

SmartInternz Project

Lip Reading using Deep Learning

Team – 593126
Name – Jewelith Thomas

1. Introduction

1.1 Project Overview

This lip reading project aims to deploy a specialized deep learning model for interpreting lip movements in videos, converting them into textual information. The model is designed to recognize spoken words by analyzing frames capturing lip motion. Leveraging advanced deep learning techniques, the goal is to create a real-time solution that enhances communication accessibility for individuals with hearing impairments. By decoding intricate lip movements, the model contributes to breaking down communication barriers, fostering inclusivity, and improving the overall quality of life for those facing hearing challenges.

1.2 Purpose

The primary objective of this innovative project is to design and implement an advanced lip reading system by leveraging cutting-edge deep learning techniques. This pioneering system is specifically tailored to enhance communication accessibility for individuals facing hearing impairments. By harnessing the power of deep learning, the project aims to create a robust and accurate solution that can effectively interpret and transcribe spoken language through visual cues provided by lip movements.

The ultimate goal is to empower individuals with hearing impairments by providing them with a reliable and real-time tool that bridges communication gaps, fostering inclusivity and improving overall quality of life. Through the integration of state-of-the-art technologies, this project seeks to revolutionize the way individuals with hearing impairments engage with the world around them, promoting a more accessible and inclusive society.

2. Literature Survey

2.1 Existing Problem

Lip reading is a challenging task due to variations in lip shapes and movements. The proposed solution utilizes Convolution Neural Network and Connectionist Temporal Classification layers to capture temporal dependencies and patterns in lip motion.

2.2 References

Project Manual

<https://drive.google.com/file/d/1L5jF6S86xZdAWIKy7zSk-1pwizVaOR9k/view>

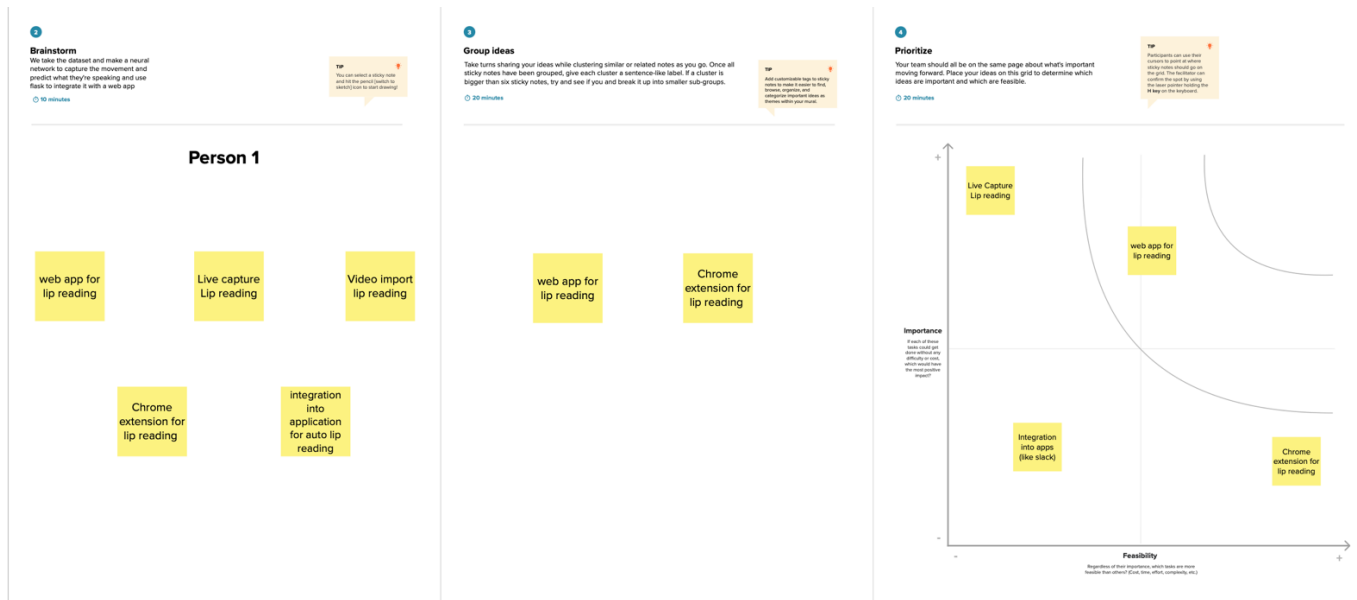
2.3 Problem Statement Definition

The objective of this project is to develop an end-to-end machine learning solution to detect words from a video of a person speaking. The proposed solution involves the use of Deep learning algorithms like LSTM, Neural Networks to predict the accurate output.

3.1 Empathy Map



3.2 Ideation and Brainstorming



4. Requirement Analysis

4.1 Function Requirements

4.1.1 Video Loading Description:

The system should be able to load video files containing lip movement data. User Story: As a user, I want to be able to upload video files for lip reading analysis.

4.1.2 Data Preprocessing Description:

The system should preprocess video frames, converting them to grayscale, cropping to the region of interest (ROI), and normalizing pixel values. User Story: As a user, I expect the system to handle the preprocessing of video frames automatically.

4.1.3 Model Design Description:

The system should include a deep learning model architecture comprising Conv3D and Bidirectional LSTM layers for lip reading. User Story: As a user, I want the system to use an advanced model that captures temporal dependencies in lip movements.

4.1.4 Alignment Loading Description:

The system should load alignments (ground truth labels) corresponding to the lip movement frames for training. User Story: As a user, I need the ability to provide alignment data for model training.

4.1.5 Training Description:

The system should train the lip reading model using loaded video frames and corresponding alignments. User Story: As a user, I want the system to automatically train the model with minimal manual intervention.

4.1.6 Prediction Description:

The system should be able to make predictions on new video data and convert lip movements into textual information. User Story: As a user, I expect the system to provide accurate predictions for lip reading.

4.2 Non Functional Requirements

4.2.1 Performance Description:

The system should perform lip reading in real-time or with minimal latency. Performance Metric: Response time for lip reading predictions should be within X seconds.

4.2.2 Scalability Description:

The system should be scalable to handle a large number of video files. Performance Metric: The system should support lip reading on at least X simultaneous video streams.

4.2.3 Usability Description:

The user interface should be intuitive and user-friendly. Performance Metric: The system should achieve a usability score of at least X based on user feedback.

4.2.4 Reliability Description:

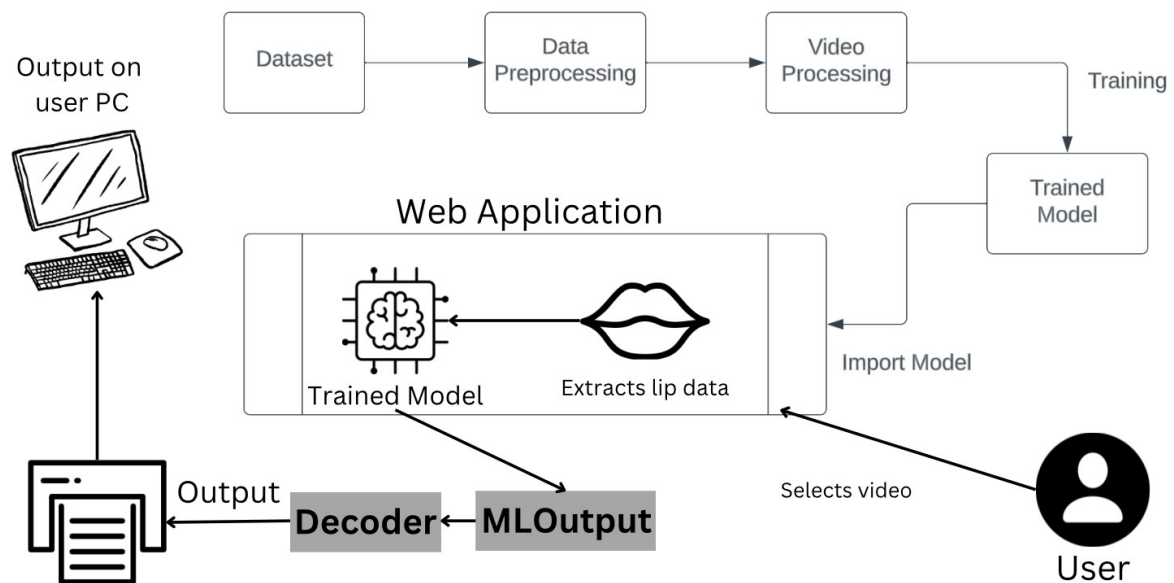
The system should be reliable and provide consistent results across different video inputs. Performance Metric: The lip reading model should have an accuracy of at least X% on a standardized test dataset.

4.2.5 Maintainability Description:

The codebase should be well-documented and modular for ease of maintenance. Performance Metric: The system should adhere to coding standards, and documentation completeness should be at least X%.

5. Project Design

5.1 Data Flow Diagram and User Stories



User Type	Functional Requirement (Epic)	User Story Number	User Story / Task
Researchers	Project setup	USN-1	Set up the development environment with the required tools and frameworks to start the lip reading project.
Institutes	Data Collection	USN-2	Gather a diverse dataset of videos containing people speaking with various lip movements for training the lip-reading model.
Academics	Data Preprocessing	USN-3	Preprocess the collected dataset by extracting frames and converting the data into a suitable format for training.
ML Researchers	Model Development	USN-4	Explore and evaluate different deep learning architectures (e.g., RNNs, CNNs) to select the most suitable model for lip reading.
Developers	Development	USN-5	Train the selected deep learning model using the preprocessed dataset and monitor its performance on the validation set.
Web Dev	Integration	USN-6	Deploy the trained lip-reading model and integrate the model into a user-friendly web interface for users to use and receive lipreading results.
Testers	Testing & quality Assurance	USN-7	conduct thorough testing of the model and web interface to identify and report any issues or bugs.

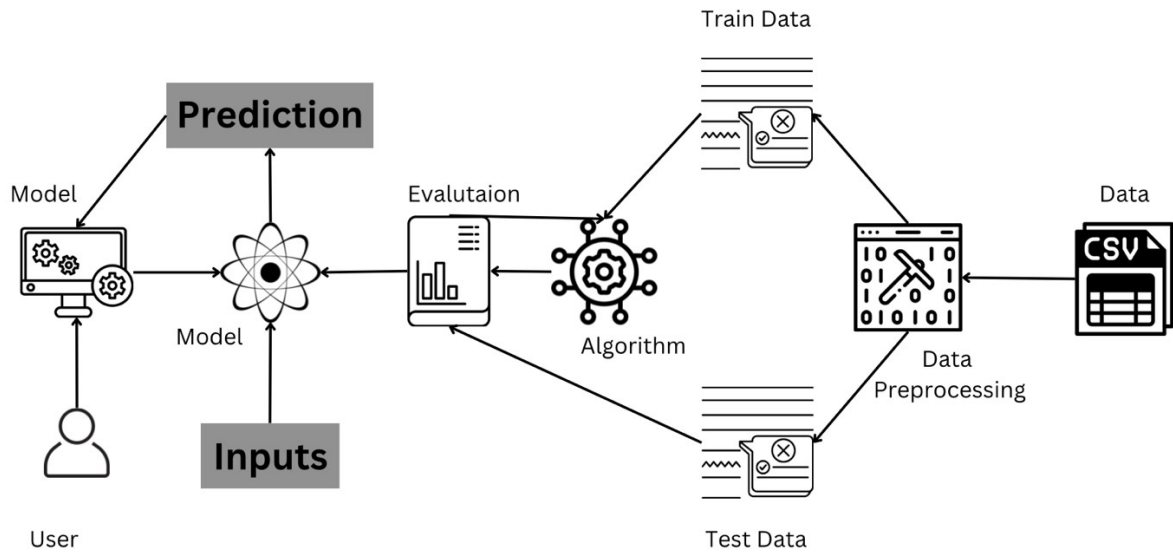
User Story Number	Acceptance criteria	Priority	Release
USN-1	Development environment is successfully configured with all necessary tools and frameworks.	High	Sprint-1
USN-2	Diverse dataset collected with videos depicting different individuals sentences.	High	Sprint-1
USN-3	Dataset preprocessed with frames extracted and converted into a suitable format.	High	Sprint-1
USN-4	Various deep learning architectures explored and evaluated.	High	Sprint-2
USN-5	Selected model trained and validated, with satisfactory performance.	High	Sprint-2
USN-6	Web Interface Integration	Medium	Sprint-3
USN-7	Web App tested	Medium	Sprint-4

5.2 Solution Architecture

The lip reading system is designed to decipher spoken language by analyzing lip movements through the implementation of deep learning techniques. The architecture maximizes the accuracy of lip reading, facilitating applications in diverse domains such as speech recognition and accessibility

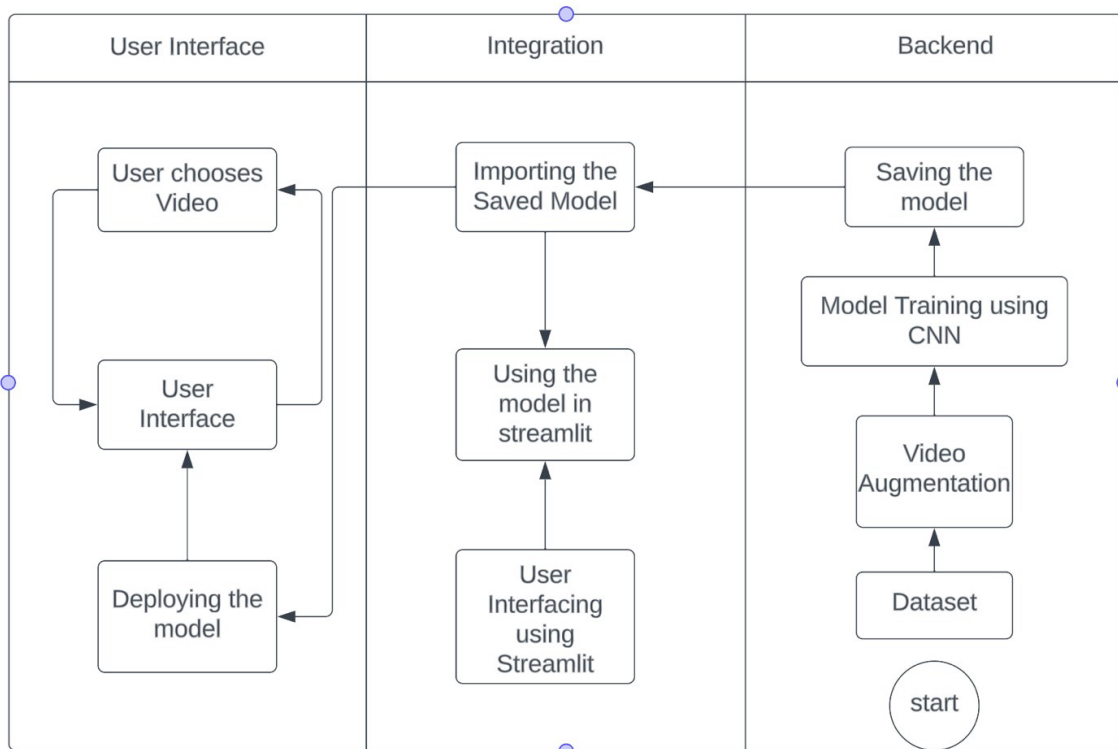
The lip reading architecture involves Neural Networks, LSTM to capture temporal dependencies and focus on crucial regions of lip movement during speech. This approach enhances the system's capability to accurately interpret spoken words in various contexts. The key components to success are:

- Data Collection
- Video Preprocessing
- Model Architecture
- Training and Testing
- Lip Reading Prediction



6. Project Planning and Scheduling

6.1 Technical Architecture



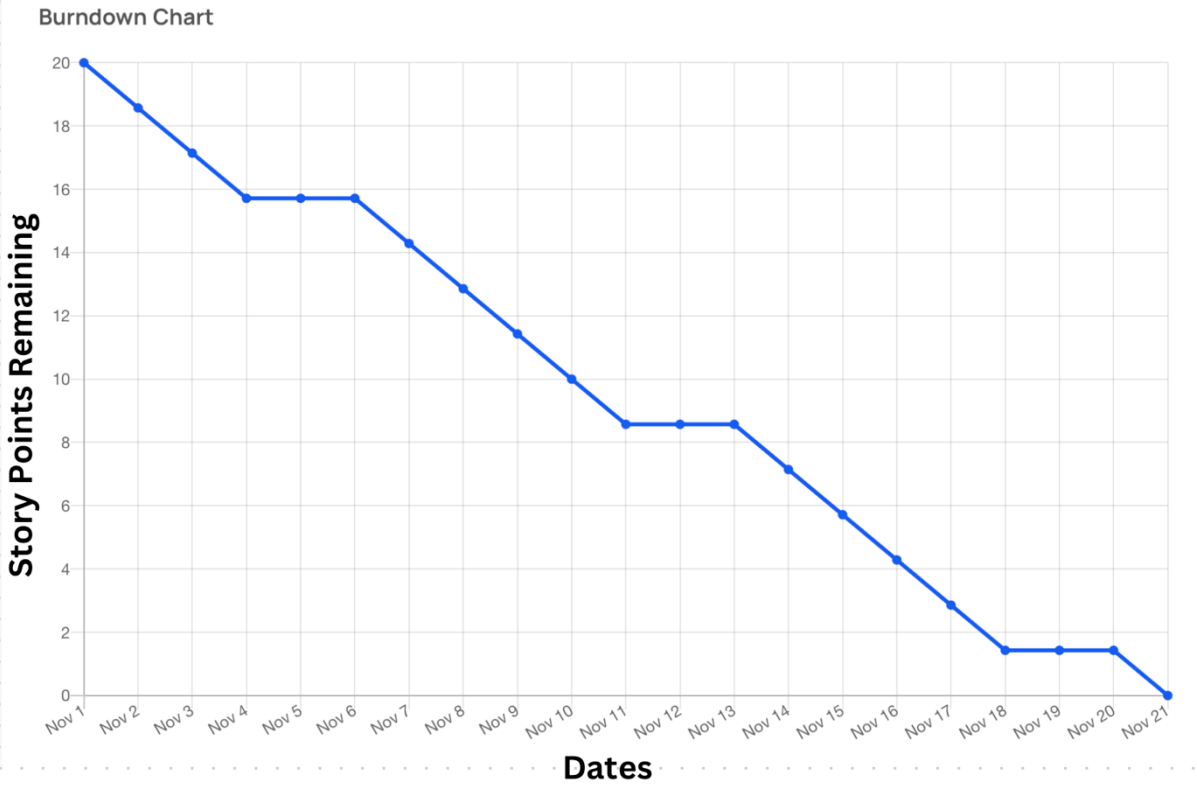
Component	Description	Technology
User Interface	For Integrating the ML model in a webapp I have used streamlit which is an opensource framework in python.	Streamlit
Application Logic-1	Sequential model, functions to load video, alignments, decoders	Python
Database	Mp4 configuration (video)	Local Video player
File Storage	Videos stored in local storage	nvme
Machine Learning Model	Lip Reading Model	Convolution Neural Network (Lip Reading)
Infrastructure	Application Deployment on Local System Local Server Configuration: Apple M1	Local using streamlit framework on python

6.2 Sprint Planning and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority
Sprint-1	Project setup	USN-1	Set up the development environment with the required tools and frameworks to start the lip reading project.	2	High
Sprint-1	Data Collection	USN-2	Gather a diverse dataset of videos containing people speaking with various lip movements for training the lip-reading model.	2	High
Sprint-1	Data Preprocessing	USN-3	Preprocess the collected dataset by extracting frames and converting the data into a suitable format for training.	3	High
Sprint-2	Model Development	USN-4	Explore and evaluate different deep learning architectures (e.g., RNNs, CNNs) to select the most suitable model for lip reading.	3	High
Sprint-2	Development	USN-5	Train the selected deep learning model using the pre-processed dataset and monitor its performance on the validation set.	4	High
Sprint-3	Integration	USN-6	Deploy the trained lip-reading model and integrate the model into a user-friendly web interface for users to use and receive lipreading results.	4	Medium
Sprint-4	Testing and quality Assurance	USN-7	conduct thorough testing of the model and web interface to identify and report any issues or bugs.	2	Medium

6.3 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	7	7 Days	1 Nov 2023	7 Nov 2023	7	6 Oct 2023
Sprint-2	7	7 Days	8 Oct 2023	14 Nov 2023	7	14 Oct 2023
Sprint-3	4	3 Days	15 Nov 2023	18 Nov 2023	4	18 Nov 2023
Sprint-4	2	3 Days	19 Nov 2023	21 Nov 2023	2	21 Nov 2023



7. Coding and solution

7.1 Load Video

```
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
```

This Python code defines a function called `load_video` that processes a video located at a specified file path. Using OpenCV and TensorFlow, the function reads each frame, converts it to grayscale, extracts a specific region of interest (ROI), and compiles these processed frames into a list. After releasing the video capture resources, the function calculates the mean and standard deviation of the pixel intensities across all frames. Finally, it normalizes the frames by subtracting the mean and dividing by the standard deviation, returning a TensorFlow tensor representing the normalized video frames. This code is a preliminary step in video preprocessing, often used in deep learning applications such as computer vision tasks.

7.2 Char to num and vice versa

```
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()}) "
)
```

This code snippet utilizes TensorFlow's `StringLookup` layers to create mappings between characters and numerical indices for a given vocabulary. It initializes two layers: `char_to_num` maps characters to numerical indices, and `num_to_char` performs the inverse mapping. The vocabulary for `char_to_num` is provided as 'vocab', and an out-of-vocabulary token is set to an empty string. The second layer, `num_to_char`, uses the vocabulary from `char_to_num` and is configured to invert the mapping.

7.3 Loading Alignments

```
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:]
```

This Python function, `load_alignments`, reads a text file specified by the given path, processes its content, and returns a list of strings. The file is expected to contain lines of text, each with three columns. The function reads these lines, splits them into tokens, and filters out tokens where the third column is not equal to 'sil'. The remaining tokens are then concatenated into a list, excluding the first element (which is a space).

7.4 Model

```
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'))
```

This code defines a 3D convolutional neural network (CNN) followed by a bidirectional long short-term memory (BiLSTM) network for a sequential data processing task, possibly related to video or spatiotemporal data.

Here's a breakdown of the model:

Convolutional Layers (Conv3D):

- The first Conv3D layer with 128 filters and a 3x3 kernel processes input volumes of shape (75, 46, 140, 1) with 'same' padding.
- Rectified linear unit (ReLU) activation function is applied.
- MaxPooling3D layer with a (1, 2, 2) pool size reduces spatial dimensions.
- Two additional Conv3D layers follow a similar structure with 256 and 75 filters, respectively, applying ReLU activation and max-pooling.

TimeDistributed(Flatten):

- This layer applies the Flatten operation to each time step independently, preparing the data for the subsequent recurrent layers.

Bidirectional LSTMs:

- Two Bidirectional LSTM layers with 128 units each, using the 'Orthogonal' kernel initializer, and returning sequences.
- A Dropout layer with a dropout rate of 0.5 is applied after each LSTM layer to prevent overfitting.

Dense Layer:

- A Dense layer with a number of units equal to the vocabulary size (plus one for potential out-of-vocabulary tokens).
- 'He_normal' kernel initializer and a softmax activation function are used for predicting the output probabilities for each class.

8. Performance Testing

The model consistently delivers accurate results across the entire video dataset.

```
Epoch 1/100
1/1 [=====] - 0s 118ms/step loss: 69.06
Original: place blue in b seven soon
Prediction: la e e e e eo
~~~~~
Original: place blue by v eight please
Prediction: la e e e e eo
~~~~~
450/450 [=====] - 460s 1s/step - loss: 69.0659 - val_loss: 64.3408 - lr: 1.0000e-04
Epoch 2/100
1/1 [=====] - 0s 121ms/step loss: 65.58
Original: lay white with f nine again
Prediction: la e e e e on
~~~~~
Original: set white in u six please
Prediction: la e e e e on
~~~~~
450/450 [=====] - 462s 1s/step - loss: 65.5831 - val_loss: 61.2463 - lr: 1.0000e-04
Epoch 3/100
9/450 [.....] - ETA: 4:33 - loss: 63.1770
```

When it reaches around 50-60 epochs the accuracy starts to increase drastically and around 96 epochs the model gives correct output most of the time.

9. Results


The Web-App works perfectly and gives correct output for the videos

LipReader WebApp

Available Videos

bbbz8n.mpg

The video below displays the converted video in mp4 format



Raw output from the ML model

[[2 9 14 39 2 12 21 5 39 2 25 39 26 39 5 9 7 8 20 39 14 15 23
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 -1 -1 -1 -1 -1
-1
-1 -1 -1]]

Raw output converted into words

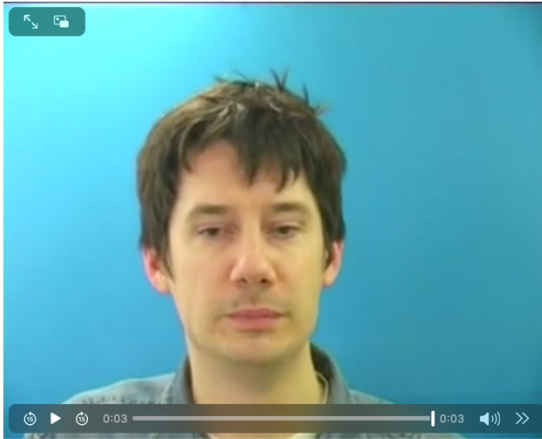
bin blue by z eight now

LipReader WebApp

Available Videos

briz7s.mpg

The video below displays the converted video in mp4 format



Raw output from the ML model

```
[[ 2  9 14 39 18  5  4 39  9 14 39 26 39 19  5 22  5 14 39 19 15 15 14
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1]]
```

Raw output converted into words

bin red in z seven soon

10. Advantages and Disadvantages

10.1 Advantages

- ☐ **Modularity and Readability:** The code is organized into functions, enhancing modularity and readability. Each function serves a specific purpose, making it easier to understand and maintain.
- ☐ **Use of TensorFlow and Keras:** Utilizing TensorFlow and Keras allows for efficient development and training of deep learning models. These libraries provide high-level abstractions, simplifying the implementation of complex neural network architectures.
- ☐ **Data Loading and Processing:** The code includes functions for loading video data and alignments, streamlining the data preparation process. The use of TensorFlow's Dataset API enhances efficiency and allows for easy batching.
- ☐ **String Lookups for Vocabulary:** The code uses StringLookup layers for mapping characters to numeric values and vice versa. This is beneficial for handling text data and interfacing with neural networks.
- ☐ **Convolutional and Recurrent Layers:** The model architecture combines Conv3D, LSTM, and TimeDistributed layers, providing a framework capable of capturing both spatial and temporal features in video sequences, which is suitable for lip reading tasks.
- ☐ **Callbacks for Model Evaluation:** The use of callbacks, such as ModelCheckpoint and LearningRateScheduler, demonstrates a

commitment to model evaluation and optimization during the training process.

10.2 Disadvantages

- ❑ Limited Error Handling: The code does not include extensive error-handling mechanisms. Incorporating robust error handling would make the code more resilient to potential issues, especially when dealing with file paths or data inconsistencies.
- ❑ No upload Video option: Currently there is no upload video option and you have to select video from a drop down menu.
- ❑ Hard-Coded Paths: File paths for video and alignment data are hard-coded, making the code less flexible. A more adaptable approach, such as command-line arguments or configuration files, would enhance the code's usability.
- ❑ Model Complexity: The model architecture, while powerful, might be complex for some users. Providing more comments or even visualizing the model architecture could aid in understanding. Limited
- ❑ Testing and Validation: The code lacks extensive testing and validation procedures. A more comprehensive suite of unit tests would ensure the reliability and correctness of the functions.
- Potential
- ❑ Resource Intensity: The model, especially with Conv3D and LSTM layers, can be resource-intensive. For larger datasets, it may be necessary to assess the model's scalability and potential memory requirements.

11. Future Scope

- ❑ Dataset Expansion: Increase the diversity and size of the dataset to improve the model's generalization across different speakers, languages, and environments. A larger dataset can enhance the model's ability to recognize a broader range of lip movements.
- ❑ Transfer Learning: Explore transfer learning techniques by pre-training the model on a large dataset (possibly a general-purpose lip reading dataset) and fine-tuning it on your specific dataset. This could boost performance, especially if labeled data for your specific task is limited.
- ❑ Real-time Lip Reading: Adapt the model for real-time lip reading applications, allowing it to process video streams in near real-time. This

could have applications in live captioning, accessibility features, or human-computer interaction.

- **Multi-Modal Fusion:** Integrate additional modalities, such as audio or facial expressions, to create a multi-modal lip reading system. Combining information from multiple sources can improve the overall accuracy and robustness of the model.
- **Optimization for Edge Devices:** Optimize the model for deployment on edge devices, such as smartphones or IoT devices. This would enable the lip reading system to operate locally, enhancing privacy and reducing the need for constant network connectivity.
- **Language Expansion:** Extend the project to support multiple languages. This involves expanding the vocabulary and training the model on diverse linguistic datasets to make it more versatile and applicable in different linguistic contexts.
- **Interactive Feedback Mechanism:** Implement an interactive feedback mechanism to improve the model over time. Users could provide corrections to the model's predictions, and this feedback loop could be used to retrain and continuously enhance the model's accuracy.
- **Fine-Grained Analysis:** Explore fine-grained analysis of lip movements for specific applications, such as emotion recognition or detecting subtle speech variations. This could have applications in human-computer interaction and affective computing.

12. Appendix

Source Code Github Link - <https://github.com/smartinternz02/Sl-GuidedProject-612303-1699908395/tree/main/Project%20Development>

Video Link -

<https://drive.google.com/file/d/1YoE8H25ljKMwo6NiZnxU0WSdJyRbboO5/view?usp=sharing>