Group 8

Almendrala, Aaron

Ambata, Jo Simon

Caguioa, JV

References

Image Data Generator - https://towardsdatascience.com/image-data-generators-in-keras-7c5fc6928400

Image Recognition with Transfer Learning - https://thedatafrog.com/en/articles/image-recognition-transfer-learning/

Base Model https://www.analyticsvidhya.com/blog/2020/10/create-image-classification-model-python-keras/

VGG16 Transfer Learning - https://www.learndatasci.com/tutorials/hands-on-transfer-learning-keras/#:~:text=In%20the%202014%20ImageNet%20Classification,present%20in%20our%20Food%20dataset.

VGG16 Architecture - https://neurohive.io/en/popular-networks/vgg16/

Dataset (collection of observations on animals such as tracks, sightings, etc. uploaded by the users)- https://www.inaturalist.org/observations?taxon_id=41636

Import Required Libraries

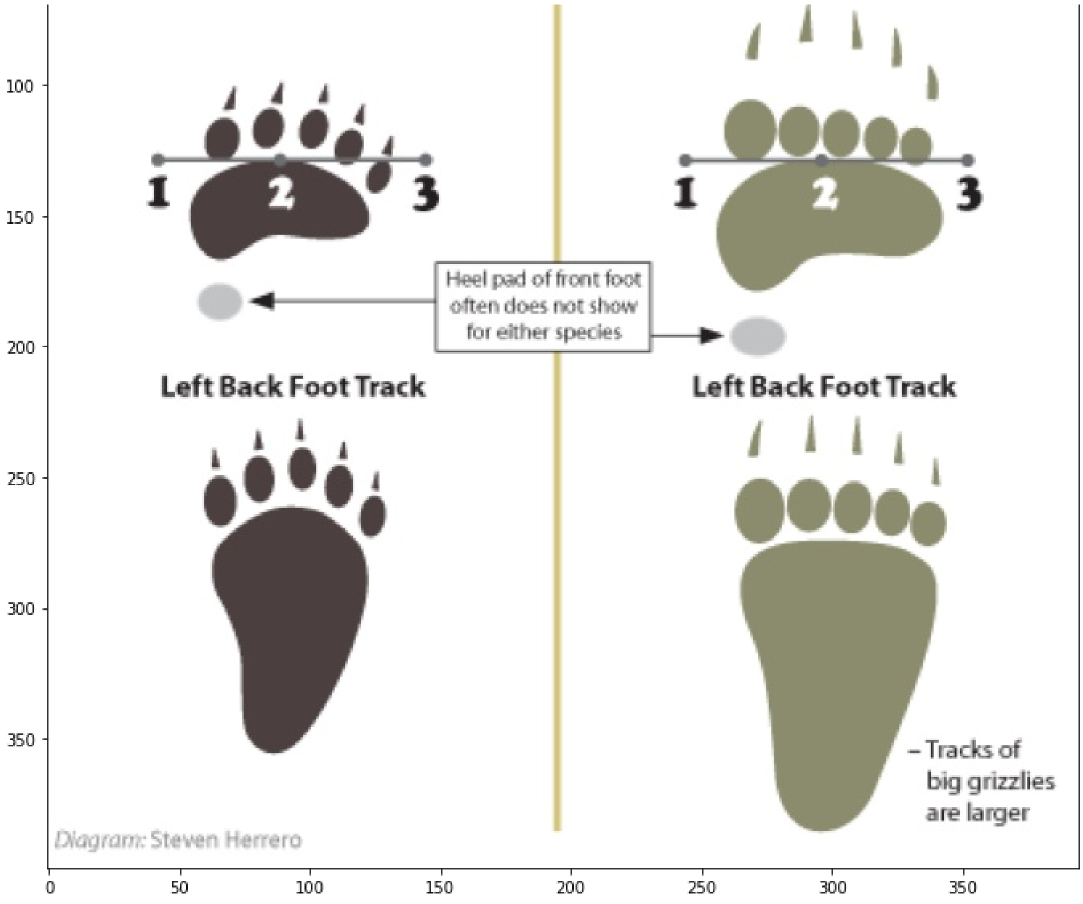
In [1]:

```
import tensorflow
import tensorflow.keras as keras
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as img
import seaborn as sns
import cv2
import os

from PIL import Image
from skimage import io
from matplotlib.pyplot import figure
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D , MaxPool2D , Flatten , Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Sequential
from sklearn.metrics import classification_report,confusion_matrix

vgg16 = tensorflow.keras.applications.vgg16
```

Dataset

In [2]:

```
figure(figsize=(12,12))
plt.imshow(img.imread('dataset/beartrack_id.png'))
plt.show()
print("Source: http://westernwildlife.org/grizzly-bear-outreach-project/bear-identification/")
```



Source: http://westernwildlife.org/grizzly-bear-outreach-project/bear-identification/

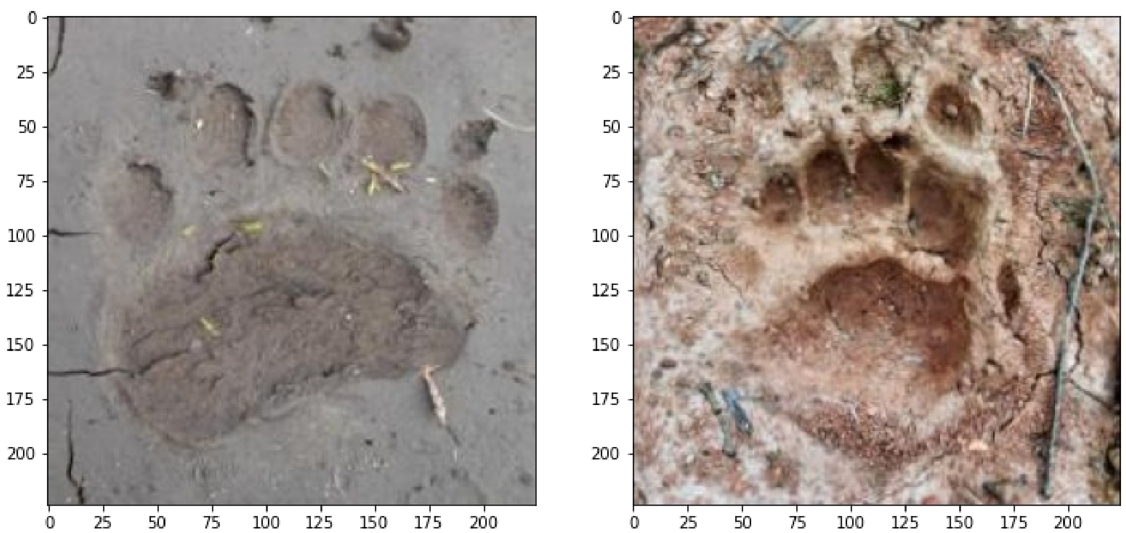
In [3]:

```
print("Sample of Black Bear footprint")
figure(figsize=(12,12))
plt.subplot(1,2,1)
plt.imshow(img.imread('dataset/black/aug_4_3612.jpg'))
plt.subplot(1,2,2)
plt.imshow(img.imread('dataset/black/aug_9_1684.jpg'))
```

Sample of Black Bear footprint

Out[3]:

<matplotlib.image.AxesImage at 0x14fa4b40688>



In [4]:

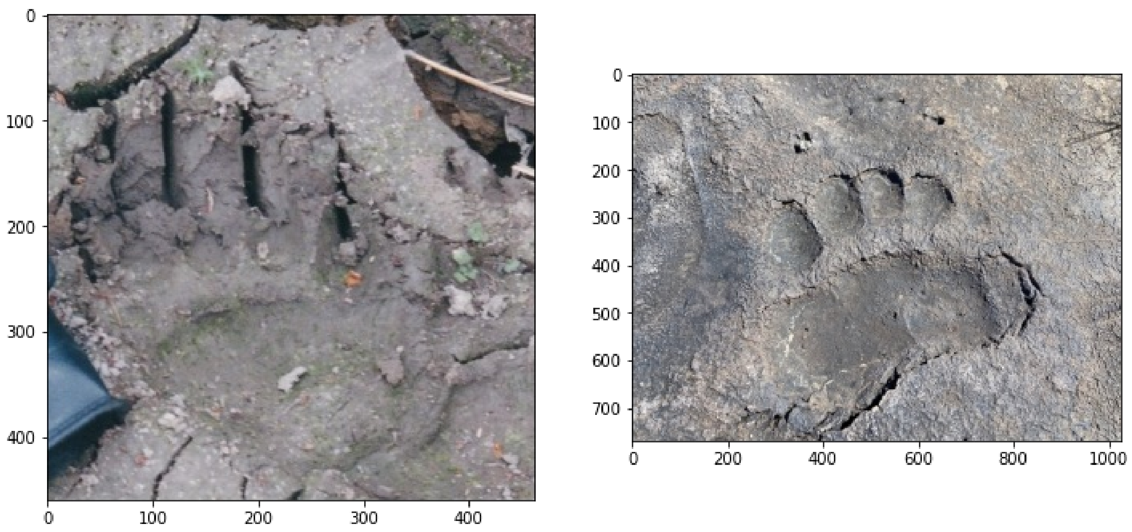
In [4]:

```
print("Sample of Grizzly Bear footprint")
figure(figsize=(12,12))
plt.subplot(1,2,1)
plt.imshow(img.imread('dataset/grizzly/large (9).jpg'))
plt.subplot(1,2,2)
plt.imshow(img.imread('dataset/grizzly/large (4).jpg'))
```

```
Sample of Grizzly Bear footprint
```

Out[4]:

```
<matplotlib.image.AxesImage at 0x14fa5300148>
```



Data Augmentation Setting

In [5]:

```
datagen = ImageDataGenerator(
            rotation_range=30,          #rotate
            width_shift_range=0.2,      #Moves left or right
            height_shift_range=0.2,     #moves up or down
            horizontal_flip=True,       #flips image
            fill_mode='constant')
```

Data Augmentation function

increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data

In [6]:

```
def Augment_data(image_path, save_to_path, data_size, no_of_generation):
    image_directory = image_path
    SIZE = 224
    dataset = []

    my_images = os.listdir(image_directory)
    for i, image_name in enumerate(my_images):
        if (image_name.split('.')[1] == 'jpg'):
            image = io.imread(image_directory + image_name)
            image = Image.fromarray(image, 'RGB')
            image = image.resize((SIZE,SIZE))
            dataset.append(np.array(image))

    x = np.array(dataset)

    i = 0
    for batch in datagen.flow(x, batch_size=data_size,
                              save_to_dir=save_to_path,
                              save_prefix='aug',
```

```
                                    save_format='jpg'):
        i += 1
        if i > no_of_generation-1:
            break
```

black bear train set

In [ ]:

```
Augment_data('train/black/',
             r'C:\Users\RAZER-MERCURY\Documents\school(3Y3T)\Artificial Intelligence 4\project\trai
n\black',
             68,
             20)
```

grizzly bear train set

In [ ]:

```
Augment_data('train/grizzly/',
             r'C:\Users\RAZER-MERCURY\Documents\school(3Y3T)\Artificial Intelligence 4\project\trai
n\grizzly',
             68,
             20)
```

black bear test set

In [ ]:

```
Augment_data('test/black/',
             r'C:\Users\RAZER-MERCURY\Documents\school(3Y3T)\Artificial Intelligence 4\project\test
\black',
             18,
             20)
```

grizzly bear test set

In [ ]:

```
Augment_data('test/grizzly/',
             r'C:\Users\RAZER-MERCURY\Documents\school(3Y3T)\Artificial Intelligence 4\project\test
\grizzly',
             18,
             20)
```

Load Data

In [7]:

```
labels = ['black', 'grizzly']
img_size = 224
def get_data(data_dir):
    data = []
    for label in labels:
        path = os.path.join(data_dir, label)
        class_num = labels.index(label)
        for img in os.listdir(path):
            try:
                img_arr = cv2.imread(os.path.join(path, img))[...,::-1] #convert BGR to RGB format
                resized_arr = cv2.resize(img_arr, (img_size, img_size)) # Reshaping images to
preferred size
                data.append([resized_arr, class_num])
            except Exception as e:
                print(e)
    return np.array(data)
```

Visualize the data

In [8]:

```
train = get_data('train')
val = get_data('test')

#these folders contains black and grizzly folders
```

```
C:\ProgramData\Anaconda3\envs\tensorflow-gpu\lib\site-packages\ipykernel_launcher.py:15:
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tu
ple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant t
o do this, you must specify 'dtype=object' when creating the ndarray.
  from ipykernel import kernelapp as app
```

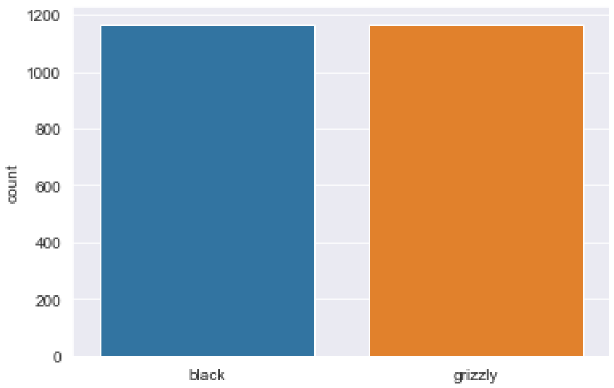
In [9]:

```
l = []
for i in train:
    if(i[1] == 0):
        l.append("black")
    else:
        l.append("grizzly")
sns.set_style('darkgrid')
sns.countplot(l)
```

```
C:\ProgramData\Anaconda3\envs\tensorflow-gpu\lib\site-packages\seaborn\_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit keyword will r
esult in an error or misinterpretation.
  FutureWarning
```

Out[9]:

```
<AxesSubplot:ylabel='count'>
```



In [10]:

```
x_train = []
y_train = []
x_val = []
y_val = []

for feature, label in train:
    x_train.append(feature)
    y_train.append(label)

for feature, label in val:
    x_val.append(feature)
    y_val.append(label)

# Normalize the data
x_train = np.array(x_train) / 255
x_val = np.array(x_val) / 255

x_train.reshape(-1, img_size, img_size, 1)
```

```
y_train = np.array(y_train)

x_val.reshape(-1, img_size, img_size, 1)
y_val = np.array(y_val)
```

Define the Model

In [11]:

```
base_model = Sequential()
base_model.add(Conv2D(32,3,padding="same", activation="relu", input_shape=(224,224,3)))
base_model.add(MaxPool2D())

base_model.add(Conv2D(32, 3, padding="same", activation="relu"))
base_model.add(MaxPool2D())

base_model.add(Conv2D(64, 3, padding="same", activation="relu"))
base_model.add(MaxPool2D())
base_model.add(Dropout(0.4))

base_model.add(Flatten())
base_model.add(Dense(128,activation="relu"))
base_model.add(Dense(2, activation="softmax"))

base_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 224, 224, 32)      896       
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 112, 112, 32)      0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 112, 112, 32)      9248      
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 56, 56, 32)        0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 56, 56, 64)        18496     
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 28, 28, 64)        0         
_________________________________________________________________
dropout (Dropout)            (None, 28, 28, 64)        0         
_________________________________________________________________
flatten (Flatten)            (None, 50176)             0         
_________________________________________________________________
dense (Dense)                (None, 128)               6422656   
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 258       
=================================================================
Total params: 6,451,554
Trainable params: 6,451,554
Non-trainable params: 0
_________________________________________________________________
```

In [14]:

```
opt = Adam(lr=0.00001)
base_model.compile(optimizer = opt , loss = tensorflow.keras.losses.SparseCategoricalCrossentropy(f
rom_logits=True) , metrics = ['acc'])
```

In [15]:

```
history = base_model.fit(x_train,y_train, epochs = 128 , validation_data = (x_val, y_val))
```

```
Train on 2336 samples, validate on 592 samples
Epoch 1/128
2336/2336 [==============================] - 6s 3ms/sample - loss: 0.6932 - acc: 0.5103 -
val_loss: 0.6943 - val_acc: 0.5000
Epoch 2/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.6855 - acc: 0.5595 -
```

```
val_loss: 0.6965 - val_acc: 0.4865
Epoch 3/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.6743 - acc: 0.5938 -
val_loss: 0.6983 - val_acc: 0.4916
Epoch 4/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.6616 - acc: 0.6173 -
val_loss: 0.6969 - val_acc: 0.5304
Epoch 5/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.6512 - acc: 0.6485 -
val_loss: 0.6883 - val_acc: 0.5507
Epoch 6/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.6410 - acc: 0.6648 -
val_loss: 0.6775 - val_acc: 0.5794
Epoch 7/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.6308 - acc: 0.6866 -
val_loss: 0.6696 - val_acc: 0.5946
Epoch 8/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.6217 - acc: 0.6973 -
val_loss: 0.6628 - val_acc: 0.6047
Epoch 9/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.6147 - acc: 0.6952 -
val_loss: 0.6543 - val_acc: 0.6115
Epoch 10/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.6057 - acc: 0.7085 -
val_loss: 0.6375 - val_acc: 0.6503
Epoch 11/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.5962 - acc: 0.7269 -
val_loss: 0.6500 - val_acc: 0.6132
Epoch 12/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.5908 - acc: 0.7192 -
val_loss: 0.6381 - val_acc: 0.6182
Epoch 13/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.5829 - acc: 0.7354 -
val_loss: 0.6278 - val_acc: 0.6385
Epoch 14/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.5746 - acc: 0.7581 -
val_loss: 0.6270 - val_acc: 0.6757
Epoch 15/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.5694 - acc: 0.7654 -
val_loss: 0.6178 - val_acc: 0.6757
Epoch 16/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.5628 - acc: 0.7770 -
val_loss: 0.6097 - val_acc: 0.6943
Epoch 17/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.5555 - acc: 0.7851 -
val_loss: 0.6131 - val_acc: 0.6503
Epoch 18/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.5521 - acc: 0.7778 -
val_loss: 0.6047 - val_acc: 0.7128
Epoch 19/128
2336/2336 [==============================] - 5s 2ms/sample - loss: 0.5442 - acc: 0.8065 -
val_loss: 0.6071 - val_acc: 0.6774
Epoch 20/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.5401 - acc: 0.8095 -
val_loss: 0.5951 - val_acc: 0.6993
Epoch 21/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.5309 - acc: 0.8215 -
val_loss: 0.5994 - val_acc: 0.6909
Epoch 22/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.5286 - acc: 0.8305 -
val_loss: 0.6015 - val_acc: 0.6706
Epoch 23/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.5227 - acc: 0.8283 -
val_loss: 0.5886 - val_acc: 0.6959
Epoch 24/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.5166 - acc: 0.8429 -
val_loss: 0.5832 - val_acc: 0.7145s - los - ETA: 1s - loss
Epoch 25/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.5117 - acc: 0.8515 -
val_loss: 0.5908 - val_acc: 0.7027
Epoch 26/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.5093 - acc: 0.8455 -
val_loss: 0.5852 - val_acc: 0.7095
Epoch 27/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.5047 - acc: 0.8553 -
val_loss: 0.5809 - val_acc: 0.7078
Epoch 28/128
```

```
Epoch 28/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.5009 - acc: 0.8549 -
val_loss: 0.5769 - val_acc: 0.7179
Epoch 29/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4960 - acc: 0.8587 -
val_loss: 0.5709 - val_acc: 0.7179
Epoch 30/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4918 - acc: 0.8716 -
val_loss: 0.5768 - val_acc: 0.6976
Epoch 31/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4921 - acc: 0.8587 -
val_loss: 0.5680 - val_acc: 0.7280
Epoch 32/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4858 - acc: 0.8669 -
val_loss: 0.5802 - val_acc: 0.7213
Epoch 33/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4826 - acc: 0.8716 -
val_loss: 0.5715 - val_acc: 0.7145
Epoch 34/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4794 - acc: 0.8733 -
val_loss: 0.5604 - val_acc: 0.7365
Epoch 35/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4779 - acc: 0.8767 -
val_loss: 0.5649 - val_acc: 0.7247
Epoch 36/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4762 - acc: 0.8797 -
val_loss: 0.5553 - val_acc: 0.7416
Epoch 37/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4715 - acc: 0.8887 -
val_loss: 0.5633 - val_acc: 0.7213
Epoch 38/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4673 - acc: 0.8870 -
val_loss: 0.5564 - val_acc: 0.7365
Epoch 39/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4643 - acc: 0.8926 -
val_loss: 0.5533 - val_acc: 0.7534
Epoch 40/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4625 - acc: 0.8870 -
val_loss: 0.5556 - val_acc: 0.7331
Epoch 41/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4577 - acc: 0.9011 -
val_loss: 0.5599 - val_acc: 0.7314
Epoch 42/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4630 - acc: 0.8896 -
val_loss: 0.5563 - val_acc: 0.7297
Epoch 43/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4532 - acc: 0.9075 -
val_loss: 0.5535 - val_acc: 0.7399
Epoch 44/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4509 - acc: 0.9067 -
val_loss: 0.5529 - val_acc: 0.7365
Epoch 45/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4512 - acc: 0.8990 -
val_loss: 0.5505 - val_acc: 0.7382
Epoch 46/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4484 - acc: 0.9088 -
val_loss: 0.5588 - val_acc: 0.7280
Epoch 47/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4485 - acc: 0.9033 -
val_loss: 0.5524 - val_acc: 0.7432
Epoch 48/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4486 - acc: 0.8977 -
val_loss: 0.5469 - val_acc: 0.7432
Epoch 49/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4422 - acc: 0.9118 -
val_loss: 0.5521 - val_acc: 0.7331
Epoch 50/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4381 - acc: 0.9165 -
val_loss: 0.5481 - val_acc: 0.7483
Epoch 51/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4378 - acc: 0.9140 -
val_loss: 0.5465 - val_acc: 0.7449
Epoch 52/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4363 - acc: 0.9217 -
val_loss: 0.5447 - val_acc: 0.7416
Epoch 53/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4361 - acc: 0.9238 -
val_loss: 0.5405 - val_acc: 0.7500
```

```
val_loss: 0.5405 - val_acc: 0.7500
Epoch 54/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4325 - acc: 0.9225 -
val_loss: 0.5401 - val_acc: 0.7500
Epoch 55/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4290 - acc: 0.9221 -
val_loss: 0.5349 - val_acc: 0.7703
Epoch 56/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4267 - acc: 0.9277 -
val_loss: 0.5479 - val_acc: 0.7483
Epoch 57/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4276 - acc: 0.9251 -
val_loss: 0.5341 - val_acc: 0.7669
Epoch 58/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4227 - acc: 0.9358 -
val_loss: 0.5363 - val_acc: 0.7601
Epoch 59/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4210 - acc: 0.9272 -
val_loss: 0.5372 - val_acc: 0.7500
Epoch 60/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4220 - acc: 0.9294 -
val_loss: 0.5434 - val_acc: 0.7534
Epoch 61/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4203 - acc: 0.9332 -
val_loss: 0.5350 - val_acc: 0.7601
Epoch 62/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4174 - acc: 0.9349 -
val_loss: 0.5499 - val_acc: 0.7432
Epoch 63/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4140 - acc: 0.9362 -
val_loss: 0.5476 - val_acc: 0.7449
Epoch 64/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4157 - acc: 0.9388 -
val_loss: 0.5389 - val_acc: 0.7551
Epoch 65/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4134 - acc: 0.9384 -
val_loss: 0.5351 - val_acc: 0.7584
Epoch 66/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4129 - acc: 0.9362 -
val_loss: 0.5346 - val_acc: 0.7635
Epoch 67/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4110 - acc: 0.9371 -
val_loss: 0.5294 - val_acc: 0.7686
Epoch 68/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4088 - acc: 0.9388 -
val_loss: 0.5284 - val_acc: 0.7720
Epoch 69/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4118 - acc: 0.9345 -
val_loss: 0.5318 - val_acc: 0.7703
Epoch 70/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4059 - acc: 0.9409 -
val_loss: 0.5363 - val_acc: 0.7584
Epoch 71/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4044 - acc: 0.9448 -
val_loss: 0.5266 - val_acc: 0.7720
Epoch 72/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4048 - acc: 0.9456 -
val_loss: 0.5448 - val_acc: 0.7517
Epoch 73/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4014 - acc: 0.9473 -
val_loss: 0.5246 - val_acc: 0.7652
Epoch 74/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.4028 - acc: 0.9456 -
val_loss: 0.5274 - val_acc: 0.7652
Epoch 75/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3967 - acc: 0.9598 -
val_loss: 0.5274 - val_acc: 0.7584
Epoch 76/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3952 - acc: 0.9572 -
val_loss: 0.5241 - val_acc: 0.7770
Epoch 77/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3968 - acc: 0.9508 -
val_loss: 0.5246 - val_acc: 0.7787
Epoch 78/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3955 - acc: 0.9533 -
val_loss: 0.5212 - val_acc: 0.7736
Epoch 79/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3936 - acc: 0.9559 -
```

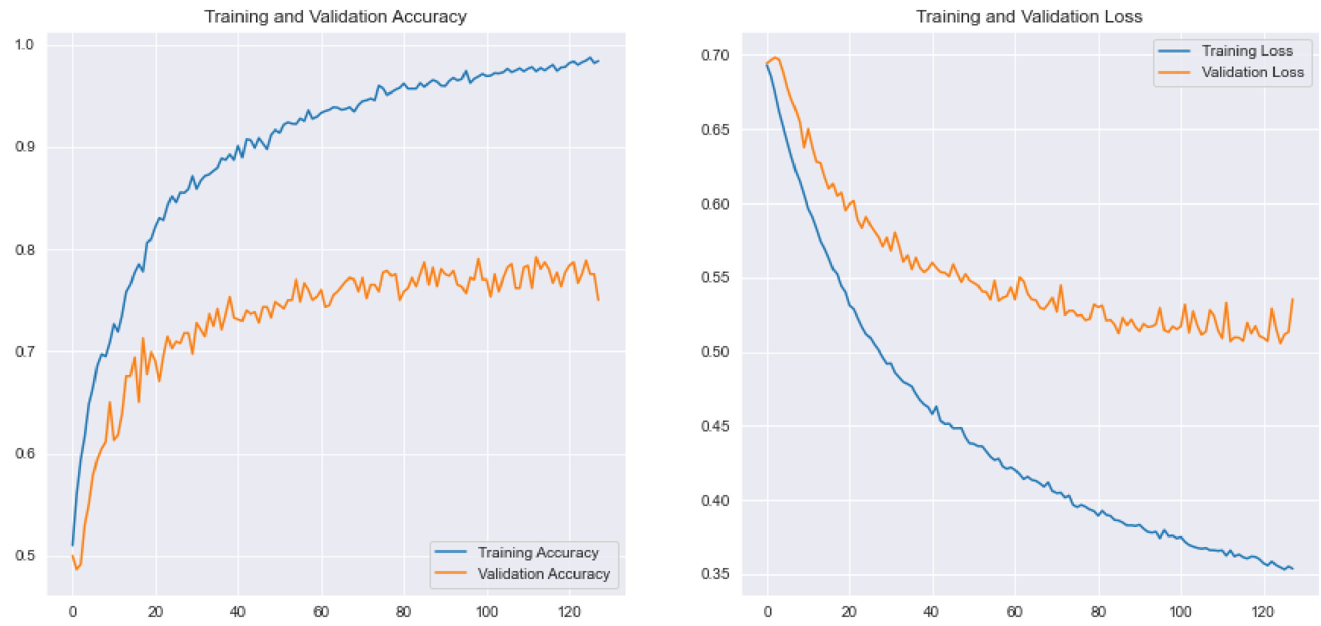
```
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3936 - acc: 0.9559 -
val_loss: 0.5221 - val_acc: 0.7753
Epoch 80/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3925 - acc: 0.9576 -
val_loss: 0.5318 - val_acc: 0.7500
Epoch 81/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3894 - acc: 0.9619 -
val_loss: 0.5297 - val_acc: 0.7584
Epoch 82/128
2336/2336 [==============================] - 5s 2ms/sample - loss: 0.3928 - acc: 0.9568 -
val_loss: 0.5309 - val_acc: 0.7618
Epoch 83/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3899 - acc: 0.9568 -
val_loss: 0.5209 - val_acc: 0.7720
Epoch 84/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3893 - acc: 0.9568 -
val_loss: 0.5214 - val_acc: 0.7635
Epoch 85/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3865 - acc: 0.9623 -
val_loss: 0.5182 - val_acc: 0.7753
Epoch 86/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3861 - acc: 0.9585 -
val_loss: 0.5122 - val_acc: 0.7872
Epoch 87/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3847 - acc: 0.9623 -
val_loss: 0.5229 - val_acc: 0.7652
Epoch 88/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3828 - acc: 0.9653 -
val_loss: 0.5178 - val_acc: 0.7821
Epoch 89/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3828 - acc: 0.9636 -
val_loss: 0.5218 - val_acc: 0.7635
Epoch 90/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3824 - acc: 0.9598 -
val_loss: 0.5166 - val_acc: 0.7804
Epoch 91/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3833 - acc: 0.9593 -
val_loss: 0.5137 - val_acc: 0.7753
Epoch 92/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3805 - acc: 0.9645 -
val_loss: 0.5188 - val_acc: 0.7736
Epoch 93/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3784 - acc: 0.9675 -
val_loss: 0.5166 - val_acc: 0.7787
Epoch 94/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3778 - acc: 0.9649 -
val_loss: 0.5169 - val_acc: 0.7652
Epoch 95/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3785 - acc: 0.9662 -
val_loss: 0.5184 - val_acc: 0.7635
Epoch 96/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3741 - acc: 0.9743 -
val_loss: 0.5293 - val_acc: 0.7568
Epoch 97/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3795 - acc: 0.9623 -
val_loss: 0.5143 - val_acc: 0.7720
Epoch 98/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3756 - acc: 0.9666 -
val_loss: 0.5131 - val_acc: 0.7703
Epoch 99/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3759 - acc: 0.9688 -
val_loss: 0.5170 - val_acc: 0.7905
Epoch 100/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3741 - acc: 0.9713 -
val_loss: 0.5150 - val_acc: 0.7703
Epoch 101/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3750 - acc: 0.9692 -
val_loss: 0.5170 - val_acc: 0.7703
Epoch 102/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3718 - acc: 0.9696 -
val_loss: 0.5316 - val_acc: 0.7534
Epoch 103/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3697 - acc: 0.9722 -
val_loss: 0.5126 - val_acc: 0.7753
Epoch 104/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3686 - acc: 0.9717 -
val_loss: 0.5271 - val_acc: 0.7584
Epoch 105/128
```

```
Epoch 105/128
2336/2336 [==============================] - 5s 2ms/sample - loss: 0.3677 - acc: 0.9730 -
val_loss: 0.5179 - val_acc: 0.7720
Epoch 106/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3671 - acc: 0.9765 -
val_loss: 0.5114 - val_acc: 0.7821
Epoch 107/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3676 - acc: 0.9730 -
val_loss: 0.5134 - val_acc: 0.7855
Epoch 108/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3661 - acc: 0.9747 -
val_loss: 0.5278 - val_acc: 0.7618
Epoch 109/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3661 - acc: 0.9769 -
val_loss: 0.5240 - val_acc: 0.7618
Epoch 110/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3656 - acc: 0.9739 -
val_loss: 0.5141 - val_acc: 0.7821
Epoch 111/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3660 - acc: 0.9765 -
val_loss: 0.5088 - val_acc: 0.7838
Epoch 112/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3624 - acc: 0.9782 -
val_loss: 0.5329 - val_acc: 0.7618
Epoch 113/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3660 - acc: 0.9739 -
val_loss: 0.5067 - val_acc: 0.7922
Epoch 114/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3619 - acc: 0.9773 -
val_loss: 0.5096 - val_acc: 0.7804
Epoch 115/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3632 - acc: 0.9747 -
val_loss: 0.5093 - val_acc: 0.7872
Epoch 116/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3613 - acc: 0.9777 -
val_loss: 0.5070 - val_acc: 0.7804
Epoch 117/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3604 - acc: 0.9803 -
val_loss: 0.5196 - val_acc: 0.7669
Epoch 118/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3618 - acc: 0.9743 -
val_loss: 0.5121 - val_acc: 0.7770
Epoch 119/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3616 - acc: 0.9777 -
val_loss: 0.5172 - val_acc: 0.7635
Epoch 120/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3600 - acc: 0.9782 -
val_loss: 0.5104 - val_acc: 0.7770
Epoch 121/128
2336/2336 [==============================] - 5s 2ms/sample - loss: 0.3572 - acc: 0.9820 -
val_loss: 0.5092 - val_acc: 0.7838
Epoch 122/128
2336/2336 [==============================] - 5s 2ms/sample - loss: 0.3557 - acc: 0.9837 -
val_loss: 0.5069 - val_acc: 0.7872
Epoch 123/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3584 - acc: 0.9803 -
val_loss: 0.5289 - val_acc: 0.7669
Epoch 124/128
2336/2336 [==============================] - 4s 2ms/sample - loss: 0.3558 - acc: 0.9829 -
val_loss: 0.5150 - val_acc: 0.7753
Epoch 125/128
2336/2336 [==============================] - 5s 2ms/sample - loss: 0.3544 - acc: 0.9846 -
val_loss: 0.5054 - val_acc: 0.7889
Epoch 126/128
2336/2336 [==============================] - 5s 2ms/sample - loss: 0.3530 - acc: 0.9876 -
val_loss: 0.5115 - val_acc: 0.7753
Epoch 127/128
2336/2336 [==============================] - 5s 2ms/sample - loss: 0.3550 - acc: 0.9820 -
val_loss: 0.5131 - val_acc: 0.7753
Epoch 128/128
2336/2336 [==============================] - 5s 2ms/sample - loss: 0.3535 - acc: 0.9842 -
val_loss: 0.5353 - val_acc: 0.7500
```

Results

In [16]:

```python
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(len(acc))

plt.figure(figsize=(15, 15))
plt.subplot(2, 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(2, 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 Validation Loss')
plt.show()
```



In [18]:

```python
predictions = base_model.predict_classes(x_val)
predictions = predictions.reshape(1,-1)[0]
print(classification_report(y_val, predictions, target_names = ['Black Bear (Class 0)','Grizzly
Bear (Class 1)']))
```

```
                      precision    recall  f1-score   support

 Black Bear (Class 0)       0.71      0.86      0.77       296
Grizzly Bear (Class 1)      0.82      0.64      0.72       296

             accuracy                           0.75       592
            macro avg       0.76      0.75      0.75       592
         weighted avg       0.76      0.75      0.75       592
```

2) Transfer Learning

Data augmentation

Using data-augmentation in order to provide enough data to our model, avoiding overfitting

In [19]:

```
#instantiate ImageDataGenerator which handles the parameter on preprocessing and transformation of the  images

imgdatagen = ImageDataGenerator(
    preprocessing_function=vgg16.preprocess_input, #since we are using vgg16 as our model, we also use its proprocessing
    horizontal_flip=True,      #flips image
    rotation_range=30,         #rotate
    width_shift_range=0.2,     #moves left or right
    height_shift_range=0.1,    #moves up or down
    validation_split = 0.2,    #divide train and test
)
```

In [20]:

```
datasetdir = 'dataset'
os.chdir(datasetdir)

#VGG16 model accept an input shape of (224,224,3)
shape = (224,224)
batch_size = 8

train_dataset = imgdatagen.flow_from_directory(
    os.getcwd(),
    target_size = shape,
    batch_size = batch_size,
    subset = 'training',
    shuffle = True,
    seed=42
)

val_dataset = imgdatagen.flow_from_directory(
    os.getcwd(),
    target_size = shape,
    batch_size = batch_size,
    subset = 'validation',
    shuffle = True,
    seed=42
)

test_dataset = imgdatagen.flow_from_directory(
    os.getcwd(),
    target_size=(224, 224),
    class_mode=None,
    batch_size=1,
    shuffle = False,
    seed=42)

print("")
print("Class names are", ' and '.join([str(x) for x in train_dataset.class_indices]))
print("")
print("Sample per class in train dataset:",
int(train_dataset.samples/len(train_dataset.class_indices)))
print("Sample per class in val dataset:", int(val_dataset.samples/len(val_dataset.class_indices)))
print("")


#the output corresponds to (batch_size, height, width, number of channels)
x,y = next(train_dataset)
print(x.shape)

#we have batch size of 8 which results to have minibatches per epoch (sample/batchsize = number of data trained per epoch)
#224x224 as we resize it to fit in the model
#3 since it is an rgb channel
```

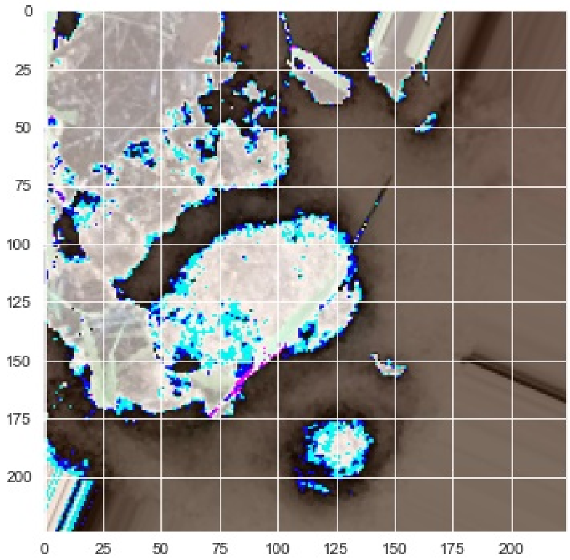
```
Found 138 images belonging to 2 classes.
Found 34 images belonging to 2 classes.
Found 172 images belonging to 2 classes.

Class names are black and grizzly

Sample per class in train dataset: 69
Sample per class in val dataset: 17
```

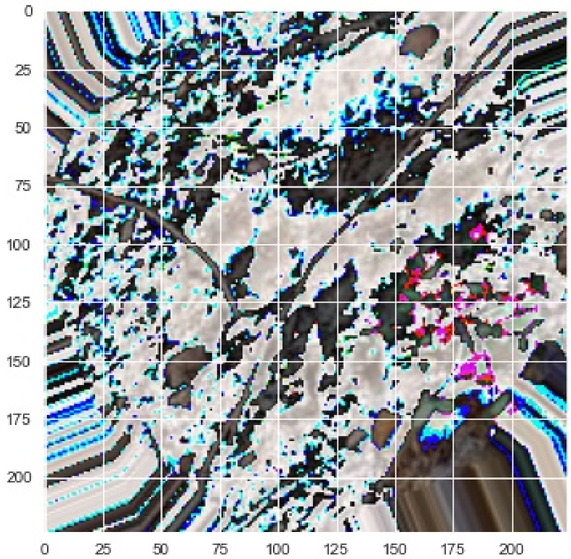
```
(8, 224, 224, 3)
```

In [21]:

```python
print("Sample augmented images from train_dataset")

x,y = train_dataset.next()
for i in range(0,2):
    image = x[i]
    figure(figsize=(6,6))
    plt.imshow(image.astype('uint8'))
    plt.show()

#the sample image is already preprocessed and transformed using the parameters in
ImageDataGenerator
```

```
Sample augmented images from train_dataset
```





Model Architecture

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous model submitted to ILSVRC-2014. It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. VGG16 was trained for weeks and was using NVIDIA Titan Black GPU’s.

Source: https://neurohive.io/en/popular-networks/vgg16/

In [22]:

```
#instantiate the vgg16 model with  weights pre-trained from imagenet
#add input shape which matches our dataset

conv_model = vgg16.VGG16(weights='imagenet', include_top=False, input_shape=(224,224,3))

# flatten the output of the convolutional part
x = keras.layers.Flatten()(conv_model.output)

# add two hidden layers to act as feature extractor
x = keras.layers.Dense(100, activation='relu')(x)
x = keras.layers.Dense(100, activation='relu')(x)

#adding dropout layer to reduce overfitting
x = keras.layers.Dropout(0.2)(x)

# two neurons since we have two classes and sigmoid activation for output layer
predictions = keras.layers.Dense(2, activation='sigmoid')(x)

# compile model
bear_foot_model = keras.models.Model(inputs=conv_model.input, outputs=predictions)

#setting the vgg16 to not be trainable so we dont change its weight
for layer in conv_model.layers:
    layer.trainable = False

bear_foot_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_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
```

```
_________________________________________________________________
dense_2 (Dense)              (None, 100)               2508900
_________________________________________________________________
dense_3 (Dense)              (None, 100)               10100
_________________________________________________________________
dropout_1 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 202
=================================================================
Total params: 17,233,890
Trainable params: 2,519,202
Non-trainable params: 14,714,688
_________________________________________________________________
```

Model Training

In [23]:

```
#we use binary_crossentropy since we are classifying two classes, and Adam for our optimizer with
the learning rate of 0.001
bear_foot_model.compile(loss='binary_crossentropy',
                  optimizer=keras.optimizers.Adam(lr=0.0001),
                  metrics=['acc'])

#fit_generator is used in order to fit the images produced from the ImageDataGenerator
history = bear_foot_model.fit_generator(
    train_dataset,
    validation_data = val_dataset,
    workers=10,
    epochs=128,
)

#save the model
bear_foot_model.save_weights('custom_vgg16.h5')
```

```
Epoch 1/128
18/18 [==============================] - 8s 424ms/step - loss: 2.2338 - acc: 0.4819 - val_loss: 1.
4857 - val_acc: 0.5441
Epoch 2/128
18/18 [==============================] - 3s 190ms/step - loss: 1.2642 - acc: 0.6449 - val_loss: 1.
7921 - val_acc: 0.4412
Epoch 3/128
18/18 [==============================] - 4s 197ms/step - loss: 1.2069 - acc: 0.5906 - val_loss: 1.
0055 - val_acc: 0.6324
Epoch 4/128
18/18 [==============================] - 4s 203ms/step - loss: 0.8335 - acc: 0.6812 - val_loss: 0.
8472 - val_acc: 0.6471
Epoch 5/128
18/18 [==============================] - 4s 198ms/step - loss: 0.7684 - acc: 0.6739 - val_loss: 0.
7337 - val_acc: 0.6618
Epoch 6/128
18/18 [==============================] - 4s 212ms/step - loss: 0.7827 - acc: 0.6957 - val_loss: 0.
8182 - val_acc: 0.6176
Epoch 7/128
18/18 [==============================] - 4s 201ms/step - loss: 0.7503 - acc: 0.7754 - val_loss: 0.
5367 - val_acc: 0.7059
Epoch 8/128
18/18 [==============================] - 3s 193ms/step - loss: 0.5881 - acc: 0.7428 - val_loss: 0.
7454 - val_acc: 0.7059
Epoch 9/128
18/18 [==============================] - 3s 191ms/step - loss: 0.5137 - acc: 0.7862 - val_loss: 0.
5585 - val_acc: 0.6765
Epoch 10/128
18/18 [==============================] - 3s 191ms/step - loss: 0.5454 - acc: 0.7609 - val_loss: 0.
4575 - val_acc: 0.7647
Epoch 11/128
18/18 [==============================] - 3s 190ms/step - loss: 0.4720 - acc: 0.8225 - val_loss: 0.
5140 - val_acc: 0.8088
Epoch 12/128
18/18 [==============================] - 3s 191ms/step - loss: 0.4690 - acc: 0.8297 - val_loss: 0.
6498 - val_acc: 0.6176
Epoch 13/128
18/18 [==============================] - 3s 190ms/step - loss: 0.5648 - acc: 0.7899 - val_loss: 0.
5851 - val_acc: 0.7500
Epoch 14/128
18/18 [==============================] - 3s 187ms/step - loss: 0.4672 - acc: 0.8333 - val_loss: 0.
```

```
18/18 [==============================] - 3s 187ms/step - loss: 0.4672 - acc: 0.8333 - val_loss: 0.
6142 - val_acc: 0.6765
Epoch 15/128
18/18 [==============================] - 3s 190ms/step - loss: 0.3344 - acc: 0.8442 - val_loss: 0.
5185 - val_acc: 0.7353
Epoch 16/128
18/18 [==============================] - 3s 191ms/step - loss: 0.4299 - acc: 0.8587 - val_loss: 0.
9115 - val_acc: 0.7500
Epoch 17/128
18/18 [==============================] - 4s 195ms/step - loss: 0.4825 - acc: 0.8333 - val_loss: 0.
9720 - val_acc: 0.6912
Epoch 18/128
18/18 [==============================] - 3s 192ms/step - loss: 0.3313 - acc: 0.8551 - val_loss: 0.
5864 - val_acc: 0.7794
Epoch 19/128
18/18 [==============================] - 3s 194ms/step - loss: 0.3337 - acc: 0.8768 - val_loss: 0.
4721 - val_acc: 0.7647
Epoch 20/128
18/18 [==============================] - 4s 204ms/step - loss: 0.3293 - acc: 0.8623 - val_loss: 0.
5418 - val_acc: 0.7794
Epoch 21/128
18/18 [==============================] - 4s 207ms/step - loss: 0.3094 - acc: 0.8841 - val_loss: 0.
5635 - val_acc: 0.7647
Epoch 22/128
18/18 [==============================] - 4s 249ms/step - loss: 0.3683 - acc: 0.8370 - val_loss: 0.
1741 - val_acc: 0.9265
Epoch 23/128
18/18 [==============================] - 4s 247ms/step - loss: 0.3067 - acc: 0.8732 - val_loss: 0.
5618 - val_acc: 0.7206
Epoch 24/128
18/18 [==============================] - 4s 233ms/step - loss: 0.2595 - acc: 0.9167 - val_loss: 0.
4002 - val_acc: 0.7941
Epoch 25/128
18/18 [==============================] - 4s 219ms/step - loss: 0.3239 - acc: 0.8696 - val_loss: 0.
5339 - val_acc: 0.7500
Epoch 26/128
18/18 [==============================] - 4s 205ms/step - loss: 0.2324 - acc: 0.9022 - val_loss: 0.
3275 - val_acc: 0.7941
Epoch 27/128
18/18 [==============================] - 4s 199ms/step - loss: 0.2711 - acc: 0.8732 - val_loss: 0.
3362 - val_acc: 0.8088
Epoch 28/128
18/18 [==============================] - 4s 206ms/step - loss: 0.3166 - acc: 0.8587 - val_loss: 0.
3283 - val_acc: 0.8382
Epoch 29/128
18/18 [==============================] - 4s 232ms/step - loss: 0.2742 - acc: 0.9058 - val_loss: 0.
3112 - val_acc: 0.8235
Epoch 30/128
18/18 [==============================] - 4s 215ms/step - loss: 0.2009 - acc: 0.9203 - val_loss: 0.
1798 - val_acc: 0.9118
Epoch 31/128
18/18 [==============================] - 4s 215ms/step - loss: 0.2387 - acc: 0.9022 - val_loss: 0.
2779 - val_acc: 0.8676
Epoch 32/128
18/18 [==============================] - 4s 216ms/step - loss: 0.2103 - acc: 0.9239 - val_loss: 0.
3217 - val_acc: 0.8529
Epoch 33/128
18/18 [==============================] - 3s 194ms/step - loss: 0.1596 - acc: 0.9275 - val_loss: 0.
3823 - val_acc: 0.8676
Epoch 34/128
18/18 [==============================] - 5s 261ms/step - loss: 0.1860 - acc: 0.9203 - val_loss: 0.
3696 - val_acc: 0.8235
Epoch 35/128
18/18 [==============================] - 4s 236ms/step - loss: 0.1399 - acc: 0.9348 - val_loss: 0.
3573 - val_acc: 0.7941
Epoch 36/128
18/18 [==============================] - 4s 233ms/step - loss: 0.1971 - acc: 0.9094 - val_loss: 0.
3796 - val_acc: 0.8382
Epoch 37/128
18/18 [==============================] - 4s 231ms/step - loss: 0.1823 - acc: 0.9275 - val_loss: 0.
2761 - val_acc: 0.8824
Epoch 38/128
18/18 [==============================] - 4s 216ms/step - loss: 0.2102 - acc: 0.8986 - val_loss: 0.
2954 - val_acc: 0.8824
Epoch 39/128
18/18 [==============================] - 4s 205ms/step - loss: 0.1413 - acc: 0.9348 - val_loss: 0.
4593 - val_acc: 0.8676
Epoch 40/128
```

```
Epoch 40/128
18/18 [==============================] - 4s 201ms/step - loss: 0.1272 - acc: 0.9348 - val_loss: 0.
1925 - val_acc: 0.8971
Epoch 41/128
18/18 [==============================] - 6s 308ms/step - loss: 0.1015 - acc: 0.9565 - val_loss: 0.
1458 - val_acc: 0.9118
Epoch 42/128
18/18 [==============================] - 5s 293ms/step - loss: 0.1154 - acc: 0.9565 - val_loss: 1.
1017 - val_acc: 0.7941
Epoch 43/128
18/18 [==============================] - 5s 271ms/step - loss: 0.2208 - acc: 0.9275 - val_loss: 0.
5225 - val_acc: 0.8382
Epoch 44/128
18/18 [==============================] - 5s 262ms/step - loss: 0.1308 - acc: 0.9493 - val_loss: 0.
4766 - val_acc: 0.8382
Epoch 45/128
18/18 [==============================] - 4s 226ms/step - loss: 0.1322 - acc: 0.9420 - val_loss: 0.
4208 - val_acc: 0.7941
Epoch 46/128
18/18 [==============================] - 4s 224ms/step - loss: 0.1187 - acc: 0.9493 - val_loss: 0.
2555 - val_acc: 0.9412
Epoch 47/128
18/18 [==============================] - 5s 254ms/step - loss: 0.1943 - acc: 0.9601 - val_loss: 0.
4116 - val_acc: 0.8529
Epoch 48/128
18/18 [==============================] - 4s 216ms/step - loss: 0.1509 - acc: 0.9529 - val_loss: 0.
3326 - val_acc: 0.8529
Epoch 49/128
18/18 [==============================] - 5s 268ms/step - loss: 0.1915 - acc: 0.9312 - val_loss: 0.
3127 - val_acc: 0.8676
Epoch 50/128
18/18 [==============================] - 5s 282ms/step - loss: 0.1027 - acc: 0.9493 - val_loss: 0.
2460 - val_acc: 0.8529
Epoch 51/128
18/18 [==============================] - 5s 265ms/step - loss: 0.0940 - acc: 0.9493 - val_loss: 0.
3086 - val_acc: 0.8382
Epoch 52/128
18/18 [==============================] - 5s 260ms/step - loss: 0.1138 - acc: 0.9565 - val_loss: 0.
3039 - val_acc: 0.9118
Epoch 53/128
18/18 [==============================] - 4s 249ms/step - loss: 0.0399 - acc: 0.9819 - val_loss: 0.
0723 - val_acc: 0.9706
Epoch 54/128
18/18 [==============================] - 4s 239ms/step - loss: 0.1190 - acc: 0.9565 - val_loss: 0.
2820 - val_acc: 0.8676
Epoch 55/128
18/18 [==============================] - 4s 241ms/step - loss: 0.0471 - acc: 0.9783 - val_loss: 0.
1615 - val_acc: 0.9265
Epoch 56/128
18/18 [==============================] - 4s 236ms/step - loss: 0.1046 - acc: 0.9529 - val_loss: 0.
1850 - val_acc: 0.9265
Epoch 57/128
18/18 [==============================] - 5s 253ms/step - loss: 0.1044 - acc: 0.9457 - val_loss: 0.
1411 - val_acc: 0.9559
Epoch 58/128
18/18 [==============================] - 4s 223ms/step - loss: 0.1247 - acc: 0.9601 - val_loss: 0.
4292 - val_acc: 0.8824
Epoch 59/128
18/18 [==============================] - 4s 218ms/step - loss: 0.3241 - acc: 0.9275 - val_loss: 0.
3333 - val_acc: 0.8529
Epoch 60/128
18/18 [==============================] - 4s 202ms/step - loss: 0.1395 - acc: 0.9348 - val_loss: 0.
2235 - val_acc: 0.8971
Epoch 61/128
18/18 [==============================] - 4s 205ms/step - loss: 0.0361 - acc: 0.9855 - val_loss: 0.
1511 - val_acc: 0.9706
Epoch 62/128
18/18 [==============================] - 4s 248ms/step - loss: 0.1431 - acc: 0.9565 - val_loss: 0.
3559 - val_acc: 0.8824
Epoch 63/128
18/18 [==============================] - 6s 338ms/step - loss: 0.0929 - acc: 0.9601 - val_loss: 0.
3049 - val_acc: 0.9118
Epoch 64/128
18/18 [==============================] - 7s 374ms/step - loss: 0.0986 - acc: 0.9529 - val_loss: 0.
1689 - val_acc: 0.9265
Epoch 65/128
18/18 [==============================] - 5s 270ms/step - loss: 0.0957 - acc: 0.9710 - val_loss: 0.
```

```
1438 - val_acc: 0.9412
Epoch 66/128
18/18 [==============================] - 5s 288ms/step - loss: 0.0530 - acc: 0.9855 - val_loss: 0.
1999 - val_acc: 0.9412
Epoch 67/128
18/18 [==============================] - 5s 284ms/step - loss: 0.0931 - acc: 0.9710 - val_loss: 0.
3584 - val_acc: 0.9118
Epoch 68/128
18/18 [==============================] - 5s 258ms/step - loss: 0.1190 - acc: 0.9638 - val_loss: 0.
1231 - val_acc: 0.9559
Epoch 69/128
18/18 [==============================] - 5s 264ms/step - loss: 0.0859 - acc: 0.9746 - val_loss: 0.
4125 - val_acc: 0.8529
Epoch 70/128
18/18 [==============================] - 4s 245ms/step - loss: 0.0208 - acc: 0.9964 - val_loss: 0.
3010 - val_acc: 0.8824
Epoch 71/128
18/18 [==============================] - 4s 230ms/step - loss: 0.0537 - acc: 0.9783 - val_loss: 0.
0982 - val_acc: 0.9412
Epoch 72/128
18/18 [==============================] - 4s 248ms/step - loss: 0.0596 - acc: 0.9783 - val_loss: 0.
3688 - val_acc: 0.9118
Epoch 73/128
18/18 [==============================] - 4s 212ms/step - loss: 0.0605 - acc: 0.9783 - val_loss: 0.
3173 - val_acc: 0.9118
Epoch 74/128
18/18 [==============================] - 4s 211ms/step - loss: 0.0430 - acc: 0.9855 - val_loss: 0.
0753 - val_acc: 0.9559
Epoch 75/128
18/18 [==============================] - 5s 262ms/step - loss: 0.0405 - acc: 0.9891 - val_loss: 0.
1045 - val_acc: 0.9559
Epoch 76/128
18/18 [==============================] - 4s 232ms/step - loss: 0.0653 - acc: 0.9710 - val_loss: 0.
2016 - val_acc: 0.9412
Epoch 77/128
18/18 [==============================] - 4s 233ms/step - loss: 0.0519 - acc: 0.9783 - val_loss: 0.
0944 - val_acc: 0.9412
Epoch 78/128
18/18 [==============================] - 4s 225ms/step - loss: 0.0610 - acc: 0.9710 - val_loss: 1.
2028 - val_acc: 0.8382
Epoch 79/128
18/18 [==============================] - 4s 217ms/step - loss: 0.0439 - acc: 0.9819 - val_loss: 0.
4222 - val_acc: 0.8971
Epoch 80/128
18/18 [==============================] - 4s 204ms/step - loss: 0.1014 - acc: 0.9674 - val_loss: 0.
1878 - val_acc: 0.8824
Epoch 81/128
18/18 [==============================] - 4s 230ms/step - loss: 0.0256 - acc: 0.9855 - val_loss: 0.
3412 - val_acc: 0.8676
Epoch 82/128
18/18 [==============================] - 4s 234ms/step - loss: 0.0472 - acc: 0.9819 - val_loss: 0.
3348 - val_acc: 0.8676
Epoch 83/128
18/18 [==============================] - 4s 244ms/step - loss: 0.0619 - acc: 0.9783 - val_loss: 0.
3320 - val_acc: 0.8824
Epoch 84/128
18/18 [==============================] - 5s 254ms/step - loss: 0.0878 - acc: 0.9710 - val_loss: 0.
2944 - val_acc: 0.9118
Epoch 85/128
18/18 [==============================] - 5s 258ms/step - loss: 0.0875 - acc: 0.9674 - val_loss: 0.
5009 - val_acc: 0.8088
Epoch 86/128
18/18 [==============================] - 4s 225ms/step - loss: 0.0352 - acc: 0.9928 - val_loss: 0.
2021 - val_acc: 0.8824
Epoch 87/128
18/18 [==============================] - 4s 217ms/step - loss: 0.0697 - acc: 0.9746 - val_loss: 0.
0615 - val_acc: 0.9706
Epoch 88/128
18/18 [==============================] - 4s 225ms/step - loss: 0.0936 - acc: 0.9746 - val_loss: 0.
1056 - val_acc: 0.9412
Epoch 89/128
18/18 [==============================] - 4s 230ms/step - loss: 0.0741 - acc: 0.9746 - val_loss: 0.
0311 - val_acc: 1.0000
Epoch 90/128
18/18 [==============================] - 4s 247ms/step - loss: 0.0367 - acc: 0.9891 - val_loss: 0.
4966 - val_acc: 0.9265
Epoch 91/128
```

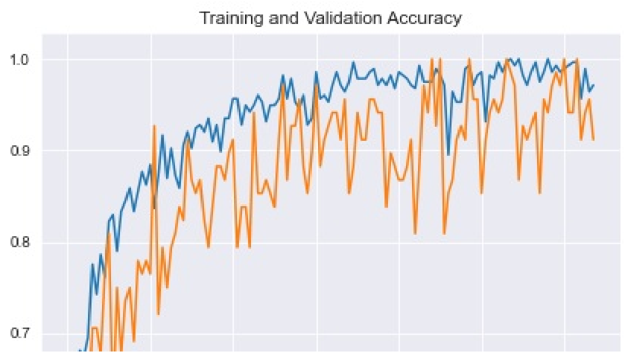
```
18/18 [==============================] - 4s 246ms/step - loss: 0.0489 - acc: 0.9819 - val_loss: 0.
0756 - val_acc: 1.0000
Epoch 92/128
18/18 [==============================] - 4s 239ms/step - loss: 0.1348 - acc: 0.9710 - val_loss: 0.
7330 - val_acc: 0.8088
Epoch 93/128
18/18 [==============================] - 4s 228ms/step - loss: 0.3086 - acc: 0.8949 - val_loss: 0.
3685 - val_acc: 0.8529
Epoch 94/128
18/18 [==============================] - 4s 233ms/step - loss: 0.0798 - acc: 0.9638 - val_loss: 0.
1904 - val_acc: 0.8676
Epoch 95/128
18/18 [==============================] - 4s 224ms/step - loss: 0.0718 - acc: 0.9529 - val_loss: 0.
2180 - val_acc: 0.9118
Epoch 96/128
18/18 [==============================] - 4s 232ms/step - loss: 0.1515 - acc: 0.9529 - val_loss: 0.
1151 - val_acc: 0.9265
Epoch 97/128
18/18 [==============================] - 4s 229ms/step - loss: 0.0351 - acc: 0.9891 - val_loss: 0.
2085 - val_acc: 0.9118
Epoch 98/128
18/18 [==============================] - 4s 214ms/step - loss: 0.0434 - acc: 0.9928 - val_loss: 0.
0404 - val_acc: 1.0000
Epoch 99/128
18/18 [==============================] - 4s 242ms/step - loss: 0.0711 - acc: 0.9710 - val_loss: 0.
0912 - val_acc: 0.9559
Epoch 100/128
18/18 [==============================] - 6s 308ms/step - loss: 0.0593 - acc: 0.9819 - val_loss: 0.
0763 - val_acc: 0.9559
Epoch 101/128
18/18 [==============================] - 4s 244ms/step - loss: 0.0622 - acc: 0.9855 - val_loss: 0.
4754 - val_acc: 0.8529
Epoch 102/128
18/18 [==============================] - 4s 224ms/step - loss: 0.1572 - acc: 0.9312 - val_loss: 0.
6159 - val_acc: 0.9118
Epoch 103/128
18/18 [==============================] - 4s 209ms/step - loss: 0.0404 - acc: 0.9819 - val_loss: 0.
0944 - val_acc: 0.9412
Epoch 104/128
18/18 [==============================] - 4s 217ms/step - loss: 0.0504 - acc: 0.9783 - val_loss: 0.
0644 - val_acc: 0.9559
Epoch 105/128
18/18 [==============================] - 4s 239ms/step - loss: 0.0126 - acc: 0.9964 - val_loss: 0.
1344 - val_acc: 0.9412
Epoch 106/128
18/18 [==============================] - 4s 240ms/step - loss: 0.0272 - acc: 0.9855 - val_loss: 0.
1435 - val_acc: 0.9559
Epoch 107/128
18/18 [==============================] - 5s 289ms/step - loss: 0.0138 - acc: 0.9964 - val_loss: 0.
0049 - val_acc: 1.0000
Epoch 108/128
18/18 [==============================] - 6s 343ms/step - loss: 0.0091 - acc: 1.0000 - val_loss: 0.
0972 - val_acc: 0.9853
Epoch 109/128
18/18 [==============================] - 5s 251ms/step - loss: 0.0214 - acc: 0.9928 - val_loss: 0.
1060 - val_acc: 0.9706
Epoch 110/128
18/18 [==============================] - 4s 244ms/step - loss: 0.0070 - acc: 1.0000 - val_loss: 0.
4544 - val_acc: 0.8676
Epoch 111/128
18/18 [==============================] - 4s 219ms/step - loss: 0.0455 - acc: 0.9819 - val_loss: 0.
2695 - val_acc: 0.9265
Epoch 112/128
18/18 [==============================] - 4s 231ms/step - loss: 0.0509 - acc: 0.9710 - val_loss: 0.
2578 - val_acc: 0.9118
Epoch 113/128
18/18 [==============================] - 4s 222ms/step - loss: 0.0453 - acc: 0.9855 - val_loss: 0.
2596 - val_acc: 0.9265
Epoch 114/128
18/18 [==============================] - 4s 224ms/step - loss: 0.0259 - acc: 0.9964 - val_loss: 0.
4961 - val_acc: 0.9412
Epoch 115/128
18/18 [==============================] - 4s 215ms/step - loss: 0.0823 - acc: 0.9746 - val_loss: 0.
8257 - val_acc: 0.8529
Epoch 116/128
18/18 [==============================] - 4s 218ms/step - loss: 0.0380 - acc: 0.9855 - val_loss: 0.
2097 - val_acc: 0.9559
```

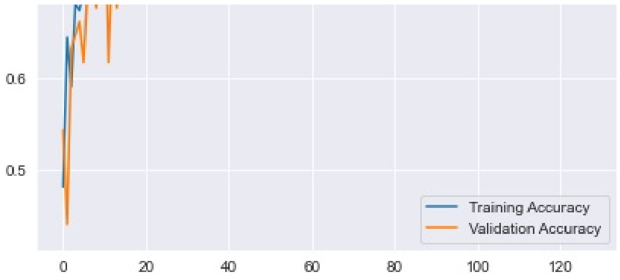
```
Epoch 117/128
18/18 [==============================] - 4s 233ms/step - loss: 0.0104 - acc: 1.0000 - val_loss: 0.
0921 - val_acc: 0.9412
Epoch 118/128
18/18 [==============================] - 4s 217ms/step - loss: 0.0580 - acc: 0.9855 - val_loss: 0.
1572 - val_acc: 0.9706
Epoch 119/128
18/18 [==============================] - 4s 214ms/step - loss: 0.0222 - acc: 0.9928 - val_loss: 0.
0288 - val_acc: 0.9853
Epoch 120/128
18/18 [==============================] - 4s 228ms/step - loss: 0.0401 - acc: 0.9855 - val_loss: 0.
0566 - val_acc: 0.9706
Epoch 121/128
18/18 [==============================] - 4s 244ms/step - loss: 0.0278 - acc: 0.9891 - val_loss: 0.
0240 - val_acc: 1.0000
Epoch 122/128
18/18 [==============================] - 4s 247ms/step - loss: 0.0390 - acc: 0.9928 - val_loss: 0.
1742 - val_acc: 0.9412
Epoch 123/128
18/18 [==============================] - 5s 303ms/step - loss: 0.0150 - acc: 0.9964 - val_loss: 0.
1041 - val_acc: 0.9412
Epoch 124/128
18/18 [==============================] - 5s 264ms/step - loss: 0.0285 - acc: 0.9964 - val_loss: 0.
0148 - val_acc: 1.0000
Epoch 125/128
18/18 [==============================] - 5s 269ms/step - loss: 0.2387 - acc: 0.9565 - val_loss: 0.
3517 - val_acc: 0.9118
Epoch 126/128
18/18 [==============================] - 4s 248ms/step - loss: 0.0392 - acc: 0.9891 - val_loss: 0.
1234 - val_acc: 0.9412
Epoch 127/128
18/18 [==============================] - 5s 268ms/step - loss: 0.0797 - acc: 0.9638 - val_loss: 0.
0792 - val_acc: 0.9559
Epoch 128/128
18/18 [==============================] - 5s 272ms/step - loss: 0.1106 - acc: 0.9710 - val_loss: 0.
1573 - val_acc: 0.9118
```

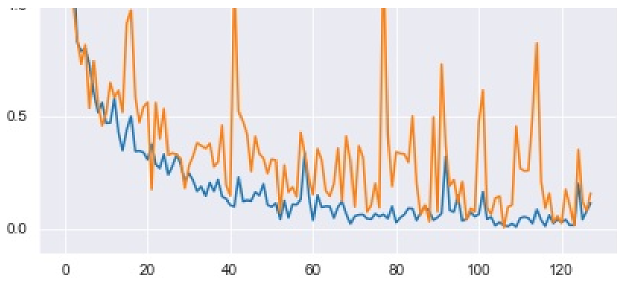
Model Evaluation

In [24]:

```python
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(len(acc))

plt.figure(figsize=(15, 15))
plt.subplot(2, 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(2, 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 Validation Loss')
plt.show()
```







In [25]:

```
true_classes = test_dataset.classes
class_indices = train_dataset.class_indices
class_indices = dict((v,k) for k,v in class_indices.items())

vgg_preds = bear_foot_model.predict(test_dataset)
vgg_pred_classes = np.argmax(vgg_preds, axis=1)

from sklearn.metrics import accuracy_score

vgg_acc = accuracy_score(true_classes, vgg_pred_classes)
print("Bear Footprint Classification using VGG16: {:.2f}%".format(vgg_acc * 100))
```

```
Bear Footprint Classification using VGG16: 98.84%
```

Image Predictions

In [26]:

```
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input

def predict_bear_footprint(img_path):
    img = image.load_img(img_path, target_size=(224,224))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    x = preprocess_input(x)
    plt.imshow(img)

    print(bear_foot_model.predict(x))

    for x in bear_foot_model.predict(x):
        for i in range(len(x)):
            if(i==0):
                print("Black Bear Confidence: ", end = " ")
            else:
                print("Grizzly Bear Confidence: ", end = " ")
            print("{0:.2%}".format(x[i]))
```

In [27]:

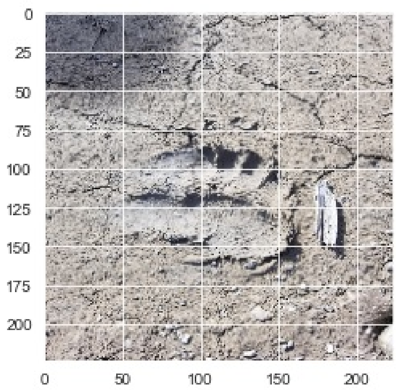
```
predict_bear_footprint('black/aug_9_1684.jpg')
```

```
[[1.0000000e+00 1.6503463e-07]]
Black Bear Confidence:  100.00%
Grizzly Bear Confidence:  0.00%
```

In [28]:

```
predict_bear_footprint('grizzly/large (25).jpg')
```

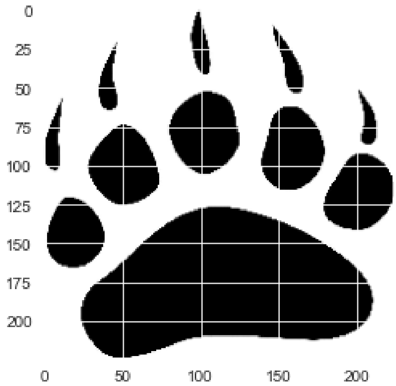
```
[[0. 1.]]
Black Bear Confidence:  0.00%
Grizzly Bear Confidence:  100.00%
```



In [29]:

```
predict_bear_footprint('black_google.jpg')
```

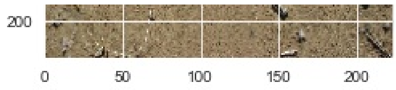
```
[[9.997819e-01 1.670952e-06]]
Black Bear Confidence:  99.98%
Grizzly Bear Confidence:  0.00%
```



In [30]:

```
predict_bear_footprint('black_google_2.jpg')
```

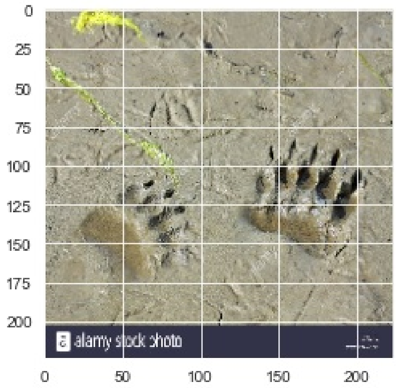
```
[[2.3745839e-04 9.8778188e-01]]
Black Bear Confidence:  0.02%
Grizzly Bear Confidence:  98.78%
```



In [31]:

```
predict_bear_footprint('grizzly_google.jpg')
```

```
[[1.8583019e-05 9.9994373e-01]]
Black Bear Confidence:  0.00%
Grizzly Bear Confidence:  99.99%
```



In [32]:

```
predict_bear_footprint('grizzly_google_2.jpg')
```

```
[[4.804680e-07 9.999999e-01]]
Black Bear Confidence:  0.00%
Grizzly Bear Confidence:  100.00%
```