PROJECT REPORT

AKHIL GUDAVALLI - A02231889

MOUNIKA ARUKALA - A02236773

SREE VIDYA SUSARLA - A02236104

SHIVAKESH REDDY ANNEPALLY - A02236102

SPEECH DIARITIZATION

PREPROCESSING

We made the following assumptions in the data:

1.If there is missing data, then we added the values of tmin and tmax in the speaker 1 and speaker 2 csv files and assigned the text column to a non-speech class which is '0'.

2.In the text column of the csv files, we encountered few literals like unicode, 1` and 12 which were assigned a value '1' that is a speech class.

Reading the audio file:

We collected the audio files and read all of them using a python module called wavio. This defines two functions called wavio.write() and wavio.read() of which the latter was used to read each file that returned sampling rate and data in numpy array format. The data is stored as a two-dimensional array in the shape (None,2) where 'None' represents the arbitrary number of rows and '2' represents the number of columns.

Now, the first column is the data of the Speaker 1 while the second one is the data of the Speaker 2.

```
for z in range(37):
    y = z
    f = "E:/Neural Nets/Project Data/Sound Files/HS_D"+a[y]+".wav"
    wav1 = wavio.read(f)
    print(wav1.rate)
    print(wav1.sampwidth)
```

DOWN SAMPLING:

The sampling rate turned out to be 44100. As the interval size is 20milli seconds, the down sampling is computed as follows:

Down sampling = (Sampling rate*20)/1000 which gives the down sampling value to be 882.

```
    s1 = wav1.data[:,0]/32767
    s2 = wav1.data[:,1]/32767
    rate = int(wav1.rate/50)
```

Here s1 represents Speaker 1 data and s2 represents Speaker 2 data.

PROJECT REPORT

AKHIL GUDAVALLI - A02231889

MOUNIKA ARUKALA - A02236773

SREE VIDYA SUSARLA - A02236104

SHIVAKESH REDDY ANNEPALLY - A02236102

SPEECH DIARITIZATION

We have the intervals as (0ms-20ms), (10ms-30ms),….(Kms, (K+20)ms) such that the overlapped interval size is 10milli seconds.

PREPARING THE TRAINING DATA:

The representation of every interval is (i,j) where the initial value of i=0 and j=882.The range (0,882) represents 20ms data of each speaker and for each iteration, i and j are incremented by the value shown below:

(down sampling value)/2= 882/2=441.

The process repeats until j<=number of rows for both the speakers in all the audio files.

```
p1 = []
i = 0
j = rate
while j<=s1.shape[0]:
    p1.append(s1[i:j])
    i = i+int(rate/2)
    j = j+int(rate/2)


p2 = []
i = 0
j = rate
while j<=s2.shape[0]:
    p2.append(s2[i:j])
    i = i+int(rate/2)
    j = j+int(rate/2)
```

The training data of the speaker 1 is in the numpy array p1 and that of speaker 2 is in the numpy array p2.

READING THE CSV FILES:

Later, csv files of each speaker were all read individually using the python package called pandas.

As the overlapped interval is 10 milli seconds and unit of the time columns (tmin and tmax) is seconds, we converted it to 10 milli seconds by multiplying it with 100.

AKHIL GUDAVALLI - A02231889

MOUNIKA ARUKALA - A02236773

SREE VIDYA SUSARLA - A02236104

SHIVAKESH REDDY ANNEPALLY - A02236102

SPEECH DIARITIZATION

We then defined an empty list t1= [ ].

Let us consider the speaker 1 data of the first audio file which has (tmin, tmax) values to be (0seconds ,6.083seconds) which is equivalent to (0,608.3 10milli seconds) and the values are rounded to the nearest integer.

Let's assume the non-speech label to be "0" for the range x to y(unit being 10 milliseconds), in which case the list would be appended with (y-x) zeros.

What follows is the explanation of the above for the first two entries (rows) in the csv file: The first entry in the file is in the range 0-608 with the non-speech label being 0 and so the list would be appended with 608 zeros calculated as t1=t1+608*[0]. For the second row the range is 608 to 1100 with the label "1" and so the list becomes 608*[0]+((1100-608)*[1]).

And the above procedure continues till the end of file. Similarly, we will repeat the above process for speaker 2 data as well.

```
    d1 = pd.read_csv("E:/Neural Nets/Project Data/CSV_Files_Final/HS_D"+a[y]+"_Spk1.csv")
    d1['tmi0'] = ((d1['tmi0']*100).round()).astype('int')
    d1['tmax'] = ((d1['tmax']*100).round()).astype('int')


    d2 = pd.read_csv("E:/Neural Nets/Project Data/CSV_Files_Final/HS_D"+a[y]+"_Spk2.csv")
    d2['tmi0'] = ((d2['tmi0']*100).round()).astype('int')
    d2['tmax'] = ((d2['tmax']*100).round()).astype('int')

    t1=[]
    t2=[]
    x = []
    for i in range(len(d1)):
        t1 = t1 + [d1.iloc[i][2]]*(d1.iloc[i][1] - d1.iloc[i][0])
        x.append([d1.iloc[i][1],len(t1)])
    for i in range(len(d2)):
        t2 = t2 + [d2.iloc[i][2]]*(d2.iloc[i][1] - d2.iloc[i][0])



    for i in range(len(t1)-1):
        t1[i] = min(t1[i],t1[i+1])
    for i in range(len(t2)-1):
        t2[i] = min(t2[i],t2[i+1])

    t1 = t1[0:len(p1)]
    t2 = t2[0:len(p2)]

    p1_final = p1_final + p1
    p2_final = p2_final + p2
    t1_final = t1_final + t1
    t2_final = t2_final + t2

print(len(p1_final))
print(len(p2_final))
print(len(t1_final))
print(len(t2_final))

p_final = p1_final+p2_final
t_final = t1_final+t2_final

for i in range(len(t_final)):
    if t_final[i] == '1`' or t_final[i]==12:
        t_final[i] = 1

print("start")
p_final = np.asarray(p_final).astype(np.float32)
t_final = np.asarray(t_final).astype(np.float32)
print("end")
```

PROJECT REPORT

AKHIL GUDAVALLI - A02231889

MOUNIKA ARUKALA - A02236773

SREE VIDYA SUSARLA - A02236104

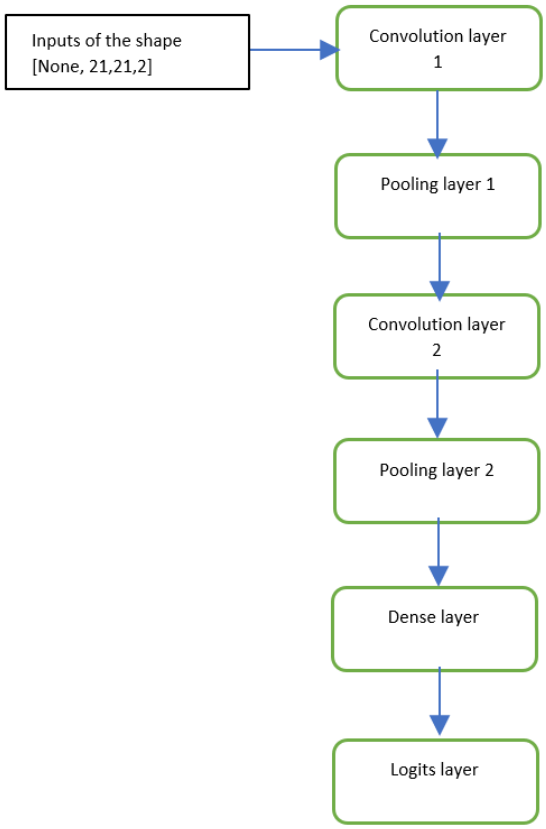
SHIVAKESH REDDY ANNEPALLY - A02236102

SPEECH DIARITIZATION

Now we have a list of speech and non-speech labels over a 10ms interval as [0ms-10ms], [10ms-20ms] and ... Now we will need to convert this list into 20ms intervals with 10ms overlap as [0ms- 20ms], [10ms -30ms],....To evaluate the interval to 20ms with 10ms overlap, we have taken the minimum value between the 0ms to 10ms and 10ms to 20ms. Then we build a convolution neural network which has a learning rate of 0.001 with Adam optimizer and batch_size=100.

CNN ARCHITECTURE

A Convolution neural network is composed of a stack of convolutional modules that perform feature extraction. Each module consists of a convolutional layer followed by a pooling layer. The last convolutional module is followed by one or more dense layers that performs the classification. The final dense layer in a CNN contains a single node for each target class in the model with a softmax activation function that generates a value between 0 – 1, for which, we can interpret the softmax values for a given data. We can then predict in which target class (Speech or Non-speech) will our data falls into.



PROJECT REPORT

AKHIL GUDAVALLI - A02231889

MOUNIKA ARUKALA - A02236773

SREE VIDYA SUSARLA - A02236104

SHIVAKESH REDDY ANNEPALLY - A02236102

SPEECH DIARITIZATION

Number of layers – six layers of which there are two convolution layers, two max pooling layers, one dense layer and an output layer.

We used softmax activation function in our output layer which is a generalization of the logistic function that "squashes" a K-dimensional vector {z} of arbitrary real values to a K-dimensional vector of real values in the range [0, 1] that add up to 1.

Convolutional Layer #1: In this layer, we applied 32 5x5 filters by extracting 5x5-pixel sub-regions with ReLU activation function. The kernel size is [5,5].

Pooling Layer #1: In this layer, we perform the max pooling with a 2x2 filter and stride of 2 which specifies that the pooled regions do not overlap.

Convolutional Layer #2: In this layer, we apply 64 5x5 filters with ReLU activation function. The kernel size is [5,5].

Pooling Layer #2: In this layer, again we perform the max pooling with a 2x2 filter and stride of 2.

Dense Layer #1: 1024 neurons with dropout regularization rate of 0.4 (probability of 0.4 that any given element will be dropped during training)

Logits Layer: In this layer we have 2 neurons, one for each target class 0 for non-speech and 1 for speech.

We have used the following functions in our model to create each type of layer:

conv2d(): This functions constructs a two-dimensional convolutional layer by taking number of filters, filter kernel size, padding, and activation function as arguments.

max_pooling2d() – This functions constructs a two-dimensional pooling layer using the max-pooling algorithm by taking pooling filter size and stride as arguments.

dense() – This function constructs a dense layer by taking the number of neurons and activation function as arguments.

tf.nn_sparse_softmax_cross_entropy_with_logits():

We have used tf.nn_sparse_softmax_cross_entropy_with_logits() as our cost function which measures the probability error in discrete classification tasks in which the classes are mutually exclusive.

PROJECT REPORT

AKHIL GUDAVALLI - A02231889

MOUNIKA ARUKALA - A02236773

SREE VIDYA SUSARLA - A02236104

SHIVAKESH REDDY ANNEPALLY - A02236102

SPEECH DIARITIZATION

Our Neural Network functions as follows:

Step 1:

By using the conv2D we will create a convolution layer 1 to which we will pass our input of the shape [None, 21,21,2]. Then we will apply 32 5*5 convolution filters to the data. For each sub-region, the layer performs a set of mathematical operations to produce a single value in the output feature map which then applies ReLU activation function to the output.

Step 2:

The generated output is then passed as an input to the max pooling layer 1 in which we down-sample our data that is extracted by the convolutional layers to reduce the dimensionality of the feature map in order to decrease processing time. We used max pooling algorithm that extracts sub-regions of the feature map(2x2-pixel) to keep the maximum value and discards all other values.

Step 3:

The output that is simulated by the pooling layer 1 is then passed as an input to the convolution layer 2 which again applies 32 5*5 convolution filters to the data for which, for each sub-region, the layer performs a set of mathematical operations to produce a single value in the output feature map which then applies ReLU activation function to the output.

Step 4:

This output is again passed as an input to the pooling layer 2 in which we again down-sample our data that is extracted by the convolutional layers to reduce the dimensionality of the feature map in order to decrease processing time. We used max pooling algorithm that extracts sub-regions of the feature map(2x2-pixel) to keep the maximum value and discards all other values.

Step 5:

This output is then passed as an input to the dense layer which performs classification on the features extracted by the convolutional layers and down-sampled by the pooling layers. Every node in the layer is connected to every node in the preceding layer. In this layer, there is a regularization rate of 0.4 that will drop any given element during the training.

PROJECT REPORT

AKHIL GUDAVALLI - A02231889

MOUNIKA ARUKALA - A02236773

SREE VIDYA SUSARLA - A02236104

SHIVAKESH REDDY ANNEPALLY - A02236102

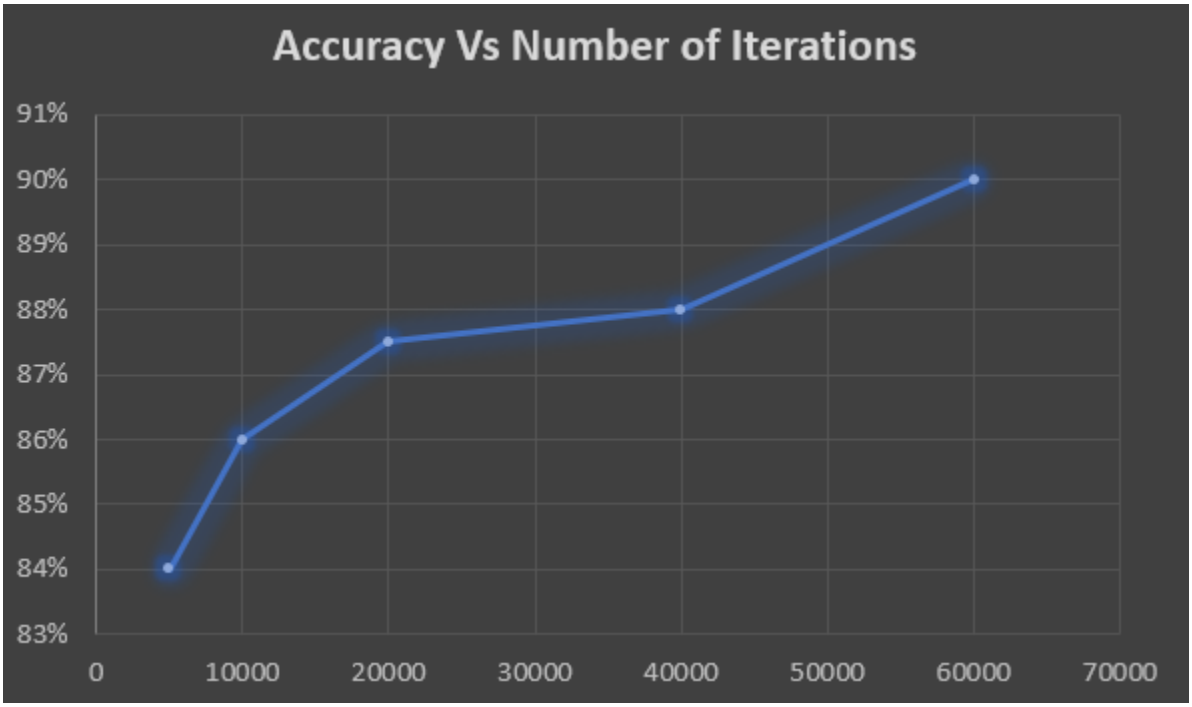
SPEECH DIARITIZATION

Step 6:

The logits layer simulates the final output which categorizes two classes 0 for non-speech and 1 for speech.

LEARNING CURVE, ACCURACY AND LOSS

Initially, we passed our data using 5000 iterations for which the accuracy was found to be around 84%. Then we increased the number of iterations to 60,000 for which the accuracy was found to be around 90% that evaluates the quality of fit relative to the data.



From our accuracy vs number of iterations scatter plot graph, we can clearly see that the more number of iterations, the better will be the accuracy.

LEARNING RATE:

Learning rate is a parameter that is chosen by the programmer. A high learning rate means that bigger steps are taken in the weight updates and thus it may take less time for model to converge to an optimal set of weights.

PROJECT REPORT

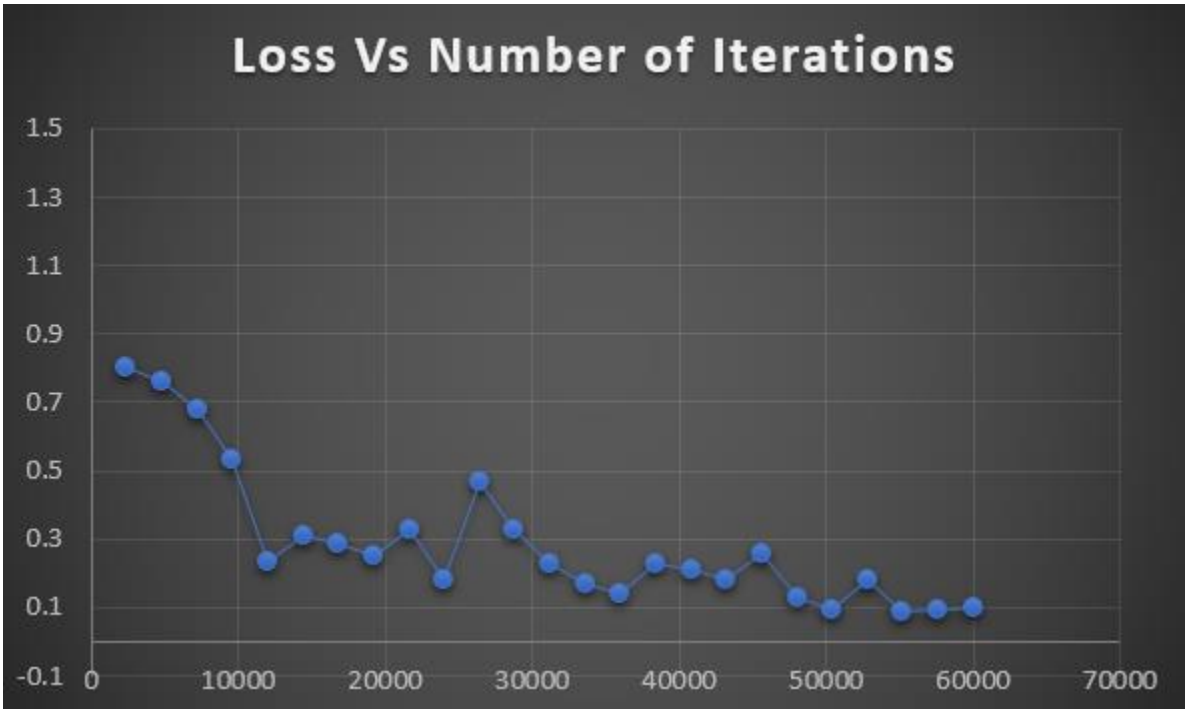
AKHIL GUDAVALLI - A02231889

MOUNIKA ARUKALA - A02236773

SREE VIDYA SUSARLA - A02236104

SHIVAKESH REDDY ANNEPALLY - A02236102

SPEECH DIARITIZATION



From our Loss vs number of iterations scatter plot graph, we can clearly see that the loss function and the number of iterations are inversely proportional to each other. In other words, as the number of iterations increases, there is a decrease in the loss values.

RESULTS

Now we took the eighteenth audio file data to test our neural network where we did the preprocessing as discussed above in the document and applied the trained model on the data. We got the speech and the non-speech classification for all the 20milli second intervals data for both the speaker 1 and speaker 2. Then we measured the evaluation metrics and found the accuracy to be 90% for the speaker 2 and 88.6% for the speaker 1 data which shows that our neural network has a high performance.

We then converted the classification list that we got into a csv file which is implemented as follows:

PROJECT REPORT

AKHIL GUDAVALLI - A02231889

MOUNIKA ARUKALA - A02236773

SREE VIDYA SUSARLA - A02236104

SHIVAKESH REDDY ANNEPALLY - A02236102

SPEECH DIARITIZATION

```
a1= list(model.predict(input_fn))
a2= list(model.predict(input_fn))

p = a1[0]
count = 0
f1 = []
for i in range(len(a1)):
    if a1[i] == p:
        count = count+1
    else:
        f1.append([a1[i-1],count])
        count = 1
        p = a1[i]

f1.append([a1[-1],count])
p = a2[0]
count = 0
f2 = []
for i in range(len(a2)):
    if a2[i] == p:
        count = count+1
    else:
        f2.append([a2[i-1],count])
        count = 1
        p = a2[i]

f2.append([a2[-1],count])

c1 = []
g = 0
for i in range(len(f1)):
    c1.append([g,g+f1[i][1]/10.0,f1[i][0],1])
    g = g+f1[i][1]/10

c2 = []
g =0
for i in range(len(f2)):
    c2.append([g,g+f2[i][1]/10.0,f2[i][0],1])
    g = g+f2[i][1]/10

df1 = pd.DataFrame(c1)
df2 = pd.DataFrame(c2)

df1.to_csv('E:/Neural Nets/speaker1_out.csv')
df2.to_csv('E:/Neural Nets/speaker2_out.csv')
```

We will evaluate the accuracy by comparing the data in the generated csv file with the original csv file that was initially given to which we applied the preprocessing steps again and applied the neural network model.

PROJECT REPORT

AKHIL GUDAVALLI - A02231889

MOUNIKA ARUKALA - A02236773

SREE VIDYA SUSARLA - A02236104

SHIVAKESH REDDY ANNEPALLY - A02236102

SPEECH DIARITIZATION

OUTPUT SCREESHOTS:

```
INFO:tensorflow:Using default config.
WARNING:tensorflow:Using temporary folder as model directory: C:\Users\Akhil\AppData
\Local\Temp\tmpj09fduuw
INFO:tensorflow:Using config: {'_model_dir': 'C:\\Users\\Akhil\\AppData\\Local\\Temp\
\tmpj09fduuw', '_tf_random_seed': None, '_save_summary_steps': 100,
'_save_checkpoints_steps': None, '_save_checkpoints_secs': 600, '_session_config':
None, '_keep_checkpoint_max': 5, '_keep_checkpoint_every_n_hours': 10000,
'_log_step_count_steps': 100, '_service': None, '_cluster_spec':
<tensorflow.python.training.server_lib.ClusterSpec object at 0x000002990D9D4390>,
'_task_type': 'worker', '_task_id': 0, '_master': '', '_is_chief': True,
'_num_ps_replicas': 0, '_num_worker_replicas': 1}
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Saving checkpoints for 1 into C:\Users\Akhil\AppData\Local\Temp
\tmpj09fduuw\model.ckpt.
INFO:tensorflow:loss = 0.667339, step = 1
INFO:tensorflow:global_step/sec: 17.3864
INFO:tensorflow:loss = 0.315289, step = 101 (5.736 sec)
INFO:tensorflow:global_step/sec: 17.7661
INFO:tensorflow:loss = 0.255685, step = 201 (5.644 sec)
INFO:tensorflow:global_step/sec: 17.6811
INFO:tensorflow:loss = 0.296074, step = 301 (5.640 sec)
INFO:tensorflow:global_step/sec: 17.6982
INFO:tensorflow:loss = 0.254315, step = 401 (5.650 sec)
INFO:tensorflow:global_step/sec: 17.6602
```

```
INFO:tensorflow:Restoring parameters from C:\Users\Akhil\AppData\Local\Temp\tmpj09fduuw
\model.ckpt-600
INFO:tensorflow:Restoring parameters from C:\Users\Akhil\AppData\Local\Temp\tmpj09fduuw
\model.ckpt-600
Testing Accuracy: 88.6
Testing Accuracy: 90
```

WHAT WE LEARNT?

We learnt the following things:

1. To detect to which target class the data belongs to by measuring the evaluation metrics with an accuracy of 90% that results in a high performance neural network.

2. To normalize the values to get accurate data that doesn't go out of range and for easy computations.

3. To decrease the learning rate during training.

4. To use an optimizer called Adam optimizer that minimizes the cost function that is more stable.

PROJECT REPORT

AKHIL GUDAVALLI - A02231889

MOUNIKA ARUKALA - A02236773

SREE VIDYA SUSARLA - A02236104

SHIVAKESH REDDY ANNEPALLY - A02236102

SPEECH DIARITIZATION

NEXT STEPS:

The accuracy can be improved by better preprocessing and by making better assumptions for the data and also by adding momentum to the neural network.

CODE:

```
import wavio

import numpy as np

import pandas as pd

a=["%02d" % (x+1) for x in range(37)]

p1_final = []

p2_final = []

t1_final = []

t2_final = []

for z in range(37):

    y = z

    f = "E:/Neural Nets/Project Data/Sound Files/HS_D"+a[y]+".wav"

    wav1 = wavio.read(f)

    print(wav1.rate)

    print(wav1.sampwidth)

    s1 = wav1.data[:,0]/32767

    s2 = wav1.data[:,1]/32767

    rate = int(wav1.rate/50)

    p1 = []

    i = 0

    j = rate
```

AKHIL GUDAVALLI - A02231889
MOUNIKA ARUKALA - A02236773
SREE VIDYA SUSARLA - A02236104
SHIVAKESH REDDY ANNEPALLY - A02236102

SPEECH DIARITIZATION

```
 while j<=s1.shape[0]:
    p1.append(s1[i:j])
    i = i+int(rate/2)
    j = j+int(rate/2)
p2 = []
 i = 0
 j = rate
 while j<=s2.shape[0]:
    p2.append(s2[i:j])
    i = i+int(rate/2)
    j = j+int(rate/2)
 d1 = pd.read_csv("E:/Neural Nets/Project Data/CSV_Files_Final/HS_D"+a[y]+"_Spk1.csv")
 d1['tmi0'] = ((d1['tmi0']*100).round()).astype('int')
 d1['tmax'] = ((d1['tmax']*100).round()).astype('int')
 d2 = pd.read_csv("E:/Neural Nets/Project Data/CSV_Files_Final/HS_D"+a[y]+"_Spk2.csv")
 d2['tmi0'] = ((d2['tmi0']*100).round()).astype('int')
 d2['tmax'] = ((d2['tmax']*100).round()).astype('int')
 t1=[]
 t2=[]
 x = []
 for i in range(len(d1)):
    t1 = t1 + [d1.iloc[i][2]]*(d1.iloc[i][1] - d1.iloc[i][0])
    x.append([d1.iloc[i][1],len(t1)])
 for i in range(len(d2)):
```

AKHIL GUDAVALLI - A02231889
MOUNIKA ARUKALA - A02236773
SREE VIDYA SUSARLA - A02236104
SHIVAKESH REDDY ANNEPALLY - A02236102

SPEECH DIARITIZATION

```
        t2 = t2 + [d2.iloc[i][2]]*(d2.iloc[i][1] - d2.iloc[i][0])
    for i in range(len(t1)-1):
        t1[i] = min(t1[i],t1[i+1])
    for i in range(len(t2)-1):
        t2[i] = min(t2[i],t2[i+1])
    t1 = t1[0:len(p1)]
    t2 = t2[0:len(p2)]
    p1_final = p1_final + p1
    p2_final = p2_final + p2
    t1_final = t1_final + t1
    t2_final = t2_final + t2
print(len(p1_final))
print(len(p2_final))
print(len(t1_final))
print(len(t2_final))
p_final = p1_final+p2_final
t_final = t1_final+t2_final
for i in range(len(t_final)):
    if t_final[i] == '1`' or t_final[i]==12:
        t_final[i] = 1
print("start")
p_final = np.asarray(p_final).astype(np.float32)
t_final = np.asarray(t_final).astype(np.float32)
print("end")
```

PROJECT REPORT

AKHIL GUDAVALLI - A02231889

MOUNIKA ARUKALA - A02236773

SREE VIDYA SUSARLA - A02236104

SHIVAKESH REDDY ANNEPALLY - A02236102

SPEECH DIARITIZATION

```
loss_g = []
import tensorflow as tf
learning_rate = 0.001
num_steps = 60000
batch_size = 100
def cnn_model_fn(features,reuse=False,training=True):
  with tf.variable_scope('neural_net_model', reuse=reuse):
    input_layer = tf.reshape(features['images'], shape=[-1, 21, 21, 2])
    conv1 = tf.layers.conv2d(input_layer, 32, 5, activation=tf.nn.relu)
    pool1 = tf.layers.max_pooling2d(conv1, 2, 2)
    conv2 = tf.layers.conv2d(pool1, 64, 3, activation=tf.nn.relu)
    pool2 = tf.layers.max_pooling2d(conv2, 2, 2)
    pool2_flat = tf.contrib.layers.flatten(pool2)
    dense = tf.layers.dense(pool2_flat, 1024)
    dropout = tf.layers.dropout(dense, rate=0.4, training=training)
    logits = tf.layers.dense(dropout, 2)
  return logits
def model_fn(features, labels, mode):
  logits_train = cnn_model_fn(features)
  logits_test = cnn_model_fn(features, reuse=True,training=False)
  predictions = tf.argmax(logits_test, axis=1)
  if mode == tf.estimator.ModeKeys.PREDICT:
    return tf.estimator.EstimatorSpec(mode, predictions=predictions)
```

PROJECT REPORT

AKHIL GUDAVALLI - A02231889

MOUNIKA ARUKALA - A02236773

SREE VIDYA SUSARLA - A02236104

SHIVAKESH REDDY ANNEPALLY - A02236102

SPEECH DIARITIZATION

```
    loss = tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(logits=logits_train, labels=tf.cast(labels, dtype=tf.int32)))
    optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate)
    train_op = optimizer.minimize(loss,global_step=tf.train.get_global_step())
    acc_op = tf.metrics.accuracy(labels=labels, predictions=predictions)
    loss_g.append(loss)
    estim_specs = tf.estimator.EstimatorSpec(
      mode=mode,
      predictions=predictions,
      loss=loss,
      train_op=train_op,
      eval_metric_ops={'accuracy': acc_op})
    return estim_specs
p1_final = []
p2_final = []
t1_final = []
t2_final = []
for z in range(1):
    y = 17
    f = "E:/Neural Nets/Project Data/Sound Files/HS_D"+a[y]+".wav"
    wav1 = wavio.read(f)
    print(wav1.rate)
    print(wav1.sampwidth)
    s1 = wav1.data[:,0]/32767
```

AKHIL GUDAVALLI - A02231889
MOUNIKA ARUKALA - A02236773
SREE VIDYA SUSARLA - A02236104
SHIVAKESH REDDY ANNEPALLY - A02236102

SPEECH DIARITIZATION

```
s2 = wav1.data[:,1]/32767
rate = int(wav1.rate/50)
p1 = []
i = 0
j = rate
while j<=s1.shape[0]:
    p1.append(s1[i:j])
    i = i+int(rate/2)
    j = j+int(rate/2)
p2 = []
i = 0
j = rate
while j<=s2.shape[0]:
    p2.append(s2[i:j])
    i = i+int(rate/2)
    j = j+int(rate/2)
d1 = pd.read_csv("E:/Neural Nets/Project Data/CSV_Files_Final/HS_D"+a[y]+"_Spk1.csv")
d1['tmi0'] = ((d1['tmi0']*100).round()).astype('int')
d1['tmax'] = ((d1['tmax']*100).round()).astype('int')
d2 = pd.read_csv("E:/Neural Nets/Project Data/CSV_Files_Final/HS_D"+a[y]+"_Spk2.csv")
d2['tmi0'] = ((d2['tmi0']*100).round()).astype('int')
d2['tmax'] = ((d2['tmax']*100).round()).astype('int')
t1=[]
t2=[]
```

PROJECT REPORT

AKHIL GUDAVALLI - A02231889

MOUNIKA ARUKALA - A02236773

SREE VIDYA SUSARLA - A02236104

SHIVAKESH REDDY ANNEPALLY - A02236102

SPEECH DIARITIZATION

```
    x = []
    for i in range(len(d1)):
        t1 = t1 + [d1.iloc[i][2]]*(d1.iloc[i][1] - d1.iloc[i][0])
        x.append([d1.iloc[i][1],len(t1)])
    for i in range(len(d2)):
        t2 = t2 + [d2.iloc[i][2]]*(d2.iloc[i][1] - d2.iloc[i][0])
    for i in range(len(t1)-1):
        t1[i] = min(t1[i],t1[i+1])
    for i in range(len(t2)-1):
        t2[i] = min(t2[i],t2[i+1])
    t1 = t1[0:len(p1)]
    t2 = t2[0:len(p2)]
    p1_final = p1_final + p1
    p2_final = p2_final + p2
    t1_final = t1_final + t1
    t2_final = t2_final + t2
p1_final = np.asarray(p1).astype(np.float32)
t1_final = np.asarray(t1).astype(np.float32)
p2_final = np.asarray(p2).astype(np.float32)
t2_final = np.asarray(t2).astype(np.float32)
model = tf.estimator.Estimator(model_fn)
input_fn = tf.estimator.inputs.numpy_input_fn(
    x={'images': p_final}, y=t_final,
    batch_size=batch_size, num_epochs=None, shuffle=True)
```

PROJECT REPORT

AKHIL GUDAVALLI - A02231889

MOUNIKA ARUKALA - A02236773

SREE VIDYA SUSARLA - A02236104

SHIVAKESH REDDY ANNEPALLY - A02236102

SPEECH DIARITIZATION

```
model.train(input_fn, steps=num_steps)
input_fn = tf.estimator.inputs.numpy_input_fn(
   x={'images': p1_final}, y=t1_final,batch_size=batch_size, shuffle=False)
e = model.evaluate(input_fn)
print("Testing Accuracy:", e['accuracy'])
input_fn = tf.estimator.inputs.numpy_input_fn(
   x={'images': p2_final}, y=t2_final,batch_size=batch_size, shuffle=False)
e = model.evaluate(input_fn)
print("Testing Accuracy:", e['accuracy'])
a1= list(model.predict(input_fn))
a2= list(model.predict(input_fn))
p = a1[0]
count = 0
f1 = []
for i in range(len(a1)):
   if a1[i] == p:
      count = count+1
   else:
      f1.append([a1[i-1],count])
      count = 1
      p = a1[i]
f1.append([a1[-1],count])
p = a2[0]
count = 0
```

PROJECT REPORT

AKHIL GUDAVALLI - A02231889

MOUNIKA ARUKALA - A02236773

SREE VIDYA SUSARLA - A02236104

SHIVAKESH REDDY ANNEPALLY - A02236102

SPEECH DIARITIZATION

```
f2 = []
for i in range(len(a2)):
   if a2[i] == p:
      count = count+1
   else:
      f2.append([a2[i-1],count])
      count = 1
      p = a2[i]
f2.append([a2[-1],count])
c1 = []
g = 0
for i in range(len(f1)):
   c1.append([g,g+f1[i][1]/10.0,f1[i][0],1])
   g = g+f1[i][1]/10
c2 = []
g =0
for i in range(len(f2)):
   c2.append([g,g+f2[i][1]/10.0,f2[i][0],1])
   g = g+f2[i][1]/10
df1 = pd.DataFrame(c1)
df2 = pd.DataFrame(c2)
df1.to_csv('E:/Neural Nets/speaker1_out.csv')
df2.to_csv('E:/Neural Nets/speaker2_out.csv')
```