Assignment No. 3

November 14, 2022

[1]:
```
import tensorflow as tf
```

[2]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

[3]:
```
import os
import matplotlib.pyplot as plt
import numpy as np
```

[4]:
```
_URL="https://storage.googleapis.com/mledu-datasets/cats_and_dogs_filtered.zip"
zip_dir = tf.keras.utils.get_file('cats_and_dogs_filterted.zip', origin=_URL,␣
 ↪extract=True)
```

[5]:
```
zip_dir_base = os.path.dirname(zip_dir)
!find $zip_dir_base -type d -print
```

```
FIND: Parameter format not correct
```

[6]:
```
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')
```

[7]:
```
total_size = len(os.listdir(train_cats_dir)) + len(os.listdir(train_dogs_dir))
total_val = len(os.listdir(validation_cats_dir)) + len(os.
 ↪listdir(validation_dogs_dir))
```

[8]:
```
print(len(os.listdir(train_cats_dir)))
print(len(os.listdir(train_dogs_dir)))
```

```
1000
1000
```

[9]:
```
# validation
print(len(os.listdir(validation_cats_dir)))
print(len(os.listdir(validation_dogs_dir)))
```

```
500
500
```

Setting Model Parameters

```
[10]: BATCH_SIZE = 100
      IMAGE_SIZE = 150
```

The loading, decoding of the image to RGB, and into proper grid format, converting them into floating point tensors, and resacling the values from 0 to 255 to 0 and 1 are done by the ImageDataGenerator

```
[11]: train_image_generator = ImageDataGenerator(rescale=1./255)
      validation_image_generator = ImageDataGenerator(rescale=1./255)
```

After defining our generators for training and validation images, flow_from_directory method will load images from the disk, apply rescaling, and resize them using single line of code.

```
[12]: train_data_gen = train_image_generator.
       ↪flow_from_directory(batch_size=BATCH_SIZE,
                                                                                    ␣
       ↪directory=train_dir,
                                                                                     shuffle=True,
                                                                                    ␣
       ↪target_size=(IMAGE_SIZE,IMAGE_SIZE),
                                                                                    ␣
       ↪class_mode='binary')
```

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

```
[13]: test_data_gen = validation_image_generator.
       ↪flow_from_directory(batch_size=BATCH_SIZE,
                                                                                   ␣
       ↪directory=validation_dir,
                                                                                    shuffle=False,
                                                                                   ␣
       ↪target_size=(IMAGE_SIZE, IMAGE_SIZE),
                                                                                   ␣
       ↪class_mode='binary')
```

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

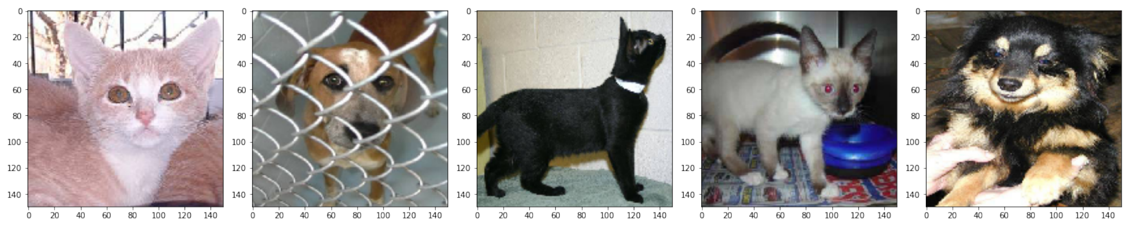
Visualizing the images

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

```
[15]: def plotImages(images_arr):
          fig, axes = plt.subplots(1,5, figsize=(20,20))
          axes = axes.flatten()
          for img, ax in zip(images_arr, axes):
```

```
        ax.imshow(img)
    plt.tight_layout()
    plt.show()
```

[16]:
```
plotImages(sample_training_images[:5])
```



Defining the model

[17]:
```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu',␣
 ↪input_shape=(150,150,3)),
    tf.keras.layers.MaxPool2D(2,2),

    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPool2D(2,2),

    tf.keras.layers.Conv2D(128,(3,3), activation='relu'),
    tf.keras.layers.MaxPool2D(2,2),

    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPool2D(2,2),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(2)
])
```

Compiling the Model

[18]:
```
model.compile(optimizer='adam',
             loss=tf.keras.losses.
 ↪SparseCategoricalCrossentropy(from_logits=True),
             metrics=['accuracy'])
```

Summary

[19]:
```
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 (MaxPooling  (None, 36, 36, 64)       0
 2D)

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

 max_pooling2d_2 (MaxPooling  (None, 17, 17, 128)      0
 2D)

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

 max_pooling2d_3 (MaxPooling  (None, 7, 7, 128)        0
 2D)

 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
_________________________________________________________________
```

Train the Model

[20]:
```
epochs = 10
history = model.fit_generator(
    train_data_gen,
    steps_per_epoch=int(np.ceil(total_size/float(BATCH_SIZE))),
    epochs=epochs,
    validation_data=test_data_gen,
    validation_steps=int(np.ceil(total_val/float(BATCH_SIZE)))
)
```

```
C:\Users\Akhil\AppData\Local\Temp/ipykernel_5900/188943310.py:2: UserWarning:
`Model.fit_generator` is deprecated and will be removed in a future version.
Please use `Model.fit`, which supports generators.
  history = model.fit_generator(
Epoch 1/10
20/20 [==============================] - 30s 497ms/step - loss: 0.7216 -
accuracy: 0.5085 - val_loss: 0.6894 - val_accuracy: 0.5050
Epoch 2/10
20/20 [==============================] - 8s 409ms/step - loss: 0.6796 -
accuracy: 0.5560 - val_loss: 0.6426 - val_accuracy: 0.6390
Epoch 3/10
20/20 [==============================] - 8s 387ms/step - loss: 0.6410 -
accuracy: 0.6300 - val_loss: 0.6205 - val_accuracy: 0.6580
Epoch 4/10
20/20 [==============================] - 8s 391ms/step - loss: 0.6358 -
accuracy: 0.6230 - val_loss: 0.6102 - val_accuracy: 0.6530
Epoch 5/10
20/20 [==============================] - 8s 389ms/step - loss: 0.5746 -
accuracy: 0.6980 - val_loss: 0.5978 - val_accuracy: 0.6790
Epoch 6/10
20/20 [==============================] - 8s 390ms/step - loss: 0.5483 -
accuracy: 0.7210 - val_loss: 0.5599 - val_accuracy: 0.7090
Epoch 7/10
20/20 [==============================] - 8s 395ms/step - loss: 0.5138 -
accuracy: 0.7485 - val_loss: 0.5888 - val_accuracy: 0.6750
Epoch 8/10
20/20 [==============================] - 8s 389ms/step - loss: 0.4744 -
accuracy: 0.7750 - val_loss: 0.5557 - val_accuracy: 0.7380
Epoch 9/10
20/20 [==============================] - 8s 405ms/step - loss: 0.4464 -
accuracy: 0.7910 - val_loss: 0.5635 - val_accuracy: 0.7050
Epoch 10/10
20/20 [==============================] - 8s 387ms/step - loss: 0.3910 -
accuracy: 0.8230 - val_loss: 0.5575 - val_accuracy: 0.7410
```

Visualizing results of the training

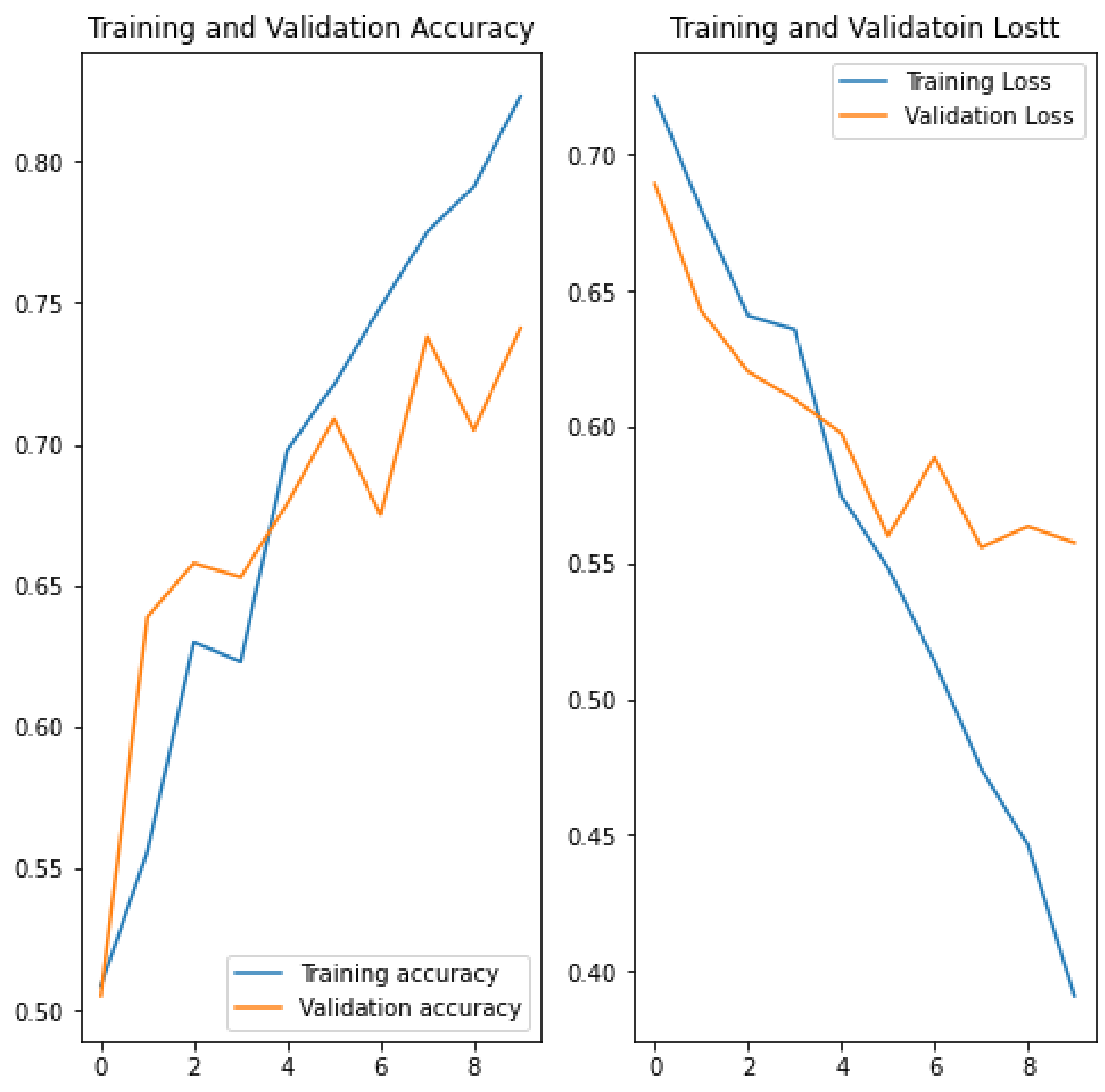
[21]:
```python
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

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

epochs_range = range(epochs)

plt.figure(figsize=(8,8))
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="upper right")
plt.title("Training and Validatoin Lostt")

plt.show()
```


Training and Validation Accuracy
Training accuracy
Validation accuracy
Training and Validatoin Lostt
Training Loss
Validation Loss