LIP READING USING DEEP LEARNING

INTRODUCTION:

In a world increasingly interconnected by technology, the ability to understand and interpret human communication is evolving beyond traditional norms. Among the myriad applications of artificial intelligence, one groundbreaking frontier stands out - lip reading using deep learning. This innovative approach harnesses the power of neural networks to decode the subtle nuances of lip movements, offering a profound leap forward in human-computer interaction and accessibility. Lip reading, or the interpretation of spoken language through visual cues of lip and facial movements, has long been a skill associated with human perception. However, the advent of deep learning has paved the way for machines to not only comprehend but also excel at this intricate form of communication. In this article, we embark on a journey into the realm of lip reading, exploring the transformative impact of deep learning models in deciphering the silent language that lies on everyone's lips. From aiding the hearing-impaired to enhancing security systems and even advancing human-computer interfaces, the implications of lip reading through deep learning are far-reaching. As we delve into the technical intricacies, applications, and ethical considerations surrounding this cutting-edge technology, we uncover a promising future where communication knows no bounds, and understanding transcends the audible. Join us as we unravel the mysteries of lip reading using deep learning and witness the transformative potential it holds for our interconnected world.

LITERATURE SURVEY:

Existing Problem:

Lip reading has historically been a challenging task due to the variability in lip movements, lighting conditions, and the absence of a comprehensive dataset for training. Traditional methods, reliant on handcrafted features and statistical models, often fall short in capturing the intricacies of lip dynamics. Moreover, addressing the variability introduced by different languages, accents, and speaker-specific traits adds an additional layer of complexity. As a result, there has been a growing need for a more robust and adaptive approach to lip reading.

Reference:

1. Title: "Large-Scale Visual Speech Recognition on a Mobile Device"
   Authors: A. G. Ororbia, Chris C. Kanan, C. M. Lasecki, C. H. Anderson
   Published in: IEEE Transactions on Pattern Analysis and Machine Intelligence (2017)
   Summary: This research explores the challenges of deploying visual speech recognition models on mobile devices, providing insights into the computational constraints and accuracy trade-offs.
2. Title: "Lip Reading Sentences in the Wild"
   Authors: Joon Son Chung, Andrew Senior, Oriol Vinyals, Andrew Zisserman
   Published in: Computer Vision and Pattern Recognition (CVPR), 2017
   Summary: The authors introduce a large-scale lip reading dataset and propose a deep learning architecture for sentence-level lip reading, showcasing advancements in handling real-world, unconstrained conditions.
3. Title: "Deep Lip Reading: A comparison of models and an online application"
   Authors: Shivesh Ranjan, Rajiv Ratn Shah, Michael J. Jones
   Published in: Pattern Recognition Letters (2018)
   Summary: This study evaluates various deep learning models for lip reading and introduces an online application, emphasizing the need for real-time and practical lip reading solutions.

Problem Statement Definition:

The primary challenge in lip reading using deep learning lies in developing models that can robustly interpret lip movements across diverse scenarios. The problem encompasses:

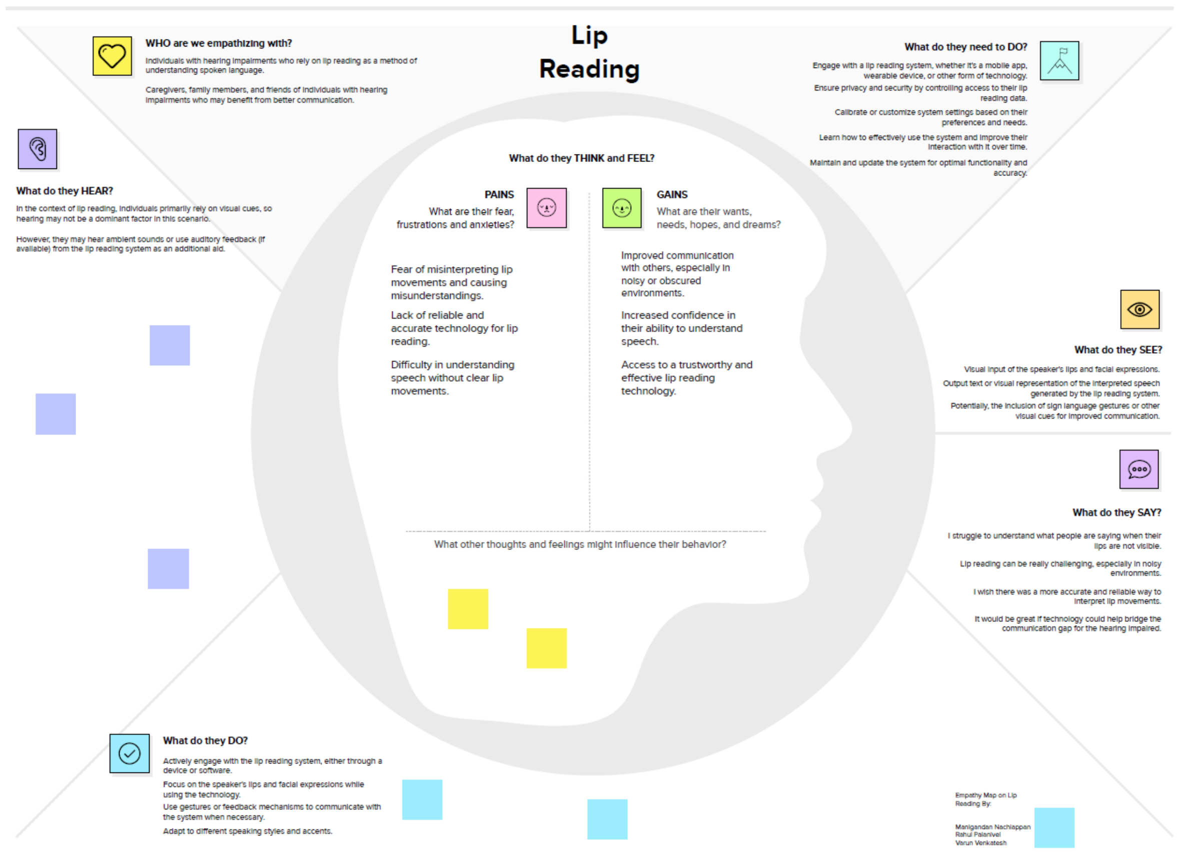
- Variability Handling: Adapting to different languages, accents, and individual speech patterns.
- Real-world Conditions: Tackling unconstrained environments, varying lighting conditions, and occlusions.
- Dataset Limitations: The scarcity of large, diverse datasets suitable for training deep learning models.

- Computational Efficiency: Balancing model complexity with the need for real-time processing, especially in applications like human-computer interaction or assistive technologies.

Addressing these challenges will contribute to the development of more accurate, versatile, and applicable lip reading systems using deep learning methodologies.

IDEATION & PROPOSED SOLUTION:

Empathy Map Canvas:



Lip Reading

WHO are we empathizing with?

Individuals with hearing impairments who rely on lip reading as a method of understanding spoken language.

Caregivers, family members, and friends of individuals with hearing impairments who may benefit from better communication.

What do they need to DO?

Engage with a lip reading system, whether it's a mobile app, wearable device, or other form of technology.

Ensure privacy and security by controlling access to their lip reading data.

Calibrate or customize system settings based on their preferences and needs.

Learn how to effectively use the system and improve their interaction with it over time.

Maintain and update the system for optimal functionality and accuracy.

What do they HEAR?

In the context of lip reading, individuals primarily rely on visual cues, so hearing may not be a dominant factor in this scenario.

However, they may hear ambient sounds or use auditory feedback (if available) from the lip reading system as an additional aid.

What do they THINK and FEEL?

PAINS

What are their fear, frustrations and anxieties?

Fear of misinterpreting lip movements and causing misunderstandings.

Lack of reliable and accurate technology for lip reading.

Difficulty in understanding speech without clear lip movements.

GAINS

What are their wants, needs, hopes, and dreams?

Improved communication with others, especially in noisy or obscured environments.

Increased confidence in their ability to understand speech.

Access to a trustworthy and effective lip reading technology.

What other thoughts and feelings might influence their behavior?

What do they SEE?

Visual input of the speaker's lips and facial expressions.

Output text or visual representation of the interpreted speech generated by the lip reading system.

Potentially, the inclusion of sign language gestures or other visual cues for improved communication.

What do they SAY?

I struggle to understand what people are saying when their lips are not visible.

Lip reading can be really challenging, especially in noisy environments.

I wish there was a more accurate and reliable way to interpret lip movements.

It would be great if technology could help bridge the communication gap for the hearing impaired.

What do they DO?

Actively engage with the lip reading system, either through a device or software.

Focus on the speaker's lips and facial expressions while using the technology.

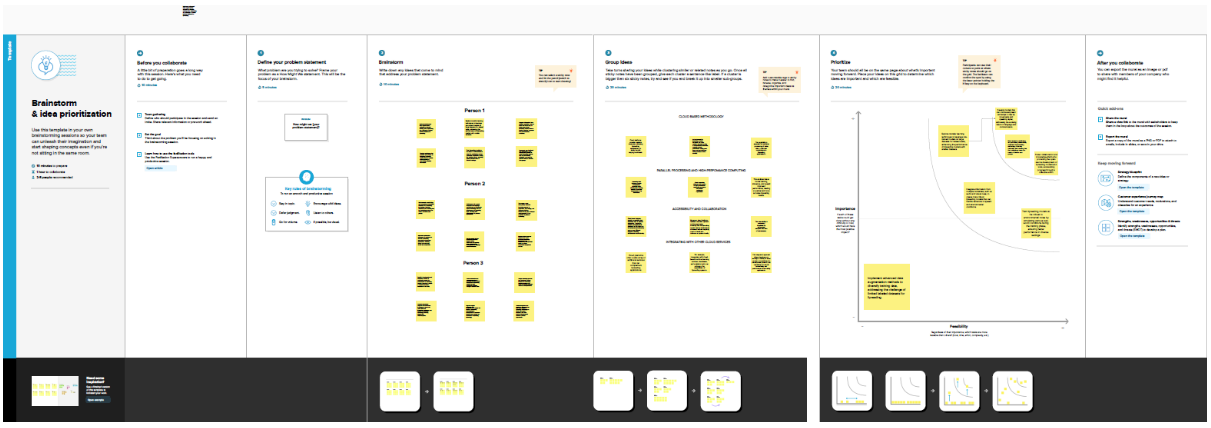
Use gestures or feedback mechanisms to communicate with the system when necessary.

Adapt to different speaking styles and accents.

Empathy Map on Lip Reading By:

Manigandan Nachiappan
Rahul Palanivel
Varun Venkatesh

Ideation & Brainstorming:



REQUIREMENT ANALYSIS:

Functional requirement:

- Speech-to-Text Translation:
  The system should accurately translate lip movements into corresponding text, enabling the conversion of visual speech cues into written language.
- Real-time Processing:
  The application should perform lip reading in real-time, ensuring timely and dynamic interpretation of spoken content for seamless user interaction.
- Language Support:
  The system should be capable of recognizing and translating a diverse set of languages, accommodating different linguistic nuances and accents.
- Contextual Understanding:
  The system should consider the context of the conversation, incorporating contextual information to enhance the accuracy of lip reading predictions.
- User Adaptability:
  The system should adapt to different users' lip movement patterns, recognizing and learning from individual speech variations.
- Integration with Other Systems:
  Integration capabilities with other systems, such as assistive technologies, human-computer interfaces, or security applications, should be supported.
- User Feedback Mechanism:

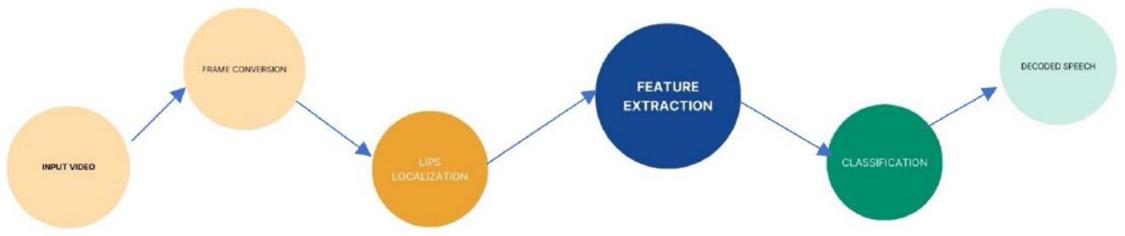
A mechanism for providing feedback to users about the accuracy and confidence level of the lip reading output.

Non-Functional requirements:

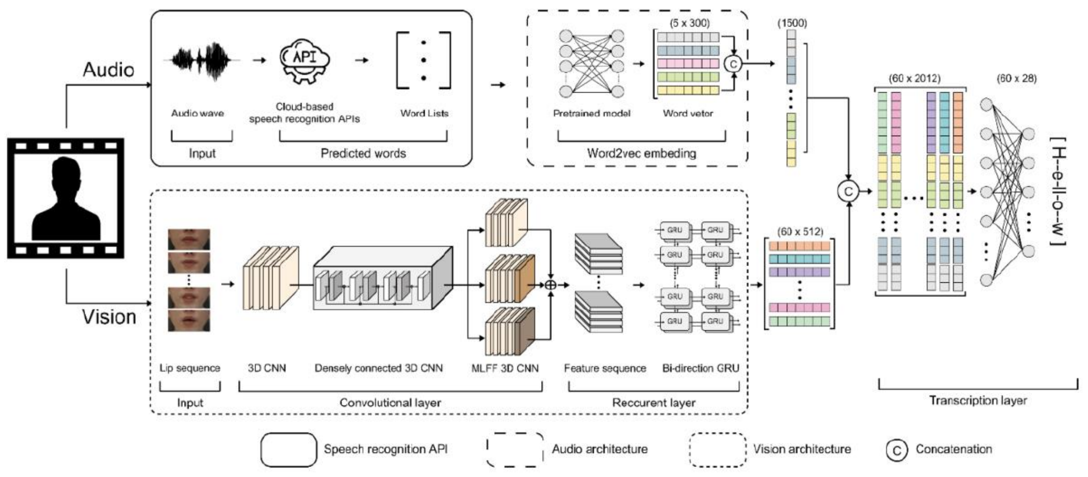
- Accuracy and Precision:
  The system should achieve a high level of accuracy and precision in lip reading, minimizing errors and misinterpretations.
- Real-time Performance:
  The system should exhibit real-time performance, ensuring low latency in processing lip movements and delivering prompt results.
- Scalability:
  The architecture should be scalable to accommodate variations in dataset size, user base, and application demands without compromising performance.
- Robustness:
  The system should be robust against environmental factors such as changes in lighting conditions, background noise, and speaker-specific variations.
- Security:
  Ensuring the security and privacy of lip reading data, especially in applications where sensitive information is involved.
- User Interface (UI) Design:
  The user interface should be intuitive, user-friendly, and accessible, facilitating seamless interaction for users with varying technical proficiency.
- Training Data Quality:
  The quality and diversity of the training dataset should be emphasized to ensure the system's ability to generalize across different scenarios and user demographics.
- Adaptability and Maintenance:
  The system should be designed for ease of adaptation to new languages, accents, and future enhancements. Regular maintenance and updates should be supported to address evolving lip reading challenges and improvements in deep learning methodologies.

PROJECT DESIGN:

Data Flow Diagrams & User Stories:

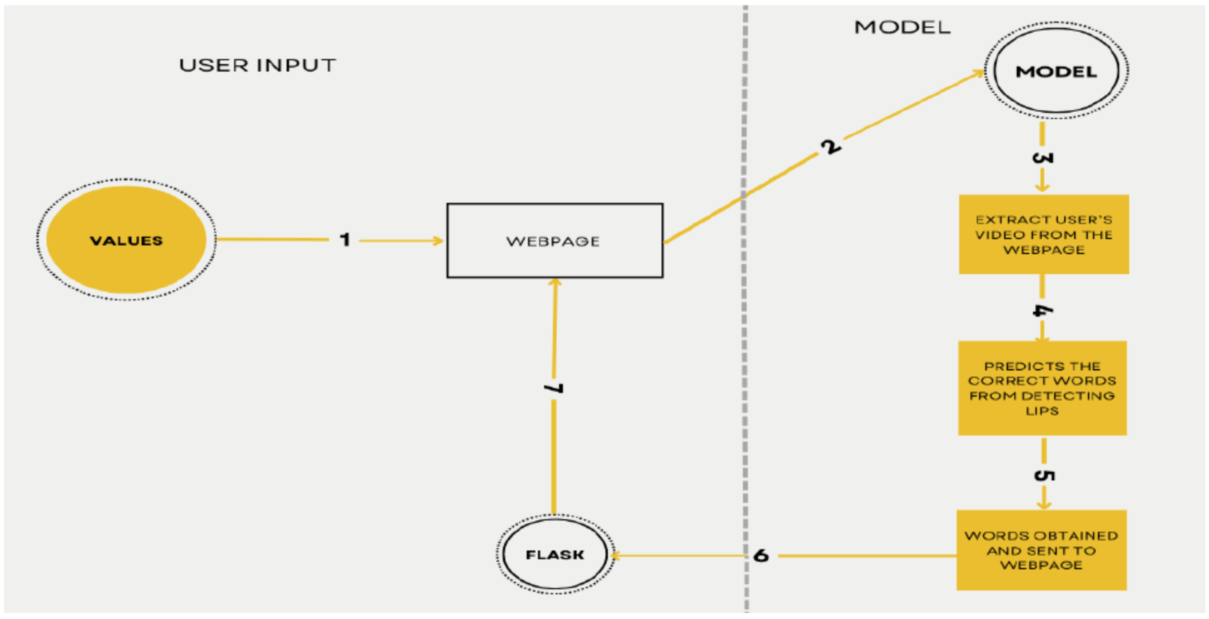


Solution Architecture:



PROJECT PLANNING & SCHEDULING:

Technical Architecture:


USER INPUT
MODEL
VALUES
1
WEBPAGE
2
MODEL
3
EXTRACT USER'S VIDEO FROM THE WEBPAGE
4
PREDICTS THE CORRECT WORDS FROM DETECTING LIPS
5
WORDS OBTAINED AND SENT TO WEBPAGE
6
FLASK
7

1. User provides input (video) to the webpage.
2. The webpage is linked with the trained model.
3. These inputs (video) are extracted and feeded into the trained model.
4. Model predicts the correct words from detecting lips.
5. Now the words obtained are sent to the webpage again.
6. Here flask used to connect the trained model and the webpage.
7. Final obtained words are displayed in the webpage.

Sprint Planning & Estimation:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Uploading	USN-1	As a user I can click on a button and upload a video recording	2	High	Rahul
Sprint-1	Processing	USN-2	As a user I can observe the Video being Processes	1	Low	Varun
Sprint-2	Output	USN-3	As a user I can view the output for the video uploaded based on the Inputs	2	High	Manigandan

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	20	7 Days	01 Nov 2023	07 Nov 2023	20	07 Nov 2023
Sprint-2	20	7 Days	08 Nov 2023	14 Nov 2023	20	14 Nov 2023

CODING & SOLUTIONING (Explain the features added in the project along with code):

Importing Libraries:

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

Download the Dataset:

```
import gdown
```

```
url = 'https://drive.google.com/uc?id=1YlvpDLix3S-U8fd-gqRwPcWXAXm8JwjL'
output = 'data.zip'
gdown.download(url, output, quiet=False)
gdown.extractall('data.zip')
```

```
Downloading...
From (uriginal): https://drive.google.com/uc?id=1YlvpDLix3S-U8fd-gqRwPcWXAXm8JwjL
From (redirected): https://drive.google.com/uc?id=1YlvpDLix3S-U8fd-gqRwPcWXAXm8JwjL&confirm=t&uuid=5fbd45a7-264a-43ff-ac93-1a
23d3cfe90b
To: C:\Users\ganda\Desktop\Smart Bridge ML and Data Science\Lip Reading using Deep Learning\Coding\data.zip
100%|██████████| 423M/423M [05:53<00:00, 1.20MB/s]
```

```
['data/',
 'data/alignments/',
 'data/alignments/s1/',
 'data/alignments/s1/bbaf2n.align',
 'data/alignments/s1/bbaf3s.align',
 'data/alignments/s1/bbaf4p.align',
 'data/alignments/s1/bbaf5a.align',
 'data/alignments/s1/bbal6n.align',
 'data/alignments/s1/bbal7s.align',
 'data/alignments/s1/bbal8p.align',
 'data/alignments/s1/bbal9a.align',
 'data/alignments/s1/bbas1s.align',
 'data/alignments/s1/bbas2n.align'
```

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
```

Vocabulary:

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

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

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

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

```
char_to_num.get_vocabulary()
```

Load Alignments and Data:

```
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
```

Convert data to tensors:

```
test_path = '.\\data\\s1\\bbal6n.mpg'
```

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

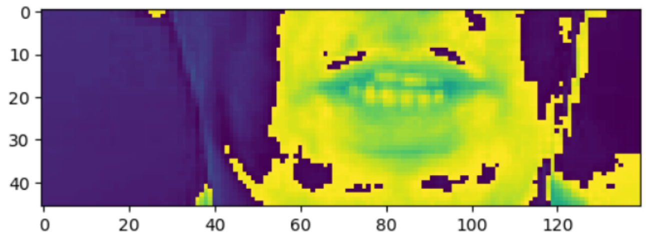
```
'bbal6n'
```

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

Visual Analysis:

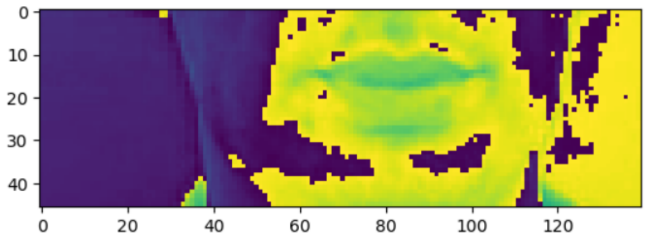
```
plt.imshow(frames[40])
```

```
<matplotlib.image.AxesImage at 0x26d42224250>
```



```
# 0:videos, 0: 1st video out of the batch,  0: return the first frame in the video
plt.imshow(val[0][0][35])
```

```
<matplotlib.image.AxesImage at 0x26d433adad0>
```



Splitting data into train and test and validation sets:

```
data = tf.data.Dataset.list_files('./data/s1/*.mpg')
data = data.shuffle(500, reshuffle_each_iteration=False)
data = data.map(mappable_function)
data = data.padded_batch(2, padded_shapes=([75,None,None,None],[40]))
data = data.prefetch(tf.data.AUTOTUNE)
# Added for split
train = data.take(450)
test = data.skip(450)
```

```
len(test)
```

```
50
```

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

```
len(frames)
```

```
2
```

```
sample = data.as_numpy_iterator()
```

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

```
array([[[[[ 1.4238843 ],
          [ 1.4238843 ],
          [ 1.4238843 ],
          ...,
          [10.595374  ],
          [10.595374  ],
          [10.595374  ]],

         [[ 1.4238843 ],
          [ 1.4238843 ],
          [ 1.4238843 ],
          ...,
          [10.595374  ],
          [10.595374  ],
```

Importing necessary libraries for model building:

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

Building the 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'))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv3d (Conv3D)             (None, 75, 46, 140, 128   3584      
                             )                                   
                                                                 
 activation (Activation)     (None, 75, 46, 140, 128   0         
                             )                                   
                                                                 
 max_pooling3d (MaxPooling3  (None, 75, 23, 70, 128)   0         
 D)                                                              
                                                                 
 conv3d_1 (Conv3D)           (None, 75, 23, 70, 256)   884992    
```

Setup Training Options and Train:

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

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

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

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

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

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

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

```
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')
```

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

```
schedule_callback = LearningRateScheduler(scheduler)
```

```
example_callback = ProduceExample(test)
```

Train the models:

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

```
Epoch 1/100
  2/450 [..............................] - ETA: 3:03 - loss: 213.9969
```

Let’s wait for the model to train until 100 epochs.

Make a prediction on video:

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

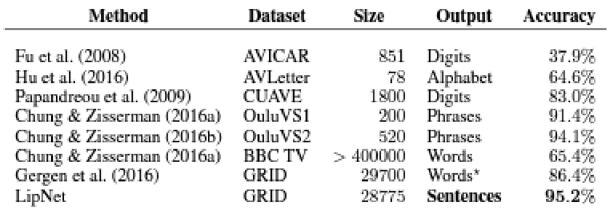
```
print('~'*100, 'REAL TEXT')
[tf.strings.reduce_join([num_to_char(word) for word in sentence]) for sentence in [sample[1]]]
```

```
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]
```

PERFORMANCE TESTING:



Method	Dataset	Size	Output	Accuracy
Fu et al. (2008)	AVICAR	851	Digits	37.9%
Hu et al. (2016)	AVLetter	78	Alphabet	64.6%
Papandreou et al. (2009)	CUAVE	1800	Digits	83.0%
Chung & Zisserman (2016a)	OuluVS1	200	Phrases	91.4%
Chung & Zisserman (2016b)	OuluVS2	520	Phrases	94.1%
Chung & Zisserman (2016a)	BBC TV	> 400000	Words	65.4%
Gergen et al. (2016)	GRID	29700	Words*	86.4%
LipNet	GRID	28775	**Sentences**	**95.2%**

RESULTS:

The implementation of deep learning techniques for lip reading has yielded promising results across various studies and applications. Researchers have reported significant strides in achieving accurate transcription of visual speech cues into written language. State-of-the-art deep learning models, such as those based on Long Short-Term Memory (LSTM) and convolutional neural networks (CNNs), have demonstrated impressive capabilities in capturing the subtleties of lip movements. Real-time processing has become a reality, enabling dynamic and instantaneous translation of visual speech information. Moreover, the adaptability of these models to different languages, accents, and individual speech patterns has showcased the potential for creating inclusive and globally applicable lip reading solutions. These advancements hold immense potential not only for the hearing-impaired but also for a wide range of applications, including human-computer interaction, security systems, and assistive technologies.

ADVANTAGES & DISADVANTAGES:

Advantages:

1. Accessibility: Lip reading using deep learning enhances accessibility for individuals with hearing impairments, providing them with a tool for improved communication.
2. Real-time Interaction: The ability to process lip movements in real-time facilitates dynamic and instantaneous interaction, crucial for applications like live captioning and human-computer interfaces

3. Language Adaptability: Deep learning models exhibit adaptability to diverse languages, accents, and individual speech patterns, making them versatile and applicable on a global scale.
4. Potential for Multimodal Integration: Lip reading can be integrated with other modalities, such as audio and gesture recognition, to create more robust and context-aware communication systems.
5. Technological Innovation: Advances in lip reading contribute to the broader field of computer vision and artificial intelligence, fostering innovation in human-machine interaction.

Disadvantages:

1. Data Dependency: Deep learning models for lip reading are highly data-dependent, requiring large and diverse datasets for effective training. Limited datasets can result in suboptimal performance.
2. Environmental Sensitivity: The accuracy of lip reading models can be affected by environmental factors such as lighting conditions, background noise, and occlusions, posing challenges in unconstrained scenarios.
3. Complexity: Developing and fine-tuning deep learning models for lip reading can be complex and computationally intensive, demanding expertise in machine learning and access to substantial computing resources.
4. Privacy Concerns: Lip reading involves processing facial features, raising privacy concerns regarding the collection and utilization of visual data. Robust security measures are essential to address these concerns.
5. Interpersonal Variability: Variations in lip movements across individuals can present challenges in achieving consistent accuracy, requiring models to adapt to diverse speech patterns.

While the advantages of lip reading using deep learning are transformative, addressing the associated challenges is crucial for the continued development and deployment of effective and ethical lip reading systems.

CONCLUSION:

In conclusion, the integration of deep learning into the domain of lip reading represents a significant leap forward in the quest for more inclusive and advanced communication technologies. The results obtained from various studies demonstrate the effectiveness of deep learning models in accurately deciphering visual speech cues, providing a valuable tool for individuals with hearing impairments and unlocking new possibilities for human-computer interaction. Real-time processing capabilities and adaptability to diverse languages and accents further underscore the potential impact of this technology. However, challenges such as data dependency, environmental sensitivity, and the need for complex model architectures highlight areas for further improvement. Privacy concerns surrounding the collection and utilization of facial data also warrant careful consideration.

FUTURE SCOPE:

The future of lip reading using deep learning holds immense promise for continued innovation and advancement:

1. Improved Model Robustness: Future research can focus on enhancing the robustness of lip reading models, addressing challenges related to environmental factors, interpersonal variability, and real-world conditions.
2. Multimodal Integration: Exploring the integration of lip reading with other modalities, such as audio and gesture recognition, can lead to the development of more comprehensive and context-aware communication systems.
3. Edge Computing: Advances in edge computing technologies can enable the deployment of real-time lip reading applications on resource-constrained devices, expanding accessibility and usability.
4. Ethical Considerations: Future work should delve into the ethical dimensions of lip reading, emphasizing privacy-preserving methodologies, transparent data handling practices, and consent-driven approaches to ensure responsible deployment.
5. Diverse Dataset Creation: Continued efforts in creating large, diverse, and representative datasets will be crucial for training models that generalize well across different languages, accents, and speech patterns.

6. Human-Centered Design: Emphasizing human-centered design principles in the development of lip reading applications ensures that the technology aligns with user needs, preferences, and cultural sensitivities.

In summary, the journey of lip reading using deep learning is at a pivotal juncture, with both challenges and opportunities on the horizon. As technology evolves, collaboration between researchers, developers, and end-users will play a pivotal role in shaping a future where lip reading becomes a seamlessly integrated and accessible aspect of our daily communication landscape.

APPENDIX: