CatsVdogS

September 3, 2021

1 Cat vs Dogs classifier

1.1 Importing required modules

[1]:

```
from __future__ import absolute_import, division, print_function

import os
import matplotlib.pyplot as plt
import numpy as np

import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator

from jupyterthemes import jtplot
jtplot.style(theme='onedork', figsize=(16,9))

tf.config.list_physical_devices('GPU')
```

[1]:

```
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

1.2 Importing Data

[2]:

```
_URL = 'https://storage.googleapis.com/mledu-datasets/cats_and_dogs_filtered.
 ↪zip'
zip_dir = tf.keras.utils.get_file('cats_and_dogs_filtered.zip', origin=_URL,␣
 ↪extract=True)
```

[3]:

```
zip_dir_base = os.path.dirname(zip_dir)
print(zip_dir_base)
!tree $zip_dir_base
```

```
C:\Users\Arunabh\.keras\datasets
Folder PATH listing for volume Windows
Volume serial number is 00000024 929F:B39D
C:\USERS\ARUNABH\.KERAS\DATASETS
  cats_and_dogs_filtered
      train
          cats
```

```
         dogs
      validation
          cats
          dogs
```

[4]:

```
base_dir = os.path.join(os.path.dirname(zip_dir), 'cats_and_dogs_filtered')
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')

train_cats_dir = os.path.join(train_dir, 'cats')
train_dogs_dir = os.path.join(train_dir, 'dogs')
validation_cats_dir = os.path.join(validation_dir, 'cats')
validation_dogs_dir = os.path.join(validation_dir, 'dogs')
```

1.3 Analysing the data

[5]:

```
num_cats_tr = len(os.listdir(train_cats_dir))
num_dogs_tr = len(os.listdir(train_dogs_dir))

num_cats_val = len(os.listdir(validation_cats_dir))
num_dogs_val = len(os.listdir(validation_dogs_dir))

total_train = num_cats_tr + num_dogs_tr
total_val = num_cats_val + num_dogs_val

print('total training cat images    :', num_cats_tr)
print('total training dog images    :', num_dogs_tr)

print('total validation cat images  :', num_cats_val)
print('total validation dog images  :', num_dogs_val)
print('--')
print('total training images        :', total_train)
print('total validation images      :', total_val)
```

```
total training cat images    : 1000
total training dog images    : 1000
total validation cat images  : 500
total validation dog images  : 500
--
total training images        : 2000
total validation images      : 1000
```

1.4 Data preparation

[6]:

```
BATCH_SIZE = 100
IMG_SHAPE = 150
```

[7]:

```
train_image_generator = ImageDataGenerator(rescale=1./255)
validation_image_generator = ImageDataGenerator(rescale=1./255)
```

[8]:

```
train_data_gen = train_image_generator.
 ↪flow_from_directory(batch_size=BATCH_SIZE,
                                                            directory=train_dir,
                                                            shuffle=True,
                                                           ␣
 ↪target_size=(IMG_SHAPE,IMG_SHAPE),
                                                            class_mode='binary')

val_data_gen = validation_image_generator.
 ↪flow_from_directory(batch_size=BATCH_SIZE,
                                                           ␣
 ↪directory=validation_dir,
                                                            shuffle=False,
                                                           ␣
 ↪target_size=(IMG_SHAPE,IMG_SHAPE),
                                                            class_mode='binary')
```

```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
```

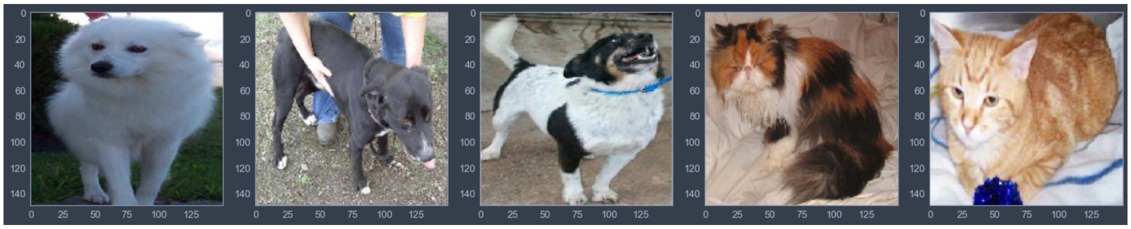
1.4.1 Visualizing the data

[22]:

```
def plotImages(images_arr):

    """Plot images in the form of a 1x5 grid"""

    fig, axes = plt.subplots(1, 5, figsize=(20,20))
    axes = axes.flatten()
    for img, ax in zip(images_arr, axes):
        ax.imshow(img)
        ax.grid(False)
    plt.tight_layout()
    plt.show()
```

[23]:

```
sample_training_images, _ = next(train_data_gen)
plotImages(sample_training_images[:5])
```



1.5 Building the model

[9]:
```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150,␣
 ↪150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(128, (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(2, activation='softmax')
])
```

[10]:
```
model.compile(optimizer='adam',
            loss='sparse_categorical_crossentropy',
            metrics=['accuracy'])
```

[11]:
```
model.summary()
```

```
Model: "sequential"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 32)      896

_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 32)        0

_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 64)        18496

_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 64)        0

_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 128)       73856

_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 128)       0

_________________________________________________________________
conv2d_3 (Conv2D)            (None, 15, 15, 128)       147584

_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 6272)              0
_________________________________________________________________
dense (Dense)                (None, 512)               3211776
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 1026
=================================================================
Total params: 3,453,634
Trainable params: 3,453,634
Non-trainable params: 0
_________________________________________________________________
```

[14]:

```
EPOCHS = 20
history = model.fit_generator(
    train_data_gen,
    steps_per_epoch=int(np.ceil(total_train / float(BATCH_SIZE))),
    epochs=EPOCHS,
    validation_data=val_data_gen,
    validation_steps=int(np.ceil(total_val / float(BATCH_SIZE)))
)
```

```
C:\Users\Arunabh\anaconda3\envs\tf-accel\lib\site-
packages\tensorflow\python\keras\engine\training.py:1940: UserWarning:
`Model.fit_generator` is deprecated and will be removed in a future version.
Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
```

```
Epoch 1/20
20/20 [==============================] - 61s 2s/step - loss: 0.7212 - accuracy:
0.4985 - val_loss: 0.6918 - val_accuracy: 0.5130
Epoch 2/20
20/20 [==============================] - 15s 744ms/step - loss: 0.6941 -
accuracy: 0.5145 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 3/20
20/20 [==============================] - 15s 740ms/step - loss: 0.6919 -
accuracy: 0.5190 - val_loss: 0.6847 - val_accuracy: 0.5960
Epoch 4/20
20/20 [==============================] - 15s 733ms/step - loss: 0.6934 -
accuracy: 0.5485 - val_loss: 0.6903 - val_accuracy: 0.5380
Epoch 5/20
20/20 [==============================] - 14s 712ms/step - loss: 0.6829 -
accuracy: 0.5910 - val_loss: 0.6545 - val_accuracy: 0.6270
Epoch 6/20
20/20 [==============================] - 14s 714ms/step - loss: 0.6801 -
accuracy: 0.5615 - val_loss: 0.6864 - val_accuracy: 0.5070
Epoch 7/20
20/20 [==============================] - 14s 717ms/step - loss: 0.6724 -
accuracy: 0.5580 - val_loss: 0.6538 - val_accuracy: 0.6220
Epoch 8/20
```

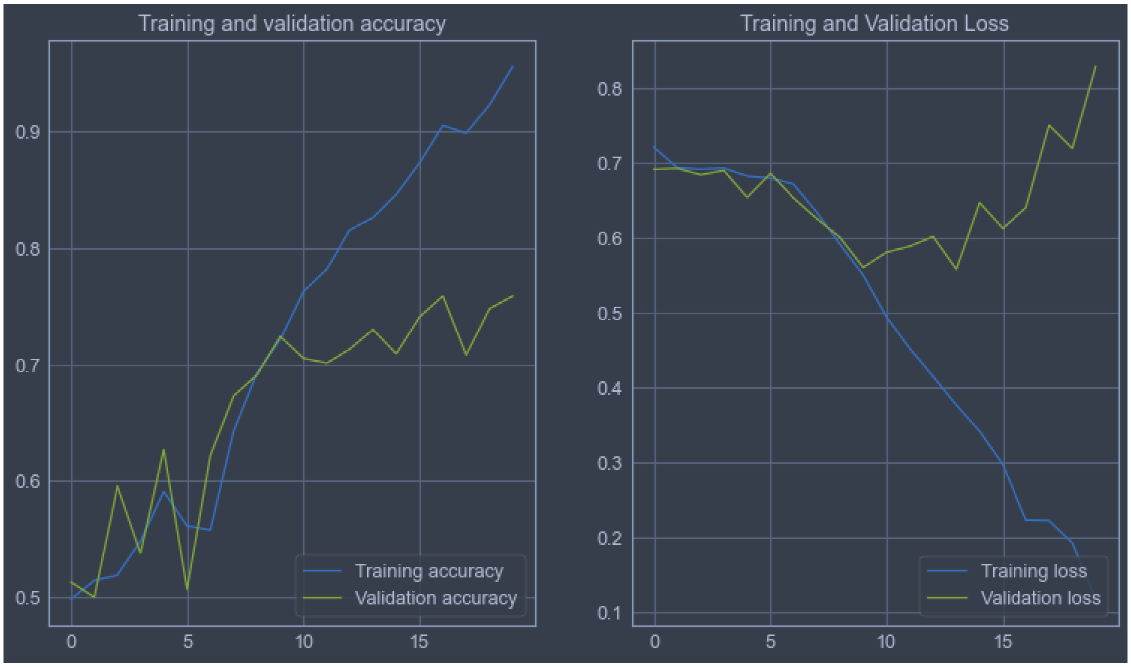
```
20/20 [==============================] - 14s 708ms/step - loss: 0.6349 -
accuracy: 0.6430 - val_loss: 0.6263 - val_accuracy: 0.6730
Epoch 9/20
20/20 [==============================] - 14s 711ms/step - loss: 0.5922 -
accuracy: 0.6925 - val_loss: 0.6013 - val_accuracy: 0.6910
Epoch 10/20
20/20 [==============================] - 14s 716ms/step - loss: 0.5509 -
accuracy: 0.7215 - val_loss: 0.5610 - val_accuracy: 0.7240
Epoch 11/20
20/20 [==============================] - 14s 706ms/step - loss: 0.4950 -
accuracy: 0.7630 - val_loss: 0.5812 - val_accuracy: 0.7050
Epoch 12/20
20/20 [==============================] - 14s 708ms/step - loss: 0.4522 -
accuracy: 0.7820 - val_loss: 0.5892 - val_accuracy: 0.7010
Epoch 13/20
20/20 [==============================] - 14s 710ms/step - loss: 0.4151 -
accuracy: 0.8155 - val_loss: 0.6023 - val_accuracy: 0.7130
Epoch 14/20
20/20 [==============================] - 14s 707ms/step - loss: 0.3766 -
accuracy: 0.8260 - val_loss: 0.5584 - val_accuracy: 0.7300
Epoch 15/20
20/20 [==============================] - 14s 709ms/step - loss: 0.3419 -
accuracy: 0.8460 - val_loss: 0.6473 - val_accuracy: 0.7090
Epoch 16/20
20/20 [==============================] - 14s 707ms/step - loss: 0.2973 -
accuracy: 0.8730 - val_loss: 0.6131 - val_accuracy: 0.7410
Epoch 17/20
20/20 [==============================] - 14s 706ms/step - loss: 0.2234 -
accuracy: 0.9050 - val_loss: 0.6407 - val_accuracy: 0.7590
Epoch 18/20
20/20 [==============================] - 14s 714ms/step - loss: 0.2225 -
accuracy: 0.8985 - val_loss: 0.7508 - val_accuracy: 0.7080
Epoch 19/20
20/20 [==============================] - 14s 706ms/step - loss: 0.1928 -
accuracy: 0.9230 - val_loss: 0.7199 - val_accuracy: 0.7480
Epoch 20/20
20/20 [==============================] - 14s 708ms/step - loss: 0.1172 -
accuracy: 0.9560 - val_loss: 0.8298 - val_accuracy: 0.7590
```

2 Understanding results

[16]:

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']
```

```
epochs_range = range(EPOCHS)

plt.figure()

plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training accuracy')
plt.plot(epochs_range, val_acc, label='Validation accuracy')
plt.legend(loc='lower right')
plt.title('Training and validation accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training loss')
plt.plot(epochs_range, val_loss, label='Validation loss')
plt.legend(loc='lower right')
plt.title('Training and Validation Loss')
plt.savefig('./loss-accuracy.png')
```


Training and validation accuracy
Training accuracy
Validation accuracy
Training and Validation Loss
Training loss
Validation loss