



UNIVERSITY OF  
SURREY

# AUDIO DRIVEN FACE ANIMATION

## Mid Term Report

For final Year Project

Due 4<sup>th</sup> January 2022

Konstantinos Koutelidis

Department of Electronic Engineering

Faculty of Engineering and Physical Sciences

University of Surrey

## Abstract

---

The Audio Driven Face Animation project intends to create a software system that uses machine learning to create a talking character that infers its face motions from speech. The dataset collecting and pre-processing stages attempt to prepare the data for the training phase, which follows. The face prototype will depict the recorded face, and the necessary steps for the face movements will be implemented with an audio to image prediction network to first load the pairs of images and audio features that are the pre-processed data aforementioned, and then design the model that maps the useful frequencies from the audio files with the pairs of images-frames extracted from the video files. An outlook is given in this project regarding the structure of the final report.

## Table of Contents

---

Abstract.....	2
Table of Contents.....	3
1. Introduction.....	5
2. Literature Review.....	7
2.1. Facial Animation–Dataset Capturing Strategies for Diverse Applications.....	7
2.2. Linguistic Based Models.....	8
2.2.1. Disadvantage of the method.....	9
2.3. Deep Fakes.....	9
2.3.1. Understanding Deep Fakes .....	9
2.4. Machine Learning Introduction .....	10
2.4.1. Applications of Machine Learning.....	11
2.4.2. Dataset Partitions .....	12
2.5. Few Shot Adversarial Learning .....	12
2.6. Audio-Visual Animation.....	13
2.7. Summary .....	15
3. Dataset Design .....	16
3.1. Harvard Sentences .....	16
3.2. Dataset Recording.....	17
4. Visual Data Pre-Processing.....	18
4.1. Data Augmentation .....	18
4.2. Face Alignment.....	19
4.2.1. FAN 2D Framework .....	19
5. Audio Pre-Processing Techniques .....	20
5.1.1. Mel-Frequency .....	20
6. Future Work.....	21
6.1. Planned Deliverables .....	21
References.....	24

Appendix A.....	25
Appendix B .....	33

## 1. Introduction

---

The aim of this project is to develop a software system capable of producing facial movements of a recorded character by using audio. The primary idea behind this research is that by providing enough information about speech and face movement to the dataset, audio-driven face animation can be created, without the added complexity of emotion. In the scope of an undergraduate Bachelor of Engineering project, this paper documents the current condition of the Audio Driven Face Animation. The impetus for this effort came from recent advances, particularly in the fields of computer vision. While computer animation research dates back to the 1940s and 1950s, it is only in the last decade that face, and character animation has progressed to the point where it provides a realistic result. As this technology is evolving, it has primarily been used in entertainment applications such as the gaming industry, but the need for remote working due to the covid outbreak has encouraged many businesses to explore augmented reality as a viable option. Any sophisticated solution, on the other hand, comes with a significant development cost, which might be seen as a barrier to growth in the field of computer vision. To make facial animation more efficient and realistic, a rising number of individuals and university researchers have made it their mission to collect data from industry and use it to improve algorithms and training systems. The emergence of augmented reality and the rise of metaverses in the previous two years, in particular, will drive demand for audio-driven facial animation to new heights. Almost all firms will work in the "digital world" in the near future, and employees will use lifelike avatars of themselves during meetings and daily tasks. The goal of this project is to create a lifelike character that can display sophisticated face motion in response to auditory input. A number of goals must be met in order to achieve this:

1. A dataset prototype has to be collected, consisting of 20 videos that utilize Harvard Sentences which were designed to evaluate audio speech intelligibility in a range of communication scenarios.
2. The dataset has to be separated into different file formats for video and audio (mp4, wav), using the ffmpeg library.
3. Pre-processing of the video dataset is one of the most important tasks that prepare the videos files for the training stage. The facial landmarks have to be detected during that stage.
4. Pre-processing of the audio dataset is also important since the training requires Mel-Frequencies that are extracted from the audio during that stage.
5. An audio to image prediction network must be created to map the Mel-Frequency features to the frame images that were extracted during step 3.

There is one major stumbling block in achieving such objectives. This project is only for one academic year, and it will take roughly ten hours each week to complete. To ensure that the most work can be done in this time limit, an effective work plan must be in place. To be on time, all stages must be completed before the first academic break, with the exception of the final one. Because the project is built using free and open-source software, there are no budget limits that could stifle development.

The following is a breakdown of the report's structure: Chapter 2 provides an overview of recent breakthroughs in the field of audio-driven face animation. The dataset design is presented in Chapter 3, while the pre-processing of the dataset and the methodologies utilised are presented in Chapter 4. Chapter 5 is a concise review of all findings, followed by a conclusion and a list of references in Chapter 6. The audio-driven face animation frames and face-alignment procedures are shown in Appendix A. Gantt Appendix B contains charts for semesters one and two.

## 2. Literature Review

Audio driven face animation plays an important role in engineering and computer systems with a vast variety of uses in the entertainment and gaming industry. Eadweard Muybridge captured the first ever motion sequence on June 15, 1898. The experiment's main purpose was to see if a sprinting horse could lift all four feet off the ground. Using analogue cameras with fast shutter speeds, Eadweard was able to produce a video that comprises of quick frames and visualizes the horse's action. Engineers and programmers were able to evaluate more complicated and detailed activities such as facial animation and pattern recognition thanks to advanced video cameras and a significant advancement in computer vision. Such analytics provide a unique perspective on day-to-day activities. By integrating the physical and digital worlds, creating a digital person, or in this case a digital face, can change the world as we know it. The sections 2.1 and 2.2 will go through how deep neural networks are utilized to generate various analytics from motion sequences in detail. Different strategies for capturing and modifying datasets, as well as developing neural networks, will also be discussed. [1]

### 2.1. Facial Animation–Dataset Capturing Strategies for Diverse Applications

Most of the time, a deep neural network is utilized to infer face movements from speech. The network makes extensive use of audio samples and waveforms. In some circumstances, the dataset contains emotional states that can be used by the neural network to improve the accuracy and detail of the face animation. Everything is happening in real time with low latency, and the network's mapping of all the different waveforms to the 3D vertex coordinates of the face aids in the creation of a better animation that could not be created only through the use of audio. [2]

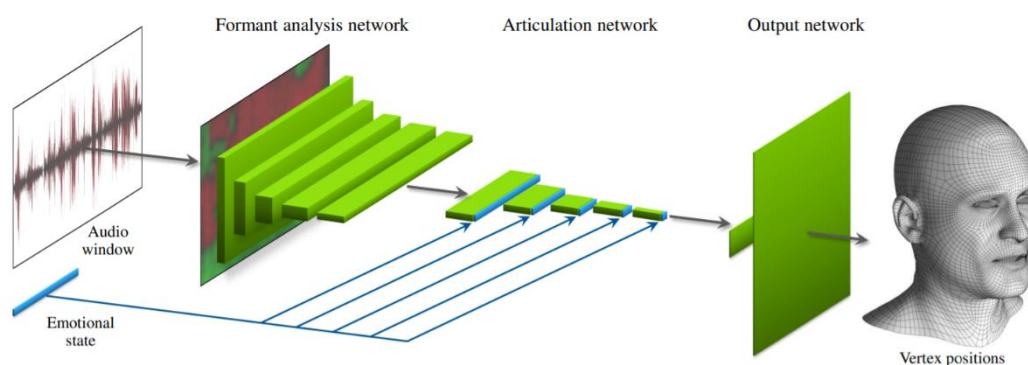


Figure 2.1.1.: Deep neural network with audio and emotional input

[https://research.nvidia.com/sites/default/files/publications/karras2017siggraph-paper\\_0.pdf](https://research.nvidia.com/sites/default/files/publications/karras2017siggraph-paper_0.pdf)

While audio-based performance capture algorithms will never be on par with vision systems in terms of fidelity, they do provide additional advantages. Most crucially, using vision-based algorithms, it is extremely expensive to produce tens of hours of speech uttered by in-game characters in many recent games. As a result, vision systems are often used to create only crucial animations, such as cinematics,

and audio and transcript systems are used to deliver the majority of in-game information. Unfortunately, that practice is far from ideal. Telepresence is another type of audio facial animation that demands real-time processing, which adds to the challenges.

In order for the output to appear realistic, the animation must account for complex and interdependent phenomena such as phoneme coarticulation, lexical stress, and the interaction between facial muscles and skin tissue. Pay attention to the entire face, not just the lips and mouth. A content-driven approach is used, training a deep neural network from start to finish to replicate the key effects observed in the training data. The difficulty may look unsolvable at first because to the inherent ambiguity of the problem—the identical sounds can be pronounced with a wide range of facial expressions, and the audio track simply does not provide enough information to distinguish between the various versions [Petrushin 1998].

While contemporary convolutional neural networks have shown to be incredibly efficient in a variety of inference and categorization tasks, they have a tendency to regress to the mean when the training data is ambiguous. We propose three important contributions to address these issues:

- A convolutional network architecture that is specifically designed to interpret human voice and generalize across multiple speakers.
- A novel method for allowing the network to detect fluctuations in the training data that cannot be explained only by the audio, such as apparent emotional state.
- Even with very confusing training data, a three-way loss function means that the network stays temporally robust and fast during animation.[2]

## 2.2. Linguistic Based Models

A transcript is frequently included with an audio file to aid in the transmission of explicit knowledge about the phoneme content. The animation is then created using complex coarticulation techniques and is based on visemes, which are the visual equivalents of phonemes. The dominance model refers to systems like this. Based on psycholinguistic considerations, JALI factors in facial animation, lip and jaw movements, and is capable of successfully reproducing a wide range of speaking styles regardless of the actual speech content. A transcript is frequently included with an audio file to aid in the transmission of explicit knowledge about the phoneme content. The animation is then created using complex coarticulation techniques and is based on visemes, which are the visual equivalents of phonemes. The dominance model refers to systems like this. Based on psycholinguistic considerations, JALI factors in facial animation, lip and jaw movements, and is capable of successfully reproducing a wide range of speaking styles regardless of the actual speech content. The key benefit of utilising this method is the explicit control over the entire process, which allows the face to explicitly ensure that the mouth closes properly when spelling out a bilabial "m", "p", "b" or that the bottom lip hits the upper teeth



while saying labiodentals "f", "v" etc. Even with vision-based capturing, both of these scenarios are challenging.

### 2.2.1. Disadvantage of the method

- Complexity
- There are language-rules that have to be followed
- Quality of datasets is not always perfect
- Non-phoneme sounds make it hard for facial reactions to be realistic
- Many parts of the face will not be animated to avoid complex computations [2]

## 2.3. Deep Fakes

Deep fakes were created as a result of recent advances in artificial intelligence and are now more commonly connected with "fake news" or "false information." Deep fakes are created by modifying existing data to create misleading data in the form of audios, videos, or images. Techniques can be used maliciously to harm one's image or create supremacy in global or local politics. There is a huge gap in the quantity of skill available for both creating and identifying deep fakes. More work is being done to invent new algorithms and strategies that will enable for deep fakes to be indistinguishable from genuine data. An example of Deep Fakes implementation would be "wombo.ai" which although is used as a social media application it has the capabilities of producing corrupted information regarding public figures. [4]

### 2.3.1. Understanding Deep Fakes

The problem posed by Deep Fakes can be solved from the ground up via algorithmic analysis. Face Swapping is one of the main techniques used for creating Deep Fakes. Deep neural networks are mainly used to create deep fakes. Face swap is one of the various strategies for creating deep fake images. Face swap can be done in a variety of ways, the most noteworthy of which are the following. Deep neural networks are primarily employed in the creation of deep fakes. Face swap is one of the many techniques used to create deep fake images. Face swapping can be accomplished in a variety of ways, the most notable of which are listed below. The first step is to use Feature Selection Networks, followed by Open-Set Identity Preserving Face Synthesis.

In Feature Selection Network face swaps, deep neural networks are used. They're based on 3D morphable models that split a face into two parts. A mesh consisting of a mean face and two matrices, one for form and the other for face texturing, was created at first. The two specified matrices describe various forms of variation of the face component from the mean. [4]

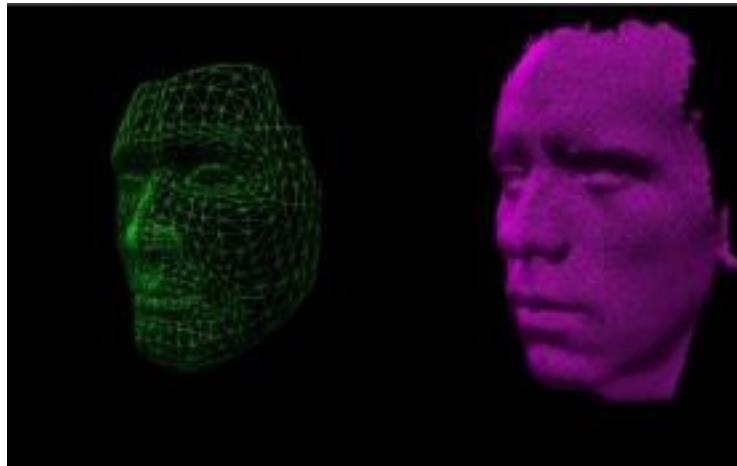


Figure 2.3.1.1: Dimensional Morphable Model

[https://www.easychair.org/publications/preprint\\_download/RK3j](https://www.easychair.org/publications/preprint/download/RK3j)

## 2.4. Machine Learning Introduction

The art of designing algorithms without explicitly developing an algorithm by programming is known as machine learning (ML). A tremendous amount of data has been accumulated in the last twenty years, and so most industries have been fully digitised. Machine learning (ML) systems use this data to construct predictive models and handle various time-consuming operations.

Flow is fully defined and understood in advance for logic-based algorithms, but there are some real-life circumstances like image classification where logic cannot be described. Machine learning has shown to be quite effective in these situations. Machine learning approaches produce reasoning associated with the input parameters and expected reference data output.

**Machine Learning is divided into two different categories.**

- **Supervised Machine Learning:** At the input, the supervised Machine Learning algorithm receives input data which most of the time is called "features" and output labelled data. They're most commonly utilised for classification and regression problems.
- **Unsupervised Machine Learning:** Unsupervised machine learning algorithms are used to cluster data into various segments associated with the input features and do not require any labelled data.

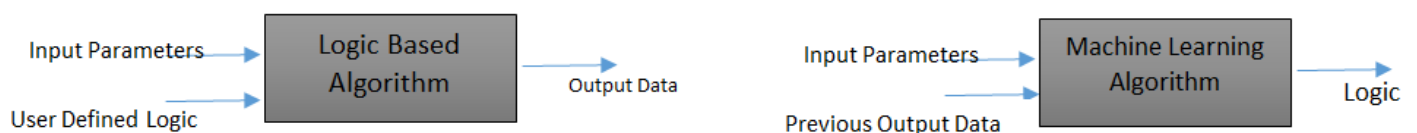


Figure 2.4.1.: Difference of Machine Learning with Explicitly Programmed Algorithm

[Introduction To Machine Learning | Application of Machine Learning \(educba.com\)](#)

### 2.4.1. Applications of Machine Learning

Healthcare, social media, digital marketing, real estate, logistics, supply chain, and manufacturing have all been revolutionized by machine learning during the last decade. Early adopters in these fields have already profited. A trained workforce with machine learning is becoming increasingly in demand to make the implementation of machine learning a global phenomenon utilized by all organizations.

#### Examples of Machine Learning Application.

- Spam Messages/Email classification: To categorise email as spam or not spam with labelled answers based on data like message content, advertising terminology, sender email account, sender IP, links, numeric formatting, and other factors.

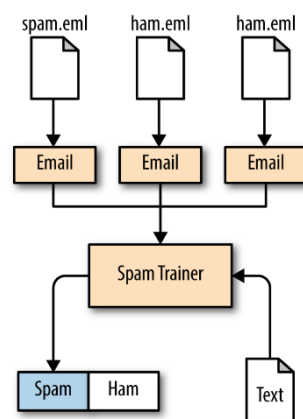


Figure 2.4.1.1: Naive Bayesian Classification

#### [4. Naive Bayesian Classification - Thoughtful Machine Learning \[Book\] \(oreilly.com\)](#)

- Cancer Detection: Medical data from prior patients is rapidly being used in machine learning for diagnosing or even cancer detection in healthcare. The training method uses inputs including tumour size, radius, curvature, and perimeter to detect breast cancer. We receive the probability of the tumour becoming cancerous or not as a result of the output.

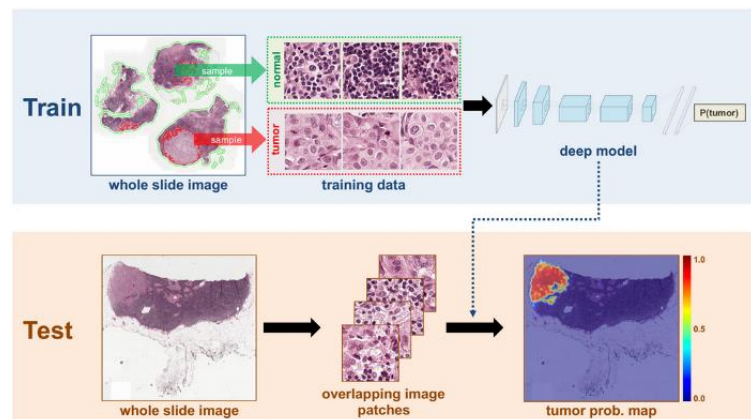


Figure 2.4.1.2: Cancer Detection Using Machine Learning.

#### [Understanding Cancer using Machine Learning - KDnuggets](#)

- Video/Audio Interpretation: Digital Assistants like Alexa, Siri, and Google are becoming increasingly clever at processing audio data in a variety of languages and accents. In these cases, a large volume of data is taught to introduce machine learning techniques. [3]

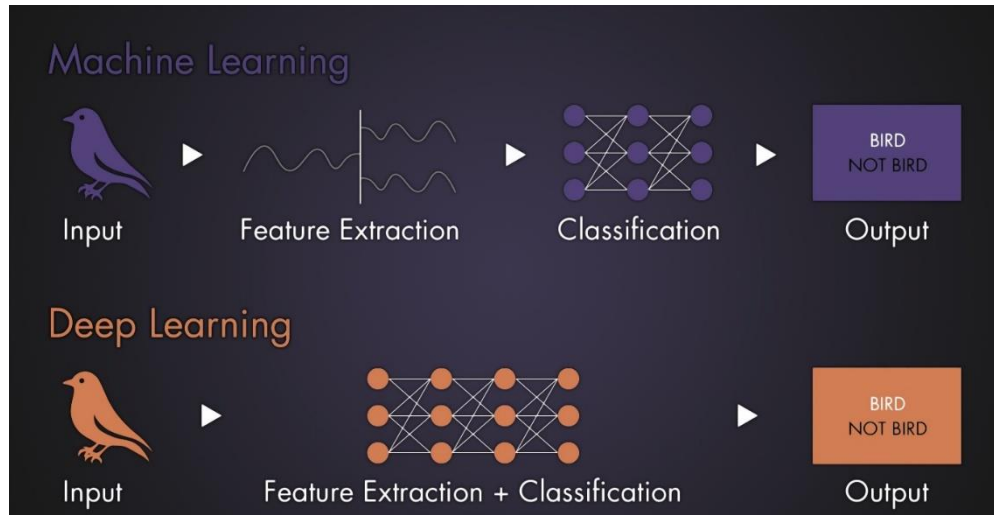


Figure 2.4.1.3: Machine learning technique for understanding a bird's sound

[Machine Learning and Deep Learning for Audio | BOOM Library](#)

## 2.4.2. Dataset Partitions

The acquisition of data in machine learning must be separated in a precise way. A total of 80% of the data is utilised to train the algorithm, 10% is used for validation, and 10% is used for testing. The test data and the validation data must be distinct. Using the same data for validation and testing is considered a bad practise since the algorithm may end training early if it performs properly with the validation data. As a result, it's critical to keep some unseen test data on hand to evaluate the trained model.

## 2.5. Few Shot Adversarial Learning

Various studies have demonstrated that convolutional neural networks can be trained to create remarkably realistic human face images. These efforts involve training on a big collection of photos of a single individual in order to construct a personalised talking face model. In practice, however, such customised talking head models must be trained from a few image scans of a person, if not just one.

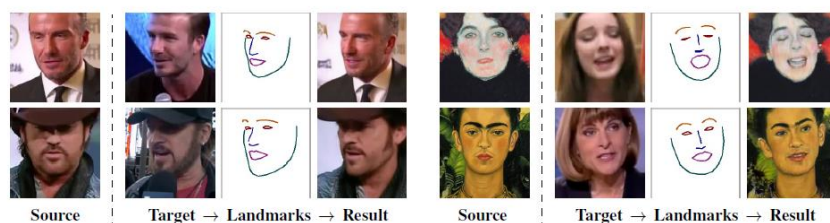


Figure 2.5: Illustration of Few Shot Adversarial Learning

<https://arxiv.org/pdf/1905.08233v2.pdf>

There are systems with such a few-shot capability, however on this project the aim is to produce one face which will be trained in detail with frames captured from the 20 videos taken during the dataset creation. The statistical modelling of the appearance of human faces has a great deal of evidence, with exceptionally good achievements obtained both with traditional techniques and, more recently, using deep learning. While modelling faces and talking heads are closely related, the latter also involves a more detailed modelling approach and can depict more characteristics such as hair, mouth cavity, and, in some cases, shoulders and upper clothing. These non-facial parts are far less accessible to registration and frequently have higher volatility and diversity than the face part, therefore they can't be handled by a simple extension of face modelling approaches. Face modelling and lips modelling findings can theoretically be merged into an existing head footage. However, because such a design does not offer complete control over the head rotation in the produced video, it does not qualify as a full-fledged talking head system. The few-shot training architecture is heavily influenced by current advances in picture generative modelling. As a result, the system incorporates adversarial training and, more precisely, the concepts behind conditional discriminators, such as projection discriminators. The adaptive instance normalisation mechanism, which has been demonstrated to be useful in large-scale conditional generation problems, is used in the meta-learning stage. We also find that using the concept of content style decomposition to separate the texture from the body pose is quite useful. The model-agnostic meta-learner (MAML) employs meta-learning to acquire an image classifier's initial state, from which it can swiftly converge to image classifiers of unknown classes with a small number of training data. Several studies have also suggested combining adversarial training and meta-learning. As a result, data-augmentation GANs, MetaGANs, and adversarial meta-learning use adversary trained systems to create extra instances for classes that were not visible during the meta-learning phase. While these techniques are focused on improving few-shot performance of the classifier, the initial method described is concerned with the training of image generation models employing adversarial objectives that are comparable to those employed in these techniques. To summarise, adversarial fine-tuning is introduced into the meta learning framework. The former is used once the meta-learning stage has yielded the initial state of the generator and discriminator networks. Finally, two recent works on text-to-speech creation are extremely similar to ours.

## 2.6. Audio-Visual Animation

In this section the normalisation of image and lip synchronisation with audio will be described for the creation of realistic interaction using faces and voices. The algorithm searches the audio & video recordings for the initial encounter of a viseme expression pair and retrieves all possible combinations. It's possible that the photos generated this way aren't aligned. If these shots are utilized to create an animation, the final sequence will contain strange and unintentional head movements. As a result, the shots must be aligned. Between photos, there are two types of movement: 3-D rigid body movement and non-rigid movement. The stiff component is caused by head rotation, translation, and other such

factors, whereas the non-rigid part is caused by changes in expressions and lip contour. Under such a viewpoint projection, the face could be represented as a single plane. The optical flows can be described using the eight-parameter framework below.

$$\begin{aligned} u(x, y) &= a_0 + a_1x + a_2y + p_0x^2 + p_1xy \\ v(x, y) &= a_3 + a_4x + a_5y + p_0xy + p_1y^2 \end{aligned}$$

Figure 2.6: Illustration of the eight-parameter framework by using to functions  $u(x,y)$  &  $v(x,y)$  [9].

<https://cpb-us-w2.wpmucdn.com/sites.uwm.edu/dist/a/171/files/2016/11/icme01-t32x3j.pdf>

Because this framework does not capture non-rigid movement of facial characteristics well, it is used to retrieve the 3D rigid body aspect of movement and align the shots. We apply a modified version of Tsai and Huang's technique to evaluate the coefficients. This method is a least square match across the shot gradients. We initially compute the 3D rigid body movement factor from I2 to I1 using face shots I1 and I2. Next, this framework is used to distort picture I2 so that it is aligned with I1 and has the same viseme contour as I1. Because of the expected planar model for the face in perspective projection, some photos may have mild face distortion. A series of photos can be aligned with regard to a single shot and then repeat the operation iteratively. [9]

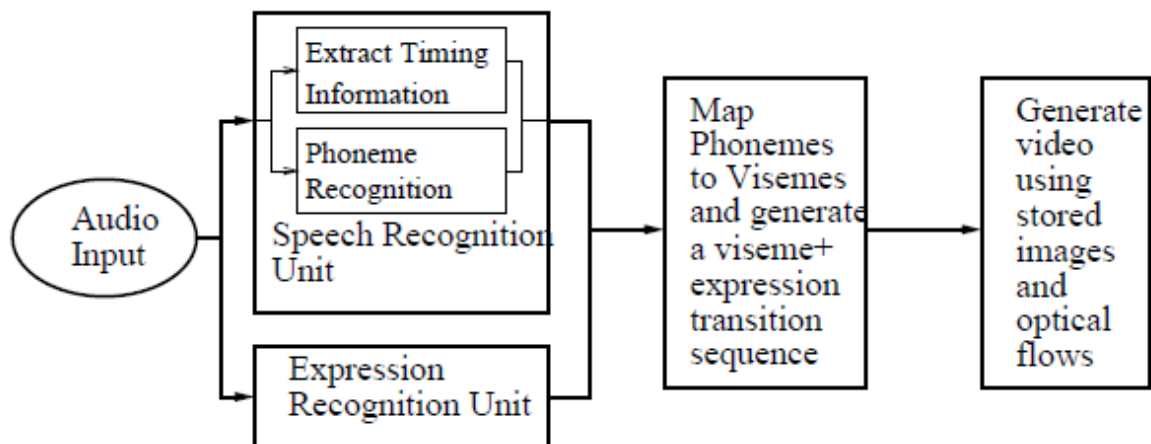


Figure 2.6.1: Unit for Synthesis [9]

The voice recognition unit extracts the time data from the entering audio stream. This temporal data controls the timing of lip movements and the amount of morph. Using visual flow driven morphing systems, transitory frames are generated from two normalized viseme pictures. Assume that the viseme change among  $u_1$  and  $u_2$  takes place at  $T$ . Image warping is utilized with visual flows to construct a frame at time  $0 < t < T$ . The visual fluxes are calculated from  $u_1$  to  $u_2$  (OF1) and from  $u_2$  to  $u_1$  (OF2). To construct the produced frame, the two acquired frames are cross decomposed in a balanced fashion.

Phoneme	Viseme No	Phoneme	Viseme No
<i>a, h</i>	<i>Viseme1</i>	<i>g, k, d, n, t, y</i>	<i>Viseme7</i>
<i>e, i</i>	<i>Viseme2</i>	<i>f, v, w</i>	<i>Viseme8</i>
<i>l</i>	<i>Viseme3</i>	<i>h, j, s, z</i>	<i>Viseme9</i>
<i>r</i>	<i>Viseme4</i>	<i>sh, ch</i>	<i>Viseme10</i>
<i>o, u</i>	<i>Viseme5</i>	<i>th</i>	<i>Viseme11</i>
<i>p, b, m</i>	<i>Viseme6</i>	<i>silence</i>	<i>Viseme12</i>

Figure 2.6.2: Table showing the mapping from Phoneme to Viseme. [9]

## 2.7. Summary

The approaches of facial animation and few-shot adversarial learning were discussed. For this project, a few-shot technique was chosen because training frames from 20 videos takes less time and requires a smaller dataset. A variety of open source and hobbyist examples of few-shot learning and audio-visual animation are also available online.

Then there was a look into Deep Fakes, which looked at how artificial intelligence might harm the world by giving false information in the form of videos, audio, and images.

To provide a wider range of implementations in real-world circumstances, a Machine Learning introduction as well as applications were provided.

Finally, a quick overview of dataset partitions was given in order to teach that data in machine learning must be separated in a particular manner.

### 3. Dataset Design

---

The dataset collection and design are the most important factors when it comes to the training of a deep neural network. In this part, a dataset of twenty videos capturing a face speaking was recorded. The sentences used for the face speaking were chosen on purpose from the “Harvard Sentences Dataset” which are phonetically balanced and therefore ensured diverse sounds in the dataset. Since the project is done for academic purposes, the equipment used was not top-notch although this did not have any effect on the quality of the outcome.

#### 3.1. Harvard Sentences

Harvard sentences, also known as Harvard lines, were designed to evaluate audio speech intelligibility in a range of communication scenarios while offering minimal semantic information. They were employed in the recorded videos because they provide standardised and repeating voice sequences that make training deep neural networks with face alignment and animation simple.

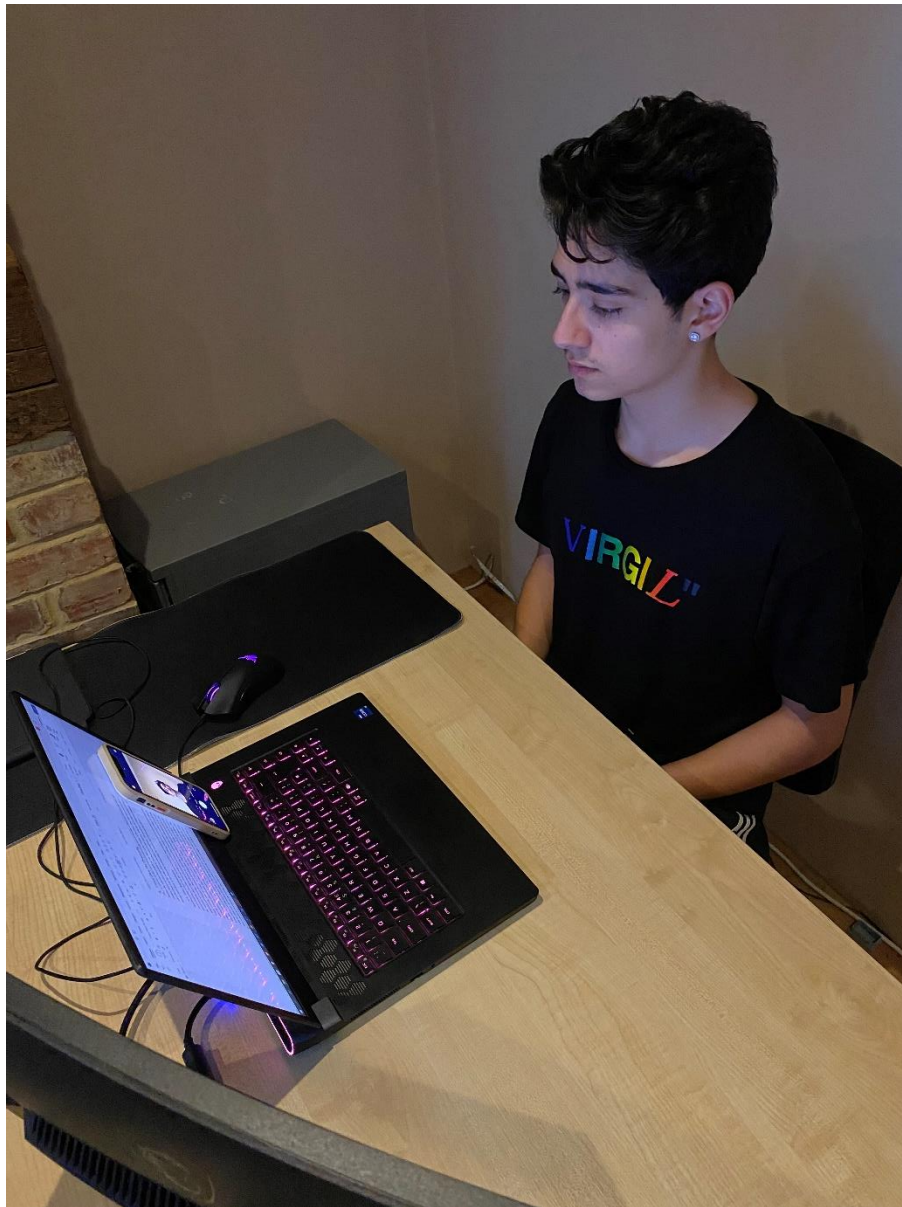
Sentences Used:

- The boy was there when the sun rose.
- A rod is used to catch pink salmon.
- The source of the huge river is the clear spring.
- Kick the ball straight and follow through.
- Help the woman get back to her feet.
- A pot of tea helps to pass the evening.
- Smoky fires lack flame and heat.
- The soft cushion broke the man's fall.
- The salt breeze came across from the sea.
- The girl at the booth sold fifty bonds.
- A king ruled the state in the early days.
- The ship was torn apart on the sharp reef.
- Sickness kept him home the third week.
- The wide road shimmered in the hot sun.
- The lazy cow lay in the cool grass.
- Lift the square stone over the fence.
- The rope will bind the seven books at once.
- Hop over the fence and plunge in.
- The friendly gang left the drug store.
- Mesh wire keeps chicks inside.



### 3.2. Dataset Recording

An iPhone was used to record a collection of videos for the dataset. Because the training portion does not necessitate high-resolution recordings, a smartphone proved sufficient. The films' background is an important component that can help with training, which is why a plain white background was chosen. The camera had to be stable while recording the twenty Harvard Sentences in order to reduce the amount of work that the face alignment algorithm had to undertake.



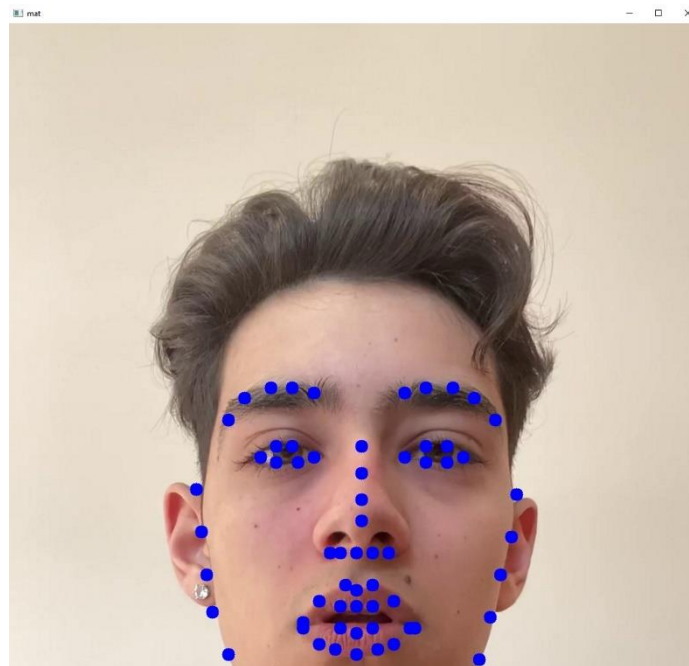
*Figure 3.2.1.: Dataset Recording Process*

## 4. Visual Data Pre-Processing

This section covers the visual data pre-processing used to develop a robust face tracking algorithm that locates the relevant key-points on the face (eyes, centre of the head, lips, and ears) and produces a contour based on those key-points. Furthermore in this section, with the aid of the produced contour, all of the frames of the video data are recorded and analysed. The procedure of aligning, cropping, and storing each frame is facilitated by constructing the contour and capturing the frames.

### 4.1. Data Augmentation

Pre-Processing of data can also be referred as Data Augmentation which is something important when we are training a classifier since we want to avoid Over-fitting which is a general problem occurring in neural networks due to high count of parameters to learn. Data Augmentation is a method of amplifying the volume of training data obtained by applying minor adjustments that distinguish it in small quantities and increase the amount of training data. Data augmentation that was performed to train the image network, take the image, and just crop a small sub-region of that image to push through that network. Picture classification algorithms such as AlexNet use 244x244 pixels in an image, which is then scaled to 256x256 pixels and cropped at a random location window to prepare it for training. Tiny affine transformations, such as rotation, scaling, and warping, are simply extremely small changes that simulate the type of typical variation that is likely to be encountered in the actual world. It's worth noting that preserving each frame aids in the development of a significant amount of data. If we train with too little data for machine learning, the system will not be diverse enough. Over-fitting of data is thus reduced by increasing the training data amount.



*Figure 4.1.1: Key-Points Identification using FAN, a state-of-the-art deep learning based on FAN face alignment method*

General problems of Object Tracking and Classification:

- Occlusion: This is the technical term used to describe the problem of an object that is moving behind or being obscured by another object. Occlusion can also occur when the object moves outside of the frame bounds.
- Clutter: In computer vision it is difficult to deal with the presence of another similar looking distracting object in the scene, this problem is called Clutter.

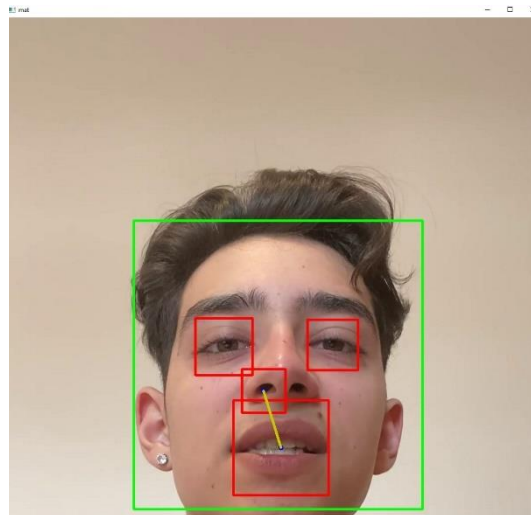


Figure 4.1.2: Cluttering Problem using OpenCV algorithm

## 4.2. Face Alignment

Face alignment is accomplished by translating numerous point sets from input data into a single coordinate system. This is known as the output coordinate system, and it serves as our stationary reference frame. The purpose is to warp and transform all input coordinates to match output coordinates. This is accomplished through the use of three fundamental affine transformations: rotation, translation, and scaling [6]. The FAN framework was used in conjunction with the OpenCV framework to achieve these translations.

### 4.2.1. FAN 2D Framework

The Fan 2D Framework relies on landmark localization, specifically facial landmark localization, commonly known as face alignment, which is an intensively investigated area in computer vision in recent decades. Depending on the objective at hand, numerous strategies for landmark localization have been utilised in the past. Prior to the introduction of neural networks, research in human pose estimation was mostly centred on picture structures and complex extensions because of their capacity to represent massive appearance variations and support a wide range of human positions.

However, for the job of face alignment, these techniques have not been proved to be capable of obtaining the high level of accuracy demonstrated by cascaded regression techniques. Recently, fully Convolutional Neural Network designs based on heatmap regression have revolutionised human posture prediction, yielding results with exceptional accuracy even for the most difficult datasets. Such methods can be easily applied to the problem of face alignment due to their end-to-end training and lack of reliance on hand engineering. FAN 2D Framework builds and trains such a powerful network for face alignment for the first time, and analyses how far it is from achieving near-saturating performance on all current 2D face alignment datasets [7].

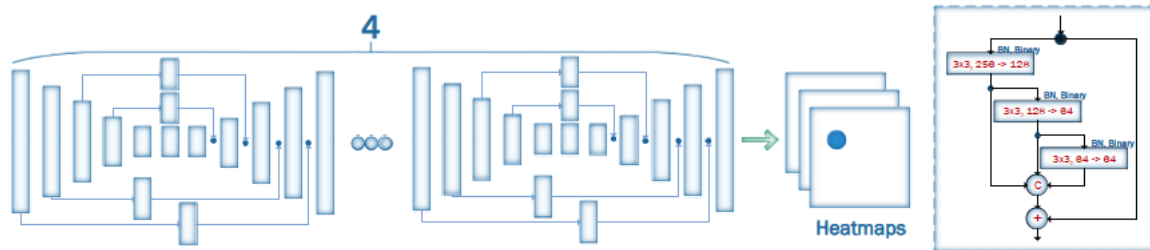


Figure 4.2.1.1: The FAN 2D Framework [7]

<https://www.adrianbulat.com/face-alignment>

## 5. Audio Pre-Processing

The initial stage in automatic speech recognition systems is to extract features or detect the input audio signal characteristics that are useful for identifying language features while ignoring everything else, such as background noise and sentiment. In this part the way of processing wav files to extract Mel-frequency cepstral coefficients (MFCC) is described.

### 5.1.1. Mel-Frequency

The most important fact to notice about speech is that the structure of the vocal tract, which includes the tongue and teeth, filters the sounds produced by a human. The sound that emerges is determined by this shape. If we can precisely establish the structure, we should be able to accurately capture the phoneme being produced. The envelope of the short time power spectrum reflects the curvature of the vocal tract, and MFCCs' purpose is to appropriately capture this envelope. Mel Frequency Cepstral Coefficients (MFCCs) are a common component in speaker detection and artificial speech systems. They were first launched by Davis and Mermelstein in the 1980s, and they have remained cutting-edge ever since. Prior to the discovery of MFCCs, the major feature category for automatic speech recognition (ASR) was linear prediction coefficients (LPCs) and linear prediction cepstral coefficients (LPCCs).

## 6. Future Work

---

### 6.1. Planned Deliverables

In order to meet the objectives of this project the following deliverables are deemed necessary.

1. Dataset Design
  - Recording of 20 videos that include the Harvard Sentences
  - Separate video and audio for all examples with ffmpeg
2. Visual Data pre-processing
  - Align, crop, resize and save all frames into a specific folder in the project directory.
3. Extraction of audio features from wav files
  - Loading of audio files into python
  - Process them and extract the MFCC features, features commonly used in speech recognition
  - Save MFCC vectors as numpy array.
4. Audio to Image prediction network
  - Creation of a pytorch dataset class to load pairs of images and audio features
  - Design a simple network that maps MFCC features to images
  - Develop a training algorithm.
5. Sufficient documentation to ensure that the work can be replicated.

The first two goals have already been achieved, and only minor additions and modifications will be made. By creating this report and documenting work on the Visual and Audio pre-processing, a big step towards deliverable four has been taken. From here on out, the focus will be on defining and executing the training algorithm.

Changes to some of the present face alignment parameters will be made in the future to provide an even better dataset for the training phase. The dataset class to import a pair of images and audio features will be constructed after the visual and audio pre-processed dataset is ready.

## 6.2. Project Plan

For both semesters, detailed Gantt Charts were developed near the start of the project, with tasks broken down into subtasks. They are located in Appendix B for readability due to their size. A project is never risk-free, regardless of how well-planned it is. Table 6.2.1 shows a summary of the project's primary risks, as well as their potential influence on the project and solutions to avoid or mitigate the problem.

<b>Risk Description</b>	<b>Consequences</b>	<b>Impact</b>	<b>Precautionary Measures</b>
Dataset might not be recorded properly.	It will take some time to record the dataset again.	Low.	Begin recording the dataset in a well-lit area with a white background and a high-resolution camera.
Script Development might give errors.	Finding out why these issues occur online may cause the project to be delayed because it cannot be completed before all of the necessary processes are coded.	Severe.	Work efficiently and strive to handle issues with the assistance that online resources provide.
Dataset might not be processed properly.	Changing the settings of the face alignment code and audio pre-processing may cause the training algorithm to take longer to develop.	Moderate.	Understanding the requirements of the training algorithm and processing the dataset in accordance with them.
Loss of code	If any necessary code is lost, it must be recreated.	Low.	Upload code to GitHub so that it is stored every time it is changed.

Figure 6.2.1.: Risks to Project, their Impacts and Consequences, and Measures to avoid them.

## 7. Conclusions

---

The Audio Driven Face Animation project has generated a set of translated frames and audio samples that will be used in the training phase. It's now time to develop a dataset class that loads pairs of photos and audio features, as well as a model that maps MFCC features to images. To construct a training model that creates a digital face that can utter additional words, a deeper understanding of Machine Learning is required. However, the project is on schedule to be completed in May 2022, as anticipated.

## References

---

- [1] Saha, A., 2017. *Read, Write and Display a video using OpenCV* /. [online] LearnOpenCV – OpenCV, PyTorch, Keras, Tensorflow examples and tutorials. Available: <<https://learnopencv.com/read-write-and-display-a-video-using-opencv-cpp-python/>> [Accessed 5 December 2021].
- [2] Karras, T., Aila, T., Laine, S., Herva, A. and Lehtinen, J., 2017. Audio-Driven Facial Animation by Joint End-to-End Learning of Pose Emotion. [online] Research.nvidia.com. Available: <[https://research.nvidia.com/sites/default/files/publications/karras2017siggraph-paper\\_0.pdf](https://research.nvidia.com/sites/default/files/publications/karras2017siggraph-paper_0.pdf)> [Accessed 5 December 2021].
- [3] Learning, I., 2021. Introduction To Machine Learning | Application of Machine Learning. [online] EDUCBA. Available: <<https://www.educba.com/introduction-to-machine-learning/?source=leftnav>> [Accessed 5 December 2021].
- [4] Ghanghav, K., 2021. *Conundrum of Deepfakes: An Overview and analysis of recent advancements*. [online] Wvww.easychair.org. Available at :<[https://wvww.easychair.org/publications/preprint\\_download/RK3j](https://wvww.easychair.org/publications/preprint_download/RK3j)> [Accessed 5 December 2021].
- [5] Zakharov, E., Shysheya, A., Burkov, E. and Lempitsky, V., 2019. Few-Shot Adversarial Learning of Realistic Neural Talking Head Models. [online] Arxiv.org. Available at: <<https://arxiv.org/pdf/1905.08233v2.pdf>> [Accessed 6 December 2021].
- [6] Master Data Science. 2021. How to align faces with OpenCV in Python. [online] Available at: <<https://datahacker.rs/010-how-to-align-faces-with-opencv-in-python/>> [Accessed 10 June 2020].
- [7] Bulat, A. and Tzimiropoulos, G., 2017. How far are we from solving the 2D & 3D Face Alignment problem? (and a dataset of 230,000 3D facial landmarks) | Adrian Bulat. [online] Adrianbulat.com. Available at: <<https://www.adrianbulat.com/face-alignment>> [Accessed 7 December 2021].
- [8] Practicalcryptography.com. 2021. Practical Cryptography. [online] Available at: <<http://www.practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/>> [Accessed 7 December 2021].
- [9] Faruque, T., Kapoor, A., Kate, R., Rajput, N. and Subramaniam, L., 2016. AUDIO DRIVEN FACIAL ANIMATION FOR AUDIO-VISUAL REALITY. [online] Cpb-us-w2.wpmucdn.com. Available at: <<https://cpb-us-w2.wpmucdn.com/sites.uwm.edu/dist/a/171/files/2016/11/icme01-t32x3j.pdf>> [Accessed 19 December 2021].



## Appendix A

The present state of the audio-driven facial animation is demonstrated in this section.

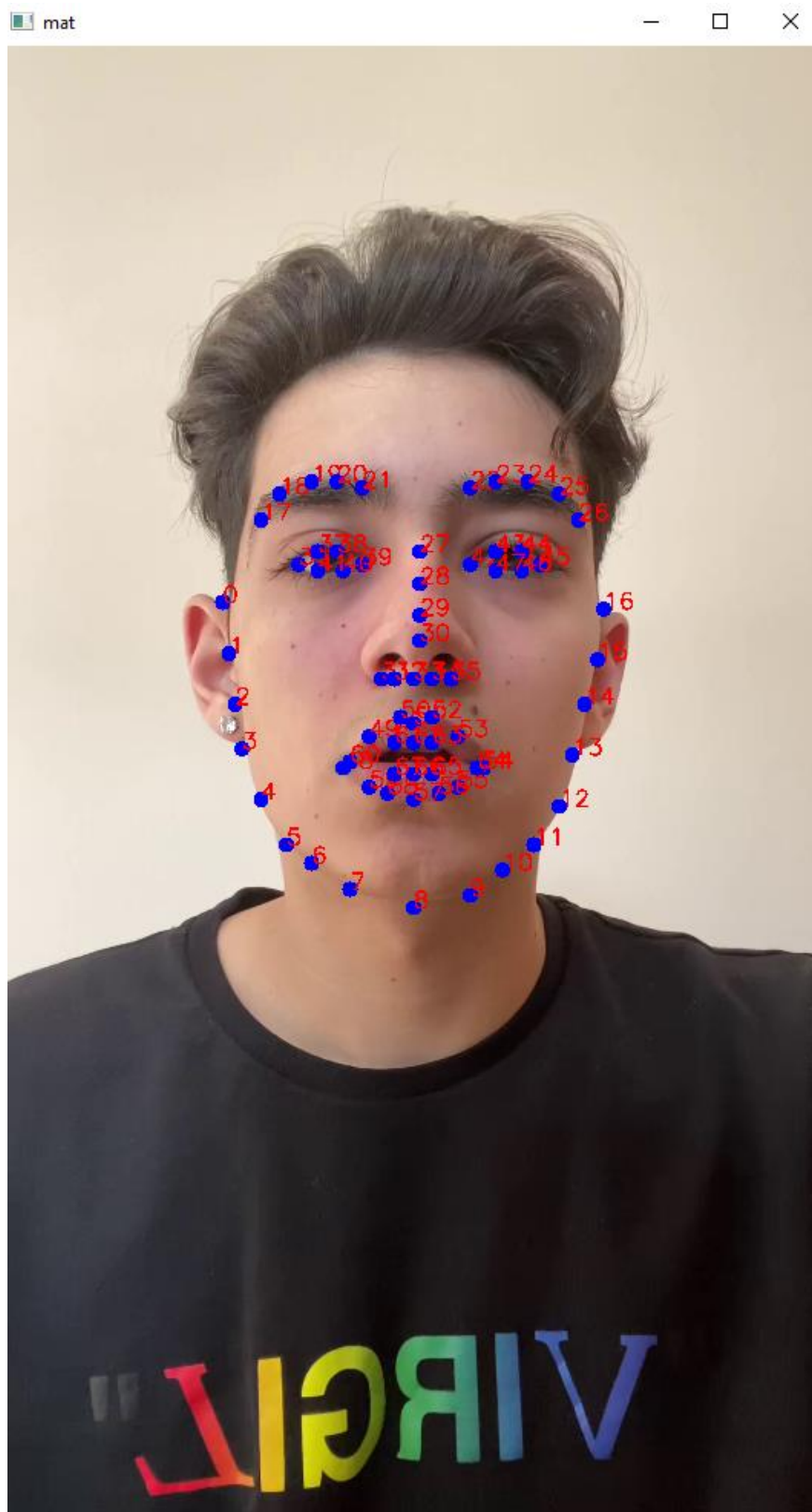


Figure 0-1: Face Alignment Key-points that help for the rotation part in pre-processing.

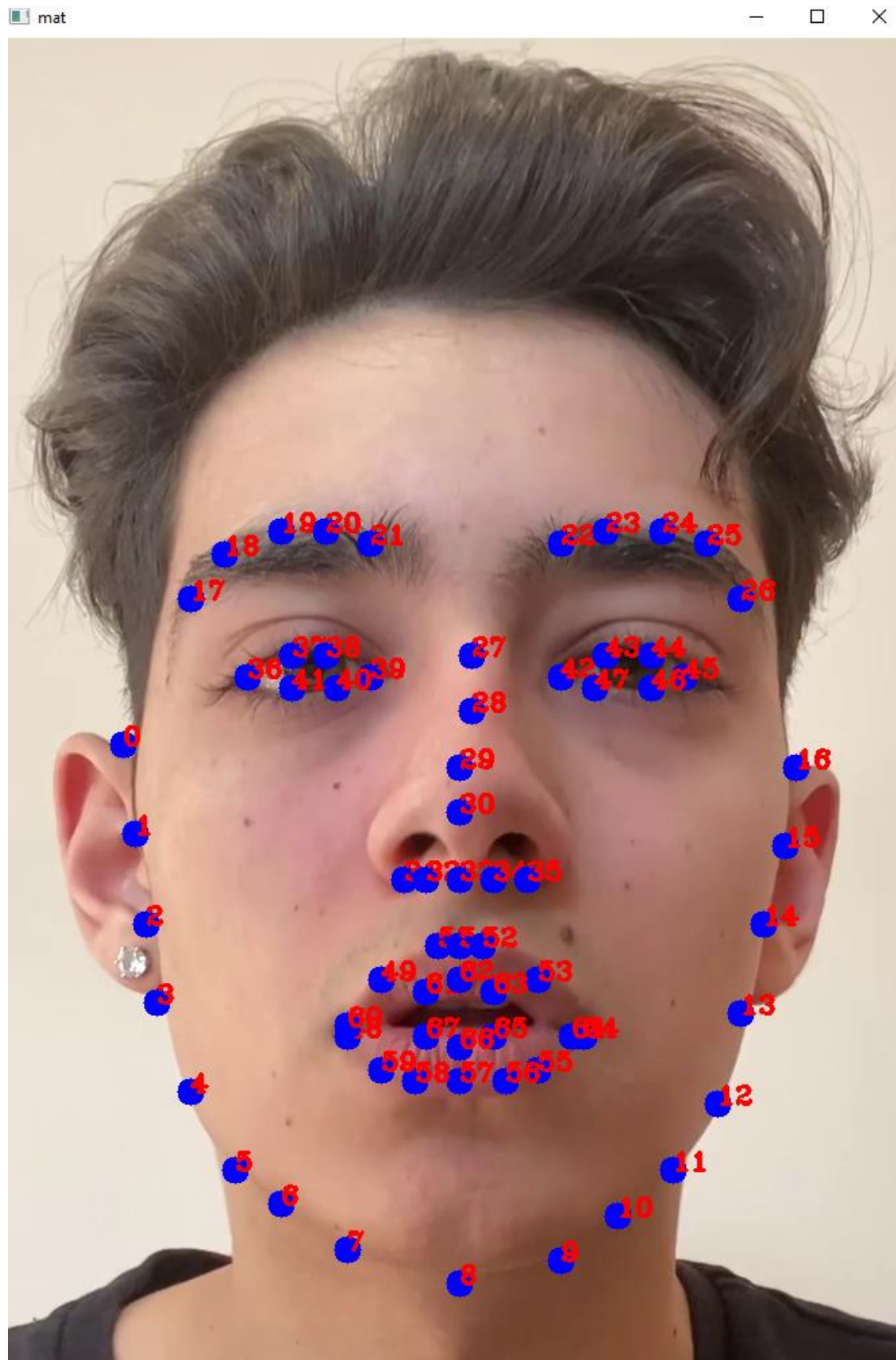


Figure 0-2: Frames after cropping

```
fps.py M X
C: > Users > dinok > Documents > Dissertation > Code > fps.py > ...
1 import face_alignment
2 import cv2
3 import matplotlib.pyplot as plt
4 import numpy as np
5 import torch
6 import os
7 from skimage import io
8 import collections
9
10
11 # Function os.scandir() iterates through a folder
12 # Providing the path of the directory
13 # r = raw string literal
14 dirloc = r"C:\Users\dinok\Documents\Dissertation\Dataset\Seperated"
15 s_path = r"C:\Users\dinok\Documents\Dissertation\Dataset\clip\images"
16
17 fa = face_alignment.FaceAlignment(face_alignment.LandmarksType._2D, device='cpu', face_detector='blazeface')
18
19
20 def fun(det):
21     return [tuple(i) for i in det]
22
23 def crop_square(img, size, interpolation=cv2.INTER_AREA):
24     h, w = img.shape[:2]
25     min_size = np.amin([h,w])
26
27     # Centralize and crop
28     crop_img = img[int(h/2-min_size/2):int(h/2+min_size/2), int(w/2-min_size/2):int(w/2+min_size/2)]
29     resized = cv2.resize(crop_img, (size, size), interpolation=interpolation)
30
31     return resized
32
33
34
35
36 i = 0
37 j = 0
38
39 # calling scandir() function
40 for file in os.scandir(dirloc):
41     if(file.path.endswith(".mp4")) and file.is_file():
42         # Create a VideoCapture object and read from input file
43         # cap = cv2.VideoCapture(file.path)-----
44         cap = cv2.VideoCapture(file.path)
45
46         print(file)
47
48         # Check if camera opened successfully
49         if (cap.isOpened() == False):
```

Figure 0-3: Visual-Data Pre-Processing code

```

50 |         print("Error opening video stream or file")
51 |
52 |         # Default resolutions of the frame are obtained. The
53 |         # default resolutions are system dependent.
54 |         # We convert the resolutions from float to integer.
55 |         frame_width = int(cap.get(3))
56 |         frame_height = int(cap.get(4))
57 |         fps = int(cap.get(5))
58 |         #print('Width %d'%frame_width)
59 |         #print('Height %d'%frame_height)
60 |         #print('fps %d'%fps)
61 |
62 |         # Define the codec and create VideoWriter object.The
63 |         # output is stored in 'outpy.avi' file.
64 |         #out = cv2.VideoWriter('outpy.avi', cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'),10, (frame_width, frame_height))
65 |         i = 1
66 |         j = 1
67 |         # Read until video is completed
68 |         while(True):
69 |             # Capture frame-by-frame
70 |             ret, frame = cap.read()
71 |             if ret == True:
72 |                 frame_temp = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)#Changes the colour space to RGB
73 |
74 |                 #Run facial detection
75 |
76 |                 det = fa.get_landmarks_from_image(frame_temp)#Process entire directory in one go
77 |
78 |                 ## plot_style = dict(marker='o',
79 |                 ##                     markersize=4,
80 |                 ##                     linestyle='-',
81 |                 ##                     lw=2)
82 |
83 |                 ## det_type = collections.namedtuple('prediction_type', ['slice', 'color'])
84 |                 ## det_types = {'face': det_type(slice(0, 17), (0.682, 0.780, 0.909, 0.5)),
85 |                 ##             'eyebrow1': det_type(slice(17, 22), (1.0, 0.498, 0.055, 0.4)),
86 |                 ##             'eyebrow2': det_type(slice(22, 27), (1.0, 0.498, 0.055, 0.4)),
87 |                 ##             'nose': det_type(slice(27, 31), (0.345, 0.239, 0.443, 0.4)),
88 |                 ##             'nostril': det_type(slice(31, 36), (0.345, 0.239, 0.443, 0.4)),
89 |                 ##             'eye1': det_type(slice(36, 42), (0.596, 0.875, 0.541, 0.3)),
90 |                 ##             'eye2': det_type(slice(42, 48), (0.596, 0.875, 0.541, 0.3)),
91 |                 ##             'lips': det_type(slice(48, 60), (0.596, 0.875, 0.541, 0.3)),
92 |                 ##             'teeth': det_type(slice(60, 68), (0.596, 0.875, 0.541, 0.4))
93 |                 ## }
94 |
95 |
96 |                 ##fig = plt.figure(figsize=plt.figaspect(.5))

```

Figure 0-4: Visual-Data Pre-Processing code

```
96     ##fig = plt.figure(figsize=plt.figaspect(.5))
97     ##ax = fig.add_subplot(1,2,1)
98     ##ax.imshow(frame)
99
100     ##for det_type in det_types.values():
101     ##     ax.plot(det[det_type.slice,0],
102     ##            det[det_type.slice,1],
103     ##            color=det_type.color, **plot_style)
104
105     ##ax.axis('off')
106     index = 0
107     #for detection in det[0]:
108     #     # print(detection[0])
109     #     # cv2.circle(frame,(int(detection[0]), int(detection[1])), 10, (255,0,0), -1)
110     #     #cv2.putText(frame, '%d'%index, (int(detection[0]), int(detection[1])), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1 ,(0,0,255), 2)
111     #     # index = index+1
112
113
114
115     #Get the average of a list
116     def Average(lst):
117         return sum(lst) / len(lst)
118
119     lst_left_eye = [det[0][36], det[0][37], det[0][38], det[0][39], det[0][40], det[0][41]]
120     lst_right_eye = [det[0][42],det[0][43], det[0][44], det[0][45], det[0][46], det[0][47]]
121
122
123
124     average_left_eye = Average(lst_left_eye)
125     average_right_eye = Average(lst_right_eye)
126     #print("Average of the list = ", average_left_eye)
127     #print("Average of the list = ", average_right_eye)
128
129     left_eye_x = average_left_eye[0]
130     left_eye_y = average_left_eye[1]
131
132     right_eye_x = average_right_eye[0]
133     right_eye_y = average_right_eye[1]
134
135     if left_eye_y > right_eye_y:
136         A = (right_eye_x, left_eye_y)
137         #Integer -1 indicates that the image will rotate in the clockwise direction
138         direction = -1
139
140     else:
141         A = (left_eye_x, right_eye_y)
142         #Integer 1 indicates that image will rotate in the counter clockwise
143         #direction
144         direction = 1
```

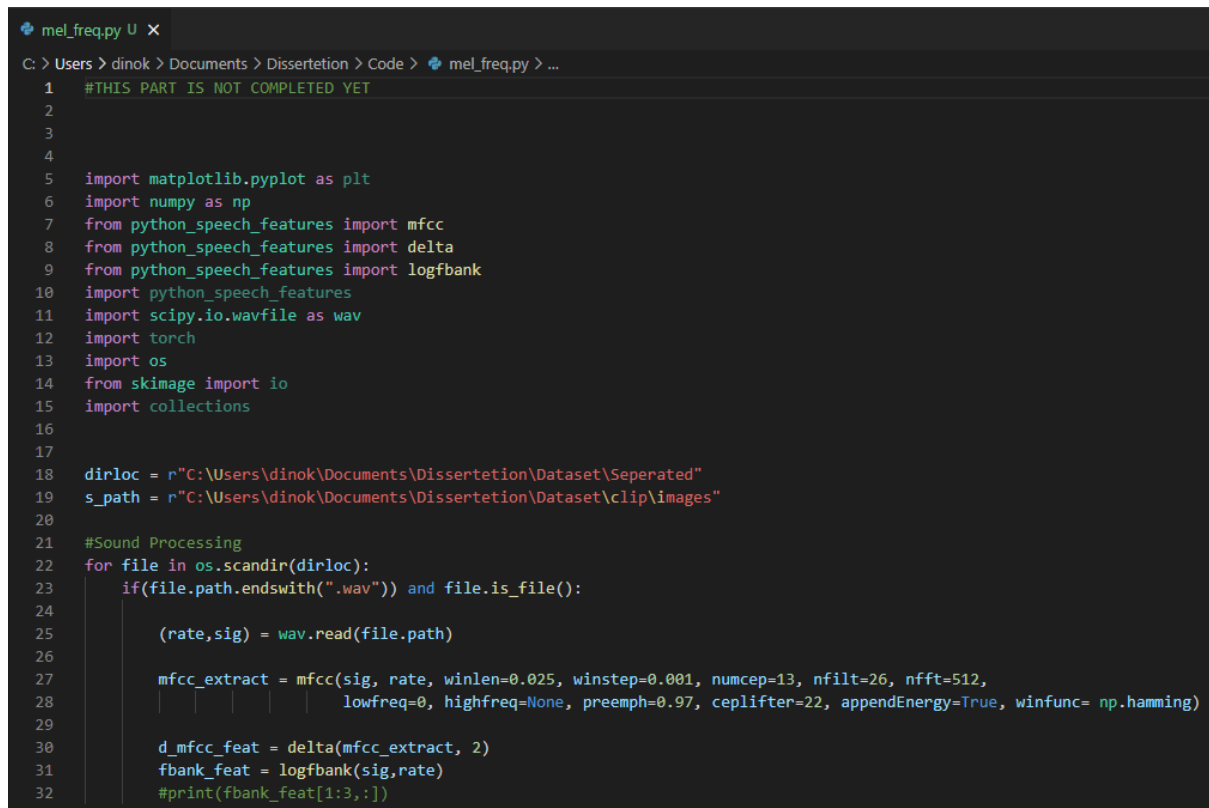
Figure 0-5: Visual-Data Pre-Processing code

```
144 |         direction = 1
145 |
146 |
147 |
148 |         delta_x = right_eye_x - left_eye_x
149 |         delta_y = right_eye_y - left_eye_y
150 |
151 |         angle = np.arctan(delta_y/delta_x)
152 |         angle = (angle * 180) / np.pi
153 |
154 |         h, w = frame.shape[:2]
155 |
156 |         #Calculating a center point of the video
157 |         #Integer division "//" ensures that we receive whole numbers
158 |         center = (w // 2, h // 2)
159 |
160 |         #Defining a matrix M and calling
161 |         #cv2.getRotationMatrix2D method
162 |         M = cv2.getRotationMatrix2D(center, (angle), 1.0)
163 |
164 |         #Applying the rotation to our video using the cv2.warpAffine method
165 |         rotated = cv2.warpAffine(frame, M, (w, h))
166 |
167 |
168 |         #Crop the image and scale to 256x256
169 |
170 |         # calculate distance between the eyes in the first image
171 |         # dist_1 = np.sqrt((delta_x * delta_x) + (delta_y * delta_y))
172 |
173 |         # calculate distance between the eyes in the second image
174 |         #dist_2 = np.sqrt((delta_x_1 * delta_x_1) + (delta_y_1 * delta_y_1))
175 |
176 |
177 |
178 |         dim = (256,256)
179 |         crop_img = rotated[200:1150, 0:1080]
180 |         #resized = crop_square(rotated, 256, interpolation=cv2.INTER_AREA)
181 |         resized = cv2.resize(crop_img, dim,interpolation=cv2.INTER_AREA)
182 |         #resized = cv2.resize(rotated,dim)
183 |         cv2.imshow('mat', resized)
184 |
185 |
186 |         #//SAVING PART//
187 |         name = './Dataset/' + file.name + '/images/%04d.png'%j
188 |         print('Creating...' + name)
189 |         os.makedirs('./Dataset/' + file.name + '/images/', exist_ok=True)
190 |         cv2.imwrite(name, resized)
191 |         j += 1
```

Figure 0-6: Visual-Data Pre-Processing code

```
190         cv2.imwrite(name, resized)
191         j += 1
192
193
194
195
196
197         # Write the frame into the file 'output.avi'
198         #out.write(frame)
199
200         # Display the resulting frame
201         #cv2.imshow('frame', frame)
202         #cv2.waitKey(0) number inside waitKey is equal to the time in milliseconds
203         #we want each frame to be displayed
204
205
206
207         # Press Q on keyboard to exit
208         if cv2.waitKey(1) & 0xFF == ord('q'):
209             break
210         # Break the loop
211     else:
212         break
213
214 # When everything done, release the video capture object
215 cap.release()
216 #out.release()
217 # Closes all the frames
218 cv2.destroyAllWindows()
219
```

Figure 0-7: Visual-Data Pre-Processing code



```
mel_freq.py U x
C: > Users > dinok > Documents > Dissertation > Code > mel_freq.py > ...
1  #THIS PART IS NOT COMPLETED YET
2
3
4
5  import matplotlib.pyplot as plt
6  import numpy as np
7  from python_speech_features import mfcc
8  from python_speech_features import delta
9  from python_speech_features import logfbank
10 import python_speech_features
11 import scipy.io.wavfile as wav
12 import torch
13 import os
14 from skimage import io
15 import collections
16
17
18 dirloc = r"C:\Users\dinok\Documents\Dissertation\Dataset\Seperated"
19 s_path = r"C:\Users\dinok\Documents\Dissertation\Dataset\clip\images"
20
21 #Sound Processing
22 for file in os.scandir(dirloc):
23     if(file.path.endswith(".wav")) and file.is_file():
24
25         (rate,sig) = wav.read(file.path)
26
27         mfcc_extract = mfcc(sig, rate, winlen=0.025, winstep=0.001, numcep=13, nfilt=26, nfft=512,
28                             lowfreq=0, highfreq=None, preemph=0.97, ceplifter=22, appendEnergy=True, winfunc= np.hamming)
29
30         d_mfcc_feat = delta(mfcc_extract, 2)
31         fbank_feat = logfbank(sig,rate)
32         #print(fbank_feat[1:3,:])
```

*Figure 0-8: Audio-Data Pre-Processing*



## Appendix B

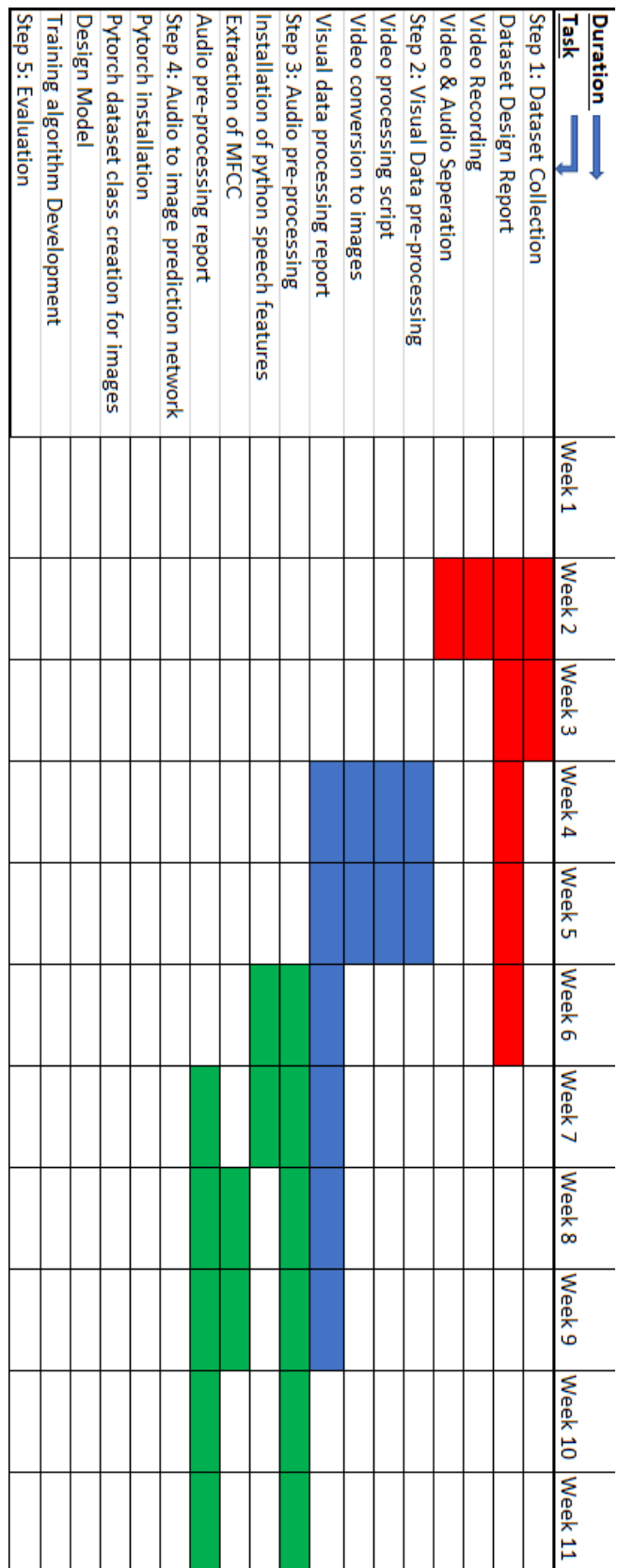


Figure 3: Gantt Chart for Semester 1

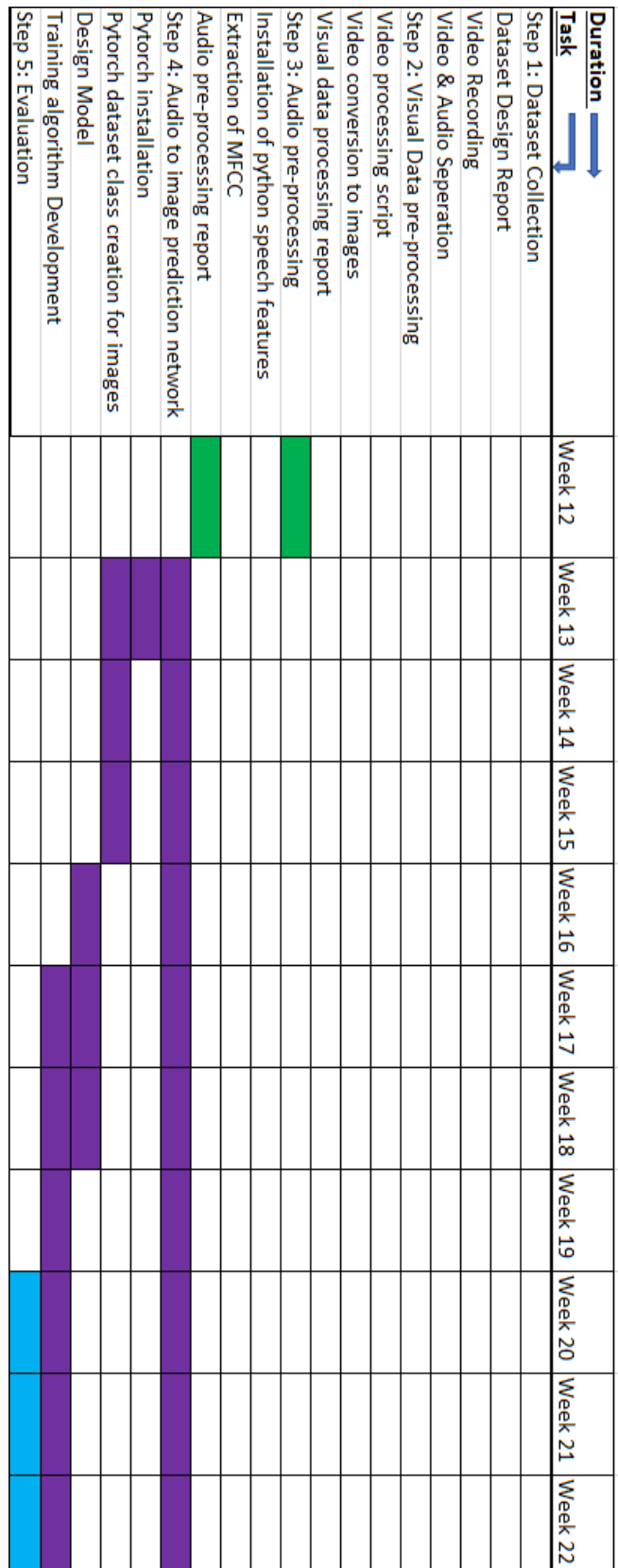


Figure 4: Gantt Chart for Semester 2