APPENDIX

DATA LABELING TRAIN AND TEST

```
try:

    import tensorflow as tf
    import cv2
    import os
    import pickle
    import numpy as np
    print("image folder Loaded Successfully")
except:
    print("no_library")


class MasterImage(object):

    def __init__(self,PATH='', IMAGE_SIZE = 50):
        self.PATH = PATH
        self.IMAGE_SIZE = IMAGE_SIZE

        self.image_data = []
        self.x_data = []
        self.y_data = []
        self.CATEGORIES = []


        self.list_categories = []

    def get_categories(self):
        for path in os.listdir(self.PATH):
            if '.DS_Store' in path:
                pass
            else:
                self.list_categories.append(path)
        print("Category_train_test ",self.list_categories,'\n')
        return self.list_categories

    def Process_Image(self):
        try:

            self.CATEGORIES = self.get_categories()
            for categories in self.CATEGORIES:

                    train_folder_path = os.path.join(self.PATH,
categories)
                class_index = self.CATEGORIES.index(categories)

                for img in os.listdir(train_folder_path):
                        new_path = os.path.join(train_folder_path,
img)
```

```
                    try:
                                                        image_data_temp  =
cv2.imread(new_path,cv2.IMREAD_GRAYSCALE)
                                                    image_temp_resize  =
cv2.resize(image_data_temp,(self.IMAGE_SIZE,self.IMAGE_SIZE))
                        self.image_data.append([image_temp_resize,
class_index])
                    except:
                        pass

            data = np.asanyarray(self.image_data)

            for x in data:
                self.x_data.append(x[0])
                self.y_data.append(x[1])

            X_Data = np.asarray(self.x_data) / (255.0)
            Y_Data = np.asarray(self.y_data)

                    X_Data  =  X_Data.reshape(-1,  self.IMAGE_SIZE,
self.IMAGE_SIZE, 1)

            return X_Data, Y_Data
        except:
            print("fail")

    def pickle_image(self):

        X_Data,Y_Data = self.Process_Image()
        pickle_out = open('X_Train','wb')
        pickle.dump(X_Data, pickle_out)
        pickle_out.close()

        pickle_out = open('Y_Test', 'wb')
        pickle.dump(Y_Data, pickle_out)
        pickle_out.close()

        print("Pickled Image Successfully ")
        return X_Data,Y_Data

    def load_dataset(self):

        try:
            # Read the Data from Pickle Object
            X_Temp = open('X_Data','rb')
            X_Data = pickle.load(X_Temp)

            Y_Temp = open('Y_Data','rb')
            Y_Data = pickle.load(Y_Temp)

            print('Save_as_pickle')

            return X_Data,Y_Data
        except:
```

```
            print('pickle_not_found')
            print('loading_data_train_test')

            X_Data,Y_Data = self.pickle_image()
            return X_Data,Y_Data

if __name__ == "__main__":
    path = '/home/william/Music/img'
    a = MasterImage(PATH=path,
                    IMAGE_SIZE=80)

    X_Data,Y_Data = a.load_dataset()
    print(X_Data.shape)
```

FEATURE IMAGE EXCTRACION AND ALGORITHM

```
import tensorflow as tf
from tensorflow.python.ops import rnn, rnn_cell
import pickle
import time



pickle_in = open("X_Train","rb")
X = pickle.load(pickle_in)

pickle_in = open("Y_Test","rb")
y = pickle.load(pickle_in)


X = X/255.0

hm_epochs = 10
n_classes = 36
batch_size = 128
chunk_size = 28
n_chunks = 28
rnn_size = 128

keep_rate = 0.8
rate = tf.compat.v1.placeholder(tf.float32)



def conv2d(x, W):
    return tf.nn.conv2d(x, W, strides=[1,1,1,1], padding='SAME')

def maxpool2d(x):

                return    tf.nn.max_pool2d    (x,    ksize=[1,2,2,1],
strides=[1,2,2,1], padding='SAME')



x = tf.compat.v1.placeholder('float', [None, n_chunks,chunk_size])
```

```
y = tf.compat.v1.placeholder('float')


def recurrent_neural_network(x):
                                                weights            =
{'W_conv1':tf.Variable(tf.random_normal([5,5,1,32])),
               'W_conv2':tf.Variable(tf.random_normal([5,5,32,64])
),
               'W_fc':tf.Variable(tf.random_normal([7*7*64,512])),
                         'out':tf.Variable(tf.random_normal([1024,
n_classes]))}

    biases = {'b_conv1':tf.Variable(tf.random_normal([32])),
               'b_conv2':tf.Variable(tf.random_normal([64])),
               'b_fc':tf.Variable(tf.random_normal([512])),
               'out':tf.Variable(tf.random_normal([n_classes]))}

    x = tf.reshape(x, shape=[1, 28, 28, 1])

          conv1   =  tf.nn.relu(conv2d(x,  weights['W_conv1'])  +
biases['b_conv1'])
    conv1 = maxpool2d(conv1)

        conv2  =  tf.nn.relu(conv2d(conv1,  weights['W_conv2'])  +
biases['b_conv2'])
    conv2 = maxpool2d(conv2)

    fc = tf.reshape(conv2,[-1, 7*7*64])
    fc = tf.nn.relu(tf.matmul(fc, weights['W_fc'])+biases['b_fc'])
    fc = tf.nn.dropout(fc, keep_rate)

    output = tf.matmul(fc, weights['out'])+biases['out']

                                                   layer              =
{'weights':tf.Variable(tf.random_normal([rnn_size,n_classes])),
             'biases':tf.Variable(tf.random_normal([n_classes]))}

    x = tf.transpose(x, [1,0,2])
    x = tf.reshape(x, [-1, chunk_size])
    x = tf.split(x, n_chunks, 0)

                                              lstm_cell            =
rnn_cell.BasicLSTMCell(rnn_size,state_is_tuple=True)
            outputs,   states   =   rnn.static_rnn(lstm_cell,   x,
dtype=tf.float32)

           output   =   tf.matmul(outputs[-1],layer['weights'])   +
layer['biases']

    return output



def train_neural_network(x):
```

```
    prediction = recurrent_neural_network(x)
                                                    cost            =
tf.reduce_mean( tf.nn.softmax_cross_entropy_with_logits(prediction
,y) )
    optimizer = tf.train.AdamOptimizer().minimize(cost)


    with tf.Session() as sess:
        sess.run(tf.initialize_all_variables())

        for epoch in range(hm_epochs):
            epoch_loss = 0
                                                      for     _     in
range(int(data.train.num_examples/batch_size)):
                                                 epoch_x,   epoch_y   =
data.train.next_batch(batch_size)
                                                        epoch_x    =
epoch_x.reshape((batch_size,n_chunks,chunk_size))

                _, c = sess.run([optimizer, cost], feed_dict={x:
epoch_x, y: epoch_y})
                epoch_loss += c

                            print('Epoch',  epoch,  'completed  out
of',hm_epochs,'loss:',epoch_loss)

         correct = tf.equal(tf.argmax(prediction, 1), tf.argmax(y,
1))

        accuracy = tf.reduce_mean(tf.cast(correct, 'float'))
        print('Accuracy:',accuracy.eval({x:data.test.images.reshap
e((-1, n_chunks, chunk_size)), y:data.test.labels}))

train_neural_network(x)
```

LOAD MODEL AND MAKE PREDICTION

```
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from    tflearn.layers.core    import    input_data,    dropout,
fully_connected
from tflearn.layers.estimator import regression
import cv2
import numpy as np
import pandas as pd

pickle_in = open("X_Train","rb")
X = pickle.load(pickle_in)

pickle_in = open("Y_Test","rb")
y = pickle.load(pickle_in)
```

```
X = X.reshape([-1, 28, 28, 1])
test_x = test_x.reshape([-1, 28, 28, 1])


convnet = input_data(shape=[None, 28, 28, 1], name='input')

convnet = conv_2d(convnet, 32, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)

convnet = conv_2d(convnet, 64, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)

convnet = fully_connected(convnet, 1024, activation='relu')
convnet = dropout(convnet, 0.8)

convnet = fully_connected(convnet, 10, activation='softmax')
convnet       =       regression(convnet,       optimizer='adam',
learning_rate=0.01,              loss='categorical_crossentropy',
name='targets')

model = tflearn.DNN(convnet)
model.load('lstm_ocr.model')

def predict(input_img):
    img = cv2.imread("sample1.jpg")
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    ret, thresh1 = cv2.threshold(gray, 0, 255, cv2.THRESH_OTSU |
cv2.THRESH_BINARY_INV)
    rect_kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (18,
18))
    dilation = cv2.dilate(thresh1, rect_kernel, iterations = 1)
    contours,     hierarchy     =     cv2.findContours(dilation,
cv2.RETR_EXTERNAL,
                                                    cv2.CHAIN_APPROX_
NONE)
    contours,     hierarchy     =     cv2.findContours(dilation,
cv2.RETR_EXTERNAL,
                                                    cv2.CHAIN_APPROX_
NONE)
    im2 = img.copy()

    for cnt in contours:
        x, y, w, h = cv2.boundingRect(cnt)

        # Drawing a rectangle on copied image
        rect = cv2.rectangle(im2, (x, y), (x + w, y + h), (0, 255,
0), 1)

        # Cropping the text block for giving input to OCR
        cropped = im2[y:y + h, x:x + w]

    img2 = cv2.imshow(detect, im2)
```

TESTING TESSERACT MODEL

```
from pytesseract import*
import argparse
import cv2


# We construct the argument parser
# and parse the arguments
ap = argparse.ArgumentParser()

ap.add_argument("-i", "--image",
                required=True,
                help="path to input image to be OCR'd")
ap.add_argument("-c", "--min-conf",
                type=int, default=0,
                help="mininum confidence value to filter weak text
detection")
args = vars(ap.parse_args())

# We load the input image and then convert
# it to RGB from BGR. We then use Tesseract
# to localize each area of text in the input
# image
images = cv2.imread(args["image"])
rgb = cv2.cvtColor(images, cv2.COLOR_BGR2RGB)
results = pytesseract.image_to_data(rgb, output_type=Output.DICT)

# Then loop over each of the individual text
# localizations
for i in range(0, len(results["text"])):
```

```
# We can then extract the bounding box coordinates
# of the text region from the current result
x = results["left"][i]
y = results["top"][i]
w = results["width"][i]
h = results["height"][i]

# We will also extract the OCR text itself along
# with the confidence of the text localization
text = results["text"][i]
conf = int(results["conf"][i])

# filter out weak confidence text localizations
if conf > args["min_conf"]:

    # We will display the confidence and text to
    # our terminal
    print("Confidence: {}".format(conf))
    print("Text: {}".format(text))
    print("")

    # We then strip out non-ASCII text so we can
    # draw the text on the image We will be using
    # OpenCV, then draw a bounding box around the
    # text along with the text itself
    text = "".join(text).strip()
    cv2.rectangle(images,
                  (x, y),
                  (x + w, y + h),
                  (0, 0, 255), 2)
    '''
```

```
            cv2.putText(images,
                        text,
                        (x, y - 0),
                        cv2.FONT_HERSHEY_SIMPLEX,
                        1.2, (0, 255, 255), 3)
        '''


# After all, we will show the output image
cv2.imshow("Image", images)
#cv2.imwrite('tesseract_10.jpg', images)
print(len(results))
cv2.waitKey(0)
```

PLAGIARISM CHECK.ORG

6.53% PLAGIARISM APPROXIMATELY

Report #12287409

1.Introduction 1.1Background Today document is many kind of type. One of is text as documents, text contain of sentences and word and many kind of language. On this modern era you can save text documents in many ways for example you can save as soft copy and store at your computer, mobile phone, external hardisk, flashdisk, or cd. However in the past text documents was stored in the form of documents that was archived with print out paper or hard copy, to reduce paper today that methode is abandoned, therefore documents that were previously in the form paper or print out paper are better scanned and saved as document file. Because documents in the form of files have many conveniences one of which is easy to carry and easy to open from computer or mobile phone. Use an image tool to turn documents text into an image that can be saved as an image file, but the image can not be changed and sometimes an image scanner cannot scan the documents optimally due several causes such as dust on the surface of the scan tool and affecting the result of image to be saved, another way is to retype each word or sentences of the text document using a word processing program and save it in editable file format this method is a bit hassle if the document has a lot of words and sentences so it take a lot of time to retype. These problems make the processing of text document into