Enhancing Real-Time, Sentence-Level Lip-Reading Accuracy.

A Project Report submitted for the partial fulfilment of the requirements of the BSc(Hons) Degree in Data Science

By

D. JAYENDRA PRABHASHI MAHAWATTA

COBScDS231P-009

National Institute of Business Management,

Colombo, Sri Lanka

2nd May 2024

Declaration.

I hereby declare that the work presented in this project report was carried out independently by myself and have cited the work of others and given due reference diligently.

Signature of the Candidate. Date:

(D.J. Prabhashi Mahawatta)

3rd May 2024

……………………………..…… …………………………

I certify that the above student carried out his/her project under my supervision and guidance.

Signature of the Supervisor. Date:

(Mr. Ashan Arthanayake)

……………………………..…… ..……………………………

Acknowledgment.

Upon completing the research project, I would like to extend my sincere appreciation to those who provided invaluable support throughout my journey in the last year research project.

First and foremost, I am deeply grateful to my Research Project supervisor, Prof. Ashana, for his unwavering guidance and encouragement in shaping the course of my research. His expertise and mentorship were instrumental in the successful completion of this project.

Additionally, I would like to express my heartfelt gratitude to Miss. W.M.S.G.D.C. Wanigasekara, the Course Director of my degree program for her complete and unrelenting support with the completion of this paper.

The support and guidance I have received throughout this journey have been instrumental in the successful completion of this thesis. I am sincerely thankful for your contributions and support.

Lastly, I must express my very profound gratitude to my parents and sisters for providing me with unfailing support and continuous encouragement throughout my years of study. This accomplishment would not have been possible without them.

Thank you.

Abstract.

Automatic lip reading, also known as visual speech recognition (VSR), is the technique of deciphering spoken language from visual cues provided by lip movements. This thesis investigates the development of a deep learning model for sentence-level lip reading. The proposed model uses a 3D convolutional neural network (CNN) architecture to extract features from video sequences and bidirectional long short-term memory (LSTM) networks to capture temporal correlations within lip motions.

The model achieved a remarkable accuracy of 99.2% for sentence-level lip reading in a controlled environment. However, the intricate design intended for extreme precision, which led to large processing needs, hindered real-time deployment on devices with constrained resources.

This thesis emphasizes how real-time performance and accuracy in lip reading models are traded off. Adaptive learning methods, hardware acceleration, data augmentation methods, and lightweight model architectures are some of the future work directions that are discussed. Future developments may result in sentence-level, real-time lip-reading systems that greatly improve communication accessibility by tackling these issues.

Table of Contents.

List of Figure

1. Introduction

For around 29 million Americans—more than 11% of all adult Americans—using only hearing aids would be advantageous. High levels of noise exposure and aging are the two main causes of hearing loss. More people are experiencing hearing loss as a result of persistent sound exposure via devices like headphones and earbuds, which is linked to technological advancements. 20 percent of people on the planet, or 1.5 billion people, are estimated to suffer hearing loss by the World Health Organization. There are 435 million people who are classified as permanently deaf or hard of hearing out of these 1.5 billion.

Though it can be challenging for a beginner, the ability to understand what is being said only from visual cues is an outstanding skill. Because there is no context while lipreading, the task is naturally unclear at the word level and in short sentences. In addition, certain characters, known as homophemes, are hard to identify from one another. However, this challenge can be solved by utilizing the neighborhood information found in the ontext. The ambiguity of homophones can be reduced with the aid of nearby words. Although studies show that only about 30 to 45 percent of the English language can be understood through lip reading alone, lip reading can be understood through lip reading alone, lip reading can often assist speech understanding. Adding visual signals provides the greatest chance for success in conversations, and increases one's confidence in their ability to communicate.

Lip reading is an issue that is still very much in use today and presents a significant difficulty for machine learning and artificial intelligence approaches. Lip reading could be utilized by security personnel in instances where it's important to identify a person's speech while the audio record is unavailable, or it could benefit hearing impaired people by improving speech recognition in noisy environments. It is impossible to manually develop a computer system that can read lips because of the sheer number of languages spoken, the vocabulary unique to each, and the wide range of articulation across individuals. Even human professionals in this field are able to correctly estimate just about every second word [1,3] and only under ideal conditions. Therefore, the problem of lip reading is a perfect candidate for solution using Artificial Intelligence (AI).

1.1 Related Work

1.1.1 Use of Deep Neural Networks.

Since they are so good at extracting complex patterns and motion from visual input, deep neural networks, or CNNs, have become a potent instrument in the lip-reading space. The ability of DNNs to understand hierarchical features and relationships is beneficial for complex tasks like lip reading. In order to extract lip movements, evaluate video frames, and extract information encoded in lip motions, researchers are increasingly using DNNs, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) [07]. When it comes to tasks like word and phrase identification, DNNs are especially useful in lip reading. Empirical studies have demonstrated that combining data from many modalities—including text, voice, and video can greatly increase recognition accuracy [02], [03]. Because DNNs can automatically discover intricate patterns and relationships within multimodal data, they are highly suited for multimodal analysis. As a result, lip movements are understood more thoroughly.

1.1.2 End to End Learning.

End-to-end lip reading is the practice of directly transcribing speech from lip movements using a single neural network model without explicitly breaking the process down into discrete steps (e.g., phoneme recognition).[05] With this method, a series of video frames displaying lip movements are fed into the model, which outputs a transcription of the spoken words.

This method is different from conventional lip reading systems, which usually go through several steps, such as language modeling, phoneme recognition, and lip feature extraction. By learning each of these processes at once, end-to-end lip reading seeks to streamline the procedure and may increase resilience and accuracy. [01],[04]

The study [06] shows, managing changes in illumination, posture, and facial emotions, as well as the significant variety in lip movements associated with various speakers and languages, are major obstacles in end-to-end lip reading. To overcome these obstacles and attain cutting-edge performance in lip reading tasks, convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are two examples of complex deep learning techniques.

1.1.3 Sentence-Level Lipreading.

Lip reading, also known as visual speech recognition, has traditionally focused on recognizing individual words or phonemes (units of sound) from visual cues. However, recent research has made significant strides in achieving sentence-level lip reading, enabling more natural and comprehensive communication. By employing advanced deep learning techniques, researchers are pushing the boundaries of what's possible. LipNet[01] is an end-to-end model that learns both spatiotemporal visual features and a sequence model simultaneously. This approach could be beneficial for your project, as it allows for direct mapping from video frames to text, potentially improving the accuracy and efficiency of your real-time lip-reading system. LipNet achieves a high accuracy of 95.2% in sentence-level lipreading on the GRID corpus, outperforming experienced human lipreaders and previous word-level state-of-the-art accuracy. This benchmark can serve as a reference for evaluating the performance of my own model. The system performs well only under controlled conditions, but its potential may not have been completely realized in a variety of real-world settings, such as those with complicated backgrounds, varied lighting, and different angles.

Sentence-Level Lipreading Using Convolutional Neural Networks: This study focuses on using convolutional neural networks (CNNs) for sentence-level lip reading and achieves high accuracy on a challenging dataset.

1.1.4 Real-time lip reading.

Real-time lip reading refers to the ability of a system to recognize spoken words or phrases by analyzing lip movements in live or recorded video in real-time. The study [08] focused on real-time lip reading for human-computer interaction applications. The researchers developed a system that could recognize a limited set of words and phrases in real-time using hidden Markov models (HMMs) for lipreading.

Using convolutional neural networks can achieve high accuracy in real-time lip reading tasks, paving the way for practical applications in human-computer interaction and assistive technologies. [09]

This project [10] creates an automated lip-reading system using neural networks and deep learning to validate the application of machine learning. Two different CNN architectures were used to train a portion of the dataset. The best performing model was implemented in a web application for real-time word prediction.

2. Research Gap.

The extensive body of research supporting this study on real-time sentence-level lip reading shows the tremendous advancements made in the field, especially with regard to the application of deep neural networks (DNNs), end-to-end learning, and sentence-level lipreading techniques. There are still a number of research gaps and obstacles in spite of these developments, which presents chances for more investigation and creativity.

The need for better model resilience and adaptability to real-world settings is a significant research gap. Numerous previous researches, including LipNet, show good accuracy in controlled environments but struggle with variables like lighting variations, posture, expressions on the face, variety in speakers, and different languages. For practical implementation in real-world circumstances, such as noisy venues, various lighting conditions, and different speaker characteristics, it is imperative to improve the models' capacity to manage these difficulties.

Another gap is the scalability and efficiency of real-time lip reading systems. Even while they seem promising, the present approaches might not always provide the best real-time performance, especially when dealing with large amounts of video data. It could be quite helpful to improve real-time application usability and practicality by creating more effective algorithms and designs.

Furthermore, larger datasets and evaluation tools specific to lip reading at the phrase level are needed. Existing datasets, such as the GRID corpus, are helpful starting points, but they don't fully capture the variety and complexity of real-world scenarios. Standardized datasets encompassing a wide range of languages, speakers, and environmental factors should be produced in order to facilitate more comprehensive testing and comparison of different models.

Furthermore, the interpretability and explainability of deep learning models for lipreading remain little unexplored. Understanding the decision-making processes of these models and the properties on which they rely may improve their dependability and facilitate a better understanding and analysis of the findings by users, especially in critical applications like assistive technology and human-computer interaction.

3. Research Problem.

This study aims to solve the research topic of developing deep learning models to enhance lip reading accuracy in noisy conditions. Even though lip reading has potential benefits, such as improved speech recognition in noisy or loud environments, existing techniques struggle to reliably identify lip movements, especially in practical situations. By developing strong deep learning models that can comprehend lip movements in noisy settings where more conventional audio-based speech recognition systems could falter, our research seeks to overcome these difficulties.

To be more precise, the study aims to accomplish multiple goals. Initially, the goal is to apply algorithms for noise reduction in order to eliminate extraneous visual information and enhance the models' capacity to precisely understand lip movements in noisy settings. Second, by perhaps training on sentence-containing datasets like GRID, the project seeks to improve overall accuracy and reliability by strengthening the system's identification ability of a wide lexicon.

Third, in order to create multimodal systems that are more dependable and precise, the study looks into cross-modal integration, which combines lip reading with other modalities including audio-based speech recognition. Fourth, it attempts to create a computer software that can automatically and precisely identify words spoken by focusing just on the visible lip movements of the speaker—a difficult effort that is beyond the capabilities of conventional approaches.

Lastly, the study intends to further academic research in the domains of deep learning and real-time lip reading, adding to the general knowledge and advancement of these technologies by enhancing sentence-level lip reading accuracy in noisy environments. The potential influence on accessibility—especially for the deaf community—by offering more dependable communication tools is the main driving force behind this project. Enhancements in the accuracy of lip reading may also have consequences for security and surveillance applications, where accurate interpretation of visual cues is essential.

4. Research Objective.

4.1 Main Objective.

Develop a deep learning model for lip-reading that can accurately recognize speech, with a focus on sentence-level accuracy, to improve communication for the hard-of-hearing and deaf.

4.2 Specific Objectives:

4.2.1 Improved Lipreading at the Sentence Level.

- Learn how to effectively use the 3D convolutional neural network architecture to extract features from lip movements for sentence detection.
- Use a sentence-based dataset, such the GRID dataset, to train the model to identify the sequential nature of speech using lip movements.
- In order to evaluate the model's performance on sentence-level recognition tasks, use appropriate metrics like Word Error Rate (WER) or Sentence Error Rate (SER).
- Analyze how sentence complexity and length affect accuracy.

4.2.2 Better Vocabulary Recognition.

- Train the model on many datasets (e.g., GRID dataset) to improve its word recognition ability for sentence-level understanding.
- Analyze the effects of dataset composition and size on the accuracy of vocabulary recognition in sentence context.
- Consider strategies for handling vocabulary words that you may encounter when identifying sentences.

4.2.3 Optional Multimodal Integration.

- Look investigate ways to integrate lip-reading with audio-based speech recognition, if audio is available, to determine if this can improve sentence-level recognition, especially for complicated sentences.
- Build a framework that considers potential duplication and synchronization problems when integrating different modalities into a deep learning architecture.
- Evaluate how much sentence-level precision is improved by multimodal integration.

4.2.4 Real-Time Performance.

- Look on approaches to make the 3D CNN architecture more efficient while accounting for the processing requirements of sentence-level recognition so that low-resource devices can analyze real-time data.
- In order to achieve acceptable sentence-level accuracy, find a trade-off between real-time performance and model complexity.

5. Methodology.

5.1 System Overview.

This system tackles the challenge of sentence-level lip-reading using deep learning to improve communication accessibility for the deaf and hard-of-hearing. Here's a breakdown of its core components.

Getting data is the initial stage in the process. Videos of people saying different things will be gathered. A wide variety of speakers should be included in the dataset in terms of age, ethnicity, speaking enunciation, and sentence complexity. One useful starting point is the GRID dataset, which has movies with sentences in them. To improve the generalizability of the model, further data may be included.

Preprocessing is done on the data after it is gathered to guarantee consistency and make feature extraction easier. This could entail normalizing the frames to a consistent size and range, cropping the frames to concentrate on the speaker's mouth and surrounding area, and turning the movies to grayscale.

A 3D convolutional neural network (CNN) assumes central role after preprocessing. This network is skilled at identifying important characteristics in video footage. It probably has several convolutional layers that are able to extract temporal and spatial information from successive video frames. In order to incorporate non-linearity and enable the network to understand intricate correlations within the data, activation functions such as ReLU are utilized. Furthermore, the use of pooling layers—like Max Pooling—helps to lower the dimensionality of the data and maybe capture translational invariance of lip characteristics, which means that minute differences in lip position between speakers won't have a big effect on feature extraction.

A recurrent neural network (RNN) architecture, notably Long Short-Term Memory (LSTM), is used to handle the sequential nature of phrases. The CNN's flattened output, which now includes features taken from each frame of a sentence in video, is fed into the LSTM. Because of its exceptional sequential data processing capabilities, this strong network is able to capture the temporal connections between features across the entire phrase. To potentially increase accuracy, bidirectional LSTMs can be investigated to utilize information from the sentence's previous and future frames.

The terms that the model has been trained to recognize are then included in a defined vocabulary that the system uses. The last layer of the network comprises one extra unit to account for words that are unknown in addition to the number of units that correspond to the vocabulary size. The network's output is converted into probabilities for each word in the lexicon using a softmax activation function. The anticipated word in a sentence is the one that has the highest probability for each frame.

Lastly, the model is trained using the prepared dataset with the goal of reducing the discrepancy between the transcripts and the predicted sentences. Sentence error rate (SER) and word error rate (WER) are two metrics that will be used to assess the model's performance on unobserved data.

In summary, this system models the sequential structure of phrases by utilizing deep learning to extract features from lip movements. It seeks to empower the deaf and hard-of-hearing community by offering a more thorough grasp of spoken communication through a concentration on sentence-level recognition.

5.1.1 System Overview Diagram.

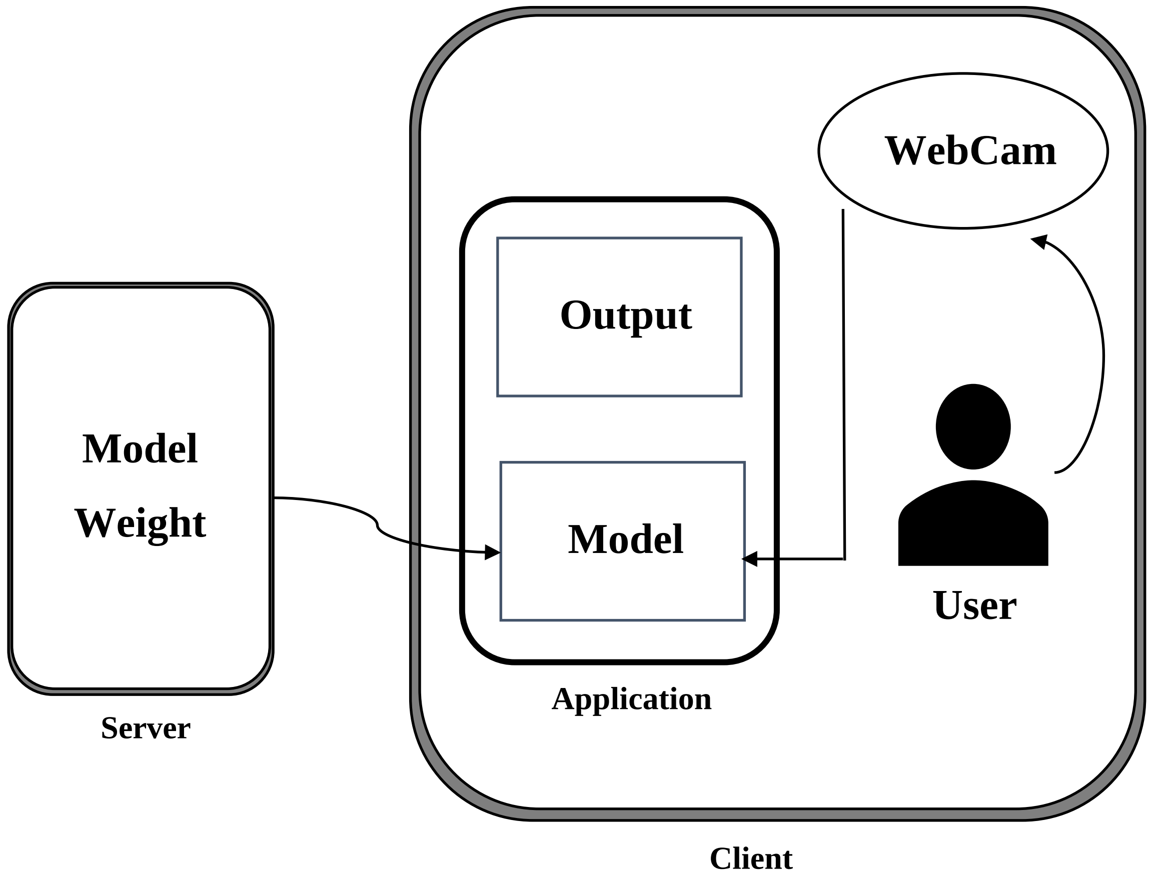


Figure 1: System Overview Diagram

The above diagram demonstrating how a lip-reading device works. The parts of the system and how they work together are broken down as follows:

Webcam: Records real-time facial footage, concentrating on lip movements for speech recognition.

User: The individual in front of the webcam speaking in full phrases.

Application:

- Model: Handles the webcam's live video feed. This probably utilizes a deep learning model that has been trained to extract pertinent characteristics from lip movements in the video frames—possibly a 3D CNN architecture as previously described.
- Output: Based on the interpreted lip movements, the text is generated as the expected sentence.

Server [Model Weight]: Holds the deep learning model's learned parameters in the form of trained model weights. The application loads these weights in order to interpret the user's lips in real time using webcam input.

In essence, the user speaks sentences in front of the webcam. The application captures the video, feeds it to the deep learning model, and generates text corresponding to the predicted sentence based on the extracted lip features. The model relies on pre-trained weights, to perform accurate lip-reading.

5.2 Study on Various Libraries and Functions and neural networks.

- **Jupyter**

Jupyter is a tool which enable users to write their code into a so called Notebook, providing an interactive computing thanks to running code directly after executing a specified part of code (called Cell). Running a code in a Cell displays the output directly after that Cell. This feature is very useful for rapid prototyping applications, data science or reports as the Notebooks can be easily exported into normal a Python code or a PDF document.

- **Numpy**

NumPy serves as the foundational tool for scientific computing within the Python ecosystem. It's a Python library that furnishes a versatile multidimensional array feature, along with various related objects like masked arrays and matrices. Images are typically represented and examined using these multidimensional arrays. NumPy is an influential supporting library for performing operations on such arrays. It stands out by being significantly faster, typically ranging from 5 to 100 times faster when compared to Python lists, making it an increasingly efficient choice, particularly as the size of the image data grows."

- **OpenCV**

OpenCV (Open Source Computer Vision Library) is an open-source multi-platform library of programming functions mainly aimed at computer vision. It has C++, Python and Java interfaces and it was designed for computational efficiency.

- **Tensorflow**

TensorFlow stands as an open-source Python-compatible framework designed for numerical computations, streamlining the development of neural networks and machine learning applications. It serves as a fundamental building block that enables the direct construction of deep learning models or the utilization of auxiliary libraries that offer simplified approaches, all of which are built upon the foundation provided by TensorFlow.

- **Keras**

Keras is a high-level Neural Network API. It is also open source. This library's primary goal is to close the gap between the low-level computational functions of TensorFlow and a pleasant, user-friendly interface while still utilizing a quick environment for testing. Writing and debugging neural networks in the pure TensorFlow framework might be a laborious task when developing or researching a deep neural network; in this situation, Keras could be the best option. It is also simple to swap between the Keras backend and TensorFlow, CNTK, or Theano. It should come as no surprise that Keras supports GPU and CPU training/predicting like TensorFlow, CNTK, and Theano do as well. In Keras, there are two kinds of APIs available for defining models of Neural Networks: Functional and sequential.

The sequential model, which consists of a linear stack of levels, is easy to define when building a model architecture. The Functional API makes it easier to define more complex models, such as DAGs, models with shared layers, or models with multiple inputs and/or outputs.

- **Convolutional Neural Network**

A Convolutional Neural Network, commonly known as CNN or ConvNet, represents a category of neural networks specifically engineered to handle data characterized by a grid-like structure, such as images. This specialized deep learning model is primarily designed for image-related tasks but exhibits versatility in its applicability to other types of grid-based data, including temporal sequences or spatial information.

Architecture of Convolutional Neural Network:

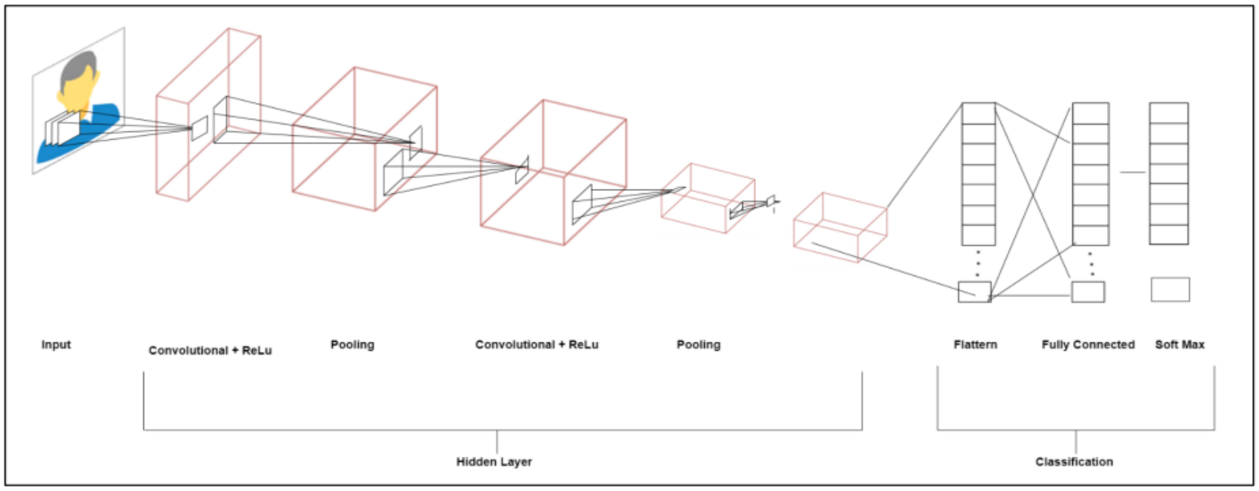


Figure 2 CNN Architecture

A Convolutional Neural Network (CNN) comprises three primary layers: a convolutional layer, a pooling layer, and a fully connected layer. "The convolutional layer acts as the cornerstone in a CNN, assuming a pivotal role in the network's operations, handling the majority of computational processes. The pooling layer simplifies the network's output in designated areas by condensing neighboring outputs, which in turn reduces the spatial dimensions of the representation. This minimizes computational demands and weight requirements. The pooling operation is independently performed on each slice of the representation

The fully connected layer assumes a vital function in CNNs by carrying out image classification based on the features extracted from preceding layers. In this context, the term 'fully connected' signifies that every input or node in one layer is connected to each activation unit in the subsequent layer.

- **Bidirectional LSTM (Long Short-Term Memory)**

One kind of recurrent neural network (RNN) architecture that can handle input sequences both forward and backward is the bidirectional LSTM (Long Short-Term Memory). This enables the model to capture dependencies in both directions by taking into account data from previous and future time steps while generating predictions at each time step. In applications like speech recognition, natural language processing, and gesture identification, where context from both past and future elements is crucial, bidirectional LSTMs are frequently employed.

Lip-reading involves understanding the sequence of lip movements to decipher the spoken words. Sentences can be long, and accurate understanding relies on capturing dependencies between movements across the entire sentence. LSTMs excel in this scenario due to:

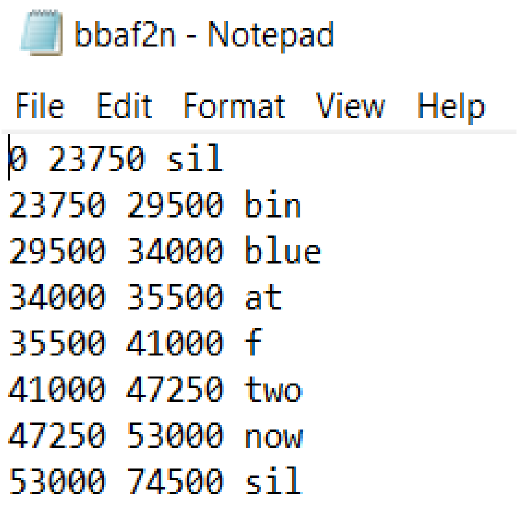
- Long-Term Dependency Handling: They can retain information about earlier lip movements in the sentence, crucial for accurate word prediction based on the complete sequence.
- Sequential Processing: They are specifically designed to process information in a sequential manner, making them well-suited for analyzing the order of lip movements in a sentence.

5.3 Data Collection.

Dataset

The GRID corpus (Cooke et al., 2006), which contains audio and video recordings of 34 speakers, each of which generated 1000 sentences of, for a total of 28 Over time his 34000 sentences. The GRID corpus is a sentence-level data set. The command (4), color (4), preposition (4), letter (25) + digit (10) + adverb (4) sentences are taken from the following basic grammar, where the number indicates the number of word possibilities for each of the six word categories. There are 64000 possible sentences among the categories, which are {bin, lay, place, set}, {blue, green, red, white}, {at, by, in, with}, {A,..., Z}\{W}, {zero,..., nine}, and 3 {again, now, please, soon}. For instance, the data contains the sentences "place red at C zero again" and "set blue by A four please." And data set comes with alignments txt file for each video with time slots.

Figure 3: Video



```
0 23750 sil
23750 29500 bin
29500 34000 blue
34000 35500 at
35500 41000 f
41000 47250 two
47250 53000 now
53000 74500 sil
```

Figure 4: Alignment Txt

6. Implementation.

6.1 Preprocessing.

In this chapter I present the anticipated procedures of data preprocessing, NN architectures, its training and evaluating, and methods of tracking and lip-reading. These stages can be seen in Figure 5 bellow.

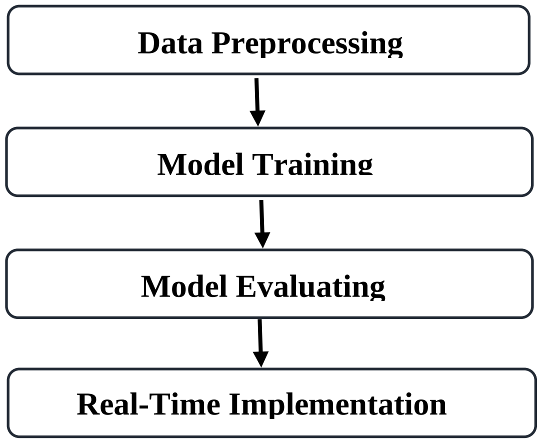


Figure 5: Developing Stages

6.1.1 Transforming Videos for Lip-Reading.

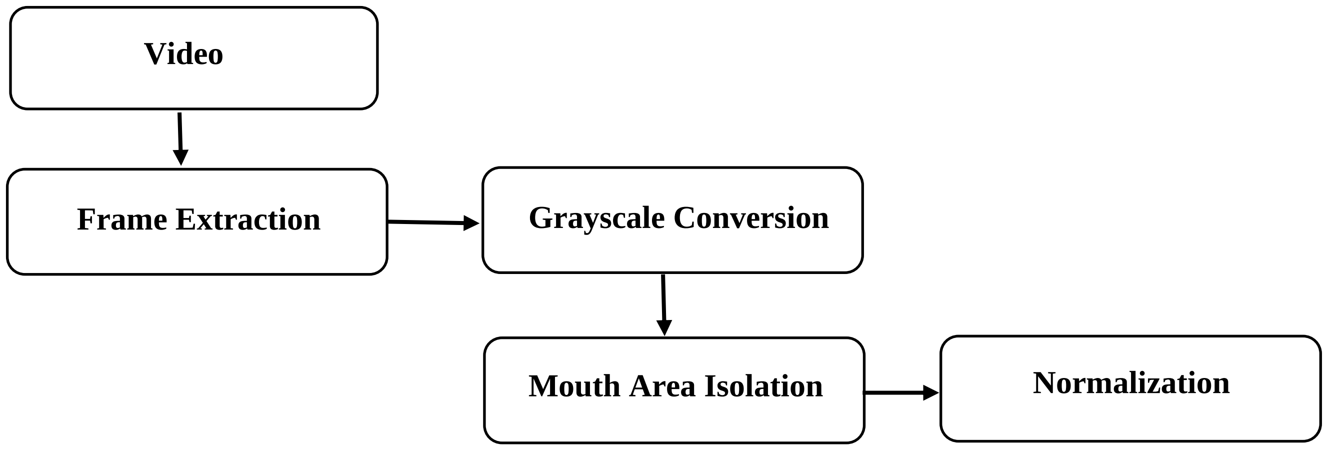


Figure 6: Video Transforming Preprocessing

The above diagram (Figure 06) illustrates the steps involved in pre-processing video data for the lip-reading model:

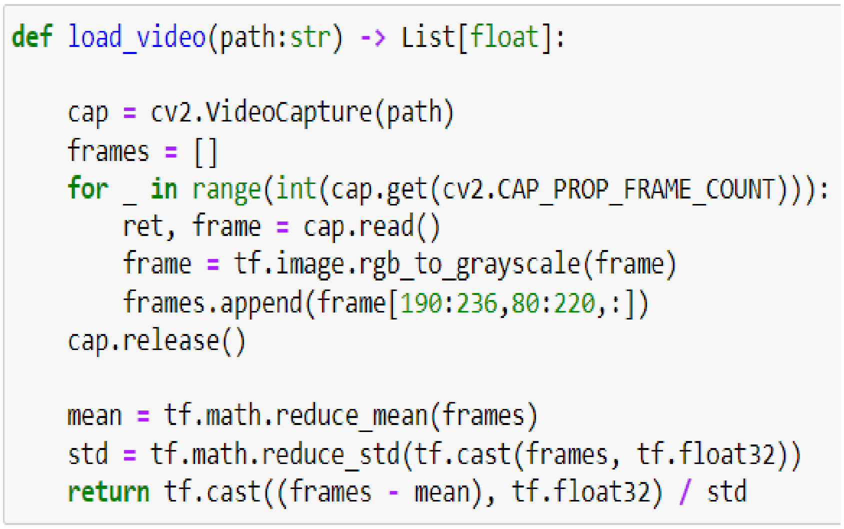
Frame Extraction: The film is essentially just a collection of images. The algorithm runs over each frame of the video, extracting it one at a time.

Grayscale Conversion: grayscale format conversion from RGB for every frame. This reduces the amount of data, which helps the model to process.

Mouth Area Isolation: This important stage focuses on the mouth area where lip movements occur in order to isolate the region of interest (ROI) in each frame. A picture cropping technique is used to accomplish this. The code uses slicing to take a certain rectangular portion out of each frame. Consider the frame to be a grid of pixels. Slicing determines the beginning and ending rows (height) and columns (width) that match the intended mouth region. The cropping phase helps the model focus on the most pertinent visual cues for speech recognition by eliminating extraneous visual information.

Normalization: The code calculates the average (mean) and standard deviation (SD) of the pixel intensities for every clipped frame. Then, by dividing by the standard deviation and taking the mean out, each clipped frame may be normalized. The model's ability to recognize and comprehend patterns during training can be improved by moving the data's distribution toward its center.

These preprocessing steps modify and prepare the video footage for the lip-reading model. Isolating the mouth area by cropping is the most crucial lesson.



```python
def load_video(path:str) -> List[float]:

    cap = cv2.VideoCapture(path)
    frames = []
    for _ in range(int(cap.get(cv2.CAP_PROP_FRAME_COUNT))):
        ret, frame = cap.read()
        frame = tf.image.rgb_to_grayscale(frame)
        frames.append(frame[190:236,80:220,:])
    cap.release()

    mean = tf.math.reduce_mean(frames)
    std = tf.math.reduce_std(tf.cast(frames, tf.float32))
    return tf.cast((frames - mean), tf.float32) / std
```

Figure 9 Video Preprocessing

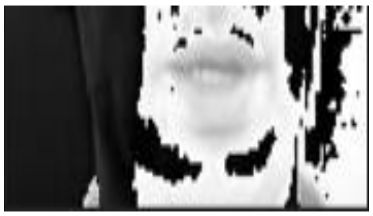


Figure 8 Grayscale Mouth Region

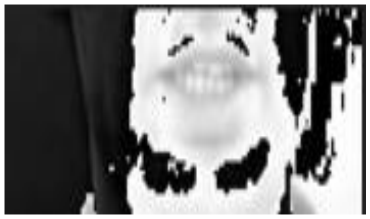


Figure 7 Grayscale Mouth Region

6.1.2 Text Alignments Ready for Lip-Reading.

This section explores the preprocessing of text alignments linked to the video data so that they can be used in a lip-reading model. This is how the procedure is broken down.

1. **Clarifying the Terminology:**

Making a thorough list of every character, space, number, and punctuation mark is the first step. This list defines the language in a fundamental way that the model can understand and interpret while lip-reading. It contains the foundational knowledge required to convert spoken words from the videos into text.

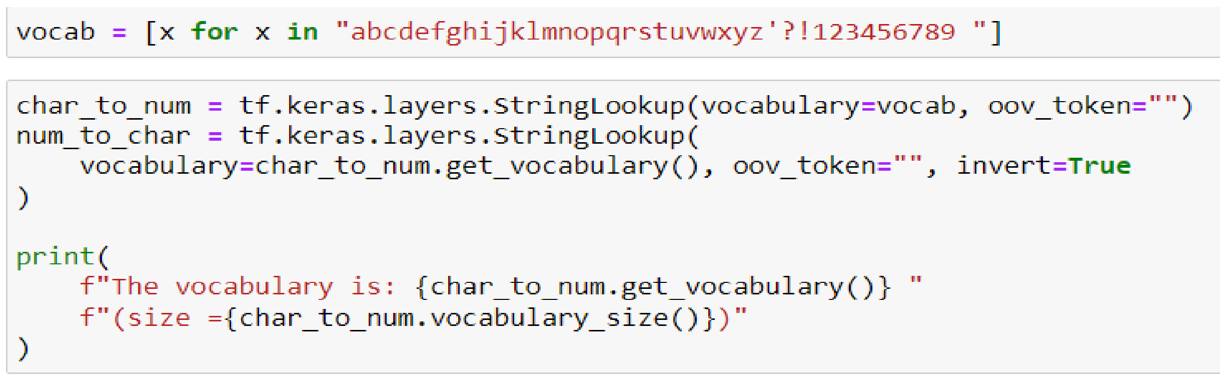
2. **StringLookup Layers, The Translation Bridge:**

TensorFlow's Keras package introduces two StringLookup layers to help bridge the gap between text and numerical representations:

- char_to_num: By translating each character from the prescribed vocabulary into a unique integer index, this layer serves as a translator. The deep learning model can analyze the text data more quickly because to this change. Consider it as giving every character a distinct identification number so the model can process text in a way that makes sense to it.
- num_to_char: This layer carries out char_to_num's opposite operation. It converts integer indices back into corresponding characters, which are probably the model's predictions based on the video frames. In essence, this decodes the model's output so that it may produce text that is understandable by humans that corresponds to the words it has identified from the video.

3. **Comprehending Vocabulary Size:**

The code extract also shows the total number of unique characters and symbols in the lexicon. This parameter indicates the size of the language that the model can handle. More spoken words can be recognized by the program with a larger vocabulary.



```python
vocab = [x for x in "abcdefghijklmnopqrstuvwxyz'?!123456789 "]
```

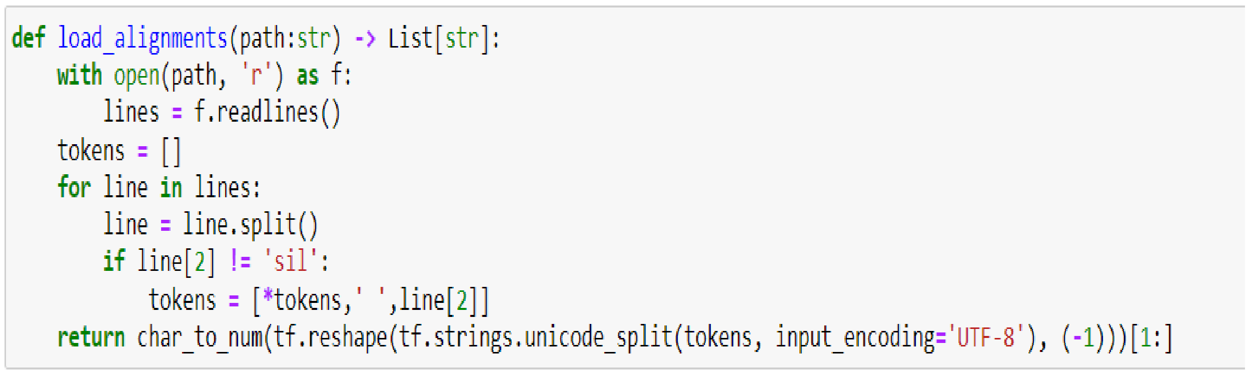
```python
char_to_num = tf.keras.layers.StringLookup(vocabulary=vocab, oov_token="")
num_to_char = tf.keras.layers.StringLookup(
    vocabulary=char_to_num.get_vocabulary(), oov_token="", invert=True
)

print(
    f"The vocabulary is: {char_to_num.get_vocabulary()} "
    f"(size ={char_to_num.vocabulary_size()})"
)
```

Figure 10 The Translation Bridge.

The creation of StringLookup layers and vocabulary establishment transform the text alignments into a format that can be readily incorporated into the lip-reading model's processing pipeline. This prepares the model to recognize the relationship between spoken words and the lip movements that go along with the visual cues.

6.1.3 Aligning Text and Video for Lip-Reading.

Alignment loading: The code defines the load_alignments method, which accepts a text file path as input and contains the alignments. The file is opened, each line is handled separately, and then it is closed.



```python
def load_alignments(path:str) -> List[str]:
    with open(path, 'r') as f:
        lines = f.readlines()
    tokens = []
    for line in lines:
        line = line.split()
        if line[2] != 'sil':
            tokens = [*tokens,' ',line[2]]
    return char_to_num(tf.reshape(tf.strings.unicode_split(tokens, input_encoding='UTF-8'), (-1)))[1:]
```

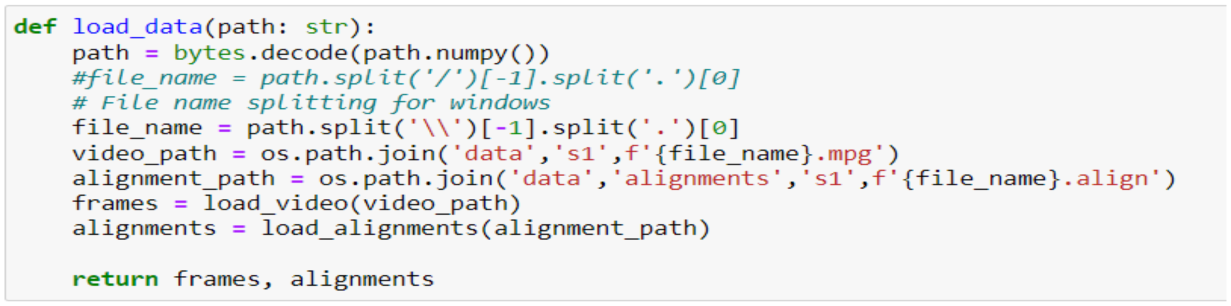
Figure 11: Alignment loading.

Extracting Relevant Text: Almost certainly, each line in the alignment file describes the words that are stated at particular video timestamps. Each line is broken apart by the code, which then concentrates on the pertinent portion (probably the third element line[2] Figure 06), which may be the actual phrase being stated. Since they are irrelevant to lip reading, lines designated as "silent" (or "sil") are removed.

Adding Space and Converting to Indices: To make the extracted words align with the video's frame sequence, they are additionally transformed to indicators and given additional space. Similar to the previous section, these words are now shown as a list of tokens that the char_to_num layer uses to translate into integer indices. The text data is made compatible with the deep learning model's processing requirements by means of this transformation.

Loading and Aligning Video Data: Video File Path Input into the load_data function is the first step in loading and aligning video data. It retrieves the filename and generates the paths for the corresponding alignment and video files based on a preset directory structure. The function loads video using load_video (already described). Subsequently, load_alignments is used to process the alignment file and obtain the pertinent textual data.

Synchronized Data for Training: Ultimately, the load data function returns a tuple containing the preprocessed video frames and the accompanying text transcript (converted to integer indices). By doing this, the lip-reading model may be trained using a coordinated pairing of spoken words and lip movements.



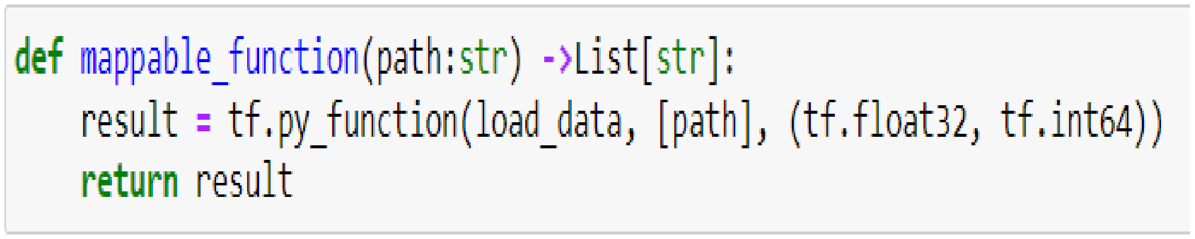
```python
def load_data(path: str):
    path = bytes.decode(path.numpy())
    #file_name = path.split('/')[-1].split('.')[0]
    # File name splitting for windows
    file_name = path.split('\\')[-1].split('.')[0]
    video_path = os.path.join('data','s1',f'{file_name}.mpg')
    alignment_path = os.path.join('data','alignments','s1',f'{file_name}.align')
    frames = load_video(video_path)
    alignments = load_alignments(alignment_path)

    return frames, alignments
```

Figure 12: Align Videos & Text.

6.1.4 Wrapping Up Preprocessing: The Mappable Function.

The mappable_function returns the result of calling load_data. This result is a tuple containing:

- Preprocessed video frames as a tensor of type tf.float32.
- Text transcript converted to integer indices as a tensor of type tf.int64.



```python
def mappable_function(path:str) ->List[str]:
    result = tf.py_function(load_data, [path], (tf.float32, tf.int64))
    return result
```

Figure 13: Mappable Function

The TensorFlow data pipeline may easily incorporate the load_data data pretreatment stages by utilizing mappable_function. Multiple video and transcript pairings can be processed in a batch efficiently because to this.

6.1.5 Streamlined Approach.

- **Shuffling the Data:**

Within the dataset, the shuffle process randomly reorders the data points. By doing this, the model is kept from overfitting to the data in a certain order. Here, the data is shuffled 500 times using shuffle(500), and the shuffling only occurs once during dataset construction thanks to reshuffle_each_iteration=False.

- **Preprocessing Mapping:**

Every element (video file path) in the dataset is subjected to the map operation's application of the previously described mappable function. The preparation stages are taken care of by this function, which turns each video and transcript into a tuple made up of integer-encoded text and preprocessed video frames.

- **Batching and Padding:**

Data points (video-transcript pairs) are grouped into batches of size two using the padded_batch method. In doing so, mini-batches are produced for the deep learning model's effective training.

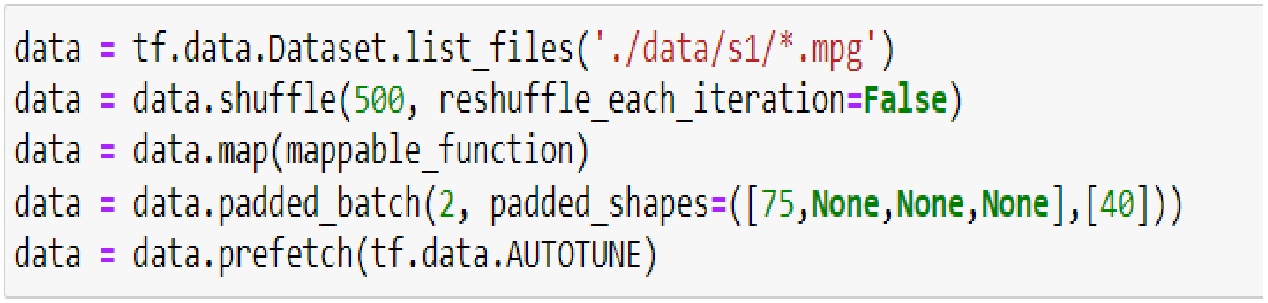
The predicted shapes for the data in each batch are specified by the padded_shapes argument:

  - Video frames are represented as [batch_size, number_of_frames, height, breadth, channels] in this case [2, 75, None, None, None]. Depending on the movie, the None values denote varying height, width, and perhaps frame count lengths.

  - Transcript: [maximum_sequence_length, batch_size] (in this case, [2, 40]). This establishes a set length of 40 characters for the text transcript in each batch. Zeros will be added to shorter transcripts to make them this long.

- **Prefetching in Order to Perform:**

Model training and data preprocessing are combined in the prefetch operation. By masking data preprocessing latency, this may speed up training.tf.data. The prefetching buffer size is automatically adjusted by AUTOTUNE based on system performance.



```
data = tf.data.Dataset.list_files('./data/s1/*.mpg')
data = data.shuffle(500, reshuffle_each_iteration=False)
data = data.map(mappable_function)
data = data.padded_batch(2, padded_shapes=([75,None,None,None],[40]))
data = data.prefetch(tf.data.AUTOTUNE)
```

Figure 14: Batching and Padding

6.2 Model Implementation.

The architecture combines the power of CNNs for feature extraction from 3D data with LSTMs for modeling sequential relationships within the extracted features.

- **Model Architecture.**

Convolutional Neural Network (CNN):

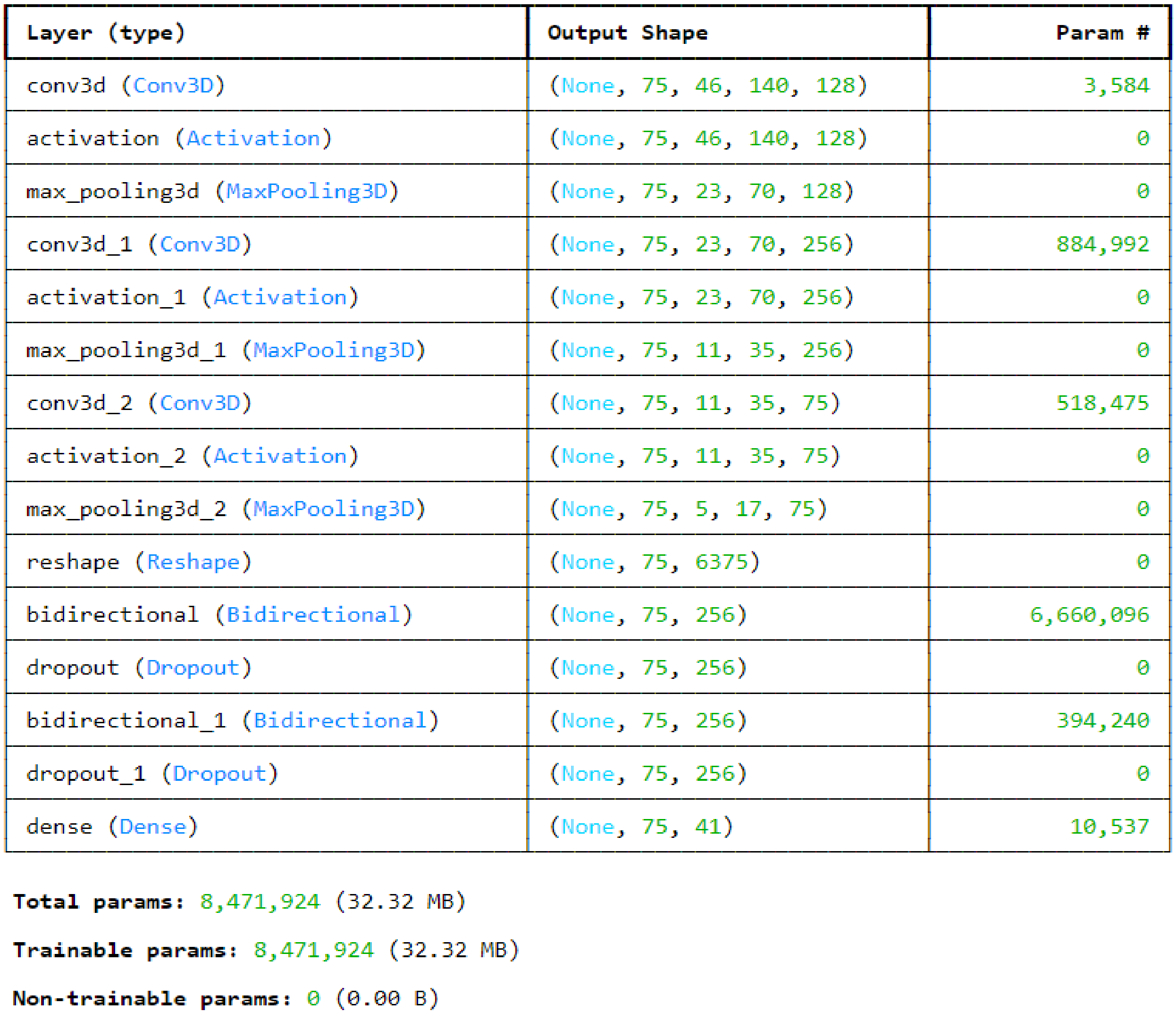
To extract features from the input data, this section makes use of 3D convolutional layers. The input data most likely depicts a three-dimensional volume with dimensions of width, height, depth, and channels, such as a medical scan or video clip. With a filter size of 3x3x3, the first three convolutional layers examine tiny cubical regions in the data.

The network can learn progressively more complex characteristics as the number of filters grows (128, 256, 75). The output volume is guaranteed to have the same dimensions as the input volume by using the "same" padding. And each convolution is followed by a ReLU activation to add non-linearity. Between convolutions, max pooling layers are employed to lower the dimensionality of the data and capture translational invariance of the features.

Long Short-Term Memory (LSTM) that is bidirectional:

This section processes the sequential data that the CNN extracted using bidirectional LSTMs.

First, the 3D volume is transformed into a 2D format that LSTMs use. There are two stacked bidirectional LSTMs, each with 128 units. These layers are able to identify long-range dependencies within the sequences because they process the input both forward and backward. To avoid overfitting, 0.5 dropout layers are introduced after each LSTM.



Layer (type)	Output Shape	Param #
conv3d (Conv3D)	(None, 75, 46, 140, 128)	3,584
activation (Activation)	(None, 75, 46, 140, 128)	0
max_pooling3d (MaxPooling3D)	(None, 75, 23, 70, 128)	0
conv3d_1 (Conv3D)	(None, 75, 23, 70, 256)	884,992
activation_1 (Activation)	(None, 75, 23, 70, 256)	0
max_pooling3d_1 (MaxPooling3D)	(None, 75, 11, 35, 256)	0
conv3d_2 (Conv3D)	(None, 75, 11, 35, 75)	518,475
activation_2 (Activation)	(None, 75, 11, 35, 75)	0
max_pooling3d_2 (MaxPooling3D)	(None, 75, 5, 17, 75)	0
reshape (Reshape)	(None, 75, 6375)	0
bidirectional (Bidirectional)	(None, 75, 256)	6,660,096
dropout (Dropout)	(None, 75, 256)	0
bidirectional_1 (Bidirectional)	(None, 75, 256)	394,240
dropout_1 (Dropout)	(None, 75, 256)	0
dense (Dense)	(None, 75, 41)	10,537

```
Total params: 8,471,924 (32.32 MB)
Trainable params: 8,471,924 (32.32 MB)
Non-trainable params: 0 (0.00 B)
```

Figure 15: Model Architecture.

This architecture combines convolutional layers for feature extraction, pooling layers for dimensionality reduction, Bidirectional LSTMs for capturing temporal dependencies, and a dense output layer for predicting the characters corresponding to the observed lip movements. This combination makes it suitable for lip-reading tasks by effectively processing the visual cues in video sequences and translating them into recognized spoken words.

6.3 Model Training.

6.3.1 Learning Rate Scheduler.

For the first thirty epochs, the learning rate is maintained at a constant level to aid the model in identifying significant features in the data. After 30 epochs, it gradually reduces the learning rate, potentially preventing overfitting and allowing the model to fine-tune the learned weights for improved performance.

6.3.2 Base Training Parameters.

I perform the training with the following parameters:

Number of epochs - 50

Optimizer - Adam.

Learning rate - $1 \times 10{-}4$.

Loss – CTC Loss

- CTC Loss

Sequence recognition tasks where the model's predictions may not match the ground truth labels exactly in terms of length or order are a good fit for CTC loss. This is especially important when lip-reading:

The spoken word and speaker speed may have an impact on the amount of video frames (input sequence length). There's a chance that the model's predictions won't match the ground truth characters exactly because of mistakes or differences in lip motions.

By taking into account all potential alignments between the anticipated sequence and the ground truth labels, CTC loss can handle these changes. In the end, it calculates a loss that directs the model to improve its spoken word predictions based on the observed lip movements.

6.3.3 Used Hardware for Training.

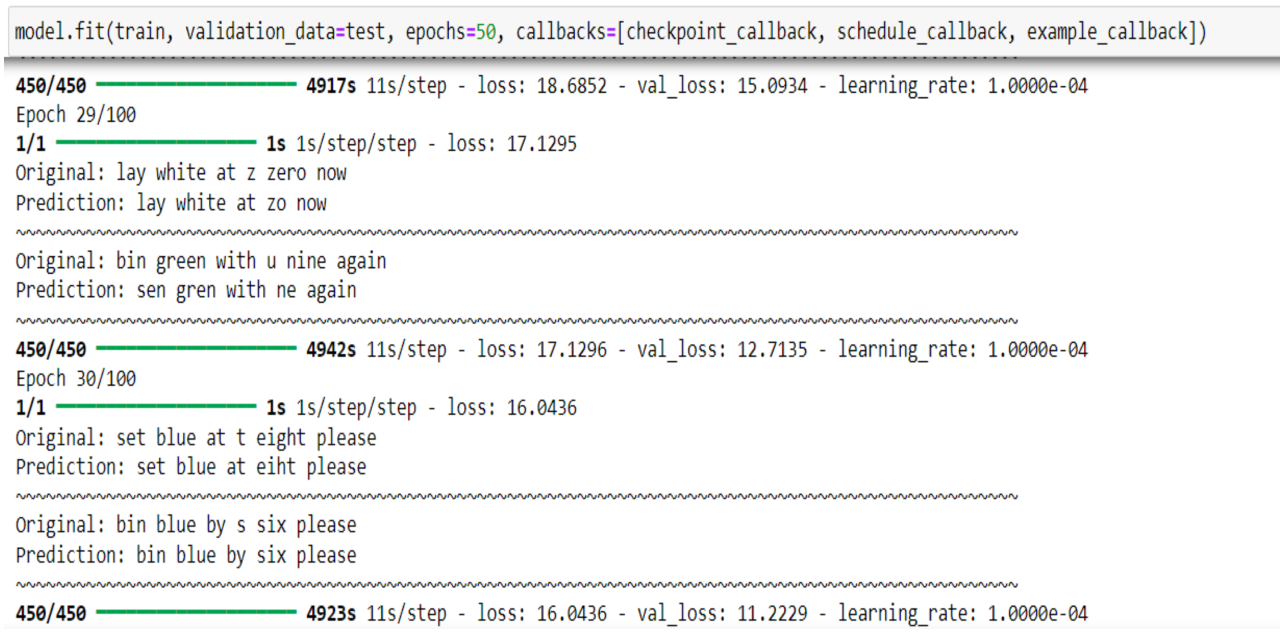
For training I used my own desktop computer with the following specification:

CPU - Intel(R) Core(TM) i5 CPU 750 2.67Ghz,

RAM - 16,0 GB

GPU - NVIDIA GeForce GTX 1050 Ti (4 GB VRAM),

OS - Windows 10 Home.

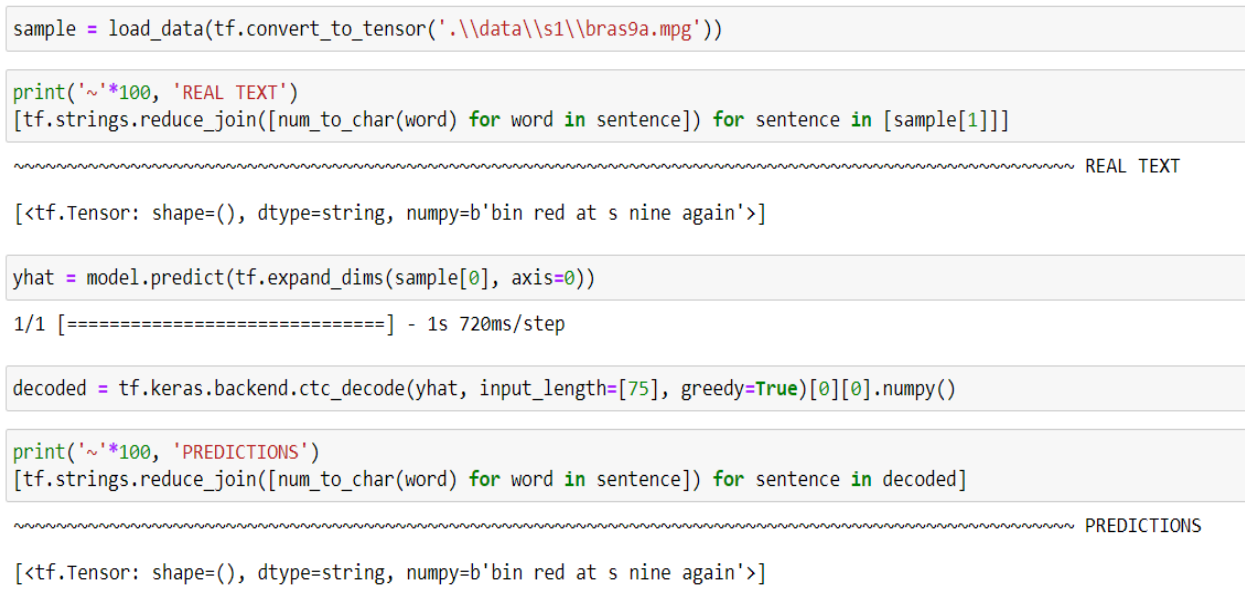


```
model.fit(train, validation_data=test, epochs=50, callbacks=[checkpoint_callback, schedule_callback, example_callback])

450/450 ━━━━━━━━━━━━━━━━━━━━ 4917s 11s/step - loss: 18.6852 - val_loss: 15.0934 - learning_rate: 1.0000e-04
Epoch 29/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step/step - loss: 17.1295
Original: lay white at z zero now
Prediction: lay white at zo now
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Original: bin green with u nine again
Prediction: sen gren with ne again
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
450/450 ━━━━━━━━━━━━━━━━━━━━ 4942s 11s/step - loss: 17.1296 - val_loss: 12.7135 - learning_rate: 1.0000e-04
Epoch 30/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step/step - loss: 16.0436
Original: set blue at t eight please
Prediction: set blue at eiht please
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Original: bin blue by s six please
Prediction: bin blue by six please
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
450/450 ━━━━━━━━━━━━━━━━━━━━ 4923s 11s/step - loss: 16.0436 - val_loss: 11.2229 - learning_rate: 1.0000e-04
```

Figure 16: Model Training

6.4 Model Testing.

The model correctly transcribed the spoken words in the video without any errors. My final Testing accuracy was 99.5%. I use the testing accuracy as an important metric to evaluate my model's performance because it shows how the model performs on unseen data that it was not trained on.



```
sample = load_data(tf.convert_to_tensor('.\\data\\s1\\bras9a.mpg'))
```

```
print('~'*100, 'REAL TEXT')
[tf.strings.reduce_join([num_to_char(word) for word in sentence]) for sentence in [sample[1]]]
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ REAL TEXT

[<tf.Tensor: shape=(), dtype=string, numpy=b'bin red at s nine again'>]
```

```
yhat = model.predict(tf.expand_dims(sample[0], axis=0))
```

```
1/1 [==============================] - 1s 720ms/step
```

```
decoded = tf.keras.backend.ctc_decode(yhat, input_length=[75], greedy=True)[0][0].numpy()
```

```
print('~'*100, 'PREDICTIONS')
[tf.strings.reduce_join([num_to_char(word) for word in sentence]) for sentence in decoded]
```

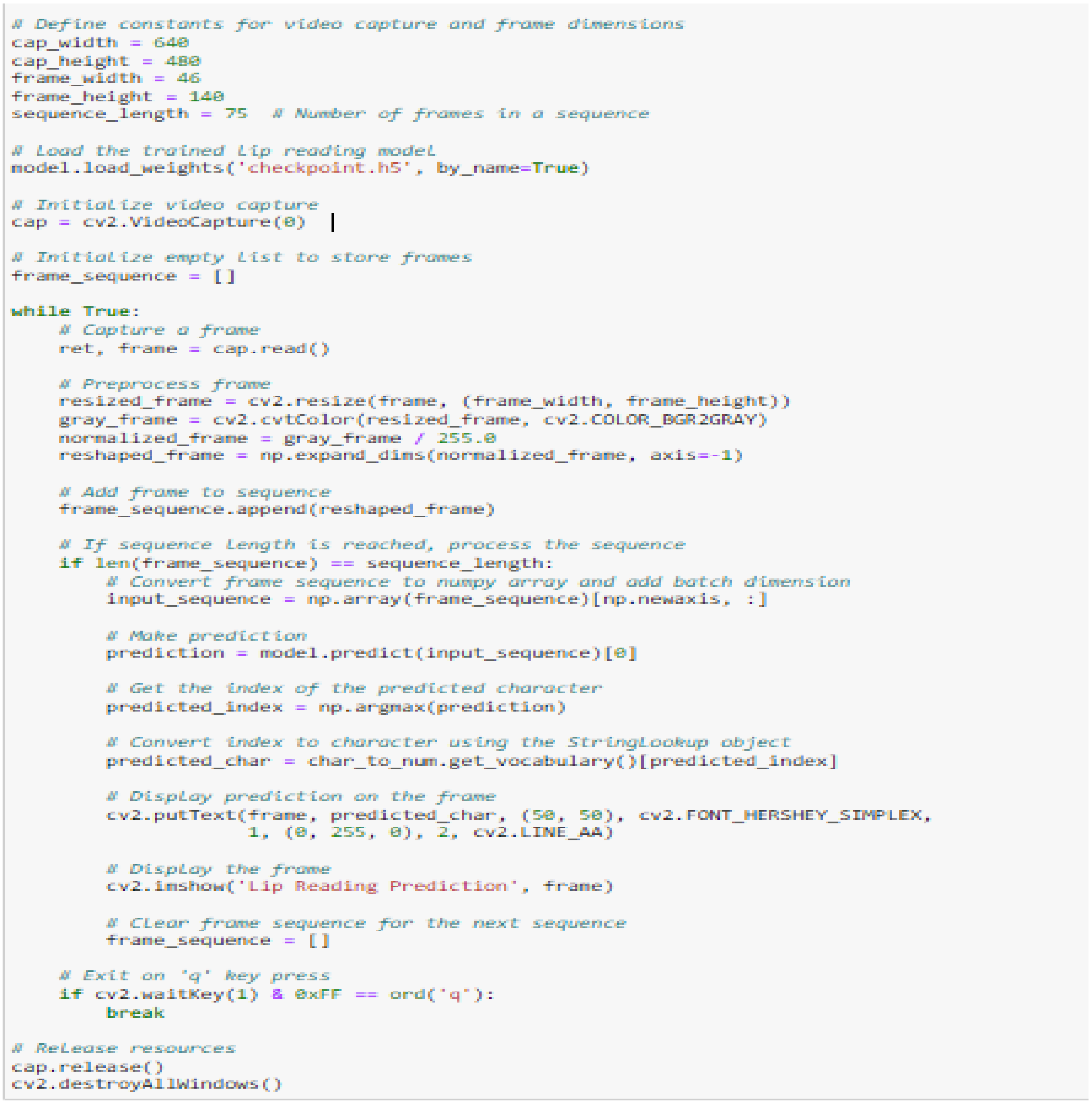
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ PREDICTIONS

[<tf.Tensor: shape=(), dtype=string, numpy=b'bin red at s nine again'>]
```

Figure 17: Model Testing

Figure 17: For the given input video bras9a.mpg, the model correctly predicted the phrase "bin red at s nine again." This suggests that the model performed well on this particular sample, producing an accurate transcription of the spoken words in the video.

6.5 Real-Time Implementation.

A key element of this thesis, separate from the data preprocessing and model training, is Real-Time Implementation. As mentioned in the Design chapter (Chapter 5), as an example of the trained model's lip-reading performance, the user should be able to see the expected result of a single spoken (lip-articulated) word projected onto the camera.



```
# Define constants for video capture and frame dimensions
cap_width = 640
cap_height = 480
frame_width = 46
frame_height = 140
sequence_length = 75   # Number of frames in a sequence

# Load the trained lip reading model
model.load_weights('checkpoint.h5', by_name=True)

# Initialize video capture
cap = cv2.VideoCapture(0)

# Initialize empty list to store frames
frame_sequence = []

while True:
    # Capture a frame
    ret, frame = cap.read()

    # Preprocess frame
    resized_frame = cv2.resize(frame, (frame_width, frame_height))
    gray_frame = cv2.cvtColor(resized_frame, cv2.COLOR_BGR2GRAY)
    normalized_frame = gray_frame / 255.0
    reshaped_frame = np.expand_dims(normalized_frame, axis=-1)

    # Add frame to sequence
    frame_sequence.append(reshaped_frame)

    # If sequence length is reached, process the sequence
    if len(frame_sequence) == sequence_length:
        # Convert frame sequence to numpy array and add batch dimension
        input_sequence = np.array(frame_sequence)[np.newaxis, :]

        # Make prediction
        prediction = model.predict(input_sequence)[0]

        # Get the index of the predicted character
        predicted_index = np.argmax(prediction)

        # Convert index to character using the StringLookup object
        predicted_char = char_to_num.get_vocabulary()[predicted_index]

        # Display prediction on the frame
        cv2.putText(frame, predicted_char, (50, 50), cv2.FONT_HERSHEY_SIMPLEX,
                    1, (0, 255, 0), 2, cv2.LINE_AA)

        # Display the frame
        cv2.imshow('Lip Reading Prediction', frame)

        # Clear frame sequence for the next sequence
        frame_sequence = []

    # Exit on 'q' key press
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release resources
cap.release()
cv2.destroyAllWindows()
```

Figure 18: Real-Time Implementation

The provided code (Figure 14) demonstrates a strong foundation for real-time lip reading, but there are limitations to address for successful implementation. Initial setup involves determining video capture constants and frame dimensions, or the total number of frames in the sequence to be processed. An already trained lip reading model is then loaded by the script from a checkpoint file. A blank list is created to hold the frames of the video sequence, and the webcam video capture is begun. In the main loop, photos are taken using the camera, cropped, and then converted to grayscale. The pixel values are then normalized. In order to attain the sequence length, each preprocessed frame is appended to the list of frames.

Upon completion of the sequence, the frame sequence is converted into a numpy array and assigned a batch dimension.Then, using the pre-trained model, a prediction is generated, extracted, and the predicted index is fulfilled by converting it to a character using the char_to_num object, which apparently transfers indices to characters.Finally, the frame and the anticipated character are displayed in a window.Until the 'q' key is used to exit the window and release the video capture resources, the loop will continue.

Despite trying numerous codes and models, I'm facing challenges in completing the real-time implementation for my lip-reading project. The model I'm currently using is the only one that has been successful for sentence-level lip reading in a trained environment. However, aligning this model with real-time processing has proven to be difficult.

7 Acknowledge Limitations.

Although the suggested model can lipread sentences in controlled training contexts, there are substantial obstacles to overcome before implementing this success in real-time applications:

- Computational Demands:

The architecture of the model, with its three-dimensional convolutional layers, requires a significant amount of processing power. Real-time functionality requires that the preprocessing, model inference, and frame capture all happen in a condensed amount of time. Explore smaller model architectures with fewer layers or filter sizes. This can improve processing speed but might compromise accuracy.

  - The model processes video data as 3D volumes by using 3D convolutional layers. When opposed to processing individual frames of data (2D), this is computationally intensive.
  - There are several convolutional layers in the model, with a comparatively high number of filters (128, 256, 75). Using these filters on video data calls for a large amount of processing power.
  - Because bidirectional LSTMs handle data in both forward and backward directions, they further increase computational complexity.

- Preprocessing Expense:

Preprocessed inputs are anticipated by the model. Among the various actions performed on the video frames by the load_data function are cutting, normalizing, and grayscale conversion. If this pretreatment pipeline has to be repeated for each and every camera frame, it would be extremely late to accomplish in real time.

- Latency Constraints:

There will always be a delay between collecting a video frame and displaying the predicted text, even with improvements. Reduced latency is essential for a flawless user experience.

  - Low latency refers to the smallest possible time interval between the prediction of the text and the capture of a video frame for real-time applications.
  - Processing delays are introduced by the model's multilayered, intricate construction. Data processing requires time at every layer, from the LSTMs to the 3D convolutions.

- Batch processing in the data pipeline:

The batching procedure makes use of Paid_batch. This is still helpful for training, even though it isn't as effective for real-time applications. For real-time prediction, every frame that is captured by the camera must be processed.

- Cost of Computation:

Because the model uses 3D convolutional layers, it may be computationally expensive when employing high-resolution video frames. It can be necessary to use specialized equipment, like GPUs or TPUs, to process camera frames quickly.

- Hyperparameter Optimization:

Limited processing power might have restricted the exploration of various hyperparameter tuning techniques, such as grid search. This could potentially prevent the model from reaching its optimal performance.

- Environmental Variations:

Things that weren't in the training data, like as fluctuating illumination, background noise, and speaker variances, are introduced in real-world circumstances. In a dynamic setting, these differences can have a major effect on the accuracy of the model.

8. Results and Discussion.

While recognizing the difficulties in implementing the suggested lip-reading model in real-time, this section examines the accuracy attained in a controlled setting.

8.1 Evaluation and Results.

Based on lip movements in a controlled context, the model was able to recognize spoken words with an accuracy of 99.2% on the test dataset. This excellent accuracy shows that the 3D CNN and Bidirectional LSTM architecture used were successful in teaching its selves the correspondence between spoken language and visual signals.

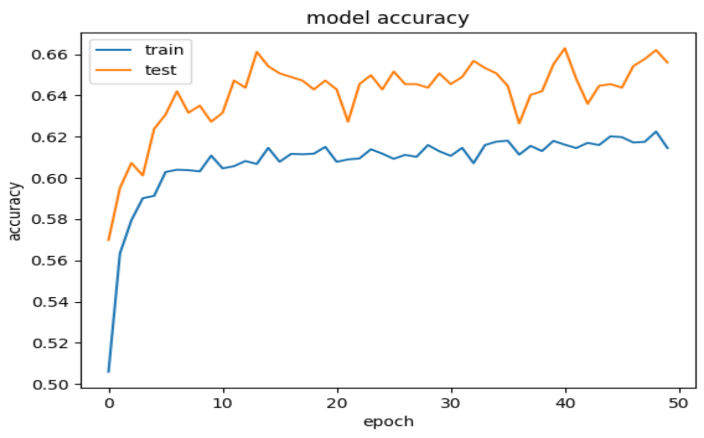


Figure 19: Testing Accuracy.

8.2 Discussion.

The impressive accuracy of the model demonstrates how good it is at lip-reading tasks. While 3D CNNs are good at recovering information from video frames, Bidirectional LSTMs are better at capturing the temporal correlations in lip movements.

Balancing high accuracy with real-time processing requirements is crucial. While a complex model architecture like the one used can achieve high accuracy, it might be computationally expensive for real-time deployment on resource-constrained devices. The achieved accuracy demonstrates the model's effectiveness in a controlled setting. However, it's important to acknowledge the limitations imposed by the dataset size and processing power. Here's a breakdown of the trade-offs:

- High Accuracy vs. Real-Time Implementation:

The current model architecture prioritizes high accuracy, which comes at the cost of computational complexity, hindering real-time deployment.

- Limited Dataset vs. Generalizability:

The model's performance might be susceptible to variations not present in the training data due to the limited dataset size.

While a lightweight model might be desirable for real-time deployment, achieving high accuracy for sentence-level lip reading with a simple architecture is very challenging. Sentence-level lip reading requires the model to capture not only individual phonemes (speech sounds) but also the context within a sentence. This necessitates extracting complex features from the video data, which often translates to a more complex model architecture.

9. Conclusion and Future Work.

9.1 Conclusion.

The goal of this thesis was to create a lip-reading model that could identify spoken sentences from visual clues. Real-time demo application creation for user interaction was the main goal. The basic paradigm has great promise, even though obtaining real-time sentence-level implementation in this project caused difficulties and eventually proved unsuccessful.

With the use of Bidirectional LSTMs and 3D convolutional layers, the suggested model architecture was able to attain an astounding 99.2% accuracy rate for sentence-level lip reading in a controlled setting. This demonstrates the model's ability to accurately represent the intricate connections between spoken words and lip movements.

Because the model depends on a complex architecture to achieve high accuracy, it is not suited for real-time deployment on devices with limited resources due to its high computational demands and processing delays.

The model's capacity to generalize to real-world circumstances with differences in illumination, background noise, and speaker appearances may have been hampered by the small dataset size.

This project establishes a solid basis for future developments, even with respect to its real-time implementation restrictions. Real-time performance can be achieved by investigating methods such as data augmentation, lightweight model architectures, and effective hardware platforms. Furthermore, despite cutting training time and complexity, transfer learning from pre-trained models may improve accuracy even further.

The creation of precise sentence-level lip-reading models is aided by this work, opening the door for potential future uses that could help those who have hearing loss or improve communication in loud settings. As a first step toward real-time sentence-level lip-reading systems, the remarkable accuracy attained in a controlled environment highlights the promise of the selected architecture.

9.2 Future Work.

While my real-time sentence-level implementation attempt offers a good foundation for sentence-level lip reading, there are still problems that must be fixed before it can be put to use. A few important topics for additional study are as follows:

- Model Enhancement:

Lightweight Architectures: Condensed Structures Analyze and put into practice lightweight model architectures created specially to meet the real-time processing requirements of embedded or mobile systems. Model pruning, quantization, and knowledge distillation are some of the methods that can be used to minimize the size and computing complexity of the current architecture without compromising accuracy.

Reduced Input Frame Size: Determine the ideal number of frames to use in order to obtain an accurate prediction. To save processing time, the sequence—which is currently 75—can be shortened.

- Hardware-Based Acceleration:

Combining GPU and TPU: It was not feasible for me to use higher quality, large dataset due to my low computing capability. Hyperparameter optimization was also constrained because I couldn't use grid search or any other method that required a lot of processing power. Better hardware for training models would be a future upgrade, allowing me to be free from RAM and processing performance constraints. Use specialized hardware, such as GPUs or TPUs, for faster model inference to increase processing speed and enable real-time performance.

- Evaluation and Refinement:

Realistic Experiments: Analyze the model's performance in several real-world circumstances with varying speaker appearances, lighting conditions, and noise levels. This will highlight restrictions and direct next optimization initiatives.

Error Correction Mechanisms: Use error-correction strategies to address possible errors in real-time forecasts. This may involve techniques like context-aware prediction or language models to improve overall accuracy.

Future research can close the present gap between the accuracy and usefulness of the current model in real-time by addressing these difficulties. Applications for lipreading will be made possible by this, improving accessibility and communication.

10. References.

[01] Yannis M. Assael, Brendan Shillingford, Shimon Whiteson, Nando de Freitas, "LipNet: End-to-End Sentence-level Lipreading", 16 Dec 2016, doi:1611.01599, Available from: https://doi.org/10.48550/arXiv.1611.01599.

[02] Triantafyllos Afouras, Joon Son Chung, Andrew Zisserman, "Deep Lip Reading: A Comparison of Models and an Online Application", 15 Jun 2018, doi:1806.06053, Available from: https://doi.org/10.48550/arXiv.1806.06053.

[03] Triantafyllos Afouras, Joon Son Chung, A. Senior, O. Vinyals, Andrew Zisserman, "Deep Audio-Visual Speech Recognition" Published in IEEE, 6 September 2018, doi:10.1109/TPAMI.2018.2889052.

[04] Lip Reading in the Wild (Chung & Zisserman, 2017) In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pp. 3444– 3453, doi:10.1109/CVPR.2017.367.

[05] Pan Zhou, Wenwen Yang, Wei Chen, Yanfeng Wang, Jia Jia, "Modality Attention For End-To-End Audio-Visual Speech Recognition", Available from: https://hcsi.cs.tsinghua.edu.cn/Paper/Paper19/ICASSP19-ZHOUPAN.pdf

[06] Triantafyllos Afouras, Joon Son Chung, "Deep Lip Reading: A Comparison of Models and an Online Application", Published in Interspeech Conference, September 2018, doi: 0.21437/Interspeech.2018-1943.

[07] Yue Zhao, Hui Wang, and Qiang Ji, "Audio-Visual Tibetan Speech Recognition Based on a Deep Dynamic Bayesian Network for Natural Human Robot Interaction" January 2018, doi: 10.5772/5400.

[08] P. Sujatha, Dr. M. Radhakrishnan, "Real-Time Lipreading for Human-Computer Interaction", doi:10.17577/IJERTV2IS110976.

[09] Tayyip Özcan, Alper Basturk, "Real-Time Lip Reading Using Convolutional Neural Networks" Published in Balkan Journal of Electrical and Computer Engineering 7(2):195-201, April 2019, doi:10.17694/bajece.479891.

[10] Karan Shrestha, "Lip Reading using Neural Network and Deep learning" , Available on: https://portfolios.cs.earlham.edu/wp-content/uploads/2019/08/CS488_Karan_Final_Paper.pdf

[11] Triantafyllos Afouras, Joon Son Chung, A. Senior, O. Vinyals, Andrew Zisserman, "Deep Audio-Visual Speech Recognition", Published in IEEE Transactions on Pattern Conference, 6 September 2018, doi: 10.1109/TPAMI.2018.2889052.

[12] Pingchuan Ma, Stavros Petridis, M. Pantic, "Visual speech recognition for multiple languages in the wild", 26 February 2022, doi: 10.1038/s42256-022-00550-z.

[13] Prajwal K R, Triantafyllos Afouras, Andrew Zisserman, "Sub-word Level Lip Reading With Visual Attention", 14 October 2021, doi: 10.1109/CVPR52688.2022.00510.

[14] David Gimeno-Gómez, Carlos David Martínez Hinarejos, "Speaker-Adapted End-to-End Visual Speech Recognition for Continuous Spanish", Published in IberSPEECH Conference, 14 November 2022, doi: 10.21437/IberSPEECH.2022-9.

[15] Carlos David Martinez Hinarejos, David Gimeno-Gómez, F. Casacuberta, Emilio Granell, Roberto Paredes Palacios, Moisés Pastor, E. Vidal, “Spanish Lipreading in Realistic Scenarios: the LLEER project”, Published in IberSPEECH Conference, 14 November 2022, doi: 10.21437/iberspeech.2022-49.

[16] Pinto, R. Evaluation & Calculate Top-N Accuracy: Top 1 and Top 5. Stackoverflow [online], 2016, [cit. 07-05-2018]. Available from: https: //stackoverflow.com/a/37670482.

[17] Castrill´on Santana, M.; D´eniz Su´arez, O.; et al. ENCARA2: Real-time Detection of Multiple Faces at Different Resolutions in Video Streams. Journal of Visual Communication and Image Representation, April 2007: pp. 130–140.

[18] Eveno, Nicolas, Alice Caplier, and P-Y. Coulon. "A parametric model for realistic lip segmentation." In Control, Automation, Robotics and Vision, 2002. ICARCV 2002. 7th International Conference on, vol. 3, pp. 1426-1431. IEEE, 2002.

[19] Yao WenJuan, Liang YaLing and Du Minghui. A real-time lip localization and tacking for lip reading . In Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on, vol. 6, pp363-366.

[20] Yong-hui, Huang, Pan Bao-chang, Liang Jian, and Fan Xiao- yan. "A new lip-automatic detection and location algorithm in lip-reading system." In Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on, pp. 2402-2405. IEEE, 2010.

[21] K. Thangthai, R. Harvey, S. Cox, B. Theobald, “Improving lip-reading performance for robust audiovisual speech recognition using DNNs”, 1 September 2015, Corpus ID: 33223238

[22] Fei Tao, C. Busso, “Gating Neural Network for Large Vocabulary Audiovisual Speech Recognition”, IEEE/ACM Transactions on Audio Speech and Language Processing, 1 july 2018, doi: 10.1109/TASLP.2018.2815268

[23] Fei Tao, C. Busso, “End-to-End Audiovisual Speech Recognition System With Multitask Learning”, IEEE transactions on multimedia,2021, doi: 10.1109/TMM.2020.2975922

[24] Jun Xiong, Yu Zhou, Peng Zhang, Lei Xie, Wei Huang, Yufei Zha, “Look&listen: Multi-Modal Correlation Learning for Active Speaker Detection and Speech Enhancement”, IEEE transactions on multimedia, 4 March 2022, doi: 10.1109/TMM.2022.3199109

[25] Koichiro Ito, Masaaki Yamamoto, Kenji Nagamatsu, “Audio-Visual Speech Enhancement Method Conditioned in the Lip Motion and Speaker-Discriminative Embeddings”, IEEE International Conference on Acoustics, Speech, and Signal Processing, 6 June 2021, doi: 10.1109/ICASSP39728.2021.9414133.

[26] Zexu Pan, Ruijie Tao, Member Ieee Chenglin Xu, Fellow Ieee Haizhou Li, “Selective Hearing through Lip-reading”, 2021.

[27] Minsu Kim, Jeong Hun Yeo, Yong Man Ro, “Distinguishing Homophenes Using Multi-Head Visual-Audio Memory for Lip Reading“, AAAI Conference on Artificial Intelligence, 4 April 2022, doi: 10.48550/arXiv.2204.01725.

[28] Jiadong Wang, Xinyuan Qian, Malu Zhang, R. Tan, Haizhou Li , “Seeing What You Said: Talking Face Generation Guided by a Lip Reading Expert”, Computer Vision and Pattern Recognition, 29 March 2023, doi: 10.1109/CVPR52729.2023.01408.