code-notebook

February 27, 2024

[11]:
```
! pip install kaggle
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: kaggle in
/home/kathir/.local/lib/python3.10/site-packages (1.5.16)
Requirement already satisfied: tqdm in /home/kathir/.local/lib/python3.10/site-
packages (from kaggle) (4.66.1)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages
(from kaggle) (6.1.0)
Requirement already satisfied: python-slugify in
/home/kathir/.local/lib/python3.10/site-packages (from kaggle) (8.0.1)
Requirement already satisfied: certifi in
/home/kathir/.local/lib/python3.10/site-packages (from kaggle) (2023.7.22)
Requirement already satisfied: python-dateutil in
/usr/local/lib/python3.10/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in
/home/kathir/.local/lib/python3.10/site-packages (from kaggle) (2.31.0)
Requirement already satisfied: urllib3 in
/home/kathir/.local/lib/python3.10/site-packages (from kaggle) (2.0.7)
Requirement already satisfied: six>=1.10 in /usr/lib/python3/dist-packages (from
kaggle) (1.16.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-
packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in
/home/kathir/.local/lib/python3.10/site-packages (from python-slugify->kaggle)
(1.3)
Requirement already satisfied: idna<4,>=2.5 in
/home/kathir/.local/lib/python3.10/site-packages (from requests->kaggle) (3.4)
Requirement already satisfied: charset-normalizer<4,>=2 in
/home/kathir/.local/lib/python3.10/site-packages (from requests->kaggle) (3.3.1)
```

[12]:
```
!pip install scikit-learn
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: scikit-learn in
/home/kathir/.local/lib/python3.10/site-packages (1.3.2)
Requirement already satisfied: numpy<2.0,>=1.17.3 in
/home/kathir/.local/lib/python3.10/site-packages (from scikit-learn) (1.26.1)
```

```
        File "/usr/lib/python3.10/distutils/dist.py", line 985, in
run_command
          cmd_obj.run()
        File "/usr/lib/python3.10/distutils/command/build_ext.py", line
340, in run
          self.build_extensions()
        File "/tmp/pip-
install-y7ybqy8f/pandas_7abd5f68d28241d484fc6ddebae9cec4/setup.py", line 372, in
build_extensions
          self.check_cython_extensions(self.extensions)
        File "/tmp/pip-
install-y7ybqy8f/pandas_7abd5f68d28241d484fc6ddebae9cec4/setup.py", line 366, in
check_cython_extensions
          raise Exception("""Cython-generated file '{src}' not found.
      Exception: Cython-generated file 'pandas/_libs/algos.c' not
found.
                      Cython is required to compile pandas from a
development branch.
                      Please install Cython or download a release
package of pandas.

      [end of output]

  note: This error originates from a subprocess, and is likely not a
problem with pip.
error: legacy-install-failure

× Encountered error while trying to install package.
 > pandas

note: This is an issue with the package mentioned above, not pip.
hint: See above for output from the failure.
Note: you may need to restart the kernel to use updated packages.
```

[23]:
```
import sklearn
```

[24]:
```
#mkdir ~/.kaggle
!touch /mnt/c//Users/kathi/Desktop/kaggle.json

api_token = {"username":"username","key":"api-key"}

import json

with open('/mnt/c//Users/kathi/Desktop/kaggle (2).json', 'w') as file:
    json.dump(api_token, file)

!chmod 600 ~/.kaggle/kaggle.json
```

```
[25]: from google.colab import drive
      drive.mount('/content/gdrive')
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[25], line 1
----> 1 from google.colab import drive
      2 drive.mount('/content/gdrive')

ModuleNotFoundError: No module named 'google.colab'
```

```
[ ]: import os
     os.environ['KAGGLE_CONFIG_DIR'] = "/content/gdrive/My Drive/Kaggle"
```

```
[ ]: %cd /content/gdrive/My Drive/Kaggle
```

```
[ ]: ! kaggle datasets download -d abishek1301/coins-dataset
```

```
[ ]: ! unzip coins-dataset.zip
```

```
[ ]: import numpy as np
     import seaborn as sns #seaborn module here is used to plot confusion matrix
     import matplotlib.pyplot as plt
     from sklearn.metrics import ConfusionMatrixDisplay
     import tensorflow as tf
     from tensorflow import keras
     from tensorflow.keras.layers import *
     from sklearn.metrics import classification_report
```

```
[ ]: import tensorflow as tf
     tf.test.gpu_device_name()
```

```
[ ]: #Image preprocessing. Keras has an inbuilt preprocessing function
     image_size = (250, 250)
     batch = 32

     train = keras.preprocessing.image_dataset_from_directory(
         'DataSet/',
         validation_split=.2, #here we take 20% of the data for validation
         subset='training',
         seed=42, #If the seed value is not set (or different seed values are used),␣
      ↪two different objects will produce different results.
                   #Since the random seed is set at the beginning of the notebook,␣
      ↪the results will be same in the sequential runs.
         image_size=image_size, #Resizes the images that the computer reads from the␣
      ↪disk to (250,250)
```

```
    batch_size=batch, #Size of the batches of the data that the model␣
↪segregates, defaults to 32 anyway
    label_mode='categorical'
)
```

[ ]:
```
validation = keras.preprocessing.image_dataset_from_directory(
    'DataSet/',
    validation_split=.2,
    subset='validation',
    seed=42,
    image_size=image_size,
    batch_size=batch,
    label_mode='categorical'
)
```

Now, we show how the image resizing affects the quality of the images that are used by the system for classification. The first attempt uses(50,50) size, second uses (100,100) size, third uses (250,250) size

[ ]:
```
def display_samples(dataset, n_samples, classes_name):
    plt.figure(figsize=(10, 10))
    for images, labels in dataset.take(1):
        for i in range(n_samples):
            ax = plt.subplot(3, 3, i + 1)
            plt.imshow(images[i].numpy().astype("uint8"))
            plt.title(classes_name[np.argmax(labels[i])])
            plt.axis("off")
display_samples(train, 9, train.class_names)
for images, labels in train.take(1):
        for i in range(1):
            print(images[i].shape)
```

[ ]:
```
def display_samples(dataset, n_samples, classes_name):
    plt.figure(figsize=(10, 10))
    for images, labels in dataset.take(1):
        for i in range(n_samples):
            ax = plt.subplot(3, 3, i + 1)
            plt.imshow(images[i].numpy().astype("uint8"))
            plt.title(classes_name[np.argmax(labels[i])])
            plt.axis("off")
display_samples(train, 9, train.class_names)
for images, labels in train.take(1):
        for i in range(1):
            print(images[i].shape)
```

[ ]:
```
def display_samples(dataset, n_samples, classes_name):
    plt.figure(figsize=(10, 10))
    for images, labels in dataset.take(1):
        for i in range(n_samples):
            ax = plt.subplot(3, 3, i + 1)
            plt.imshow(images[i].numpy().astype("uint8"))
            plt.title(classes_name[np.argmax(labels[i])])
            plt.axis("off")
display_samples(train, 9, train.class_names)
for images, labels in train.take(1):
        for i in range(1):
            print(images[i].shape)
```

[ ]:
```
class_names = train.class_names
labels = np.array([])
for _, label in train:
    labels = np.concatenate((labels, np.argmax(label, axis=-1)))
_, counts = np.unique(labels, return_counts=True)
counts
```

[ ]:
```
input_shape = (image_size[0], image_size[1], 3)# This means that the image is␣
 ↪the tensor with 250 elements in the first and second dimension,
                                               # 3 elements in the third
reg = keras.regularizers.l2(0.0005) #Here, regularizer helps us put a penalty␣
 ↪on larger layer parameters, or simply, put a penalty on higher density images
                                    #l2 is just the regularization parameter

model = keras.Sequential() #Sequential provides training and inference models␣
 ↪to the object associated. Basically if you run 10 epochs, then reset the␣
 ↪runtime
                           #and run the model again for 2 epochs, the accuracy␣
 ↪for the recent 2 epochs will be higher than the 10th epoch accuracy.
model.add(Conv2D(32, (3, 3), padding="same", activation="relu",␣
 ↪input_shape=image_size + (3,), kernel_regularizer=reg))
model.add(MaxPooling2D(pool_size=(2, 2)))


model.add(Conv2D(64, (3, 3), padding="same", activation="relu",␣
 ↪kernel_regularizer=reg))
model.add(MaxPooling2D(pool_size=(2, 2)))


model.add(Conv2D(128, (3, 3), padding="same", activation="relu",␣
 ↪kernel_regularizer=reg))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(len(train.class_names), activation='softmax'))


model.summary()
```

Starting the model training

```
[ ]: model.compile(
         loss="categorical_crossentropy",
         optimizer="adam",
         metrics=["accuracy"]
     )

     epochs = 10
     model.fit(
         train,
         epochs=epochs,
         validation_data=validation
     );
```

Plotting graphs for training and validation data performance

```
[ ]: epochs_range = [i+1 for i in range(epochs)]
     plt.plot(epochs_range, model.history.history['accuracy'], '-o', label='Train')
     plt.plot(epochs_range, model.history.history['val_accuracy'],␣
       ↪'-x',label='Validation')

     plt.ylabel('Accuracy')
     plt.xlabel('Epochs')

     plt.legend()
```

Plotting confusion matrix

```
[ ]: y_pred = np.argmax(model.predict(validation), axis=-1)

     predictions = np.array([])
     labels =  np.array([])
     for x, y in validation:
         predictions = np.concatenate([predictions, np.argmax(model.predict(x),␣
       ↪axis=-1)])
         labels = np.concatenate([labels, np.argmax(y.numpy(), axis=-1)])
```

```
[ ]: conf = tf.math.confusion_matrix(labels=labels, predictions=predictions)
     sns.heatmap(conf, annot=True, cmap='Blues', yticklabels=class_names,␣
      ↪xticklabels=class_names)
```