|  |
| --- |
| Import os |
|  | import zipfile |
|  | import random |
|  | import tensorflow as tf |
|  | from tensorflow.keras.optimizers import RMSprop |
|  | from tensorflow.keras.preprocessing.image import ImageDataGenerator |
|  | from shutil import copyfile |
|  |  |
|  | # If the URL doesn't work, visit https://www.microsoft.com/en-us/download/confirmation.aspx?id=54765 |
|  | # And right click on the 'Download Manually' link to get a new URL to the dataset |
|  |  |
|  | # Note: This is a very large dataset and will take time to download |
|  |  |
|  | !wget --no-check-certificate \ |
|  | "https://download.microsoft.com/download/3/E/1/3E1C3F21-ECDB-4869-8368-6DEBA77B919F/kagglecatsanddogs\_3367a.zip" \ |
|  | -O "/tmp/cats-and-dogs.zip" |
|  |  |
|  | local\_zip = '/tmp/cats-and-dogs.zip' |
|  | zip\_ref = zipfile.ZipFile(local\_zip, 'r') |
|  | zip\_ref.extractall('/tmp') |
|  | zip\_ref.close() |
|  |  |
|  | print(len(os.listdir('/tmp/PetImages/Cat/'))) |
|  | print(len(os.listdir('/tmp/PetImages/Dog/'))) |
|  |  |
|  | try: |
|  | os.mkdir('/tmp/cats-v-dogs') |
|  | os.mkdir('/tmp/cats-v-dogs/training') |
|  | os.mkdir('/tmp/cats-v-dogs/testing') |
|  | os.mkdir('/tmp/cats-v-dogs/training/cats') |
|  | os.mkdir('/tmp/cats-v-dogs/training/dogs') |
|  | os.mkdir('/tmp/cats-v-dogs/testing/cats') |
|  | os.mkdir('/tmp/cats-v-dogs/testing/dogs') |
|  | except OSError: |
|  | pass |
|  |  |
|  | def split\_data(SOURCE, TRAINING, TESTING, SPLIT\_SIZE): |
|  | files = [] |
|  | for filename in os.listdir(SOURCE): |
|  | file = SOURCE + filename |
|  | if os.path.getsize(file) > 0: |
|  | files.append(filename) |
|  | else: |
|  | print(filename + " is zero length, so ignoring.") |
|  |  |
|  | training\_length = int(len(files) \* SPLIT\_SIZE) |
|  | testing\_length = int(len(files) - training\_length) |
|  | shuffled\_set = random.sample(files, len(files)) |
|  | training\_set = shuffled\_set[0:training\_length] |
|  | testing\_set = shuffled\_set[-testing\_length:] |
|  |  |
|  | for filename in training\_set: |
|  | this\_file = SOURCE + filename |
|  | destination = TRAINING + filename |
|  | copyfile(this\_file, destination) |
|  |  |
|  | for filename in testing\_set: |
|  | this\_file = SOURCE + filename |
|  | destination = TESTING + filename |
|  | copyfile(this\_file, destination) |
|  |  |
|  |  |
|  | CAT\_SOURCE\_DIR = "/tmp/PetImages/Cat/" |
|  | TRAINING\_CATS\_DIR = "/tmp/cats-v-dogs/training/cats/" |
|  | TESTING\_CATS\_DIR = "/tmp/cats-v-dogs/testing/cats/" |
|  | DOG\_SOURCE\_DIR = "/tmp/PetImages/Dog/" |
|  | TRAINING\_DOGS\_DIR = "/tmp/cats-v-dogs/training/dogs/" |
|  | TESTING\_DOGS\_DIR = "/tmp/cats-v-dogs/testing/dogs/" |
|  |  |
|  | split\_size = .9 |
|  | split\_data(CAT\_SOURCE\_DIR, TRAINING\_CATS\_DIR, TESTING\_CATS\_DIR, split\_size) |
|  | split\_data(DOG\_SOURCE\_DIR, TRAINING\_DOGS\_DIR, TESTING\_DOGS\_DIR, split\_size) |
|  |  |
|  | print(len(os.listdir('/tmp/cats-v-dogs/training/cats/'))) |
|  | print(len(os.listdir('/tmp/cats-v-dogs/training/dogs/'))) |
|  | print(len(os.listdir('/tmp/cats-v-dogs/testing/cats/'))) |
|  | print(len(os.listdir('/tmp/cats-v-dogs/testing/dogs/'))) |
|  |  |
|  | model = tf.keras.models.Sequential([ |
|  | tf.keras.layers.Conv2D(16, (3, 3), activation='relu', input\_shape=(150, 150, 3)), |
|  | tf.keras.layers.MaxPooling2D(2, 2), |
|  | tf.keras.layers.Conv2D(32, (3, 3), activation='relu'), |
|  | tf.keras.layers.MaxPooling2D(2, 2), |
|  | tf.keras.layers.Conv2D(64, (3, 3), activation='relu'), |
|  | tf.keras.layers.MaxPooling2D(2, 2), |
|  | tf.keras.layers.Flatten(), |
|  | tf.keras.layers.Dense(512, activation='relu'), |
|  | tf.keras.layers.Dense(1, activation='sigmoid') |
|  | ]) |
|  |  |
|  | model.compile(optimizer=RMSprop(lr=0.001), loss='binary\_crossentropy', metrics=['acc']) |
|  |  |
|  |  |
|  | TRAINING\_DIR = "/tmp/cats-v-dogs/training/" |
|  | train\_datagen = ImageDataGenerator(rescale=1.0/255.) |
|  | train\_generator = train\_datagen.flow\_from\_directory(TRAINING\_DIR, |
|  | batch\_size=100, |
|  | class\_mode='binary', |
|  | target\_size=(150, 150)) |
|  |  |
|  | VALIDATION\_DIR = "/tmp/cats-v-dogs/testing/" |
|  | validation\_datagen = ImageDataGenerator(rescale=1.0/255.) |
|  | validation\_generator = validation\_datagen.flow\_from\_directory(VALIDATION\_DIR, |
|  | batch\_size=100, |
|  | class\_mode='binary', |
|  | target\_size=(150, 150)) |
|  |  |
|  | # Note that this may take some time. |
|  | history = model.fit\_generator(train\_generator, |
|  | epochs=4, |
|  | verbose=1, |
|  | validation\_data=validation\_generator) |
|  |  |
|  |  |
|  |  |
|  | %matplotlib inline |
|  |  |
|  | import matplotlib.image as mpimg |
|  | import matplotlib.pyplot as plt |
|  |  |
|  | #----------------------------------------------------------- |
|  | # Retrieve a list of list results on training and test data |
|  | # sets for each training epoch |
|  | #----------------------------------------------------------- |
|  | acc=history.history['acc'] |
|  | val\_acc=history.history['val\_acc'] |
|  | loss=history.history['loss'] |
|  | val\_loss=history.history['val\_loss'] |
|  |  |
|  | epochs=range(len(acc)) # Get number of epochs |
|  |  |
|  | #------------------------------------------------ |
|  | # Plot training and validation accuracy per epoch |
|  | #------------------------------------------------ |
|  | plt.plot(epochs, acc, 'r', "Training Accuracy") |
|  | plt.plot(epochs, val\_acc, 'b', "Validation Accuracy") |
|  | plt.title('Training and validation accuracy') |
|  | plt.figure() |
|  |  |
|  | #------------------------------------------------ |
|  | # Plot training and validation loss per epoch |
|  | #------------------------------------------------ |
|  | plt.plot(epochs, loss, 'r', "Training Loss") |
|  | plt.plot(epochs, val\_loss, 'b', "Validation Loss") |
|  | plt.figure() |
|  |  |
|  | # Here's a codeblock just for fun. You should be able to upload an image here |
|  | # and have it classified without crashing |
|  | import numpy as np |
|  | from google.colab import files |
|  | from keras.preprocessing import image |
|  |  |
|  | uploaded = files.upload() |
|  |  |
|  | for fn in uploaded.keys(): |
|  |  |
|  | # predicting images |
|  | path = '/content/' + fn |
|  | img = image.load\_img(path, target\_size=(150, 150)) |
|  | x = image.img\_to\_array(img) |
|  | x = np.expand\_dims(x, axis=0) |
|  |  |
|  | images = np.vstack([x]) |
|  | classes = model.predict(images, batch\_size=10) |
|  | print(classes[0]) |
|  | if classes[0]>0.5: |
|  | print(fn + " is a dog") |
|  | else: |
|  | print(fn + " is a cat") |