LIPSENSE

DEEP LEARNING BASED SILENT SPEECH RECOGNITION USING 3D CONVOLUTIONAL NEURAL NETWORKS AND GATED RECURRENT UNITS

1.Introduction

1.1 Overview

The field of speech recognition has witnessed significant advancements over the years, with a focus on auditory input for deciphering spoken language. However, in scenarios where auditory information is either unavailable or compromised, the comprehension of speech becomes challenging. This is particularly pertinent for individuals with hearing impairments or in noise-affected environments. In response to this challenge, the present study delves into the realm of silent speech recognition, capitalising on the visual modality of speech perception. The project seeks to develop an innovative solution by harnessing deep learning techniques, specifically, a fusion of 3D Convolutional Neural Networks (CNNs) and Gated Recurrent Units (GRUs), to facilitate automatic lip reading. The resultant model endeavours to transform lip movements into coherent textual representations, thereby bridging the gap between speech and visual cues.

1.2 Purpose

Speech is not always an effective form of communication under different circumstances.First, the environment could not be conducive to speech-based communication; for example, someone might be in a noisy restaurant or close to a crowded market where it is difficult to understand speech. Second, a person can be in a public place where others don't want to be disturbed, like a library or museum, and not want to talk aloud out of worries for their privacy and confidentiality. Finally, and most crucially, a large number of people struggle to talk or are completely speechless as a result of a variety of neurological and speech impairments.

The overarching purpose of this project is to engineer a sophisticated and accurate silent speech recognition system by means of a novel fusion of 3D CNNs and GRUs. This system has the potential to not only augment the quality of life for individuals with hearing impairments but also to transcend to diverse domains where audio data may be compromised or unavailable. In envisaging an innovative solution that leverages the inherent cues present in lip movements,

this project strives to empower individuals to communicate effectively through visual speech cues. Moreover, the fusion of 3D CNNs and GRUs promises to yield a comprehensive understanding of spatio-temporal aspects of lip motion, leading to enhanced accuracy and robustness in silent speech recognition. As such, this project seeks to break barriers in communication, revolutionising the way speech is perceived and interpreted in various spheres of human interaction and technology deployment.

2.Literature Survey

2.1 Existing Problem

This section presents an overview of prevailing methodologies in the domain of automated lipreading. The existing landscape encompasses a range of techniques, where the emphasis often lies on overcoming challenges associated with speech recognition in noise-affected, privacy-conscious, or speech-impaired scenarios. Prior endeavours delve into automated lipreading, utilising diverse methods that do not extensively harness deep learning. Many such approaches necessitate intricate preprocessing of frames for feature extraction, temporal preprocessing for video feature extraction, or employ manually crafted vision pipelines.

Goldschen et al. pioneered visual-only sentence-level lipreading through hidden Markov models (HMMs), albeit in a limited dataset. Neti et al. ventured into sentence-level audiovisual speech recognition, melding HMMs with engineered features. Akin to these endeavours, Potamianos et al. reported substantial speaker-independent and speaker-adapted improvements using similar techniques. Furthermore, Gergen et al. attained the then-state-of-the-art on the GRID corpus by employing speaker-dependent training in an HMM/GMM system, emphasising the challenges of generalisation across speakers and motion feature extraction.

Classification with deep learning has surfaced in recent years, often targeting word or phoneme classification. However, the transformative aspect of full sentence sequence prediction, a cornerstone of LipNet's approach, remains less explored. Notable attempts include Ngiam et al., Sui et al., Ninomiya et al., Petridis & Pantic, and others, who engage in multimodal audio-visual representations and integrate visual features into traditional speech-style processing pipelines. Spatial and spatiotemporal convolutional neural networks, proposed by Chung & Zisserman, demonstrate promise in word classification, yet lack the robustness for sentence-level sequence prediction. Similarly, other studies like Chung & Zisserman (audio-visual max-margin matching

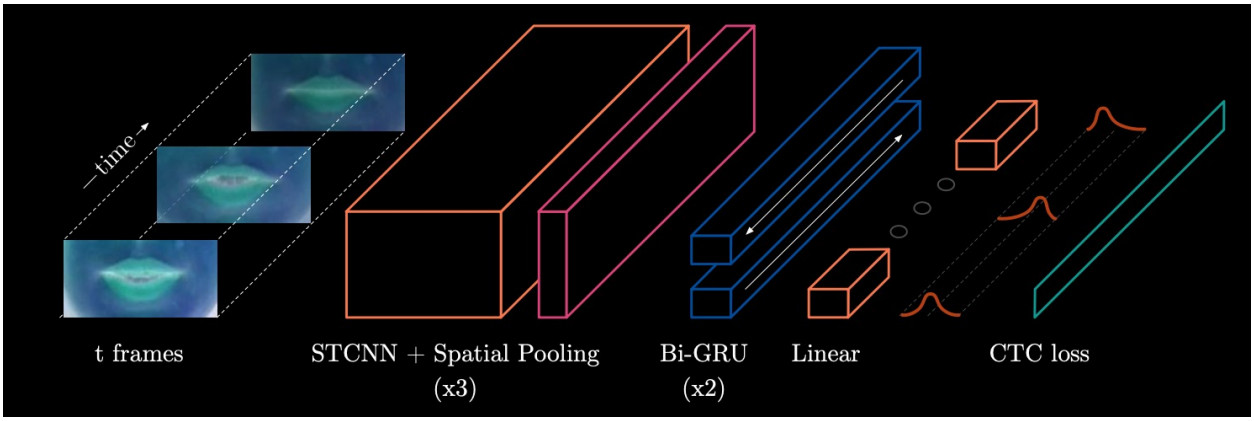
model), Wand et al. (LSTM recurrent neural networks), and Garg et al. (VGG pre-trained model) have yielded advancements within specific classifications, though not entirely encompassing sentence-level sequence prediction.

2.2 Proposed Solution

In light of the existing landscape, this project advocates for the adoption of LipNet—a groundbreaking end-to-end model that pioneers sentence-level sequence prediction for visual speech recognition. Unlike prior methodologies, LipNet efficiently processes sequences of images and yields output distributions over sequences of tokens. This innovative framework leverages the power of Connectionist Temporal Classification (CTC) loss, facilitating end-to-end training and obviating the need for intricate alignments. In a field that has often paralleled early automatic speech recognition (ASR) progress, LipNet emerges as a notable leap by transcending the limits of classification-based approaches, enhancing sentence-level sequence prediction.

3.Theoretical Analysis

3.1 Block Diagram



3.2 Hardware and Software Design

During the training phase, a powerful GPU is essential for accelerated deep learning model training. For this project, an Nvidia RTX 2070 Super was utilized. Specifications include:

- NVIDIA GeForce RTX 2070 Super
- CUDA Cores: 2560
- Base Clock: 1605 MHz
- Boost Clock: 1770 MHz
- VRAM: 8 GB GDDR6

- Memory Speed: 14 Gbps
- Memory Bus: 256-bit

For running the wep app a standard configuration of suffiecient RAM and CPU is required.

Software frameworks used:

- IBM Cloud
- IBM Watson Machine Learning API Client
- Python
- Numpy
- Tensorflow
- OpenCV
- Matplotlib
- ImageIO
- Streamlit

4.Experimental Investigations

4.1 Data collection

Evaluating which dataset would be the most suitable for training the deep learning model was analysed. The Grid corpus dataset was chosen.The dataset was then split to training, testing and validation sections.

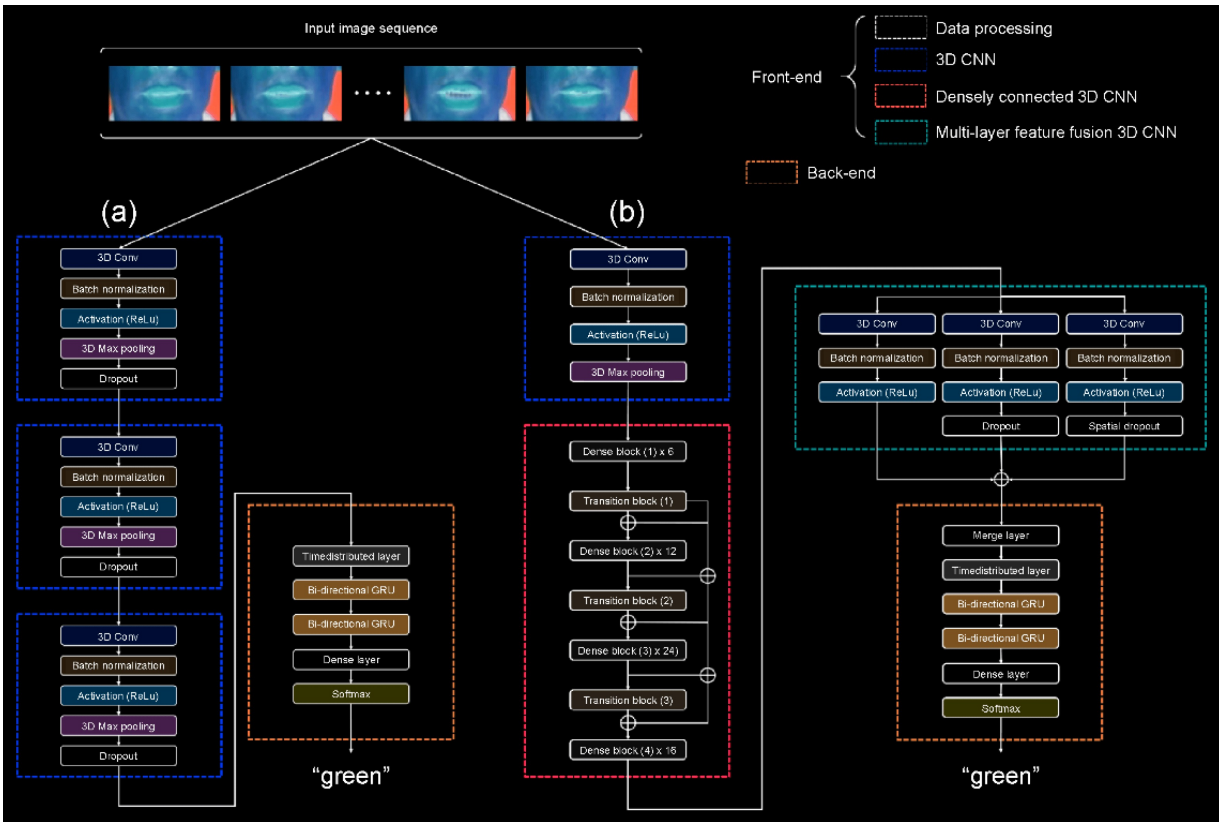
4.2 Neural Architecture

Then the design and architecture of the silent speech recognizer was chosen. The Lipnet architecture was strategically chosen to holistically capture both spatial and temporal features embedded in the lip motion sequences. The architecture is based upon 3D CNN and Bi-GRUs.

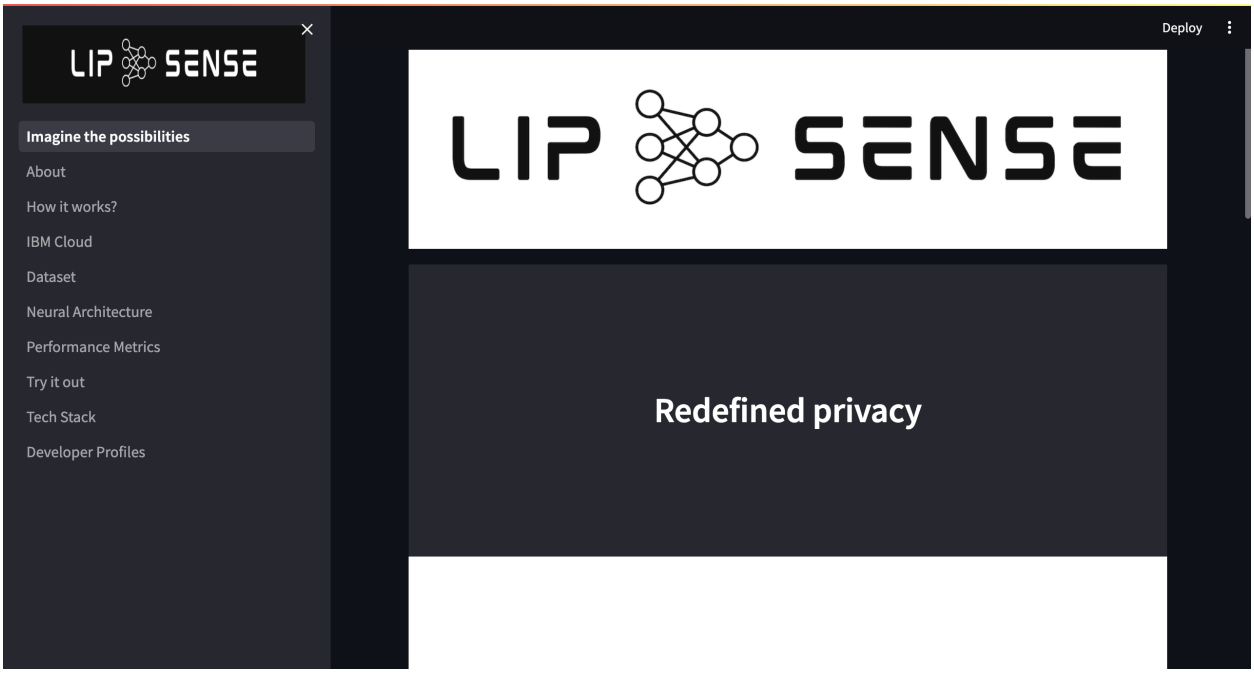
4.3 Training and Evaluation Metrics

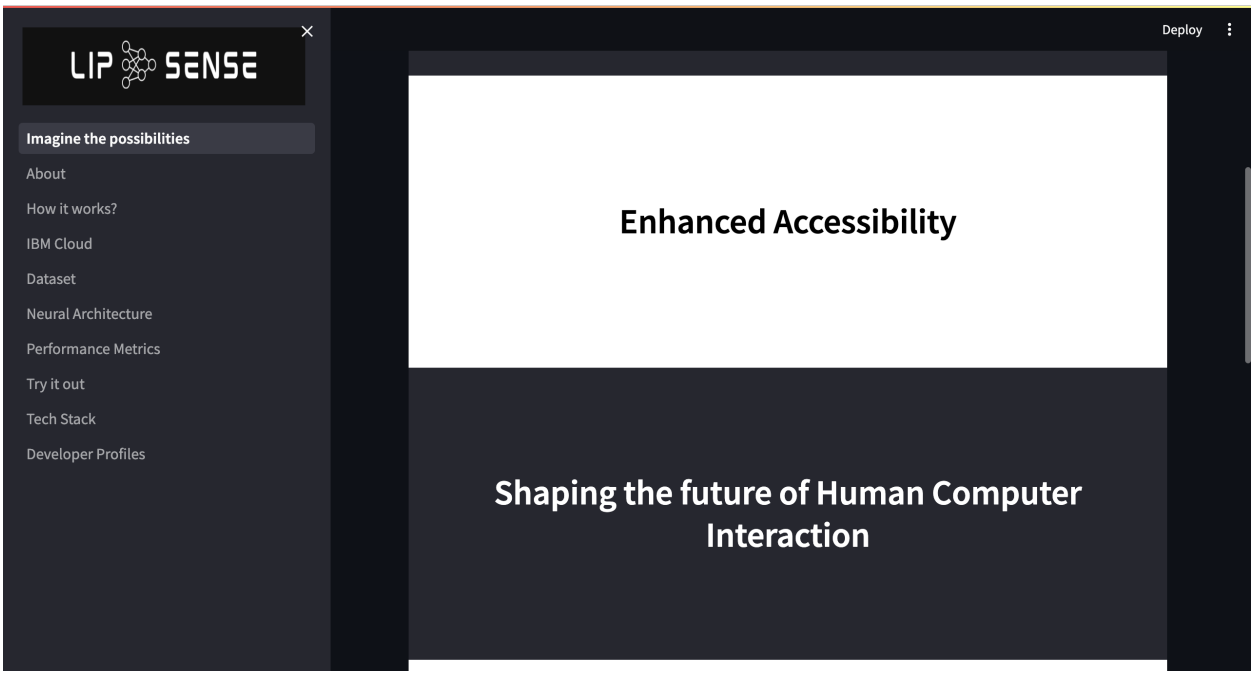
The training process was refined.We periodically performed validation to monitor accuracy and mitigate overfitting concerns.Thorough hyperparameter optimization was carried out. This involved fine-tuning parameters like learning rate, batch size, and dropout rates to maximise model performance.Word error rate and character error rate were then analysed to evaluate the performance of the model.

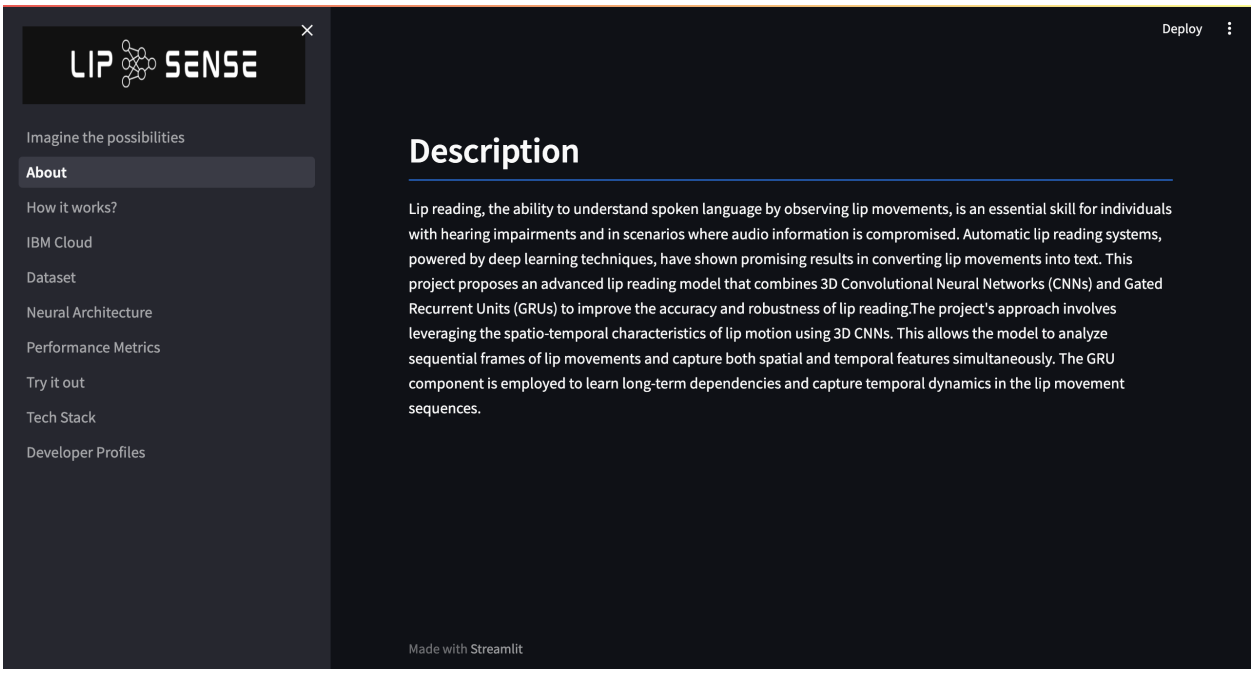
5. Flowchart



6.Result




LIP SENSE
Imagine the possibilities
About
How it works?
IBM Cloud
Dataset
Neural Architecture
Performance Metrics
Try it out
Tech Stack
Developer Profiles
Deploy
Enhanced Accessibility
Shaping the future of Human Computer Interaction


LIP SENSE
Imagine the possibilities
About
How it works?
IBM Cloud
Dataset
Neural Architecture
Performance Metrics
Try it out
Tech Stack
Developer Profiles
Deploy
Description
Lip reading, the ability to understand spoken language by observing lip movements, is an essential skill for individuals with hearing impairments and in scenarios where audio information is compromised. Automatic lip reading systems, powered by deep learning techniques, have shown promising results in converting lip movements into text. This project proposes an advanced lip reading model that combines 3D Convolutional Neural Networks (CNNs) and Gated Recurrent Units (GRUs) to improve the accuracy and robustness of lip reading.The project's approach involves leveraging the spatio-temporal characteristics of lip motion using 3D CNNs. This allows the model to analyze sequential frames of lip movements and capture both spatial and temporal features simultaneously. The GRU component is employed to learn long-term dependencies and capture temporal dynamics in the lip movement sequences.
Made with Streamlit

LIP SENSE

Imagine the possibilities
About
How it works?
IBM Cloud
Dataset
Neural Architecture
Performance Metrics
Try it out
Tech Stack
Developer Profiles

Deploy

How it works?

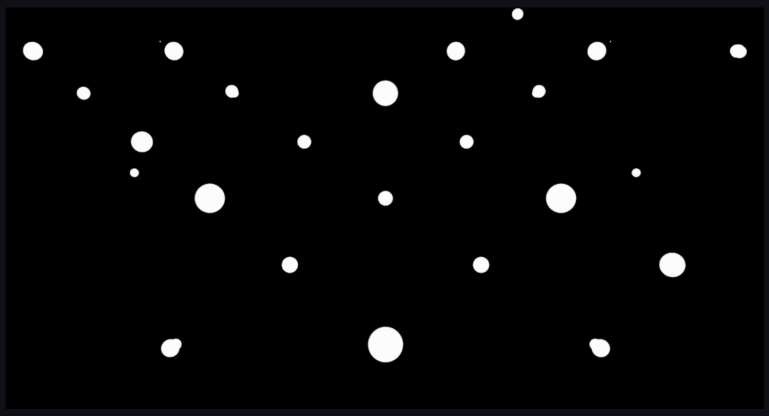
- The LipNet architecture operates through a meticulously designed sequence of operations.
- It begins with a series of three stages, each involving spatiotemporal convolutions, channel-wise dropout, and spatial max-pooling.
- This initial stage extracts crucial information from the input, encapsulating essential features.
- The architecture introduces two Bidirectional Gated Recurrent Units (Bi-GRUs), a pivotal element in the process.
- These Bi-GRUs facilitate the effective aggregation of outputs from the earlier spatiotemporal convolutions.
- Following this, each time-step undergoes a linear transformation.
- Subsequently, a softmax operation covers a vocabulary expanded to encompass the CTC blank symbol.
- The architecture's learning process is driven by the application of the Connectionist Temporal Classification (CTC) loss.
- Throughout the architecture, the Rectified Linear Unit (ReLU) serves as the activation function in all layers.
- This consistent choice of activation function contributes to the model's efficiency and performance.
- The orchestrated sequence of operations culminates in the LipNet architecture's proficiency in lip reading tasks.

LIP SENSE

Imagine the possibilities
About
How it works?
IBM Cloud
Dataset
Neural Architecture
Performance Metrics
Try it out
Tech Stack
Developer Profiles

Deploy

Deployment made easy with IBM Cloud



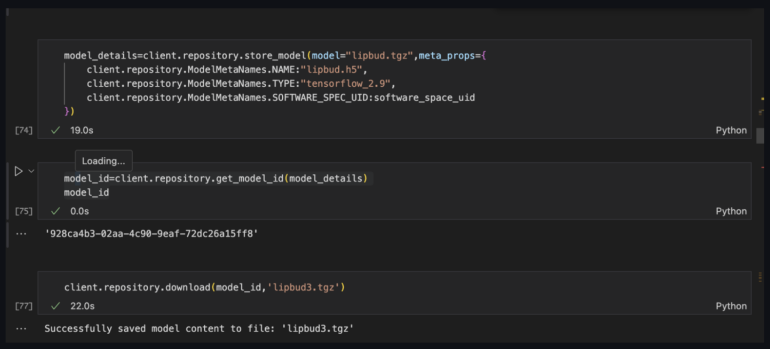
IBM Watson Machine Learning is a full-service IBM Cloud offering that makes it easy for developers and data scientists to work together to integrate predictive capabilities with their applications. The Watson Machine Learning service is a set of REST APIs that you can call from any programming language to develop applications that make smarter decisions, solve tough problems, and improve user outcomes.

LIP SENSE

- Imagine the possibilities
- About
- How it works?
- **IBM Cloud**
- Dataset
- Neural Architecture
- Performance Metrics
- Try it out
- Tech Stack
- Developer Profiles

Deploy

IBM Watson Machine Learning is a full-service IBM Cloud offering that makes it easy for developers and data scientists to work together to integrate predictive capabilities with their applications. The Watson Machine Learning service is a set of REST APIs that you can call from any programming language to develop applications that make smarter decisions, solve tough problems, and improve user outcomes.



Using the IBM Watson Machine Learning API client

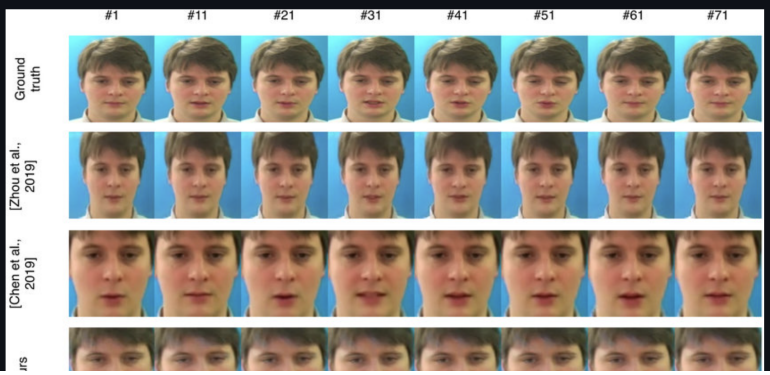
LIP SENSE

- Imagine the possibilities
- About
- How it works?
- IBM Cloud
- **Dataset**
- Neural Architecture
- Performance Metrics
- Try it out
- Tech Stack
- Developer Profiles

Deploy

Dataset

We employ the GRID corpus for training due to its sentence-level nature and extensive dataset. The sentences are structured using a straightforward grammar pattern: command(4) + color(4) + preposition(4) + letter(25) + digit(10) + adverb(4). Each number indicates the available word choices within the corresponding word category. These categories encompass {bin, lay, place, set}, {blue, green, red, white}, {at, by, in, with}, {A,...,Z}{W}, {zero, ..., nine}, and {again, now, please, soon}, resulting in a pool of 64,000 potential sentences. For instance, exemplars from this dataset include 'set blue by A four please' and 'place red at C zero again'. Model training was conducted on the first 450 videos of speaker one.

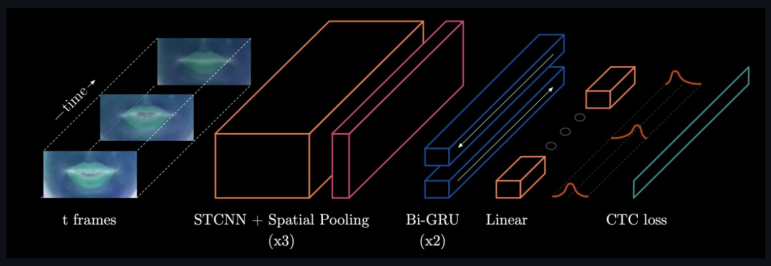


LIP SENSE

- Imagine the possibilities
- About
- How it works?
- IBM Cloud
- Dataset
- **Neural Architecture**
- Performance Metrics
- Try it out
- Tech Stack
- Developer Profiles

Deploy

Neural Architecture



	Layer (type)	Output Shape	Param #
0	conv3d_3 (Conv3D)	(None, 75, 46, 140, 128)	3584
1	activation_3 (Activation)	(None, 75, 46, 140, 128)	0
2	max_pooling3d_3 (MaxPooling3D)	(None, 75, 23, 70, 128)	0
3	conv3d_4 (Conv3D)	(None, 75, 23, 70, 256)	884992
4	activation_4 (Activation)	(None, 75, 23, 70, 256)	0
5	max_pooling3d_4 (MaxPooling3D)	(None, 75, 11, 35, 256)	0
6	conv3d_5 (Conv3D)	(None, 75, 11, 35, 75)	518475
7	activation_5 (Activation)	(None, 75, 11, 35, 75)	0

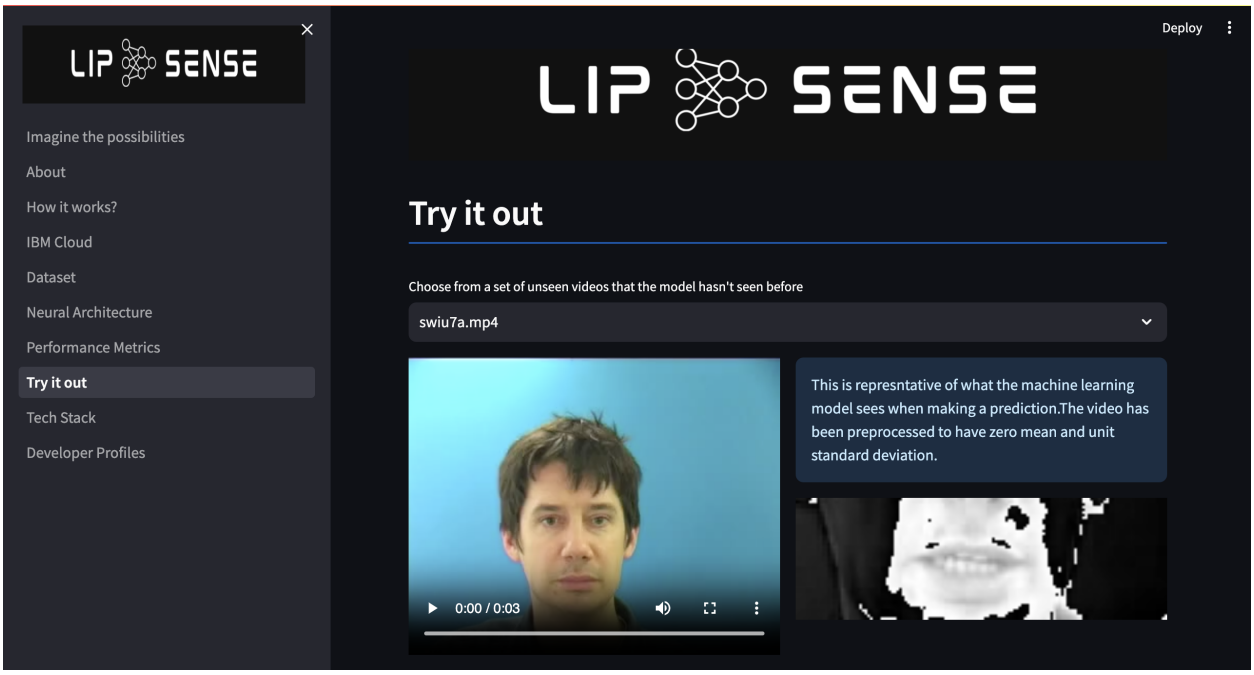
LIP SENSE

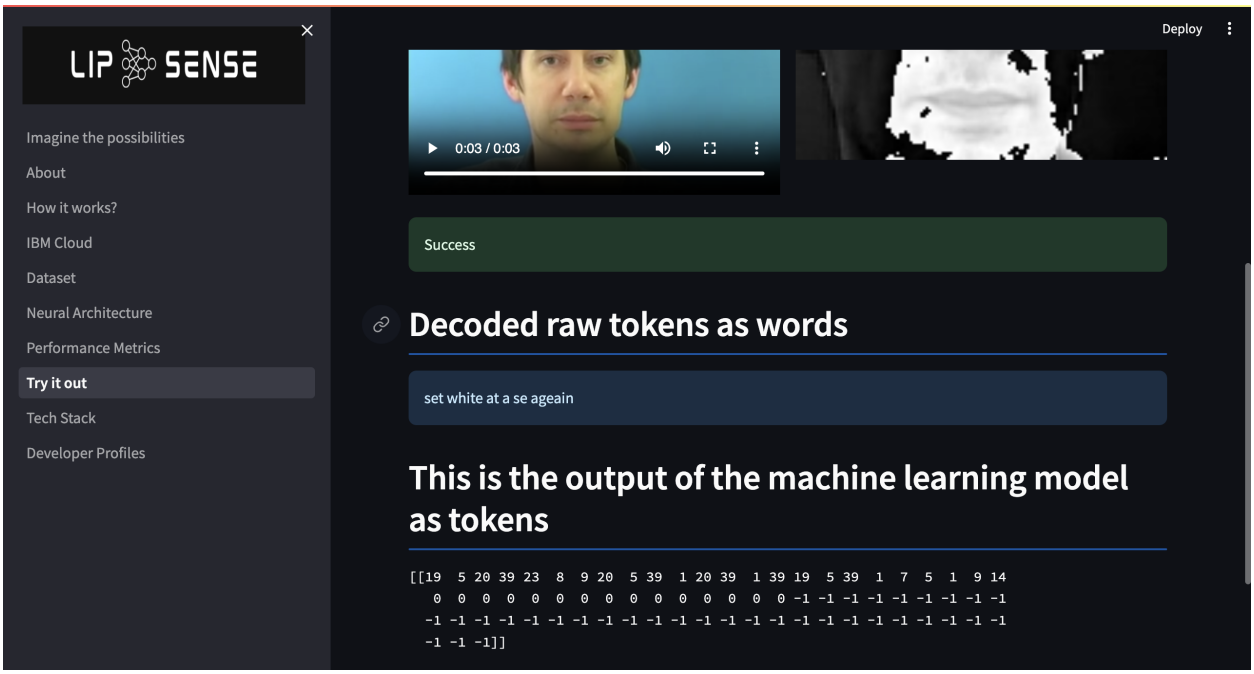
- Imagine the possibilities
- About
- How it works?
- IBM Cloud
- Dataset
- Neural Architecture
- **Performance Metrics**
- Try it out
- Tech Stack
- Developer Profiles

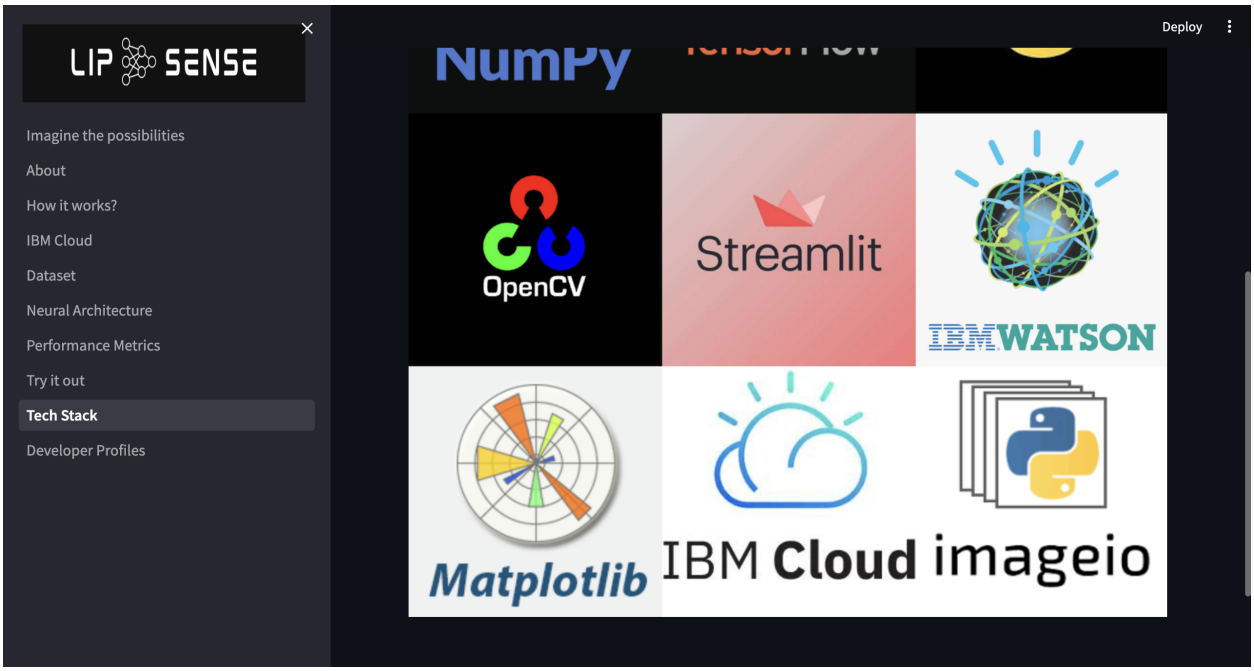
Deploy

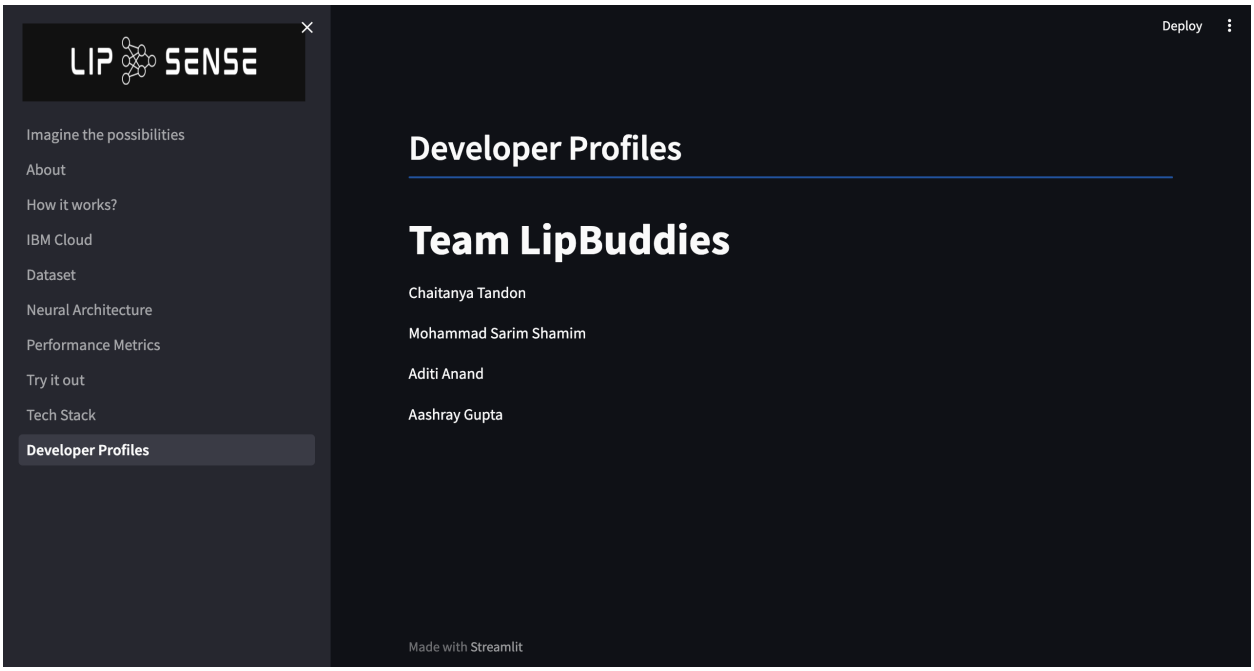
Performance Metrics

Outperforms human by	Word Error Rate	Sentence accuracy
4.1x	4.8%	95.2%
Training Videos	Unseen CER	Vocabulary tokens
450	6.4%	41
Trainable parameters	STCNN Layers	Bi-GRU
8471924	x3	x2


LIP SENSE
Imagine the possibilities
About
How it works?
IBM Cloud
Dataset
Neural Architecture
Performance Metrics
Try it out
Tech Stack
Developer Profiles
Deploy
LIP SENSE
Try it out
Choose from a set of unseen videos that the model hasn't seen before
swiu7a.mp4
0:00 / 0:03
This is represntative of what the machine learning model sees when making a prediction.The video has been preprocessed to have zero mean and unit standard deviation.


LIP SENSE
Imagine the possibilities
About
How it works?
IBM Cloud
Dataset
Neural Architecture
Performance Metrics
Try it out
Tech Stack
Developer Profiles
Deploy
0:03 / 0:03
Success
Decoded raw tokens as words
set white at a se ageain
This is the output of the machine learning model as tokens
[[19 5 20 39 23 8 9 20 5 39 1 20 39 1 39 19 5 39 1 7 5 1 9 14
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
-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]]





7.Advantages and Disadvantages

7.1 Advantages

1. Enhanced Accessibilty: The project significantly improves accessibility for individuals with speech impairments or in scenarios where vocalization is challenging.

2. Noise proof system: In noisy environments where traditional speech recognition struggles, this system remains effective. It is capable of interpreting lip movements even in acoustically challenging settings.

3. Privacy and Discretion: The system offers a discreet communication mode for situations where speaking aloud may disrupt the environment or raise privacy concerns, such as in libraries or public transportation.

4. Human-Machine Interaction:Integrating this technology into voice-controlled virtual assistants and technology interfaces enhances human-machine interaction, making it more intuitive and accessible.

7.2 Disadvantages

1.Hardware Requirements:Real-time processing and accuracy may demand substantial computational resources, potentially limiting the system's accessibility on low-end devices.

2. Lighting Dependence: Variations in lighting conditions can impact system performance. A consistent light source may be required for optimal accuracy.

4. Multilingual Training:Training the system for multilingual support is resource-intensive and time-consuming, requiring a diverse dataset for each language.

6. Accuracy Challenges:While the system can be highly accurate, challenges may arise in accurately interpreting subtle lip movements or dialect-specific variations.

8. Applications

The silent speech recognition system offers versatile applications, transcending speech limitations in various scenarios:

1.Noisy Environments:Thrives in bustling places where noise hinders speech recognition, ensuring effective communication.

2.Privacy and Security:Enables discreet communication in settings like libraries, preserving privacy.

3.Speech and Neurological Disorders: A lifeline for those with speech difficulties, fostering inclusivity.

4.AAC Enhancement:Enhances Augmentative and Alternative Communication (AAC) by providing natural communication via lip movements.

5.Human-Machine Interaction:Revolutionizes voice-controlled technology interaction, boosting accessibility.

6.Cross-Cultural Communication:Breaks language barriers, facilitating global communication.

7.Entertainment and Gaming:Enhances gaming experiences by responding to users' lip movements.

In summary, this system redefines communication across diverse contexts, offering inclusive, efficient, and engaging interactions

9.Conclusion

Our project employs a model to apply deep learning to end-to-end learning of a model that maps sequences of image frames of a speaker’s mouth to entire sentences. The end-to-end model eliminates the need to segment videos into words before predicting a sentence. This model requires neither hand-engineered spatiotemporal visual features nor a separately-trained sequence model.

However, to extend the range of potential applications of LipNet, we aim to apply this approach to a jointly trained audio- visual speech recognition model, where visual input assists with robustness in noisy environments.

10.Future Scope

1. Individual Calibration: Tailoring the model to individual users can significantly enhance its accuracy and personalization. An adaptive calibration mechanism, accounting for variations in lip movements and speech patterns across individuals, promises to refine the system's performance.
2. Integration with Other Expressions: Expanding beyond speech recognition, integrating the system with facial expressions could enable the interpretation of a broader spectrum of communication cues.
3. Spelling Correction: Extending the model to offer spelling correction based on character-level predictions is a logical extension. Such a feature would not only enhance the accuracy of the recognized words but also contribute to the user's communication fluency.
4. Light Enhancement Module: Incorporating a module for light enhancement could mitigate challenges posed by varying lighting conditions. This enhancement would ensure consistent performance across diverse environments, further boosting the reliability of the system.
5. Multilingual Support:The project's potential can be broadened by incorporating multilingual support. By training the model on diverse languages, the silent speech recognition system can accommodate a global user base, transcending linguistic barriers.
6. Gesture Interpretation: Expanding the system's capabilities to include interpretation of hand gestures and body language could result in a comprehensive communication platform that

encompasses various non-verbal cues.

7. Continuous Learning: Implementing mechanisms for continuous learning can equip the system with the ability to adapt and evolve over time, ensuring that it remains relevant and effective in dynamic communication scenarios.

8. Mobile Application Integration: Developing a user-friendly mobile application can democratize access to the system, allowing users to engage with it seamlessly on their smartphones and tablets.

11. Bibliography

1. [LipNet: End-to-End Sentence-level Lipreading](https://arxiv.org/abs/1611.01599)
2. [CS230 - Deep Learning Course](https://stanford.edu/~shervine/teaching/cs-230/)
3. [LipType: A Silent Speech Recognizer Augmented with an Independent Repair Model](URL)
4. [YouTube Video](https://youtu.be/qFJeN9V1ZsI)
5. (https://github.com/nicknochnack/LipNet/blob/main/LipNet.ipynb)
6.[YouTubeVideo](https://www.youtube.com/watch?v=gZmobeGL0Yg&list=PLZbbT5o_s2xq7LwI2y8_QtvuXZedL6tQU&ab_channel=deeplizard)
7. [YouTube Video](https://www.youtube.com/watch?v=Zm8-qssPE58&t=698s)
8. [YouTube Video](https://youtu.be/TysuP3KgSzc?si=-MMwMK3AmoVBvqtX)

Appendix

```
import streamlit as st
st.markdown(
        """
        <style>
            [data-testid="stSidebarNav"] {
                background-image: url("https://www.dropbox.com/scl/fi/1diny80z7lfy9xnif7wa7/Screenshot_2023-08-30_at_11.47.40_PM.png?rlkey=30biilu3jrlpdjlird0itaco9&dl=0&raw=1");
                background-repeat: no-repeat;
                padding-top: 20px;
                background-size: 290px 80px;
                background-position: 20px 20px;
            }
        </style>
        """,
        unsafe_allow_html=True,
    )
#st.image('/Users/chaitanyatandon/Desktop/LIPBUDDIES/Screenshot_2023-08-30_at_11.47.40_PM.png')
st.header('Description', divider='blue')

st.write("Lip reading, the ability to understand spoken language by observing lip movements, is an essential skill for individuals with hearing impairments and in scenarios where audio information is compromised. Automatic lip reading systems, powered by deep learning techniques, have shown promising results in converting lip movements into text. This project proposes an advanced lip reading model that combines 3D Convolutional Neural Networks (CNNs) and Gated Recurrent Units (GRUs) to improve the accuracy and robustness of lip reading.The project's approach involves leveraging the spatio-temporal characteristics of lip motion using 3D CNNs. This allows the model to analyze sequential frames of lip movements and capture both spatial and temporal features simultaneously. The GRU component is employed to learn long-term dependencies and capture temporal dynamics in the lip movement sequences.")
```

```
import streamlit as st


st.markdown(
        """
        <style>
            [data-testid="stSidebarNav"] {
                background-image: url("https://www.dropbox.com/scl/fi/1diny80z7lfy9xnif7wa7/Screenshot_2023-08-30_at_11.47.40_PM.png?rlkey=30biilu3jrlpdjlird0itaco9&dl=0&raw=1");
                background-repeat: no-repeat;
                padding-top: 20px;
                background-size: 290px 80px;
                background-position: 20px 20px;
            }
        </style>
        """,
        unsafe_allow_html=True,
    )
st.header('How it works?', divider='blue')

points = [
    "The LipNet architecture operates through a meticulously designed sequence of operations.",
    "It begins with a series of three stages, each involving spatiotemporal convolutions, channel-wise dropout, and spatial max-pooling.",
    "This initial stage extracts crucial information from the input, encapsulating essential features.",
    "The architecture introduces two Bidirectional Gated Recurrent Units (Bi-GRUs), a pivotal element in the process.",
    "These Bi-GRUs facilitate the effective aggregation of outputs from the earlier spatiotemporal convolutions.",
    "Following this, each time-step undergoes a linear transformation.",
    "Subsequently, a softmax operation covers a vocabulary expanded to encompass the CTC blank symbol.",
    "The architecture's learning process is driven by the application of the Connectionist Temporal Classification (CTC) loss.",
    "Throughout the architecture, the Rectified Linear Unit (ReLU) serves as the activation function in all layers.",
    "This consistent choice of activation function contributes to the model's efficiency and performance.",
    "The orchestrated sequence of operations culminates in the LipNet architecture's proficiency in lip reading tasks."
]

for point in points:
    st.write(f"- {point}")

st.image('/Users/chaitanyatandon/Desktop/LIPBUDDIES/image (1).jpg')
```

```
import streamlit as st

st.header('Deployment made easy with IBM Cloud',divider="blue")
st.markdown(
        """
        <style>
            [data-testid="stSidebarNav"] {
                background-image: url("https://www.dropbox.com/scl/fi/1diny80z7lfy9xnif7wa7/Screenshot_2023-08-30_at_11.47.40_PM.png?rlkey=30biilu3jrlpdjlird0itaco9&dl=0&raw=1");
                background-repeat: no-repeat;
                padding-top: 20px;
                background-size: 290px 80px;
```

```
                    background-position: 20px 20px;
                }
            </style>
            """,
            unsafe_allow_html=True,
        )
    st.image('/Users/chaitanyatandon/Desktop/LIPBUDDIES/hero_animation.gif')
    #st.video('/Users/chaitanyatandon/Desktop/LIPBUDDIES/hero_animation.mp4')

    st.write('IBM Watson Machine Learning is a full-service IBM Cloud offering that makes it easy for developers and data scientists to work together to integrate predictive capabilities with their applications. The Watson Machine Learning service is a set of REST APIs that you can call from any programming language to develop applications that make smarter decisions, solve tough problems, and improve user outcomes.')

    st.image('/Users/chaitanyatandon/Desktop/LIPBUDDIES/Screenshot 2023-09-03 at 11.04.04 PM.png',use_column_width="always")
    st.info('Using the IBM Watson Machine Learning API client')
    import streamlit as st

    st.markdown(
            """
            <style>
                [data-testid="stSidebarNav"] {
                    background-image: url("https://www.dropbox.com/scl/fi/1diny80z7lfy9xnif7wa7/Screenshot_2023-08-30_at_11.47.40_PM.png?rlkey=30biilu3jrlpdjlird0itaco9&dl=0&raw=1");
                    background-repeat: no-repeat;
                    padding-top: 20px;
                    background-size: 290px 80px;
                    background-position: 20px 20px;
                }
            </style>
            """,
            unsafe_allow_html=True,
        )
    st.header('Dataset', divider='blue')


    text = "We employ the GRID corpus for training due to its sentence-level nature and extensive dataset. " \
           "The sentences are structured using a straightforward grammar pattern: command(4) + color(4) + " \
           "preposition(4) + letter(25) + digit(10) + adverb(4). Each number indicates the available word " \
           "choices within the corresponding word category. These categories encompass {bin, lay, place, set}, " \
           "{blue, green, red, white}, {at, by, in, with}, {A,...,Z}{W}, {zero, ..., nine}, and " \
           "{again, now, please, soon}, resulting in a pool of 64,000 potential sentences. For instance, " \
           "exemplars from this dataset include 'set blue by A four please' and 'place red at C zero again'. " \
           "Model training was conducted on the first 450 videos of speaker one."

    st.write(text)

    st.image('/Users/chaitanyatandon/Desktop/LIPBUDDIES/GRID-dataset-Frames-generated-by-different-models-for-the-speech-input-Bin-Blue-at-C-2.png')
    st.info('GRID dataset: Frames generated by different models for the speech input "Bin Blue at C 2 Soon" (#t indicates the time index)')

    import streamlit as st
    import pandas as pd

    st.markdown(
            """
            <style>
                [data-testid="stSidebarNav"] {
                    background-image: url("https://www.dropbox.com/scl/fi/1diny80z7lfy9xnif7wa7/Screenshot_2023-08-30_at_11.47.40_PM.png?rlkey=30biilu3jrlpdjlird0itaco9&dl=0&raw=1");
                    background-repeat: no-repeat;
                    padding-top: 20px;
                    background-size: 290px 80px;
                    background-position: 20px 20px;
                }
            </style>
            """,
            unsafe_allow_html=True,
        )

    st.header('Neural Architecture', divider='blue')

    st.image('/Users/chaitanyatandon/Desktop/LIPBUDDIES/image.jpg')


    data = {
        "Layer (type)": ["conv3d_3 (Conv3D)", "activation_3 (Activation)", "max_pooling3d_3 (MaxPooling3D)",
                         "conv3d_4 (Conv3D)", "activation_4 (Activation)", "max_pooling3d_4 (MaxPooling3D)",
                         "conv3d_5 (Conv3D)", "activation_5 (Activation)", "max_pooling3d_5 (MaxPooling3D)",
                         "time_distributed_1 (TimeDistributed)", "bidirectional_2 (Bidirectional)",
                         "dropout_2 (Dropout)", "bidirectional_3 (Bidirectional)", "dropout_3 (Dropout)",
                         "dense (Dense)"],
        "Output Shape": ["(None, 75, 46, 140, 128)", "(None, 75, 46, 140, 128)", "(None, 75, 23, 70, 128)",
                         "(None, 75, 23, 70, 256)", "(None, 75, 23, 70, 256)", "(None, 75, 11, 35, 256)",
                         "(None, 75, 11, 35, 75)", "(None, 75, 11, 35, 75)", "(None, 75, 5, 17, 75)",
                         "(None, 75, 6375)", "(None, 75, 256)", "(None, 75, 256)", "(None, 75, 256)",
```

```
                      "(None, 75, 256)", "(None, 75, 41)"],
    "Param #": ["3584", "0", "0", "884992", "0", "0", "518475", "0", "0", "0", "6660096", "0", "394240",
                "0", "10537"]
}

df = pd.DataFrame(data)

st.table(df)

total_params = "8471924"
trainable_params = "8471924"
non_trainable_params = "0"

st.header("Model Parameters",divider='blue')
st.write("Total params:", total_params, "(", f"{int(total_params) / (1024 * 1024):.2f}", "MB)")
st.write("Trainable params:", trainable_params, "(", f"{int(trainable_params) / (1024 * 1024):.2f}", "MB)")
st.write("Non-trainable params:", non_trainable_params, "(", f"{int(non_trainable_params) / (1024 * 1024):.2f}", "Byte)")
import streamlit as st
st.set_page_config(layout="wide")
st.markdown(
        """
        <style>
            [data-testid="stSidebarNav"] {
                background-image: url("https://www.dropbox.com/scl/fi/1diny80z7lfy9xnif7wa7/Screenshot_2023-08-30_at_11.47.40_PM.png?rlkey=30biilu3jrlpdjlird0itaco9&dl=0&raw=1");
                background-repeat: no-repeat;
                padding-top: 20px;
                background-size: 290px 80px;
                background-position: 20px 20px;
            }
        </style>
        """,
        unsafe_allow_html=True,
    )

#st.image('/Users/chaitanyatandon/Desktop/LIPBUDDIES/Screenshot_2023-08-30_at_11.47.40_PM.png')
st.header('Performance Metrics', divider='blue')

col1, col2, col3 = st.columns(3)
col1.metric("Outperforms human by", "4.1x")
col2.metric("Word Error Rate", "4.8%")
col3.metric("Sentence accuracy", "95.2%")

col1, col2, col3 = st.columns(3)
col1.metric("Training Videos", "450")
col2.metric("Unseen CER", "6.4%")
col3.metric("Vocabulary tokens", "41")

col1, col2, col3 = st.columns(3)
col1.metric("Trainable parameters", "8471924")
col2.metric("STCNN Layers", "x3")
col3.metric("Bi-GRU", "x2")

st.image('/Users/chaitanyatandon/Desktop/LIPBUDDIES/ezgif.com-video-to-gif.gif',use_column_width="always")
import streamlit as st
from PIL import Image
st.image('/Users/chaitanyatandon/Desktop/LIPBUDDIES/Screenshot_2023-08-30_at_11.47.40_PM.png')
st.header('Try it out', divider='blue')




import streamlit as st
import os
import imageio
import time

import tensorflow as tf
from typing import List
import cv2

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv3D, LSTM, Dense, Dropout, Bidirectional, MaxPool3D, Activation, Reshape, SpatialDropout3D, BatchNormalization, TimeDistributed, Flatten

st.markdown(
        """
        <style>
            [data-testid="stSidebarNav"] {
                background-image: url("https://www.dropbox.com/scl/fi/1diny80z7lfy9xnif7wa7/Screenshot_2023-08-30_at_11.47.40_PM.png?rlkey=30biilu3jrlpdjlird0itaco9&dl=0&raw=1");
                background-repeat: no-repeat;
                padding-top: 20px;
                background-size: 290px 80px;
```

```
                    background-position: 20px 20px;
                }
            </style>
            """,
            unsafe_allow_html=True,
        )



vocab = [x for x in "abcdefghijklmnopqrstuvwxyz'?!123456789 "]
char_to_num = tf.keras.layers.StringLookup(vocabulary=vocab, oov_token="")
# Mapping integers back to original characters
num_to_char = tf.keras.layers.StringLookup(
    vocabulary=char_to_num.get_vocabulary(), oov_token="", invert=True
)

def load_model() -> Sequential:
    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(41, kernel_initializer='he_normal', activation='softmax'))

    model.load_weights(os.path.join('..','models','checkpoint'))

    return model

def load_video(path:str) -> List[float]:
    #print(path)
    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
def load_alignments(path:str) -> List[str]:
    #print(path)
    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]
    video_path = os.path.join('..','data','s1',f'{file_name}.mp4')
    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



# Generating a list of options or videos
options = os.listdir(os.path.join('..', 'data', 's1'))
selected_video = st.selectbox('Choose from a set of unseen videos that the model hasn\'t seen before', options)

```

```
# Generate two columns
col1, col2 = st.columns(2)

if options:

    # Rendering the video
    with col1:
        file_path = os.path.join('..','data','s1', selected_video)
        video = open(file_path, 'rb')
        video_bytes = video.read()
        st.video(video_bytes)
        #st.info('The video that you have selected')

    with col2:
        st.info('This is represntative of what the machine learning model sees when making a prediction.The video has been preprocessed to have zero mean and unit standard deviation.')
        video, annotations = load_data(tf.convert_to_tensor(file_path))
        st.image('/Users/chaitanyatandon/Desktop/LIPBUDDIES/animation.gif',use_column_width="always")

    model = load_model()
    yhat = model.predict(tf.expand_dims(video, axis=0))
    with st.spinner('Predicting'):
        time.sleep(5)
        st.success('Success')
    decoder = tf.keras.backend.ctc_decode(yhat, [75], greedy=True)[0][0].numpy()

        # Convert prediction to text
    st.header('Decoded raw tokens as words',divider='blue')
    converted_prediction = tf.strings.reduce_join(num_to_char(decoder)).numpy().decode('utf-8')
    st.info(converted_prediction)

    st.header('This is the output of the machine learning model as tokens',divider='blue')
    st.text(decoder)

import streamlit as st
st.markdown(
        """
        <style>
            [data-testid="stSidebarNav"] {
                background-image: url("https://www.dropbox.com/scl/fi/1diny80z7lfy9xnif7wa7/Screenshot_2023-08-30_at_11.47.40_PM.png?rlkey=30biilu3jrlpdjlird0itaco9&dl=0&raw=1");
                background-repeat: no-repeat;
                padding-top: 20px;
                background-size: 290px 80px;
                background-position: 20px 20px;
            }
        </style>
        """,
        unsafe_allow_html=True,
    )
st.image('/Users/chaitanyatandon/Desktop/LIPBUDDIES/Screenshot 2023-08-31 at 2.22.51 AM.png')
st.header('Tech Stack',divider="blue")

st.image('/Users/chaitanyatandon/Desktop/LIPBUDDIES/tech stack.png')
import streamlit as st
st.markdown(
        """
        <style>
            [data-testid="stSidebarNav"] {
                background-image: url("https://www.dropbox.com/scl/fi/1diny80z7lfy9xnif7wa7/Screenshot_2023-08-30_at_11.47.40_PM.png?rlkey=30biilu3jrlpdjlird0itaco9&dl=0&raw=1");
                background-repeat: no-repeat;
                padding-top: 20px;
                background-size: 290px 80px;
                background-position: 20px 20px;
            }
        </style>
        """,
        unsafe_allow_html=True,
    )
#st.image('/Users/chaitanyatandon/Desktop/LIPBUDDIES/Screenshot 2023-08-31 at 2.22.51 AM.png')
st.header('Developer Profiles',divider="blue")
st.title('Team LipBuddies')
st.write('Chaitanya Tandon')
st.write('Mohammad Sarim Shamim')
st.write('Aditi Anand')
st.write('Aashray Gupta')

!pip3 install tensorflow
!pip3 list
import tensorflow as tf
from ibm_watson_machine_learning import APIClient

wml_credentials={
```

```
        "url":"https://us-south.ml.cloud.ibm.com",
        "apikey":"nF2BU5KRYc9z2ZnMhNuzVcghozZaznLHue1h5h0crcVk"
    }
    client=APIClient(wml_credentials)
    client
    client.spaces.list()
    space_uid="f9a08d7a-1730-4d9f-b753-50c531422a76"
    client.set.default_space(space_uid)
    client.software_specifications.list()
    software_space_uid=client.software_specifications.get_uid_by_name("runtime-22.2-py3.10")
    software_space_uid
    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
    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'))
    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
    )
    model.summary()
    model.save("lipbud.h5")
    !tar -zcvf lipbud.tgz lipbud.h5
    model_details=client.repository.store_model(model="lipbud.tgz",meta_props={
        client.repository.ModelMetaNames.NAME:"lipbud.h5",
        client.repository.ModelMetaNames.TYPE:"tensorflow_2.9",
        client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_space_uid
    })
    model_id=client.repository.get_model_id(model_details)
    model_id
    client.repository.download(model_id,'lipbud3.tgz')
    model_id=client.repository.get_model_id(model_details)
    model_id
    model.load_weights('models/checkpoint')
    model.save("lipbud.h5")
    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')
    !pip install opencv-python matplotlib imageio gdown tensorflow
    import os

    import tensorflow as tf
    import numpy as np
    from typing import List
    from matplotlib import pyplot as plt
    import imageio
    !pip3 install opencv-python matplotlib imageio gdown tensorflow
    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 gdown
    model.load_weights('models/checkpoint')
    model.save("lipbud.h5")
    model.summary()
    !tar -zcvf lipbud.tgz lipbud.h5
    model_details=client.repository.store_model(model="lipbud.tgz",meta_props={
```

```
        client.repository.ModelMetaNames.NAME:"lipbud.h5",
        client.repository.ModelMetaNames.TYPE:"tensorflow_2.9",
        client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_space_uid
    })
    model_id=client.repository.get_model_id(model_details)
    model_id
    client.repository.download(model_id,'lipbud2.tgz')
    import numpy as np
    import os
    from tensorflow.keras.models import load_model
    from tensorflow.keras.preprocessing import image
    from flask import Flask , request, render_template
    !pip3 install flask
    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
    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()})"
    )
    char_to_num.get_vocabulary()
    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 = '/Users/chaitanyatandon/Desktop/LIPBUDDIES/data/s1/bbal6n.mpg'
    tf.convert_to_tensor(test_path).numpy().decode('utf-8').split('/')[-1].split('.')[0]
    frames, alignments = load_data(tf.convert_to_tensor(test_path))
    plt.imshow(frames[40])
    alignments
    tf.strings.reduce_join([bytes.decode(x) for x in num_to_char(alignments.numpy()).numpy()])
    def mappable_function(path:str) ->List[str]:
        result = tf.py_function(load_data, [path], (tf.float32, tf.int64))
        return result
    from matplotlib import pyplot as plt
    sample = load_data(tf.convert_to_tensor('/Users/chaitanyatandon/Desktop/LIPBUDDIES/data/s1/bras9a.mpg'))
    print('~'*100, 'REAL TEXT')
    [tf.strings.reduce_join([num_to_char(word) for word in sentence]) for sentence in [sample[1]]]

    yhat = model.predict(tf.expand_dims(sample[0], axis=0))
    decoded = tf.keras.backend.ctc_decode(yhat, input_length=[75], greedy=True)[0][0].numpy()
    print('~'*100, 'PREDICTIONS')
    [tf.strings.reduce_join([num_to_char(word) for word in sentence]) for sentence in decoded]

    model.save("lipbud.h5")

```