Assignment 6

November 11, 2021

0.1 Assignment 6

Nick Reardon

0.1.1 6.1

[35]:
```
# Import libraries

from keras import layers
from keras import models
from keras.datasets import mnist
from keras.utils import to_categorical
from pathlib import Path
import os
import matplotlib.pyplot as plt
import numpy as np
from keras import losses
from keras import metrics
```

[36]:
```
# 5.1

model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

[37]:
```
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 11, 11, 64)        18496
```

```
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 3, 3, 64)          36928
=================================================================
Total params: 55,744
Trainable params: 55,744
Non-trainable params: 0
_________________________________________________________________
```

[38]:
```
# 5.2

model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

[39]:
```
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 3, 3, 64)          36928
_________________________________________________________________
flatten_3 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_6 (Dense)              (None, 64)                36928
_________________________________________________________________
dense_7 (Dense)              (None, 10)                650
=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
_________________________________________________________________
```

[40]:
```
# 5.3

(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
train_images = train_images.reshape((60000, 28, 28, 1))
```

```
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape((10000, 28, 28, 1))
test_images = test_images.astype('float32') / 255

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)

model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
history = model.fit(train_images, train_labels, epochs=5, batch_size=64,␣
 ↪validation_split=0.1)
```

```
Epoch 1/5
844/844 [==============================] - 12s 13ms/step - loss: 0.4310 -
accuracy: 0.8617 - val_loss: 0.0472 - val_accuracy: 0.9867
Epoch 2/5
844/844 [==============================] - 11s 13ms/step - loss: 0.0536 -
accuracy: 0.9832 - val_loss: 0.0392 - val_accuracy: 0.9887
Epoch 3/5
844/844 [==============================] - 11s 13ms/step - loss: 0.0360 -
accuracy: 0.9888 - val_loss: 0.0442 - val_accuracy: 0.9875
Epoch 4/5
844/844 [==============================] - 11s 13ms/step - loss: 0.0270 -
accuracy: 0.9919 - val_loss: 0.0340 - val_accuracy: 0.9895
Epoch 5/5
844/844 [==============================] - 11s 13ms/step - loss: 0.0190 -
accuracy: 0.9935 - val_loss: 0.0396 - val_accuracy: 0.9913
```

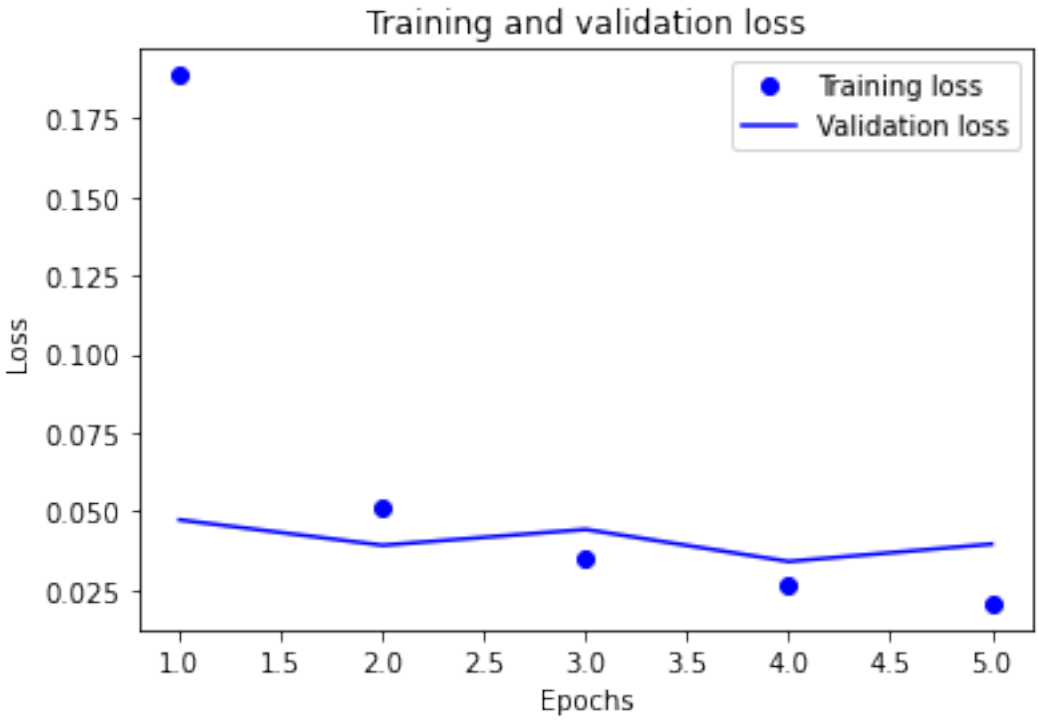
[41]:
```
train_loss, train_acc = model.evaluate(train_images, train_labels)
print('Test Accuracy:',train_acc)
print('Test Loss:', train_loss)
```

```
1875/1875 [==============================] - 9s 4ms/step - loss: 0.0159 -
accuracy: 0.9952
Test Accuracy: 0.9952499866485596
Test Loss: 0.015877896919846535
```

[42]:
```
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Test Accuracy:',test_acc)
print('Test Loss:', test_loss)
```

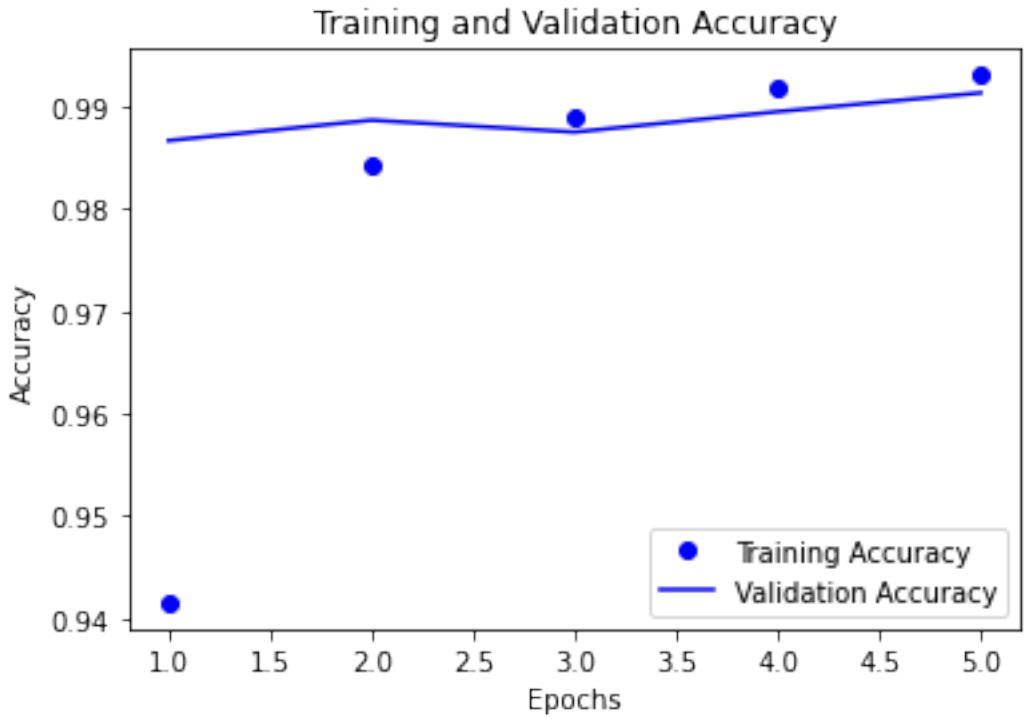
```
313/313 [==============================] - 1s 4ms/step - loss: 0.0304 -
accuracy: 0.9906
Test Accuracy: 0.9905999898910522
Test Loss: 0.030374450609087944
```

[43]:
```
history_dict = history.history
acc_values = history_dict['accuracy']
val_acc_values = history_dict['val_accuracy']
val_loss = history_dict['loss']
val_loss_values = history_dict['val_loss']

epochs = range(1, len(acc_values) + 1)

plt.plot(epochs, val_loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss_values, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```



[44]:
```
plt.plot(epochs, acc_values, 'bo', label='Training Accuracy')
plt.plot(epochs, val_acc_values, 'b', label = 'Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
```

```
plt.show()
```



0.2 6.2

0.2.1 6.2.a

[45]:
```
from keras import layers
from keras import models
from keras import optimizers
from keras.datasets import cifar10
from matplotlib import pyplot
from keras.utils import to_categorical
from pathlib import Path
import os
import matplotlib.pyplot as plt
import numpy as np
from keras import losses
from keras import metrics
```

[46]:
```
model2 = models.Sequential()
model2.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32,␣
↪3)))
```

```
model2.add(layers.MaxPooling2D((2, 2)))
model2.add(layers.Conv2D(64, (3, 3), activation='relu'))
model2.add(layers.MaxPooling2D((2, 2)))
model2.add(layers.Conv2D(128, (3, 3), activation='relu'))
```

[47]:
```
model2.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 30, 30, 32)        896
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 4, 4, 128)         73856
=================================================================
Total params: 93,248
Trainable params: 93,248
Non-trainable params: 0
_________________________________________________________________
```

[48]:
```
model2.add(layers.Flatten())
model2.add(layers.Dense(512, activation='relu'))
model2.add(layers.Dense(10, activation='softmax'))
```

[49]:
```
model2.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 30, 30, 32)        896
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 4, 4, 128)         73856
_________________________________________________________________
flatten_4 (Flatten)          (None, 2048)              0
```

```
_________________________________________________________________
dense_8 (Dense)              (None, 512)               1049088
_________________________________________________________________
dense_9 (Dense)              (None, 10)                5130
=================================================================
Total params: 1,147,466
Trainable params: 1,147,466
Non-trainable params: 0
_________________________________________________________________
```

[50]:
```
(train_images2, train_labels2), (test_images2, test_labels2) = cifar10.
↪load_data()

train_images2 = train_images2.reshape((50000, 32, 32, 3))
test_images2 = test_images2.reshape((10000, 32, 32, 3))

train_labels2 = to_categorical(train_labels2)
test_labels2 = to_categorical(test_labels2)
```

[51]:
```
# 5.6

model2.compile(optimizer='rmsprop',
               loss='categorical_crossentropy',
               metrics=['accuracy'])
history2 = model2.fit(train_images2, train_labels2, epochs=20, batch_size=64,␣
↪validation_data=(test_images2, test_labels2))
```

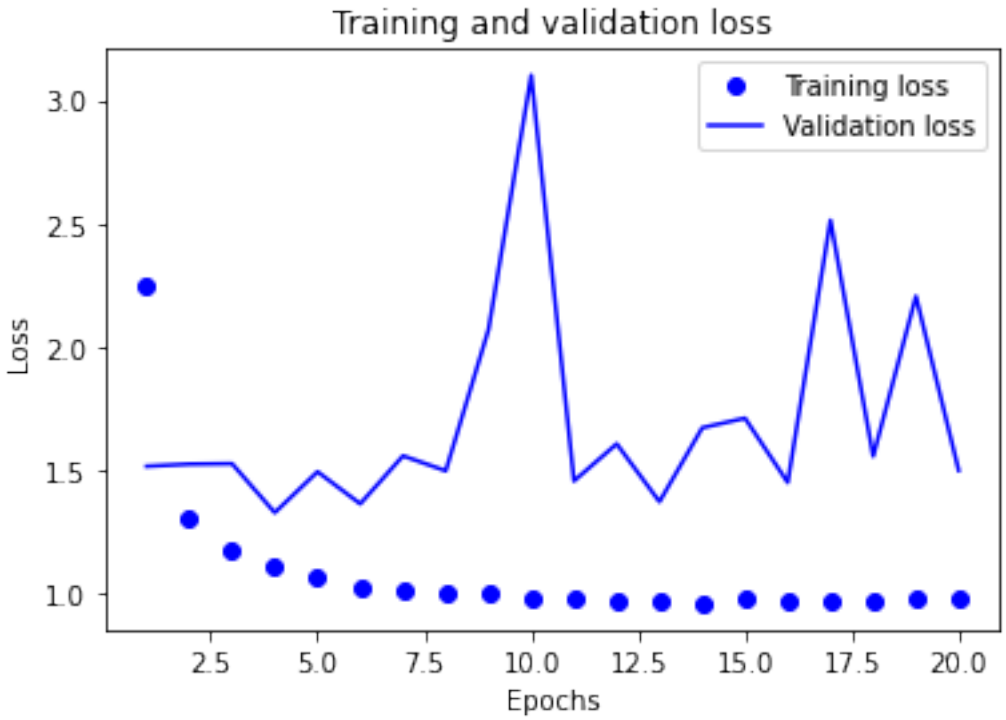
```
Epoch 1/20
782/782 [==============================] - 15s 18ms/step - loss: 4.6486 -
accuracy: 0.2602 - val_loss: 1.5164 - val_accuracy: 0.4936
Epoch 2/20
782/782 [==============================] - 14s 17ms/step - loss: 1.3386 -
accuracy: 0.5394 - val_loss: 1.5250 - val_accuracy: 0.4803
Epoch 3/20
782/782 [==============================] - 14s 18ms/step - loss: 1.1760 -
accuracy: 0.6056 - val_loss: 1.5269 - val_accuracy: 0.5511
Epoch 4/20
782/782 [==============================] - 14s 18ms/step - loss: 1.0932 -
accuracy: 0.6346 - val_loss: 1.3283 - val_accuracy: 0.5431
Epoch 5/20
782/782 [==============================] - 14s 18ms/step - loss: 1.0515 -
accuracy: 0.6537 - val_loss: 1.4941 - val_accuracy: 0.5598
Epoch 6/20
782/782 [==============================] - 13s 17ms/step - loss: 1.0046 -
accuracy: 0.6672 - val_loss: 1.3641 - val_accuracy: 0.5289
Epoch 7/20
782/782 [==============================] - 14s 18ms/step - loss: 0.9975 -
accuracy: 0.6722 - val_loss: 1.5585 - val_accuracy: 0.5759
```

```
Epoch 8/20
782/782 [==============================] - 14s 17ms/step - loss: 0.9793 -
accuracy: 0.6756 - val_loss: 1.4979 - val_accuracy: 0.5467
Epoch 9/20
782/782 [==============================] - 14s 18ms/step - loss: 0.9722 -
accuracy: 0.6876 - val_loss: 2.0705 - val_accuracy: 0.5072
Epoch 10/20
782/782 [==============================] - 15s 19ms/step - loss: 0.9759 -
accuracy: 0.6842 - val_loss: 3.1018 - val_accuracy: 0.4191
Epoch 11/20
782/782 [==============================] - 14s 19ms/step - loss: 0.9734 -
accuracy: 0.6877 - val_loss: 1.4568 - val_accuracy: 0.5247
Epoch 12/20
782/782 [==============================] - 15s 19ms/step - loss: 0.9555 -
accuracy: 0.6935 - val_loss: 1.6071 - val_accuracy: 0.5782
Epoch 13/20
782/782 [==============================] - 14s 18ms/step - loss: 0.9536 -
accuracy: 0.6953 - val_loss: 1.3725 - val_accuracy: 0.6028
Epoch 14/20
782/782 [==============================] - 14s 18ms/step - loss: 0.9395 -
accuracy: 0.6990 - val_loss: 1.6724 - val_accuracy: 0.5883
Epoch 15/20
782/782 [==============================] - 14s 19ms/step - loss: 0.9676 -
accuracy: 0.6914 - val_loss: 1.7113 - val_accuracy: 0.5964
Epoch 16/20
782/782 [==============================] - 15s 19ms/step - loss: 0.9465 -
accuracy: 0.7005 - val_loss: 1.4506 - val_accuracy: 0.5782
Epoch 17/20
782/782 [==============================] - 14s 18ms/step - loss: 0.9461 -
accuracy: 0.7006 - val_loss: 2.5139 - val_accuracy: 0.4644
Epoch 18/20
782/782 [==============================] - 15s 19ms/step - loss: 0.9695 -
accuracy: 0.6888 - val_loss: 1.5570 - val_accuracy: 0.6305
Epoch 19/20
782/782 [==============================] - 15s 19ms/step - loss: 0.9790 -
accuracy: 0.6903 - val_loss: 2.2056 - val_accuracy: 0.6279
Epoch 20/20
782/782 [==============================] - 15s 19ms/step - loss: 0.9641 -
accuracy: 0.6975 - val_loss: 1.4994 - val_accuracy: 0.5927
```

[52]:
```
test_loss2, test_acc2 = model2.evaluate(test_images2, test_labels2)
print('Test Accuracy:',test_acc2)
print('Test Loss:', test_loss2)
```

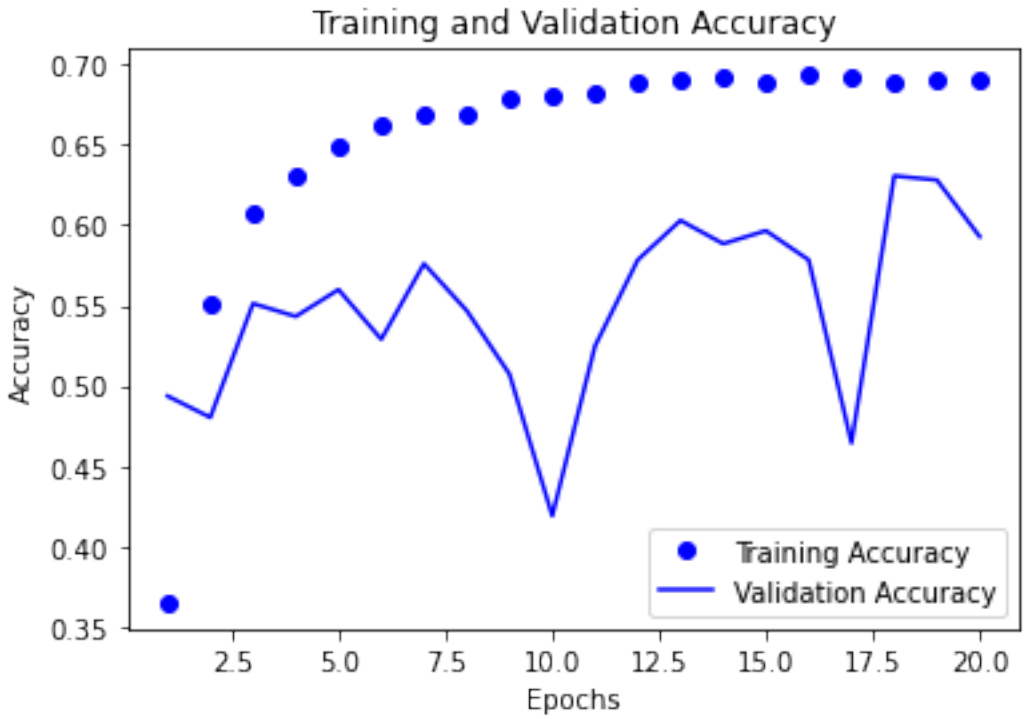
```
313/313 [==============================] - 2s 5ms/step - loss: 1.4994 -
accuracy: 0.5927
Test Accuracy: 0.5927000045776367
Test Loss: 1.4994142055511475
```

[53]:
```
history_dict2 = history2.history
acc_values2 = history_dict2['accuracy']
val_acc_values2 = history_dict2['val_accuracy']
val_loss2 = history_dict2['loss']
val_loss_values2 = history_dict2['val_loss']

epochs2 = range(1, len(val_loss2) + 1)

plt.plot(epochs2, val_loss2, 'bo', label='Training loss')
plt.plot(epochs2, val_loss_values2, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```



[54]:
```
plt.plot(epochs2, acc_values2, 'bo', label='Training Accuracy')
plt.plot(epochs2, val_acc_values2, 'b', label = 'Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
```

```
plt.show()
```



0.2.2 6.2.b

[55]:
```
# Import libraries
from keras import layers
from keras import models
from keras import optimizers
from keras.datasets import cifar10
from matplotlib import pyplot
from keras.utils import to_categorical
from pathlib import Path
import os
import matplotlib.pyplot as plt
import numpy as np
from keras import losses
from keras import metrics
from keras.preprocessing.image import ImageDataGenerator
```

[56]:
```
model3 = models.Sequential()
model3.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32,␣
↪3)))
```

```
model3.add(layers.MaxPooling2D((2, 2)))
model3.add(layers.Dropout(0.2))
model3.add(layers.Conv2D(64, (3, 3), activation='relu'))
model3.add(layers.MaxPooling2D((2, 2)))
model3.add(layers.Dropout(0.2))
model3.add(layers.Conv2D(128, (3, 3), activation='relu'))
```

[57]:
```
model3.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_15 (Conv2D)           (None, 30, 30, 32)        896
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 15, 15, 32)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 6, 6, 64)          0
_________________________________________________________________
dropout_3 (Dropout)          (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 4, 4, 128)         73856
=================================================================
Total params: 93,248
Trainable params: 93,248
Non-trainable params: 0
_________________________________________________________________
```

[58]:
```
model3.add(layers.Flatten())
model3.add(layers.Dense(512, activation='relu'))
model3.add(layers.Dense(10, activation='softmax'))
```

[59]:
```
model3.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_15 (Conv2D)           (None, 30, 30, 32)        896
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 15, 15, 32)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 15, 15, 32)        0
```

```
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 6, 6, 64)          0
_________________________________________________________________
dropout_3 (Dropout)          (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 4, 4, 128)         73856
_________________________________________________________________
flatten_5 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_10 (Dense)             (None, 512)               1049088
_________________________________________________________________
dense_11 (Dense)             (None, 10)                5130
=================================================================
Total params: 1,147,466
Trainable params: 1,147,466
Non-trainable params: 0
_________________________________________________________________
```

[60]:
```
(train_images3, train_labels3), (test_images3, test_labels3) = cifar10.
↪load_data()

train_images3 = train_images3.reshape((50000, 32, 32, 3))
test_images3 = test_images3.reshape((10000, 32, 32, 3))

train_labels3 = to_categorical(train_labels3)
test_labels3 = to_categorical(test_labels3)
```

[61]:
```
model3.compile(optimizer='rmsprop',
               loss='categorical_crossentropy',
               metrics=['accuracy'])

train_datagen = ImageDataGenerator(width_shift_range=0.1, height_shift_range=0.
↪1, horizontal_flip = True)
train_generator = train_datagen.flow(train_images3, train_labels3, batch_size =␣
↪64)
```

[62]:
```
history3 = model3.fit(train_generator, epochs = 20, validation_data =␣
↪(test_images3, test_labels3))
```

```
Epoch 1/20
782/782 [==============================] - 36s 45ms/step - loss: 7.0143 -
accuracy: 0.1753 - val_loss: 1.6006 - val_accuracy: 0.4108
Epoch 2/20
782/782 [==============================] - 35s 45ms/step - loss: 1.6302 -
accuracy: 0.4178 - val_loss: 1.3813 - val_accuracy: 0.5129
Epoch 3/20
```

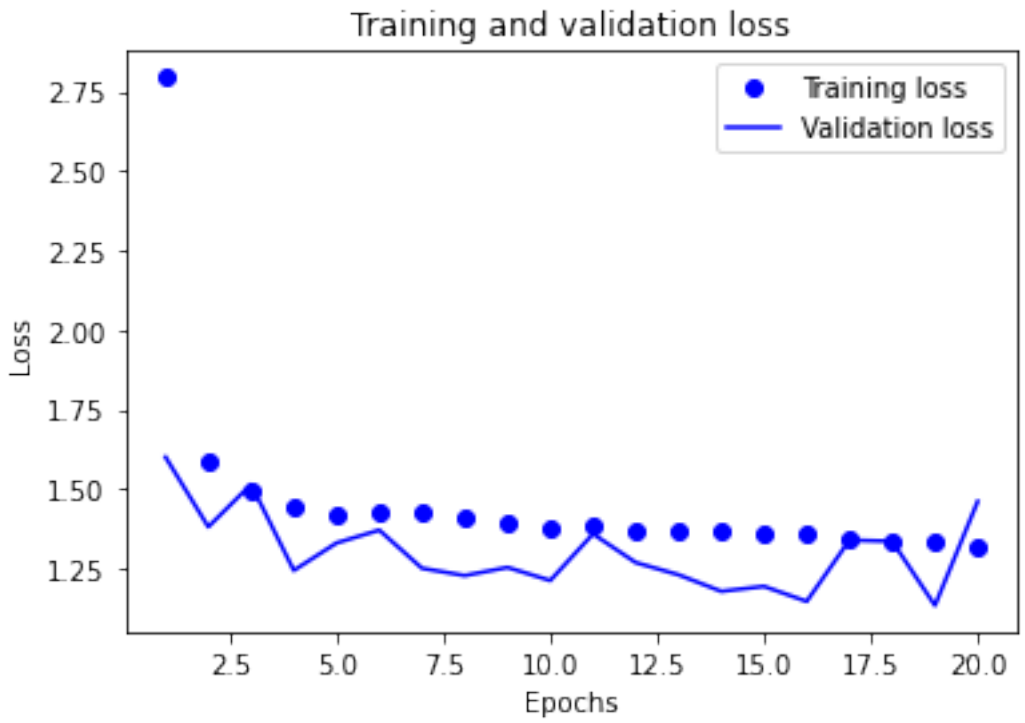
```
782/782 [==============================] - 35s 45ms/step - loss: 1.4982 -
accuracy: 0.4720 - val_loss: 1.5150 - val_accuracy: 0.4575
Epoch 4/20
782/782 [==============================] - 35s 45ms/step - loss: 1.4530 -
accuracy: 0.4928 - val_loss: 1.2444 - val_accuracy: 0.5622
Epoch 5/20
782/782 [==============================] - 35s 45ms/step - loss: 1.4269 -
accuracy: 0.4973 - val_loss: 1.3307 - val_accuracy: 0.5223
Epoch 6/20
782/782 [==============================] - 35s 45ms/step - loss: 1.4219 -
accuracy: 0.5038 - val_loss: 1.3711 - val_accuracy: 0.5296
Epoch 7/20
782/782 [==============================] - 35s 45ms/step - loss: 1.4240 -
accuracy: 0.5055 - val_loss: 1.2511 - val_accuracy: 0.5584
Epoch 8/20
782/782 [==============================] - 35s 45ms/step - loss: 1.4124 -
accuracy: 0.5076 - val_loss: 1.2283 - val_accuracy: 0.5725
Epoch 9/20
782/782 [==============================] - 35s 45ms/step - loss: 1.4001 -
accuracy: 0.5156 - val_loss: 1.2538 - val_accuracy: 0.5602
Epoch 10/20
782/782 [==============================] - 35s 44ms/step - loss: 1.3632 -
accuracy: 0.5258 - val_loss: 1.2127 - val_accuracy: 0.5774
Epoch 11/20
782/782 [==============================] - 36s 46ms/step - loss: 1.3844 -
accuracy: 0.5238 - val_loss: 1.3592 - val_accuracy: 0.5215
Epoch 12/20
782/782 [==============================] - 35s 45ms/step - loss: 1.3546 -
accuracy: 0.5331 - val_loss: 1.2691 - val_accuracy: 0.5760
Epoch 13/20
782/782 [==============================] - 35s 45ms/step - loss: 1.3653 -
accuracy: 0.5280 - val_loss: 1.2308 - val_accuracy: 0.5804
Epoch 14/20
782/782 [==============================] - 35s 45ms/step - loss: 1.3646 -
accuracy: 0.5301 - val_loss: 1.1787 - val_accuracy: 0.5873
Epoch 15/20
782/782 [==============================] - 35s 45ms/step - loss: 1.3533 -
accuracy: 0.5323 - val_loss: 1.1943 - val_accuracy: 0.5826
Epoch 16/20
782/782 [==============================] - 35s 44ms/step - loss: 1.3561 -
accuracy: 0.5302 - val_loss: 1.1471 - val_accuracy: 0.6032
Epoch 17/20
782/782 [==============================] - 35s 45ms/step - loss: 1.3360 -
accuracy: 0.5426 - val_loss: 1.3397 - val_accuracy: 0.5394
Epoch 18/20
782/782 [==============================] - 35s 45ms/step - loss: 1.3419 -
accuracy: 0.5442 - val_loss: 1.3360 - val_accuracy: 0.5361
Epoch 19/20
```

```
782/782 [==============================] - 35s 45ms/step - loss: 1.3351 -
accuracy: 0.5474 - val_loss: 1.1342 - val_accuracy: 0.5984
Epoch 20/20
782/782 [==============================] - 35s 44ms/step - loss: 1.3137 -
accuracy: 0.5539 - val_loss: 1.4613 - val_accuracy: 0.5291
```

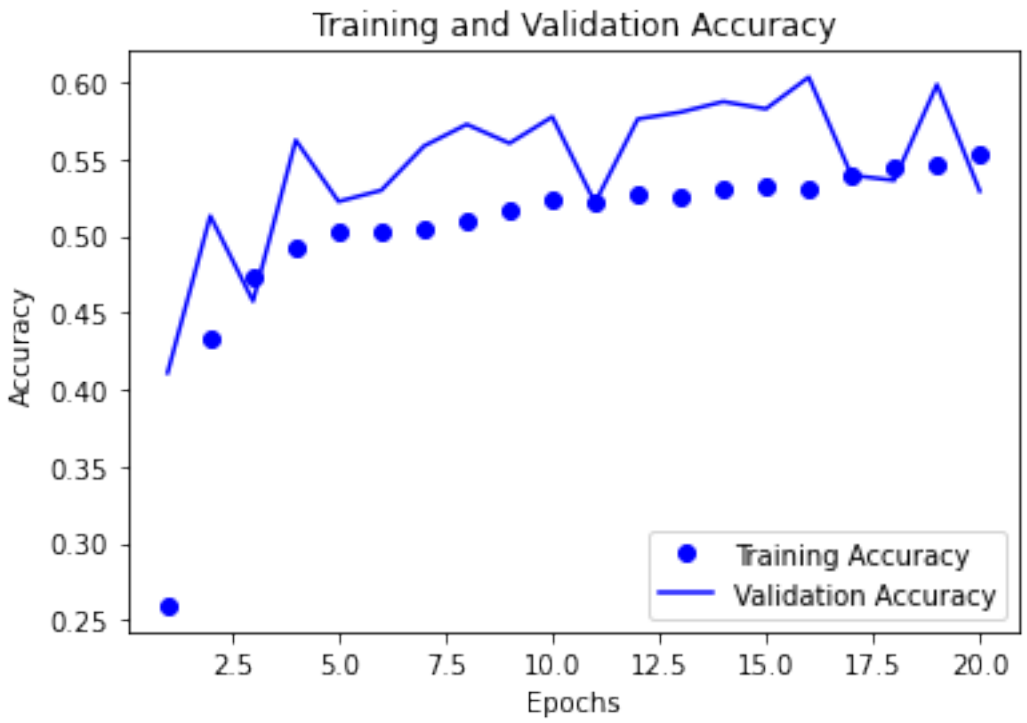
[63]:
```
test_loss3, test_acc3 = model3.evaluate(test_images3, test_labels3)
print('Test Accuracy:',test_acc3)
print('Test Loss:', test_loss3)
```

```
313/313 [==============================] - 2s 5ms/step - loss: 1.4613 -
accuracy: 0.5291
Test Accuracy: 0.5291000008583069
Test Loss: 1.4613064527511597
```

[64]:
```
history_dict3 = history3.history
acc_values3 = history_dict3['accuracy']
val_acc_values3 = history_dict3['val_accuracy']
val_loss3 = history_dict3['loss']
val_loss_values3 = history_dict3['val_loss']

epochs3 = range(1, len(val_loss3) + 1)

plt.plot(epochs3, val_loss3, 'bo', label='Training loss')
plt.plot(epochs3, val_loss_values3, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```



[65]:
```
plt.plot(epochs3, acc_values3, 'bo', label='Training Accuracy')
plt.plot(epochs3, val_acc_values3, 'b', label = 'Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```



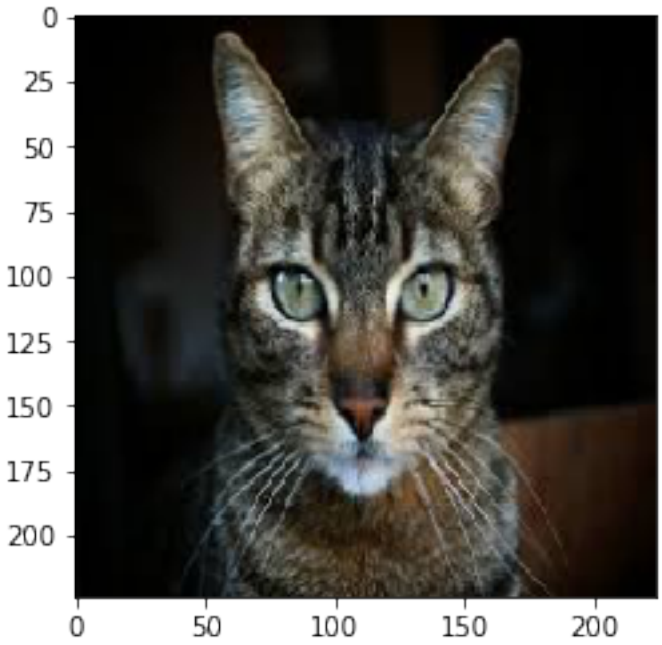
0.3 6.3

[66]:
```
# Import libraries
import os
from pathlib import Path

import numpy as np

from keras.preprocessing import image
from keras.preprocessing.image import img_to_array
from keras.applications.resnet50 import ResNet50
from keras.applications.resnet50 import preprocess_input
from keras.applications.imagenet_utils import decode_predictions
import matplotlib.pyplot as plt
```

[67]:
```
current_dir = Path(os.getcwd()).absolute()
images_dir = current_dir.joinpath('images')
images_dir.mkdir(parents=True, exist_ok = True)
```

[68]:
```
img1 = image.load_img('images/cat.jpg', target_size = (224, 224))

plt.imshow(img1)
```

[68]: <matplotlib.image.AxesImage at 0x7f44d8530340>

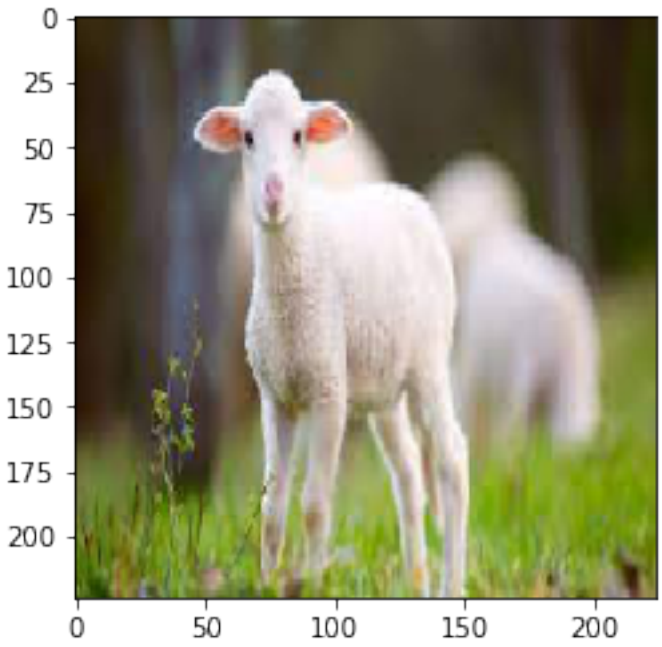


[69]:
```
img1 = image.img_to_array(img1)
img1 = np.expand_dims(img1, axis = 0)
img1 = preprocess_input(img1)
model = ResNet50(weights = 'imagenet')
preds1 = model.predict(img1)
print('Predicted:', decode_predictions(preds1, top = 1)[0])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels.h5
102973440/102967424 [==============================] - 16s 0us/step
Downloading data from https://storage.googleapis.com/download.tensorflow.org/dat
a/imagenet_class_index.json
40960/35363 [==================================] - 0s 1us/step
Predicted: [('n02124075', 'Egyptian_cat', 0.48118547)]
```

[70]:
```
img2 = image.load_img('images/sheep.jpg', target_size = (224, 224))

plt.imshow(img2)
```

[70]: <matplotlib.image.AxesImage at 0x7f44e0084d30>

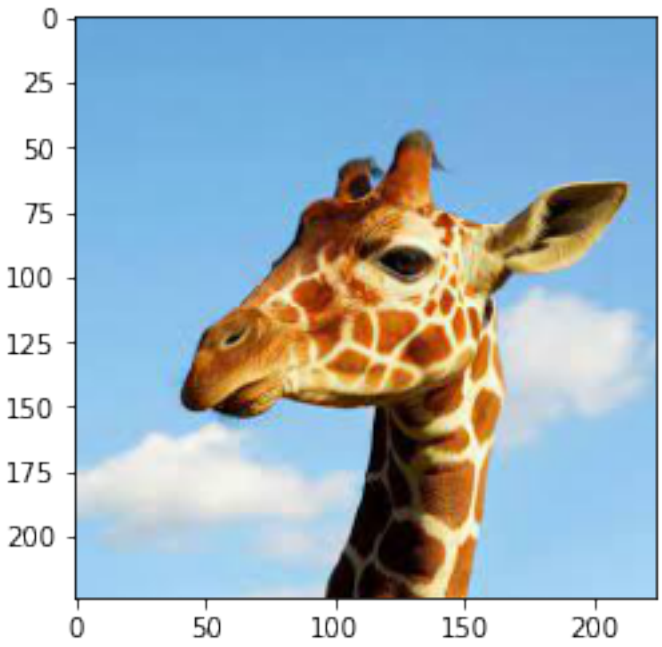


[71]:
```
img2 = image.img_to_array(img2)
img2 = np.expand_dims(img2, axis = 0)
img2 = preprocess_input(img2)
model2 = ResNet50(weights = 'imagenet')
preds2 = model.predict(img2)
print('Predicted:', decode_predictions(preds2, top = 1)[0])
```

```
Predicted: [('n02412080', 'ram', 0.5580335)]
```

[72]:
```
img3 = image.load_img('images/girrafe.jpg', target_size = (224, 224))

plt.imshow(img3)
```

[72]:
```
<matplotlib.image.AxesImage at 0x7f44d032fa60>
```

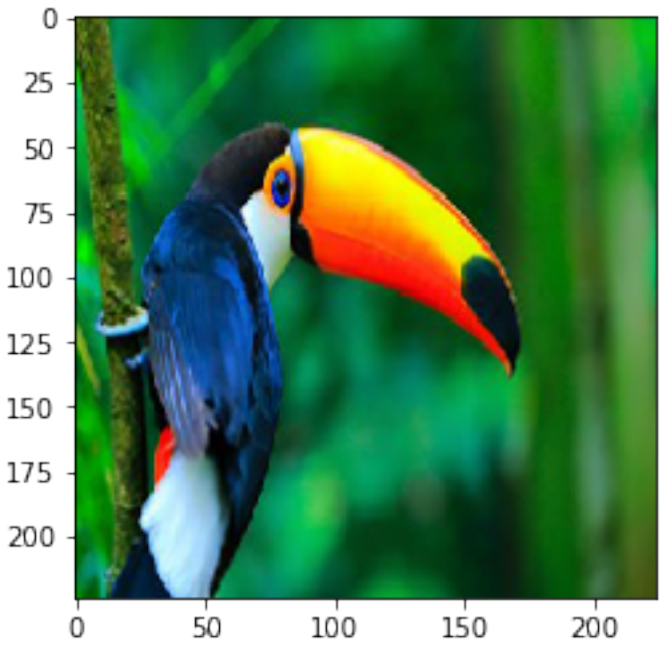


[73]:
```
img3 = image.img_to_array(img3)
img3 = np.expand_dims(img3, axis = 0)
img3 = preprocess_input(img3)
model3 = ResNet50(weights = 'imagenet')
preds3 = model.predict(img3)
print('Predicted:', decode_predictions(preds3, top = 1)[0])
```

```
Predicted: [('n02422699', 'impala', 0.3296979)]
```

[74]:
```
img4 = image.load_img('images/bird-toucan.jpg', target_size = (224, 224))

plt.imshow(img4)
```

[74]:
```
<matplotlib.image.AxesImage at 0x7f44f04117f0>
```

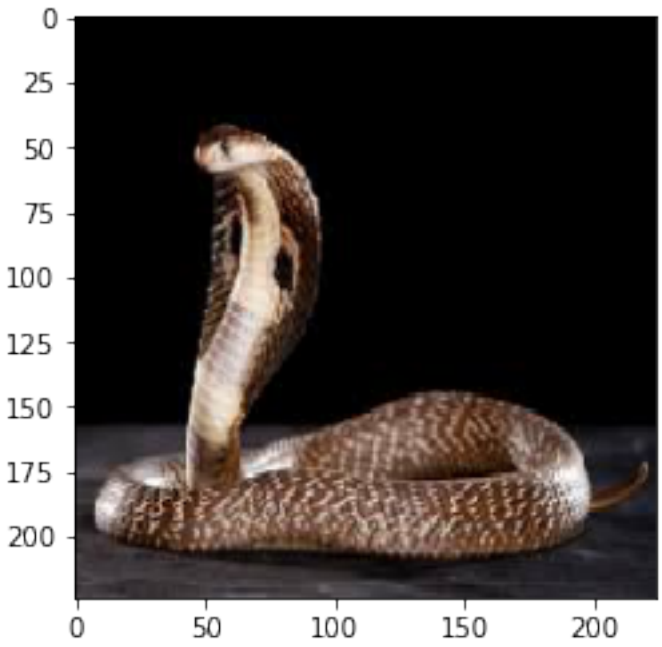


[75]:
```
img4 = image.img_to_array(img4)
img4 = np.expand_dims(img4, axis = 0)
img4 = preprocess_input(img4)
model4 = ResNet50(weights = 'imagenet')
preds4 = model.predict(img4)
print('Predicted:', decode_predictions(preds4, top = 1)[0])
```

```
Predicted: [('n01843383', 'toucan', 0.99644655)]
```

[76]:
```
img5 = image.load_img('images/snake.jpg', target_size = (224, 224))

plt.imshow(img5)
```

[76]:
```
<matplotlib.image.AxesImage at 0x7f44d00c0df0>
```

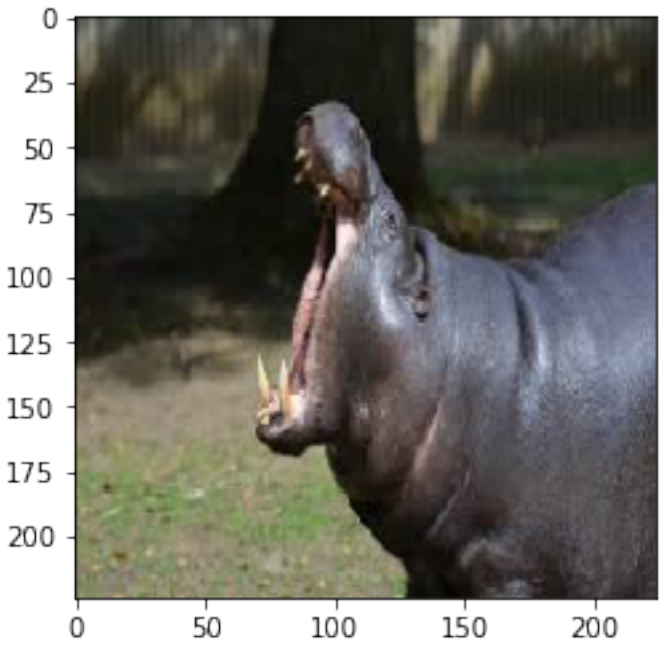


[77]:
```
img5 = image.img_to_array(img5)
img5 = np.expand_dims(img5, axis = 0)
img5 = preprocess_input(img5)
model5 = ResNet50(weights = 'imagenet')
preds5 = model.predict(img5)
print('Predicted:', decode_predictions(preds5, top = 1)[0])
```

```
Predicted: [('n01748264', 'Indian_cobra', 0.9999243)]
```

[78]:
```
img6 = image.load_img('images/hippo.jpg', target_size = (224, 224))

plt.imshow(img6)
```

[78]:
```
<matplotlib.image.AxesImage at 0x7f4494672220>
```

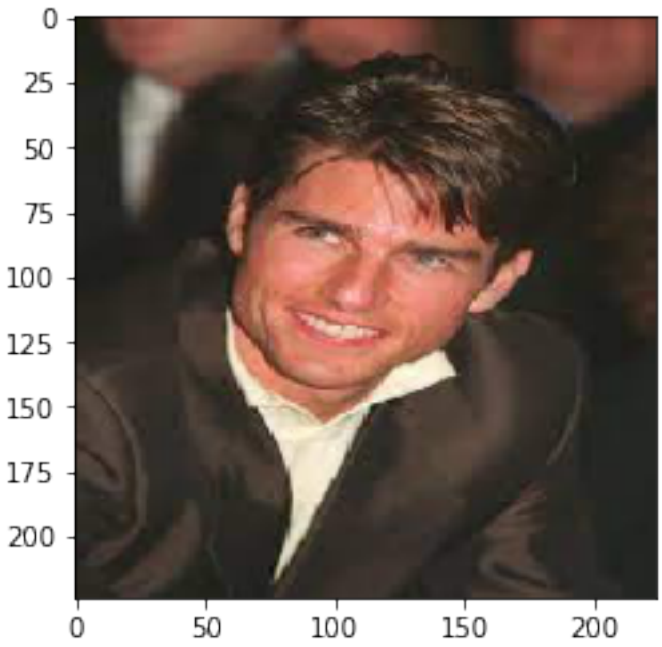


[79]:
```
img6 = image.img_to_array(img6)
img6 = np.expand_dims(img6, axis = 0)
img6 = preprocess_input(img6)
model6 = ResNet50(weights = 'imagenet')
preds6 = model.predict(img6)
print('Predicted:', decode_predictions(preds6, top = 1)[0])
```

```
Predicted: [('n02398521', 'hippopotamus', 0.99706525)]
```

[80]:
```
img7 = image.load_img('images/tomcruise.jpg', target_size = (224, 224))

plt.imshow(img7)
```

[80]:
```
<matplotlib.image.AxesImage at 0x7f44943de6d0>
```



[81]:
```
img7 = image.img_to_array(img7)
img7 = np.expand_dims(img7, axis = 0)
img7 = preprocess_input(img7)
model7 = ResNet50(weights = 'imagenet')
preds7 = model.predict(img7)
print('Predicted:', decode_predictions(preds7, top = 1)[0])
```

```
Predicted: [('n04350905', 'suit', 0.17670709)]
```

[ ]:
```

```