[ ]:

AI SiGN LANGUAGE RECOGNITION

November 21, 2022

.

[ ]: **from skimage import** transform **from skimage import** data

**import matplotlib.pyplot as plt import os**

**import numpy as np**

**from skimage.color import** rgb2gray **import random**

**import tensorflow as tf** .

[ ]: **def** load\_data(data\_directory):

directories = [d **for** d **in** os.listdir(data\_directory)

**if** os.path.isdir(os.path.join(data\_directory, d))]

labels = []

images = []

**for** d **in** directories:

label\_directory = os.path.join(data\_directory, d)

file\_names = [os.path.join(label\_directory, f) **for** f **in** os. *,→*listdir(label\_directory)]

**for** f **in** file\_names:

images.append(data.imread(f))

labels.append(ord(d))

**return** images, labels

ROOT\_PATH="../input/project"

train\_data\_directory=os.path.join(ROOT\_PATH, "train")

images, labels=load\_data(train\_data\_directory) .

[ ]: images\_array = np.array(images)

labels\_array = np.array(labels)

*# Print the numbezr of `images`'s elements* print("Total number of images:",images\_array.size) *# Count the number of labels*

print("Total No of classes:",len(set(labels\_array))) .

1

.print("Label Array: ",[chr(X) **for** X **in** set(labels)])

[ ]: *# Determine the (random) indexes of the images that you want to see* hand\_signs = [12,45,65,35]

*# Fill out the subplots with the random images that you defined* **for** i **in** range(len(hand\_signs)):

plt.subplot(1, 4, i+1)

plt.axis('off')

plt.imshow(images[hand\_signs[i]])

plt.subplots\_adjust(wspace=0.5)

plt.show() .

[ ]: *# Determine the (random) indexes of the images*

hand\_signs = [300, 1250, 2650, 3000]

*# Fill out the subplots with the random images and add shape, min and max values* **for** i **in** range(len(hand\_signs)):

plt.subplot(1, 4, i+1)

plt.axis('off')

plt.imshow(images[hand\_signs[i]])

plt.subplots\_adjust(wspace=0.5)

plt.show()

print("shape: **{0}**, min: **{1}**, max: **{2}**".format(images[hand\_signs[i]].shape, images[hand\_signs[i]].min(),

.

[ ]: *# Get the unique labels*

unique\_labels = set(labels)

*# Initialize the figure*

plt.figure(figsize=(15, 15))

*# Set a counter*

i = 1

*# For each unique label,*

**for** label **in** unique\_labels:

*# You pick the first image for each label* image = images[labels.index(label)]

*# Define 64 subplots*

plt.subplot(8, 8, i)

*# Don't include axes*

plt.axis('off')

*# Add a title to each subplot*

images[hand\_signs[i]].max()))

plt.title("Label **{0}** (**{1}**)".format(chr(label), labels.count(label))) .

2

*# Add 1 to the counter*

i += 1

*# And you plot this first image* plt.imshow(image)

*# Show the plot*

plt.show()

.

[ ]: *# Resize images*

images32 = [transform.resize(image, (28, 28,3)) **for** image **in** images]

images32 = np.array(images32) .

[ ]: images32 = rgb2gray(np.array(images32)) .

[ ]: **for** i **in** range(len(hand\_signs)):

plt.subplot(1, 4, i+1)

plt.axis('off')

plt.imshow(images32[hand\_signs[i]], cmap="gray") plt.subplots\_adjust(wspace=0.5)

plt.show()

print(images32.shape) .

[ ]: x = tf.placeholder(dtype = tf.float32, shape = [**None**, 28, 28]) y = tf.placeholder(dtype = tf.int32, shape = [**None**])

images\_flat = tf.contrib.layers.flatten(x)

logits = tf.contrib.layers.fully\_connected(images\_flat, 100, tf.nn.relu) loss = tf.reduce\_mean(tf.nn.sparse\_softmax\_cross\_entropy\_with\_logits(labels =␣ *,→*y, logits = logits))

train\_op = tf.train.AdamOptimizer(learning\_rate=0.001).minimize(loss) correct\_pred = tf.argmax(logits, 1)

accuracy = tf.reduce\_mean(tf.cast(correct\_pred, tf.float32))

print("images\_flat: ", images\_flat)

print("logits: ", logits)

print("loss: ", loss)

print("predicted\_labels: ", correct\_pred) .

[ ]: sess = tf.Session()

sess.run(tf.global\_variables\_initializer())

**for** i **in** range(201):

print('EPOCH', i)

\_, accuracy\_val = sess.run([train\_op, accuracy], feed\_dict={x: images32,␣

*,→*y: labels})

.

3

**if** i % 10 == 0:

print("Loss: ", loss)

print('DONE WITH EPOCH')

.

[ ]: *# Pick 10 random images*

sample\_indexes = random.sample(range(len(images32)), 10) sample\_images = [images32[i] **for** i **in** sample\_indexes]

sample\_labels = [labels[i] **for** i **in** sample\_indexes]

*# Run the "predicted\_labels" op.*

predicted = sess.run([correct\_pred], feed\_dict={x: sample\_images})[0]

*# Print the real and predicted labels*

print(sample\_labels)

print(predicted) .

[ ]: *# Display the predictions and the ground truth visually.*

fig = plt.figure(figsize=(10, 10))

**for** i **in** range(len(sample\_images)):

truth = sample\_labels[i]

prediction = predicted[i]

plt.subplot(5, 2,1+i)

plt.axis('off')

color='green' **if** truth == prediction **else** 'red'

plt.text(40, 10, "Truth: **{0}\n**Prediction: **{1}**".format(chr(truth),␣ *,→*chr(prediction)),

fontsize=12, color=color)

plt.imshow(sample\_images[i],cmap='gray')

plt.show() .

[ ]: sess.close()

.

4