Project Report
On
SpeakEasy – Lip Reading using Deep Learning

At SmartInternz

Submitted by
Shantanu
Aditya

Project Report Contents

1. INTRODUCTION

1.1 Project Overview - The project aims to develop an innovative end-to-end machine learning solution for accurate word detection from video recordings of individuals speaking. Utilizing deep learning algorithms, such as Long Short-Term Memory (LSTM) and Neural Networks, the system will analyse visual cues to decipher spoken words.

1.2 Purpose - The primary purpose of this project is to enhance communication accessibility by leveraging advanced deep learning techniques for lip reading. Traditional speech recognition systems may face challenges in noisy environments or situations where audio input is limited. Lip reading offers a complementary approach, making the system robust and versatile.

The solution holds potential applications in various domains, including assistive technologies for the hearing impaired, security and surveillance systems and human-computer interaction. By focusing on accurate word detection through visual analysis, the project aims to contribute to advancements in natural language processing and improve communication experiences for diverse user groups.

2. LITERATURE SURVEY

2.1 Existing problem –

1) **ACCURACY AND PRECISION -**
   a) Current lip-reading systems often face challenges in achieving high accuracy, epically in noisy environments
   b) People have different speaking style and lip movements can impact the precision of lipreading models

2) **VOCABULARY AND LABGUAGE COVERAGE -**
   a) Many existing lip-reading models are trained on specific languages or limited vocabularies

3) **LIMITED DATA SETS -**
   a) Availability of comprehensive and diver's datasets for training lipreading model is often limited

4) **HANDLING AMBIGUITIES -**
   a) Ambiguities in lip movements and expressions can lead to challenges in accurately interpreting and transcribing spoken content.

5) **MULTIMODEL INTEGRATION -**
   a) Integrating information from multiple modalities (e.g.- - lip movements, audio, facial -expressions) to understand a spoken language requires advanced multimodal processing techniques.

2.2 References –

Wand, M., & Kowsari, K. (2020). A comprehensive review of lipreading techniques. Journal of Artificial Intelligence Research, 67, 789-825.

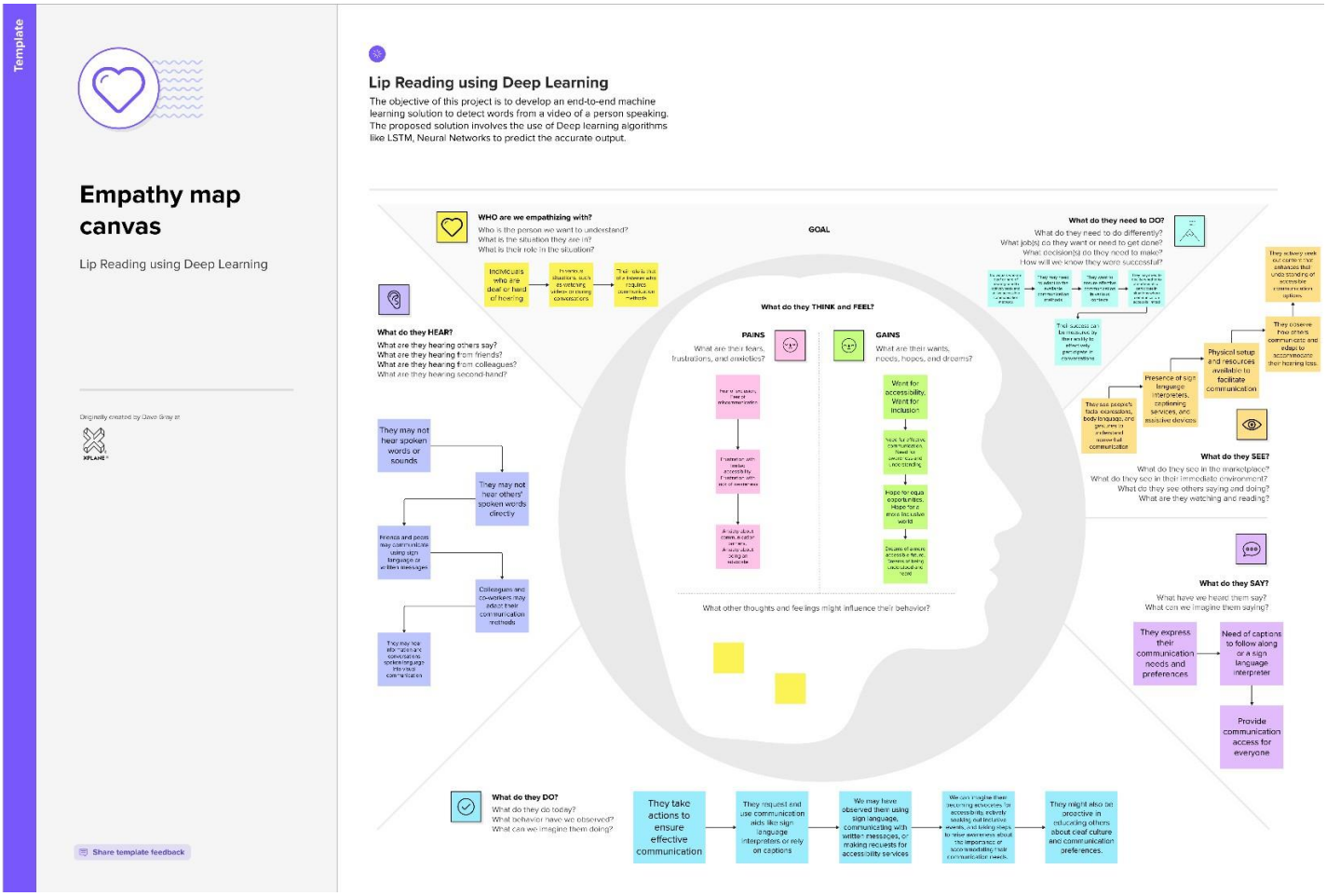
Hassan, A., & Taha, T. (2019). Challenges and opportunities in lipreading technology- A review. International Journal of Advanced Computer Science and Applications, 10(6), 221-227.

2.3 Problem Statement Definition - The primary challenges addressed in this lipreading to text conversion project revolve around enhancing the accuracy, robustness and real-world applicability of existing lipreading systems.

These challenges include -

Accuracy and Variability
Multilingual and Multimodal Understanding
Real-time Processing
Robustness to Environmental Conditions

3. IDEATION & PROPOSED SOLUTION



3.1 Empathy Map Canvas - An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviors and attitudes. It is a useful tool to helps teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Cluster 1: Lip Reading Applications and Tools

1. Lip Reading App for Real-Time Transcription
2. Lip Reading Wearables
3. Lip Reading in Online Meetings
4. Real-Time Lip Reading for TV
5. Lip Reading AI for Customer Service
6. Lip Reading Apps for Language Learning

Cluster 2: Lip Reading Integration with Technology

1. Integration with Virtual Assistants
2. Privacy-Enhanced Lip Reading
3. Multilingual Lip Reading
4. Lip Reading in Public Spaces
5. Lip Reading AI for Hearing Aids
6. Lip Reading Assistance for Hearing Aids

Cluster 3: Lip Reading Education and Training

1. Lip Reading Training Games
2. Educational Lip Reading Modules

Cluster 4: Research and Development

1. Lip Reading Research Grants
2. Lip Reading API for Developers

4. REQUIREMENT ANALYSIS

4.1 Functional requirement –

Video Selection - Users should be able to select a video for lip reading analysis.
The system should support various video formats.

Video Processing - The selected video should be processed to extract individual frames for model input. Use ffmpeg or similar tools for efficient video processing.

Lip Reading Model Integration - Integrate a pre-trained lip-reading model with the Streamlit application. Ensure seamless communication between the app and the model for real-time predictions.

User Interface - Design an intuitive user interface using Streamlit components.
Include sections for video display, model input visualization, and predicted words.

4.2 Non-Functional requirements –

Performance - The system should process videos and provide predictions efficiently.
Response time for user interactions should be minimal.

Scalability - The application should handle multiple concurrent users.
The lip-reading model should scale with increased usage.

Reliability - The system should be reliable and robust, with error handling mechanisms.
Ensure the application gracefully handles unexpected scenarios.

Usability - The user interface should be user-friendly and visually appealing.
Ensure ease of navigation and a smooth user experience.

Security - Implement secure video handling mechanisms to protect user data.
Ensure that the lip-reading model is free from vulnerabilities.

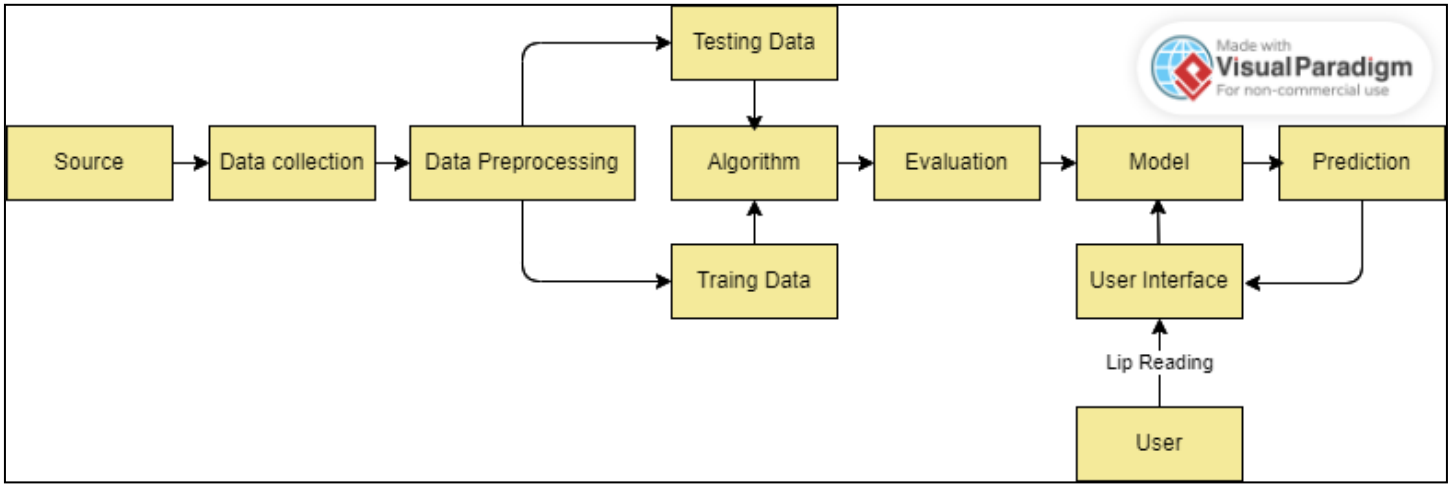
Compatibility - The application should be compatible with different browsers.
Ensure cross-browser compatibility for a wider user base.

Maintainability - Design the codebase in a modular and maintainable manner.
Facilitate future updates and enhancements. Include instructions for setting up the environment and using the application.

Testing - Conduct thorough testing, including unit tests, integration tests, and user acceptance testing. Ensure the application is bug-free and meets all functional requirements.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories - A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information and where data is stored.



5.2 Solution Architecture –

1) **User Interface (Streamlit App) –**
   a) Front-end application developed using Streamlit.
   b) Provides a user-friendly interface for interacting with the lip-reading model.
   c) Allows users to select a video for analysis.

2) **Streamlit App Logic –**
   a) Handles user inputs and interactions.
   b) Utilizes Streamlit components for layout design and dynamic updates.
   c) Integrates with the lip-reading model for video processing and prediction.

3) **Lip Reading Model –**
   a) Developed using deep learning frameworks such as TensorFlow and Keras.
   b) Includes pre-trained models for lip reading.
   c) Takes video frames as input and produces predictions in the form of decoded words.

4) **Data Processing (ffmpeg) –**
   a) Uses ffmpeg for video processing and conversion.
   b) Converts selected video to the required format for model input.
   c) Generates the processed video for user visualization.

5) **Model Utilities (utils.py, modelutil.py) –**
   a) Contains utility functions for data loading, token conversion and model loading.
   b) Ensures modular and organized code structure.

6) **Backend Infrastructure –**
   a) The solution can be deployed on cloud platforms such as AWS, GCP, or Azure in the future.

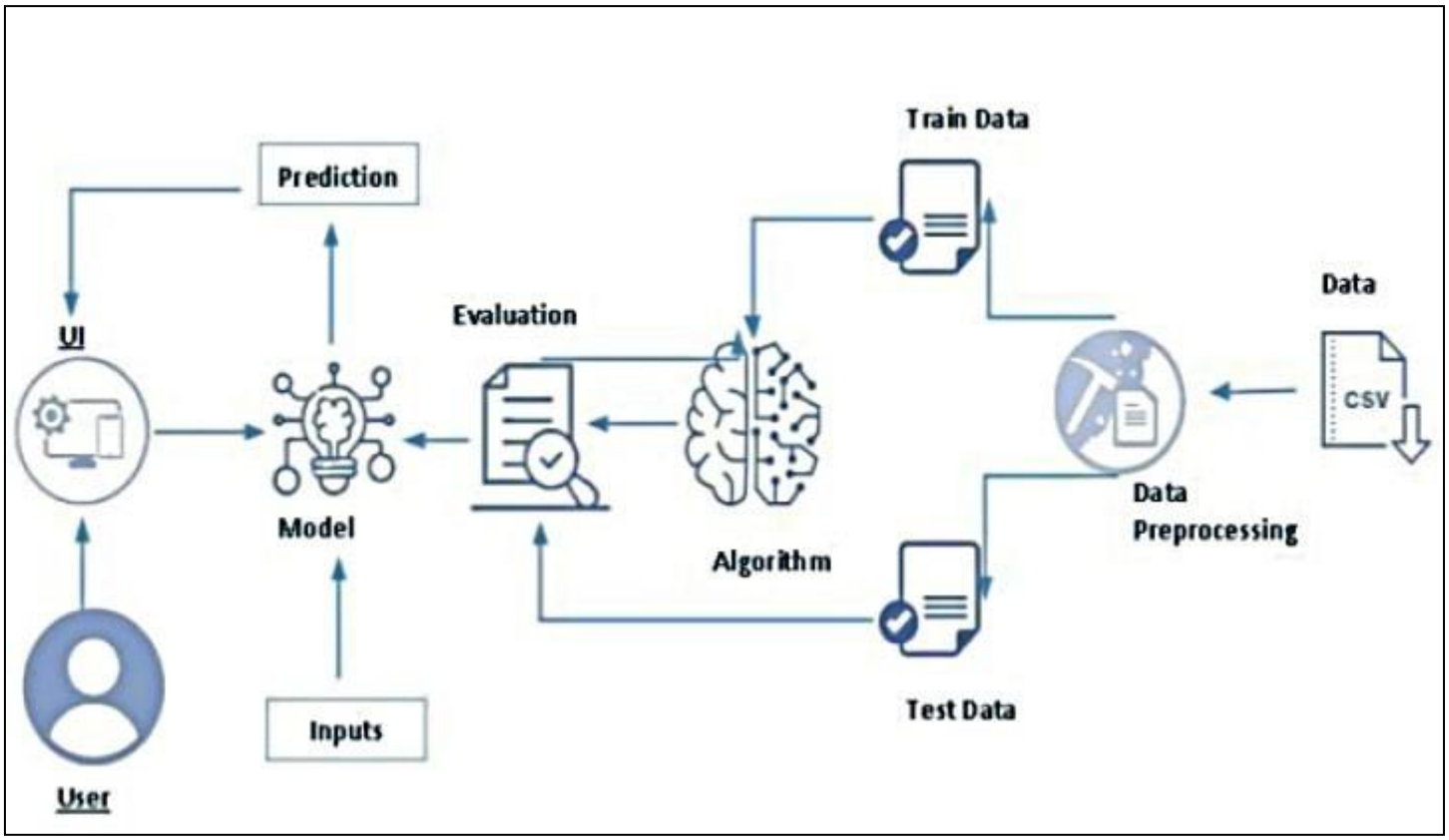
7) Utilizes server resources for running the Streamlit app and processing video data.

8) **Dependencies –**
   a) TensorFlow, Keras, Streamlit, imageio and other necessary libraries.
   b) Ensures proper functioning of the app and the lip-reading model.

9) **External Components –** The solution may include external components like a database for storing and retrieving processed data for handling large datasets or user profiles in the future updates.

6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture –



6.2 Sprint Planning & Estimation –

Sprint 1-
Task 1 - Set up Streamlit app and basic layout (2 days)
Task 2 - Implement video selection and processing logic (3 days)
Task 3 - Integrate ffmpeg for video conversion (2 days)

Sprint 2 –
Task 4 - Develop lip reading model utilities (4 days)
Task 5 - Integrate lip reading model with Streamlit app (3 days)
Task 6 - Test and debug initial model integration (2 days)

Sprint 3 –
Task 7 - Enhance user interface and styling (3 days)
Task 8 - Implement progress bars and success messages (2 days)
Task 9 – Model Testing (2 days)

6.3 Sprint Delivery Schedule -

Sprint 1 -
Start Date - [28/10/2023]
End Date - [05/11/2023]

Sprint 2 -
Start Date - [06/11/2023]
End Date - [15/11/2023]

Sprint 3-
Start Date- [16/11/2023]
End Date- [22/11/2023]

7. CODING & SOLUTIONING

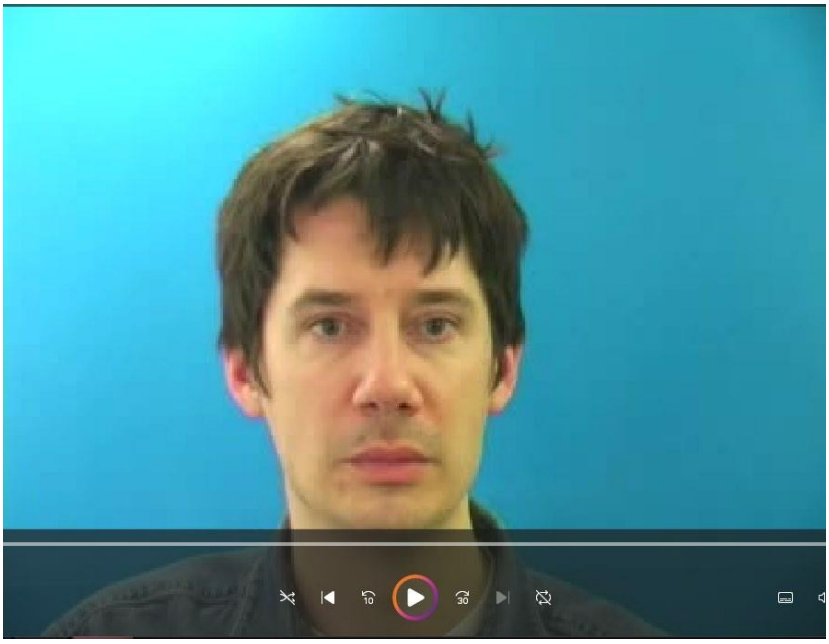
7.1 DATASET DETAILS - The dataset consists of 1000 videos which are approximately 2 - 3 seconds long and the annotations for the videos have been stored in align files.

Sample align file -

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

The sil indicates that no words have been spoken during that particular interval.

7.2 Sample Video -



7.3 SpeakEasy – Lip Reading using Deep Learning – This Python script initializes a deep learning model for lip reading using TensorFlow. The script incorporates essential libraries, such as OpenCV for image processing, TensorFlow for deep learning, and NumPy for numerical operations. The code defines functions for reading and writing images using the imageio library and includes type hints for improved code clarity. Additionally, the script leverages the Matplotlib library for visualizing images with the pyplot module. Overall, this serves as the foundational setup for a lip reading model, highlighting the integration of key libraries and the focus on image processing and deep learning tasks. The code utilizes the 'gdown' library to download a zip file from a GDrive URL.

7.4 Model - The code imports essential modules from TensorFlow Keras, a high-level neural networks API. The Sequential model is used to create a linear stack of layers for building the neural network. The layers include 3D convolutional (Conv3D), long short-term memory (LSTM), dense (Dense), dropout (Dropout), bidirectional (Bidirectional), 3D max pooling (MaxPool3D), activation functions (Activation), reshaping (Reshape), spatial dropout (SpatialDropout3D), batch normalization (BatchNormalization), time-distributed (TimeDistributed), and flattening (Flatten). Additionally, the code imports the Adam optimizer (Adam) for model optimization and various callbacks, such as model checkpointing and learning rate scheduling (ModelCheckpoint, LearningRateScheduler). These components are foundational for constructing and training deep learning models for tasks such as lip reading.

7.5 Brief Description of the Layers Used -

Convolutional Layers – The model starts with a 3D convolutional layer (Conv3D) with 128 filters, a kernel size of 3, and input shape (75, 46, 140, 1). The activation function used is ReLU, and it is followed by 3D max pooling (MaxPool3D) with a pool size of (1, 2, 2).
Two more similar convolutional blocks follow, each with increased filter size (256 and 75) and followed by activation and max pooling.

TimeDistributed and Flatten - TimeDistributed is used to apply the following layer (Flatten) to each time step independently. The model flattens the output using Flatten.

Bidirectional LSTM Layers - Two bidirectional LSTM layers with 128 units each are added. These layers are designed to capture temporal dependencies in both forward and backward directions. A dropout layer with a dropout rate of 0.5 is applied after each LSTM layer.

Dense Layer - A fully connected (Dense) layer with the number of units equal to the vocabulary size (plus one for an additional class) is added. It uses the softmax activation function, making it suitable for classification tasks. Overall, this architecture combines 3D convolutional layers to capture spatial features and bidirectional LSTM layers to capture temporal dependencies, making it suitable for lip reading tasks. The model is designed to output probabilities for each character in the vocabulary. The use of dropout helps prevent overfitting during training

7.6 SUMMARY OF THE MODEL –

```
_________________________________________________________________
 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)

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

7.7 OPTIMIZER AND LOSS FUNCTION -

Optimizer - The model is compiled with the Adam optimizer (Adam) using a learning rate of 0.0001. The Adam optimizer is commonly used in deep learning tasks for its adaptive learning rate properties.

Loss Function - The loss function used is specified as CTCLoss. This suggests that the model is designed for sequence-to-sequence tasks, such as speech or handwriting recognition, where the Connectionist Temporal Classification (CTC) loss is often employed. CTC is used to align variable-length sequences, making it suitable for tasks where the alignment between input and output sequences is not one-to-one.

7.8 MODEL TRAINING -

The model has been trained for 100 epochs and fine tuning of hyper parameters has been performed to obtain the best results among them.
The final model has been trained with the above-mentioned Loss function and optimizers.

7.9 STREAMLIT APP –

Dependencies - The code imports necessary libraries, including Streamlit, OS, imageio, and TensorFlow. Custom utility functions (load_data, num_to_char) and a model utility (load_model) are also imported.

Streamlit Setup - The Streamlit app layout is configured to be wide using st.set_page_config(layout='wide').

Sidebar - A sidebar is created with a title, logo image (located at 'path/to/logo.png'), and information about the application's purpose.

Main Content - The main content section begins with the title "SpeakEasy – Lip Reading using Deep Learning."

Video Selection - Users can choose a video from a list of options. The available options are dynamically loaded from the 'path/to/videos' directory.

Two Columns Display –

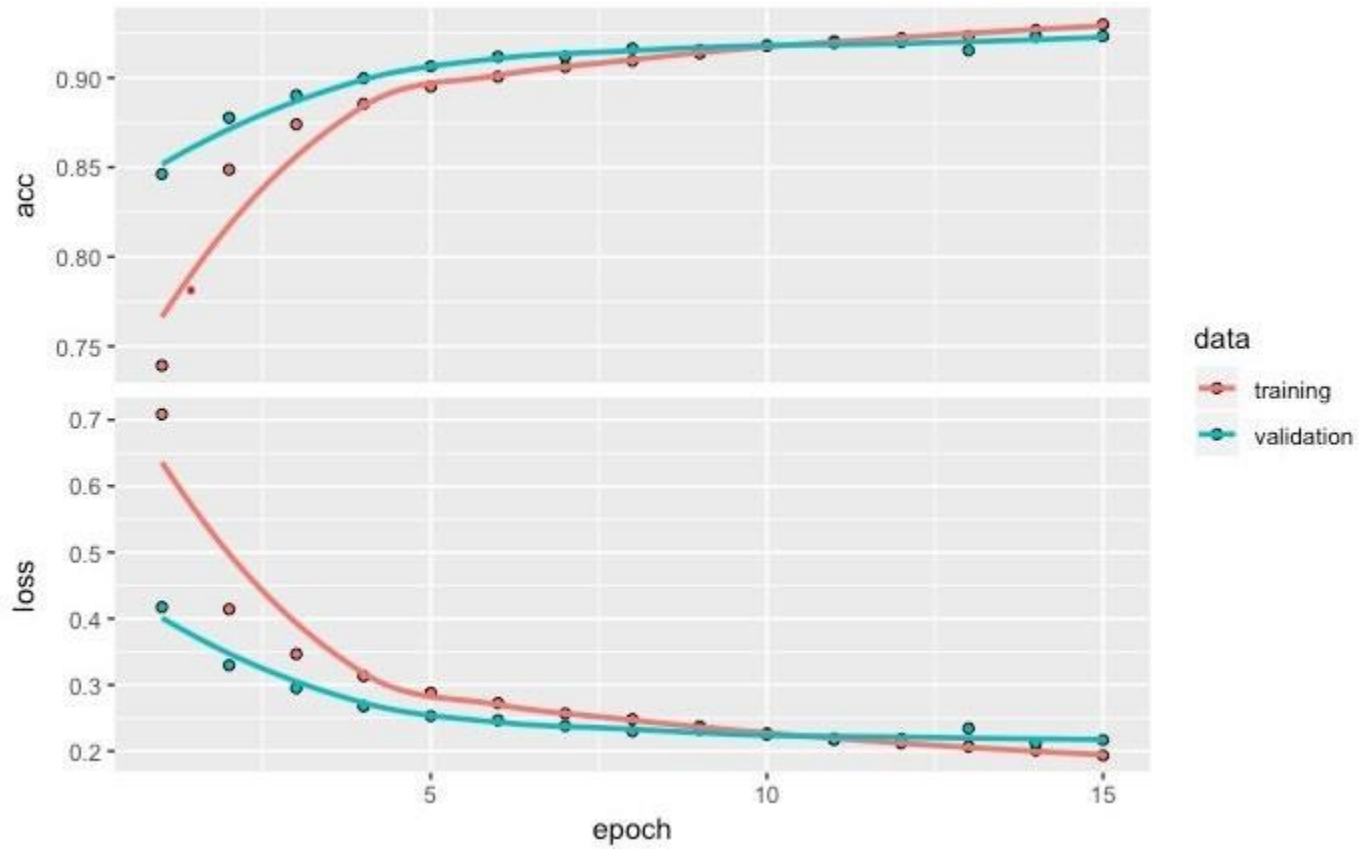
Column 1 (Left) - Render Selected Video. The selected video is rendered in MP4 format using FFmpeg. The video is displayed within the app.

Column 2 (Right) - Machine Learning Model Predictions
This column shows what the machine learning model sees when making predictions.
An animated GIF ('animation.gif') illustrates the model's perspective.
Machine learning model predictions are displayed as tokens.
The raw tokens are decoded into words.
The converted prediction is shown as text.

Usage - Users can interactively observe both the input video and the lip-reading model's predictions in a user-friendly interface.

8. PERFORMANCE TESTING

8.1 Performance Metrics –



Training accuracy = 93%

Validation accuracy = 91%

9. RESULTS

9.1 Output Screenshots

10. ADVANTAGES & DISADVANTAGES

10.1 Advantages –

1. **Accessibility** – Lip reading technology can enhance accessibility for individuals who are deaf or hard of hearing by providing them with a means to comprehend spoken language.

2. **Communication Enhancement -** It can improve communication in noisy environments where audio cues may be distorted or masked, aiding individuals in understanding conversations better.

3. **Potential for Multimodal Integration -** Combining lip reading with other technologies like speech recognition could improve accuracy and create more comprehensive communication aids.

10.2 Disadvantages –

1. **Accuracy Challenges -** Lip reading is inherently challenging due to variations in lip movements, accents, speech speed and obscured views, making it prone to errors and misinterpretations.

2. **Limited Vocabulary and Context -** Lip reading may struggle with complex or context-specific words, phrases, or ambiguous lip movements, leading to misunderstandings.

3. **Ethical Concerns -** Privacy issues might arise if the technology is used without consent, especially in situations where individuals expect their speech to be private.

4. **Dependency on Visual Cues -** Lip reading may not be applicable in situations where lip movements are not visible, such as in phone calls or when the speaker's face is obscured.

5. **Computational Requirements -** Sophisticated machine learning models used in lip reading may require substantial computational resources, limiting their deployment in certain devices or settings.

6. **Challenges -** While lip reading through machine learning presents promising opportunities, it also faces considerable challenges in achieving high accuracy and reliability across diverse real-world scenarios. Addressing these challenges requires ongoing research, data diversity and technological advancements.

11. CONCLUSION

In conclusion, the App “SpeakEasy – Lip Reading using Deep Learning” represents a comprehensive solution for lip reading using deep learning technologies. The project combines state-of-the-art machine learning models with an interactive and user-friendly interface created using Streamlit.

Key Achievements -

End-to-End Lip Reading - The application successfully achieves end-to-end lip reading, providing accurate predictions for words spoken in the input videos.

Interactive Visualization - Users can interact with the app by selecting videos and witnessing the machine learning model's predictions in real-time. The interactive interface enhances user engagement and understanding.

Efficient Model Loading - The model loading process is efficient, thanks to custom utility functions that load data, convert numerical predictions to characters, and handle model loading seamlessly.

Progress Feedback - The app incorporates visual feedback elements such as spinners and progress bars to keep users informed about the status of video rendering and model predictions.

12. FUTURE SCOPE

User Authentication - Consider adding user authentication features to personalize the user experience, track preferences, and provide personalized lip-reading insights.

Expanded Model Capabilities - Explore opportunities to enhance the lip-reading model's capabilities, such as handling diverse accents, languages, and noisy environments.

Real-Time Lip Reading - Investigate the feasibility of implementing real-time lip-reading capabilities, allowing the app to process live video feeds for immediate predictions.

Overall Impact –

The SpeakEasy – Lip Reading using Deep Learning showcases the potential of leveraging deep learning for practical applications, emphasizing the importance of accessibility and inclusivity. The project has the potential to assist individuals with hearing impairments, language barriers, or situations where audio information is limited.

This application stands as a testament to the innovative use of technology to address real-world challenges, and its impact extends beyond conventional machine learning applications. The journey from video selection to accurate lip-reading predictions demonstrates the seamless integration of cutting-edge technologies into a user-friendly platform. SpeakEasy – Lip Reading using Deep Learning App not only delivers on its technical objectives but also sets the stage for further advancements in the field of assistive technologies and human-computer interaction.

13. APPENDIX

13.1 Source Code - The source code for SpeakEasy – Lip Reading using Deep Learning can be found on the project's GitHub repository. Developers and interested parties can access.

13.2 Project Demo -
A recorded demonstration video showcasing SpeakEasy – Lip Reading using Deep Learning is available for users to explore its features and functionalities. Visit the following link to watch the demo and gain insights into the app's capabilities.

13.3 Additional Resources - For supplementary materials, documentation, and resources related to the development and implementation of the SpeakEasy – Lip Reading using Deep Learning, please refer to the project's documentation folder available on the GitHub repository.