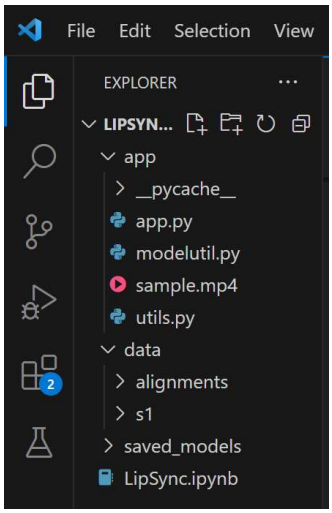
PROJECT MANUAL

Team Members

Rakesh Indupuri – 21BCE9838

Pavan Kumar Paidi – 21BCE9353

Project Structure:



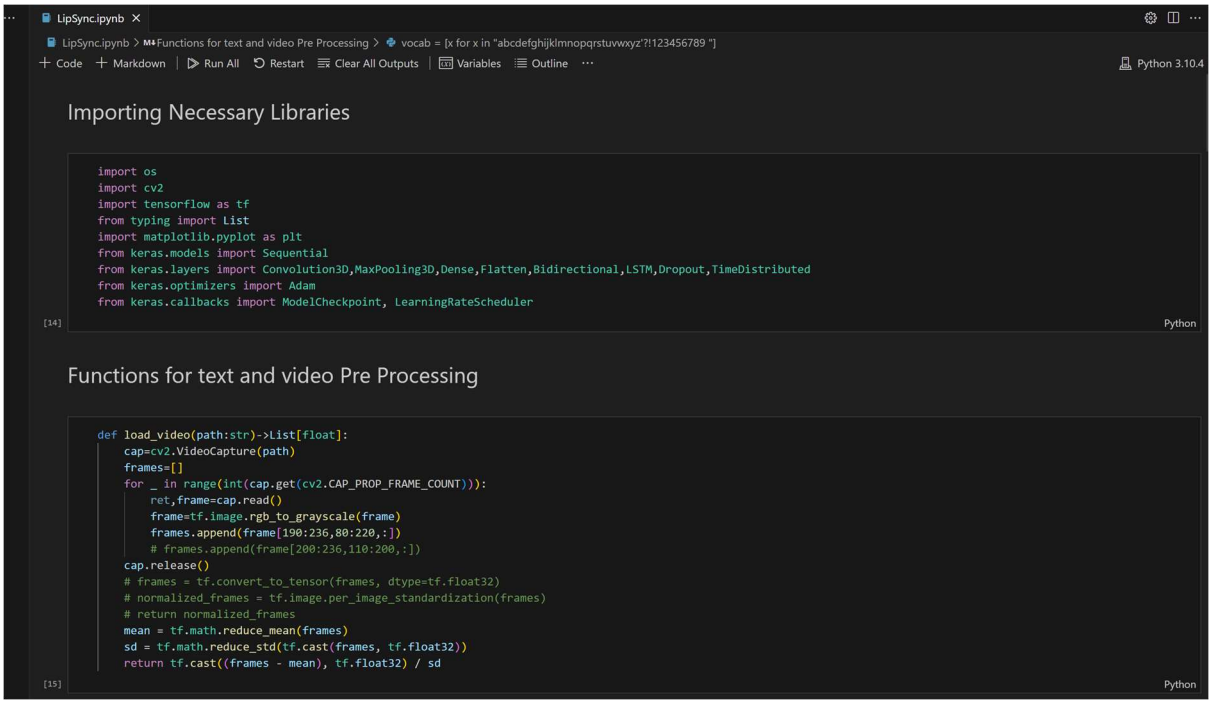
1)app folder contains the app.py which is used for building web site for our model it also contains the utils.py and modelutil.py. modelutil.py is used for loading the model and utils.py contains the functions used in website building.

2)Saved_models folder contains the models saved after the model training.

3)The LipSync.ipynb notebook will have the model which we are building using the Deep Learning.

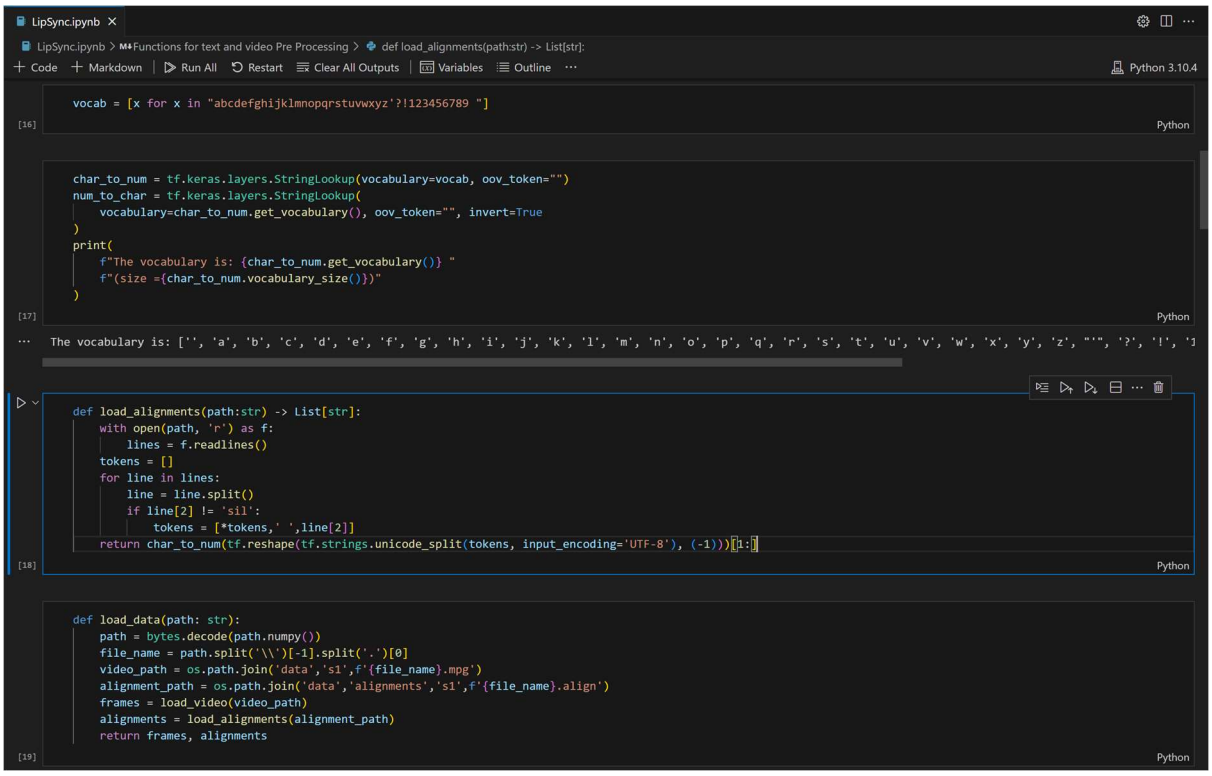
4)data folder contains s1 and alignments folders. S1 folder contains the videos for the training and alignments contains the annotations of the video.

MODEL (LipSync.ipynb):



In First cell we are importing all the necessary libraries used for the model building.

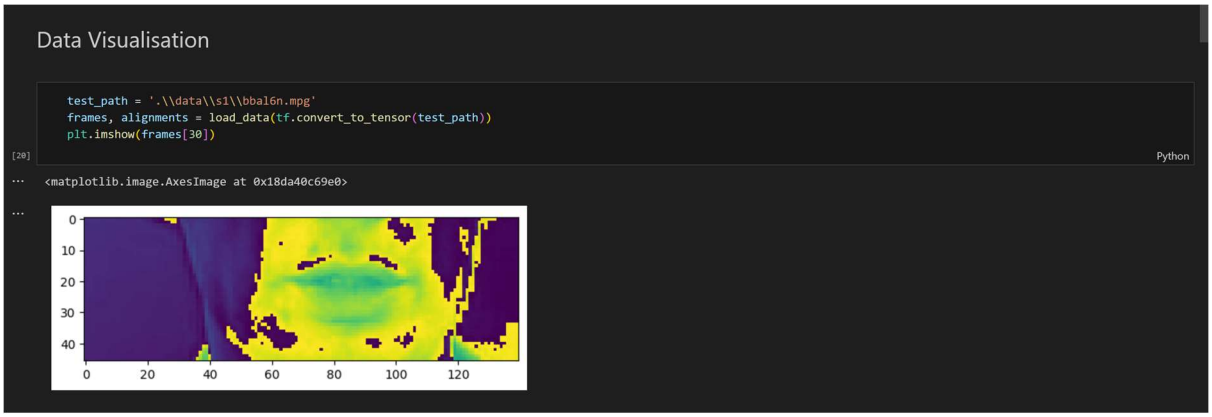
In second cell we are using a load_video(path) used for load video of the particular path , in this we are using the open cv library to capture the frames of the video and we are trimming the frames to capture the particularly lips region of the each frame and then we are normalising the frames for optimisation.

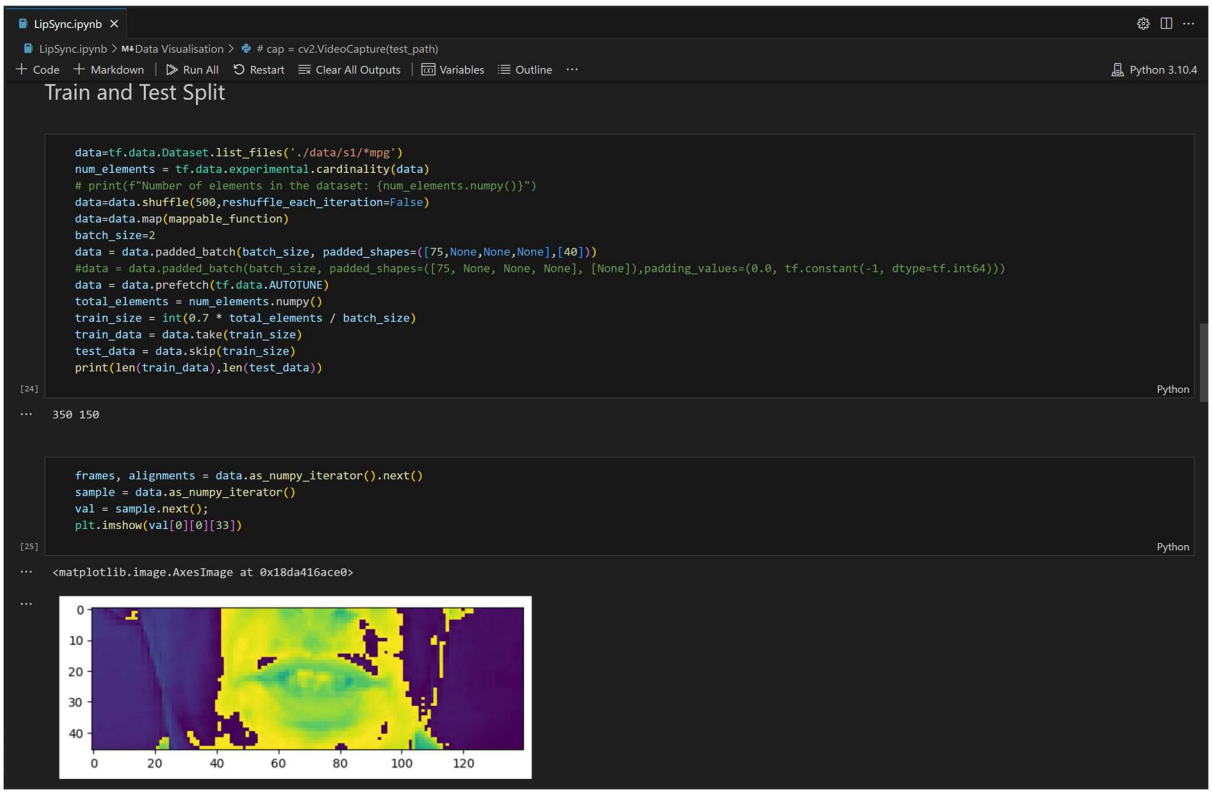


We are using char to num and num to char functions ,as the model will take the inputs of numbers these are used in the further parts.

load_alignments(path) function is used for the loading of captions or annotations of the video with given path

load_data(path) is used to load video and captions.





We are splitting the data into two batches and training and testing with 70% and 30% of the data in batch.

Model Building

```
model=Sequential()
model.add(Convolution3D(128,3,input_shape=(75,46,140,1),padding="same",activation="relu"))
model.add(MaxPooling3D(pool_size=(1,2,2)))
model.add(Convolution3D(256,3,padding="same",activation="relu"))
model.add(MaxPooling3D(pool_size=(1,2,2)))
model.add(Convolution3D(75,3,padding="same",activation="relu"))
model.add(MaxPooling3D(pool_size=(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(128,activation="relu"))
model.add(Dense(char_to_num.vocabulary_size()+1, kernel_initializer="he_normal", activation="softmax"))
model.output_shape
```

[28] Python

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

We are using CNN and RNN algorithms for building this model.

```
model.summary()
```

[29] Python

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

 max_pooling3d (MaxPooling3  (None, 75, 23, 70, 128)   0
 D)

 conv3d_1 (Conv3D)           (None, 75, 23, 70, 256)   884992

 max_pooling3d_1 (MaxPoolin  (None, 75, 11, 35, 256)   0
 g3D)

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

 max_pooling3d_2 (MaxPoolin  (None, 75, 5, 17, 75)     0
 g3D)

 time_distributed (TimeDist  (None, 75, 6375)          0
 ributed)

 bidirectional (Bidirection  (None, 75, 256)           6660096
 al)
...
Total params: 8499572 (32.42 MB)
Trainable params: 8499572 (32.42 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Training and Saving the Model

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

[ ] Python

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

[17] Python

```
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)
```

[ ] Python

CTC loss is the loss function is used In compiling model it is more appropriate loss function for this model.

```python
model.compile(optimizer=Adam(learning_rate=0.0001), loss=CTCLoss)
checkpoint_callback = ModelCheckpoint(os.path.join('saved_models','checkpoint'), monitor='loss', save_weights_only=True)
schedule_callback = LearningRateScheduler(scheduler)
example_callback = ProduceExample(test_data)
model.fit(train_data, validation_data=test_data, epochs=100, callbacks=[checkpoint_callback, schedule_callback, example_callback])
```

After 100 epochs the model will be saved in the saved_models folder with name checkpoint callbacks is used for optimising the model just as kind of early stopping.

Testing

```python
model.load_weights('saved_models/checkpoint')
```

```python
test= test_data.as_numpy_iterator()
sample = test.next()
y_pred = model.predict(sample[0])
```

```python
print('~'*100, 'Original Text')
[tf.strings.reduce_join([num_to_char(word) for word in sentence]) for sentence in sample[1]]
decoded = tf.keras.backend.ctc_decode(y_pred, input_length=[75,75], greedy=True)[0][0].numpy()
print('~'*100, 'Predicted Text')
[tf.strings.reduce_join([num_to_char(word) for word in sentence]) for sentence in decoded]
```

Original:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

[<tf.Tensor: shape=(), dtype=string, numpy=b'place white at x six please'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'lay blue in x four now'>]
```

Predicted:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

[<tf.Tensor: shape=(), dtype=string, numpy=b'place white at x six please'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'lay blue in x four now'>]
```

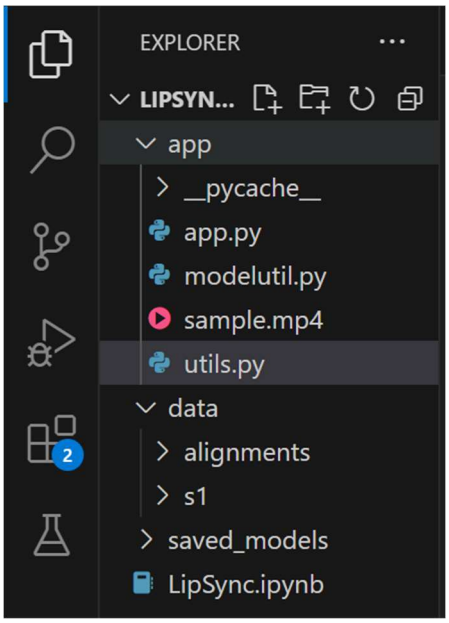
WEB PAGE BUILDING USING STREAMLIT:

```python
import streamlit as st
import os
import numpy as np
import tensorflow as tf
from utils import load_data, num_to_char
from modelutil import load_model


st.set_page_config(layout='wide')
with st.sidebar:
    st.image("https://cdn.pixabay.com/photo/2013/07/12/18/17/equalizer-153212_1280.png")
    st.title('LipSync Studio')
    st.info('This is made by using Deep Learning with help of CNN and RNN algorithms.')
    st.info('© Rakesh Indupuri and Pavan kumar Paidi.')

st.title('LipSync Studio')
options = os.listdir(os.path.join('..', 'data', 's1'))
selected_video = st.selectbox('Upload a video', options)
print("selected video is :"+selected_video)
col1, col2 = st.columns(2)

if options:
    with col1:
        st.info('Uploaded video:')
        file_path = os.path.join('..','data','s1', selected_video)
        conversion_command = f'ffmpeg -i {file_path} -vcodec libx264 sample.mp4 -y'
        conversion_result = os.system(conversion_command)

        if conversion_result == 0:
            st.video('sample.mp4')
        else:
            st.error('Video conversion failed. Check the FFmpeg command and try again.')
            video = open('test_video.mp4', 'rb')
            video_bytes = video.read()
            st.video(video_bytes)
    with col2:
        model = load_model()
        video, annotations = load_data(tf.convert_to_tensor(file_path))
        yhat = model.predict(tf.expand_dims(video, axis=0))
        decoder = tf.keras.backend.ctc_decode(yhat, [75], greedy=True)[0][0].numpy()
        st.info('Text spoken by person is:')
        converted_prediction = tf.strings.reduce_join(num_to_char(decoder)).numpy().decode('utf-8')
        st.markdown("Text generated from the video is : "+f'<span style="color: red;">{converted_prediction}</span>', unsafe_allow_html=True)
```

We are setting the layout of the webpage as wide using the streamlit library then we are setting up the sidebar with a image and some texts.

Next we are having two coulms in which one column is for selecting the video and diaplaying the selected video.Also as our video data is in form of mpg we are converting the mpg to mp4 with use of the ffmpeg library as stramlit library doesn't support playing the mpg format video.

Another column is for displaying the predicted text from video.



The video will be saved with sample.mp4 an this what we are playing in website by retrieving.

LipSync.ipynb | app.py | modelutil.py

app > modelutil.py > load_model

```python
import os
from keras.models import Sequential
from keras.layers import Conv3D, LSTM, Dense, Dropout, Bidirectional, MaxPool3D, Activation,TimeDistributed, Flatten

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('..','saved_models','checkpoint'))

    return model
```

LipSync.ipynb | app.py | utils.py

app > utils.py > ...

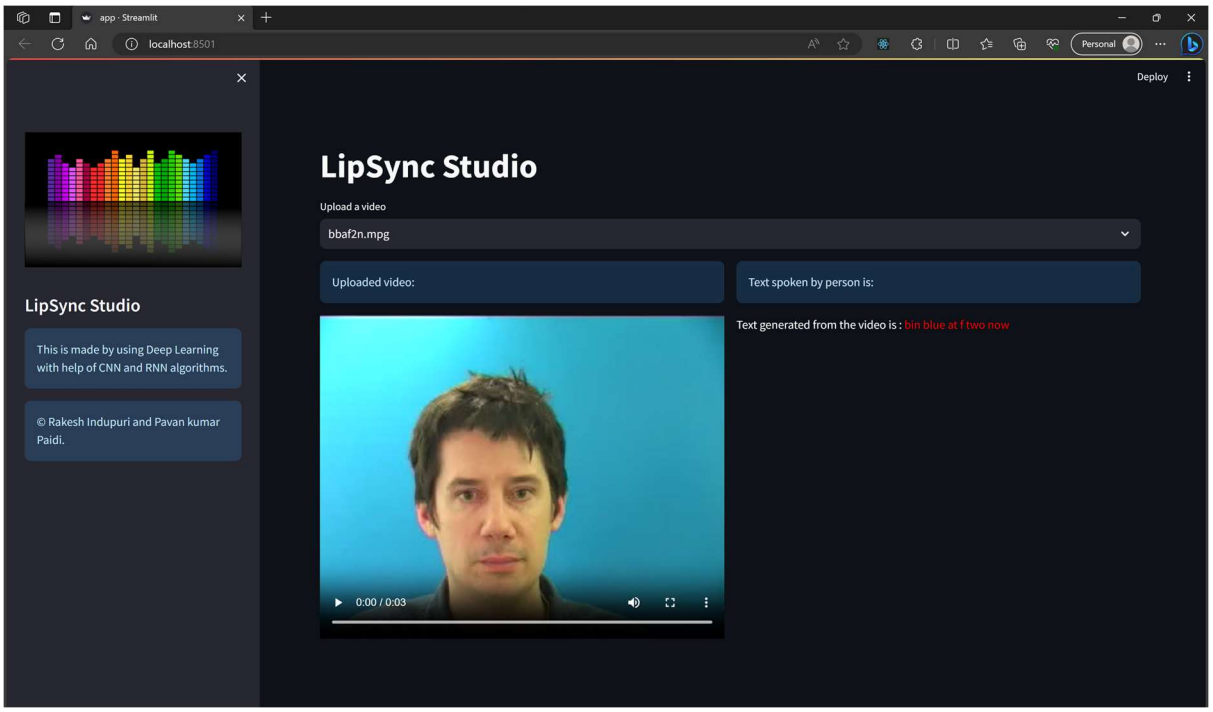
```python
import tensorflow as tf
from typing import List
import cv2
import os

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
)

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

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]
    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
```

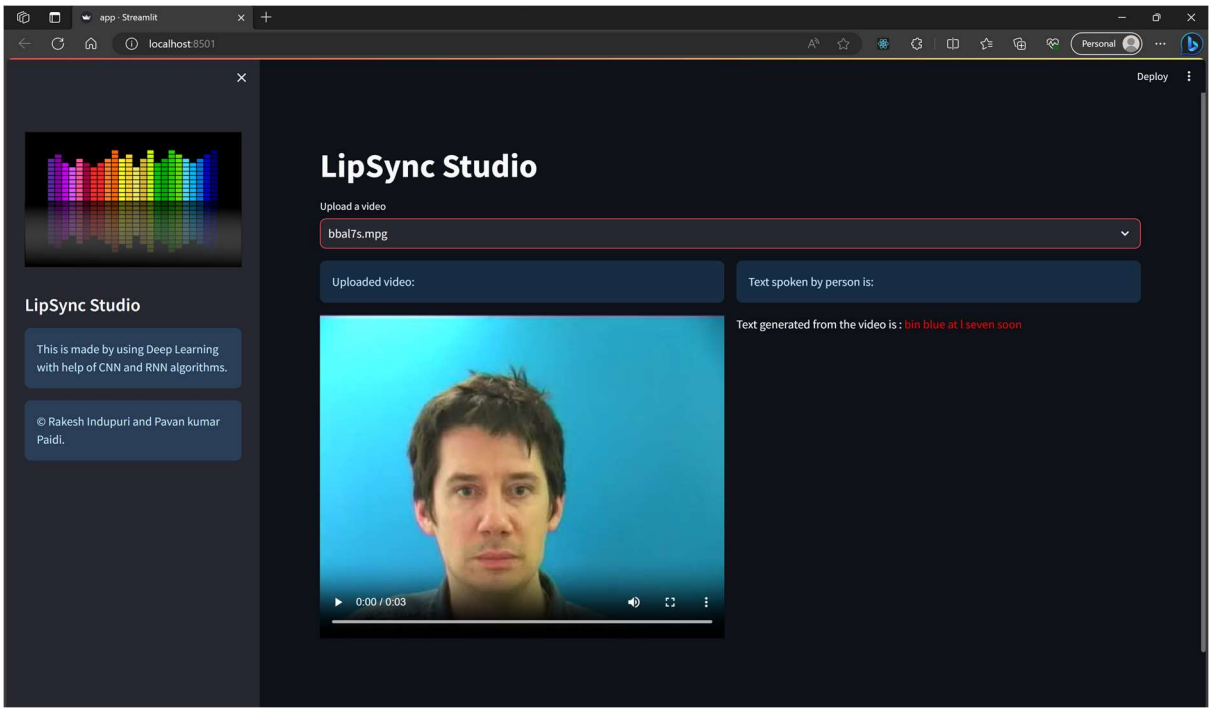
These are the same functions we are used in model building.

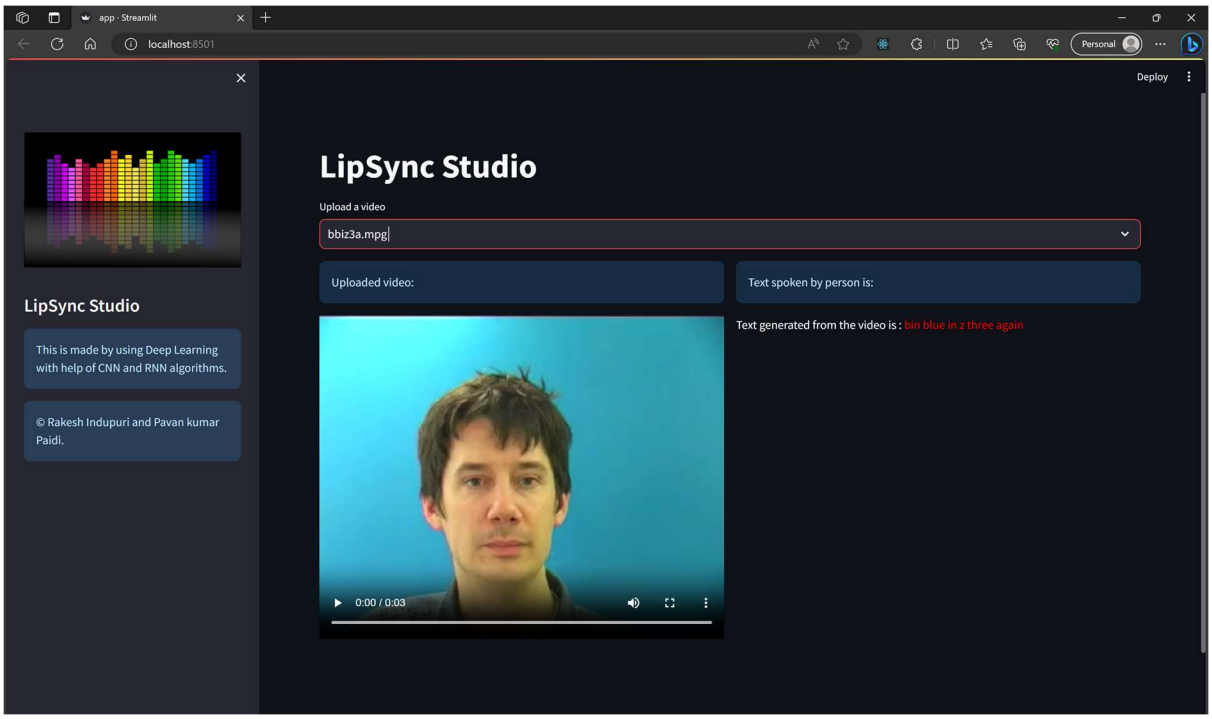
WEBSITE:

```
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enable it, run
'Import-Module PSReadLine'.

PS C:\Users\indup\OneDrive - vitap.ac.in\Desktop\Project_Development_Phase\LipSyncStudio> cd app
PS C:\Users\indup\OneDrive - vitap.ac.in\Desktop\Project_Development_Phase\LipSyncStudio\app> streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.104.11:8501

2023-11-23 18:42:47.809759: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in perf
ormance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX AVX2 AVX512F AVX512_VNNI FMA, in other operations, rebuild TensorFlow with the appropria
```

This is how we will run the website






app · Streamlit
localhost:8501
Personal
Deploy
LipSync Studio
This is made by using Deep Learning with help of CNN and RNN algorithms.
© Rakesh Indupuri and Pavan kumar Paidi.
LipSync Studio
Upload a video
bbiz3a.mpg
Uploaded video:
Text spoken by person is:
0:00 / 0:03
Text generated from the video is : bin blue in z three again