six_classify

September 7, 2020

[1]:
```
# Load the TensorBoard notebook extension
%load_ext tensorboard
```

[2]:
```
import tensorflow as tf
from keras import layers
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
import keras.backend as K
K.set_image_data_format('channels_last')

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.pyplot import imshow
import datetime
from alexnet import AlexNet
```

[3]:
```
# Set the GPU growth in order to avoid the sudden stop of the runtime.
gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)
```

[4]:
```
# Give the global constants.Please notify BATCH_SIZE for model.fit() and␣
 ↪Batch_Size for
# model.evaluate() and model.predict()
EPOCHS = 50
BATCH_SIZE = 32
Batch_Size = 1
image_width = 227
image_height = 227
channels = 3
num_classes = 6
```

[5]:
```
# Call the alexnet model in alexnet.py
model = AlexNet((image_width,image_height,channels), num_classes)
```

[6]:
```
# Compile the model
model.compile(optimizer=tf.keras.optimizers.Adam(0.001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

[7]:

```
# It will output the AlexNet model after executing the command
model.summary()
```

```
Model: "alex_net"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 55, 55, 96)        34944
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 27, 27, 96)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 27, 27, 256)       614656
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 256)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 13, 13, 384)       885120
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 13, 13, 384)       1327488
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 13, 13, 256)       884992
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 9216)              0
_________________________________________________________________
dense (Dense)                (None, 4096)              37752832
_________________________________________________________________
dropout (Dropout)            (None, 4096)              0
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              16781312
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_2 (Dense)              (None, 1000)              4097000
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 6006
=================================================================
Total params: 62,384,350
Trainable params: 62,384,350
Non-trainable params: 0
_________________________________________________________________
```

[8]:

```
train_dir = '/home/mike/Documents/Six_Classify_AlexNet/seg_train/seg_train'
test_dir = '/home/mike/Documents/Six_Classify_AlexNet/seg_test/seg_test'
```

```
predict_dir = '/home/mike/Documents/Six_Classify_AlexNet/seg_pred/'
```

[9]:

```
# keras.preprocessing.image.ImageDataGenerator
train_datagen = ImageDataGenerator(rescale=1.0/255)

# keras.preprocessing.image.DirectoryIterator
train_generator = train_datagen.flow_from_directory(train_dir,
                                                   ␣
 ↪target_size=(image_width,image_height),
                                                    class_mode='categorical')

train_num = train_generator.samples
```

```
Found 14034 images belonging to 6 classes.
```

[10]:

```
# Start Tensorboard --logdir logs/fit
log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir)
callback_list = [tensorboard_callback]
```

[11]:

```
# Set verbose=1 (or verbose=0) for visibale (or invisible) training procedure.
model.fit(train_generator,
          epochs=EPOCHS,
          steps_per_epoch=train_num//BATCH_SIZE,
          callbacks=callback_list,
          verbose=1)
```

```
Epoch 1/50
438/438 [==============================] - 24s 55ms/step - loss: 1.7000 -
accuracy: 0.3986
Epoch 2/50
438/438 [==============================] - 16s 36ms/step - loss: 1.0280 -
accuracy: 0.6016
Epoch 3/50
438/438 [==============================] - 16s 36ms/step - loss: 0.9226 -
accuracy: 0.6450
Epoch 4/50
438/438 [==============================] - 16s 36ms/step - loss: 0.8542 -
accuracy: 0.6784
Epoch 5/50
438/438 [==============================] - 16s 36ms/step - loss: 0.8028 -
accuracy: 0.7041
Epoch 6/50
438/438 [==============================] - 16s 36ms/step - loss: 0.7288 -
accuracy: 0.7337
Epoch 7/50
438/438 [==============================] - 16s 36ms/step - loss: 0.6962 -
```

```
accuracy: 0.7483
Epoch 8/50
438/438 [==============================] - 16s 36ms/step - loss: 0.6328 -
accuracy: 0.7779
Epoch 9/50
438/438 [==============================] - 16s 36ms/step - loss: 0.6266 -
accuracy: 0.7769
Epoch 10/50
438/438 [==============================] - 16s 36ms/step - loss: 0.5795 -
accuracy: 0.7966
Epoch 11/50
438/438 [==============================] - 16s 36ms/step - loss: 0.5597 -
accuracy: 0.8065
Epoch 12/50
438/438 [==============================] - 16s 36ms/step - loss: 0.5405 -
accuracy: 0.8154
Epoch 13/50
438/438 [==============================] - 16s 36ms/step - loss: 0.5115 -
accuracy: 0.8248
Epoch 14/50
438/438 [==============================] - 16s 36ms/step - loss: 0.4910 -
accuracy: 0.8287
Epoch 15/50
438/438 [==============================] - 16s 36ms/step - loss: 0.4928 -
accuracy: 0.8304
Epoch 16/50
438/438 [==============================] - 16s 36ms/step - loss: 0.4638 -
accuracy: 0.8388
Epoch 17/50
438/438 [==============================] - 16s 36ms/step - loss: 0.4366 -
accuracy: 0.8496
Epoch 18/50
438/438 [==============================] - 16s 36ms/step - loss: 0.4207 -
accuracy: 0.8579
Epoch 19/50
438/438 [==============================] - 16s 36ms/step - loss: 0.4173 -
accuracy: 0.8602
Epoch 20/50
438/438 [==============================] - 16s 37ms/step - loss: 0.3991 -
accuracy: 0.8627
Epoch 21/50
438/438 [==============================] - 16s 37ms/step - loss: 0.4020 -
accuracy: 0.8672
Epoch 22/50
438/438 [==============================] - 16s 37ms/step - loss: 0.3782 -
accuracy: 0.8740
Epoch 23/50
438/438 [==============================] - 16s 37ms/step - loss: 0.3824 -
```

```
accuracy: 0.8749
Epoch 24/50
438/438 [==============================] - 16s 37ms/step - loss: 0.3312 -
accuracy: 0.8887
Epoch 25/50
438/438 [==============================] - 17s 38ms/step - loss: 0.3652 -
accuracy: 0.8805
Epoch 26/50
438/438 [==============================] - 16s 37ms/step - loss: 0.3286 -
accuracy: 0.8927
Epoch 27/50
438/438 [==============================] - 16s 37ms/step - loss: 0.3036 -
accuracy: 0.9005
Epoch 28/50
438/438 [==============================] - 16s 37ms/step - loss: 0.3273 -
accuracy: 0.8935
Epoch 29/50
438/438 [==============================] - 17s 38ms/step - loss: 0.3107 -
accuracy: 0.8967
Epoch 30/50
438/438 [==============================] - 16s 37ms/step - loss: 0.2917 -
accuracy: 0.9077
Epoch 31/50
438/438 [==============================] - 17s 38ms/step - loss: 0.2435 -
accuracy: 0.9214
Epoch 32/50
438/438 [==============================] - 17s 39ms/step - loss: 0.2651 -
accuracy: 0.9169
Epoch 33/50
438/438 [==============================] - 17s 39ms/step - loss: 0.3002 -
accuracy: 0.9064
Epoch 34/50
438/438 [==============================] - 17s 39ms/step - loss: 0.2608 -
accuracy: 0.9171
Epoch 35/50
438/438 [==============================] - 17s 38ms/step - loss: 0.2831 -
accuracy: 0.9099
Epoch 36/50
438/438 [==============================] - 17s 38ms/step - loss: 0.2871 -
accuracy: 0.9099
Epoch 37/50
438/438 [==============================] - 16s 37ms/step - loss: 0.2332 -
accuracy: 0.9262
Epoch 38/50
438/438 [==============================] - 16s 37ms/step - loss: 0.2171 -
accuracy: 0.9330
Epoch 39/50
438/438 [==============================] - 16s 37ms/step - loss: 0.2182 -
```

```
accuracy: 0.9309
Epoch 40/50
438/438 [==============================] - 16s 37ms/step - loss: 0.2403 -
accuracy: 0.9259
Epoch 41/50
438/438 [==============================] - 17s 38ms/step - loss: 0.5131 -
accuracy: 0.8249
Epoch 42/50
438/438 [==============================] - 17s 38ms/step - loss: 0.3985 -
accuracy: 0.8719
Epoch 43/50
438/438 [==============================] - 16s 38ms/step - loss: 0.4105 -
accuracy: 0.8699
Epoch 44/50
438/438 [==============================] - 16s 38ms/step - loss: 0.2782 -
accuracy: 0.9143
Epoch 45/50
438/438 [==============================] - 17s 38ms/step - loss: 0.2710 -
accuracy: 0.9189
Epoch 46/50
438/438 [==============================] - 17s 38ms/step - loss: 0.3167 -
accuracy: 0.9019
Epoch 47/50
438/438 [==============================] - 17s 38ms/step - loss: 0.2383 -
accuracy: 0.9282
Epoch 48/50
438/438 [==============================] - 16s 38ms/step - loss: 0.2565 -
accuracy: 0.9228
Epoch 49/50
438/438 [==============================] - 16s 37ms/step - loss: 0.2467 -
accuracy: 0.9299
Epoch 50/50
438/438 [==============================] - 16s 37ms/step - loss: 0.2256 -
accuracy: 0.9324
```

[11]:
```
<tensorflow.python.keras.callbacks.History at 0x7f9f6024cad0>
```

[12]:
```
%tensorboard --logdir logs/fit
```

```
<IPython.core.display.HTML object>
```

[13]:
```
# It is the test generator as similar as the above.
test_datagen = ImageDataGenerator(rescale=1.0/255)

test_generator = test_datagen.flow_from_directory(test_dir,␣
 ↪target_size=(image_width,image_height),
                                                  class_mode='categorical')

test_num = test_generator.samples
```

```
Found 3000 images belonging to 6 classes.
```

[14]:

```
# Evalute the trained model and return both the loss and the test accuracy.
evals = model.evaluate(test_generator,
                       verbose=1,
                       batch_size=Batch_Size,
                       steps=test_num//Batch_Size)

print("Loss = " + str(evals[0]))
print("Test Accuracy = " + str(evals[1]))
```

```
3000/3000 [==============================] - 87s 29ms/step - loss: 0.9312 -
accuracy: 0.7851
Loss = 0.9311785697937012
Test Accuracy = 0.7850593328475952
```

[15]:

```
# Give the implicit steps=7301 for selecting the specific image number.
predict_datagen = ImageDataGenerator(rescale=1.0/255)

predict_generator = predict_datagen.flow_from_directory(predict_dir,
                                                       ␣
 ↪target_size=(image_width,image_height),
                                                        batch_size=Batch_Size,
                                                       ␣
 ↪class_mode='categorical')

predict_num = predict_generator.samples
```

```
Found 7301 images belonging to 1 classes.
```

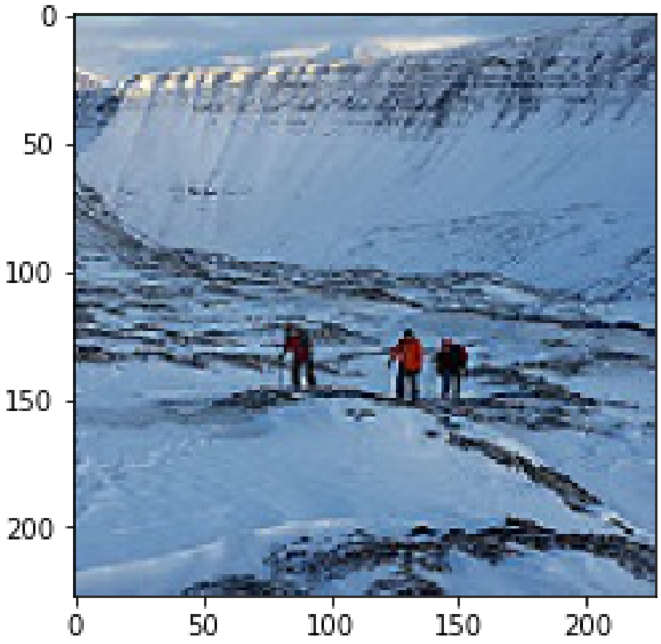
[16]:

```
# Make the prediction for any one of the predicted images
predictions = model.predict(predict_generator,
                            verbose=1,
                            batch_size=Batch_Size,
                            steps=predict_num//Batch_Size)
```

```
7301/7301 [==============================] - 23s 3ms/step
```

[17]:

```
# Plot the discriptive diagram
imshow(predict_generator[256][0][0])
plt.imsave("predicted1.png",predict_generator[256][0][0])
```



[18]:
```
predictions[256]
```

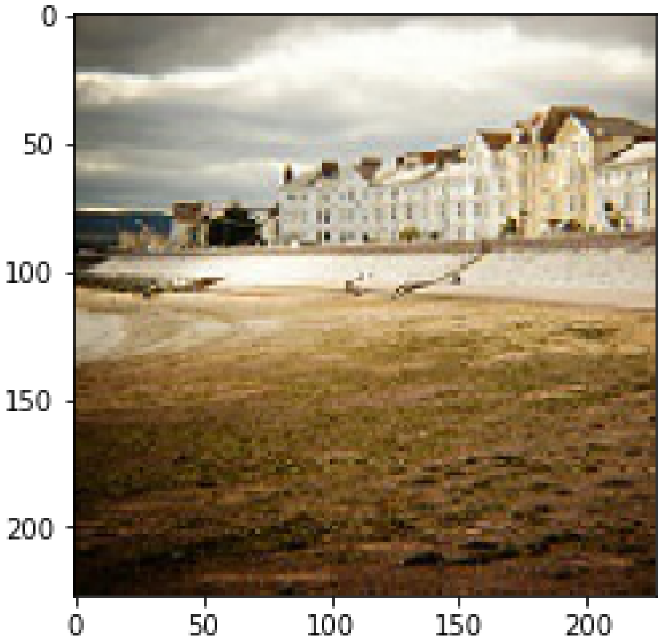
[18]:
```
array([9.9994230e-01, 4.7316188e-15, 1.1871999e-12, 8.8631305e-14,
       1.2097377e-13, 5.7670841e-05], dtype=float32)
```

[19]:
```
print(predictions[256])
```

```
[9.9994230e-01 4.7316188e-15 1.1871999e-12 8.8631305e-14 1.2097377e-13
 5.7670841e-05]
```

[20]:
```
imshow(predict_generator[1024][0][0])
```

[20]:
```
<matplotlib.image.AxesImage at 0x7f9f100e5750>
```



```
[21]: predictions[1024]
```

```
[21]: array([3.5577407e-01, 7.1741693e-02, 3.4085230e-04, 1.7436147e-04,
             3.0319974e-05, 5.7193863e-01], dtype=float32)
```

```
[22]: import os

      def get_category(predicted_output):
          return os.listdir(train_dir)[np.argmax(predicted_output)]
```

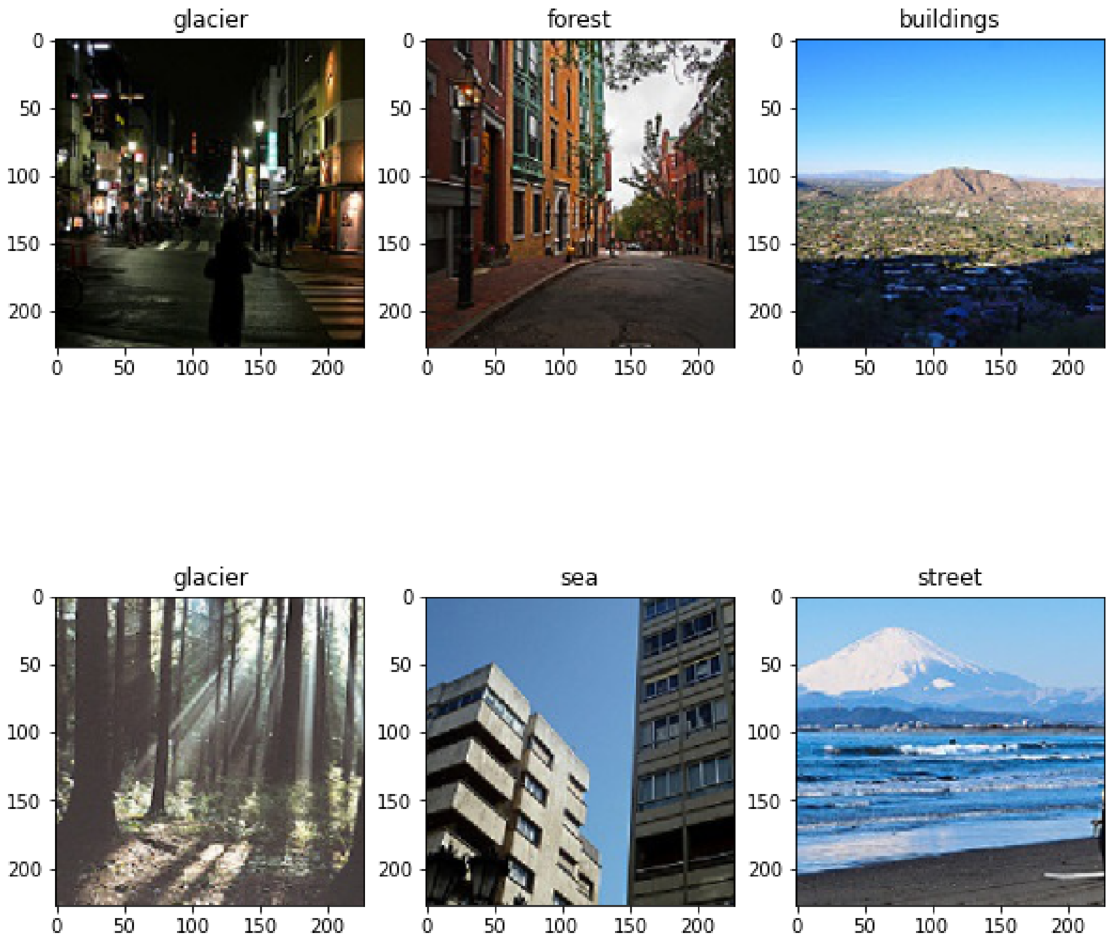
```
[23]: print(get_category(predictions[2048]))
```

```
forest
```

```
[24]: fig, axs = plt.subplots(2, 3, figsize=(10,10))

      axs[0][0].imshow(predict_generator[32][0][0])
      axs[0][0].set_title(get_category(predictions[32]))

      axs[0][1].imshow(predict_generator[64][0][0])
      axs[0][1].set_title(get_category(predictions[64]))

      axs[0][2].imshow(predict_generator[128][0][0])
      axs[0][2].set_title(get_category(predictions[128]))
```

```
axs[1][0].imshow(predict_generator[512][0][0])
axs[1][0].set_title(get_category(predictions[512]))

axs[1][1].imshow(predict_generator[1000][0][0])
axs[1][1].set_title(get_category(predictions[1000]))

axs[1][2].imshow(predict_generator[2000][0][0])
axs[1][2].set_title(get_category(predictions[2000]))
```

```
[24]: Text(0.5, 1.0, 'street')
```



```
[ ]:
```