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)))
```

```
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(),
                                                  images[hand_signs[i]].max()))
```

[ ]:
```
# 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
    plt.title("Label {0} ({1})".format(chr(label), labels.count(label)))
```

```
    # 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})
```

```
        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}\nPrediction: {1}".format(chr(truth),␣
 ↪chr(prediction)),
             fontsize=12, color=color)
    plt.imshow(sample_images[i],cmap='gray')

plt.show()
```

[ ]:
```
sess.close()
```