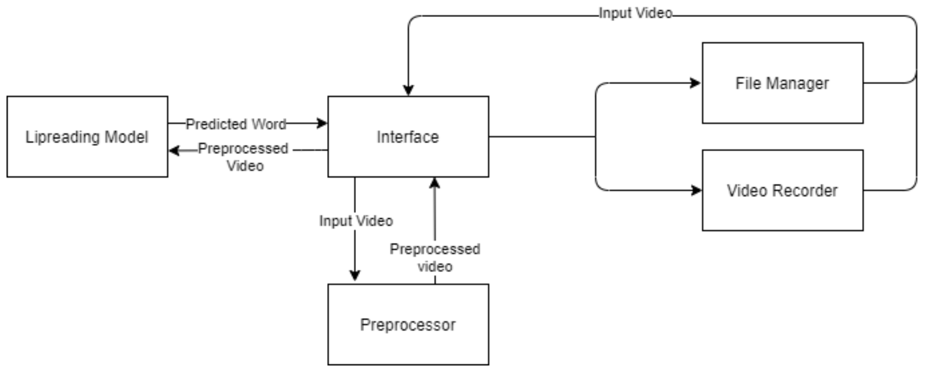


Figure 5: System architecture

Interface: displays the user interface (UI), handles the user input and serves as the main logic controller by connecting the other components.

File Manager: handles the logic of reading videos from files, and assuring they meet the required specifications.

Video Recorder: handles the operations needed to operate the system's camera, allows the recording of videos directly in the application.

Preprocessing: handles the logic of detecting the region of interest in the input video and cropping it accordingly. To achieve this, Dlib's face detector and 68-point landmark predictor are used.

Lipreading Model: Handles the logic of initiating TensorFlow, creating the model, and predicting results.

5.5 Database Design

Due to the nature of the problem being solved, the system does not require the use of a database.

5.6 User Interface

To make the use of Shefah easier and friendly, an interface will be implemented. As shown in Figure 6. At the start of the program, the user can choose to either select a video or record a video. When the user selects a video or records a new one, an alert will appear as in Figure7 showing the restrictions on using Shefah to guarantee that the input video matches Shefah's requirements.

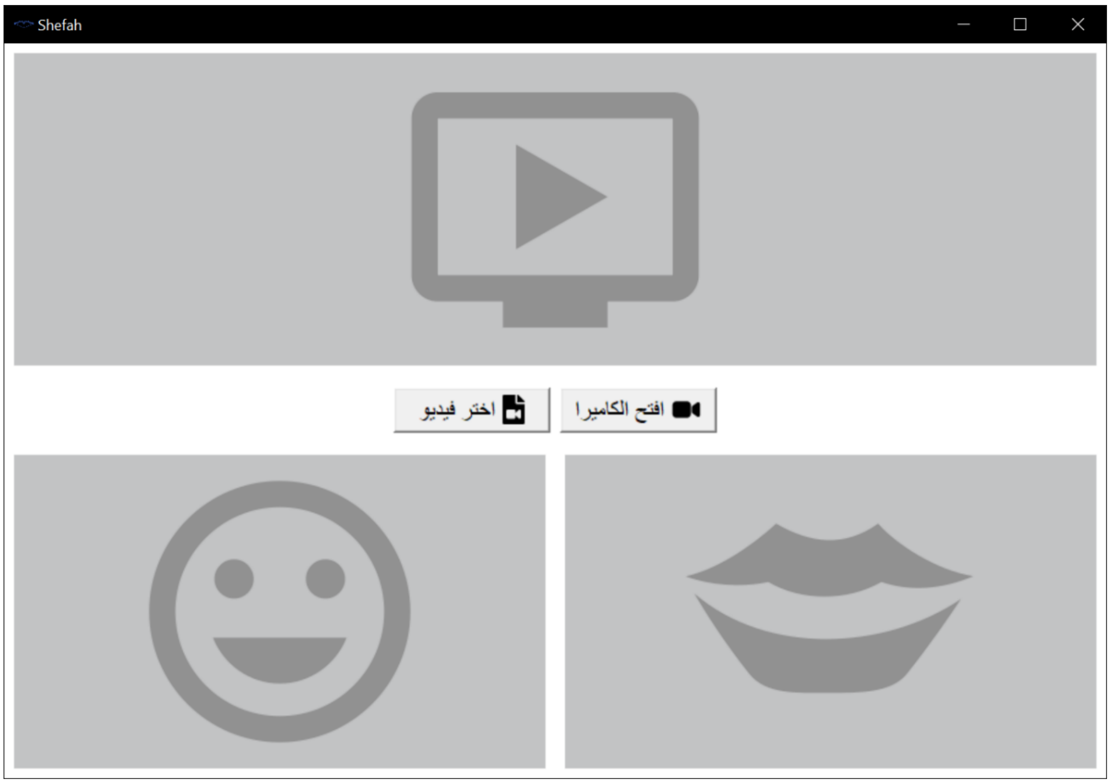
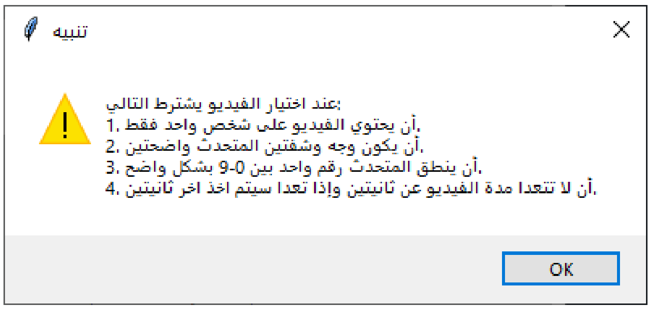


Figure 6: Shefah's Interface



تنبيه

عند اختيار الفيديو يشترط التالي:
1. أن يحتوي الفيديو على شخص واحد فقط.
2. أن يكون وجه وشفتين المتحدث واضحتين.
3. أن ينطق المتحدث رقم واحد بين 0-9 بشكل واضح.
4. أن لا تتعدا مدة الفيديو عن ثانيتين وإذا تعدا سيتم اخذ اخر ثانيتين.

OK

Figure 7: Warning message

After clicking “choose file” a warring message appear as in Figure 7. Choosing a file should meet these requirements.

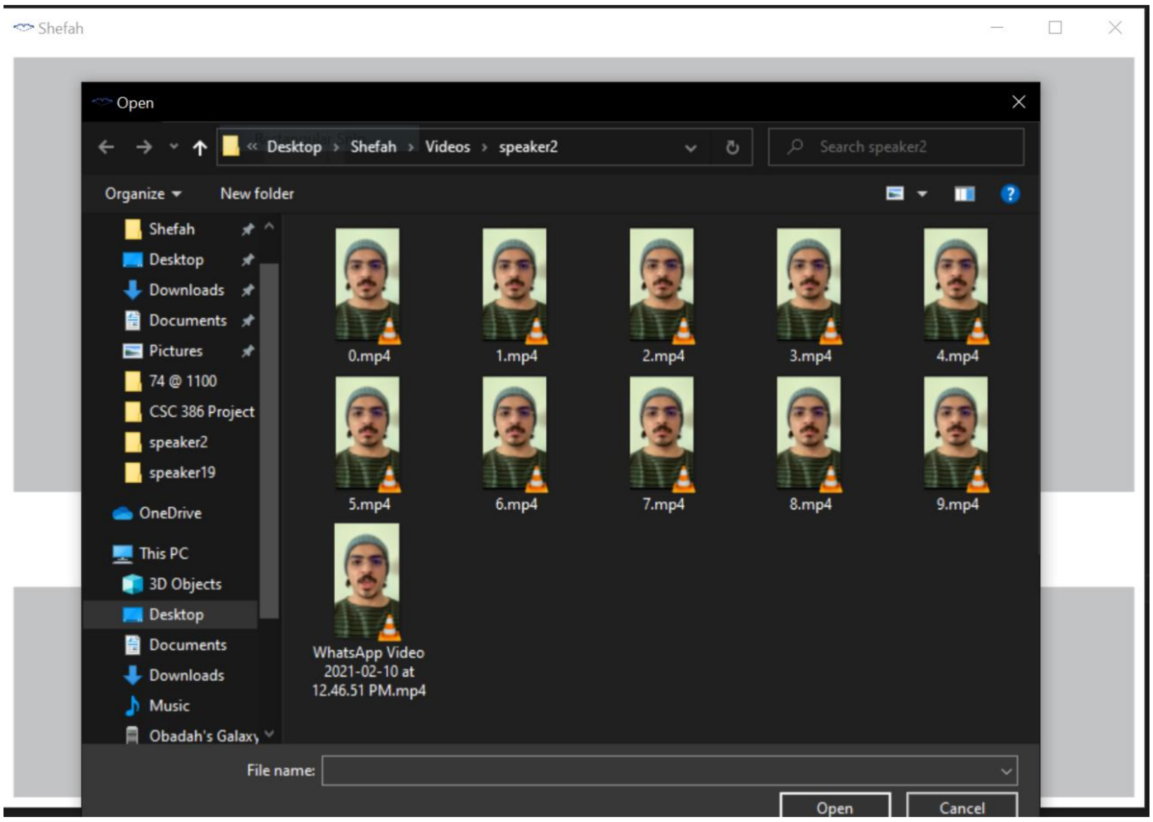


Figure 8: File dialog

After that, when the user chooses to select a file, a file dialog will appear As in Figure8.

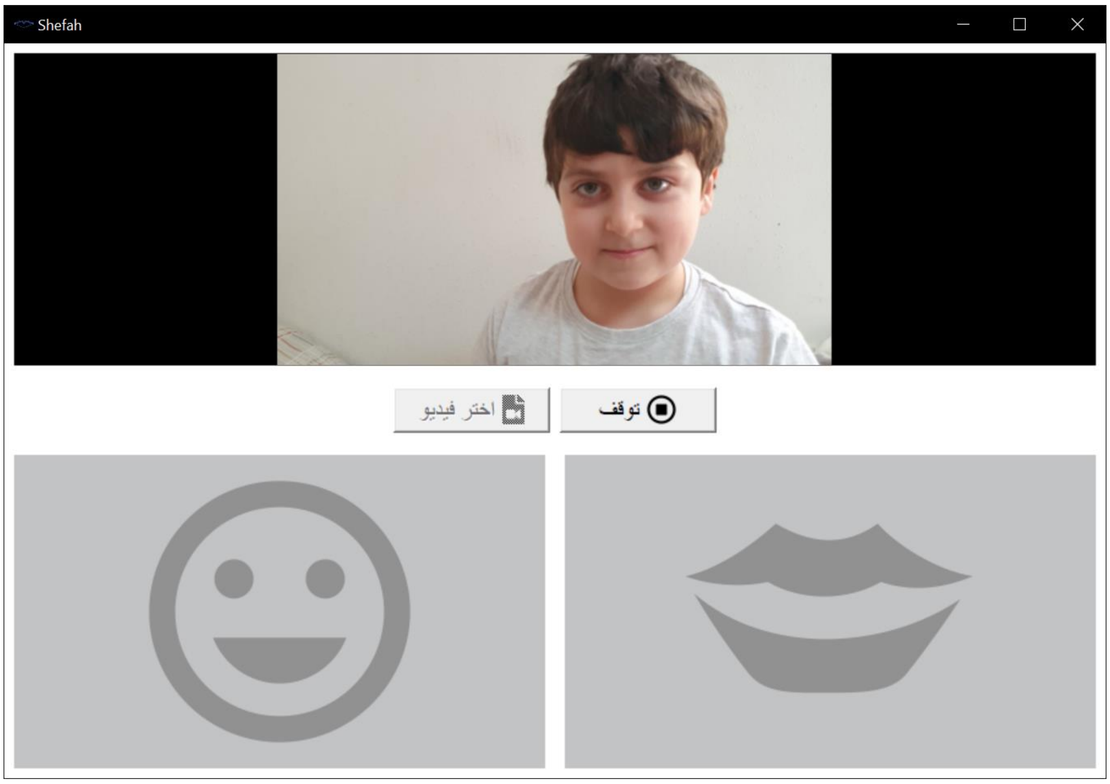


Figure 9: While recording

Alternatively, when the user chooses to record a video, the program opens the camera, and the button “record a video” disappears, and the “start recording” and “stop recording” buttons appear with the latter being disabled. as shown in Figure #. After the user starts recording, the “stop recording” button will be enabled and disables the “Select file” button. Then, the user is ready for the next step.

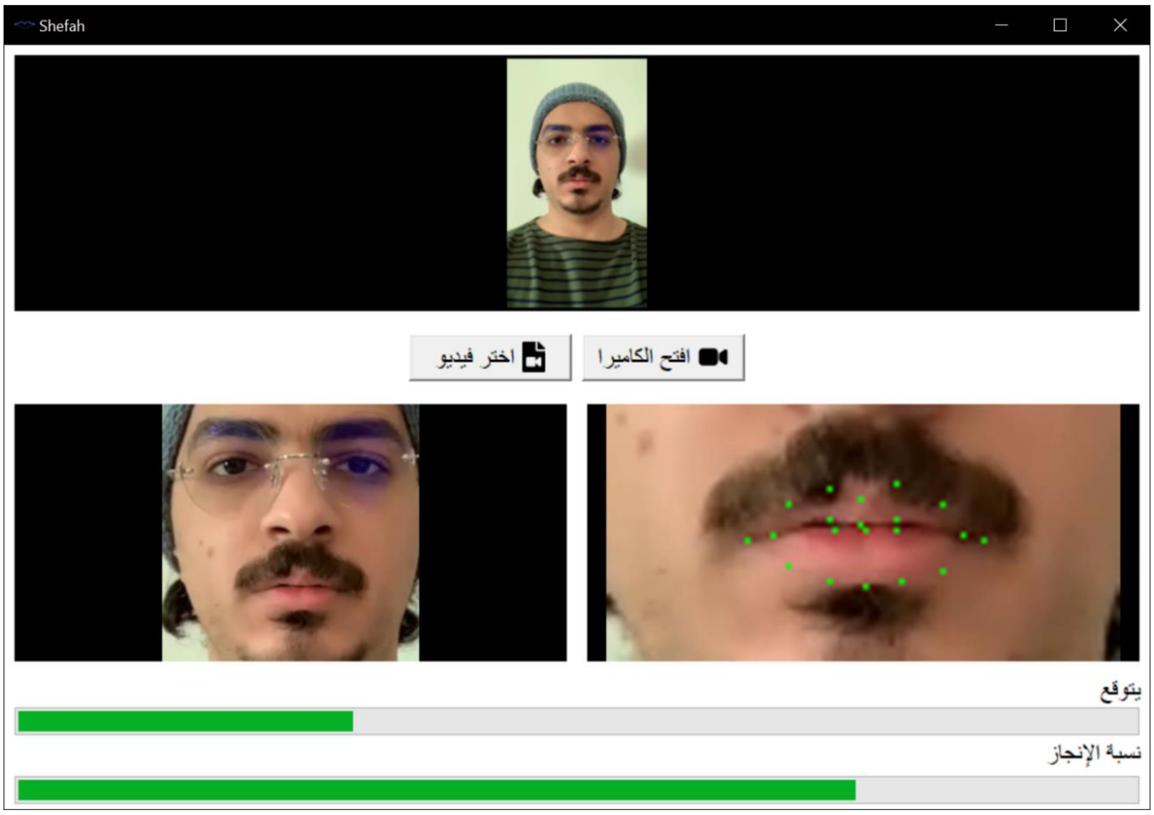


Figure 10: Processing

After choosing the input, the system automatically sends the video to Shefah's model to preprocess and predict what was spoken. With that, the two progress bars will appear as in Figure 10 representing the current activity progress and the total progress, respectively. After finishing the face detection activity, the detected face will be displayed in the bottom left corner. While the bottom right corner will display the cropped region of interest. Finally, the result will be displayed as text below the progress bars as shown in Figure 11.

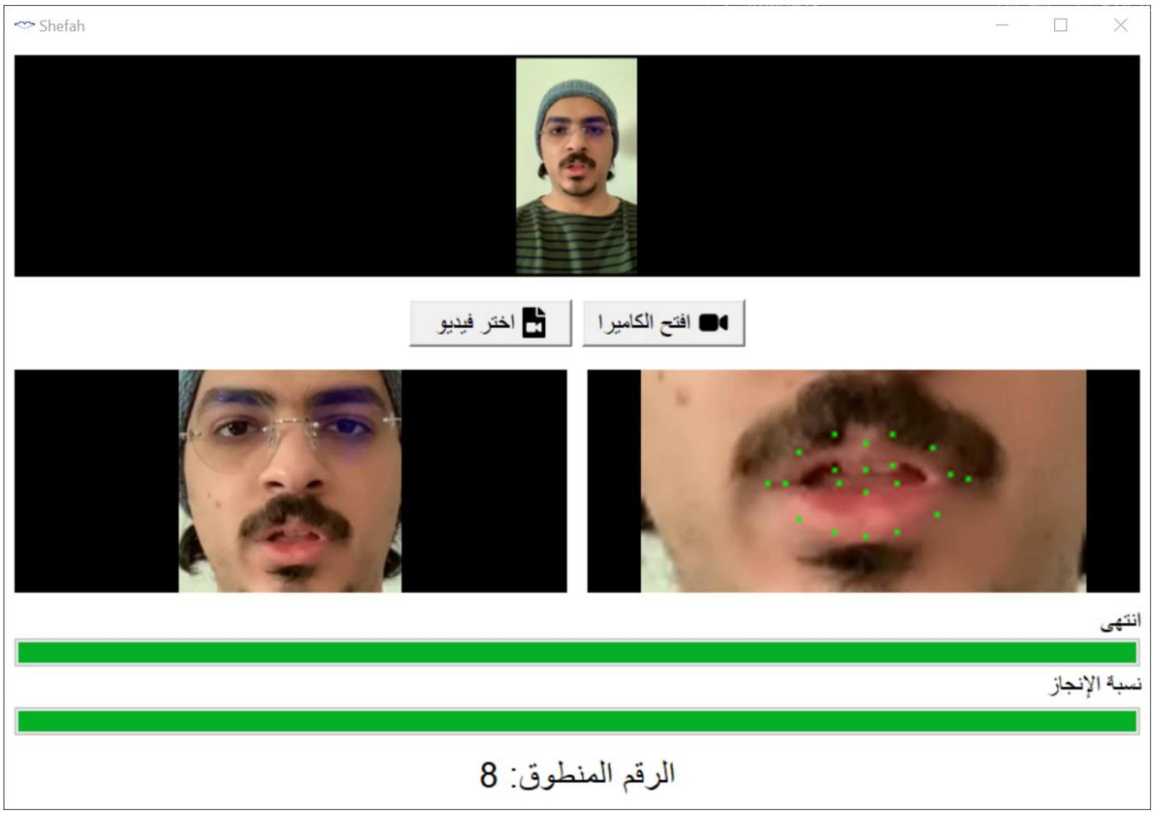


Figure 11: Showing the result.

After processing and showing the result, the user can choose either to select another video to predict or exit Shefah.

5.7 Algorithms

This section discusses the used model, and the main algorithms used to prepare the data before using it in the model. To help the model achieve the best possible score, the data should be minimized and preprocessed. The following pseudo-code shows the basic structure for the preprocessing operation.

```
preprocess(video){
   face_detection = detect_faces(Video)
   if(face_detection.shape.y != 1)
      throw new exception()
   first_face_location_and_size = face_detection[][0][]
   cropped_face = crop_video(video,first_face_location_and_size)

   facial_features = identify_facial_features(cropped_face)

   frames = segement_video(cropped_face)
   foreach(frame in frames){
      foreach(feature in facial_features[frame]){
         if(feature > 48) //points representing the mouth have an index of 49 and
above
      }
      average = average(mouthPos)
      saveFrame(croppedFace.crop(average,roi_size))
   }
}
```

As seen from the code, the preprocessing is done in three steps:

1. Detect the location of the face in the video.
2. Extract the facial landmarks for the face.
3. Detect the region of interest and crop the video to size around the region of interest.

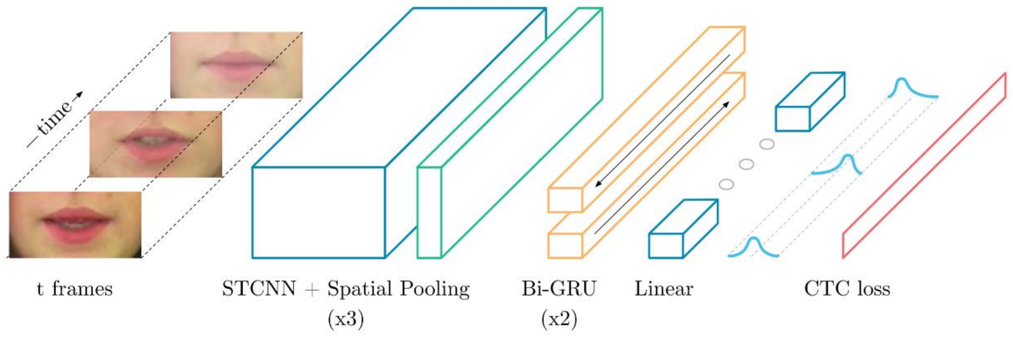


Figure 12: Model architecture

After preprocessing is done, the data can be used in the model. As can be seen in figure 12, the model consists of three layers of Spatiotemporal Convolutional neural networks (SCNN), followed by two bi-gated recurrent units, and the final layer is a connectionist temporal classification layer. The following pseudo-code shows the fitting, and usage of the model to predict the output.

```
fit_model(){
   input_layer = Input()
   scnn1 = Conv3D(input_layer);
   scnn2 = Conv3D(scnn1);
   scnn3 = Conv3D(scnn2);
   gru1 = Bidirectional(GRU(scnn3))
   gru2 = Bidirectional(GRU(gru1))
   ctc = CTC(gru2)
   model = Model(ctc)
   model.compile()

   train_gen = data_generator(train_path)
   validate_gen = data_generator(validate_path)
   model.fit(generator = gen, validation_data=validate_gen)
}

data_generator(path){
   foreach(video in load_folder(path + "\videos")
            & alignment in load_folder(path + "\alignment")){
```

```
        yield preprocess(video), alignment
    }
}
predict(video){
    input_layer = Input()
    scnn1 = Conv3D(input_layer);
    scnn2 = Conv3D(scnn1);
    scnn3 = Conv3D(scnn2);
    gru1 = Bidirectional(GRU(scnn3))
    gru2 = Bidirectional(GRU(gru1))
    ctc = CTC(gru2)
    model = Model(ctc)

    model.compile()
    model.load_weights()
    return model.predict(segement_video(video))
}
```

The raw output of the model has then to be decoded and approximated to the nearest number. This is done by utilizing fuzzy search logic.

```
closest = 0
ratio = 0
for i in 1 to 10{
    r = fuzzyRatio(i,output)
    if(r > ratio){
        closest = i
        ratio = r
    }
}
```

Chapter 6: System Implementation

6.1 Technologies used

Shefah is built using Python. It was chosen because of its simplicity. More importantly, it offers a huge collection of libraries that can be utilized for machine learning such as TensorFlow, desktop applications such as Tkinter, image processing such as Dlib, and many others. However, Python's major limiting factor is its speed as it is an interpreted language. Moreover, Python's libraries usually suffer from version conflicts and can cause some unexpected behavior.

Preprocessing the data was done in three different steps using mainly Dlib. First, faces were detected throughout the video using Dlib's facial detection model. Then, by using Dlib's 68 facial landmark predictor, the facial landmarks are extracted and used to calculate the region of interest. Finally, the video is cropped to show only the region of interest.

TensorFlow was chosen to build the machine learning model because of its ease of use, and its maturity. Moreover, TensorFlow specializes in deep learning and uses GPU hardware acceleration allowing intricate models to be built and tested easily and efficiently.

Tkinter was used to make Shefah's graphical user interface. This is due to its simplicity and robustness. Also, it offers a cross-platform solution that allows the program to run Windows, Linux, and macOS. Shefah can use the device's camera to record videos by using the OpenCV library. it will handle opening the camera, recording, saving, and displaying the video.

6.2 Dataset

Like any other machine learning project, a dataset is needed for training the model. Unfortunately, no dataset was found that meets Shefah's requirements. So, a dataset was created meeting the following requirements:

1. The dataset is in the Arabic language.
2. The dataset consists of videos of people speaking numbers from 0-9 to the camera.
3. Each speaker pronounces each number at least once, each pronunciation is stored in an independent video.

At first, the plan was to make each video contain a pair of numbers, so the scope can be expanded, and the model could predict composite numbers. However, due to the lack of data, the model underfitted. To achieve the targeted accuracy more data was needed, but due to the lockdowns caused by covid-19 creating that number of data was difficult.

The second approach was to create and collect videos of single numbers only. This approach allowed a higher accuracy to be achieved with the limited data available. Giving the circumstances, the videos were received through WhatsApp. The volunteers were given the instructions to record themselves while pronouncing numbers from 0 to 9 and then from 9 to 0. However, some only followed the first part which caused inconsistency in the data, Moreover, due to not being able to directly supervise the volunteers, the lightning and the angles of recording differed from one another.

A total of 77 volunteers participated in the creation of the dataset. However, due to technical difficulties, one speaker was dropped. The recorded videos were cut into intervals manually, each interval covers a single number. Resulting in a total of 939 usable videos.

6.2.1 Professional and Legal Responsibilities

All participants were recording themselves voluntarily, knowing their videos will only be used for this project. For privacy, ids were given to each participant. Also, the data was only accessible by the authors.

Chapter 7: System Testing

The system should be tested to check if it meets the mentioned requirements and meets the desired output.

7.1 Unit testing

The following methodology was followed to test the individual components manually.

1. File manager:

1	Test-case description	Ensure get_video() returns a compatible video file.
	Steps to implement test case	Open file dialog and choose a compatible video file.
	Expected result	Returns a video object containing the video file path.
	Actual result	As expected.
	Test result	Pass.

2	Test-case description	Ensure get_video() throws an exception when the video file is incompatible.
	Steps to implement test case	Open file dialog and choose an incompatible video file.
	Expected result	Throws an exception.
	Actual result	As expected.
	Test result	Pass.

3	Test-case description	Ensure get_video() throws an exception when the video file does not exist.
	Steps to implement test case	Open file dialog and choose a non-existing file.
	Expected result	Throws an exception.
	Actual result	As expected.
	Test result	Pass.

4	Test-case description	Ensure is_valid_video_file() returns true when a valid path pointing to a supported video.
	Steps to implement test case	Call the method with a valid path pointing to a supported video.
	Expected result	Return true.
	Actual result	As expected.
	Test result	Pass.

5	Test-case description	Ensure is_valid_video_file() returns false when a valid path pointing to a non-supported video.
	Steps to implement test case	Call the method with a non-existence path
	Expected result	Return false.
	Actual result	As expected.
	Test result	Pass.

6	Test-case description	Ensure is_valid_video_file() returns false with a non-existence path.
	Steps to implement test case	Call the method with a non-existence path
	Expected result	Return false.
	Actual result	As expected.
	Test result	Pass.

2. Video recorder:

1	Test-case description	Ensure open_camera() opens the default camera of the system. with the system having one only.
	Steps to implement test case	Call the method.
	Expected result	Camera opened.
	Actual result	As expected.
	Test result	Pass.

2	Test-case description	Ensure open_camera() throws an exception while the system does not have a camera.
	Steps to implement test case	Call the method.
	Expected result	Throws an exception.
	Actual result	As expected.
	Test result	Pass.

3	Test-case description	Ensure start_recording() starts recording from the opened camera. while the system is not recording.
	Steps to implement test case	Call the method.
	Expected result	Start recording
	Actual result	As expected.
	Test result	Pass.

4	Test-case description	Ensure start_recording() throws an exception when the system is recording.
	Steps to implement test case	Call the method.
	Expected result	Throws an exception.
	Actual result	As expected.
	Test result	Pass.

5	Test-case description	Ensure stop_recording() stops the recording of the running video.
	Steps to implement test case	Call the method with a video being recorded.
	Expected result	Stop recording.
	Actual result	As expected.
	Test result	Pass.

6	Test-case description	Ensure stop_recording() throws an exception when there is no video being recorded.
	Steps to implement test case	Call the method.
	Expected result	Throws an exception.
	Actual result	As expected.
	Test result	Pass.

3. Preprocessing:

1	Test-case description	Ensure preprocess_video() detects one face and crop the video to the region of interest.
	Steps to implement test case	Call the method and pass a video with one face in it.
	Expected result	A video cropped to the region of interest.
	Actual result	As expected.
	Test result	Pass.

2	Test-case description	Ensure preprocess_video() throws an exception when there is no face detected.
	Steps to implement test case	Call the method and pass a video with no face in it.
	Expected result	Throws an exception.
	Actual result	As expected.
	Test result	Pass.

3	Test-case description	Ensure preprocess_video() crops the region of interest of the largest face when more than one face is detected.
	Steps to implement test case	Call the method and pass a video with more than one face in it.
	Expected result	A video cropped to the region of interest of the largest face.
	Actual result	As expected.
	Test result	Pass.

4	Test-case description	Ensure detect_face() returns the face dimensions when one face is present.
	Steps to implement test case	Call the method and pass a video one face in it.
	Expected result	The dimensions of the face.
	Actual result	As expected.
	Test result	Pass.

5	Test-case description	Ensure detect_face() returns the dimensions of the largest face when more than one face is present.
	Steps to implement test case	Call the method and pass a video with more than one face in it.
	Expected result	The dimensions of the largest face.
	Actual result	As expected
	Test result	Pass.

6	Test-case description	Ensure detect_face() throws an exception when no faces are present
	Steps to implement test case	Call the method and pass a video with no faces in it.
	Expected result	Throws an exception.
	Actual result	As expected.
	Test result	Pass.

7	Test-case description	Ensure identify_facial_features() returns a 2D array with the facial features stored in it when the given video contains a face.
	Steps to implement test case	Call the method and pass a video with face in it.
	Expected result	2D array with the facial features stored in it.
	Actual result	As expected.
	Test result	Pass.

8	Test-case description	Ensure identify_facial_features() throws an exception when a video does not contain any face.
	Steps to implement test case	Call the method and pass a video with no face in it.
	Expected result	Throws an exception.
	Actual result	As expected.
	Test result	Pass.

9	Test-case description	Ensure crop_video() returns a cropped video with the given dimensions.
	Steps to implement test case	Call the method and pass a video and two-dimensional array of the area to crop.
	Expected result	Cropped video.
	Actual result	As expected.
	Test result	Pass.

4. Model:

1	Test-case description	Ensure predict()’s word error rate is less than 20% for unseen speakers.
	Steps to implement test case	Call the method multiple times on videos that were not used for fitting the model. Each video must be preprocessed before calling the predict method. For each compares the prediction with the expected output, and calculate the word error rate.
	Expected result	Word error rate less than 20% for unseen speakers.
	Actual result	Word error rate was 22% for unseen speakers.

7.2 Integration and regression testing

To test the integration of the system each relation between the component were tested as follows:

1	Test-case description	Ensure clicking on "choose file" in the interface, opens the file dialog from the file manager component.
	Steps to implement test case	1- Clicks on the "Select a File" button
	Expected result	a file dialog appears.
	Actual result	As expected.
	Test result	Pass.

2	Test-case description	Ensure choosing a supported file from the file dialog, the interface displays the chosen file.
	Steps to implement test case	1- Choosing a file from the file dialog.
	Expected result	The chosen file is played in the interface.
	Actual result	As expected.
	Test result	Pass.

3	Test-case description	Ensure choosing an unsupported file from the file dialog, the interface prompts a warning.
	Steps to implement test case	1- Choosing a file from the file dialog.
	Expected result	A warning prompt appears.
	Actual result	As expected.
	Test result	Pass.

4	Test-case description	Ensure clicking on the “record a video” button from the interface opens the default camera from the video recorder component.
	Steps to implement test case	1- Clicking on the “record a video” button.
	Expected result	A stream from the default camera is displayed in the interface.
	Actual result	As expected.
	Test result	Pass.

5	Test-case description	Ensure clicking on the “record a video” button from the interface while there is no camera from the video recorder component prompts a warning.
	Steps to implement test case	1- Clicking on the “record a video” button.
	Expected result	A warning prompt appears.
	Actual result	As expected.
	Test result	Pass.

6	Test-case description	Ensure the interface sends a valid input video to the preprocessor component.
	Steps to implement test case	1- Send an input file to the preprocessor component.
	Expected result	The input video is displayed cropped to the detected face and region of interest.
	Actual result	As expected.
	Test result	Pass.

7	Test-case description	Ensure that sending an invalid input video from the interface to the preprocessor component prompts a warning.
	Steps to implement test case	1- Send an input file to the preprocessor component.
	Expected result	A warning prompt appears.
	Actual result	As expected.
	Test result	Pass.

8	Test-case description	Ensure that the interface sends a preprocessed video to the lipreading model.
	Steps to implement test case	1- Sends a preprocessed video from the interface to the lipreading component.
	Expected result	Predicted word.
	Actual result	As expected.
	Test result	Pass.

7.3 Performance and stress testing

To test the performance of the system, it was tested on three different computers. The execution time will be split into three different steps and be tested independently: loading the model, video preprocessing, and prediction. To avoid outliers, an average of five runs was taken. The same video was used for all the runs with the resolution of 352x640 with a length of 37 frames.

	CPU: AMD 5600x GPU: RTX 3080	**CPU: i7-7500U GPU: NONE**	**CPU: i7-7700HQ GPU: NONE**
Video preprocessing	0.94 sec	2.26 sec	2.05 sec
Loading the model	3.19 sec	7.08 sec	3.81 sec
Prediction	2.54 sec	1.29 sec	1.92 sec

Table 1: Comparing running time with the same video.

To test the effect of different video resolutions on the performance of preprocessing, three different video resolutions were tested on three different systems. To avoid outliers, an average of three runs was taken.

	CPU: AMD 5600x GPU: RTX 3080	**CPU: i7-7500U GPU: NONE**	**CPU: i7-7700HQ GPU: NONE**
480x270	0.77 sec	1.47 sec	1.47 sec
720x406	1.44 sec	2.99 sec	2.83 sec
1080x608	2.98 sec	6.38 sec	6.23 sec
2160x1216	10.76 sec	24.4 sec	22.58 sec

Table 2: Comparing running time with different resolutions.

7.4 User acceptance testing

Shefah was tested on six users. The users were given test data to use in the testing. Each of them was asked to choose a file and record a video. All the testers managed to successfully satisfy the test condition for each use case.

Tester 1 stated that the “start recording” button and “stop recording” button should be the same to make it easier instead of moving the mouse from one button to another.

Tester 2 stated that the style is old and dull however, it does the job.

Tester 3 stated that when recording there is nothing that indicates he is recording for example blinking red icon or a timer.

Tester 4 stated that after using the program for the first run, there is no indication that she can use it for another file or recording.

Tester 5 stated that the total progress bar can felt out of place as the first one can be used with a ping pong effect to do the job.

Tester 6 stated that after done processing, hide the progress bar and show the result in the same place.

These comments were taken into consideration and the interface was redesigned accordingly.

7.5 Test cases

In this section, each test case will be shown with a description of it, the steps to implement it, and the expected result.

1	Test-case description	Ensure that the system shows an error message when selecting an incompatible file.
	Steps to implement test case	1- Clicks on the “Select a File” button. 2- Selects a random text file through the file dialogue.
	Expected result	A pop-up will be shown telling the user the input file is incompatible.
	Actual result	As expected.
	Test result	Pass.

2	Test-case description	Ensure that the system shows an error message when selecting a video with no faces shown in it.
	Steps to implement test case	1- Clicks on the “Select a File” button 2- Selects a compatible video file that contains no faces through the file dialogue.
	Expected result	A pop-up will be shown telling the user that no faces were detected.
	Actual result	As expected.
	Test result	Pass.

3	Test-case description	Ensure that the system processes and correctly displays a compatible selected file.
	Steps to implement test case	1- Clicks on the “Select a File” button. 2- Selects a compatible video file that contains one face pronouncing a number through the file dialogue.
	Expected result	1- The interface displays the input file. 2- The program starts preprocessing the file and progress bars are shown and are updated through the preprocessing steps. 3- Preprocessing steps are displayed to the interface. 4- The prediction is displayed on the interface.
	Actual result	As expected.
	Test result	Pass.

4	Test-case description	Ensure that the system shows an error message when trying to record with no connected cameras.
	Steps to implement test case	The user clicks on “Record a Video”.
	Expected result	A pop-up appears. “No camera was found. Please select a file”.
	Actual result	As expected.
	Test result	Pass.

5	Test-case description	Ensure that the system shows an error message after recording a video with no faces shown in it.
	Steps to implement test case	1- The user clicks on “Record a Video”. 2- The user clicks on “Start Recording”. 3- Make sure no faces are shown to the camera. 4- The user clicks on “Stop Recording”.
	Expected result	A pop-up will be shown telling the user that no faces were detected.
	Actual result	As expected.
	Result	Pass.

6	Test-case description	Ensure that the system processes and correctly displays a compatible recorded video.
	Steps to implement test case	1- The user clicks on “Record a Video”. 2- The user clicks on “Start Recording”. 3- Record a speaker pronouncing an Arabic number. 4- The user clicks on “Stop Recording”.
	Expected result	1- The interface displays the input file. 2- The program starts preprocessing the file and progress bars are shown and are updated through the preprocessing steps. 3- Preprocessing steps are displayed to the interface. 4- The prediction is displayed on the interface.
	Actual result	As expected.
	Test Result	Pass.

Chapter 8: Experimentation and Findings

This chapter discusses the results and findings of the experiments done on Shefah's model and the dataset.

8.1 A pair of numbers or just one

The first data collected was videos containing a pair of numbers for each video. After training and testing the model, the accuracy was not promising at all. So, the data has been changed to videos containing one number instead of a pair. Training and testing the model gave great accuracy compared to using a pair.

8.2 Mirrored images

One of the ways to avoid underfitting is to increase the size of the data by mirroring the images. This approach was used in the preprocessing step. The resulting accuracy was higher than not mirroring the images.

8.3 Transfer Training

Seeing that Shefah and LipNet share the same model design, implementing transfer training might help Shefah overcome its lack of training data. This is achieved by loading the weights of LipNet and locking all of the layers before the classification layers and then training the classification layers on the training data.

Theoretically, the model should have higher accuracy compared to training from scratch. However, as seen in table-3 the results were the opposite. This is probably caused by the bias in Shefah's dataset, as it has no female participants unlike Lipnet's.

Shefah without transfer training	**78%**
LipNet overlapped data weights	95%
LipNet unseen data weights	89%
Shefah transferred from overlapped LipNet	67%
Shefah transferred from unseen LipNet	66%

Table 3: Transfer training accuracy.

8.4 Random First Frame Duplication

To simulate the random delay that comes with recording videos, some of the training data was forward-shifted by a random amount and the first frame was duplicated to simulate the delay. The results of this experiment were as shown in table 4.

Trained	Tested	Accuracy
Without duplication	Without duplication	**78%**
	With duplication	55%
With duplication	without duplication	68%
	With duplication	66%

Table 4: Random first frame duplication results.

As can be seen, the model trained without duplication had higher accuracy on similar videos, but the accuracy decreased by 23% when it was exposed to videos with random duplication. On the other hand, when the model was trained with data that had its first frame randomly duplicated, it had an accuracy of 68% but it was more resilient to change and only decreased by 2% when tested on the videos without duplication.

8.5 Preprocessing dots

The proposed idea was to add dots and lines to the lip from the 68 facial landmark predictor. However, the idea was scrapped due to some problems detected in the predictor.

8.6 Regions of Interest

As for regions of interest, two different regions were tested as in Figure 13. The left frame takes more surrounding area around the lip than the right frame. The resulting accuracy was the increased region achieved higher accuracy than the regular one.

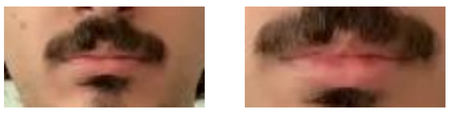


Figure 13: Region of interest

What can be inferred is the area around the lip can be utilized to achieve higher accuracies.

8.7 Cross-validation

To get an average accuracy on how the model reacts to each subset of the data cross-validation was used with 5 folds. The data was split into 5 subsets. The first subset was used for testing and the other 4 used for training and so on. The average accuracy over all the folds was 69.63%.

Chapter 9: Conclusion

The objective of this project was to create a visual speech recognition system that utilizes deep learning. So, Shefah was designed, implemented, and trained to be able to recognize Arabic numbers from a given video in a user-friendly and easy-to-use graphical user interface. To train the model, a dataset was created consisting of 939 videos with the help of 77 voluntaries. Even though the data was very limited, the model achieved an accuracy of 78% on unseen speakers. This was achieved through mirroring the data, and careful selection of the region of interest.

Shefah can play a major role in increasing the accuracy of speech recognition systems when integrated with voice-based speech recognition. Moreover, the scope of Shefah can be easily expanded to cover the rest of the Arabic language if enough data can be provided.

Finally, we hope that Shefah serves as a building block towards the digitalization of the Arabic language.

References

[1] L. Woodhouse, L. Hickson and B. Dodd, "Review of visual speech perception by hearing and hearing-impaired people: Clinical implications," *International journal of language & communication disorders / Royal College of Speech & Language Therapists,* vol. 44, pp. 253-70, 2008.

[2] G. G. Chowdhury, "Natural language processing," *Annual Review of Information Science and Technology,* vol. 37, pp. 51-89, 2003.

[3] R. Szeliski, Computer Vision: Algorithms and Applications, Springer, 2010.

[4] A. Burkov, The Hundred-Page Machine Learning Book, 2019.

[5] T. DeVries and G. W. Taylor, "Improved Regularization of Convolutional Neural Networks with Cutout," 2017.

[6] A. Fernandez-Lopez and F. M. Sukno, "Lip-Reading with Limited-Data Network," in *27th European Signal Processing Conference (EUSIPCO)*, A Coruna, Spain, Spain, 2019.

[7] K. Noda, Y. Yamaguchi , K. Nakadai, H. G. Okuno and T. Ogata, "Lipreading using Convolutional Neural Network," in *15th Annual Conference of the International Speech Communication Association*, Singapore, 2014.

[8] A. Garg, J. Noyola and S. Bagadia, "Lip reading using CNN and LSTM," *Technical report, Stanford University, CS231 n project report,* 2016.

[9] J. S. Chung and A. Zisserman, "Lip Reading in the Wild," in *Computer Vision -- ACCV 2016*, Springer International Publishing, 2017, pp. 87-103.

[10] M. Wand, J. Koutník and J. Schmidhuber, "Lipreading with Long Short-Term Memory," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2016, pp. 6115-6119.

[11] T. Afouras, J. S. Chung, A. Senior, O. Vinyals and A. Zisserman, "Deep Audio-visual Speech Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 2018.

[12] T. Afouras, J. S. Chung and A. Zisserman, "arXiv.org e-Print archive," 15 June 2018. [Online]. Available: https://arxiv.org/abs/1806.06053. [Accessed 17 October 2020].

[13] J. Shin, J. Lee and D. Kim, "Real-time lip reading system for isolated Korean word recognition," *Pattern Recognition,* vol. 44, no. 3, pp. 559-571, 2011.

[14] Y. Assael, B. Shillingford, S. Whiteson and N. Freitas, "LipNet: Sentence-level Lipreading," arxiv, 2016.

[15] M. Faisal and S. Manzoor, "Deep Learning for Lip Reading using Audio-Visual Information for Urdu Language," arxiv, 2018.

[16] Y. Zhang, S. Yang, J. Xiao, S. Shan and X. Chen, "Can We Read Speech Beyond the Lips? Rethinking RoI Selection for Deep Visual Speech Recognition," in *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020) (FG)*, 2020.

[17] Y. Zhang, S. Yang, J. Xiao, S. Shan and X. Chen, "Hearing Lips: Improving Lip Reading by Distilling Speech Recognizers," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[18] carykh, "Github," 23 March 2018. [Online]. Available: https://github.com/carykh/videoToVoice. [Accessed 7 November 2020].