LipSync.ai

Introduction: In a world striving for inclusivity, accessibility is paramount. Communication barriers can isolate individuals from fully participating in society. Recognizing this, our project seeks to harness the power of machine learning to break down one such barrier: the communication gap between the hearing-impaired and others.

Objective: Our aim is to develop an advanced lip reading model using state-of-the-art deep learning techniques. By leveraging computer vision and deep neural networks, we endeavor to create a tool that not only enhances accessibility for the hearing-impaired but also empowers society to utilize machine learning for positive impact.

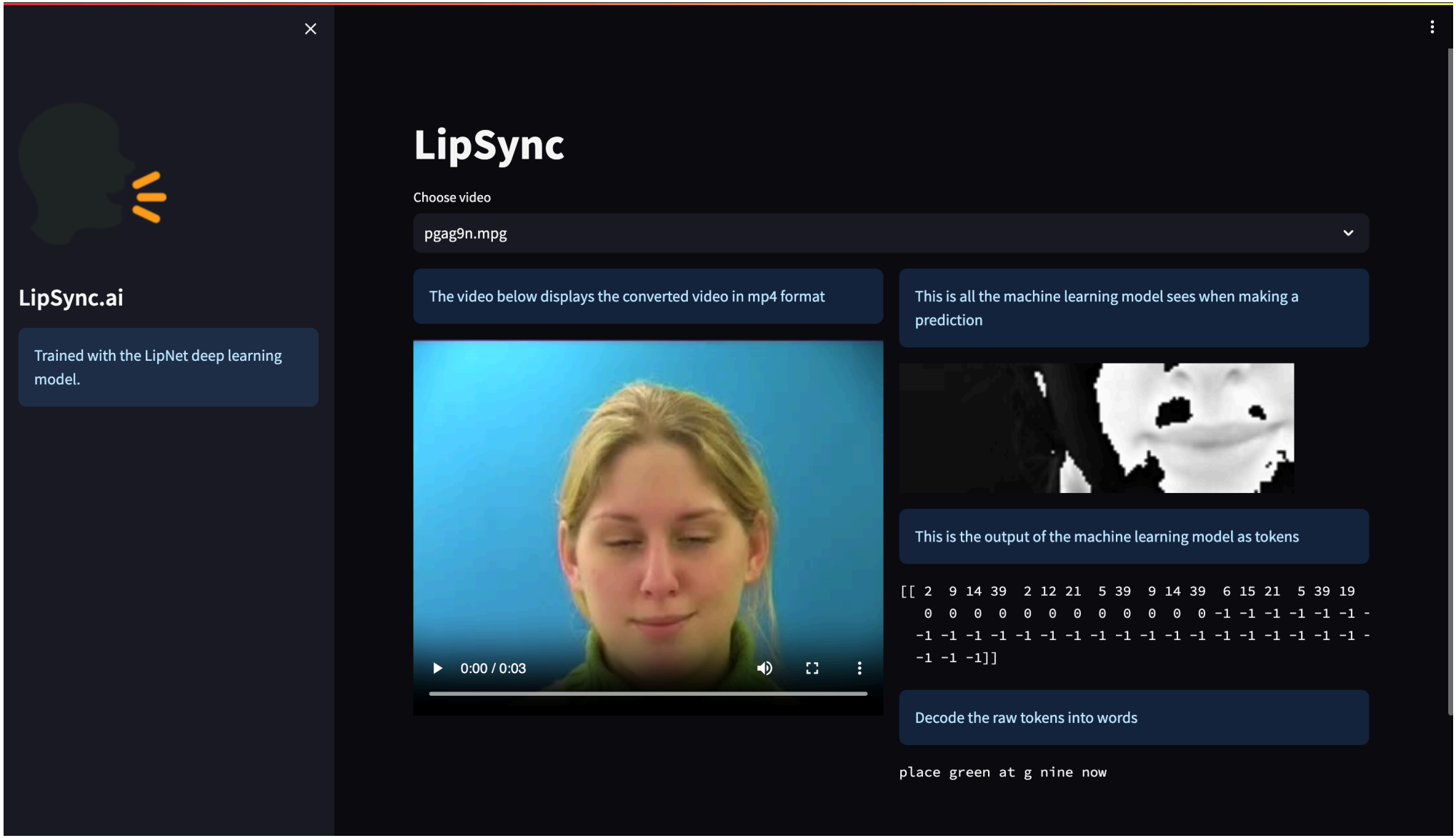
Technological Foundation: Our project integrates cutting-edge technologies, including OpenCV for video input processing, TensorFlow for building and training the deep learning model, and a client-conversation format for user interaction. We utilize Gdown for data acquisition and preprocessing, ensuring seamless integration and efficiency in our workflow.

Model Architecture: The core of our system lies in a deep neural network, employing 3D convolutions to extract meaningful features from video input. This architecture allows us to condense complex visual data into a classification dense layer, predicting characters accurately. To optimize training and inference, we incorporate a specialized loss function known as Connectionist Temporal Classification (CTC).

Data Pipeline and Training: A robust data pipeline, developed using TensorFlow, ensures smooth data flow from acquisition to model training. By automating data preprocessing and feeding it into the training pipeline, we streamline the development process. This approach not only enhances efficiency but also ensures scalability and reproducibility.

Outcome: The culmination of our efforts is a powerful lip reading model capable of converting video input into corresponding text. By harnessing the synergy of advanced machine learning techniques, we bridge the communication gap, fostering inclusivity and empowerment within our society.

Inferencing:



Refernces:

- https://arxiv.org/pdf/1611.01599.pdf?uuid=Fqbse38nqebdFpys3035
- https://keras.io/examples/audio/ctc_asr/

- https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv3D

0. Install and import Dependencies

```
! pip list
```

```
tzlocal                         5.2
uc-micro-py                     1.0.3
uritemplate                     4.1.1
urllib3                         2.0.7
vega-datasets                   0.9.0
wadllib                         1.3.6
wasabi                          1.1.2
wcwidth                         0.2.13
weasel                          0.3.4
webcolors                       1.13
webencodings                    0.5.1
websocket-client                1.7.0
Werkzeug                        3.0.2
wheel                           0.43.0
widgetsnbextension              3.6.6
wordcloud                       1.9.3
wrapt                           1.14.1
xarray                          2023.7.0
xarray-einstats                 0.7.0
xgboost                         2.0.3
xlrd                            2.0.1
xyzservices                     2024.4.0
yarl                            1.9.4
yellowbrick                     1.5
yfinance                        0.2.38
zict                            3.0.0
zipp                            3.18.1
```

```
!pip install opencv-python matplotlib imageio gdown tensorflow
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: imageio in /usr/local/lib/python3.10/dist-packages (2.31.6)
Requirement already satisfied: gdown in /usr/local/lib/python3.10/dist-packages (4.7.3)
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.10.0)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from opencv-python) (1.25.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (24.0)
```

```
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from gdown) (3.13.4)
Requirement already satisfied: requests[socks] in /usr/local/lib/python3.10/dist-packages (from gdown) (2.31.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from gdown) (1.16.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from gdown) (4.66.2)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (from gdown) (4.12.3)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.3.25)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.62.1)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)
Requirement already satisfied: keras<2.11,>=2.10.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.10.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.1.2
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.19.6)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: tensorboard<2.11,>=2.10 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.10.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorfl
Requirement already satisfied: tensorflow-estimator<2.11,>=2.10.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.11.0)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorf
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.11,>=2.10
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.11,>=2.10->tens
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorb
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.1
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.11,>=2.10->tens
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4->gdown) (2.5)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdo
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
```

```
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboar
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->g
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->goo
```

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

```
tf.config.list_physical_devices('GPU')
```

```
[]
```

```
physical_devices = tf.config.list_physical_devices('GPU')
try:
    tf.config.experimental.set_memory_growth(physical_devices[0], True)
except:
    pass
```

1. Build Data Loading Functions

```
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')
```

```
 'data/alignments/s1/swiu7a.align',
 'data/alignments/s1/swwc4n.align',
 'data/alignments/s1/swwc5s.align',
 'data/alignments/s1/swwc6p.align',
 'data/alignments/s1/swwc7a.align',
 'data/alignments/s1/swwi8n.align',
 'data/alignments/s1/swwi9s.align',
 'data/alignments/s1/swwj1a.align',
 'data/alignments/s1/swwjzp.align',
 'data/alignments/s1/swwp2n.align',
 'data/alignments/s1/swwp3s.align',
 'data/alignments/s1/swwp4p.align',
 'data/alignments/s1/swwp5a.align',
 'data/alignments/s1/swwv6n.align',
 ...]
```

```
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 "]
```

```
vocab
```

```
['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',
         ' ']
```

```
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',
```

```
char_to_num(['w','c','g','z'])
```

```
<tf.Tensor: shape=(4,), dtype=int64, numpy=array([23,  3,  7, 26])>
```

```
num_to_char([23, 3, 7, 26])
```

```
<tf.Tensor: shape=(4,), dtype=string, numpy=array([b'w', b'c', b'g', b'z'], dtype=object)>
```

```
char_to_num.get_vocabulary()
```

```
['',
 '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',
        ' ']
```

```
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
```

```
test_path = '/content/data/s1/bbal6n.mpg'
```

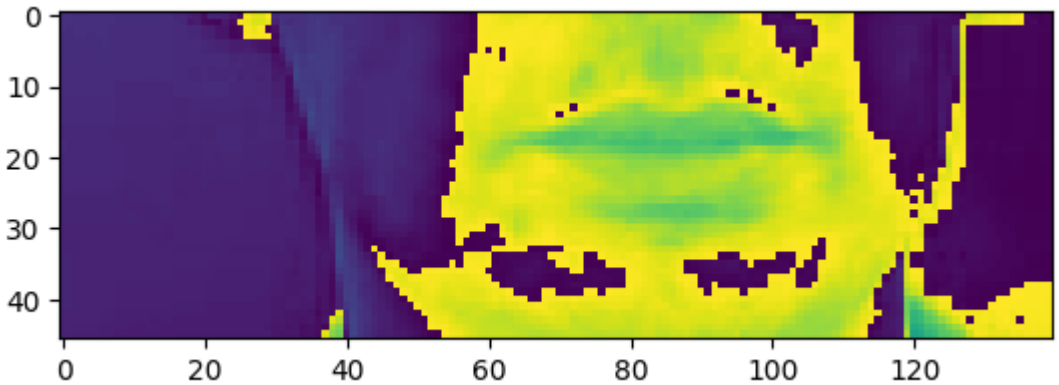
```
tf.convert_to_tensor(test_path).numpy().decode('utf-8').split('/')[-1].split('.')[0]
```

```
'bbal6n'
```

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

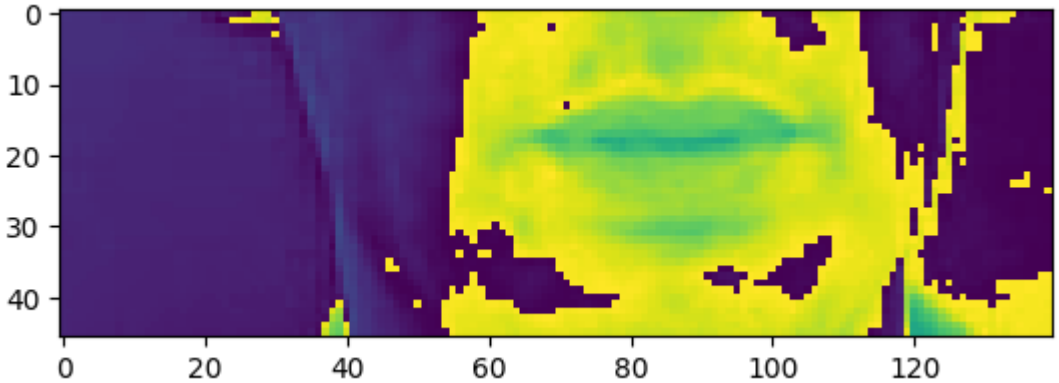
```
plt.imshow(frames[2])
```

```
<matplotlib.image.AxesImage at 0x7a3e400d7130>
```



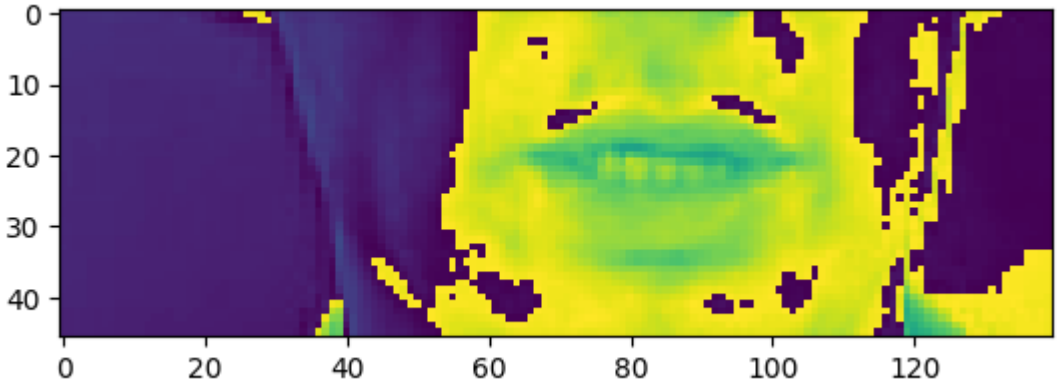

```
plt.imshow(frames[17])
```

```
<matplotlib.image.AxesImage at 0x7a3e2ffdf8b0>
```



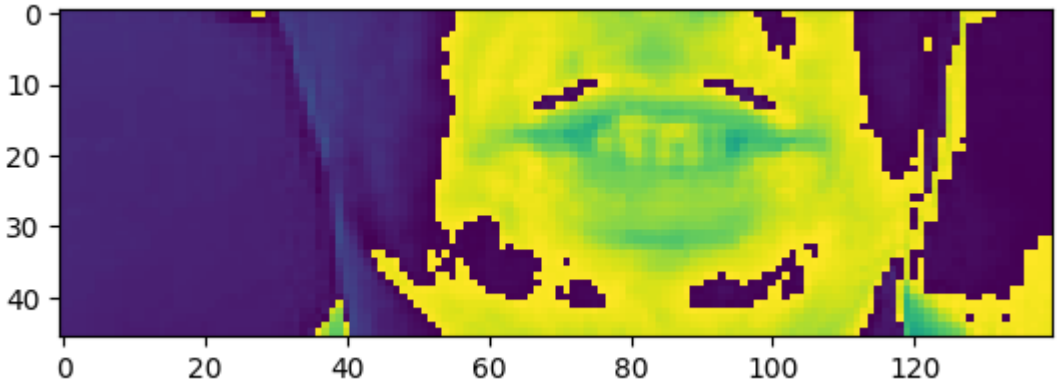

```
plt.imshow(frames[29])
```

```
<matplotlib.image.AxesImage at 0x7a3e29998790>
```



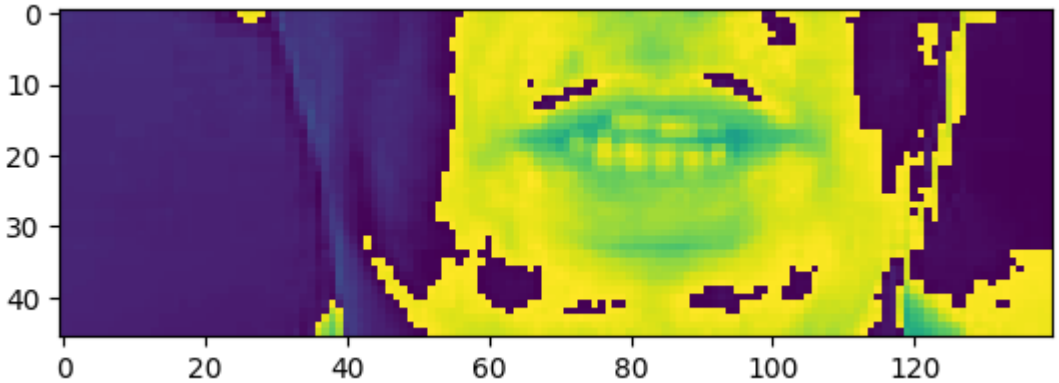

```
plt.imshow(frames[35])
```

```
<matplotlib.image.AxesImage at 0x7a3ee2b350f0>
```




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

```
<matplotlib.image.AxesImage at 0x7a3de8485690>
```




```
alignments
```

```
<tf.Tensor: shape=(21,), dtype=int64, numpy=
array([ 2,  9, 14, 39,  2, 12, 21,  5, 39,  1, 20, 39, 12, 39, 19,  9, 24,
       39, 14, 15, 23])>
```

```
print([bytes.decode(x) for x in num_to_char(alignments.numpy()).numpy()])
```

```
['b', 'i', 'n', ' ', 'b', 'l', 'u', 'e', ' ', 'a', 't', ' ', 'l', ' ', 's', 'i', 'x', ' ', 'n', 'o', 'w']
```

```
tf.strings.reduce_join([bytes.decode(x) for x in num_to_char(alignments.numpy()).numpy()])
```

```
<tf.Tensor: shape=(), dtype=string, numpy=b'bin blue at l six now'>
```

```
def mappable_function(path:str) ->List[str]:
    result = tf.py_function(load_data, [path], (tf.float32, tf.int64))
    return result
```

2. Create data Pipeline

```
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)
train = data.take(450)
test = data.skip(450)
```

```
len(test)
```

```
50
```

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

```
len(frames)
```

```
2
```

```
alignments
```

```
array([[ 2,  9, 14, 39, 18,  5,  4, 39, 23,  9, 20,  8, 39,  7, 39, 19,
         9, 24, 39, 14, 15, 23,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0],
       [19,  5, 20, 39,  7, 18,  5,  5, 14, 39,  9, 14, 39,  3, 39, 26,
         5, 18, 15, 39, 16, 12,  5,  1, 19,  5,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0]])
```

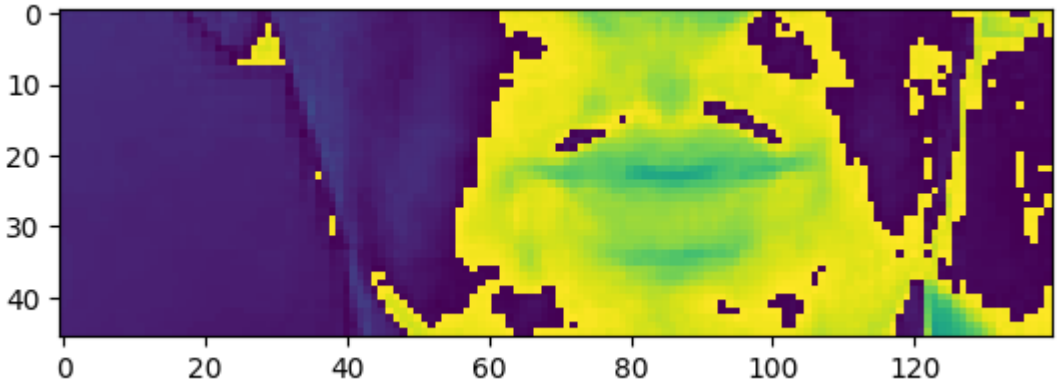
```
sample = data.as_numpy_iterator()
```

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

```
       [ 1.402874  ],
       [ 1.0706143 ],
       ...,
       [ 0.33225963],
       [ 0.33225963],
       [ 0.33225963]],

      ...,

      [[ 1.107532  ],
       [ 1.0706143 ],
       [ 1.0336967 ],
       ...,
       [ 0.07383547],
       [ 0.07383547],
       [ 0.07383547]],

      [[ 1.107532  ],
       [ 1.0336967 ],
       [ 0.99677885],
       ...,
       [ 0.07383547],
       [ 0.07383547],
       [ 0.03691773]],

      [[ 1.0706143 ],
       [ 1.0336967 ],
       [ 0.99677885],
       ...,
       [ 0.07383547],
       [ 0.03691773],
       [ 0.03691773]]]]], dtype=float32)
```

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

```
<matplotlib.image.AxesImage at 0x7a3de83acbe0>
```




```
tf.strings.reduce_join([num_to_char(word) for word in val[1][0]])
```

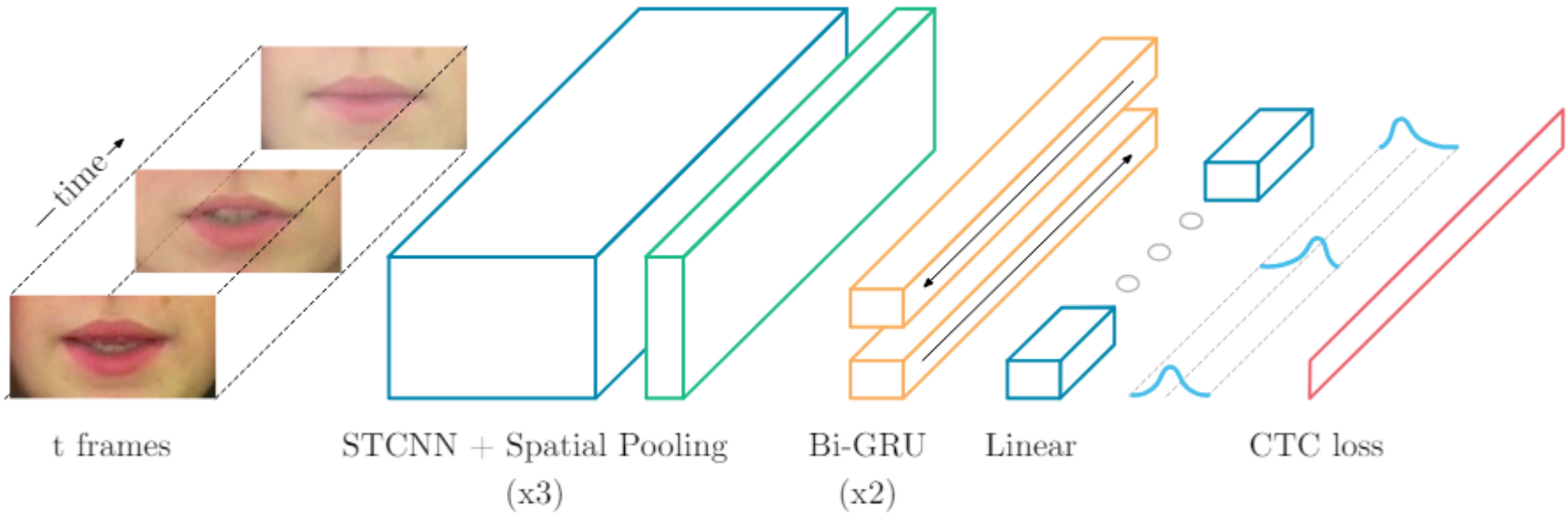
```
<tf.Tensor: shape=(), dtype=string, numpy=b'set white by c one soon'>
```

3. Design the Deep Neural Network

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

```
data.as_numpy_iterator().next()[0][0].shape
```

```
(75, 46, 140, 1)
```



- **LipNet architecture** A sequence of T frames is used as input, and is processed by 3 layers of STCNN, each followed by a spatial max-pooling layer. The features extracted are processed by 2 Bi-GRUs; each time-step of the GRU output is processed by a linear layer and a softmax. This end-to-end model is trained with CTC.

```
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 (MaxPooling3D  (None, 75, 23, 70, 128)  0
 )

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

 activation_1 (Activation)   (None, 75, 23, 70, 256)   0
```

```
 max_pooling3d_1 (MaxPooling  (None, 75, 11, 35, 256)  0
 3D)

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

 activation_2 (Activation)   (None, 75, 11, 35, 75)    0

 max_pooling3d_2 (MaxPooling  (None, 75, 5, 17, 75)    0
 3D)

 time_distributed (TimeDistr  (None, 75, 6375)         0
 ibuted)

 bidirectional (Bidirectiona  (None, 75, 256)          6660096
 l)

 dropout (Dropout)           (None, 75, 256)           0

 bidirectional_1 (Bidirectio  (None, 75, 256)          394240
 nal)

 dropout_1 (Dropout)         (None, 75, 256)           0

 dense (Dense)               (None, 75, 41)            10537

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

```
char_to_num.vocab_size()
```

```
WARNING:tensorflow:vocab_size is deprecated, please use vocabulary_size.
40
```

```
5*17*75
```

```
6375
```

```
yhat = model.predict(val[0])
```

```
1/1 [==============================] - 15s 15s/step
```

```
tf.strings.reduce_join([num_to_char(x) for x in tf.argmax(yhat[0],axis=1)])
```

```
<tf.Tensor: shape=(), dtype=string, numpy=b'88888888888888aaaaaaaaaaaaaaaa88888888888888888888888888888888888888aaaaaaa'>
```

```
yhat[0]
```

```
array([[0.02506956, 0.02532368, 0.01739012, ..., 0.02384461, 0.02869177,
        0.02334795],
       [0.02524308, 0.02624132, 0.0166348 , ..., 0.02367346, 0.02784845,
        0.02315191],
       [0.02517485, 0.02738713, 0.01626411, ..., 0.02387015, 0.02704719,
        0.02339865],
       ...,
       [0.02168335, 0.03577807, 0.0208487 , ..., 0.0242131 , 0.02703556,
        0.02512283],
       [0.02152027, 0.03488359, 0.02194611, ..., 0.02433235, 0.02663311,
        0.02547426],
       [0.02131919, 0.03352631, 0.02322174, ..., 0.02469384, 0.02594409,
        0.02587787]], dtype=float32)
```

```
yhat[0].shape
```

```
(75, 41)
```

```
tf.argmax(yhat[0],axis=1)
```

```
<tf.Tensor: shape=(75,), dtype=int64, numpy=
array([37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1, 37, 37, 37, 37,
       37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37,
```

```
            37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37,
             1,  1,  1,  1,  1,  1,  1])>
```

```
[num_to_char(x) for x in tf.argmax(yhat[0],axis=1)]
```

```
      <tf.Tensor: shape=(), dtype=string, numpy=b'a'>,
      <tf.Tensor: shape=(), dtype=string, numpy=b'a'>,
      <tf.Tensor: shape=(), dtype=string, numpy=b'a'>,
      <tf.Tensor: shape=(), dtype=string, numpy=b'a'>,
      <tf.Tensor: shape=(), dtype=string, numpy=b'a'>,
      <tf.Tensor: shape=(), dtype=string, numpy=b'a'>,
      <tf.Tensor: shape=(), dtype=string, numpy=b'a'>,
      <tf.Tensor: shape=(), dtype=string, numpy=b'a'>,
      <tf.Tensor: shape=(), dtype=string, numpy=b'a'>,
      <tf.Tensor: shape=(), dtype=string, numpy=b'a'>,
      <tf.Tensor: shape=(), dtype=string, numpy=b'a'>,
      <tf.Tensor: shape=(), dtype=string, numpy=b'a'>,
      <tf.Tensor: shape=(), dtype=string, numpy=b'a'>,
      <tf.Tensor: shape=(), dtype=string, numpy=b'8'>,
      <tf.Tensor: shape=(), dtype=string, numpy=b'8'>,
      <tf.Tensor: shape=(), dtype=string, numpy=b'8'>,
      <tf.Tensor: shape=(), dtype=string, numpy=b'8'>,
      <tf.Tensor: shape=(), dtype=string, numpy=b'8'>,
```

```
 <tf.Tensor: shape=(), dtype=string, numpy=b'8'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'8'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'8'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'8'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'8'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'8'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'8'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'8'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'8'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'8'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'8'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'8'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'8'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'8'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'8'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'8'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'a'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'a'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'a'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'a'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'a'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'a'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'a'>]
```

```
tf.strings.reduce_join([num_to_char(tf.argmax(x)) for x in yhat[0]])
```

```
<tf.Tensor: shape=(), dtype=string, numpy=b'88888888888888aaaaaaaaaaaaaaaa88888888888888888888888888888888888888aaaaaaa'>
```

```
model.input_shape
```

```
(None, 75, 46, 140, 1)
```

```
model.output_shape
```

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

4. 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)
```

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

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

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

Make a Prediction

```
import gdown
```

```
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')
```

```
Downloading...
From (original): https://drive.google.com/uc?id=1vWscXs4Vt0a_1IH1-ct2TCgXAZT-N3_Y
From (redirected): https://drive.google.com/uc?id=1vWscXs4Vt0a_1IH1-ct2TCgXAZT-N3_Y&confirm=t&uuid=92ea3337-2244-47a6-97bb-2d44f
To: /content/checkpoints.zip
100%|██████████| 94.5M/94.5M [00:00<00:00, 125MB/s]
['models/checkpoint.index',
 'models/__MACOSX/._checkpoint.index',
 'models/checkpoint.data-00000-of-00001',
 'models/__MACOSX/._checkpoint.data-00000-of-00001',
 'models/checkpoint',
 'models/__MACOSX/._checkpoint']
```

```
!pip install tensorflow==2.10 opencv-python matplotlib imageio gdown
```

```
Requirement already satisfied: opencv-python in /usr/local/lib/python3.10/dist-packages (4.8.0.76)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: imageio in /usr/local/lib/python3.10/dist-packages (2.31.6)
```

```
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (1.6.3)
Requirement already satisfied: flatbuffers>=2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (24.3.25)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (0.2.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (1.62.1
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (3.9.0)
Requirement already satisfied: keras<2.11,>=2.10.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (2.10.0
Requirement already satisfied: keras-preprocessing>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (18.1.1)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (1.25.2)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (24.0)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (3.19
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (1.16.0)
Requirement already satisfied: tensorboard<2.11,>=2.10 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (2.
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorfl
Requirement already satisfied: tensorflow-estimator<2.11,>=2.10.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (4
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (1.14.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from gdown) (3.13.4)
Requirement already satisfied: requests[socks] in /usr/local/lib/python3.10/dist-packages (from gdown) (2.31.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from gdown) (4.66.2)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (from gdown) (4.12.3)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorf
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.11,>=2.10
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.11,>=2.10->tens
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorb
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.1
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.11,>=2.10->tens
```

```
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorbo
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboar
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->g
```

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

```
<tensorflow.python.checkpoint.checkpoint.CheckpointLoadStatus at 0x7a3dd8a666b0>
```

```python
test_data = test.as_numpy_iterator()
```

```python
sample = test_data.next()
```

```python
yhat = model.predict(sample[0])
```

```
1/1 [==============================] - 11s 11s/step
```

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

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ REAL TEXT
[<tf.Tensor: shape=(), dtype=string, numpy=b'lay red in k four please'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'place sp blue at i six please'>]
```

```python
decoded = tf.keras.backend.ctc_decode(yhat, input_length=[75,75], greedy=True)[0][0].numpy()
```

```
print('~'*100, 'PREDICTIONS')
[tf.strings.reduce_join([num_to_char(word) for word in sentence]) for sentence in decoded]
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ PREDICTIONS
```