|  |
| --- |
| from google.colab import drive |
|  | drive.mount('/content/drive') |
|  | Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force\_remount=True). |
|  | from tensorflow.keras.layers import Dense, Flatten, Input |
|  | from tensorflow.keras.models import Model |
|  | from tensorflow.keras.preprocessing import image |
|  | from tensorflow.keras.preprocessing.image import ImageDataGenerator, load\_img |
|  | from tensorflow.keras.applications.vgg16 import VGG16, preprocess\_input |
|  | from glob import glob |
|  | import numpy as np |
|  | import matplotlib.pyplot as plt |
|  | imageSize = [224, 224] |
|  |  |
|  | trainPath = r"/content/drive/MyDrive/dataset1/body/training" |
|  |  |
|  | testPath = r"/content/drive/MyDrive/dataset1/body/validation" |
|  | # adding preprocessing layers to the front of vgg |
|  |  |
|  | vgg = VGG16(input\_shape=imageSize + [3], weights='imagenet',include\_top=False) |
|  | Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16\_weights\_tf\_dim\_ordering\_tf\_kernels\_notop.h5 |
|  | 58889256/58889256 [==============================] - 0s 0us/step |
|  | # don't train existing weights |
|  | for layer in vgg.layers: |
|  | layer.trainable = False |
|  | # our layers - you can add more if you want |
|  | x = Flatten()(vgg.output) |
|  | prediction = Dense(3, activation='softmax')(x) |
|  | # create a model object |
|  | model = Model(inputs=vgg.input, outputs=prediction) |
|  | # view the structure of the model |
|  | model.summary() |
|  | Model: "model" |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | Layer (type) Output Shape Param # |
|  | ================================================================= |
|  | input\_1 (InputLayer) [(None, 224, 224, 3)] 0 |
|  |  |
|  | block1\_conv1 (Conv2D) (None, 224, 224, 64) 1792 |
|  |  |
|  | block1\_conv2 (Conv2D) (None, 224, 224, 64) 36928 |
|  |  |
|  | block1\_pool (MaxPooling2D) (None, 112, 112, 64) 0 |
|  |  |
|  | block2\_conv1 (Conv2D) (None, 112, 112, 128) 73856 |
|  |  |
|  | block2\_conv2 (Conv2D) (None, 112, 112, 128) 147584 |
|  |  |
|  | block2\_pool (MaxPooling2D) (None, 56, 56, 128) 0 |
|  |  |
|  | block3\_conv1 (Conv2D) (None, 56, 56, 256) 295168 |
|  |  |
|  | block3\_conv2 (Conv2D) (None, 56, 56, 256) 590080 |
|  |  |
|  | block3\_conv3 (Conv2D) (None, 56, 56, 256) 590080 |
|  |  |
|  | block3\_pool (MaxPooling2D) (None, 28, 28, 256) 0 |
|  |  |
|  | block4\_conv1 (Conv2D) (None, 28, 28, 512) 1180160 |
|  |  |
|  | block4\_conv2 (Conv2D) (None, 28, 28, 512) 2359808 |
|  |  |
|  | block4\_conv3 (Conv2D) (None, 28, 28, 512) 2359808 |
|  |  |
|  | block4\_pool (MaxPooling2D) (None, 14, 14, 512) 0 |
|  |  |
|  | block5\_conv1 (Conv2D) (None, 14, 14, 512) 2359808 |
|  |  |
|  | block5\_conv2 (Conv2D) (None, 14, 14, 512) 2359808 |
|  |  |
|  | block5\_conv3 (Conv2D) (None, 14, 14, 512) 2359808 |
|  |  |
|  | block5\_pool (MaxPooling2D) (None, 7, 7, 512) 0 |
|  |  |
|  | flatten (Flatten) (None, 25088) 0 |
|  |  |
|  | dense (Dense) (None, 3) 75267 |
|  |  |
|  | ================================================================= |
|  | Total params: 14,789,955 |
|  | Trainable params: 75,267 |
|  | Non-trainable params: 14,714,688 |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | # tell the model what cost and optimization method to use |
|  | model.compile( |
|  | loss='categorical\_crossentropy', |
|  | optimizer='adam', |
|  | metrics=['accuracy'] |
|  | ) |
|  | train\_datagen = ImageDataGenerator(rescale = 1./255, |
|  | shear\_range = 0.2, |
|  | zoom\_range = 0.2, |
|  | horizontal\_flip = True) |
|  |  |
|  | test\_datagen = ImageDataGenerator(rescale = 1./255) |
|  | training\_set = train\_datagen.flow\_from\_directory(trainPath, |
|  | target\_size = (224, 224), |
|  | batch\_size = 10, |
|  | class\_mode = 'categorical') |
|  |  |
|  | test\_set = test\_datagen.flow\_from\_directory(testPath, |
|  | target\_size = (224, 224), |
|  | batch\_size = 10, |
|  | class\_mode = 'categorical') |
|  | Found 979 images belonging to 3 classes. |
|  | Found 171 images belonging to 3 classes. |
|  | import sys |
|  | # fit the model |
|  | r = model.fit\_generator( |
|  | training\_set, |
|  | validation\_data=test\_set, |
|  | epochs=10, |
|  | steps\_per\_epoch=979//10, |
|  | validation\_steps=171//10) |
|  | /usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:8: UserWarning: `Model.fit\_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators. |
|  |  |
|  | Epoch 1/10 |
|  | 97/97 [==============================] - 596s 6s/step - loss: 1.2100 - accuracy: 0.5253 - val\_loss: 1.0260 - val\_accuracy: 0.6294 |
|  | Epoch 2/10 |
|  | 97/97 [==============================] - 588s 6s/step - loss: 0.7407 - accuracy: 0.7110 - val\_loss: 0.9575 - val\_accuracy: 0.6706 |
|  | Epoch 3/10 |
|  | 97/97 [==============================] - 589s 6s/step - loss: 0.6132 - accuracy: 0.7534 - val\_loss: 0.8389 - val\_accuracy: 0.7000 |
|  | Epoch 4/10 |
|  | 97/97 [==============================] - 587s 6s/step - loss: 0.4645 - accuracy: 0.8390 - val\_loss: 1.3165 - val\_accuracy: 0.6412 |
|  | Epoch 5/10 |
|  | 97/97 [==============================] - 589s 6s/step - loss: 0.4019 - accuracy: 0.8514 - val\_loss: 1.0316 - val\_accuracy: 0.6529 |
|  | Epoch 6/10 |
|  | 97/97 [==============================] - 588s 6s/step - loss: 0.2716 - accuracy: 0.8999 - val\_loss: 1.0882 - val\_accuracy: 0.6529 |
|  | Epoch 7/10 |
|  | 97/97 [==============================] - 594s 6s/step - loss: 0.2722 - accuracy: 0.9071 - val\_loss: 1.0481 - val\_accuracy: 0.6765 |
|  | Epoch 8/10 |
|  | 97/97 [==============================] - 592s 6s/step - loss: 0.2265 - accuracy: 0.9289 - val\_loss: 1.3173 - val\_accuracy: 0.6059 |
|  | Epoch 9/10 |
|  | 97/97 [==============================] - 593s 6s/step - loss: 0.2981 - accuracy: 0.8751 - val\_loss: 1.1330 - val\_accuracy: 0.6941 |
|  | Epoch 10/10 |
|  | 97/97 [==============================] - 592s 6s/step - loss: 0.2247 - accuracy: 0.9123 - val\_loss: 1.5393 - val\_accuracy: 0.5706 |
|  | #save the model |
|  | model.save('body.h5') |
|  | #import load\_model class for loading h5 file |
|  | from tensorflow.keras.models import load\_model |
|  | #import image class to process the images |
|  | from tensorflow.keras.preprocessing import image |
|  | from tensorflow.keras.applications.inception\_v3 import preprocess\_input |
|  | import numpy as np |
|  | #load one random image from local system |
|  | img=image.load\_img(r'/content/drive/MyDrive/dataset1/body/training/00-front/0002.JPEG',target\_size=(224,224)) |
|  | #convert image to array format |
|  | x=image.img\_to\_array(img) |
|  | import numpy as np |
|  | x=np.expand\_dims(x,axis=0) |
|  | img\_data=preprocess\_input(x) |
|  | img\_data.shape |
|  | (1, 224, 224, 3) |
|  | img\_data.shape |
|  | (1, 224, 224, 3) |
|  | model.predict(img\_data) |
|  | 1/1 [==============================] - 1s 732ms/step |
|  | array([[9.9837422e-01, 1.6256354e-03, 7.2354190e-08]], dtype=float32) |
|  | output=np.argmax(model.predict(img\_data), axis=1) |
|  | output |
|  | 1/1 [==============================] - 1s 535ms/step |
|  | array([0]) |
|  | imageSize = [224, 224] |
|  |  |
|  | trainPath = r"/content/drive/MyDrive/dataset1/level/training" |
|  |  |
|  | testPath = r"/content/drive/MyDrive/dataset1/level/validation" |
|  | vgg1 = VGG16(input\_shape=imageSize + [3], weights='imagenet',include\_top=False) |
|  | for layer in vgg1.layers: |
|  | layer.trainable = False |
|  | # our layers - you can add more if you want |
|  | x = Flatten()(vgg1.output) |
|  | prediction = Dense(3, activation='softmax')(x) |
|  | # create a model object |
|  | model1 = Model(inputs=vgg1.input, outputs=prediction) |
|  | # tell the model what cost and optimization method to use |
|  | model1.compile( |
|  | loss='categorical\_crossentropy', |
|  | optimizer='adam', |
|  | metrics=['accuracy'] |
|  | ) |
|  | train\_datagen = ImageDataGenerator(rescale = 1./255, |
|  | shear\_range = 0.2, |
|  | zoom\_range = 0.2, |
|  | horizontal\_flip = True) |
|  |  |
|  | test\_datagen = ImageDataGenerator(rescale = 1./255) |
|  | training\_set = train\_datagen.flow\_from\_directory(trainPath, |
|  | target\_size = (224, 224), |
|  | batch\_size = 10, |
|  | class\_mode = 'categorical') |
|  |  |
|  | test\_set = test\_datagen.flow\_from\_directory(testPath, |
|  | target\_size = (224, 224), |
|  | batch\_size = 10, |
|  | class\_mode = 'categorical') |
|  | Found 979 images belonging to 3 classes. |
|  | Found 171 images belonging to 3 classes. |
|  | r = model1.fit\_generator( |
|  | training\_set, |
|  | validation\_data=test\_set, |
|  | epochs=10, |
|  | steps\_per\_epoch=979//10, |
|  | validation\_steps=171//10) |
|  | /usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:6: UserWarning: `Model.fit\_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators. |
|  |  |
|  | Epoch 1/10 |
|  | 97/97 [==============================] - 600s 6s/step - loss: 0.9744 - accuracy: 0.6336 - val\_loss: 0.9600 - val\_accuracy: 0.6353 |
|  | Epoch 2/10 |
|  | 97/97 [==============================] - 594s 6s/step - loss: 0.7424 - accuracy: 0.7069 - val\_loss: 0.9975 - val\_accuracy: 0.6353 |
|  | Epoch 3/10 |
|  | 97/97 [==============================] - 595s 6s/step - loss: 0.5972 - accuracy: 0.7812 - val\_loss: 1.1393 - val\_accuracy: 0.6176 |
|  | Epoch 4/10 |
|  | 97/97 [==============================] - 594s 6s/step - loss: 0.4651 - accuracy: 0.8122 - val\_loss: 1.1309 - val\_accuracy: 0.5941 |
|  | Epoch 5/10 |
|  | 97/97 [==============================] - 595s 6s/step - loss: 0.3979 - accuracy: 0.8349 - val\_loss: 1.1914 - val\_accuracy: 0.5706 |
|  | Epoch 6/10 |
|  | 97/97 [==============================] - 595s 6s/step - loss: 0.3280 - accuracy: 0.8689 - val\_loss: 1.2503 - val\_accuracy: 0.5824 |
|  | Epoch 7/10 |
|  | 97/97 [==============================] - 596s 6s/step - loss: 0.3338 - accuracy: 0.8741 - val\_loss: 1.0894 - val\_accuracy: 0.6176 |
|  | Epoch 8/10 |
|  | 97/97 [==============================] - 599s 6s/step - loss: 0.2654 - accuracy: 0.9123 - val\_loss: 1.1027 - val\_accuracy: 0.6471 |
|  | Epoch 9/10 |
|  | 97/97 [==============================] - 595s 6s/step - loss: 0.2324 - accuracy: 0.9143 - val\_loss: 1.2071 - val\_accuracy: 0.6118 |
|  | Epoch 10/10 |
|  | 97/97 [==============================] - 596s 6s/step - loss: 0.1750 - accuracy: 0.9381 - val\_loss: 1.1278 - val\_accuracy: 0.6353 |
|  | #save the model |
|  | model.save('level.h5') |