VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi – 590 018

PROJECT PHASE-II (18CSP77)

On

"LIP READER USING DEEP LEARNING MODEL "

Submitted in partial fulfilment of Bachelor of engineering

in

COMPUTER SCIENCE AND ENGINEERING

Academic year 2023-2024

BY

DARSHAN R (1GD20CS010)

VARUN REDDY B (1GD20CS052)

T SAI (1GD20CS048)

ARYA BISWAS (1GD20CS006)

Under the guidance of

Dr. R. G. Sakthivelan

Professor, Department of

CSE, GCEM

GOPALAN COLLEGE OF ENGINEERING & MANAGEMENT

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BANGALORE-560 048

GOPALAN COLLEGE OF ENGINEERING AND MANAGEMENT

[ISO Certified 9001:2015, Affiliated to VTU, Belgavi, Approved by AICTE, New Delhi]

181/1, Hoodi Village, Sonnenahalli, K.R. Puram, Bangalore – 560 048

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the mini-project report entitled **"LIP READER USING DEEP LEARNING MODEL"** is a bonafide work carried out by **DARSHAN R (1GD20CS010), VARUN REDDY B (1GD20CS052), T SAI (1GD20CS048)** and **ARYA BISWAS (1GD20CS006)** in partial fulfillment of requirement of VII semester, **Bachelor of Engineering in Computer Science and Engineering** of the **Visvesvaraya Technological University**, Belgaum during the year, 2023-24. It is certified that all correction/suggestion indicated for internal assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirement in respect of project work prescribed for the said degree.

Signature of the Guide	Signature of the HOD
Dr. R. G. Sakthivelan	**Dr. Swathi Y**

External Viva

Name of the Examiners	**Signature with date**
1.	
2.	

DECLARATION

We, **DARSHAN R (1GD20CS010), ARYA BISWAS (1GD20CS006, T SAI (1GD20CS048)** and **VARUN REDDY B (1GD20CS052)**, students of VII semester B.E. in Computer Science and Engineering, **Gopalan College of Engineering and Management**, Bangalore, hereby declare that the project work entitled **"LIP READER USING DEEP LEARNING MODEL"** submitted to the **Visvesvaraya Technological University** during the academic year 2023-24,is a record of an original work done by us under the guidance of **Dr. R. G. Sakthivelan** Professor, Department of Computer Science and Engineering, Gopalan College of Engineering and Management, Bangalore. This project work is submitted in partial fulfilment of the requirements for the award ofthe degree of **Bachelor of Engineering in Computer Science and Engineering**. The results embodiedin this project have not been submitted to any other University or Institute for the award of any degree.

Date:

Place: Bangalore

DARSHAN R (1GD20CS010)

ARYA BISWAS (1GD20CS006)

T SAI (1GD20CS048)

VARUN REDDY B (1GD20CS052)

ABSTRACT

Lip reading, the process of understanding spoken language by observing lip movements, is an intricate task that combines visual perception with linguistic comprehension. In this study, we propose a novel deep learning architecture tailored specifically for lip reading tasks. Our model leverages a combination of 3D convolutional neural networks (CNNs) to capture spatial-temporal features from lip video sequences and bidirectional long short-term memory (BiLSTM) networks to effectively model temporal dependencies within the lip movements. Through extensive experimentation on diverse datasets, we demonstrate the superior performance of our approach compared to state-of-the-art methods.

Furthermore, we conduct thorough analyses to understand the model's robustness to variations in speaking rates, accents, and environmental conditions. Our findings underscore the potential of deep learning techniques in advancing the field of lip reading, with implications for applications in assistive technologies, human-computer interaction, and security systems. This research contributes to the broader goal of enhancing communication accessibility for individuals with hearing impairments and addressing real-world challenges in noisy or audio-restricted environments.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany a successful completion of any task would be incomplete without the mention of people who made it possible, success is the epitome of hard work and perseverance, but steadfast of all is encouraging guidance.

We express our gratitude towards **Dr. C. Prabhakar**, Director, Gopalan Foundation and **Dr. Arun Vikas Singh**, Principal, Gopalan College of Engineering and Management, for providing with the infrastructure and resources which helped in completion of this project.

We are really thankful to **Dr. Swathi Y**, Professor and HOD of Computer Science & Engineering department, GCEM for her constant support extended towards us during the course of this project.

We are really thankful to **Dr. R. G. Sakthivelan**,Professor, our guide, Computer Science & Engineering department, GCEM for his constant support extended towards us during the course of this project.

Finally, we would like to extend our thanks to our friends and family who supported us and provided us withsuggestions to make the project better.

DARSHAN R (1GD20CS010)
ARYA BISWAS (1GD20CS006)
T SAI (1GD20CS048)
VARUN REDDY B (1GD20CS052)

TABLE OF CONTENTS

List of figure

CHAPTER 1
INTRODUCTION

CHAPTER 1

INTRODUCTION

The comprehension of spoken language is a fundamental aspect of human communication, serving as the cornerstone of interactions in diverse social, professional, and personal contexts. However, there exist scenarios where traditional audio-based communication channels encounter limitations, impeding effective understanding and communication. In such cases, visual cues, particularly those derived from observing lip movements, offer a valuable alternative or complement to auditory signals.

Lip reading, the process of deciphering spoken words by analyzing the movements of the lips, holds immense potential in mitigating communication barriers, especially for individuals with hearing impairments or in environments with high levels of noise. While lip reading has long been studied and practiced, recent advancements in deep learning have opened up new avenues for automatic lip reading systems, promising improved accuracy, efficiency, and adaptability. In this context, this study aims to develop and evaluate a deep learning-based approach for automatic lip reading, leveraging cutting-edge techniques to enhance the accessibility and effectiveness of communication in challenging audio environments.

1.1 BACKGROUND

Lip reading, also known as speechreading, is a communication technique used by individuals with hearing impairments and in noisy environments. Traditionally, lip reading relied on manual transcription or rule-based systems, limiting its effectiveness. However, recent advances in deep learning, particularly convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have shown promise in automatic lip reading. Despite progress, challenges such as limited datasets, variability in speech patterns, and real-time processing remain. Overcoming these challenges requires interdisciplinary collaboration to develop robust algorithms that bridge visual cues and spoken language understanding.

1.1.1 Driving Force behind the idea

The driving force behind developing a lip reader lies in addressing security concerns where traditional methods may fall short. By harnessing advancements in deep learning and computer vision, a lip reader interprets lip movements into understandable speech, offering an additional layer of authentication and surveillance. This technology not only assists individuals with hearing impairments but also enhances security systems by providing reliable identification through lip reading-based authentication. In essence, the aim is to bolster security measures while promoting inclusivity and accessibility.

1.2 AIM OF THE PROJECT

The aim of the project is to develop a robust lip reading system using deep learning techniques to accurately interpret and transcribe spoken words solely from video input of lip movements. This system aims to improve accessibility for individuals with hearing impairments and enhance communication in noisy environments. Additionally, the project seeks to explore the potential applications of lip reading technology in security systems, human-computer interaction, and other domains where visual cues can supplement or replace auditory information. Ultimately, the goal is to promote inclusivity, accessibility, and effective communication for all individuals, regardless of their hearing abilities or environmental challenges.

1.3 PROBLEM STATEMENT

The problem addressed by this project revolves around the limitations of traditional communication methods, particularly for individuals with hearing impairments or in environments with high levels of noise. Despite advancements in technology, existing solutions often struggle to accurately interpret spoken language in such contexts. This poses significant challenges for effective communication and accessibility. Furthermore, conventional lip reading methods are often manual, subjective, and prone to errors. Therefore, the project aims to develop an automated lip reading system using deep learning techniques to overcome these limitations. Specifically, the system aims to accurately transcribe spoken words solely from video input of lip movements, thereby improving accessibility for individuals with hearing impairments and enhancing communication in noisy or audio-restricted environments. Additionally, the project seeks to explore the potential applications of lip reading technology in security systems, human-computer interaction, and other domains, addressing the broader challenge of bridging the gap between visual information and spoken language understanding.

1.4 OBJECTIVE

The primary objectives of the project, "Lip Reader using deep learning model" can be further detailed as follows:

1. **Model Development:** The project aims to develop a sophisticated lip reading model using deep learning techniques. This model will accurately transcribe spoken words solely from video input of lip movements, catering to the communication needs of individuals with hearing impairments and in noisy environments.

2. **Dataset Collection and Preparation**: A diverse dataset of lip movement videos will be collected and preprocessed to facilitate robust training and evaluation of the lip reading model. This dataset will encompass variations in speakers, languages, and environmental conditions, enhancing the model's adaptability and performance.

3. **Implementation and Optimization**: The lip reading model will be implemented and optimized using cutting-edge deep learning techniques. This includes leveraging convolutional neural networks (CNNs) for feature extraction and recurrent neural networks (RNNs) for modeling temporal dependencies, ensuring the model's accuracy and efficiency.

4. **Performance Evaluation:** Extensive evaluation of the lip reading model will be conducted on benchmark datasets and real-world scenarios. Performance metrics such as accuracy,

robustness, and real-time processing capabilities will be assessed to validate the effectiveness of the developed system.

1.5 ORGANIZATION OF THE PROJECT

The project report id organized as follows:

- **Chapter 1-Introduction**

  This chapter tells about the problem statement, background of the project, motivation, its existing system, and its effect as well as proposed system with its theoretical outline.

- **Chapter 2-Literature Survey**

  Gives brief overview of the paper and the research sources that have been studied to establish through a n understanding of the under consideration.

- **Chapter 3-System Requirements**

  Discuss in detail about the different kind of requirement needed to successfully complete the project.

- **Chapter 4-System Analysis**

  Gives detail about several analysis that are performed to facilitate taking decision of whether the project isfeasible enough or not.

- **Chapter 5-System Design**

  Gives the design description of the project, conceptual and detailed design well supported with the designdiagrams.

- **Chapter 6-Conclusion And Future Enhancement**

  Discussed about how the project is successfully implemented and what the other features that has to considered to enhance at the future.

CHAPTER 2
LITERATURE SURVEY

LITERATURE SURVEY

A literature survey or a literature review in a project report shows the various analyses and research made in the field of interest and the results already published, taking into account the various parameters of the project and the extent of the project. Literature survey is mainly carried out in order to analyze the background of the current project which helps to find out flaws in the existing system & guides on which unsolved problems we can work out. So, the following topics not only illustrate the background of the project but also uncover the problems and flaws which motivated to propose solutions and work on this project.

A literature survey is a text of a scholarly paper, which includes the current knowledge including substantive findings, as well as theoretical and methodological contributions to a particular topic. Literature reviews use secondary sources, and do not report new or original experimental work. Most often associated with academic- oriented literature, such as a thesis, dissertation or a peer-reviewed journal article, a literature review usually precedes the methodology and results sectional though this is not always the case. Literature reviews are also common in are search proposal or prospectus (the document that is approved before a student formally begins a dissertation or thesis). Its main goals are to situate the current study within the body of literature and to provide context for the particular reader. Literature reviews are a basis for researching nearly every academic field.

A literature survey includes the following:

- Existing theories about the topic which are accepted universally.
- Books written on the topic, both generic and specific.
- Research done in the field usually in the order of oldest to latest.
- Challenges being faced and on-going work, if available.

Objectives of Literature Survey:

- Learning the definitions of the concepts.
- Access to latest approaches, methods and theories.
- Discovering research topics based on the existing research
- Concentrate on your own field of expertise- Even if another field uses the same words, they usually mean completely.
- It improves the quality of the literature survey to exclude sidetracks- Remember to explicate what is excluded.

2.1 EXISTING SYSTEM

Traditional methods of lip reading often rely on manual transcription or rule-based systems, which are labor-intensive, subjective, and prone to errors. These approaches require extensive training and expertise, making them inaccessible to many individuals. Furthermore, traditional lip reading methods may struggle to accurately interpret speech in noisy environments or when speakers exhibit variations in lip movements or speaking styles. Additionally, these methods typically lack scalability and real-time processing capabilities, limiting their practicality in dynamic settings. Overall, the existing systems for lip reading face significant challenges in terms of accuracy, scalability, and accessibility, highlighting the need for more advanced and automated solutions to address the communication needs of individuals with hearing impairments and in challenging audio environments.

2.2 PROPOSED SYSTEM

The proposed lip reading system aims to overcome the limitations of traditional methods by leveraging advancements in deep learning and computer vision. Through the integration of convolutional neural networks (CNNs) and recurrent neural networks (RNNs), the system will automatically transcribe spoken words solely from video input of lip movements. This approach offers several advantages over existing systems, including higher accuracy, scalability, and adaptability to varying environmental conditions and speaking styles. By training the model on diverse datasets encompassing different speakers, languages, and environmental factors, the proposed system seeks to improve generalization and robustness. Furthermore, the system will be optimized for real-time processing, enabling its deployment in dynamic settings where timely communication is crucial. Overall, the proposed lip reading system represents a significant advancement in accessibility and communication for individuals with hearing impairments and in noisy or audio-restricted environments.

2.3 RELATED WORKS

No.	TITLE	AUTHOR	OBSERVATIONS	LIMITATIONS
1	LipNet:End-to-End Sentence-level Lipreading Feature Selection	Yannis M. Assael, Brendan Shilingford, Simon Whiteson, Nando de Freitas	Maps a variable length sequence of video frames to text, making use of Spatiotemporal convolutions, RNN,trained entirely End-to-End	Accuracy is limited to 92.5%. Complex to implement in Real Time
2	Lip Reading Sentences in the Wild	Joon Son Chung, A. Senior, O. Vinyals, Andrew Zisserman	(1) a Watch, Listen, Attend and Spell (WLAS) network that learns to transcribe videos of mouth motion to characters, (2) a curriculum learning strategy to accelerate training.	Generalization to Diverse Accents and Languages Lack of Emotional and Expressive Context
3	Combining Residual Networks	Themos Stafylakis, Georgios	The system is a combination of spatiotemporal	Limited Vocabulary and Word Diversity

	with LSTMs for Lipreading	Tzimiropoulos	convolutional, residual and bidirectional Long Short-Term Memory networks.	Sensitivity to Noise and Variability

CHAPTER 3
SYSTEM REQUIREMENTS

CHAPTER 3

SYSTEM REQUIREMENT SPECIFICATION

3.1 FUNCTIONAL REQUIREMENTS

The functional requirements for the project "Lip Reader using Deep learning model" include:

3.1.1 Video Input Processing: The system should be capable of processing video input containing lip movements, extracting relevant frames for analysis.

3.1.2 Feature Extraction: It must implement algorithms to extract spatial-temporal features from lip images within video frames, utilizing convolutional neural networks (CNNs).

3.1.3 Speech Transcription: The system needs to decode the sequence of features into phonemes or words, providing an accurate transcription of the spoken words represented by the observed lip movements.

3.1.4 Real-time Processing: It should be optimized for real-time performance, ensuring minimal delay in transcribing lip movements into speech, suitable for dynamic communication scenarios.

3.1.5 Robustness to Environmental Factors: The system must exhibit robustness to environmental factors such as lighting conditions, background noise, speaker variability, and occlusions, ensuring accurate transcription across diverse conditions.

3.2 NON-FUNCTIONAL REQUIREMENTS

3.2.1 **Accuracy:** The system must achieve a minimum accuracy level of 90% in transcribing lip movements into spoken words to ensure reliable communication.

3.2.2 **Real-time Performance:** The system should process lip movements and provide transcription results within 250 milliseconds to facilitate seamless and natural communication.

3.2.3 **Scalability:** It should be designed to handle concurrent requests from multiple users and accommodate a growing user base without degradation in performance.

3.2.4 **User Interface Responsiveness:** The user interface should respond to user interactions within 100 milliseconds to provide a smooth and responsive user experience.

3.2.5 **Data Privacy:** The system must comply with data privacy regulations, ensuring that user data, including lip movement videos, is encrypted during transmission and storage and accessible only to authorized personnel.

3.3 Resource Requirements:

3.3.1 Hardware:

We need 1 machine with following minimal requirements

CPU	: Intel 2.1 GHZ
Memory	: 4GB
Disk	: 40GB
Display	: 15 inch color

3.3.2 Software:

The proposed solution will be implemented in Python

Coding	: Python
Platform	: Python 3.7 and above
Tool	: Anaconda Spyder / Google Colab
Libraries	: pandas, numpy, sklearn ,tensorflow

CHAPTER 4
SYSTEM ANALYSIS

CHAPTER 4

SYSTEM ANALYSIS

A system is an orderly group of interdependent components linked together according to a plan to achieve a specific objective. Its main characteristics are organization, interaction, interdependence, integration and a central objective.

System analysis and design are the application of the system approach to problem solving generally using computers. To reconstruct a system the analyst must consider its elements output and inputs, processors, controls, feedback and environment.

Analysis is a detailed study of the various operations performed by a system and their relationships within and outside of the system. One aspect of analysis is defining the boundaries of the system and determining whether or not a candidate system should consider other related systems. During analysis data are collected on the available files decision points and transactions handled by the present system. This involves gathering information and using structured tools for analysis.

4.1 FEASIBILITY STUDY

In order to assess the viability of implementing the proposed project, a comprehensive feasibility study is conducted, covering economic, technical, and social aspects.

4.1.1 ECONOMIC FEASIBILITY

4.1.1.1 COST ESTIMATION

Development Costs:

The development costs encompass expenses related to the design, implementation, and testing of Lip reader system. This includes expenditures on software development tools, machine learning libraries, and personnel salaries.

Operational Costs:

Operational costs involve ongoing expenses for system maintenance, cloud service usage fees, and periodic updates. Additionally, costs associated with training personnel for effective system utilization are considered.

4.1.1.2 BENEFIT ESTIMATION

1. **Enhanced Communication Accessibility:**

The lip reading system improves communication accessibility for individuals with hearing impairments, facilitating their participation in social, educational, and professional settings.

2. **Improved Security and Surveillance:**

The system enhances security measures by providing additional authentication methods and assists in surveillance applications by identifying individuals or detecting suspicious behavior based on lip movements captured in video footage.

4.1.2 TECHNICAL FEASIBILITY

4.1.2.1 Hardware and Software Requirements: Identify the necessary hardware components (e.g., CPUs, GPUs) and software tools (e.g., TensorFlow, OpenCV) required for developing and deploying the lip reading system.

4.1.2.2 Availability of Required Technologies: Assess the accessibility and compatibility of the technologies needed for the system, ensuring that essential hardware, software, and libraries are readily available and supported by vibrant developer communities.

4.1.2.3 Model Optimization Techniques: Evaluate techniques such as transfer learning, model pruning, and quantization to optimize deep learning models, ensuring efficient performance and scalability for real-world deployment of the lip reading system.

4.1.3 SOCIAL FEASIBILITY

4.1.3.1 User Acceptance and Accessibility: Assess the system's appeal to users, particularly those with hearing impairments, focusing on usability, inclusivity, and accessibility for individuals with disabilities.

4.1.3.2 Ethical and Cultural Considerations: Evaluate the system's adherence to ethical guidelines regarding privacy, consent, and data protection, while also considering cultural sensitivities and communication norms to ensure social acceptance and trust.

In conclusion, the feasibility study indicates that the proposed " Lip Reader using Deep learning model " project is economically, technically, and socially viable. The anticipated benefits in terms of cost savings, revenue generation, and improved security justify the investment in the development and implementation of the anomaly detection system.

CHAPTER 5
SYSTEM DESIGN

CHAPTER 5

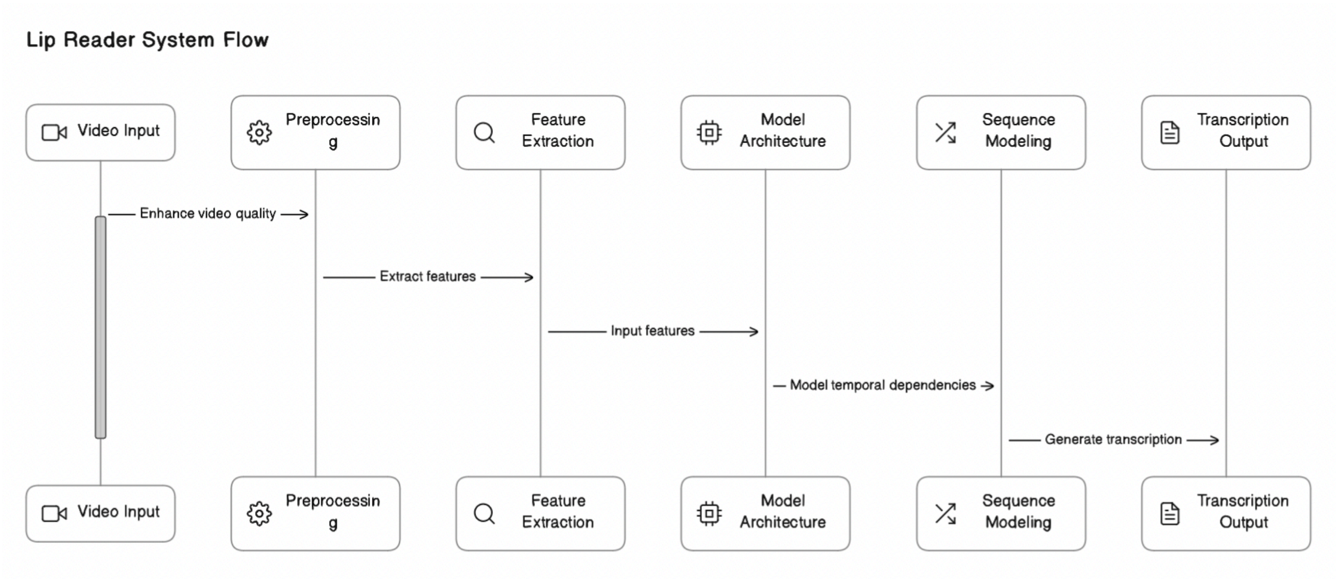
SYSTEM DESIGN

Software design is the process by which an agent creates a specification of a software artifact, intended to accomplish goals, using a set of primitive components and subject to constraints. Software design may refer to either "all the activity involved in conceptualizing, framing, implementing, commissioning, and ultimately modifying complex systems" or "the activity following requirements specification and before programming, as a stylized software engineering process.". The software design of the system is documented in this chapter.

5.1 ARCHITECTURE

System architecture is the conceptual design that defines the structure and behavior of a system. An architecture description is a formal description of a system, organized in a way that supports reasoning about the structural properties of the system. It defines the system components or building blocks and provides a plan from which products can be procured, and systems developed, that will work together to implement the overall system.
The System architecture is shown below.

This design illustrates the processing pipeline of a lip reader system, from input video to transcription output, highlighting the key components involved in the process.



5.1.1 **Video Input:** The system takes video input containing lip movements**.**

5.1.2 **Preprocessing:** The input video undergoes preprocessing to enhance its quality and prepare it for feature extraction.

5.1.3 **Feature Extraction:** Relevant spatial-temporal features are extracted from the preprocessed video data.

5.1.4 **Model Architecture:** The system employs a model architecture**,** including both feature extraction and sequence modeling components, often based on deep learning techniques.

5.1.5 **Sequence Modeling:** The sequence modeling component captures temporal dependencies in the sequence of lip movements.

5.1.6 **Transcription Output:** The final output of the system is the transcription of spoken words inferred from the lip movements captured in the input video.

CHAPTER 6
SYSTEM IMPLEMENTATION

CHAPTER 6

IMPLEMENTATION/METHODOLOGY

6.1 INTRODUCTON

This chapter outlines the detailed plan of implementation and methodology for the project on "Anomaly Detection in Cloud Networks using Machine Learning Techniques." The strategic approach encompasses various stages, including data collection, feature engineering, model development, and evaluation. The objective is to systematically implement the proposed anomaly detection system, ensuring robustness, efficiency, and effectiveness in securing cloud networks.

6.2 STAGES OF IMPLEMENTATION:

6.2.1 Research and Requirements Gathering:

Conduct thorough research on existing lip reading systems, deep learning techniques, and relevant technologies. Gather requirements from stakeholders, including users with hearing impairments, to understand their needs and preferences.

6.2.2 Data Collection and Annotation:

Collect a diverse dataset of lip movement videos and annotate them with corresponding spoken words. Ensure the dataset encompasses variations in speakers, languages, and environmental conditions to facilitate robust training and evaluation of the system.

6.2.3 Model Development and Training:

Develop the lip reading model using deep learning techniques, such as convolutional neural networks (CNNs) for feature extraction and recurrent neural networks (RNNs) for temporal modeling. Train the model on the annotated dataset, optimizing its parameters for accuracy and efficiency.

6.2.4 Validation and Evaluation:

Validate the performance of the lip reading system using benchmark datasets and real-world scenarios. Evaluate its accuracy, robustness to environmental factors, and real-time processing capabilities, ensuring it meets the specified requirements and user expectations.

6.2.5 User Interface Design and Development:

Design and develop a user-friendly interface for interacting with the lip reading system. Ensure accessibility and inclusivity for individuals with disabilities, considering factors such as visual impairments and motor disabilities.

6.2.6 Testing and Quality Assurance:

Conduct rigorous testing of the system to identify and address any issues or bugs. Perform functional testing, usability testing, and accessibility testing to ensure the system meets quality standards and user needs.

6.2.7 Deployment and Integration:

Deploy the lip reading system in relevant environments, such as assistive technologies for individuals with hearing impairments or security systems for authentication and surveillance. Integrate the system with existing technologies and infrastructure, ensuring compatibility and seamless operation.

6.2 8 Training and Documentation:

Provide training and support to users and administrators on how to use and maintain the lip reading system effectively. Develop comprehensive documentation, including user manuals, technical guides, and troubleshooting resources, to facilitate adoption and usage.

CHAPTER 7
TESTING

CHAPTER 7

TESTING

7.1: Unit Testing: For unit testing accuracy measurement in TensorFlow for lip reading tasks, we create test cases to ensure the accuracy computation functions correctly. A typical test case involves generating sample predictions and ground truth labels, computing accuracy using TensorFlow's built-in accuracy metric, and asserting that the computed accuracy matches the expected value. For example, we can use the tf.keras.metrics.Accuracy() class to calculate accuracy by updating its internal state with the labels and predictions. By comparing the computed accuracy with the expected value, we verify that the accuracy metric behaves as expected during training and evaluation of lip reading models. This unit testing approach helps ensure the reliability and accuracy of the system's performance metrics, providing confidence in its effectiveness for interpreting visual lip movements accurately.

7.2: Validation Testing

Validation testing for a lip reading system involves evaluating its performance on a separate validation dataset to ensure reliable and consistent behavior across diverse samples and conditions. After preparing the validation dataset, the trained model is used to make predictions on this data. By processing the video inputs through the model, predicted text or phonetic representations are obtained for each sample. Performance metrics such as accuracy, precision, recall, and F1 score are then calculated to assess the model's effectiveness in accurately interpreting lip movements and classifying spoken words. The analysis of these metrics provides valuable insights into the system's strengths and weaknesses, guiding potential refinements or enhancements to improve its overall performance. Through rigorous validation testing, the lip reading system can be validated for its reliability and suitability in real-world scenarios, ultimately ensuring its effectiveness in facilitating communication for individuals with hearing impairments.

CHAPTER 8
CONCLUSION AND FUTURE ENHANCEMENTS

CHAPTER 8:

CONCLUSION AND FUTURE ENHANCEMENTS

As technology continues to advance, there are several promising avenues for enhancing the capabilities of the lip reading system. One such enhancement involves the integration of multi-modal fusion techniques, which combine information from various sources such as audio signals and facial expressions. By leveraging multiple modalities, the system can improve transcription accuracy and robustness, particularly in challenging environments with background noise or variations in speech patterns.

Additionally, future enhancements could focus on implementing adaptive learning mechanisms within the lip reading system. These mechanisms would enable the system to adapt to individual user preferences and speaking styles over time, leading to personalized transcription experiences. By continuously learning from user interactions and feedback, the system can refine its algorithms and improve transcription accuracy for each user.

References

[1]. LipNet: End-to-End Sentence-level Lipreading by Yannis M. Assael, Brendan Shillingford, Shimon Whiteson, Nando de Freitas.

[2] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, et al. Deep Speech 2: End-to-end speech recognition in English and Mandarin. arXiv preprint arXiv:1512.02595, 2015.

[3] J. S. Chung and A. Zisserman. Lip reading in the wild. In Asian Conference on Compute Vision, 2016a.

[4] J. S. Chung and A. Zisserman. Out of time: automated lip sync in the wild. In Workshop on.Multi-view Lip-reading, ACCV, 2016b.

APPENDIX A
SNAPSHOTS

SNAPSHOTS

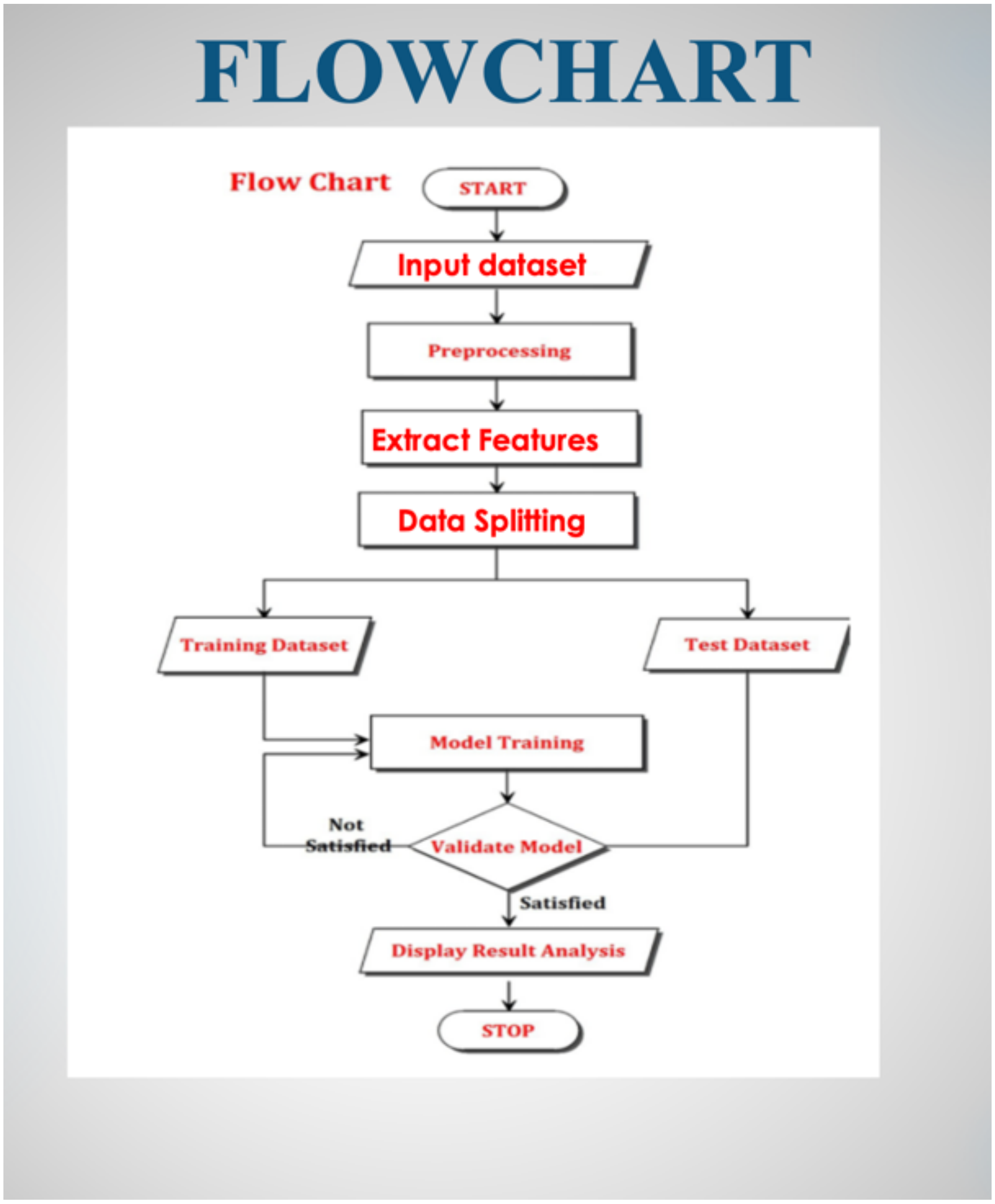


Figure-1: Flow Chart

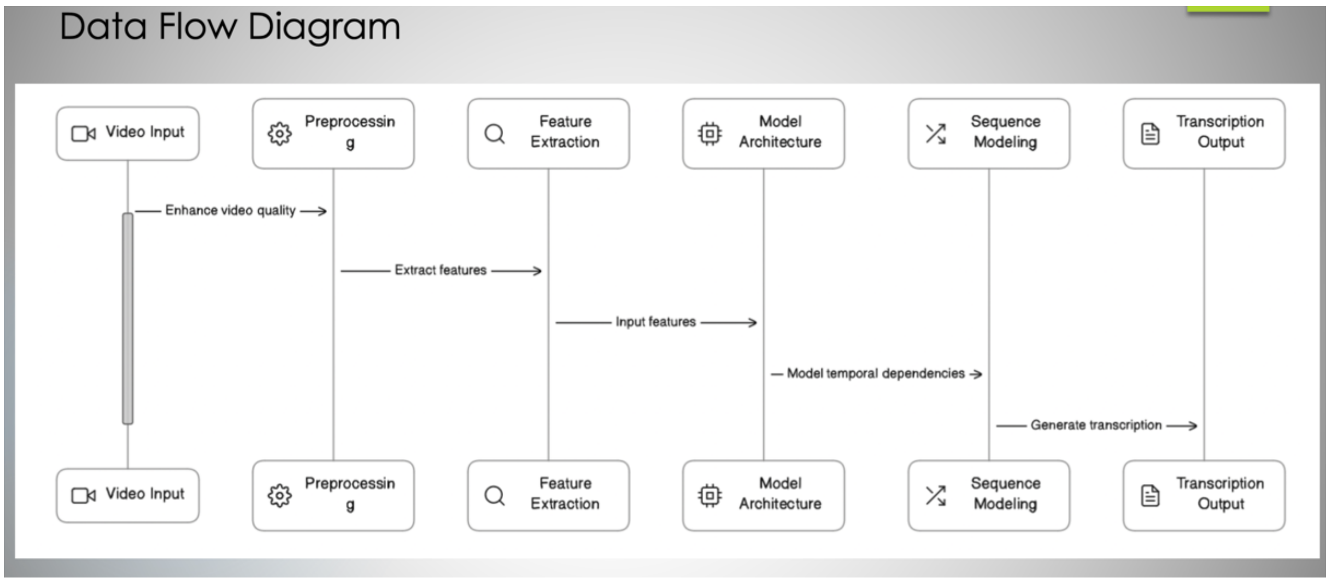


Figure-2: Data Flow

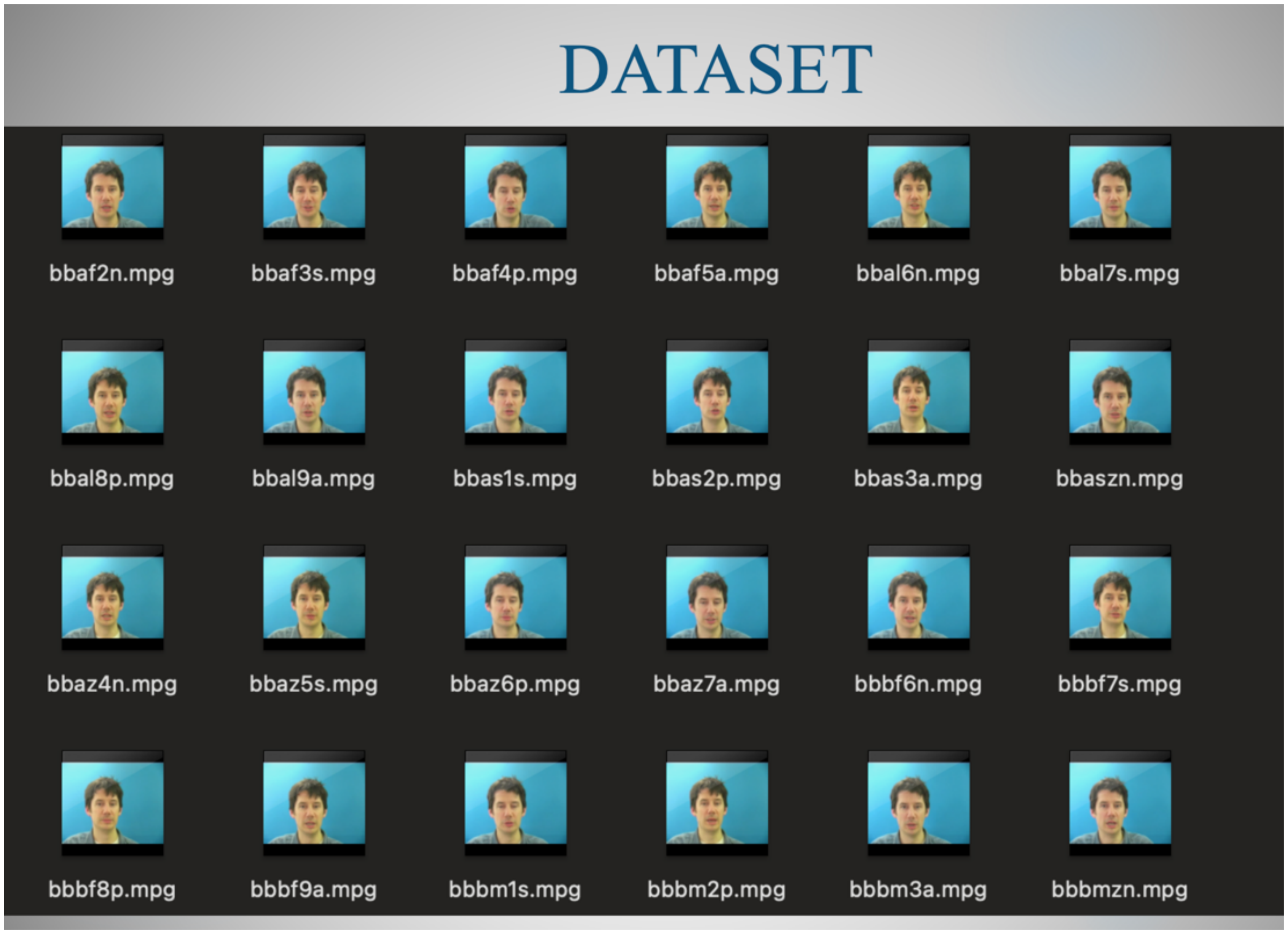
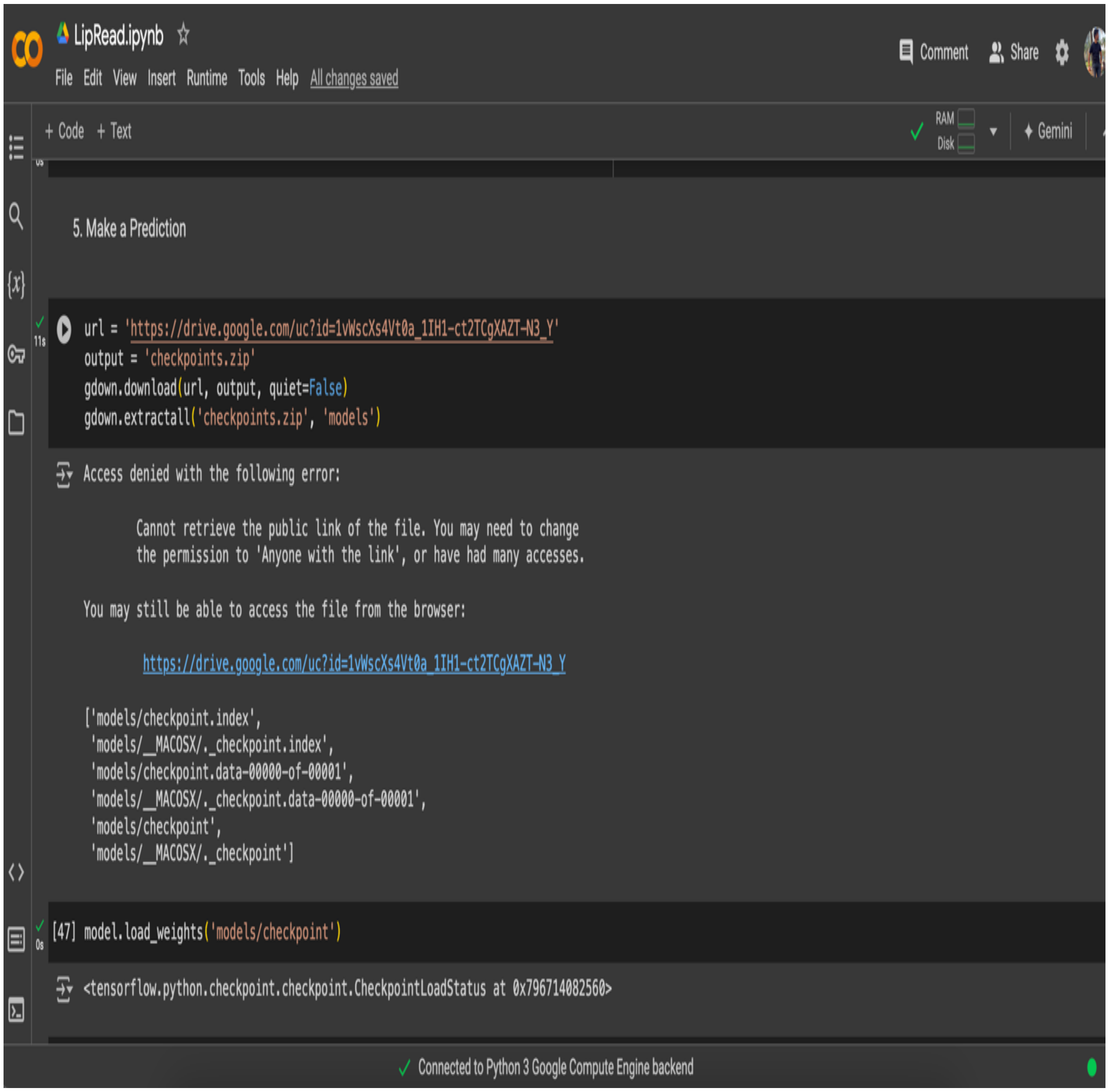


Figure-3: DataSet



5. Make a Prediction

```
url = 'https://drive.google.com/uc?id=1vWscXs4Vt0a_1IH1-ct2TCgXAZT-N3_Y'
output = 'checkpoints.zip'
gdown.download(url, output, quiet=False)
gdown.extractall('checkpoints.zip', 'models')
```

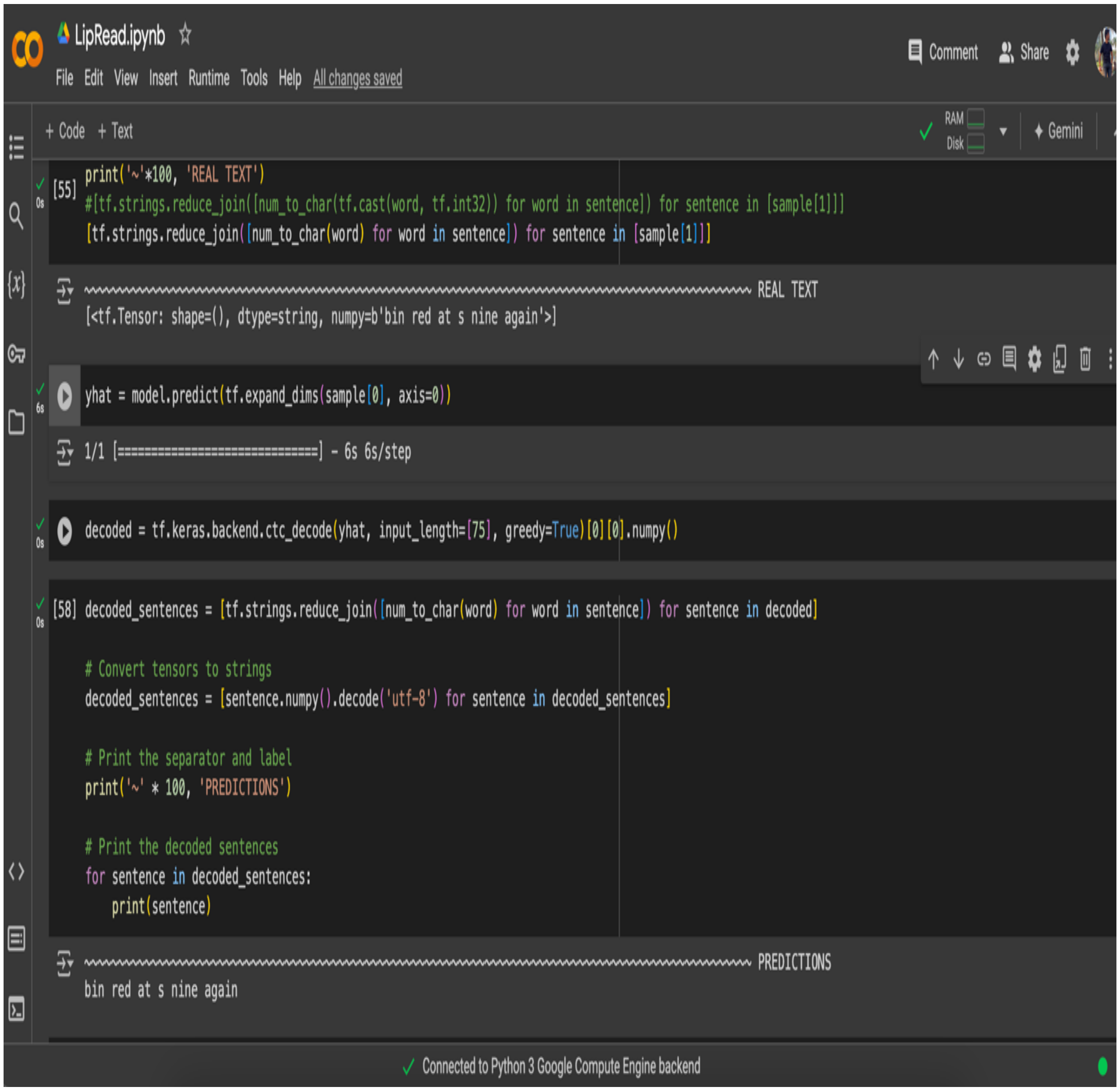
```
Access denied with the following error:

        Cannot retrieve the public link of the file. You may need to change
        the permission to 'Anyone with the link', or have had many accesses.

You may still be able to access the file from the browser:

         https://drive.google.com/uc?id=1vWscXs4Vt0a_1IH1-ct2TCgXAZT-N3_Y

['models/checkpoint.index',
 'models/__MACOSX/._checkpoint.index',
 'models/checkpoint.data-00000-of-00001',
 'models/__MACOSX/._checkpoint.data-00000-of-00001',
 'models/checkpoint',
 'models/__MACOSX/._checkpoint']
```

```
[47] model.load_weights('models/checkpoint')
```

```
<tensorflow.python.checkpoint.checkpoint.CheckpointLoadStatus at 0x796714082560>
```

Connected to Python 3 Google Compute Engine backend

Figure-4: Prediction



```
[55] print('~'*100, 'REAL TEXT')
     #[tf.strings.reduce_join([num_to_char(tf.cast(word, tf.int32)) for word in sentence]) for sentence in [sample[1]]]
     [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 [==============================] - 6s 6s/step

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

[58] decoded_sentences = [tf.strings.reduce_join([num_to_char(word) for word in sentence]) for sentence in decoded]

     # Convert tensors to strings
     decoded_sentences = [sentence.numpy().decode('utf-8') for sentence in decoded_sentences]

     # Print the separator and label
     print('~' * 100, 'PREDICTIONS')

     # Print the decoded sentences
     for sentence in decoded_sentences:
         print(sentence)

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ PREDICTIONS
bin red at s nine again
```

✓ Connected to Python 3 Google Compute Engine backend

Figure-5: Output

APPENDIX B
SOURCE CODE

Source Code

1. Installing And Importing Dependecies

```
!pip install opencv-python==4.6.0.66 tensorflow==2.10.1 imageio==2.9.0
matplotlib==3.6.2 gdown==4.6.0
# !pip install opencv-python matplotlib imageio gdown tensorflow


# !pip install imageio==2.23.0
# !pip install --upgrade gdown





import os
import cv2
import tensorflow as tf
import numpy as np
from typing import List
from matplotlib import pyplot as plt
import imageio


import sys
print(sys.version)


3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0]

tf.config.list_physical_devices('GPU')

[]
```

To Prevent Exponential Memory Growth

```
physical_devices = tf.config.list_physical_devices('GPU')
try:
  tf.config.experimental.set_memory_growth(physical_devices[0], True)
except:
  pass
```

1. Build Data Loading Functions

```
import gdown


 url = 'https://drive.google.com/uc?id=1YlvpDLix3S-U8fd-gqRwPcWXAXm8JwjL'
output = 'data.zip'
```

```
gdown.download(url, output, quiet=False)
gdown.extractall('data.zip')




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,:])  #To isolate the lip region
with this statiscal value
    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


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


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()})"
)

The vocabulary is: ['', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k'
, 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z',
"'", '?', '!', '1', '2', '3', '4', '5', '6', '7', '8', '9', ' '] (size =40)

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:]


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
```

```
test_path = './data/alignments/s1/bbal6n.align'
```

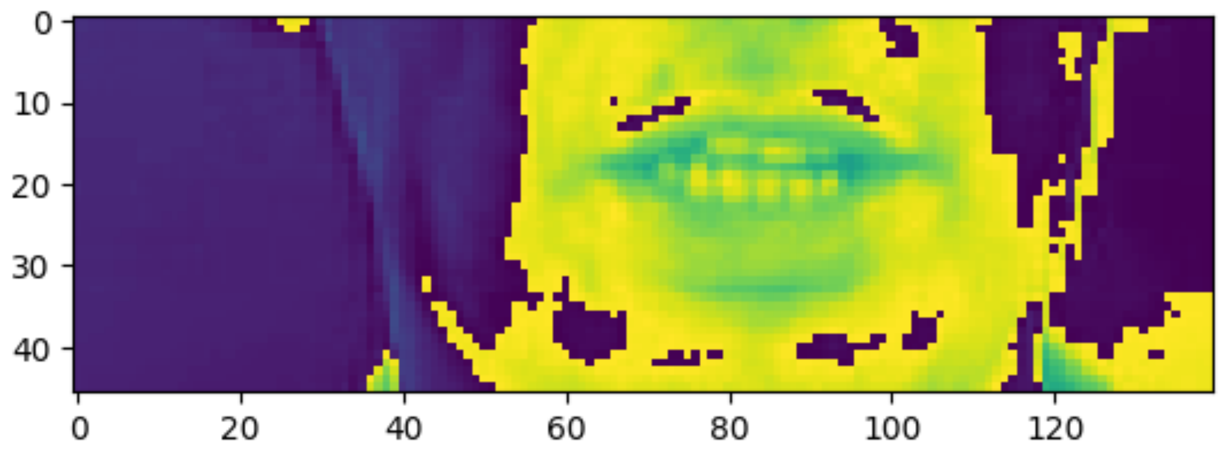
```
tf.convert_to_tensor(test_path).numpy().decode('utf-8').split('/')[-
1].split('.')[0]
```

```
'bbal6n'
```

```
frames, alignments = load_data(tf.convert_to_tensor(test_path))
```

```
plt.imshow(frames[40])
```

```
<matplotlib.image.AxesImage at 0x79672723d8d0>
```



```
alignments
```

```
<tf.Tensor: shape=(21,), dtype=int64, numpy=
array([ 2,  9, 14, 39,  2, 12, 21,  5, 39,  1, 20, 39, 12, 39, 19,  9, 24,
       39, 14, 15, 23])>
```

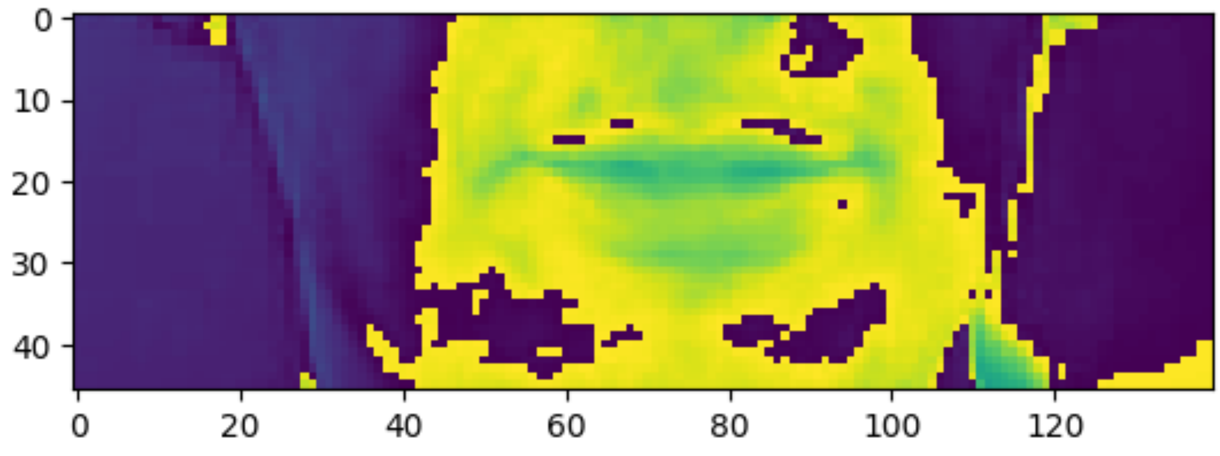
```
tf.strings.reduce_join([bytes.decode(x) for x in
num_to_char(alignments.numpy()).numpy()])

<tf.Tensor: shape=(), dtype=string, numpy=b'bin blue at l six now'>

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

2. Create Data Pipeline

```
from matplotlib import pyplot as plt


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)
# Added for split
train = data.take(450)
test = data.skip(450)


frames, alignments = data.as_numpy_iterator().next()


sample = data.as_numpy_iterator()


val = sample.next(); val[0]

array([[[[[ 1.4848511 ],
          [ 1.6052444 ],
          [ 1.5249822 ],
          ...,
          [ 0.28091776],
          [ 0.28091776],
          [ 0.20065555]],

         [[ 1.5249822 ],
          [ 1.6052444 ],
          [ 1.44472   ],
          ...,
          [ 0.28091776],
          [ 0.28091776],
          [ 0.20065555]],

         [[ 1.3644577 ],
          [ 1.3644577 ],
          [ 1.44472   ],
          ...,
          [ 0.28091776],
          [ 0.16052444],
          [ 0.16052444]],

         ...,

         [[ 1.0434089 ],
          [ 1.0434089 ],
          [ 1.0434089 ],
          ...,
          [10.233433  ],
          [10.233433  ],
          [10.233433  ]],

         [[ 1.0032778 ],
          [ 1.0032778 ],
          [ 0.9631467 ],
          ...,
          [10.193302  ],
          [10.193302  ],
          [10.193302  ]],

         [[ 0.9631467 ],
          [ 0.9631467 ],
          [ 0.9631467 ],
          ...,
          [10.193302  ],
          [10.193302  ],
          [10.193302  ]]],


        [[[ 1.4045888 ],
          [ 1.6052444 ],
          [ 1.5651133 ],
          ...,
          [ 0.24078667],
          [ 0.20065555],
          [ 0.16052444]],

         [[ 1.5249822 ],
          [ 1.4848511 ],
          [ 1.44472   ],
          ...,
          [ 0.24078667],
          [ 0.20065555],
          [ 0.16052444]],
  [[ 1.3644577 ],
   [ 1.3644577 ],
   [ 1.4045888 ],
   ...,
   [ 0.24078667],
   [ 0.16052444],
   [ 0.16052444]],

  ...,

  [[ 1.0032778 ],
   [ 1.0032778 ],
   [ 1.0032778 ],
   ...,
   [10.233433  ],
   [10.233433  ],
   [10.233433  ]],

  [[ 1.0032778 ],
   [ 1.0032778 ],
   [ 0.9631467 ],
   ...,
   [10.193302  ],
   [10.193302  ],
   [10.193302  ]],

  [[ 0.9631467 ],
   [ 0.9631467 ],
   [ 0.9631467 ],
   ...,
   [10.193302  ],
   [10.193302  ],
   [10.193302  ]]],


[[[ 1.44472   ],
  [ 1.6052444 ],
  [ 1.5651133 ],
  ...,
  [ 0.20065555],
  [ 0.16052444],
  [ 0.16052444]],

 [[ 1.4848511 ],
  [ 1.4848511 ],
  [ 1.44472   ],
  ...,
  [ 0.20065555],
  [ 0.16052444],
 [[ 1.5651133 ],
  [ 1.5651133 ],
  [ 1.44472   ],
  ...,
  [ 0.12039334],
  [ 0.16052444],
  [ 0.16052444]],

 [[ 1.44472   ],
  [ 1.44472   ],
  [ 1.4848511 ],
  ...,
  [ 0.16052444],
  [ 0.16052444],
  [ 0.16052444]],

 ...,

 [[ 1.0032778 ],
  [ 1.0032778 ],
  [ 1.0032778 ],
  ...,
  [10.11304   ],
  [10.11304   ],
  [10.11304   ]],

 [[ 1.0032778 ],
  [ 1.0032778 ],
  [ 1.0032778 ],
  ...,
  [10.11304   ],
  [10.11304   ],
  [10.11304   ]],

 [[ 1.0032778 ],
  [ 1.0032778 ],
  [ 0.9631467 ],
  ...,
  [10.11304   ],
  [10.11304   ],
  [10.11304   ]]],


[[[ 1.5249822 ],
  [ 1.5249822 ],
  [ 1.44472   ],
  ...,
  [ 0.16052444],
  [ 0.12039334],
  [ 0.12039334]],

 [[ 1.44472   ],
  [ 1.5651133 ],
  [ 1.4848511 ],
  ...,
  [ 0.16052444],
  [ 0.12039334],
  [ 0.12039334]],

 [[ 1.4848511 ],
  [ 1.4848511 ],
  [ 1.5249822 ],
  ...,
  [ 0.20065555],
  [ 0.16052444],
  [ 0.16052444]],

 ...,

 [[ 1.0032778 ],
  [ 1.0032778 ],
  [ 0.9631467 ],
  ...,
  [10.193302  ],
  [10.193302  ],
  [10.193302  ]],

 [[ 1.0032778 ],
  [ 0.9631467 ],
  [ 0.9631467 ],
  ...,
  [10.193302  ],
  [10.193302  ],
  [10.193302  ]],

 [[ 0.9631467 ],
  [ 0.9631467 ],
  [ 0.9631467 ],
  ...,
  [10.193302  ],
  [10.193302  ],
  [10.193302  ]]],


[[[ 1.5249822 ],
  [ 1.5249822 ],
  [ 1.4045888 ],
  ...,
  [ 0.20065555],
  [ 0.12039334],
  [ 0.12039334]],

 [[ 1.44472   ],
  [ 1.6052444 ],
  [ 1.44472   ],
  ...,
  [ 0.20065555],
  [ 0.12039334],
  [ 0.12039334]],

 [[ 1.44472   ],
  [ 1.44472   ],
  [ 1.5249822 ],
  ...,
  [ 0.20065555],
  [ 0.16052444],
  [ 0.20065555]],

 ...,

 [[ 1.0032778 ],
  [ 1.0032778 ],
  [ 1.0032778 ],
  ...,
  [10.193302  ],
  [10.193302  ],
  [10.193302  ]],

 [[ 1.0032778 ],
  [ 1.0032778 ],
  [ 0.9631467 ],
  ...,
  [10.193302  ],
  [10.193302  ],
  [10.193302  ]],

 [[ 1.0032778 ],
  [ 0.9631467 ],
  [ 0.9631467 ],
  ...,
  [10.193302  ],
  [10.193302  ],
  [10.193302  ]]]],
[[[[ 1.347844  ],
   [ 1.3842722 ],
   [ 1.4935569 ],
   ...,
   [ 0.40071037],
   [ 0.36428216],
   [ 0.36428216]],

  [[ 1.3114158 ],
   [ 1.4207004 ],
   [ 1.4207004 ],
   ...,
   [ 0.40071037],
   [ 0.36428216],
   [ 0.36428216]],

  [[ 1.4935569 ],
   [ 1.4935569 ],
   [ 1.5299851 ],
   ...,
   [ 0.40071037],
   [ 0.36428216],
   [ 0.36428216]],

  ...,

  [[ 1.0564183 ],
   [ 1.0564183 ],
   [ 1.0564183 ],
   ...,
   [ 0.03642822],
   [ 0.03642822],
   [ 9.289195  ]],

  [[ 1.0564183 ],
   [ 1.0564183 ],
   [ 1.0564183 ],
   ...,
   [ 0.14571287],
   [ 0.14571287],
   [ 0.10928465]],

  [[ 1.0564183 ],
   [ 1.0564183 ],
   [ 1.0564183 ],
   ...,
   [ 0.10928465],
   [ 0.10928465],
   [ 0.07285643]]],
 [[[ 1.347844  ],
   [ 1.4935569 ],
   [ 1.4935569 ],
   ...,
   [ 0.36428216],
   [ 0.29142573],
   [ 0.29142573]],

  [[ 1.3842722 ],
   [ 1.4207004 ],
   [ 1.4935569 ],
   ...,
   [ 0.36428216],
   [ 0.29142573],
   [ 0.29142573]],

  [[ 1.4207004 ],
   [ 1.4207004 ],
   [ 1.4935569 ],
   ...,
   [ 0.32785395],
   [ 0.29142573],
   [ 0.29142573]],

  ...,

  [[ 1.0928465 ],
   [ 1.0928465 ],
   [ 1.0928465 ],
   ...,
   [ 0.        ],
   [ 0.        ],
   [ 9.252767  ]],

  [[ 1.0564183 ],
   [ 1.0564183 ],
   [ 1.0564183 ],
   ...,
   [ 0.03642822],
   [ 0.07285643],
   [ 0.07285643]],

  [[ 1.0564183 ],
   [ 1.0564183 ],
   [ 1.0564183 ],
   ...,
   [ 0.03642822],
   [ 0.10928465],
   [ 0.10928465]]],


 [[[ 1.4207004 ],
   [ 1.4935569 ],
   [ 1.4571286 ],
   ...,
   [ 0.36428216],
   [ 0.25499752],
   [ 0.25499752]],

  [[ 1.4935569 ],
   [ 1.3842722 ],
   [ 1.3842722 ],
   ...,
   [ 0.36428216],
   [ 0.25499752],
   [ 0.25499752]],

  [[ 1.4571286 ],
   [ 1.4571286 ],
   [ 1.4935569 ],
   ...,
   [ 0.32785395],
   [ 0.29142573],
   [ 0.29142573]],

  ...,

  [[ 1.0928465 ],
   [ 1.0928465 ],
   [ 1.0928465 ],
   ...,
   [ 0.07285643],
   [ 0.07285643],
   [ 0.03642822]],

  [[ 1.0564183 ],
   [ 1.0564183 ],
   [ 1.0564183 ],
   ...,
   [ 0.07285643],
   [ 0.07285643],
   [ 0.        ]],

  [[ 1.0564183 ],
   [ 1.0564183 ],
   [ 1.0564183 ],
    ...,
    [ 0.10928465],
    [ 0.07285643],
    [ 0.        ]]],


  ...,


  [[[ 1.5299851 ],
    [ 1.4935569 ],
    [ 1.3114158 ],
    ...,
    [ 0.29142573],
    [ 0.2185693 ],
    [ 0.2185693 ]],

   [[ 1.6028415 ],
    [ 1.5664133 ],
    [ 1.3114158 ],
    ...,
    [ 0.29142573],
    [ 0.2185693 ],
    [ 0.2185693 ]],

   [[ 1.4207004 ],
    [ 1.4207004 ],
    [ 1.347844  ],
    ...,
    [ 0.40071037],
    [ 0.36428216],
    [ 0.36428216]],

   ...,

   [[ 1.0564183 ],
    [ 1.0564183 ],
    [ 1.0564183 ],
    ...,
    [ 0.10928465],
    [ 0.10928465],
    [ 0.10928465]],

   [[ 1.0564183 ],
    [ 1.0564183 ],
    [ 1.0564183 ],
    ...,
    [ 0.18214108],
    [ 0.14571287],
imageio.mimsave('./animation.gif', val[0][0], fps=10)
```

```
plt.imshow(val[0][0][35])
```

```
<matplotlib.image.AxesImage at 0x796714f0d5d0>
```



```
tf.strings.reduce_join([num_to_char(word) for word in val[1][0]])
```

```
<tf.Tensor: shape=(), dtype=string, numpy=b'bin green in a five again'>
```

3. Design the Deep Neural Network *italicised text*

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv3D, LSTM, Dense, Dropout,
Bidirectional, MaxPool3D, Activation, Reshape, SpatialDropout3D,
BatchNormalization, TimeDistributed, Flatten
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint,
LearningRateScheduler
```

```
data.as_numpy_iterator().next()[0][0].shape
```

```
(75, 46, 140, 1)
```

```
model = Sequential()
model.add(Conv3D(128, 3, input_shape=(75,46,140,1), padding='same'))
model.add(Activation('relu'))
model.add(MaxPool3D((1,2,2)))

model.add(Conv3D(256, 3, padding='same'))
model.add(Activation('relu'))
model.add(MaxPool3D((1,2,2)))
model.add(Conv3D(75, 3, padding='same'))
model.add(Activation('relu'))
model.add(MaxPool3D((1,2,2)))

model.add(TimeDistributed(Flatten()))

model.add(Bidirectional(LSTM(128, kernel_initializer='Orthogonal',
return_sequences=True)))
model.add(Dropout(.5))

model.add(Bidirectional(LSTM(128, kernel_initializer='Orthogonal',
return_sequences=True)))
model.add(Dropout(.5))

model.add(Dense(char_to_num.vocabulary_size()+1,
kernel_initializer='he_normal', activation='softmax'))


model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv3d (Conv3D)             (None, 75, 46, 140, 128)  3584

 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)   884992

 activation_1 (Activation)   (None, 75, 23, 70, 256)   0

 max_pooling3d_1 (MaxPooling  (None, 75, 11, 35, 256)  0
 3D)

 conv3d_2 (Conv3D)           (None, 75, 11, 35, 75)    518475

 activation_2 (Activation)   (None, 75, 11, 35, 75)    0

 max_pooling3d_2 (MaxPooling  (None, 75, 5, 17, 75)    0
 3D)

 time_distributed (TimeDistr  (None, 75, 6375)         0
 ibuted)

 bidirectional (Bidirectiona  (None, 75, 256)          6660096
 l)
 dropout (Dropout)            (None, 75, 256)          0

 bidirectional_1 (Bidirectio  (None, 75, 256)          394240
 nal)

 dropout_1 (Dropout)          (None, 75, 256)          0

 dense (Dense)                (None, 75, 41)           10537

=================================================================
Total params: 8,471,924
Trainable params: 8,471,924
Non-trainable params: 0
_________________________________________________________________
```

```
yhat = model.predict(val[0])
```

```
1/1 [==============================] - 16s 16s/step
```

```
tf.strings.reduce_join([num_to_char(x) for x in
tf.argmax(yhat[0],axis=1)])
```

```
<tf.Tensor: shape=(), dtype=string, numpy=b'hnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn'>
```

```
tf.strings.reduce_join([num_to_char(tf.argmax(x)) for x in yhat[0]])
```

```
<tf.Tensor: shape=(), dtype=string, numpy=b'hnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn'>
```

```
model.input_shape
```

```
(None, 75, 46, 140, 1)
```

```
model.output_shape
```

```
(None, 75, 41)
```

4. Setup Training Options and Train

```
def scheduler(epoch, lr):
    if epoch < 30:
        return lr
    else:
        return lr * tf.math.exp(-0.1)
```

```
def CTCLoss(y_true, y_pred):
    batch_len = tf.cast(tf.shape(y_true)[0], dtype="int64")
    input_length = tf.cast(tf.shape(y_pred)[1], dtype="int64")
    label_length = tf.cast(tf.shape(y_true)[1], dtype="int64")

    input_length = input_length * tf.ones(shape=(batch_len, 1),
dtype="int64")
    label_length = label_length * tf.ones(shape=(batch_len, 1),
dtype="int64")

    loss = tf.keras.backend.ctc_batch_cost(y_true, y_pred, input_length,
label_length)
    return loss


class ProduceExample(tf.keras.callbacks.Callback):
    def __init__(self, dataset) -> None:
        self.dataset = dataset.as_numpy_iterator()

    def on_epoch_end(self, epoch, logs=None) -> None:
        data = self.dataset.next()
        yhat = self.model.predict(data[0])
        decoded = tf.keras.backend.ctc_decode(yhat, [75,75],
greedy=False)[0][0].numpy()
        for x in range(len(yhat)):
            print('Original:',
tf.strings.reduce_join(num_to_char(data[1][x])).numpy().decode('utf-8'))
            print('Prediction:',
tf.strings.reduce_join(num_to_char(decoded[x])).numpy().decode('utf-8'))
            print('~'*100)



model.compile(optimizer=Adam(learning_rate=0.0001), loss=CTCLoss)



checkpoint_callback = ModelCheckpoint(os.path.join('models','checkpoint'),
monitor='loss', save_weights_only=True)



schedule_callback = LearningRateScheduler(scheduler)



example_callback = ProduceExample(test)



# model.fit(train, validation_data=test, epochs=100,
callbacks=[checkpoint_callback, schedule_callback, example_callback])
```

5. Make a Prediction

```
url = 'https://drive.google.com/uc?id=1vWscXs4Vt0a_1IH1-ct2TCgXAZT-N3_Y'
output = 'checkpoints.zip'
gdown.download(url, output, quiet=False)
gdown.extractall('checkpoints.zip', 'models')
```

```
Access denied with the following error:
        Cannot retrieve the public link of the file. You may need to change
        the permission to 'Anyone with the link', or have had many accesses.

You may still be able to access the file from the browser:

         https://drive.google.com/uc?id=1vWscXs4Vt0a_1IH1-ct2TCgXAZT-N3_Y

['models/checkpoint.index',
 'models/__MACOSX/._checkpoint.index',
 'models/checkpoint.data-00000-of-00001',
 'models/__MACOSX/._checkpoint.data-00000-of-00001',
 'models/checkpoint',
 'models/__MACOSX/._checkpoint']
```

```
model.load_weights('models/checkpoint')
```

```
<tensorflow.python.checkpoint.checkpoint.CheckpointLoadStatus at 0x7967140825
60>
```

```
test_data = test.as_numpy_iterator()


 # sample = test_data.next()


 # yhat = model.predict(sample[0])


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


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


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

`# This is formatted as code`

Test on a Video

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


print('~'*100, 'REAL TEXT')
#[tf.strings.reduce_join([num_to_char(tf.cast(word, tf.int32)) for word in
sentence]) for sentence in [sample[1]]]
[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 [==============================] - 6s 6s/step

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


decoded_sentences = [tf.strings.reduce_join([num_to_char(word) for word in
sentence]) for sentence in decoded]

# Convert tensors to strings
decoded_sentences = [sentence.numpy().decode('utf-8') for sentence in
decoded_sentences]

# Print the separator and label
print('~' * 100, 'PREDICTIONS')

# Print the decoded sentences
for sentence in decoded_sentences:
    print(sentence)

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~ PREDICTIONS
bin red at s nine again

from tensorflow.keras.models import load_model
model.save("/content/drive/MyDrive/Mycheckpoint2.h5")
```