Install necessary dependencies

In [1]:

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

Build data loading functions

In [2]:

```
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 [3]:

```
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
)
```

In [4]:

```
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='
```

In [5]:

```
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
```

In [6]:

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

Design the neural network

In [7]:

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

In [8]:

```
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
model.add(Dropout(.5))

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

model.add(Dense(char_to_num.vocabulary_size()+1, kernel_initializer='he_normal', ac
```

In [9]:

```
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
_________________________________________________________________
```

Loading the trained model

In [10]:
```
new_model = tf.keras.models.load_model('LipReaderDNN')
```

```
WARNING:tensorflow:No training configuration found in save file, so the model was *n
ot* compiled. Compile it manually.
```

Test on video

In [11]:

```
filename = "bbaf3s.mpg"
filename = "pbiv1a.mpg"

sample = load_data(tf.convert_to_tensor(f'.\\data\\s1\\{filename}'))
```

In [12]:

```
# get original words
print('~'*100, 'REAL TEXT')
[tf.strings.reduce_join([num_to_char(word) for word in sentence]) for sentence in [
```

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

Out[12]:

```
[<tf.Tensor: shape=(), dtype=string, numpy=b'place blue in v one again'>]
```

In [13]:

```
# get predicted words
yhat = new_model.predict(tf.expand_dims(sample[0], axis=0))

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

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

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

Out[13]:

```
[<tf.Tensor: shape=(), dtype=string, numpy=b'place blue in v one again'>]
```

In [ ]: