IMPORT LIBRARIES

In [7]:
```
import os
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import tensorflow as tf
from tensorflow.keras.layers import Dense, Input
from tensorflow.keras import Sequential
from tensorflow.keras.losses import MeanSquaredLogarithmicError
from tensorflow import keras as k
import tensorflow as tf
import plotly.express as px
import random
import cv2
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import Dropout
from sklearn.metrics import confusion_matrix, classification_report
```

READ THE DATA

In [156]:
```
ls
```

```
 Volume in drive C is Windows
 Volume Serial Number is BC27-F91B

 Directory of C:\Users\757538\OneDrive - hull.ac.uk\Data

24/08/2023  16:34    <DIR>          .
24/08/2023  16:34    <DIR>          ..
24/08/2023  12:19    <DIR>          .ipynb_checkpoints
24/08/2023  16:34         4,431,191 IMAGE ANALYSIS 2.ipynb
24/08/2023  13:51       284,742,226 IMAGE ANALYSIS.ipynb
23/08/2023  13:24            (35,545) MSR-LA - 3467.docx
24/08/2023  12:59    <DIR>          PetImages
23/08/2023  13:28               (104) readme[1].txt
23/08/2023  17:17            (12,370) Untitled1.ipynb
               5 File(s)    289,221,436 bytes
               4 Dir(s)  346,828,148,736 bytes free
```

In [2]:
```
fig=plt.figure(figsize=(13, 6))
    # Define row and cols in the figure
    rows, cols = 4, 4
    nm=list(plt.cm.datad.keys())
    arr=np.random.randint(0,10000,(rows*cols,))
    c=np.random.choice(['Cat','Dog'],(rows*cols,))
    # Display first four images
    i=0
    for j in range(0, cols*rows):
      fig.add_subplot(rows, cols, j+1)
      plt.title(c[i])
      plt.imshow(plt.imread(f'PetImages/{c[i]}/{arr[i]}.jpg'))
      i+=1
      plt.axis('off')
```

```
plt.show()
```

DATA EXPLORATION

ANIMAL CLASS DIDTRIBUTION IN THE DATASET

In [8]:
```
class_names = ['Cat', 'Dog']

n_dogs = len(os.listdir('PetImages/Dog'))
n_cats = len(os.listdir('PetImages/Cat'))
n_images = [n_cats, n_dogs]
px.pie(names=class_names, values=n_images)
```

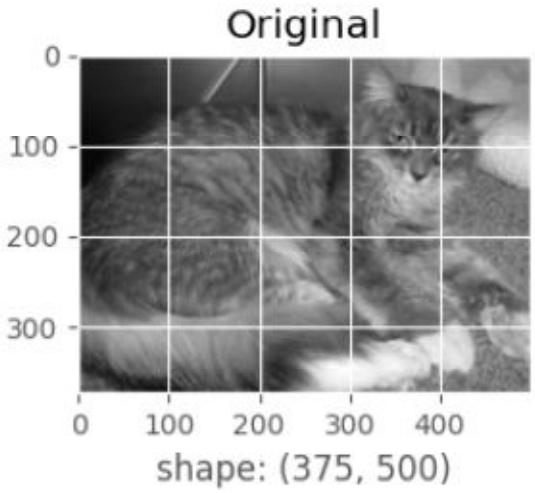
SETTING PARAMETERS

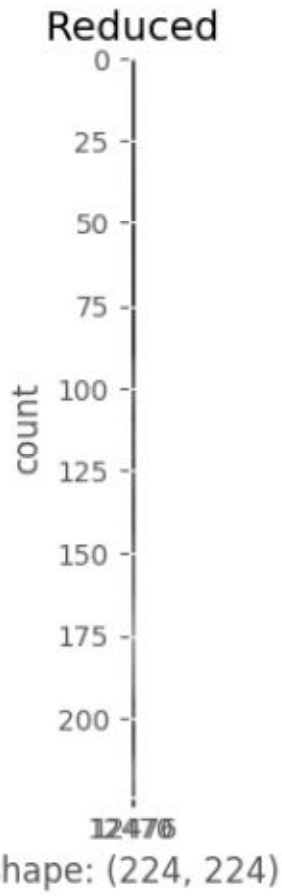
In [149]:
```
DIRECTORY = 'PetImages'
CATAGORY = ['Cat', 'Dog']
IMG_SIZE = 224
arr = cv2.imread(os.path.join(DIRECTORY,CATAGORY[0],'11590.jpg'), cv2.IMRE
plt.subplot(1,2,1)
plt.imshow(arr, cmap='gray')
plt.title('Original')
plt.xlabel('shape: {}'.format(arr.shape))
plt.subplot(1,2,2)

arr = cv2.resize(arr, (IMG_SIZE,IMG_SIZE))
plt.imshow(arr, cmap='gray')
plt.title('Reduced')
plt.xlabel('shape: {}'.format(arr.shape))
training_data=[]
class_list =[]

# there are some broken imgs(corrupted)
# we will ignore/pass them as we encounter them through the try except block
def create_training_data():
```

```
    for category in CATAGORY:
        path = os.path.join(DIRECTORY,category)
        class_num = CATAGORY.index(category)
        for img in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path,img), cv2.IMREAD_
                img_array = cv2.resize(img_array, (IMG_SIZE,IMG_SIZE))
                training_data.append([img_array, class_num])
                class_list.append(class_num)
            except Exception as e:
                pass

create_training_data()
print(len(training_data))
sns.countplot(pd.Series(class_list).value_counts())
random.shuffle(training_data)
for sample in training_data[:10]:
    print(sample[1])
X = []
y = []

for features, labels in training_data:
    X.append(features)
    y.append(labels)

    # (number of images, height, width, num of channels)
X = np.array(X)#.reshape(IMG_SIZE, IMG_SIZE, 3)
y = np.array(y)

# normalising the data
```

```
24946
1
1
0
0
0
0
1
0
1
0
```

```
C:\Program Files\Python310\lib\site-packages\seaborn\_decorators.py:36: FutureW
arning:

Pass the following variable as a keyword arg: x. From version 0.12, the only va
lid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
```





MODELLING

In [150]:
```
X_train, X_test ,y_train,y_test = train_test_split(X, y, test_size=0.20, r
```

In [ ]:

In [157]:
```
import os
```

```
import shutil
import random


DIRECTORY = 'PetImages'
CATEGORIES = ['Cat', 'Dog']
TRAIN_RATIO = 0.8
TEST_RATIO = 0.2


# Ensure train and test directories exist
if not os.path.exists(os.path.join(DIRECTORY, 'train')):
    os.makedirs(os.path.join(DIRECTORY, 'train'))
if not os.path.exists(os.path.join(DIRECTORY, 'test')):
    os.makedirs(os.path.join(DIRECTORY, 'test'))


for category in CATEGORIES:
    path = os.path.join(DIRECTORY, category)
    images = os.listdir(path)


    # Shuffle the list of images to ensure randomness
    random.shuffle(images)


    # Calculate the number of images for training and testing
    train_size = int(TRAIN_RATIO * len(images))
    train_images = images[:train_size]
    test_images = images[train_size:]


    # Ensure category directory exists in train and test directories
    if not os.path.exists(os.path.join(DIRECTORY, 'train', category)):
        os.makedirs(os.path.join(DIRECTORY, 'train', category))
    if not os.path.exists(os.path.join(DIRECTORY, 'test', category)):
        os.makedirs(os.path.join(DIRECTORY, 'test', category))


    # Move the images
    for img in train_images:
        shutil.move(os.path.join(path, img), os.path.join(DIRECTORY, 'trai
    for img in test_images:
        shutil.move(os.path.join(path, img), os.path.join(DIRECTORY, 'test
```


In [163]: pip install scikit-image

```
Defaulting to user installation because normal site-packages is not writeable
Collecting scikit-image
  Downloading scikit_image-0.21.0-cp310-cp310-win_amd64.whl (22.8 MB)
     ---------------------------------------- 22.8/22.8 MB 92.9 MB/s eta 0:00:00
Requirement already satisfied: scipy>=1.8 in c:\program files\python310\lib\sit
e-packages (from scikit-image) (1.9.1)
Collecting networkx>=2.8
  Downloading networkx-3.1-py3-none-any.whl (2.1 MB)
     ---------------------------------------- 2.1/2.1 MB 66.5 MB/s eta 0:00:00
Collecting imageio>=2.27
  Downloading imageio-2.31.1-py3-none-any.whl (313 kB)
     ---------------------------------------- 313.2/313.2 kB ? eta 0:00:00
Collecting PyWavelets>=1.1.1
  Downloading PyWavelets-1.4.1-cp310-cp310-win_amd64.whl (4.2 MB)
     ---------------------------------------- 4.2/4.2 MB 67.0 MB/s eta 0:00:00
Requirement already satisfied: pillow>=9.0.1 in c:\program files\python310\lib\
site-packages (from scikit-image) (9.2.0)
Collecting tifffile>=2022.8.12
  Downloading tifffile-2023.8.12-py3-none-any.whl (220 kB)
     ---------------------------------------- 221.0/221.0 kB ? eta 0:00:00
Collecting lazy_loader>=0.2
  Downloading lazy_loader-0.3-py3-none-any.whl (9.1 kB)
Requirement already satisfied: packaging>=21 in c:\program files\python310\lib\
site-packages (from scikit-image) (21.3)
Requirement already satisfied: numpy>=1.21.1 in c:\program files\python310\lib\
site-packages (from scikit-image) (1.23.2)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\program files\pyt
hon310\lib\site-packages (from packaging>=21->scikit-image) (3.0.9)
Installing collected packages: tifffile, PyWavelets, networkx, lazy_loader, ima
geio, scikit-image
Successfully installed PyWavelets-1.4.1 imageio-2.31.1 lazy_loader-0.3 network
x-3.1 scikit-image-0.21.0 tifffile-2023.8.12
Note: you may need to restart the kernel to use updated packages.

  WARNING: The scripts lsm2bin.exe, tiff2fsspec.exe, tiffcomment.exe and tifffi
le.exe are installed in 'C:\Users\757538\AppData\Roaming\Python\Python310\Scrip
ts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this war
ning, use --no-warn-script-location.
  WARNING: The scripts imageio_download_bin.exe and imageio_remove_bin.exe are
installed in 'C:\Users\757538\AppData\Roaming\Python\Python310\Scripts' which i
s not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this war
ning, use --no-warn-script-location.

[notice] A new release of pip available: 22.2.2 -> 23.2.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [165]:
```
import os
import cv2
```

```
import numpy as np

from tqdm import tqdm

import skimage.transform
```

Splitting the data

In [166]:

```
DIRECTORY = 'PetImages'

CATEGORIES = ['Cat', 'Dog']  # changed CATAGORY to CATEGORIES for naming c

IMG_SIZE = 224


LOAD_FROM_IMAGES = True


def get_data(folder):

    x = []

    y = []

    for folderName in os.listdir(folder):

        if not folderName.startswith("."):

            if folderName == "Cat":

                label = 0

            elif folderName == "Dog":

                label = 1

            else:

                continue  # If the folder is neither Cat nor Dog, we skip


            for image_filename in tqdm(os.listdir(os.path.join(folder, fol

                try:

                    img_file = cv2.imread(os.path.join(folder, folderName,

                    if img_file is not None:
```

```
                img_file = skimage.transform.resize(img_file, (IMG
                img_arr = np.asarray(img_file)
                x.append(img_arr)
                y.append(label)
            except Exception as e:
                pass

    x = np.asarray(x)
    y = np.asarray(y)
    return x, y

if LOAD_FROM_IMAGES:
    X_train, y_train = get_data(os.path.join(DIRECTORY, 'train'))  # Assum
    X_test, y_test = get_data(os.path.join(DIRECTORY, 'test'))

    np.save("xtrain.npy", X_train)
    np.save("ytrain.npy", y_train)
    np.save("xtest.npy", X_test)
    np.save("ytest.npy", y_test)
else:
    X_train = np.load("xtrain.npy")
    y_train = np.load("ytrain.npy")
    X_test = np.load("xtest.npy")
    y_test = np.load("ytest.npy")
```

```
100%|████████████████████████████████████████████████████████████████
██| 10000/10000 [05:56<00:00, 28.08it/s]
100%|████████████████████████████████████████████████████████████████
██| 10000/10000 [05:48<00:00, 28.67it/s]
```

```
---------------------------------------------------------------------------
MemoryError                               Traceback (most recent call last)
Input In [166], in <cell line: 67>()
     63     return x, y
     67 if LOAD_FROM_IMAGES:
---> 69     X_train, y_train = get_data(os.path.join(DIRECTORY, 'train'))  # As
suming you have train and test folders under PetImages
     71     X_test, y_test = get_data(os.path.join(DIRECTORY, 'test'))
     75     np.save("xtrain.npy", X_train)

Input In [166], in get_data(folder)
     53             except Exception as e:
     55                 pass
---> 59 x = np.asarray(x)
     61 y = np.asarray(y)
     63 return x, y

MemoryError: Unable to allocate 22.4 GiB for an array with shape (19957, 224, 2
24, 3) and data type float64
```

In [ ]:

CNN Model

In [11]:
```
model=Sequential([k.layers.BatchNormalization(),k.layers.Conv2D(18,(3,3),ac
model.add(k.layers.BatchNormalization())
model.add(Dropout(0.25))

model.add(k.layers.Conv2D(32,(3,3),activation='relu'))
model.add(k.layers.BatchNormalization())

model.add(k.layers.MaxPooling2D((2,2)))
model.add(Dropout(0.25))

model.add(k.layers.Conv2D(64,(3,3),activation='relu'))
model.add(k.layers.BatchNormalization())

model.add(k.layers.MaxPooling2D((2,2)))
model.add(Dropout(0.25))

model.add(k.layers.Conv2D(128,(3,3),activation='relu'))

model.add(k.layers.BatchNormalization())
model.add(k.layers.MaxPooling2D((2,2)))
model.add(Dropout(0.25))

model.add(k.layers.Conv2D(256,(3,3),activation='relu'))
model.add(k.layers.BatchNormalization())
model.add(k.layers.MaxPooling2D())
```

```
model.add(Dropout(0.25))

model.add(k.layers.Flatten())
model.add(k.layers.Dense(256,activation='elu',kernel_initializer='he_normal
model.add(k.layers.BatchNormalization())
model.add(Dropout(0.25))

model.add(k.layers.Dense(512,activation='elu',kernel_initializer='he_normal
model.add(k.layers.BatchNormalization())
model.add(Dropout(0.25))

model.add(k.layers.Dense(1,activation='sigmoid'))
model.compile(loss=k.losses.BinaryCrossentropy(from_logits=True),optimizer=
```

Fitting the Model

In [12]:
```
hist=model.fit(X_train,y_train,epochs=10,batch_size=32,validation_split=.1,
```

```
Epoch 1/10

C:\Program Files\Python310\lib\site-packages\tensorflow\python\util\dispatch.p
y:1082: UserWarning:

"`binary_crossentropy` received `from_logits=True`, but the `output` argument w
as produced by a sigmoid or softmax activation and thus does not represent logi
ts. Was this intended?"


562/562 [==============================] - 64s 52ms/step - loss: 4.9502 - accur
acy: 0.5795 - val_loss: 1.2467 - val_accuracy: 0.5476
Epoch 2/10
562/562 [==============================] - 12s 22ms/step - loss: 0.7862 - accur
acy: 0.6668 - val_loss: 0.6472 - val_accuracy: 0.6688
Epoch 3/10
562/562 [==============================] - 12s 22ms/step - loss: 0.5697 - accur
acy: 0.7379 - val_loss: 0.5676 - val_accuracy: 0.7164
Epoch 4/10
562/562 [==============================] - 12s 22ms/step - loss: 0.5095 - accur
acy: 0.7734 - val_loss: 0.6545 - val_accuracy: 0.7260
Epoch 5/10
562/562 [==============================] - 12s 22ms/step - loss: 0.4759 - accur
acy: 0.7948 - val_loss: 0.4548 - val_accuracy: 0.7961
Epoch 6/10
562/562 [==============================] - 12s 22ms/step - loss: 0.4550 - accur
acy: 0.8089 - val_loss: 0.5169 - val_accuracy: 0.7921
Epoch 7/10
562/562 [==============================] - 12s 22ms/step - loss: 0.4372 - accur
acy: 0.8220 - val_loss: 0.4535 - val_accuracy: 0.8101
Epoch 8/10
562/562 [==============================] - 13s 22ms/step - loss: 0.4193 - accur
acy: 0.8307 - val_loss: 0.4770 - val_accuracy: 0.8066
Epoch 9/10
562/562 [==============================] - 12s 22ms/step - loss: 0.4064 - accur
acy: 0.8394 - val_loss: 0.5809 - val_accuracy: 0.7485
Epoch 10/10
562/562 [==============================] - 13s 22ms/step - loss: 0.3912 - accur
acy: 0.8492 - val_loss: 0.5026 - val_accuracy: 0.7816
```

In [ ]:

GENERATING MODEL SUMMARY

In [13]:
```
model.summary()
```

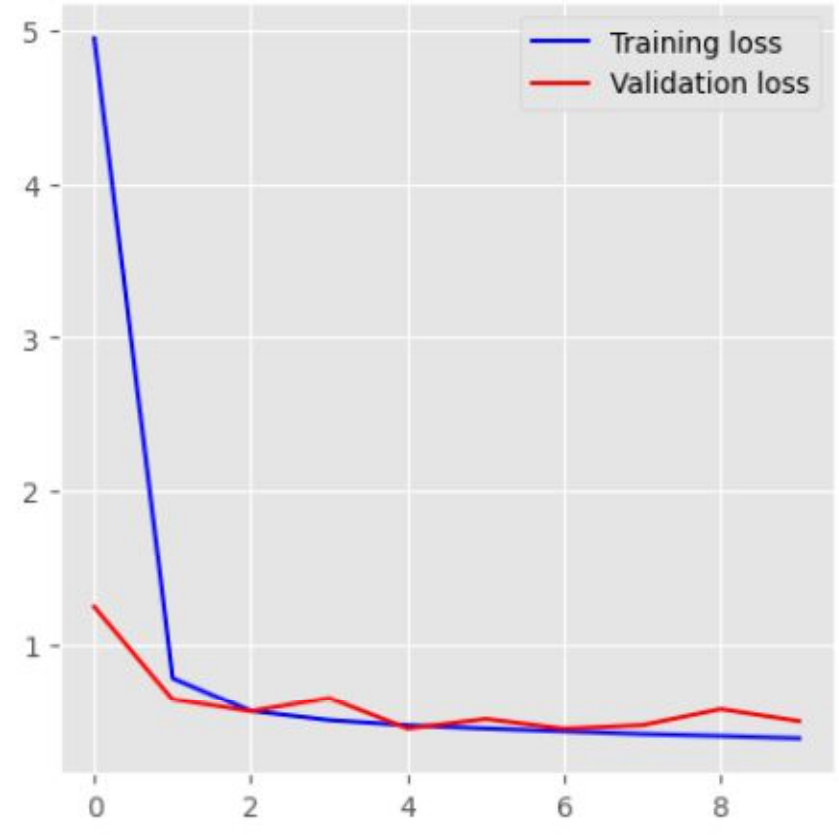
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 batch_normalization (BatchN  (None, 60, 60, 1)        4
 ormalization)

 conv2d (Conv2D)             (None, 58, 58, 18)        180

 batch_normalization_1 (Batc  (None, 58, 58, 18)       72
 hNormalization)

 dropout (Dropout)           (None, 58, 58, 18)        0

 conv2d_1 (Conv2D)           (None, 56, 56, 32)        5216

 batch_normalization_2 (Batc  (None, 56, 56, 32)       128
 hNormalization)

 max_pooling2d (MaxPooling2D  (None, 28, 28, 32)       0
 )

 dropout_1 (Dropout)         (None, 28, 28, 32)        0

 conv2d_2 (Conv2D)           (None, 26, 26, 64)        18496

 batch_normalization_3 (Batc  (None, 26, 26, 64)       256
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 13, 13, 64)       0
 2D)

 dropout_2 (Dropout)         (None, 13, 13, 64)        0

 conv2d_3 (Conv2D)           (None, 11, 11, 128)       73856

 batch_normalization_4 (Batc  (None, 11, 11, 128)      512
 hNormalization)

 max_pooling2d_2 (MaxPooling  (None, 5, 5, 128)        0
 2D)

 dropout_3 (Dropout)         (None, 5, 5, 128)         0

 conv2d_4 (Conv2D)           (None, 3, 3, 256)         295168

 batch_normalization_5 (Batc  (None, 3, 3, 256)        1024
 hNormalization)

 max_pooling2d_3 (MaxPooling  (None, 1, 1, 256)        0
 2D)
```

Visualize the Learning Process

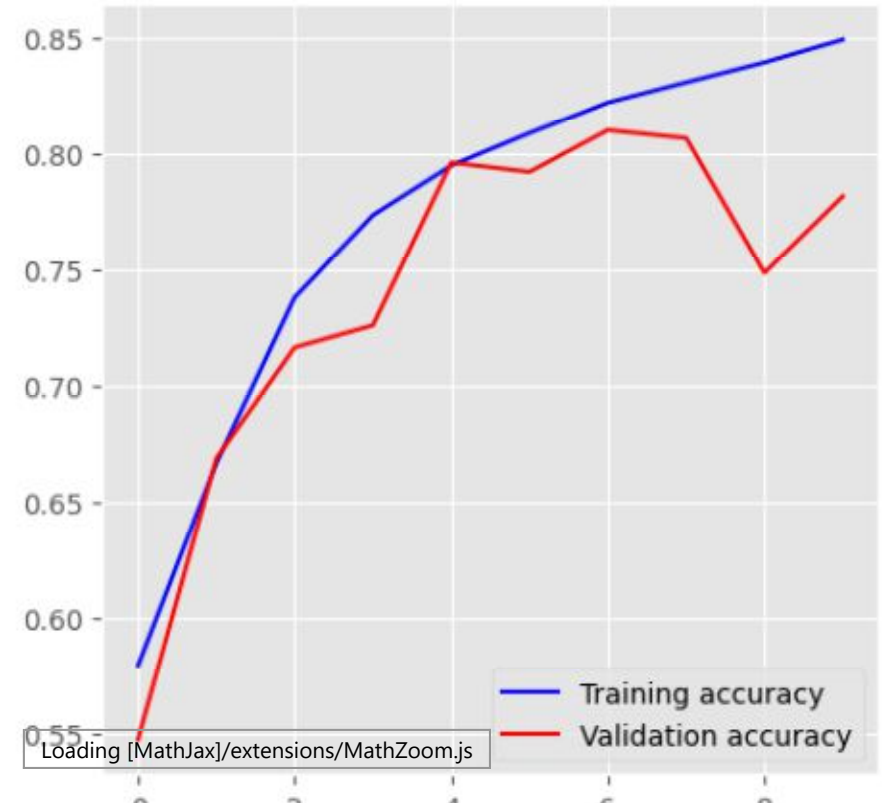
Model Evaluation

In [ ]:
```
import matplotlib.pyplot as plt
```

In [16]:
```
plt.figure(figsize=(5,5))
    plt.style.use("ggplot")
    plt.plot(hist.history['loss'], color='b', label="Training loss")
    plt.plot(hist.history['val_loss'], color='r', label="Validation loss")
    plt.legend()
    plt.show()

    plt.figure()

    plt.figure(figsize=(5,5))
    plt.style.use("ggplot")
    plt.plot(hist.history['accuracy'], color='b', label="Training accuracy")
    plt.plot(hist.history['val_accuracy'], color='r',label="Validation accuracy
    plt.legend(loc = "lower right")
    plt.show()
```




```
<Figure size 640x480 with 0 Axes>
```



Evaluating the Model on Test Data

In [15]:
```
model.evaluate (X_test, y_test)
```

```
156/156 [==============================] - 1s 5ms/step - loss: 0.4827 - accurac
y: 0.7848
```

Out[15]:

```
[0.48270389437675476, 0.7847695350646973]
```

Predictions with Test data

In [22]:
```
y_pred=model.predict(X_test)
```

```
156/156 [==============================] - 1s 4ms/step
```

In [47]:
```
print(X_train.shape)
print(X_test.shape)
```

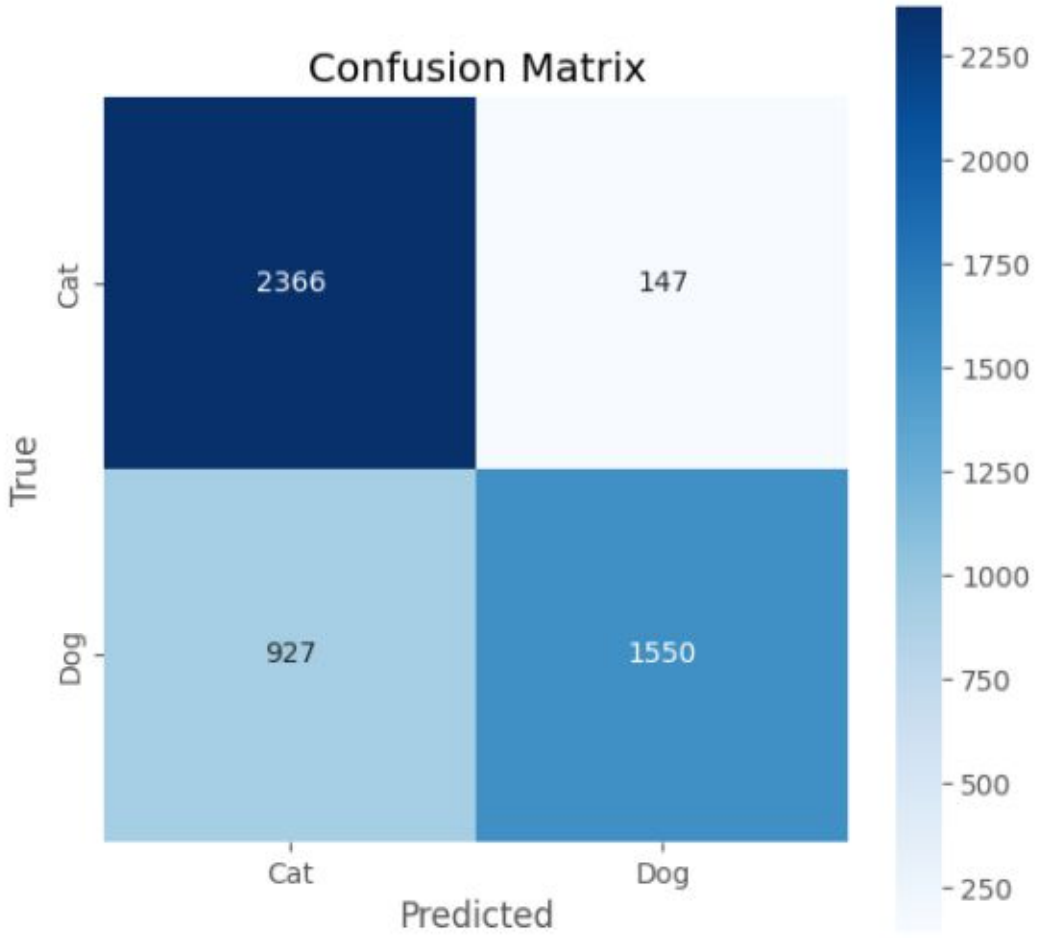
```
(19956, 60, 60, 1)
(4990, 60, 60, 1)
```

In [50]:
```
print(y_train.shape)
print(y_test.shape)
```

```
(19956,)
(4990,)
```

CONFUSION MATRIX

In [57]:
```
# Convert predicted probabilities to class labels (0 or 1)
y_pred_labels = np.round(y_pred).flatten()

# Create confusion matrix
confusion = confusion_matrix(y_test, y_pred_labels)

# Display the confusion matrix as a heatmap
plt.figure(figsize=(6, 6))
sns.heatmap(confusion, annot=True, fmt="d", cmap="Blues", square=True,
            xticklabels=['Cat', 'Dog'],
            yticklabels=['Cat', 'Dog'])
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix")
plt.show()

# Print classification report
print('Classification Report')
print(classification_report(y_test, y_pred_labels, target_names=['Cat', 'Do
```




```
Classification Report
              precision    recall  f1-score   support

         Cat       0.72      0.94      0.82      2513
         Dog       0.91      0.63      0.74      2477

    accuracy                           0.78      4990
   macro avg       0.82      0.78      0.78      4990
weighted avg       0.82      0.78      0.78      4990
```

PRE-TRAINED MODELS

RESNET50

In [92]:
```
import tensorflow as tf
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.applications.resnet50 import preprocess_input, decode
from tensorflow.keras.preprocessing import image
import numpy as np
```

In [151]:
```
import tensorflow as tf
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.models import Model
from tensorflow.keras import layers
```

Loading [MathJax]/extensions/MathZoom.js

```
      # Create a ResNet-50 model with pre-trained weights (optional)
      resnet_base = ResNet50(include_top=False, weights='imagenet', input_shape=

      resnet_base.trainable = False

      model = models.Sequential()

      model.add(resnet_base)

      model.add(layers.Dense(128, activation='relu'))
      model.add(layers.Dense(2, activation='softmax'))
```

In [152]:
```
# Print the model summary
model.summary()
```

```
Model: "sequential_17"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 7, 7, 2048)        23587712

 dense_49 (Dense)            (None, 7, 7, 128)         262272

 dense_50 (Dense)            (None, 7, 7, 2)           258

=================================================================
Total params: 23,850,242
Trainable params: 262,530
Non-trainable params: 23,587,712
_________________________________________________________________
```

In [153]:
```
X_train.shape
```

Out[153]:
```
(19956, 224, 224)
```

In [154]:
```
# Compile the model with appropriate loss, optimizer, and metrics
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=[
```

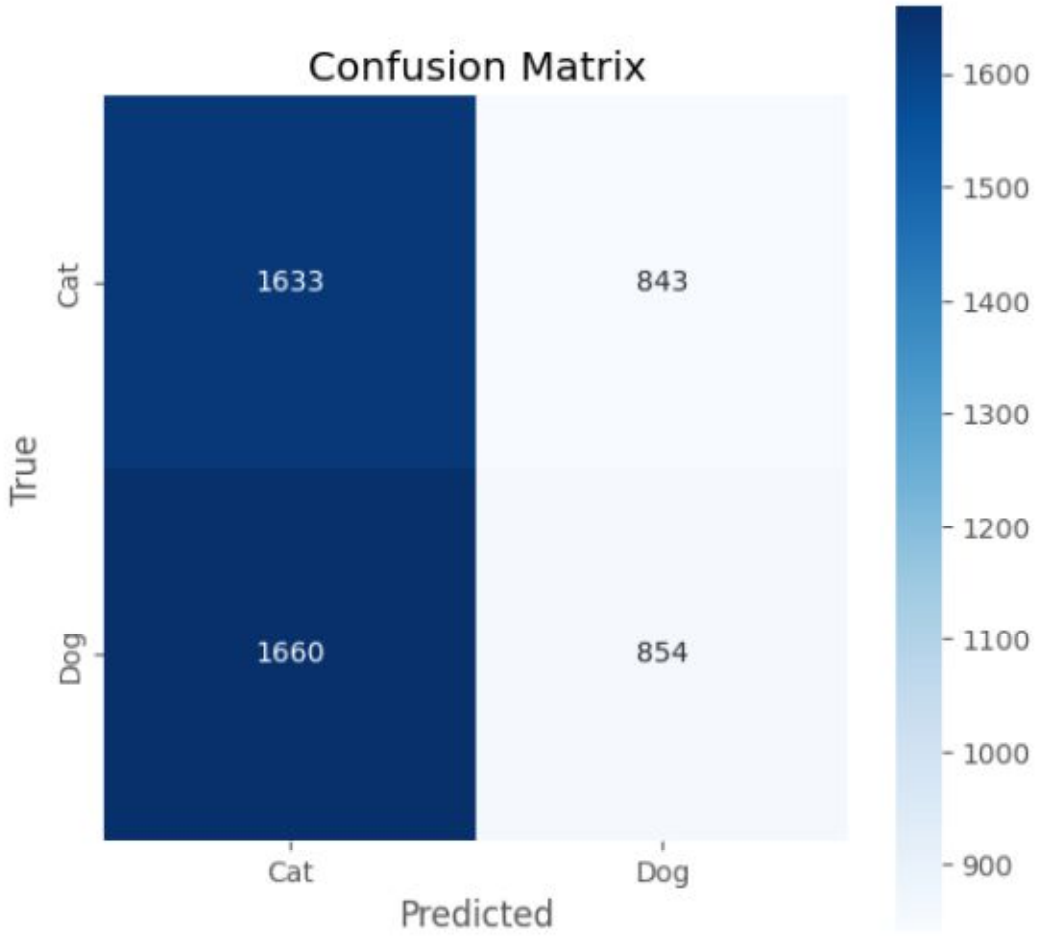
In [ ]:

In [ ]:
```
# Train the model and capture the training history
num_epochs = 10
history = model.fit(
    X_train,y_train,
    epochs=num_epochs)

# Access the training history
print(history.history)
```

In [ ]:

In [204]:
```
# Convert predicted probabilities to class labels (0 or 1)
resnet_base = np.round(y_pred).flatten()

# Create confusion matrix
confusion = confusion_matrix(y_test, y_pred_labels)

# Display the confusion matrix as a heatmap
plt.figure(figsize=(6, 6))
```

```
    sns.heatmap(confusion, annot=True, fmt="d", cmap="Blues", square=True,
                xticklabels=['Cat', 'Dog'],
                yticklabels=['Cat', 'Dog'])
    plt.xlabel("Predicted")
    plt.ylabel("True")
    plt.title("Confusion Matrix")
    plt.show()

    # Print classification report
    print('Classification Report')
    print(classification_report(y_test, y_pred_labels, target_names=['Cat', 'D
```




```
Classification Report
              precision    recall  f1-score   support

         Cat       0.50      0.66      0.57      2476
         Dog       0.50      0.34      0.41      2514

    accuracy                           0.50      4990
   macro avg       0.50      0.50      0.49      4990
weighted avg       0.50      0.50      0.49      4990
```


VGG16 Model

In [167]:
```
# VGG16 Model
def build_vgg16():
    model = models.Sequential()

    # Block 1
    model.add(layers.Conv2D(64, (3, 3), activation='relu', padding='same',
    model.add(layers.Conv2D(64, (3, 3), activation='relu', padding='same')
    model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))

    # Block 2
    model.add(layers.Conv2D(128, (3, 3), activation='relu', padding='same'
    model.add(layers.Conv2D(128, (3, 3), activation='relu', padding='same'
    model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))

    # Block 3
    model.add(layers.Conv2D(256, (3, 3), activation='relu', padding='same'
    model.add(layers.Conv2D(256, (3, 3), activation='relu', padding='same'
    model.add(layers.Conv2D(256, (3, 3), activation='relu', padding='same'
    model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))

    # flatten and Fully Connected Layers
    model.add(layers.Flatten())
    model.add(layers.Dense(4096, activation='relu'))
    model.add(layers.Dense(4096, activation='relu'))
    model.add(layers.Dense(10, activation='softmax'))

    return model
```

In [168]:
```
# Build the VGG16 model
model = build_vgg16()
```

In [169]:
```
# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=[
```

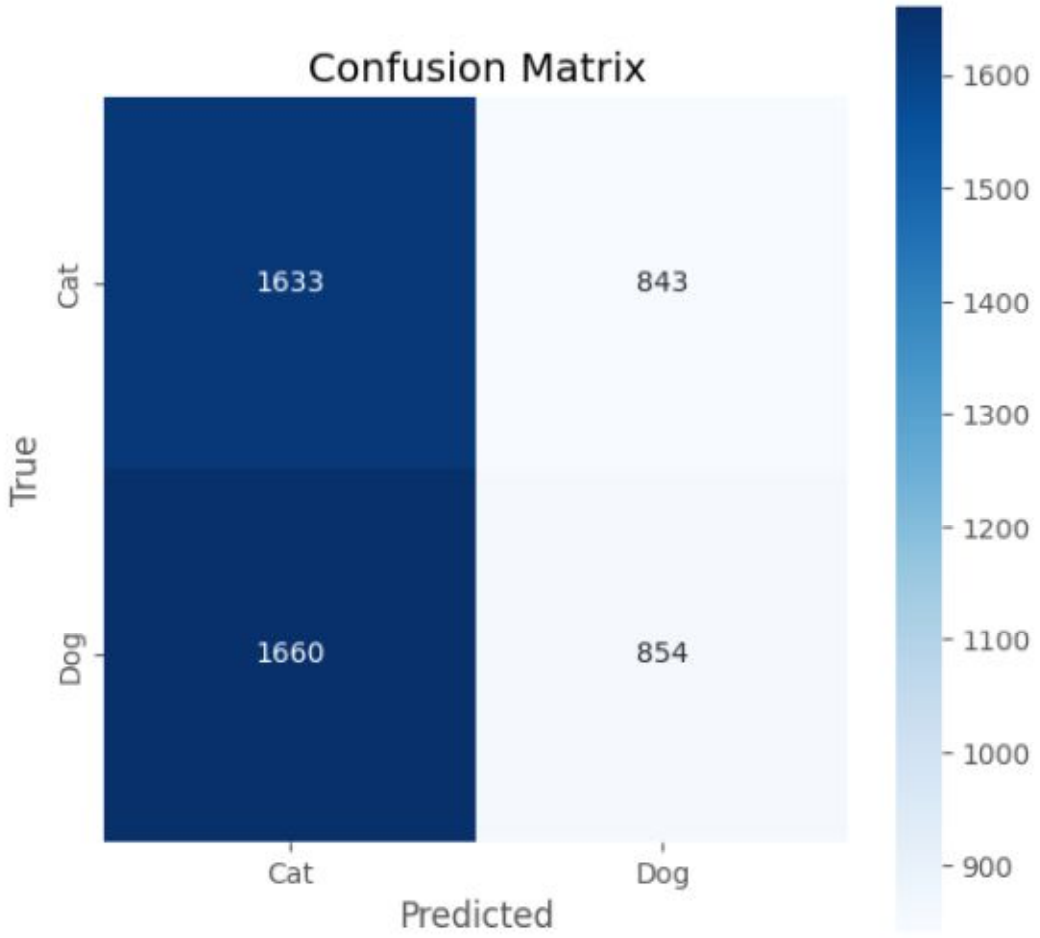
In [ ]:

In [179]:
```
from tensorflow.keras.models import Model
from keras.applications.vgg16 import VGG16

vgg_model = VGG16(weights='imagenet', include_top = False, input_shape = (
vgg_model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applicati
ons/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 2s 0us/step
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_34 (InputLayer)       [(None, 64, 64, 3)]       0

 block1_conv1 (Conv2D)       (None, 64, 64, 64)        1792

 block1_conv2 (Conv2D)       (None, 64, 64, 64)        36928

 block1_pool (MaxPooling2D)  (None, 32, 32, 64)        0

 block2_conv1 (Conv2D)       (None, 32, 32, 128)       73856

 block2_conv2 (Conv2D)       (None, 32, 32, 128)       147584

 block2_pool (MaxPooling2D)  (None, 16, 16, 128)       0

 block3_conv1 (Conv2D)       (None, 16, 16, 256)       295168

 block3_conv2 (Conv2D)       (None, 16, 16, 256)       590080

 block3_conv3 (Conv2D)       (None, 16, 16, 256)       590080

 block3_pool (MaxPooling2D)  (None, 8, 8, 256)         0

 block4_conv1 (Conv2D)       (None, 8, 8, 512)         1180160

 block4_conv2 (Conv2D)       (None, 8, 8, 512)         2359808

 block4_conv3 (Conv2D)       (None, 8, 8, 512)         2359808

 block4_pool (MaxPooling2D)  (None, 4, 4, 512)         0

 block5_conv1 (Conv2D)       (None, 4, 4, 512)         2359808

 block5_conv2 (Conv2D)       (None, 4, 4, 512)         2359808

 block5_conv3 (Conv2D)       (None, 4, 4, 512)         2359808

 block5_pool (MaxPooling2D)  (None, 2, 2, 512)         0

=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```


In [202]:
```
# Convert predicted probabilities to class labels (0 or 1)
```

```
vgg_model = np.round(y_pred).flatten()

# Create confusion matrix
confusion = confusion_matrix(y_test, y_pred_labels)

# Display the confusion matrix as a heatmap
plt.figure(figsize=(6, 6))
sns.heatmap(confusion, annot=True, fmt="d", cmap="Blues", square=True,
            xticklabels=['Cat', 'Dog'],
            yticklabels=['Cat', 'Dog'])
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix")
plt.show()

# Print classification report
print('Classification Report')
print(classification_report(y_test, y_pred_labels, target_names=['Cat', 'D
```




```
Classification Report
              precision    recall  f1-score   support

         Cat       0.50      0.66      0.57      2476
         Dog       0.50      0.34      0.41      2514

    accuracy                           0.50      4990
   macro avg       0.50      0.50      0.49      4990
weighted avg       0.50      0.50      0.49      4990
```

In [180]:
```
# Making all the layers of the VGG model non-trainable. i.e. freezing them
for layer in vgg_model.layers:
    layer.trainable = False

new_model = Sequential()
```

In [183]:
```
from tensorflow.keras.optimizers import Adam
```

In [184]:
```
new_model = Sequential()

# Adding the convolutional part of the VGG16 model from above
new_model.add(vgg_model)

# Flattening the output of the VGG16 model because it is from a convolutio
```

```
      new_model.add(Flatten())

      # Adding a dense output layer
      new_model.add(Dense(64, activation='relu'))
      new_model.add(Dropout(0.2))
      new_model.add(Dense(32, activation='relu'))
      new_model.add(Dense(10, activation='softmax'))
      opt=Adam()
      # Compile model
      new_model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=
In [185]:# Generating the summary of the model
      new_model.summary()

Model: "sequential_22"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 2, 2, 512)         14714688

 flatten_9 (Flatten)         (None, 2048)              0

 dense_60 (Dense)            (None, 64)                131136

 dropout_9 (Dropout)         (None, 64)                0

 dense_61 (Dense)            (None, 32)                2080

 dense_62 (Dense)            (None, 10)                330

=================================================================
Total params: 14,848,234
Trainable params: 133,546
Non-trainable params: 14,714,688
_________________________________________________________________
In [ ]:
In [ ]:plt.plot(history_vgg16.history['accuracy'])
    plt.plot(history_vgg16.history['val_accuracy'])
    plt.title('Model Accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Validation'], loc='upper left')
    plt.show()
In [ ]:
In [ ]:# Plotting the Confusion Matrix using confusion matrix() function which is a
    confusion_matrix = tf.math.confusion_matrix(y_test_arg,y_pred_arg)
    f, ax = plt.subplots(figsize=(10, 8))
    sns.heatmap(
        confusion_matrix,
        annot=True,
        linewidths=.4,
        fmt="d",
        square=True,
```

```
        ax=ax
    )
    plt.show()
```

Summary of the Train Accuracy of all the models

In [207]:
```
pd.DataFrame({'Models':['CNN Model','RESNET50','VGG16 Model'],'Train Accur
```

Out[20

	Models	Train Accura
0	CNN Model	78%
1	RESNET50	50%

In [ ]:

In [ ]: