0. Install and Import Dependencies

In [127]:
```
!pip list
```

```
tornado                       6.2
tqdm                          4.65.0
traitlets                     5.9.0
typing_extensions             4.5.0
tzdata                        2023.3
uc-micro-py                   1.0.1
unicodedata2                  15.0.0
urllib3                       1.26.15
uvicorn                       0.21.1
wcwidth                       0.2.6
webencodings                  0.5.1
websocket-client              1.5.1
websockets                    11.0.2
Werkzeug                      2.2.3
wheel                         0.40.0
widgetsnbextension            4.0.7
wrapt                         1.14.1
yarl                          1.9.1
zipp                          3.15.0
```

In [128]:
```
!pip install opencv-python matplotlib imageio gdown tensorflow-maco
```

```
Requirement already satisfied: opencv-python in /Users/htrap1211/t
ensorflow-test/env/lib/python3.8/site-packages (4.7.0.72)
Requirement already satisfied: matplotlib in /Users/htrap1211/tens
orflow-test/env/lib/python3.8/site-packages (3.7.1)
Requirement already satisfied: imageio in /Users/htrap1211/tensorf
low-test/env/lib/python3.8/site-packages (2.27.0)
Requirement already satisfied: gdown in /Users/htrap1211/tensorflo
w-test/env/lib/python3.8/site-packages (4.7.1)
Requirement already satisfied: tensorflow-macos in /Users/htrap121
1/tensorflow-test/env/lib/python3.8/site-packages (2.12.0)
Requirement already satisfied: numpy>=1.21.0 in /Users/htrap1211/t
ensorflow-test/env/lib/python3.8/site-packages (from opencv-python
) (1.23.2)
Requirement already satisfied: contourpy>=1.0.1 in /Users/htrap121
1/tensorflow-test/env/lib/python3.8/site-packages (from matplotlib
) (1.0.7)
Requirement already satisfied: cycler>=0.10 in /Users/htrap1211/te
nsorflow-test/env/lib/python3.8/site-packages (from matplotlib) (0
.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /Users/htrap12
```

In [129]:
```
import os
import cv2
import tensorflow as tf
import numpy as np
from typing import List
from matplotlib import pyplot as plt
import imageio
```

In [130]:
```
tf.config.list_physical_devices('GPU')
```

Out[130]:
```
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

In [131]:
```
physical_devices = tf.config.list_physical_devices('GPU')
try:
    tf.config.experimental.set_memory_growth(physical_devices[0], T
except:
    pass
```

1. Build Data Loading Functions

In [132]:
```
import gdown
```

In [133]:
```
url = 'https://drive.google.com/uc?id=1YlvpDLix3S-U8fd-gqRwPcWXAXm8
output = 'data.zip'
gdown.download(url, output, quiet=False)
gdown.extractall('data.zip')
```

```
 'data/alignments/s1/bgbu3s.align',
 'data/alignments/s1/bgbu4p.align',
 'data/alignments/s1/bgbu5a.align',
 'data/alignments/s1/bgia2n.align',
 'data/alignments/s1/bgia3s.align',
 'data/alignments/s1/bgia4p.align',
 'data/alignments/s1/bgia5a.align',
 'data/alignments/s1/bgig6n.align',
 'data/alignments/s1/bgig7s.align',
 'data/alignments/s1/bgig8p.align',
 'data/alignments/s1/bgig9a.align',
 'data/alignments/s1/bgin1s.align',
 'data/alignments/s1/bgin2p.align',
 'data/alignments/s1/bgin3a.align',
 'data/alignments/s1/bginzn.align',
 'data/alignments/s1/bgit4n.align',
 'data/alignments/s1/bgit5s.align',
 'data/alignments/s1/bgit6p.align',
 'data/alignments/s1/bgit7a.align',
 'data/alignments/s1/bgwb4n.align',
```

In [153]:
```
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
```

In [154]:
```
vocab = [x for x in "abcdefghijklmnopqrstuvwxyz'?!123456789 "]
```

In [155]:
```
char_to_num = tf.keras.layers.StringLookup(vocabulary=vocab, oov_to
num_to_char = tf.keras.layers.StringLookup(
    vocabulary=char_to_num.get_vocabulary(), oov_token="", invert=T
)

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)
```

In [156]:
```
char_to_num.get_vocabulary()
```

Out[156]:
```
['',
 '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',
 ' ']
```

In [157]:
```
char_to_num(['n','i','c','k'])
```

Out[157]:
```
<tf.Tensor: shape=(4,), dtype=int64, numpy=array([14,  9,  3, 11])
>
```

In [158]:
```
num_to_char([14,  9,  3, 11])
```

Out[158]:
```
<tf.Tensor: shape=(4,), dtype=string, numpy=array([b'n', b'i', b'c
', b'k'], dtype=object)>
```

In [159]:
```
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,
```

In [160]:
```
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_
    frames = load_video(video_path)
    alignments = load_alignments(alignment_path)

    return frames, alignments
```

In [161]:
```
test_path = '.\\data\\s1\\bbal6n.mpg'
```

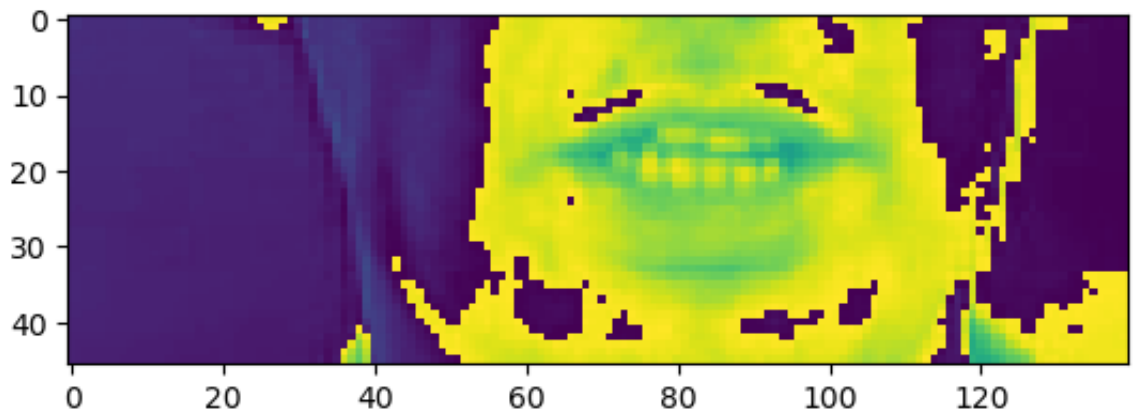
In [162]:
```
tf.convert_to_tensor(test_path).numpy().decode('utf-8').split('\\')
```

Out[162]: 'bbal6n'

In [163]:
```
ents = load_data(tf.convert_to_tensor('http://localhost:8888/edit/U
```

In [164]:
```
plt.imshow(frames[40])
```

Out[164]: <matplotlib.image.AxesImage at 0x2aaa420a0>



In [165]:
```
alignments
```

Out[165]:
```
<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])>
```

In [166]:
```
tf.strings.reduce_join([bytes.decode(x) for x in num_to_char(alignm
```

Out[166]:
```
<tf.Tensor: shape=(), dtype=string, numpy=b'bin blue at l six now'
>
```

In [167]:
```
def mappable_function(path:str) ->List[str]:
    result = tf.py_function(load_data, [path], (tf.float32, tf.int6
    return result
```

2. Create Data Pipeline

In [168]:
```
from matplotlib import pyplot as plt
```

In [169]:
```
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)
```

In [170]:
```
len(test)
```

Out[170]:
```
50
```

In [171]:
```
frames, alignments = data.as_numpy_iterator().next()
```

In [172]:
```
len(frames)
```

Out[172]:
```
2
```

In [173]:
```
sample = data.as_numpy_iterator()
```

In [174]:
```
val = sample.next(); val[0]
```

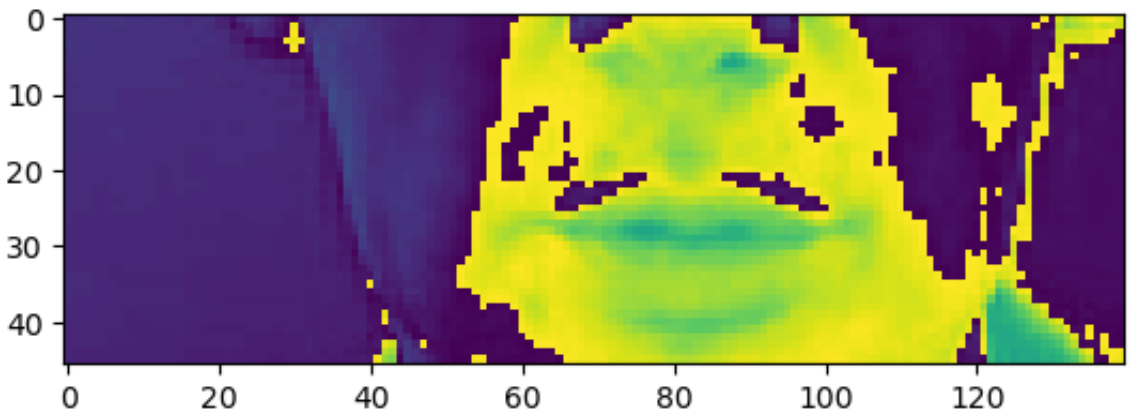
Out[174]:
```
array([[[[[1.3942066 ],
          [1.3942066 ],
          [1.357517  ],
          ...,
          [0.47696543],
          [0.5870344 ],
          [9.172412  ]],

         [[1.3942066 ],
          [1.3942066 ],
          [1.357517  ],
          ...,
          [0.2201379 ],
          [0.03668965],
          [8.622067  ]],

         [[1.3942066 ],
          [1.3942066 ],
          [1.3942066 ],
```

In [175]:
```
imageio.mimsave('./animation.gif', val[0][0], fps=10)
```

```
, 9.35586071014404 3]. Convert image to uint8 prior to saving to su
ppress this warning.
WARNING:imageio:Lossy conversion from float32 to uint8. Range [0.0
, 9.35586071014404 3]. Convert image to uint8 prior to saving to su
ppress this warning.
WARNING:imageio:Lossy conversion from float32 to uint8. Range [0.0
, 9.35586071014404 3]. Convert image to uint8 prior to saving to su
ppress this warning.
WARNING:imageio:Lossy conversion from float32 to uint8. Range [0.0
, 9.35586071014404 3]. Convert image to uint8 prior to saving to su
ppress this warning.
WARNING:imageio:Lossy conversion from float32 to uint8. Range [0.0
, 9.35586071014404 3]. Convert image to uint8 prior to saving to su
ppress this warning.
WARNING:imageio:Lossy conversion from float32 to uint8. Range [0.0
, 9.35586071014404 3]. Convert image to uint8 prior to saving to su
ppress this warning.
WARNING:imageio:Lossy conversion from float32 to uint8. Range [0.0
, 9.35586071014404 3]. Convert image to uint8 prior to saving to su
ppress this warning.
WARNING:imageio:Lossy conversion from float32 to uint8. Range [0.0
```

In [176]:
```
# 0:videos, 0: 1st video out of the batch,  0: return the first fra
plt.imshow(val[0][0][35])
```

Out[176]:
```
<matplotlib.image.AxesImage at 0x2aaa534f0>
```



In [177]:
```
tf.strings.reduce_join([num_to_char(word) for word in val[1][0]])
```

Out[177]:
```
<tf.Tensor: shape=(), dtype=string, numpy=b'bin white with u two p
lease'>
```

3. Design the Deep Neural Network

In [178]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv3D, LSTM, Dense, Dropout, B
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint, LearningRat
```

In [179]:
```
data.as_numpy_iterator().next()[0][0].shape
```

Out[179]:
```
(75, 46, 140, 1)
```

In [180]:
```
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',
model.add(Dropout(.5))

model.add(Bidirectional(LSTM(128, kernel_initializer='Orthogonal',
model.add(Dropout(.5))

model.add(Dense(char_to_num.vocabulary_size()+1, kernel_initializer
```

In [181]:
```
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv3d_9 (Conv3D)           (None, 75, 46, 140, 128)  3584

 activation_9 (Activation)   (None, 75, 46, 140, 128)  0

 max_pooling3d_9 (MaxPooling  (None, 75, 23, 70, 128)  0
 3D)

 conv3d_10 (Conv3D)          (None, 75, 23, 70, 256)   884992

 activation_10 (Activation)  (None, 75, 23, 70, 256)   0

 max_pooling3d_10 (MaxPoolin  (None, 75, 11, 35, 256)  0
 g3D)

 conv3d_11 (Conv3D)          (None, 75, 11, 35, 75)    518475

 activation_11 (Activation)  (None, 75, 11, 35, 75)    0

 max_pooling3d_11 (MaxPoolin  (None, 75, 5, 17, 75)    0
 g3D)

 time_distributed_3 (TimeDis  (None, 75, 6375)         0
 tributed)

 bidirectional_6 (Bidirectio  (None, 75, 256)          6660096
 nal)

 dropout_6 (Dropout)         (None, 75, 256)           0

 bidirectional_7 (Bidirectio  (None, 75, 256)          394240
 nal)

 dropout_7 (Dropout)         (None, 75, 256)           0

 dense_3 (Dense)             (None, 75, 41)            10537

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

In [182]:
```
5*17*75
```

Out[182]:
```
6375
```

In [183]:
```
yhat = model.predict(val[0])
```

```
1/1 [==============================] - 2s 2s/step
```

In [184]:
```
tf.strings.reduce_join([num_to_char(x) for x in tf.argmax(yhat[0],a
```

Out[184]:
```
<tf.Tensor: shape=(), dtype=string, numpy=b'rr5555ppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppppppppppz'>
```

In [185]:
```
tf.strings.reduce_join([num_to_char(tf.argmax(x)) for x in yhat[0]]
```

Out[185]:
```
<tf.Tensor: shape=(), dtype=string, numpy=b'rr5555ppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppppppppppz'>
```

In [186]:
```
model.input_shape
```

Out[186]:
```
(None, 75, 46, 140, 1)
```

In [187]:
```
model.output_shape
```

Out[187]:
```
(None, 75, 41)
```

4. Setup Training Options and Train

In [188]:
```
def scheduler(epoch, lr):
    if epoch < 30:
        return lr
    else:
        return lr * tf.math.exp(-0.1)
```

In [189]:
```
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), dty
    label_length = label_length * tf.ones(shape=(batch_len, 1), dty

    loss = tf.keras.backend.ctc_batch_cost(y_true, y_pred, input_le
    return loss
```

In [190]:
```
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
        for x in range(len(yhat)):
            print('Original:', tf.strings.reduce_join(num_to_char(d
            print('Prediction:', tf.strings.reduce_join(num_to_char
            print('~'*100)
```

In [191]:
```
model.compile(optimizer=Adam(learning_rate=0.0001), loss=CTCLoss)
```

```
WARNING:absl:At this time, the v2.11+ optimizer `tf.keras.optimize
rs.Adam` runs slowly on M1/M2 Macs, please use the legacy Keras op
timizer instead, located at `tf.keras.optimizers.legacy.Adam`.
WARNING:absl:There is a known slowdown when using v2.11+ Keras opt
imizers on M1/M2 Macs. Falling back to the legacy Keras optimizer,
i.e., `tf.keras.optimizers.legacy.Adam`.
```

In [192]:
```
checkpoint_callback = ModelCheckpoint(os.path.join('models','checkp
```

In [193]:
```
schedule_callback = LearningRateScheduler(scheduler)
```

In [194]:
```
example_callback = ProduceExample(test)
```

In [195]:
```
model.fit(train, validation_data=test, epochs=2, callbacks=[checkpo
```

```
Epoch 1/2
291/450 [==================>............] - ETA: 31:40 - loss: 76.8
788

[mpeg1video @ 0x3b050a150] ac-tex damaged at 22 17
[mpeg1video @ 0x3b050a150] Warning MVs not available

450/450 [==============================] - ETA: 0s - loss: 67.8628

[mpeg1video @ 0x3acac2880] ac-tex damaged at 22 17
[mpeg1video @ 0x3acac2880] Warning MVs not available

1/1 [==============================] - 2s 2s/step
Original: place white with k five soon
Prediction: e e
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Original: set white at u nine soon
Prediction: e e
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
450/450 [==============================] - 5762s 13s/step - loss:
67.8628 - val_loss: 93.3061 - lr: 1.0000e-04
Epoch 2/2
124/450 [=======>......................] - ETA: 1:00:49 - loss: 47
.5391

[mpeg1video @ 0x3a3b83170] ac-tex damaged at 22 17
[mpeg1video @ 0x3a3b83170] Warning MVs not available

450/450 [==============================] - ETA: 0s - loss: 44.7423

[mpeg1video @ 0x3aa83f260] ac-tex damaged at 22 17
[mpeg1video @ 0x3aa83f260] Warning MVs not available
[mpeg1video @ 0x2d3d527d0] ac-tex damaged at 22 17
[mpeg1video @ 0x2d3d527d0] Warning MVs not available

1/1 [==============================] - 1s 1s/step
Original: lay red with f zero now
Prediction: e e e
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Original: lay white with m zero now
Prediction: e e e
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
450/450 [==============================] - 5359s 12s/step - loss:
44.7423 - val_loss: 97.6913 - lr: 1.0000e-04
```

Out[195]:
```
<keras.callbacks.History at 0x2d31c8220>
```

5. Make a Prediction

In [196]:
```
url = 'https://drive.google.com/uc?id=1vWscXs4Vt0a_1IH1-ct2TCgXAZT-
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 (https://drive.google.com/uc?id=1vWscXs4Vt0a_1IH1-
ct2TCgXAZT-N3_Y)
From (redirected): https://drive.google.com/uc?id=1vWscXs4Vt0a_1IH
1-ct2TCgXAZT-N3_Y&confirm=t&uuid=e0c8a6d4-11f9-4464-85c4-90808da97
bb4 (https://drive.google.com/uc?id=1vWscXs4Vt0a_1IH1-ct2TCgXAZT-N
3_Y&confirm=t&uuid=e0c8a6d4-11f9-4464-85c4-90808da97bb4)
To: /Users/htrap1211/Untitled Folder 1/LipNet/checkpoints.zip
100%|████████████████████████████████████████| 94.5M/94.5M [00:14<00
:00, 6.36MB/s]
```

Out[196]:
```
['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']
```

In [197]:
```
model.load_weights('models/checkpoint')
```

Out[197]:
```
<tensorflow.python.checkpoint.checkpoint.CheckpointLoadStatus at 0
x2ceeb3c40>
```

In [198]:
```
test_data = test.as_numpy_iterator()
```

In [201]:
```
sample = test_data.next()
```

In [202]:
```
yhat = model.predict(sample[0])
```

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

In [203]:
```
print('~'*100, 'REAL TEXT')
[tf.strings.reduce_join([num_to_char(word) for word in sentence]) f
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ REAL TEXT
```

Out[203]:
```
[<tf.Tensor: shape=(), dtype=string, numpy=b'bin green at t nine s
oon'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'lay green at z nine a
gain'>]
```

In [204]:
```
decoded = tf.keras.backend.ctc_decode(yhat, input_length=[75,75], g
```

In [205]:
```
print('~'*100, 'PREDICTIONS')
[tf.strings.reduce_join([num_to_char(word) for word in sentence]) f
```

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

Out[205]:
```
[<tf.Tensor: shape=(), dtype=string, numpy=b'bin green at t nine s
oon'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'lay green at z nine a
gain'>]
```

Test on a Video

In [207]:
```
sample = load_data(tf.convert_to_tensor('http://localhost:8888/edit
```

In [208]:
```
print('~'*100, 'REAL TEXT')
[tf.strings.reduce_join([num_to_char(word) for word in sentence]) f
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ REAL TEXT
```

Out[208]:
```
[<tf.Tensor: shape=(), dtype=string, numpy=b'bin blue at l six now
'>]
```

In [209]:
```
yhat = model.predict(tf.expand_dims(sample[0], axis=0))
```

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

In [210]:
```
decoded = tf.keras.backend.ctc_decode(yhat, input_length=[75], gree
```

In [211]:
```
print('~'*100, 'PREDICTIONS')
[tf.strings.reduce_join([num_to_char(word) for word in sentence]) f
```

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

Out[211]:
```
[<tf.Tensor: shape=(), dtype=string, numpy=b'bin blue at l six now
'>]
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: