CONVOLUTION REPORT
SUDARSHAN RAYAPATI

Introduction: Convolutional Neural Networks (CNNs) are a powerful class of deep learning models widely used for image classification tasks. In this report, we explore the performance of CNNs trained from scratch and using pre-trained models on a Cats & Dogs dataset. We vary the training sample sizes, employ optimization techniques, and compare the results to determine the most effective approach for image classification.

Methodology:

- Data Preparation:

We utilize the Cats & Dogs dataset, consisting of images of cats and dogs, divided into training, validation, and test sets.

- Training from Scratch:

1. We implement CNNs using TensorFlow/Keras, employing techniques like data augmentation and regularization to mitigate overfitting.
2. Training iterations are performed over 10 epochs, with batch sizes of 20.
3. We gradually increase the training sample sizes: 1000, 2000, and 2500 images, and test and validation sample size of 500

Optimization techniques include:

4. Data augmentation: Randomly rotating, shifting, and flipping images to increase dataset diversity.
5. Regularization: L2 regularization to penalize large weights and prevent overfitting.
6. Test accuracies are recorded for each training sample size.

- Pre-trained Networks:

1. We leverage a pre-trained ResNet50 model for transfer learning.
2. Training iterations are conducted over 10 epochs, with batch sizes of 20.
3. We evaluate the model using training sample sizes of 1000, 1500, and 2000 images, and test and validation sample size of 500
4. Optimization techniques include:
5. Fine-tuning: We freeze the pre-trained layers and only train the newly added classifier layers.
6. Learning rate adjustment: We adjust the learning rate for fine-tuning the pre-trained model.
7. Test accuracies are recorded for each training sample size.

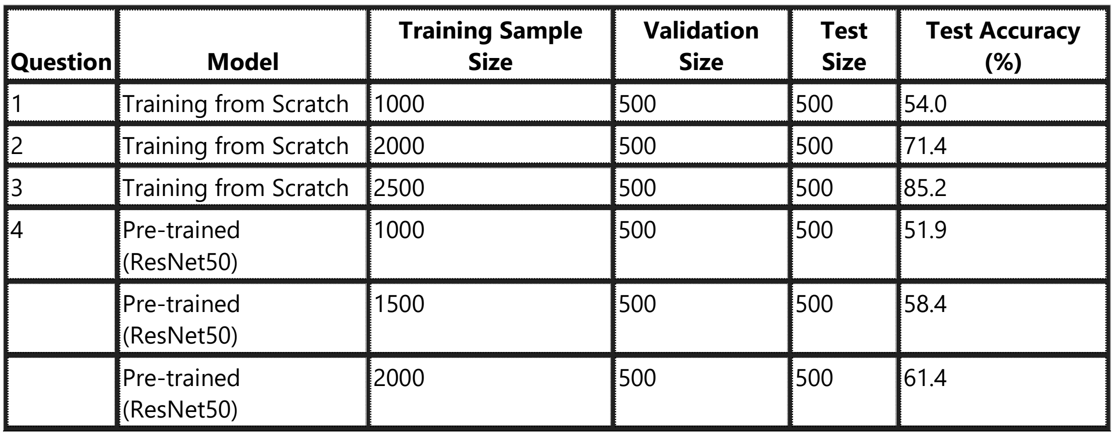
Results:

Training from Scratch:

1. Test accuracies steadily increase with larger training sample sizes: 54.0% (1000), 71.4% (2000), and 85.2% (2500).

Pre-trained Networks (ResNet50):

1. Test accuracies vary: 51.9% (1000), 58.4% (1500), and 61.4% (2000).
2. The pre-trained model exhibits competitive performance, particularly with larger training sample sizes.



Question	Model	Training Sample Size	Validation Size	Test Size	Test Accuracy (%)
1	Training from Scratch	1000	500	500	54.0
2	Training from Scratch	2000	500	500	71.4
3	Training from Scratch	2500	500	500	85.2
4	Pre-trained (ResNet50)	1000	500	500	51.9
	Pre-trained (ResNet50)	1500	500	500	58.4
	Pre-trained (ResNet50)	2000	500	500	61.4

Assignment 2: Convolution Sudarshan Rayapati

In [1]:

```
from google.colab import drive
import zipfile
import os
```

In [2]:

```
# Mount Google Drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [3]:

```
import os

# Directory containing the extracted files
extracted_dir_path = '/content/drive/MyDrive'

# List all files in the directory
files = os.listdir(extracted_dir_path)

print(files)
```

```
['cats_vs_dogs_small_dataset.zip', 'Colab Notebooks', 'cats_vs_dogs_small_dataset']
```

In [4]:

```
# Path to the zip file
zip_file_path = '/content/drive/My Drive/cats_vs_dogs_small_dataset.zip'

# Directory to extract the files
extracted_dir_path = '/content/drive/My Drive/cats_vs_dogs_small_dataset'
```

In [ ]:

```
# Unzip the file
with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    zip_ref.extractall(extracted_dir_path)

# Check the extracted files
extracted_files = os.listdir(extracted_dir_path)
print("Files extracted successfully:", extracted_files)
```

In [5]:

```
# Path to the 'cat' and 'dog' folders
cat_folder_path = os.path.join(extracted_dir_path, 'cat')
dog_folder_path = os.path.join(extracted_dir_path, 'dog')

# Function to count the number of images in a folder
def count_images(folder_path):
    # List all files in the directory
    files = os.listdir(folder_path)
    # Count only files with .jpg or .png extension
    image_files = [file for file in files if file.endswith('.jpg') or file.endswith('.
    return len(image_files)

# Count the number of images in the 'cat' and 'dog' folders
num_cat_images = count_images(cat_folder_path)
num_dog_images = count_images(dog_folder_path)

# Display the results
```

```
print("Number of images in 'cat' folder:", num_cat_images)
print("Number of images in 'dog' folder:", num_dog_images)
```

```
Number of images in 'cat' folder: 2000
Number of images in 'dog' folder: 2000
```

1. Consider the Cats & Dogs example. Start initially with a training sample of 1000, a validation sample of 500, and a test sample of 500 (like in the text). Use any technique to reduce overfitting and improve performance in developing a network that you train from scratch. What performance did you achieve?

In [6]:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
import matplotlib.pyplot as plt
import os
import shutil
import random
```

Splitting The Dataset

In [7]:

```
base_dir = '/content/cats_vs_dogs_dataset'
os.makedirs(base_dir, exist_ok=True)
train_dir = os.path.join(base_dir, 'train')
os.makedirs(train_dir, exist_ok=True)
validation_dir = os.path.join(base_dir, 'validation')
os.makedirs(validation_dir, exist_ok=True)
test_dir = os.path.join(base_dir, 'test')
os.makedirs(test_dir, exist_ok=True)
```

In [8]:

```
train_cats_dir = os.path.join(train_dir, 'cat')
os.makedirs(train_cats_dir, exist_ok=True)
train_dogs_dir = os.path.join(train_dir, 'dog')
os.makedirs(train_dogs_dir, exist_ok=True)

validation_cats_dir = os.path.join(validation_dir, 'cat')
os.makedirs(validation_cats_dir, exist_ok=True)
validation_dogs_dir = os.path.join(validation_dir, 'dog')
os.makedirs(validation_dogs_dir, exist_ok=True)

test_cats_dir = os.path.join(test_dir, 'cat')
os.makedirs(test_cats_dir, exist_ok=True)
test_dogs_dir = os.path.join(test_dir, 'dog')
os.makedirs(test_dogs_dir, exist_ok=True)
```

In [9]:

```
def copy_images(src_dir, dst_dir, file_list):
    for file in file_list:
        src_path = os.path.join(src_dir, file)
        dst_path = os.path.join(dst_dir, file)
        shutil.copyfile(src_path, dst_path)
```

In [10]:

```
cat_images = os.listdir(cat_folder_path)
dog_images = os.listdir(dog_folder_path)
```

```
random.shuffle(cat_images)
random.shuffle(dog_images)
```

In [11]:

```
train_samples = 1000
validation_samples = 500
test_samples = 500
```

In [12]:

```
copy_images(cat_folder_path, train_cats_dir, cat_images[:train_samples//2])
copy_images(dog_folder_path, train_dogs_dir, dog_images[:train_samples//2])

copy_images(cat_folder_path, validation_cats_dir, cat_images[train_samples//2:train_sa
copy_images(dog_folder_path, validation_dogs_dir, dog_images[train_samples//2:train_sa

copy_images(cat_folder_path, test_cats_dir, cat_images[train_samples//2 + validation_s
copy_images(dog_folder_path, test_dogs_dir, dog_images[train_samples//2 + validation_s
```

In [13]:

```
batch_size = 20
image_size = (150, 150)

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

validation_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='binary'
)

validation_generator = validation_datagen.flow_from_directory(
    validation_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='binary'
)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='binary'
)
```

```
Found 1000 images belonging to 2 classes.
Found 500 images belonging to 2 classes.
Found 500 images belonging to 2 classes.
```

Model Definition

In [14]:

```
# Define the model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Display the model summary
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 32)      896       
                                                                 
 max_pooling2d (MaxPooling2  (None, 74, 74, 32)        0         
 D)                                                              
                                                                 
 conv2d_1 (Conv2D)           (None, 72, 72, 64)        18496     
                                                                 
 max_pooling2d_1 (MaxPoolin  (None, 36, 36, 64)        0         
 g2D)                                                            
                                                                 
 conv2d_2 (Conv2D)           (None, 34, 34, 128)       73856     
                                                                 
 max_pooling2d_2 (MaxPoolin  (None, 17, 17, 128)       0         
 g2D)                                                            
                                                                 
 conv2d_3 (Conv2D)           (None, 15, 15, 128)       147584    
                                                                 
 max_pooling2d_3 (MaxPoolin  (None, 7, 7, 128)         0         
 g2D)                                                            
                                                                 
 flatten (Flatten)           (None, 6272)              0         
                                                                 
 dense (Dense)               (None, 512)               3211776   
                                                                 
 dropout (Dropout)           (None, 512)               0         
                                                                 
 dense_1 (Dense)             (None, 1)                 513       
                                                                 
=================================================================
Total params: 3453121 (13.17 MB)
Trainable params: 3453121 (13.17 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Training the Model

In [15]:

```python
# Define parameters for training
epochs = 10

# Train the model
history = model.fit(
    train_generator,
    steps_per_epoch=train_samples // batch_size,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=validation_samples // batch_size
)
```

```
Epoch 1/10
50/50 [==============================] - 36s 699ms/step - loss: 0.7013 - accuracy: 0.
5040 - val_loss: 0.6882 - val_accuracy: 0.5000
Epoch 2/10
50/50 [==============================] - 39s 790ms/step - loss: 0.6938 - accuracy: 0.
5290 - val_loss: 0.6897 - val_accuracy: 0.5720
Epoch 3/10
50/50 [==============================] - 36s 711ms/step - loss: 0.6857 - accuracy: 0.
5490 - val_loss: 0.6719 - val_accuracy: 0.5580
Epoch 4/10
50/50 [==============================] - 35s 691ms/step - loss: 0.6815 - accuracy: 0.
5360 - val_loss: 0.6760 - val_accuracy: 0.5580
Epoch 5/10
50/50 [==============================] - 34s 684ms/step - loss: 0.6712 - accuracy: 0.
5970 - val_loss: 0.6553 - val_accuracy: 0.5740
Epoch 6/10
50/50 [==============================] - 34s 680ms/step - loss: 0.6559 - accuracy: 0.
5990 - val_loss: 0.6465 - val_accuracy: 0.6160
Epoch 7/10
50/50 [==============================] - 34s 681ms/step - loss: 0.6561 - accuracy: 0.
5860 - val_loss: 0.6640 - val_accuracy: 0.5900
Epoch 8/10
50/50 [==============================] - 33s 658ms/step - loss: 0.6616 - accuracy: 0.
5830 - val_loss: 0.6795 - val_accuracy: 0.5400
Epoch 9/10
50/50 [==============================] - 34s 684ms/step - loss: 0.6729 - accuracy: 0.
5640 - val_loss: 0.6605 - val_accuracy: 0.5780
Epoch 10/10
50/50 [==============================] - 34s 684ms/step - loss: 0.6658 - accuracy: 0.
5800 - val_loss: 0.6650 - val_accuracy: 0.5660
```

In [16]:

```
# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(test_generator, steps=test_samples // batch_
print("Test accuracy:", test_accuracy)
```

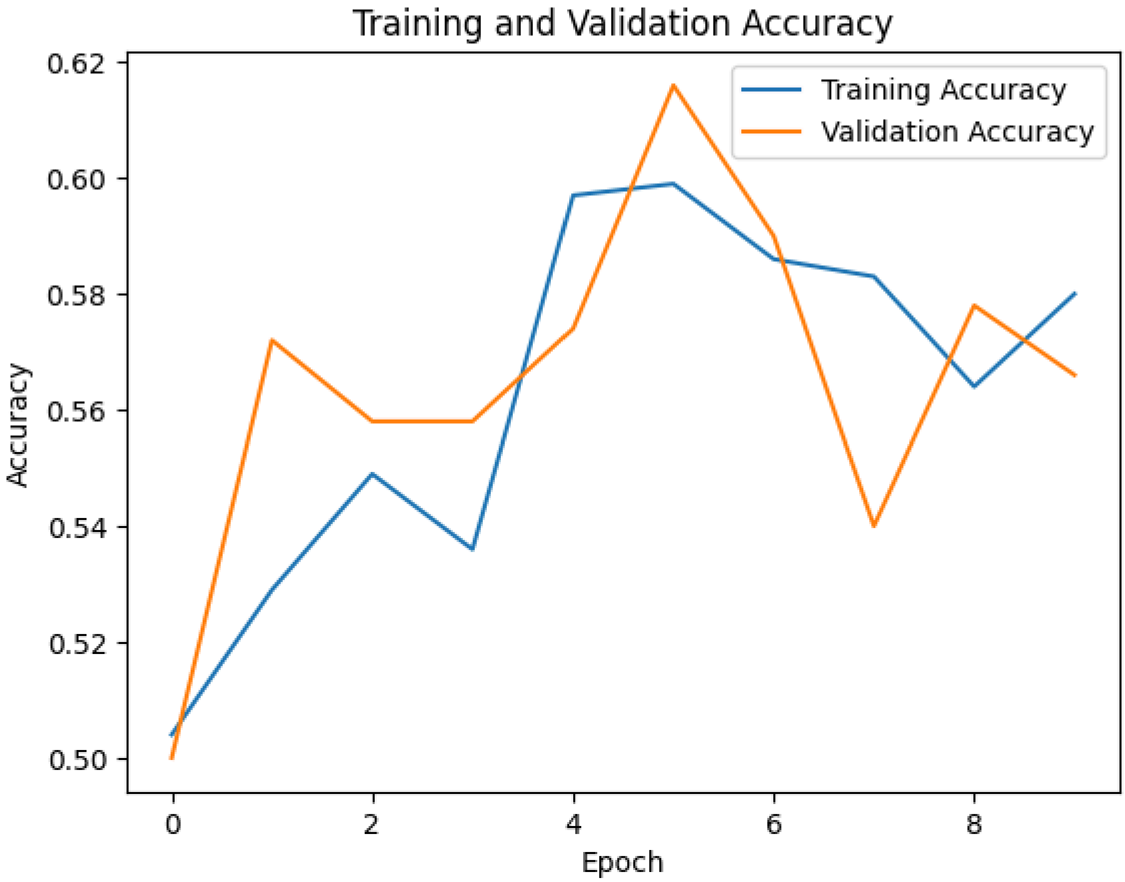
```
25/25 [==============================] - 5s 181ms/step - loss: 0.6759 - accuracy: 0.5
460
Test accuracy: 0.5460000038146973
```

Perfomance Metrics

In [17]:

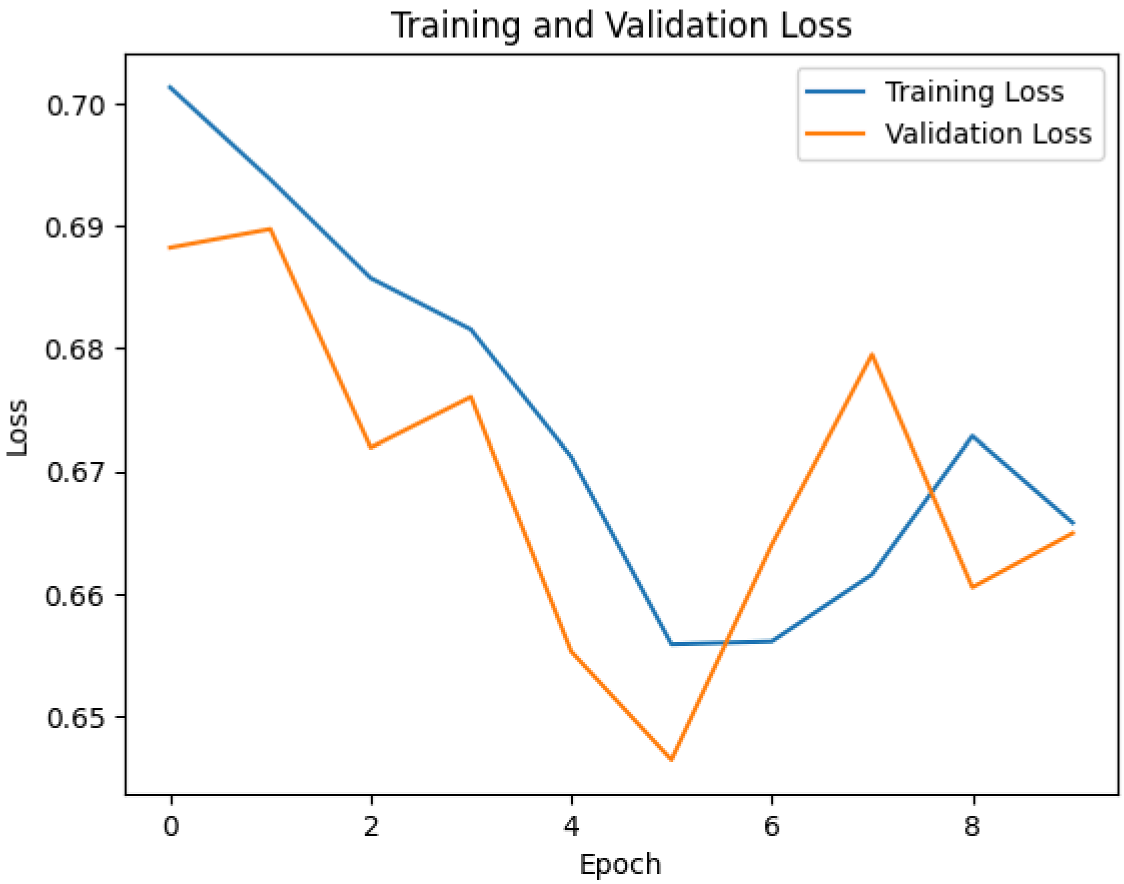
```
import matplotlib.pyplot as plt

# Plot training and validation accuracy
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



In [18]:

```
# Plot training and validation loss
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



1. Increase your training sample size. You may pick any amount. Keep the validation and test samples the same as above. Optimize your network (again training from scratch). What performance did you achieve?

Splitting The Dataset

In [22]:

```
# Paths to the 'cat' and 'dog' folders
cat_folder_path = '/content/drive/MyDrive/cats_vs_dogs_small_dataset/cat'
dog_folder_path = '/content/drive/MyDrive/cats_vs_dogs_small_dataset/dog'

# Define new sample sizes
new_train_samples = 2000

# Copy images to training directory
copy_images(cat_folder_path, train_cats_dir, cat_images[:new_train_samples//2])
copy_images(dog_folder_path, train_dogs_dir, dog_images[:new_train_samples//2])

# Create ImageDataGenerator for training set with augmentation
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='binary'
)
```

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

Training the model

In [24]:

```
# Define parameters for training
epochs = 10

# Train the model
history = model.fit(
    train_generator,
    steps_per_epoch=new_train_samples // batch_size,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=validation_samples // batch_size
)

# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(test_generator, steps=test_samples // batch_
print("Test accuracy with increased training sample size:", test_accuracy)
```

```
Epoch 1/10
100/100 [==============================] - 64s 641ms/step - loss: 0.5881 - accuracy:
0.6870 - val_loss: 0.5327 - val_accuracy: 0.7280
Epoch 2/10
100/100 [==============================] - 64s 641ms/step - loss: 0.5934 - accuracy:
0.6745 - val_loss: 0.5311 - val_accuracy: 0.7520
Epoch 3/10
100/100 [==============================] - 65s 647ms/step - loss: 0.5693 - accuracy:
0.7060 - val_loss: 0.5224 - val_accuracy: 0.7480
Epoch 4/10
100/100 [==============================] - 65s 650ms/step - loss: 0.5803 - accuracy:
0.6895 - val_loss: 0.5420 - val_accuracy: 0.7080
Epoch 5/10
100/100 [==============================] - 63s 632ms/step - loss: 0.5869 - accuracy:
0.6935 - val_loss: 0.5203 - val_accuracy: 0.7480
Epoch 6/10
100/100 [==============================] - 65s 646ms/step - loss: 0.5724 - accuracy:
0.7025 - val_loss: 0.5686 - val_accuracy: 0.7160
Epoch 7/10
100/100 [==============================] - 64s 637ms/step - loss: 0.5733 - accuracy:
0.7035 - val_loss: 0.4933 - val_accuracy: 0.7720
Epoch 8/10
100/100 [==============================] - 64s 639ms/step - loss: 0.5651 - accuracy:
0.7175 - val_loss: 0.5412 - val_accuracy: 0.7440
Epoch 9/10
100/100 [==============================] - 64s 636ms/step - loss: 0.5558 - accuracy:
0.7120 - val_loss: 0.4924 - val_accuracy: 0.7900
Epoch 10/10
100/100 [==============================] - 64s 639ms/step - loss: 0.5454 - accuracy:
0.7125 - val_loss: 0.4975 - val_accuracy: 0.7760
25/25 [==============================] - 5s 186ms/step - loss: 0.5214 - accuracy: 0.7
140
Test accuracy with increased training sample size: 0.7139999866485596
```

In [25]:

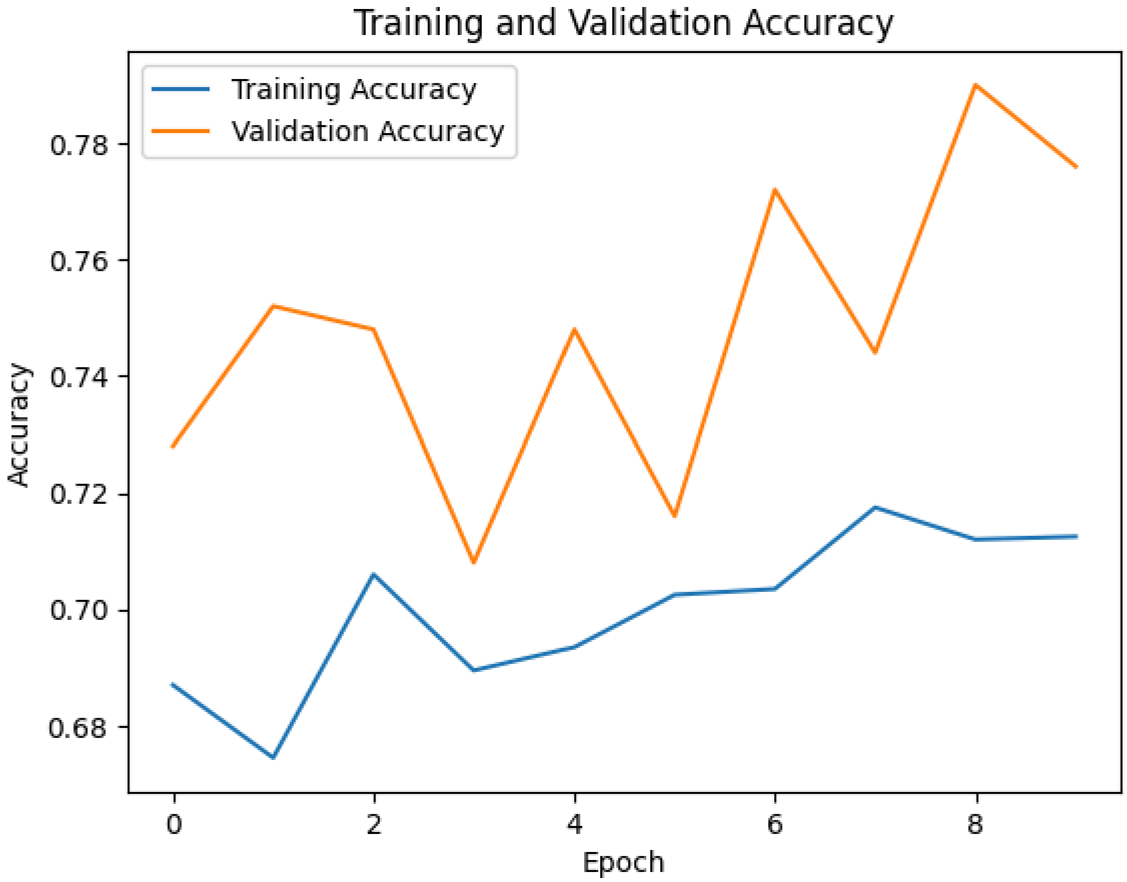
```
# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(test_generator, steps=test_samples // batch_
print("Test accuracy:", test_accuracy)
```

```
25/25 [==============================] - 4s 167ms/step - loss: 0.5214 - accuracy: 0.7
140
Test accuracy: 0.7139999866485596
```

Perfomance Metrics

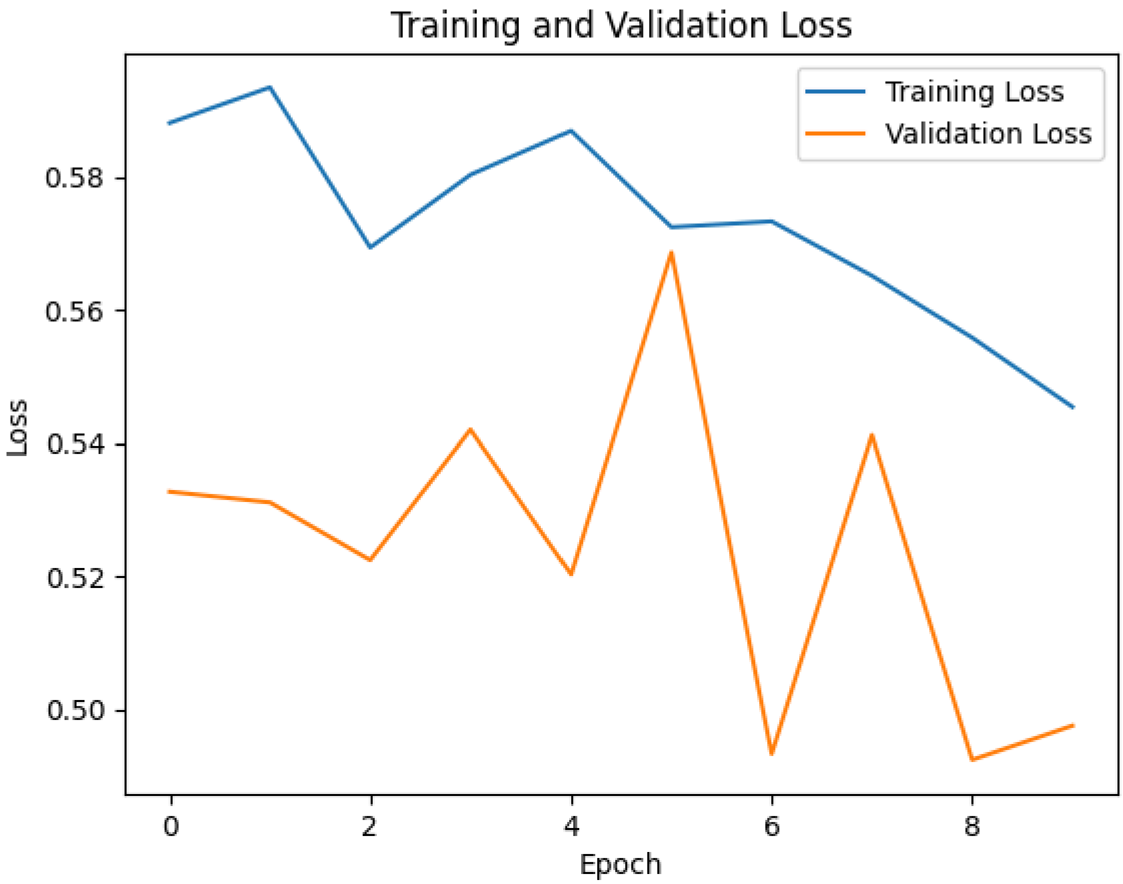
In [26]:

```
# Plot training and validation accuracy
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



In [27]:

```
# Plot training and validation loss
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



1. Now change your training sample so that you achieve better performance than those from Steps 1 and 2. This sample size may be larger, or smaller than those in the previous steps. The objective is to find the ideal training sample size to get best prediction results

Choosing different training samples

In [28]:
```
training_sample_sizes = [500, 1000, 1500, 2000, 2500]
```

Training the Models on different test samples

In [35]:
```
# Iterate over the list of training sample sizes
for sample_size in training_sample_sizes:
    # Clear the existing training directory
    shutil.rmtree(train_dir)
    os.makedirs(train_dir, exist_ok=True)
    train_cats_dir = os.path.join(train_dir, 'cat')
    os.makedirs(train_cats_dir, exist_ok=True)
    train_dogs_dir = os.path.join(train_dir, 'dog')
    os.makedirs(train_dogs_dir, exist_ok=True)

    # Copy images to training directory based on the current sample size
    copy_images(cat_folder_path, train_cats_dir, cat_images[:sample_size//2])
    copy_images(dog_folder_path, train_dogs_dir, dog_images[:sample_size//2])

    # Create ImageDataGenerator for training set with augmentation
    train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=image_size,
        batch_size=batch_size,
        class_mode='binary'
    )

    # Train the model
    history = model.fit(
        train_generator,
        steps_per_epoch=sample_size // batch_size,
        epochs=epochs,
        validation_data=validation_generator,
        validation_steps=validation_samples // batch_size,
        verbose=0  # Disable verbose output for cleaner logging
    )

    # Evaluate the model on the test set
    test_loss, test_accuracy = model.evaluate(test_generator, steps=test_samples // ba
    test_accuracies.append(test_accuracy)

    print(f"Test accuracy with training sample size {sample_size}: {test_accuracy}")

# Find the best performing training sample size
best_sample_size = training_sample_sizes[test_accuracies.index(max(test_accuracies))]
print(f"\nBest performing training sample size: {best_sample_size} with test accuracy:
```

```
Found 500 images belonging to 2 classes.
Test accuracy with training sample size 500: 0.7120000123977661
Found 1000 images belonging to 2 classes.
Test accuracy with training sample size 1000: 0.7160000205039978
Found 1500 images belonging to 2 classes.
Test accuracy with training sample size 1500: 0.734000027179718
Found 2000 images belonging to 2 classes.
Test accuracy with training sample size 2000: 0.7919999957084656
Found 2500 images belonging to 2 classes.
Test accuracy with training sample size 2500: 0.8519999980926514

Best performing training sample size: 2500 with test accuracy: 0.8519999980926514
```

In [36]:

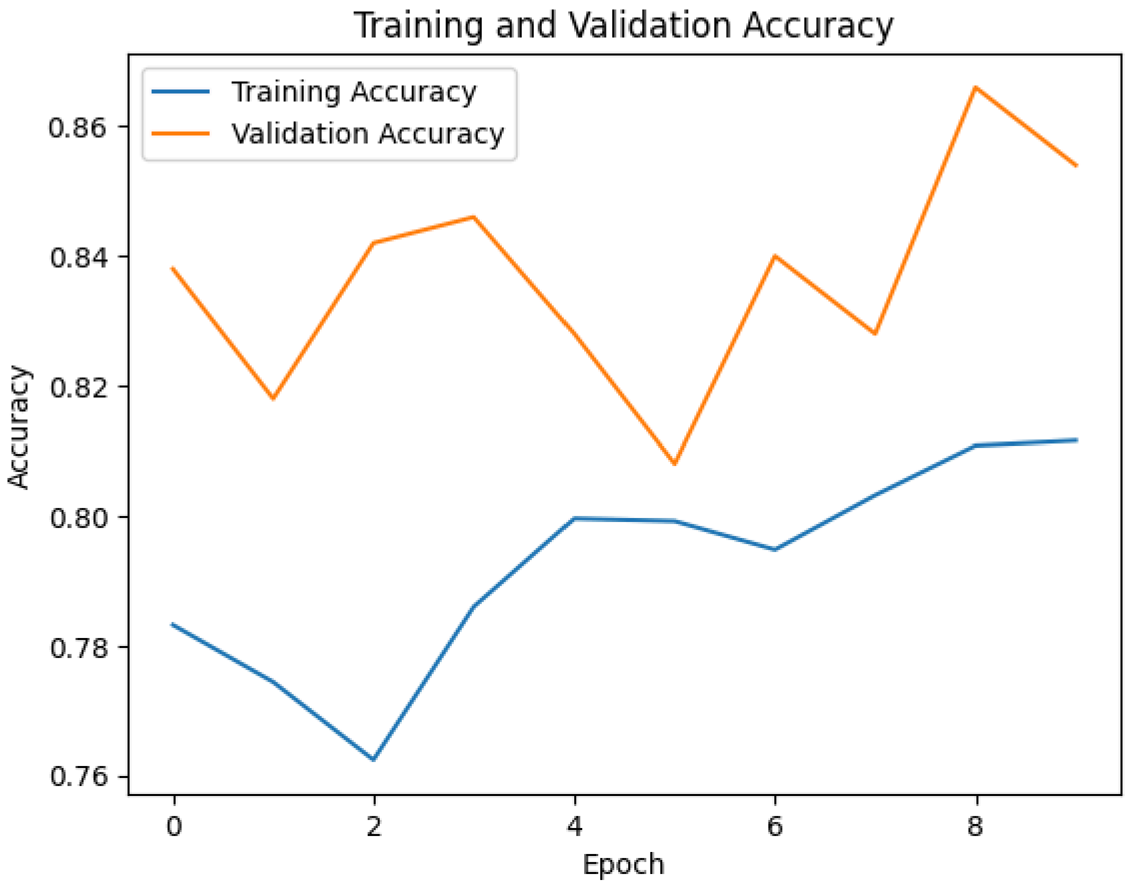
```
# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(test_generator, steps=test_samples // batch_
print("Test accuracy:", test_accuracy)
```

```
25/25 [==============================] - 8s 315ms/step - loss: 0.3270 - accuracy: 0.8
520
Test accuracy: 0.8519999980926514
```

Evaluation Metrics

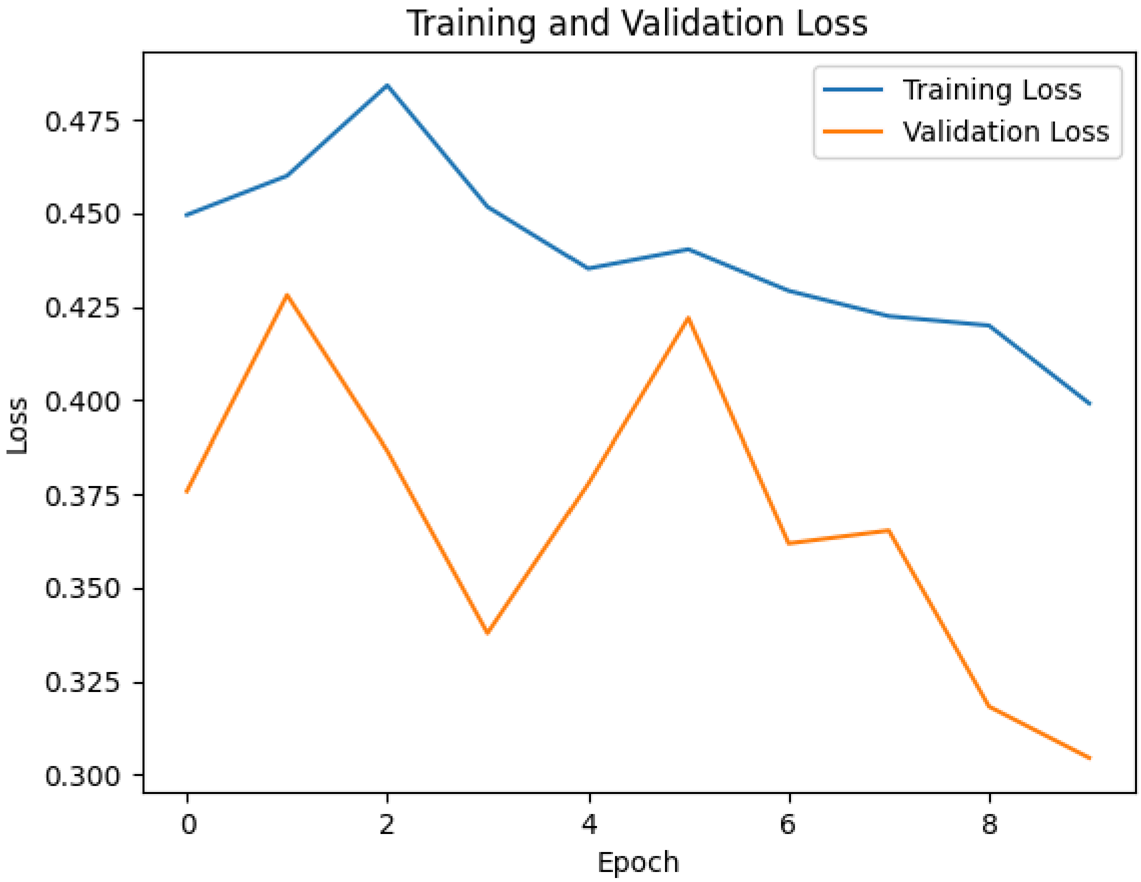
In [37]:

```
# Plot training and validation accuracy
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



In [38]:

```
# Plot training and validation loss
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



1. Repeat Steps 1-3, but now using a pretrained network. The sample sizes you use in Steps 2 and 3 for the pretrained network may be the same or different from those using the network where you trained from scratch. Again, use any and all optimization techniques to get best performance.

In [39]:

```
import tensorflow as tf
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Model
```

Using ResNet

In [40]:

```
# Load pre-trained ResNet50 model without the top classification layers
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(150, 150, 3)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/re
snet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 [==============================] - 1s 0us/step
```

In [41]:

```
# Freeze the weights of the pre-trained layers
for layer in base_model.layers:
    layer.trainable = False

# Add custom classification layers
x = Flatten()(base_model.output)
x = Dense(256, activation='relu')(x)
output = Dense(1, activation='sigmoid')(x)
```

```
# Create the model
model = Model(inputs=base_model.input, outputs=output)

# Compile the model
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

model.summary()
```

```
Model: "model"
______________________________________________________________________________________
_____________
 Layer (type)                Output Shape                 Param #   Connected to
======================================================================================
=============
 input_1 (InputLayer)        [(None, 150, 150, 3)]        0         []

 conv1_pad (ZeroPadding2D)   (None, 156, 156, 3)          0         ['input_1[0][0]']

 conv1_conv (Conv2D)         (None, 75, 75, 64)           9472      ['conv1_pad[0]
[0]']

 conv1_bn (BatchNormalizati  (None, 75, 75, 64)           256       ['conv1_conv[0]
[0]']
 on)

 conv1_relu (Activation)     (None, 75, 75, 64)           0         ['conv1_bn[0]
[0]']

 pool1_pad (ZeroPadding2D)   (None, 77, 77, 64)           0         ['conv1_relu[0]
[0]']

 pool1_pool (MaxPooling2D)   (None, 38, 38, 64)           0         ['pool1_pad[0]
[0]']

 conv2_block1_1_conv (Conv2  (None, 38, 38, 64)           4160      ['pool1_pool[0]
[0]']
 D)

 conv2_block1_1_bn (BatchNo  (None, 38, 38, 64)           256       ['conv2_block1_1_
conv[0][0]']
 rmalization)

 conv2_block1_1_relu (Activ  (None, 38, 38, 64)           0         ['conv2_block1_1_
bn[0][0]']
 ation)

 conv2_block1_2_conv (Conv2  (None, 38, 38, 64)           36928     ['conv2_block1_1_
relu[0][0]']
 D)

 conv2_block1_2_bn (BatchNo  (None, 38, 38, 64)           256       ['conv2_block1_2_
conv[0][0]']
 rmalization)

 conv2_block1_2_relu (Activ  (None, 38, 38, 64)           0         ['conv2_block1_2_
bn[0][0]']
 ation)

 conv2_block1_0_conv (Conv2  (None, 38, 38, 256)          16640     ['pool1_pool[0]
[0]']
 D)

 conv2_block1_3_conv (Conv2  (None, 38, 38, 256)          16640     ['conv2_block1_2_
relu[0][0]']
 D)

 conv2_block1_0_bn (BatchNo  (None, 38, 38, 256)          1024      ['conv2_block1_0_
conv[0][0]']
```

```
 rmalization)

 conv2_block1_3_bn (BatchNo  (None, 38, 38, 256)       1024      ['conv2_block1_3_
conv[0][0]']
 rmalization)

 conv2_block1_add (Add)      (None, 38, 38, 256)       0         ['conv2_block1_0_
bn[0][0]',

                                                                  'conv2_block1_3_
bn[0][0]']

 conv2_block1_out (Activati  (None, 38, 38, 256)       0         ['conv2_block1_ad
d[0][0]']
 on)

 conv2_block2_1_conv (Conv2  (None, 38, 38, 64)        16448     ['conv2_block1_ou
t[0][0]']
 D)

 conv2_block2_1_bn (BatchNo  (None, 38, 38, 64)        256       ['conv2_block2_1_
conv[0][0]']
 rmalization)

 conv2_block2_1_relu (Activ  (None, 38, 38, 64)        0         ['conv2_block2_1_
bn[0][0]']
 ation)

 conv2_block2_2_conv (Conv2  (None, 38, 38, 64)        36928     ['conv2_block2_1_
relu[0][0]']
 D)

 conv2_block2_2_bn (BatchNo  (None, 38, 38, 64)        256       ['conv2_block2_2_
conv[0][0]']
 rmalization)

 conv2_block2_2_relu (Activ  (None, 38, 38, 64)        0         ['conv2_block2_2_
bn[0][0]']
 ation)

 conv2_block2_3_conv (Conv2  (None, 38, 38, 256)       16640     ['conv2_block2_2_
relu[0][0]']
 D)

 conv2_block2_3_bn (BatchNo  (None, 38, 38, 256)       1024      ['conv2_block2_3_
conv[0][0]']
 rmalization)

 conv2_block2_add (Add)      (None, 38, 38, 256)       0         ['conv2_block1_ou
t[0][0]',

                                                                  'conv2_block2_3_
bn[0][0]']

 conv2_block2_out (Activati  (None, 38, 38, 256)       0         ['conv2_block2_ad
d[0][0]']
 on)

 conv2_block3_1_conv (Conv2  (None, 38, 38, 64)        16448     ['conv2_block2_ou
t[0][0]']
 D)
 conv2_block3_1_bn (BatchNo  (None, 38, 38, 64)           256       ['conv2_block3_1_
conv[0][0]']
 rmalization)

 conv2_block3_1_relu (Activ  (None, 38, 38, 64)           0         ['conv2_block3_1_
bn[0][0]']
 ation)

 conv2_block3_2_conv (Conv2  (None, 38, 38, 64)           36928     ['conv2_block3_1_
relu[0][0]']
 D)

 conv2_block3_2_bn (BatchNo  (None, 38, 38, 64)           256       ['conv2_block3_2_
conv[0][0]']
 rmalization)

 conv2_block3_2_relu (Activ  (None, 38, 38, 64)           0         ['conv2_block3_2_
bn[0][0]']
 ation)

 conv2_block3_3_conv (Conv2  (None, 38, 38, 256)          16640     ['conv2_block3_2_
relu[0][0]']
 D)

 conv2_block3_3_bn (BatchNo  (None, 38, 38, 256)          1024      ['conv2_block3_3_
conv[0][0]']
 rmalization)

 conv2_block3_add (Add)      (None, 38, 38, 256)          0         ['conv2_block2_ou
t[0][0]',

                                                                     'conv2_block3_3_
bn[0][0]']

 conv2_block3_out (Activati  (None, 38, 38, 256)          0         ['conv2_block3_ad
d[0][0]']
 on)

 conv3_block1_1_conv (Conv2  (None, 19, 19, 128)          32896     ['conv2_block3_ou
t[0][0]']
 D)

 conv3_block1_1_bn (BatchNo  (None, 19, 19, 128)          512       ['conv3_block1_1_
conv[0][0]']
 rmalization)

 conv3_block1_1_relu (Activ  (None, 19, 19, 128)          0         ['conv3_block1_1_
bn[0][0]']
 ation)

 conv3_block1_2_conv (Conv2  (None, 19, 19, 128)          147584    ['conv3_block1_1_
relu[0][0]']
 D)

 conv3_block1_2_bn (BatchNo  (None, 19, 19, 128)          512       ['conv3_block1_2_
conv[0][0]']
 rmalization)

 conv3_block1_2_relu (Activ  (None, 19, 19, 128)          0         ['conv3_block1_2_
bn[0][0]']
 ation)
 conv3_block1_0_conv (Conv2  (None, 19, 19, 512)          131584    ['conv2_block3_ou
t[0][0]']
 D)

 conv3_block1_3_conv (Conv2  (None, 19, 19, 512)          66048     ['conv3_block1_2_
relu[0][0]']
 D)

 conv3_block1_0_bn (BatchNo  (None, 19, 19, 512)          2048      ['conv3_block1_0_
conv[0][0]']
 rmalization)

 conv3_block1_3_bn (BatchNo  (None, 19, 19, 512)          2048      ['conv3_block1_3_
conv[0][0]']
 rmalization)

 conv3_block1_add (Add)      (None, 19, 19, 512)          0         ['conv3_block1_0_
bn[0][0]',
                                                                     'conv3_block1_3_
bn[0][0]']

 conv3_block1_out (Activati  (None, 19, 19, 512)          0         ['conv3_block1_ad
d[0][0]']
 on)

 conv3_block2_1_conv (Conv2  (None, 19, 19, 128)          65664     ['conv3_block1_ou
t[0][0]']
 D)

 conv3_block2_1_bn (BatchNo  (None, 19, 19, 128)          512       ['conv3_block2_1_
conv[0][0]']
 rmalization)

 conv3_block2_1_relu (Activ  (None, 19, 19, 128)          0         ['conv3_block2_1_
bn[0][0]']
 ation)

 conv3_block2_2_conv (Conv2  (None, 19, 19, 128)          147584    ['conv3_block2_1_
relu[0][0]']
 D)

 conv3_block2_2_bn (BatchNo  (None, 19, 19, 128)          512       ['conv3_block2_2_
conv[0][0]']
 rmalization)

 conv3_block2_2_relu (Activ  (None, 19, 19, 128)          0         ['conv3_block2_2_
bn[0][0]']
 ation)

 conv3_block2_3_conv (Conv2  (None, 19, 19, 512)          66048     ['conv3_block2_2_
relu[0][0]']
 D)

 conv3_block2_3_bn (BatchNo  (None, 19, 19, 512)          2048      ['conv3_block2_3_
conv[0][0]']
 rmalization)

 conv3_block2_add (Add)      (None, 19, 19, 512)          0         ['conv3_block1_ou
t[0][0]',
                                                                     'conv3_block2_3_
bn[0][0]']

 conv3_block2_out (Activati  (None, 19, 19, 512)          0         ['conv3_block2_ad
d[0][0]']
 on)

 conv3_block3_1_conv (Conv2  (None, 19, 19, 128)          65664     ['conv3_block2_ou
t[0][0]']
 D)

 conv3_block3_1_bn (BatchNo  (None, 19, 19, 128)          512       ['conv3_block3_1_
conv[0][0]']
 rmalization)

 conv3_block3_1_relu (Activ  (None, 19, 19, 128)          0         ['conv3_block3_1_
bn[0][0]']
 ation)

 conv3_block3_2_conv (Conv2  (None, 19, 19, 128)          147584    ['conv3_block3_1_
relu[0][0]']
 D)

 conv3_block3_2_bn (BatchNo  (None, 19, 19, 128)          512       ['conv3_block3_2_
conv[0][0]']
 rmalization)

 conv3_block3_2_relu (Activ  (None, 19, 19, 128)          0         ['conv3_block3_2_
bn[0][0]']
 ation)

 conv3_block3_3_conv (Conv2  (None, 19, 19, 512)          66048     ['conv3_block3_2_
relu[0][0]']
 D)

 conv3_block3_3_bn (BatchNo  (None, 19, 19, 512)          2048      ['conv3_block3_3_
conv[0][0]']
 rmalization)

 conv3_block3_add (Add)      (None, 19, 19, 512)          0         ['conv3_block2_ou
t[0][0]',
                                                                     'conv3_block3_3_
bn[0][0]']

 conv3_block3_out (Activati  (None, 19, 19, 512)          0         ['conv3_block3_ad
d[0][0]']
 on)

 conv3_block4_1_conv (Conv2  (None, 19, 19, 128)          65664     ['conv3_block3_ou
t[0][0]']
 D)

 conv3_block4_1_bn (BatchNo  (None, 19, 19, 128)          512       ['conv3_block4_1_
conv[0][0]']
 rmalization)

 conv3_block4_1_relu (Activ  (None, 19, 19, 128)          0         ['conv3_block4_1_
bn[0][0]']
 ation)
 conv3_block4_2_conv (Conv2  (None, 19, 19, 128)          147584    ['conv3_block4_1_
relu[0][0]']
 D)

 conv3_block4_2_bn (BatchNo  (None, 19, 19, 128)          512       ['conv3_block4_2_
conv[0][0]']
 rmalization)

 conv3_block4_2_relu (Activ  (None, 19, 19, 128)          0         ['conv3_block4_2_
bn[0][0]']
 ation)

 conv3_block4_3_conv (Conv2  (None, 19, 19, 512)          66048     ['conv3_block4_2_
relu[0][0]']
 D)

 conv3_block4_3_bn (BatchNo  (None, 19, 19, 512)          2048      ['conv3_block4_3_
conv[0][0]']
 rmalization)

 conv3_block4_add (Add)      (None, 19, 19, 512)          0         ['conv3_block3_ou
t[0][0]',
                                                                     'conv3_block4_3_

bn[0][0]']

 conv3_block4_out (Activati  (None, 19, 19, 512)          0         ['conv3_block4_ad
d[0][0]']
 on)

 conv4_block1_1_conv (Conv2  (None, 10, 10, 256)          131328    ['conv3_block4_ou
t[0][0]']
 D)

 conv4_block1_1_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block1_1_
conv[0][0]']
 rmalization)

 conv4_block1_1_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block1_1_
bn[0][0]']
 ation)

 conv4_block1_2_conv (Conv2  (None, 10, 10, 256)          590080    ['conv4_block1_1_
relu[0][0]']
 D)

 conv4_block1_2_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block1_2_
conv[0][0]']
 rmalization)

 conv4_block1_2_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block1_2_
bn[0][0]']
 ation)

 conv4_block1_0_conv (Conv2  (None, 10, 10, 1024)         525312    ['conv3_block4_ou
t[0][0]']
 D)

 conv4_block1_3_conv (Conv2  (None, 10, 10, 1024)         263168    ['conv4_block1_2_
relu[0][0]']
 D)
 conv4_block1_0_bn (BatchNo  (None, 10, 10, 1024)         4096      ['conv4_block1_0_
conv[0][0]']
 rmalization)

 conv4_block1_3_bn (BatchNo  (None, 10, 10, 1024)         4096      ['conv4_block1_3_
conv[0][0]']
 rmalization)

 conv4_block1_add (Add)      (None, 10, 10, 1024)         0         ['conv4_block1_0_
bn[0][0]',
                                                                     'conv4_block1_3_
bn[0][0]']

 conv4_block1_out (Activati  (None, 10, 10, 1024)         0         ['conv4_block1_ad
d[0][0]']
 on)

 conv4_block2_1_conv (Conv2  (None, 10, 10, 256)          262400    ['conv4_block1_ou
t[0][0]']
 D)

 conv4_block2_1_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block2_1_
conv[0][0]']
 rmalization)

 conv4_block2_1_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block2_1_
bn[0][0]']
 ation)

 conv4_block2_2_conv (Conv2  (None, 10, 10, 256)          590080    ['conv4_block2_1_
relu[0][0]']
 D)

 conv4_block2_2_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block2_2_
conv[0][0]']
 rmalization)

 conv4_block2_2_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block2_2_
bn[0][0]']
 ation)

 conv4_block2_3_conv (Conv2  (None, 10, 10, 1024)         263168    ['conv4_block2_2_
relu[0][0]']
 D)

 conv4_block2_3_bn (BatchNo  (None, 10, 10, 1024)         4096      ['conv4_block2_3_
conv[0][0]']
 rmalization)

 conv4_block2_add (Add)      (None, 10, 10, 1024)         0         ['conv4_block1_ou
t[0][0]',
                                                                     'conv4_block2_3_
bn[0][0]']

 conv4_block2_out (Activati  (None, 10, 10, 1024)         0         ['conv4_block2_ad
d[0][0]']
 on)

 conv4_block3_1_conv (Conv2  (None, 10, 10, 256)          262400    ['conv4_block2_ou
t[0][0]']
 D)

 conv4_block3_1_bn (BatchNo  (None, 10, 10, 256)          1024        ['conv4_block3_1_
conv[0][0]']
 rmalization)

 conv4_block3_1_relu (Activ  (None, 10, 10, 256)          0           ['conv4_block3_1_
bn[0][0]']
 ation)

 conv4_block3_2_conv (Conv2  (None, 10, 10, 256)          590080      ['conv4_block3_1_
relu[0][0]']
 D)

 conv4_block3_2_bn (BatchNo  (None, 10, 10, 256)          1024        ['conv4_block3_2_
conv[0][0]']
 rmalization)

 conv4_block3_2_relu (Activ  (None, 10, 10, 256)          0           ['conv4_block3_2_
bn[0][0]']
 ation)

 conv4_block3_3_conv (Conv2  (None, 10, 10, 1024)         263168      ['conv4_block3_2_
relu[0][0]']
 D)

 conv4_block3_3_bn (BatchNo  (None, 10, 10, 1024)         4096        ['conv4_block3_3_
conv[0][0]']
 rmalization)

 conv4_block3_add (Add)      (None, 10, 10, 1024)         0           ['conv4_block2_ou
t[0][0]',
                                                                       'conv4_block3_3_
bn[0][0]']

 conv4_block3_out (Activati  (None, 10, 10, 1024)         0           ['conv4_block3_ad
d[0][0]']
 on)

 conv4_block4_1_conv (Conv2  (None, 10, 10, 256)          262400      ['conv4_block3_ou
t[0][0]']
 D)

 conv4_block4_1_bn (BatchNo  (None, 10, 10, 256)          1024        ['conv4_block4_1_
conv[0][0]']
 rmalization)

 conv4_block4_1_relu (Activ  (None, 10, 10, 256)          0           ['conv4_block4_1_
bn[0][0]']
 ation)

 conv4_block4_2_conv (Conv2  (None, 10, 10, 256)          590080      ['conv4_block4_1_
relu[0][0]']
 D)

 conv4_block4_2_bn (BatchNo  (None, 10, 10, 256)          1024        ['conv4_block4_2_
conv[0][0]']
 rmalization)
 conv4_block4_2_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block4_2_
bn[0][0]']
 ation)

 conv4_block4_3_conv (Conv2  (None, 10, 10, 1024)         263168    ['conv4_block4_2_
relu[0][0]']
 D)

 conv4_block4_3_bn (BatchNo  (None, 10, 10, 1024)         4096      ['conv4_block4_3_
conv[0][0]']
 rmalization)

 conv4_block4_add (Add)      (None, 10, 10, 1024)         0         ['conv4_block3_ou
t[0][0]',

                                                                     'conv4_block4_3_
bn[0][0]']

 conv4_block4_out (Activati  (None, 10, 10, 1024)         0         ['conv4_block4_ad
d[0][0]']
 on)

 conv4_block5_1_conv (Conv2  (None, 10, 10, 256)          262400    ['conv4_block4_ou
t[0][0]']
 D)

 conv4_block5_1_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block5_1_
conv[0][0]']
 rmalization)

 conv4_block5_1_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block5_1_
bn[0][0]']
 ation)

 conv4_block5_2_conv (Conv2  (None, 10, 10, 256)          590080    ['conv4_block5_1_
relu[0][0]']
 D)

 conv4_block5_2_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block5_2_
conv[0][0]']
 rmalization)

 conv4_block5_2_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block5_2_
bn[0][0]']
 ation)

 conv4_block5_3_conv (Conv2  (None, 10, 10, 1024)         263168    ['conv4_block5_2_
relu[0][0]']
 D)

 conv4_block5_3_bn (BatchNo  (None, 10, 10, 1024)         4096      ['conv4_block5_3_
conv[0][0]']
 rmalization)

 conv4_block5_add (Add)      (None, 10, 10, 1024)         0         ['conv4_block4_ou
t[0][0]',

                                                                     'conv4_block5_3_
bn[0][0]']

 conv4_block5_out (Activati  (None, 10, 10, 1024)         0         ['conv4_block5_ad
d[0][0]']
```

```
 on)

 conv4_block6_1_conv (Conv2  (None, 10, 10, 256)          262400    ['conv4_block5_ou
t[0][0]']
 D)

 conv4_block6_1_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block6_1_
conv[0][0]']
 rmalization)

 conv4_block6_1_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block6_1_
bn[0][0]']
 ation)

 conv4_block6_2_conv (Conv2  (None, 10, 10, 256)          590080    ['conv4_block6_1_
relu[0][0]']
 D)

 conv4_block6_2_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block6_2_
conv[0][0]']
 rmalization)

 conv4_block6_2_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block6_2_
bn[0][0]']
 ation)

 conv4_block6_3_conv (Conv2  (None, 10, 10, 1024)         263168    ['conv4_block6_2_
relu[0][0]']
 D)

 conv4_block6_3_bn (BatchNo  (None, 10, 10, 1024)         4096      ['conv4_block6_3_
conv[0][0]']
 rmalization)

 conv4_block6_add (Add)      (None, 10, 10, 1024)         0         ['conv4_block5_ou
t[0][0]',

                                                                     'conv4_block6_3_

bn[0][0]']

 conv4_block6_out (Activati  (None, 10, 10, 1024)         0         ['conv4_block6_ad
d[0][0]']
 on)

 conv5_block1_1_conv (Conv2  (None, 5, 5, 512)            524800    ['conv4_block6_ou
t[0][0]']
 D)

 conv5_block1_1_bn (BatchNo  (None, 5, 5, 512)            2048      ['conv5_block1_1_
conv[0][0]']
 rmalization)

 conv5_block1_1_relu (Activ  (None, 5, 5, 512)            0         ['conv5_block1_1_
bn[0][0]']
 ation)

 conv5_block1_2_conv (Conv2  (None, 5, 5, 512)            2359808   ['conv5_block1_1_
relu[0][0]']
 D)

 conv5_block1_2_bn (BatchNo  (None, 5, 5, 512)            2048      ['conv5_block1_2_
conv[0][0]']
 rmalization)

 conv5_block1_2_relu (Activ  (None, 5, 5, 512)           0         ['conv5_block1_2_
bn[0][0]']
 ation)

 conv5_block1_0_conv (Conv2  (None, 5, 5, 2048)          2099200   ['conv4_block6_ou
t[0][0]']
 D)

 conv5_block1_3_conv (Conv2  (None, 5, 5, 2048)          1050624   ['conv5_block1_2_
relu[0][0]']
 D)

 conv5_block1_0_bn (BatchNo  (None, 5, 5, 2048)          8192      ['conv5_block1_0_
conv[0][0]']
 rmalization)

 conv5_block1_3_bn (BatchNo  (None, 5, 5, 2048)          8192      ['conv5_block1_3_
conv[0][0]']
 rmalization)

 conv5_block1_add (Add)      (None, 5, 5, 2048)          0         ['conv5_block1_0_
bn[0][0]',

                                                                     'conv5_block1_3_
bn[0][0]']

 conv5_block1_out (Activati  (None, 5, 5, 2048)          0         ['conv5_block1_ad
d[0][0]']
 on)

 conv5_block2_1_conv (Conv2  (None, 5, 5, 512)           1049088   ['conv5_block1_ou
t[0][0]']
 D)

 conv5_block2_1_bn (BatchNo  (None, 5, 5, 512)           2048      ['conv5_block2_1_
conv[0][0]']
 rmalization)

 conv5_block2_1_relu (Activ  (None, 5, 5, 512)           0         ['conv5_block2_1_
bn[0][0]']
 ation)

 conv5_block2_2_conv (Conv2  (None, 5, 5, 512)           2359808   ['conv5_block2_1_
relu[0][0]']
 D)

 conv5_block2_2_bn (BatchNo  (None, 5, 5, 512)           2048      ['conv5_block2_2_
conv[0][0]']
 rmalization)

 conv5_block2_2_relu (Activ  (None, 5, 5, 512)           0         ['conv5_block2_2_
bn[0][0]']
 ation)

 conv5_block2_3_conv (Conv2  (None, 5, 5, 2048)          1050624   ['conv5_block2_2_
relu[0][0]']
 D)
 conv5_block2_3_bn (BatchNo  (None, 5, 5, 2048)          8192      ['conv5_block2_3_
conv[0][0]']
 rmalization)

 conv5_block2_add (Add)      (None, 5, 5, 2048)          0         ['conv5_block1_ou
t[0][0]',
                                                                    'conv5_block2_3_
bn[0][0]']

 conv5_block2_out (Activati  (None, 5, 5, 2048)          0         ['conv5_block2_ad
d[0][0]']
 on)

 conv5_block3_1_conv (Conv2  (None, 5, 5, 512)           1049088   ['conv5_block2_ou
t[0][0]']
 D)

 conv5_block3_1_bn (BatchNo  (None, 5, 5, 512)           2048      ['conv5_block3_1_
conv[0][0]']
 rmalization)

 conv5_block3_1_relu (Activ  (None, 5, 5, 512)           0         ['conv5_block3_1_
bn[0][0]']
 ation)

 conv5_block3_2_conv (Conv2  (None, 5, 5, 512)           2359808   ['conv5_block3_1_
relu[0][0]']
 D)

 conv5_block3_2_bn (BatchNo  (None, 5, 5, 512)           2048      ['conv5_block3_2_
conv[0][0]']
 rmalization)

 conv5_block3_2_relu (Activ  (None, 5, 5, 512)           0         ['conv5_block3_2_
bn[0][0]']
 ation)

 conv5_block3_3_conv (Conv2  (None, 5, 5, 2048)          1050624   ['conv5_block3_2_
relu[0][0]']
 D)

 conv5_block3_3_bn (BatchNo  (None, 5, 5, 2048)          8192      ['conv5_block3_3_
conv[0][0]']
 rmalization)

 conv5_block3_add (Add)      (None, 5, 5, 2048)          0         ['conv5_block2_ou
t[0][0]',
                                                                    'conv5_block3_3_
bn[0][0]']

 conv5_block3_out (Activati  (None, 5, 5, 2048)          0         ['conv5_block3_ad
d[0][0]']
 on)

 flatten_1 (Flatten)         (None, 51200)               0         ['conv5_block3_ou
t[0][0]']

 dense_2 (Dense)             (None, 256)                 1310745   ['flatten_1[0]
[0]']
                                                         6
 dense_3 (Dense)             (None, 1)                    257       ['dense_2[0][0]']

==================================================================================================
Total params: 36695425 (139.98 MB)
Trainable params: 13107713 (50.00 MB)
Non-trainable params: 23587712 (89.98 MB)
__________________________________________________________________________________________________
```

For Training Sample of Size 1000

In [43]:

```python
# Clear existing data in the training directory
shutil.rmtree(train_dir)
os.makedirs(train_dir, exist_ok=True)

train_cats_dir = os.path.join(train_dir, 'cat')
os.makedirs(train_cats_dir, exist_ok=True)
train_dogs_dir = os.path.join(train_dir, 'dog')
os.makedirs(train_dogs_dir, exist_ok=True)

# Assign 500 cat and 500 dog images to the training directory
def copy_images(source_dir, destination_dir, images):
    for image in images:
        shutil.copy(os.path.join(source_dir, image), destination_dir)

# Randomize the order of cat and dog images
random.shuffle(cat_images)
random.shuffle(dog_images)

# Copy images to training directory
copy_images(cat_folder_path, train_cats_dir, cat_images[:500])
copy_images(dog_folder_path, train_dogs_dir, dog_images[:500])
```

In [45]:

```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Define data generators for training, validation, and testing
batch_size = 20
image_size = (150, 150)

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='binary'
)
```

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

Runnning The Model

In [46]:

```
# Train the model
history = model.fit(
    train_generator,
    steps_per_epoch=1000 // batch_size,  # 1000 images in total (500 cat + 500 dog)
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=validation_samples // batch_size
)
```

```
Epoch 1/10
50/50 [==============================] - 105s 2s/step - loss: 1.0692 - accuracy: 0.50
50 - val_loss: 0.6885 - val_accuracy: 0.5000
Epoch 2/10
50/50 [==============================] - 101s 2s/step - loss: 0.6928 - accuracy: 0.51
80 - val_loss: 0.6672 - val_accuracy: 0.5940
Epoch 3/10
50/50 [==============================] - 86s 2s/step - loss: 0.6998 - accuracy: 0.523
0 - val_loss: 0.7027 - val_accuracy: 0.5000
Epoch 4/10
50/50 [==============================] - 101s 2s/step - loss: 0.6888 - accuracy: 0.53
50 - val_loss: 0.6523 - val_accuracy: 0.6160
Epoch 5/10
50/50 [==============================] - 87s 2s/step - loss: 0.6762 - accuracy: 0.581
0 - val_loss: 0.6677 - val_accuracy: 0.5400
Epoch 6/10
50/50 [==============================] - 101s 2s/step - loss: 0.6999 - accuracy: 0.55
90 - val_loss: 0.6537 - val_accuracy: 0.6160
Epoch 7/10
50/50 [==============================] - 86s 2s/step - loss: 0.6864 - accuracy: 0.553
0 - val_loss: 0.6449 - val_accuracy: 0.6460
Epoch 8/10
50/50 [==============================] - 86s 2s/step - loss: 0.6802 - accuracy: 0.571
0 - val_loss: 0.6518 - val_accuracy: 0.6340
Epoch 9/10
50/50 [==============================] - 101s 2s/step - loss: 0.6785 - accuracy: 0.56
60 - val_loss: 0.6753 - val_accuracy: 0.5100
Epoch 10/10
50/50 [==============================] - 101s 2s/step - loss: 0.6807 - accuracy: 0.56
30 - val_loss: 0.7005 - val_accuracy: 0.5580
```

In [47]:

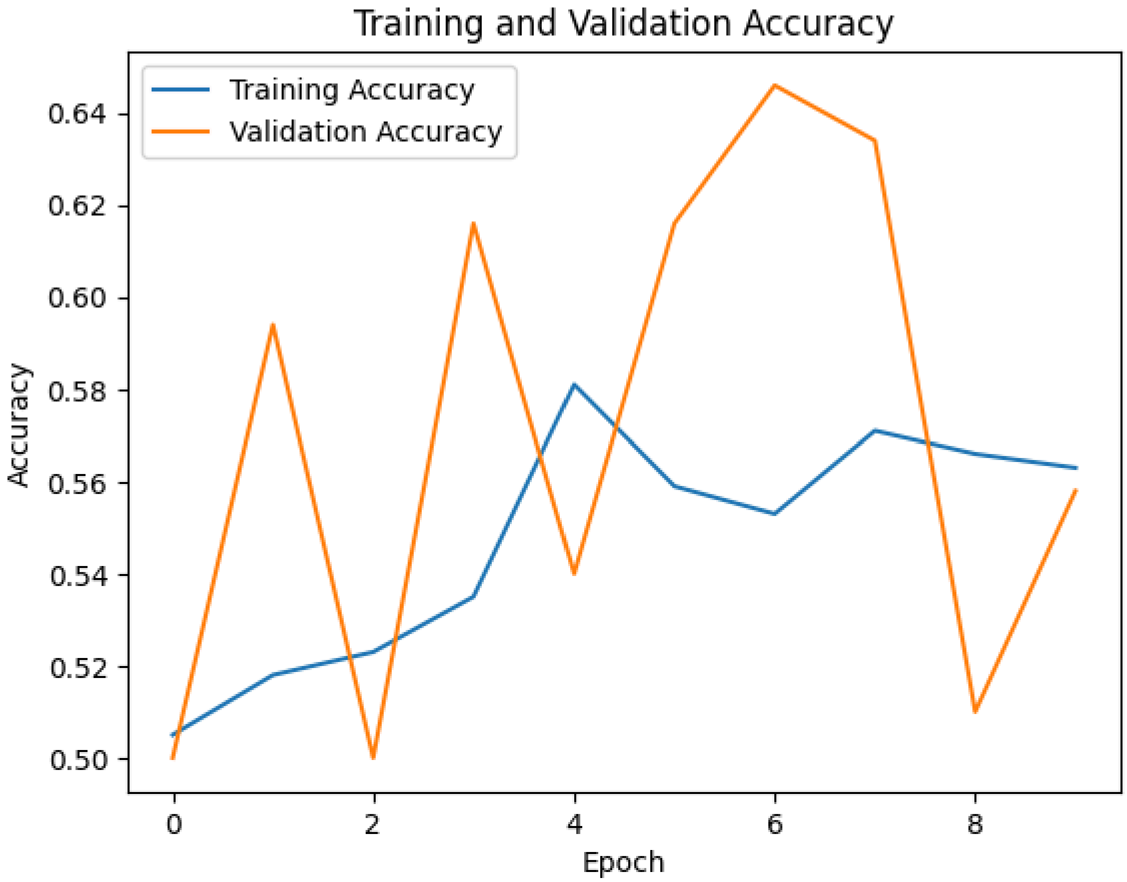
```
# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(test_generator, steps=test_samples // batch_
print("Test accuracy:", test_accuracy)
```

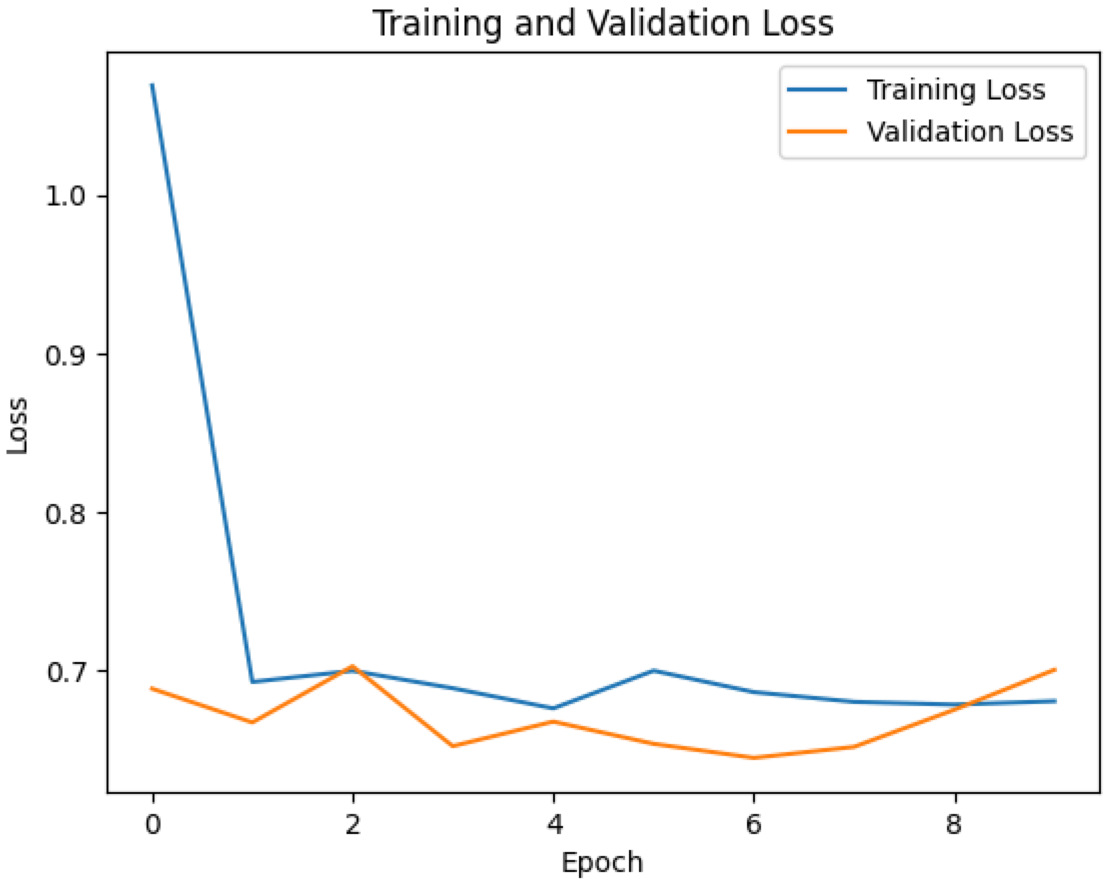
```
25/25 [==============================] - 27s 1s/step - loss: 0.7303 - accuracy: 0.520
0
Test accuracy: 0.5199999809265137
```

Perfomance Metrics

In [50]:

```
# Plot training and validation accuracy
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
# Plot training and validation loss
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```





For Training Samples of size 1500

In [49]:

```
# Clear existing data in the training directory
shutil.rmtree(train_dir)
os.makedirs(train_dir, exist_ok=True)

train_cats_dir = os.path.join(train_dir, 'cat')
os.makedirs(train_cats_dir, exist_ok=True)
train_dogs_dir = os.path.join(train_dir, 'dog')
os.makedirs(train_dogs_dir, exist_ok=True)

# Assign 500 cat and 500 dog images to the training directory
def copy_images(source_dir, destination_dir, images):
    for image in images:
        shutil.copy(os.path.join(source_dir, image), destination_dir)

# Randomize the order of cat and dog images
random.shuffle(cat_images)
random.shuffle(dog_images)

# Copy images to training directory
copy_images(cat_folder_path, train_cats_dir, cat_images[:750])
copy_images(dog_folder_path, train_dogs_dir, dog_images[:750])

from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Define data generators for training, validation, and testing
batch_size = 20
image_size = (150, 150)
```

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='binary'
)
```

```
Found 1500 images belonging to 2 classes.
```

In [51]:

```
# Freeze the weights of the pre-trained layers
for layer in base_model.layers:
    layer.trainable = False

# Add custom classification layers
x = Flatten()(base_model.output)
x = Dense(256, activation='relu')(x)
output = Dense(1, activation='sigmoid')(x)

# Create the model
model = Model(inputs=base_model.input, outputs=output)

# Compile the model
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

model.summary()
```

```
Model: "model_1"
_____________________________________________________________________________________
_____________
 Layer (type)                Output Shape                 Param #   Connected to
=====================================================================================
=============
 input_1 (InputLayer)        [(None, 150, 150, 3)]        0         []

 conv1_pad (ZeroPadding2D)   (None, 156, 156, 3)          0         ['input_1[0][0]']

 conv1_conv (Conv2D)         (None, 75, 75, 64)           9472      ['conv1_pad[0]
[0]']

 conv1_bn (BatchNormalizati  (None, 75, 75, 64)           256       ['conv1_conv[0]
[0]']
 on)

 conv1_relu (Activation)     (None, 75, 75, 64)           0         ['conv1_bn[0]
[0]']

 pool1_pad (ZeroPadding2D)   (None, 77, 77, 64)           0         ['conv1_relu[0]
[0]']

 pool1_pool (MaxPooling2D)   (None, 38, 38, 64)           0         ['pool1_pad[0]
[0]']

 conv2_block1_1_conv (Conv2  (None, 38, 38, 64)           4160      ['pool1_pool[0]
[0]']
 D)

 conv2_block1_1_bn (BatchNo  (None, 38, 38, 64)           256       ['conv2_block1_1_
conv[0][0]']
 rmalization)

 conv2_block1_1_relu (Activ  (None, 38, 38, 64)           0         ['conv2_block1_1_
bn[0][0]']
 ation)

 conv2_block1_2_conv (Conv2  (None, 38, 38, 64)           36928     ['conv2_block1_1_
relu[0][0]']
 D)

 conv2_block1_2_bn (BatchNo  (None, 38, 38, 64)           256       ['conv2_block1_2_
conv[0][0]']
 rmalization)

 conv2_block1_2_relu (Activ  (None, 38, 38, 64)           0         ['conv2_block1_2_
bn[0][0]']
 ation)

 conv2_block1_0_conv (Conv2  (None, 38, 38, 256)          16640     ['pool1_pool[0]
[0]']
 D)

 conv2_block1_3_conv (Conv2  (None, 38, 38, 256)          16640     ['conv2_block1_2_
relu[0][0]']
 D)

 conv2_block1_0_bn (BatchNo  (None, 38, 38, 256)          1024      ['conv2_block1_0_
conv[0][0]']
```

```
 rmalization)

 conv2_block1_3_bn (BatchNo  (None, 38, 38, 256)          1024      ['conv2_block1_3_
conv[0][0]']
 rmalization)

 conv2_block1_add (Add)      (None, 38, 38, 256)          0         ['conv2_block1_0_
bn[0][0]',
                                                                     'conv2_block1_3_
bn[0][0]']

 conv2_block1_out (Activati  (None, 38, 38, 256)          0         ['conv2_block1_ad
d[0][0]']
 on)

 conv2_block2_1_conv (Conv2  (None, 38, 38, 64)           16448     ['conv2_block1_ou
t[0][0]']
 D)

 conv2_block2_1_bn (BatchNo  (None, 38, 38, 64)           256       ['conv2_block2_1_
conv[0][0]']
 rmalization)

 conv2_block2_1_relu (Activ  (None, 38, 38, 64)           0         ['conv2_block2_1_
bn[0][0]']
 ation)

 conv2_block2_2_conv (Conv2  (None, 38, 38, 64)           36928     ['conv2_block2_1_
relu[0][0]']
 D)

 conv2_block2_2_bn (BatchNo  (None, 38, 38, 64)           256       ['conv2_block2_2_
conv[0][0]']
 rmalization)

 conv2_block2_2_relu (Activ  (None, 38, 38, 64)           0         ['conv2_block2_2_
bn[0][0]']
 ation)

 conv2_block2_3_conv (Conv2  (None, 38, 38, 256)          16640     ['conv2_block2_2_
relu[0][0]']
 D)

 conv2_block2_3_bn (BatchNo  (None, 38, 38, 256)          1024      ['conv2_block2_3_
conv[0][0]']
 rmalization)

 conv2_block2_add (Add)      (None, 38, 38, 256)          0         ['conv2_block1_ou
t[0][0]',
                                                                     'conv2_block2_3_
bn[0][0]']

 conv2_block2_out (Activati  (None, 38, 38, 256)          0         ['conv2_block2_ad
d[0][0]']
 on)

 conv2_block3_1_conv (Conv2  (None, 38, 38, 64)           16448     ['conv2_block2_ou
t[0][0]']
 D)
 conv2_block3_1_bn (BatchNo  (None, 38, 38, 64)          256       ['conv2_block3_1_
conv[0][0]']
 rmalization)

 conv2_block3_1_relu (Activ  (None, 38, 38, 64)          0         ['conv2_block3_1_
bn[0][0]']
 ation)

 conv2_block3_2_conv (Conv2  (None, 38, 38, 64)          36928     ['conv2_block3_1_
relu[0][0]']
 D)

 conv2_block3_2_bn (BatchNo  (None, 38, 38, 64)          256       ['conv2_block3_2_
conv[0][0]']
 rmalization)

 conv2_block3_2_relu (Activ  (None, 38, 38, 64)          0         ['conv2_block3_2_
bn[0][0]']
 ation)

 conv2_block3_3_conv (Conv2  (None, 38, 38, 256)         16640     ['conv2_block3_2_
relu[0][0]']
 D)

 conv2_block3_3_bn (BatchNo  (None, 38, 38, 256)         1024      ['conv2_block3_3_
conv[0][0]']
 rmalization)

 conv2_block3_add (Add)      (None, 38, 38, 256)         0         ['conv2_block2_ou
t[0][0]',

                                                                    'conv2_block3_3_

bn[0][0]']

 conv2_block3_out (Activati  (None, 38, 38, 256)         0         ['conv2_block3_ad
d[0][0]']
 on)

 conv3_block1_1_conv (Conv2  (None, 19, 19, 128)         32896     ['conv2_block3_ou
t[0][0]']
 D)

 conv3_block1_1_bn (BatchNo  (None, 19, 19, 128)         512       ['conv3_block1_1_
conv[0][0]']
 rmalization)

 conv3_block1_1_relu (Activ  (None, 19, 19, 128)         0         ['conv3_block1_1_
bn[0][0]']
 ation)

 conv3_block1_2_conv (Conv2  (None, 19, 19, 128)         147584    ['conv3_block1_1_
relu[0][0]']
 D)

 conv3_block1_2_bn (BatchNo  (None, 19, 19, 128)         512       ['conv3_block1_2_
conv[0][0]']
 rmalization)

 conv3_block1_2_relu (Activ  (None, 19, 19, 128)         0         ['conv3_block1_2_
bn[0][0]']
 ation)
 conv3_block1_0_conv (Conv2  (None, 19, 19, 512)          131584    ['conv2_block3_ou
t[0][0]']
 D)

 conv3_block1_3_conv (Conv2  (None, 19, 19, 512)          66048     ['conv3_block1_2_
relu[0][0]']
 D)

 conv3_block1_0_bn (BatchNo  (None, 19, 19, 512)          2048      ['conv3_block1_0_
conv[0][0]']
 rmalization)

 conv3_block1_3_bn (BatchNo  (None, 19, 19, 512)          2048      ['conv3_block1_3_
conv[0][0]']
 rmalization)

 conv3_block1_add (Add)      (None, 19, 19, 512)          0         ['conv3_block1_0_
bn[0][0]',
                                                                     'conv3_block1_3_
bn[0][0]']

 conv3_block1_out (Activati  (None, 19, 19, 512)          0         ['conv3_block1_ad
d[0][0]']
 on)

 conv3_block2_1_conv (Conv2  (None, 19, 19, 128)          65664     ['conv3_block1_ou
t[0][0]']
 D)

 conv3_block2_1_bn (BatchNo  (None, 19, 19, 128)          512       ['conv3_block2_1_
conv[0][0]']
 rmalization)

 conv3_block2_1_relu (Activ  (None, 19, 19, 128)          0         ['conv3_block2_1_
bn[0][0]']
 ation)

 conv3_block2_2_conv (Conv2  (None, 19, 19, 128)          147584    ['conv3_block2_1_
relu[0][0]']
 D)

 conv3_block2_2_bn (BatchNo  (None, 19, 19, 128)          512       ['conv3_block2_2_
conv[0][0]']
 rmalization)

 conv3_block2_2_relu (Activ  (None, 19, 19, 128)          0         ['conv3_block2_2_
bn[0][0]']
 ation)

 conv3_block2_3_conv (Conv2  (None, 19, 19, 512)          66048     ['conv3_block2_2_
relu[0][0]']
 D)

 conv3_block2_3_bn (BatchNo  (None, 19, 19, 512)          2048      ['conv3_block2_3_
conv[0][0]']
 rmalization)

 conv3_block2_add (Add)      (None, 19, 19, 512)          0         ['conv3_block1_ou
t[0][0]',
                                                                     'conv3_block2_3_
bn[0][0]']

 conv3_block2_out (Activati  (None, 19, 19, 512)       0         ['conv3_block2_ad
d[0][0]']
 on)

 conv3_block3_1_conv (Conv2  (None, 19, 19, 128)       65664     ['conv3_block2_ou
t[0][0]']
 D)

 conv3_block3_1_bn (BatchNo  (None, 19, 19, 128)       512       ['conv3_block3_1_
conv[0][0]']
 rmalization)

 conv3_block3_1_relu (Activ  (None, 19, 19, 128)       0         ['conv3_block3_1_
bn[0][0]']
 ation)

 conv3_block3_2_conv (Conv2  (None, 19, 19, 128)       147584    ['conv3_block3_1_
relu[0][0]']
 D)

 conv3_block3_2_bn (BatchNo  (None, 19, 19, 128)       512       ['conv3_block3_2_
conv[0][0]']
 rmalization)

 conv3_block3_2_relu (Activ  (None, 19, 19, 128)       0         ['conv3_block3_2_
bn[0][0]']
 ation)

 conv3_block3_3_conv (Conv2  (None, 19, 19, 512)       66048     ['conv3_block3_2_
relu[0][0]']
 D)

 conv3_block3_3_bn (BatchNo  (None, 19, 19, 512)       2048      ['conv3_block3_3_
conv[0][0]']
 rmalization)

 conv3_block3_add (Add)      (None, 19, 19, 512)       0         ['conv3_block2_ou
t[0][0]',
                                                                     'conv3_block3_3_
bn[0][0]']

 conv3_block3_out (Activati  (None, 19, 19, 512)       0         ['conv3_block3_ad
d[0][0]']
 on)

 conv3_block4_1_conv (Conv2  (None, 19, 19, 128)       65664     ['conv3_block3_ou
t[0][0]']
 D)

 conv3_block4_1_bn (BatchNo  (None, 19, 19, 128)       512       ['conv3_block4_1_
conv[0][0]']
 rmalization)

 conv3_block4_1_relu (Activ  (None, 19, 19, 128)       0         ['conv3_block4_1_
bn[0][0]']
 ation)
 conv3_block4_2_conv (Conv2  (None, 19, 19, 128)          147584    ['conv3_block4_1_
relu[0][0]']
 D)

 conv3_block4_2_bn (BatchNo  (None, 19, 19, 128)          512       ['conv3_block4_2_
conv[0][0]']
 rmalization)

 conv3_block4_2_relu (Activ  (None, 19, 19, 128)          0         ['conv3_block4_2_
bn[0][0]']
 ation)

 conv3_block4_3_conv (Conv2  (None, 19, 19, 512)          66048     ['conv3_block4_2_
relu[0][0]']
 D)

 conv3_block4_3_bn (BatchNo  (None, 19, 19, 512)          2048      ['conv3_block4_3_
conv[0][0]']
 rmalization)

 conv3_block4_add (Add)      (None, 19, 19, 512)          0         ['conv3_block3_ou
t[0][0]',
                                                                     'conv3_block4_3_
bn[0][0]']

 conv3_block4_out (Activati  (None, 19, 19, 512)          0         ['conv3_block4_ad
d[0][0]']
 on)

 conv4_block1_1_conv (Conv2  (None, 10, 10, 256)          131328    ['conv3_block4_ou
t[0][0]']
 D)

 conv4_block1_1_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block1_1_
conv[0][0]']
 rmalization)

 conv4_block1_1_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block1_1_
bn[0][0]']
 ation)

 conv4_block1_2_conv (Conv2  (None, 10, 10, 256)          590080    ['conv4_block1_1_
relu[0][0]']
 D)

 conv4_block1_2_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block1_2_
conv[0][0]']
 rmalization)

 conv4_block1_2_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block1_2_
bn[0][0]']
 ation)

 conv4_block1_0_conv (Conv2  (None, 10, 10, 1024)         525312    ['conv3_block4_ou
t[0][0]']
 D)

 conv4_block1_3_conv (Conv2  (None, 10, 10, 1024)         263168    ['conv4_block1_2_
relu[0][0]']
 D)
 conv4_block1_0_bn (BatchNo  (None, 10, 10, 1024)         4096      ['conv4_block1_0_
conv[0][0]']
 rmalization)

 conv4_block1_3_bn (BatchNo  (None, 10, 10, 1024)         4096      ['conv4_block1_3_
conv[0][0]']
 rmalization)

 conv4_block1_add (Add)      (None, 10, 10, 1024)         0         ['conv4_block1_0_
bn[0][0]',
                                                                     'conv4_block1_3_
bn[0][0]']

 conv4_block1_out (Activati  (None, 10, 10, 1024)         0         ['conv4_block1_ad
d[0][0]']
 on)

 conv4_block2_1_conv (Conv2  (None, 10, 10, 256)          262400    ['conv4_block1_ou
t[0][0]']
 D)

 conv4_block2_1_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block2_1_
conv[0][0]']
 rmalization)

 conv4_block2_1_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block2_1_
bn[0][0]']
 ation)

 conv4_block2_2_conv (Conv2  (None, 10, 10, 256)          590080    ['conv4_block2_1_
relu[0][0]']
 D)

 conv4_block2_2_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block2_2_
conv[0][0]']
 rmalization)

 conv4_block2_2_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block2_2_
bn[0][0]']
 ation)

 conv4_block2_3_conv (Conv2  (None, 10, 10, 1024)         263168    ['conv4_block2_2_
relu[0][0]']
 D)

 conv4_block2_3_bn (BatchNo  (None, 10, 10, 1024)         4096      ['conv4_block2_3_
conv[0][0]']
 rmalization)

 conv4_block2_add (Add)      (None, 10, 10, 1024)         0         ['conv4_block1_ou
t[0][0]',
                                                                     'conv4_block2_3_
bn[0][0]']

 conv4_block2_out (Activati  (None, 10, 10, 1024)         0         ['conv4_block2_ad
d[0][0]']
 on)

 conv4_block3_1_conv (Conv2  (None, 10, 10, 256)          262400    ['conv4_block2_ou
t[0][0]']
 D)

 conv4_block3_1_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block3_1_
conv[0][0]']
 rmalization)

 conv4_block3_1_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block3_1_
bn[0][0]']
 ation)

 conv4_block3_2_conv (Conv2  (None, 10, 10, 256)          590080    ['conv4_block3_1_
relu[0][0]']
 D)

 conv4_block3_2_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block3_2_
conv[0][0]']
 rmalization)

 conv4_block3_2_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block3_2_
bn[0][0]']
 ation)

 conv4_block3_3_conv (Conv2  (None, 10, 10, 1024)         263168    ['conv4_block3_2_
relu[0][0]']
 D)

 conv4_block3_3_bn (BatchNo  (None, 10, 10, 1024)         4096      ['conv4_block3_3_
conv[0][0]']
 rmalization)

 conv4_block3_add (Add)      (None, 10, 10, 1024)         0         ['conv4_block2_ou
t[0][0]',

                                                                     'conv4_block3_3_
bn[0][0]']

 conv4_block3_out (Activati  (None, 10, 10, 1024)         0         ['conv4_block3_ad
d[0][0]']
 on)

 conv4_block4_1_conv (Conv2  (None, 10, 10, 256)          262400    ['conv4_block3_ou
t[0][0]']
 D)

 conv4_block4_1_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block4_1_
conv[0][0]']
 rmalization)

 conv4_block4_1_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block4_1_
bn[0][0]']
 ation)

 conv4_block4_2_conv (Conv2  (None, 10, 10, 256)          590080    ['conv4_block4_1_
relu[0][0]']
 D)

 conv4_block4_2_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block4_2_
conv[0][0]']
 rmalization)
 conv4_block4_2_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block4_2_
bn[0][0]']
 ation)

 conv4_block4_3_conv (Conv2  (None, 10, 10, 1024)         263168    ['conv4_block4_2_
relu[0][0]']
 D)

 conv4_block4_3_bn (BatchNo  (None, 10, 10, 1024)         4096      ['conv4_block4_3_
conv[0][0]']
 rmalization)

 conv4_block4_add (Add)      (None, 10, 10, 1024)         0         ['conv4_block3_ou
t[0][0]',

                                                                     'conv4_block4_3_
bn[0][0]']

 conv4_block4_out (Activati  (None, 10, 10, 1024)         0         ['conv4_block4_ad
d[0][0]']
 on)

 conv4_block5_1_conv (Conv2  (None, 10, 10, 256)          262400    ['conv4_block4_ou
t[0][0]']
 D)

 conv4_block5_1_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block5_1_
conv[0][0]']
 rmalization)

 conv4_block5_1_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block5_1_
bn[0][0]']
 ation)

 conv4_block5_2_conv (Conv2  (None, 10, 10, 256)          590080    ['conv4_block5_1_
relu[0][0]']
 D)

 conv4_block5_2_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block5_2_
conv[0][0]']
 rmalization)

 conv4_block5_2_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block5_2_
bn[0][0]']
 ation)

 conv4_block5_3_conv (Conv2  (None, 10, 10, 1024)         263168    ['conv4_block5_2_
relu[0][0]']
 D)

 conv4_block5_3_bn (BatchNo  (None, 10, 10, 1024)         4096      ['conv4_block5_3_
conv[0][0]']
 rmalization)

 conv4_block5_add (Add)      (None, 10, 10, 1024)         0         ['conv4_block4_ou
t[0][0]',

                                                                     'conv4_block5_3_
bn[0][0]']

 conv4_block5_out (Activati  (None, 10, 10, 1024)         0         ['conv4_block5_ad
d[0][0]']
```

```
 on)

 conv4_block6_1_conv (Conv2  (None, 10, 10, 256)          262400    ['conv4_block5_ou
t[0][0]']
 D)

 conv4_block6_1_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block6_1_
conv[0][0]']
 rmalization)

 conv4_block6_1_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block6_1_
bn[0][0]']
 ation)

 conv4_block6_2_conv (Conv2  (None, 10, 10, 256)          590080    ['conv4_block6_1_
relu[0][0]']
 D)

 conv4_block6_2_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block6_2_
conv[0][0]']
 rmalization)

 conv4_block6_2_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block6_2_
bn[0][0]']
 ation)

 conv4_block6_3_conv (Conv2  (None, 10, 10, 1024)         263168    ['conv4_block6_2_
relu[0][0]']
 D)

 conv4_block6_3_bn (BatchNo  (None, 10, 10, 1024)         4096      ['conv4_block6_3_
conv[0][0]']
 rmalization)

 conv4_block6_add (Add)      (None, 10, 10, 1024)         0         ['conv4_block5_ou
t[0][0]',
                                                                     'conv4_block6_3_

bn[0][0]']

 conv4_block6_out (Activati  (None, 10, 10, 1024)         0         ['conv4_block6_ad
d[0][0]']
 on)

 conv5_block1_1_conv (Conv2  (None, 5, 5, 512)            524800    ['conv4_block6_ou
t[0][0]']
 D)

 conv5_block1_1_bn (BatchNo  (None, 5, 5, 512)            2048      ['conv5_block1_1_
conv[0][0]']
 rmalization)

 conv5_block1_1_relu (Activ  (None, 5, 5, 512)            0         ['conv5_block1_1_
bn[0][0]']
 ation)

 conv5_block1_2_conv (Conv2  (None, 5, 5, 512)            2359808   ['conv5_block1_1_
relu[0][0]']
 D)

 conv5_block1_2_bn (BatchNo  (None, 5, 5, 512)            2048      ['conv5_block1_2_
conv[0][0]']
 rmalization)

 conv5_block1_2_relu (Activ  (None, 5, 5, 512)           0         ['conv5_block1_2_
bn[0][0]']
 ation)

 conv5_block1_0_conv (Conv2  (None, 5, 5, 2048)          2099200   ['conv4_block6_ou
t[0][0]']
 D)

 conv5_block1_3_conv (Conv2  (None, 5, 5, 2048)          1050624   ['conv5_block1_2_
relu[0][0]']
 D)

 conv5_block1_0_bn (BatchNo  (None, 5, 5, 2048)          8192      ['conv5_block1_0_
conv[0][0]']
 rmalization)

 conv5_block1_3_bn (BatchNo  (None, 5, 5, 2048)          8192      ['conv5_block1_3_
conv[0][0]']
 rmalization)

 conv5_block1_add (Add)      (None, 5, 5, 2048)          0         ['conv5_block1_0_
bn[0][0]',

                                                                    'conv5_block1_3_
bn[0][0]']

 conv5_block1_out (Activati  (None, 5, 5, 2048)          0         ['conv5_block1_ad
d[0][0]']
 on)

 conv5_block2_1_conv (Conv2  (None, 5, 5, 512)           1049088   ['conv5_block1_ou
t[0][0]']
 D)

 conv5_block2_1_bn (BatchNo  (None, 5, 5, 512)           2048      ['conv5_block2_1_
conv[0][0]']
 rmalization)

 conv5_block2_1_relu (Activ  (None, 5, 5, 512)           0         ['conv5_block2_1_
bn[0][0]']
 ation)

 conv5_block2_2_conv (Conv2  (None, 5, 5, 512)           2359808   ['conv5_block2_1_
relu[0][0]']
 D)

 conv5_block2_2_bn (BatchNo  (None, 5, 5, 512)           2048      ['conv5_block2_2_
conv[0][0]']
 rmalization)

 conv5_block2_2_relu (Activ  (None, 5, 5, 512)           0         ['conv5_block2_2_
bn[0][0]']
 ation)

 conv5_block2_3_conv (Conv2  (None, 5, 5, 2048)          1050624   ['conv5_block2_2_
relu[0][0]']
 D)
 conv5_block2_3_bn (BatchNo  (None, 5, 5, 2048)         8192      ['conv5_block2_3_
conv[0][0]']
 rmalization)

 conv5_block2_add (Add)      (None, 5, 5, 2048)         0         ['conv5_block1_ou
t[0][0]',
                                                                   'conv5_block2_3_
bn[0][0]']

 conv5_block2_out (Activati  (None, 5, 5, 2048)         0         ['conv5_block2_ad
d[0][0]']
 on)

 conv5_block3_1_conv (Conv2  (None, 5, 5, 512)          1049088   ['conv5_block2_ou
t[0][0]']
 D)

 conv5_block3_1_bn (BatchNo  (None, 5, 5, 512)          2048      ['conv5_block3_1_
conv[0][0]']
 rmalization)

 conv5_block3_1_relu (Activ  (None, 5, 5, 512)          0         ['conv5_block3_1_
bn[0][0]']
 ation)

 conv5_block3_2_conv (Conv2  (None, 5, 5, 512)          2359808   ['conv5_block3_1_
relu[0][0]']
 D)

 conv5_block3_2_bn (BatchNo  (None, 5, 5, 512)          2048      ['conv5_block3_2_
conv[0][0]']
 rmalization)

 conv5_block3_2_relu (Activ  (None, 5, 5, 512)          0         ['conv5_block3_2_
bn[0][0]']
 ation)

 conv5_block3_3_conv (Conv2  (None, 5, 5, 2048)         1050624   ['conv5_block3_2_
relu[0][0]']
 D)

 conv5_block3_3_bn (BatchNo  (None, 5, 5, 2048)         8192      ['conv5_block3_3_
conv[0][0]']
 rmalization)

 conv5_block3_add (Add)      (None, 5, 5, 2048)         0         ['conv5_block2_ou
t[0][0]',
                                                                   'conv5_block3_3_
bn[0][0]']

 conv5_block3_out (Activati  (None, 5, 5, 2048)         0         ['conv5_block3_ad
d[0][0]']
 on)

 flatten_2 (Flatten)         (None, 51200)              0         ['conv5_block3_ou
t[0][0]']

 dense_4 (Dense)             (None, 256)                1310745   ['flatten_2[0]
[0]']
                                                        6
 dense_5 (Dense)             (None, 1)                    257       ['dense_4[0][0]']

=====================================================================================
=============
Total params: 36695425 (139.98 MB)
Trainable params: 13107713 (50.00 MB)
Non-trainable params: 23587712 (89.98 MB)
_____________________________________________________________________________________
_____________
```

Training the Model

In [53]:

```
# Train the model
history = model.fit(
    train_generator,
    steps_per_epoch=1500 // batch_size,  # 1500 images in total (750 cat + 750 dog)
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=validation_samples // batch_size
)
```

```
Epoch 1/10
75/75 [==============================] - 118s 2s/step - loss: 0.7582 - accuracy: 0.52
93 - val_loss: 0.6655 - val_accuracy: 0.5820
Epoch 2/10
75/75 [==============================] - 117s 2s/step - loss: 0.6777 - accuracy: 0.56
20 - val_loss: 0.6520 - val_accuracy: 0.6340
Epoch 3/10
75/75 [==============================] - 132s 2s/step - loss: 0.7155 - accuracy: 0.52
47 - val_loss: 0.7538 - val_accuracy: 0.5000
Epoch 4/10
75/75 [==============================] - 117s 2s/step - loss: 0.6804 - accuracy: 0.58
07 - val_loss: 0.6468 - val_accuracy: 0.6240
Epoch 5/10
75/75 [==============================] - 131s 2s/step - loss: 0.6775 - accuracy: 0.57
07 - val_loss: 0.6354 - val_accuracy: 0.6500
Epoch 6/10
75/75 [==============================] - 131s 2s/step - loss: 0.6753 - accuracy: 0.58
13 - val_loss: 0.6817 - val_accuracy: 0.5360
Epoch 7/10
75/75 [==============================] - 116s 2s/step - loss: 0.6746 - accuracy: 0.58
67 - val_loss: 0.6352 - val_accuracy: 0.6500
Epoch 8/10
75/75 [==============================] - 131s 2s/step - loss: 0.6772 - accuracy: 0.57
60 - val_loss: 0.6924 - val_accuracy: 0.5160
Epoch 9/10
75/75 [==============================] - 131s 2s/step - loss: 0.6939 - accuracy: 0.50
67 - val_loss: 0.6685 - val_accuracy: 0.5000
Epoch 10/10
75/75 [==============================] - 132s 2s/step - loss: 0.6845 - accuracy: 0.56
60 - val_loss: 0.6692 - val_accuracy: 0.6320
```

In [54]:

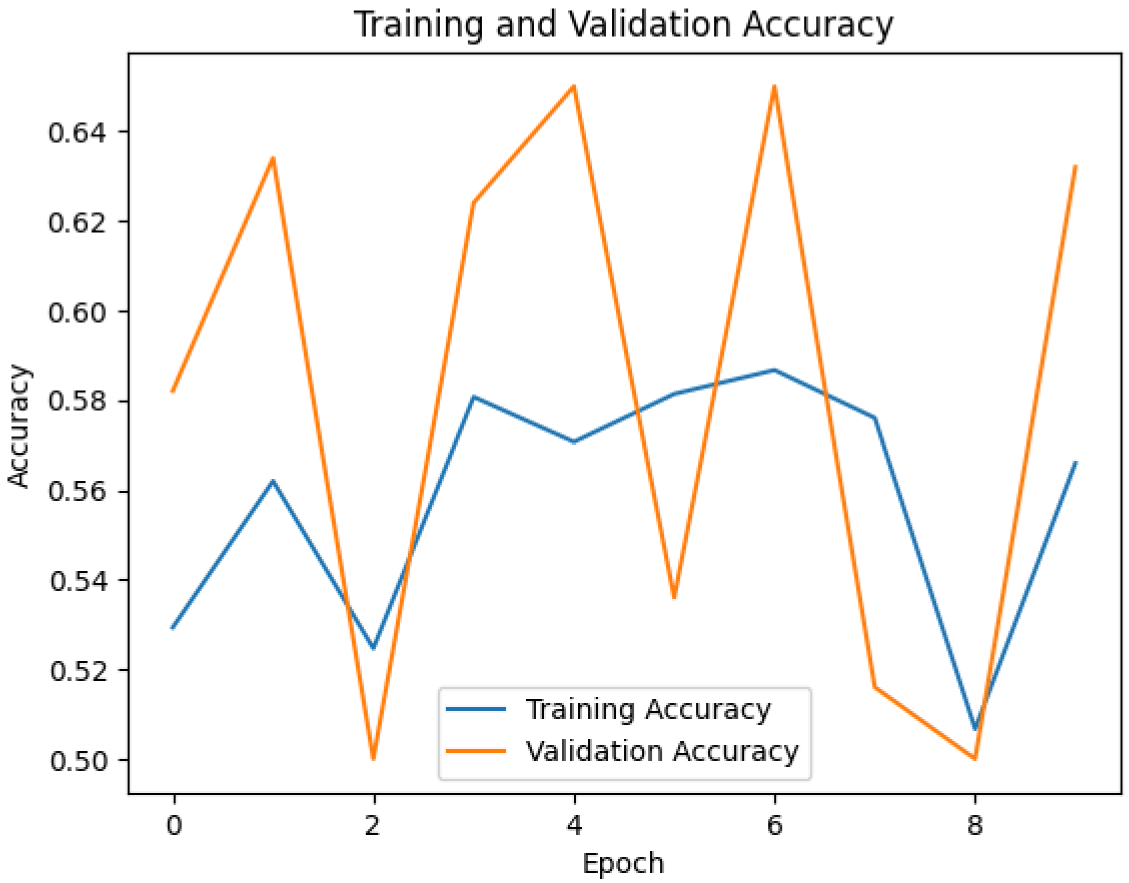
```
# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(test_generator, steps=test_samples // batch_
print("Test accuracy:", test_accuracy)
```

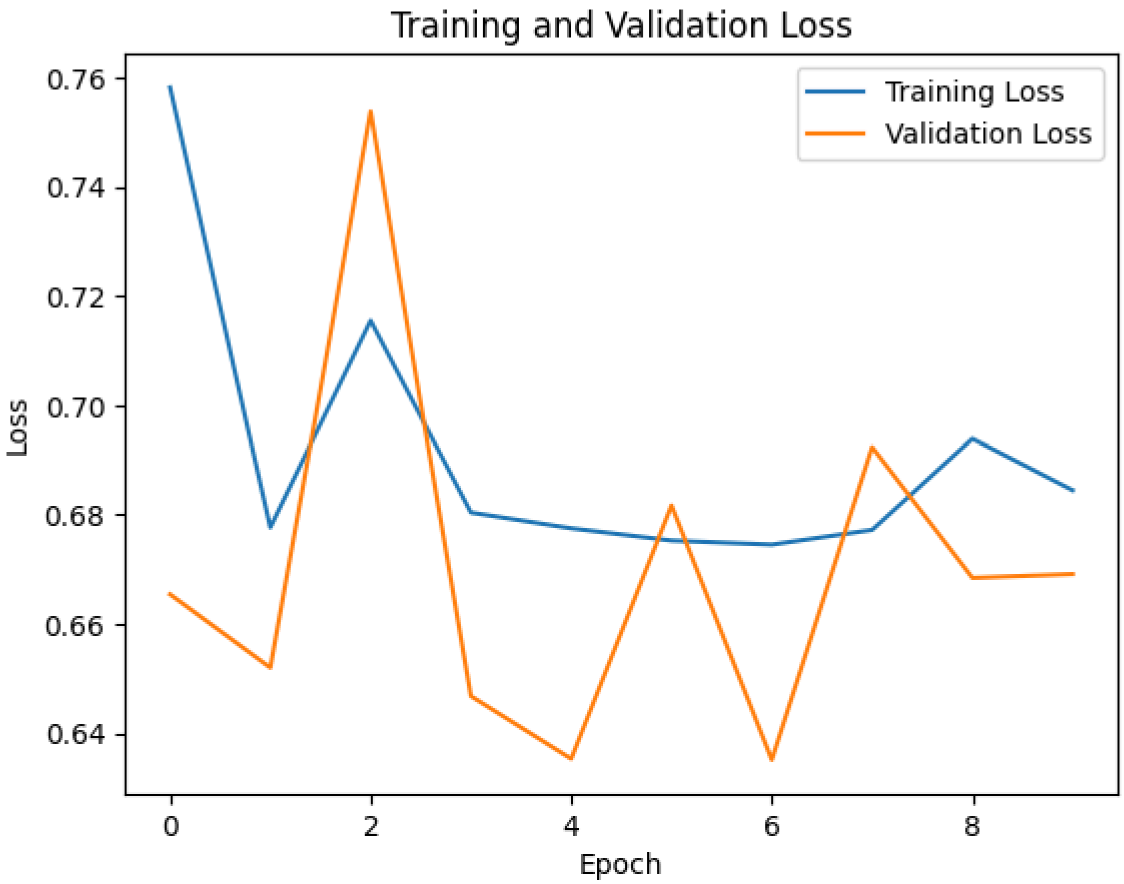
```
25/25 [==============================] - 27s 1s/step - loss: 0.6766 - accuracy: 0.584
0
Test accuracy: 0.5839999914169312
```

Evaluation Metrics

In [55]:

```
# Plot training and validation accuracy
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
# Plot training and validation loss
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```





For Training Sample of size 2000

In [56]:

```
# Clear existing data in the training directory
shutil.rmtree(train_dir)
os.makedirs(train_dir, exist_ok=True)

train_cats_dir = os.path.join(train_dir, 'cat')
os.makedirs(train_cats_dir, exist_ok=True)
train_dogs_dir = os.path.join(train_dir, 'dog')
os.makedirs(train_dogs_dir, exist_ok=True)

# Assign 500 cat and 500 dog images to the training directory
def copy_images(source_dir, destination_dir, images):
    for image in images:
        shutil.copy(os.path.join(source_dir, image), destination_dir)

# Randomize the order of cat and dog images
random.shuffle(cat_images)
random.shuffle(dog_images)

# Copy images to training directory
copy_images(cat_folder_path, train_cats_dir, cat_images[:1000])
copy_images(dog_folder_path, train_dogs_dir, dog_images[:1000])

from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Define data generators for training, validation, and testing
batch_size = 20
image_size = (150, 150)
```

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='binary'
)
```

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

In [57]:

```
# Freeze the weights of the pre-trained layers
for layer in base_model.layers:
    layer.trainable = False

# Add custom classification layers
x = Flatten()(base_model.output)
x = Dense(256, activation='relu')(x)
output = Dense(1, activation='sigmoid')(x)

# Create the model
model = Model(inputs=base_model.input, outputs=output)

# Compile the model
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

model.summary()
```

```
Model: "model_2"
_____________________________________________________________________________________
_____________
 Layer (type)                Output Shape                 Param #   Connected to
=====================================================================================
=============
 input_1 (InputLayer)        [(None, 150, 150, 3)]        0         []

 conv1_pad (ZeroPadding2D)   (None, 156, 156, 3)          0         ['input_1[0][0]']

 conv1_conv (Conv2D)         (None, 75, 75, 64)           9472      ['conv1_pad[0]
[0]']

 conv1_bn (BatchNormalizati  (None, 75, 75, 64)           256       ['conv1_conv[0]
[0]']
 on)

 conv1_relu (Activation)     (None, 75, 75, 64)           0         ['conv1_bn[0]
[0]']

 pool1_pad (ZeroPadding2D)   (None, 77, 77, 64)           0         ['conv1_relu[0]
[0]']

 pool1_pool (MaxPooling2D)   (None, 38, 38, 64)           0         ['pool1_pad[0]
[0]']

 conv2_block1_1_conv (Conv2  (None, 38, 38, 64)           4160      ['pool1_pool[0]
[0]']
 D)

 conv2_block1_1_bn (BatchNo  (None, 38, 38, 64)           256       ['conv2_block1_1_
conv[0][0]']
 rmalization)

 conv2_block1_1_relu (Activ  (None, 38, 38, 64)           0         ['conv2_block1_1_
bn[0][0]']
 ation)

 conv2_block1_2_conv (Conv2  (None, 38, 38, 64)           36928     ['conv2_block1_1_
relu[0][0]']
 D)

 conv2_block1_2_bn (BatchNo  (None, 38, 38, 64)           256       ['conv2_block1_2_
conv[0][0]']
 rmalization)

 conv2_block1_2_relu (Activ  (None, 38, 38, 64)           0         ['conv2_block1_2_
bn[0][0]']
 ation)

 conv2_block1_0_conv (Conv2  (None, 38, 38, 256)          16640     ['pool1_pool[0]
[0]']
 D)

 conv2_block1_3_conv (Conv2  (None, 38, 38, 256)          16640     ['conv2_block1_2_
relu[0][0]']
 D)

 conv2_block1_0_bn (BatchNo  (None, 38, 38, 256)          1024      ['conv2_block1_0_
conv[0][0]']
```

```
 rmalization)

 conv2_block1_3_bn (BatchNo  (None, 38, 38, 256)          1024      ['conv2_block1_3_
conv[0][0]']
 rmalization)

 conv2_block1_add (Add)      (None, 38, 38, 256)          0         ['conv2_block1_0_
bn[0][0]',
                                                                     'conv2_block1_3_

bn[0][0]']

 conv2_block1_out (Activati  (None, 38, 38, 256)          0         ['conv2_block1_ad
d[0][0]']
 on)

 conv2_block2_1_conv (Conv2  (None, 38, 38, 64)           16448     ['conv2_block1_ou
t[0][0]']
 D)

 conv2_block2_1_bn (BatchNo  (None, 38, 38, 64)           256       ['conv2_block2_1_
conv[0][0]']
 rmalization)

 conv2_block2_1_relu (Activ  (None, 38, 38, 64)           0         ['conv2_block2_1_
bn[0][0]']
 ation)

 conv2_block2_2_conv (Conv2  (None, 38, 38, 64)           36928     ['conv2_block2_1_
relu[0][0]']
 D)

 conv2_block2_2_bn (BatchNo  (None, 38, 38, 64)           256       ['conv2_block2_2_
conv[0][0]']
 rmalization)

 conv2_block2_2_relu (Activ  (None, 38, 38, 64)           0         ['conv2_block2_2_
bn[0][0]']
 ation)

 conv2_block2_3_conv (Conv2  (None, 38, 38, 256)          16640     ['conv2_block2_2_
relu[0][0]']
 D)

 conv2_block2_3_bn (BatchNo  (None, 38, 38, 256)          1024      ['conv2_block2_3_
conv[0][0]']
 rmalization)

 conv2_block2_add (Add)      (None, 38, 38, 256)          0         ['conv2_block1_ou
t[0][0]',
                                                                     'conv2_block2_3_

bn[0][0]']

 conv2_block2_out (Activati  (None, 38, 38, 256)          0         ['conv2_block2_ad
d[0][0]']
 on)

 conv2_block3_1_conv (Conv2  (None, 38, 38, 64)           16448     ['conv2_block2_ou
t[0][0]']
 D)
 conv2_block3_1_bn (BatchNo  (None, 38, 38, 64)          256       ['conv2_block3_1_
conv[0][0]']
 rmalization)

 conv2_block3_1_relu (Activ  (None, 38, 38, 64)          0         ['conv2_block3_1_
bn[0][0]']
 ation)

 conv2_block3_2_conv (Conv2  (None, 38, 38, 64)          36928     ['conv2_block3_1_
relu[0][0]']
 D)

 conv2_block3_2_bn (BatchNo  (None, 38, 38, 64)          256       ['conv2_block3_2_
conv[0][0]']
 rmalization)

 conv2_block3_2_relu (Activ  (None, 38, 38, 64)          0         ['conv2_block3_2_
bn[0][0]']
 ation)

 conv2_block3_3_conv (Conv2  (None, 38, 38, 256)         16640     ['conv2_block3_2_
relu[0][0]']
 D)

 conv2_block3_3_bn (BatchNo  (None, 38, 38, 256)         1024      ['conv2_block3_3_
conv[0][0]']
 rmalization)

 conv2_block3_add (Add)      (None, 38, 38, 256)         0         ['conv2_block2_ou
t[0][0]',

                                                                    'conv2_block3_3_
bn[0][0]']

 conv2_block3_out (Activati  (None, 38, 38, 256)         0         ['conv2_block3_ad
d[0][0]']
 on)

 conv3_block1_1_conv (Conv2  (None, 19, 19, 128)         32896     ['conv2_block3_ou
t[0][0]']
 D)

 conv3_block1_1_bn (BatchNo  (None, 19, 19, 128)         512       ['conv3_block1_1_
conv[0][0]']
 rmalization)

 conv3_block1_1_relu (Activ  (None, 19, 19, 128)         0         ['conv3_block1_1_
bn[0][0]']
 ation)

 conv3_block1_2_conv (Conv2  (None, 19, 19, 128)         147584    ['conv3_block1_1_
relu[0][0]']
 D)

 conv3_block1_2_bn (BatchNo  (None, 19, 19, 128)         512       ['conv3_block1_2_
conv[0][0]']
 rmalization)

 conv3_block1_2_relu (Activ  (None, 19, 19, 128)         0         ['conv3_block1_2_
bn[0][0]']
 ation)
 conv3_block1_0_conv (Conv2  (None, 19, 19, 512)          131584    ['conv2_block3_ou
t[0][0]']
 D)

 conv3_block1_3_conv (Conv2  (None, 19, 19, 512)          66048     ['conv3_block1_2_
relu[0][0]']
 D)

 conv3_block1_0_bn (BatchNo  (None, 19, 19, 512)          2048      ['conv3_block1_0_
conv[0][0]']
 rmalization)

 conv3_block1_3_bn (BatchNo  (None, 19, 19, 512)          2048      ['conv3_block1_3_
conv[0][0]']
 rmalization)

 conv3_block1_add (Add)      (None, 19, 19, 512)          0         ['conv3_block1_0_
bn[0][0]',

                                                                     'conv3_block1_3_
bn[0][0]']

 conv3_block1_out (Activati  (None, 19, 19, 512)          0         ['conv3_block1_ad
d[0][0]']
 on)

 conv3_block2_1_conv (Conv2  (None, 19, 19, 128)          65664     ['conv3_block1_ou
t[0][0]']
 D)

 conv3_block2_1_bn (BatchNo  (None, 19, 19, 128)          512       ['conv3_block2_1_
conv[0][0]']
 rmalization)

 conv3_block2_1_relu (Activ  (None, 19, 19, 128)          0         ['conv3_block2_1_
bn[0][0]']
 ation)

 conv3_block2_2_conv (Conv2  (None, 19, 19, 128)          147584    ['conv3_block2_1_
relu[0][0]']
 D)

 conv3_block2_2_bn (BatchNo  (None, 19, 19, 128)          512       ['conv3_block2_2_
conv[0][0]']
 rmalization)

 conv3_block2_2_relu (Activ  (None, 19, 19, 128)          0         ['conv3_block2_2_
bn[0][0]']
 ation)

 conv3_block2_3_conv (Conv2  (None, 19, 19, 512)          66048     ['conv3_block2_2_
relu[0][0]']
 D)

 conv3_block2_3_bn (BatchNo  (None, 19, 19, 512)          2048      ['conv3_block2_3_
conv[0][0]']
 rmalization)

 conv3_block2_add (Add)      (None, 19, 19, 512)          0         ['conv3_block1_ou
t[0][0]',
                                                                     'conv3_block2_3_
bn[0][0]']

 conv3_block2_out (Activati  (None, 19, 19, 512)          0         ['conv3_block2_ad
d[0][0]']
 on)

 conv3_block3_1_conv (Conv2  (None, 19, 19, 128)          65664     ['conv3_block2_ou
t[0][0]']
 D)

 conv3_block3_1_bn (BatchNo  (None, 19, 19, 128)          512       ['conv3_block3_1_
conv[0][0]']
 rmalization)

 conv3_block3_1_relu (Activ  (None, 19, 19, 128)          0         ['conv3_block3_1_
bn[0][0]']
 ation)

 conv3_block3_2_conv (Conv2  (None, 19, 19, 128)          147584    ['conv3_block3_1_
relu[0][0]']
 D)

 conv3_block3_2_bn (BatchNo  (None, 19, 19, 128)          512       ['conv3_block3_2_
conv[0][0]']
 rmalization)

 conv3_block3_2_relu (Activ  (None, 19, 19, 128)          0         ['conv3_block3_2_
bn[0][0]']
 ation)

 conv3_block3_3_conv (Conv2  (None, 19, 19, 512)          66048     ['conv3_block3_2_
relu[0][0]']
 D)

 conv3_block3_3_bn (BatchNo  (None, 19, 19, 512)          2048      ['conv3_block3_3_
conv[0][0]']
 rmalization)

 conv3_block3_add (Add)      (None, 19, 19, 512)          0         ['conv3_block2_ou
t[0][0]',
                                                                     'conv3_block3_3_
bn[0][0]']

 conv3_block3_out (Activati  (None, 19, 19, 512)          0         ['conv3_block3_ad
d[0][0]']
 on)

 conv3_block4_1_conv (Conv2  (None, 19, 19, 128)          65664     ['conv3_block3_ou
t[0][0]']
 D)

 conv3_block4_1_bn (BatchNo  (None, 19, 19, 128)          512       ['conv3_block4_1_
conv[0][0]']
 rmalization)

 conv3_block4_1_relu (Activ  (None, 19, 19, 128)          0         ['conv3_block4_1_
bn[0][0]']
 ation)
 conv3_block4_2_conv (Conv2  (None, 19, 19, 128)          147584    ['conv3_block4_1_
relu[0][0]']
 D)

 conv3_block4_2_bn (BatchNo  (None, 19, 19, 128)          512       ['conv3_block4_2_
conv[0][0]']
 rmalization)

 conv3_block4_2_relu (Activ  (None, 19, 19, 128)          0         ['conv3_block4_2_
bn[0][0]']
 ation)

 conv3_block4_3_conv (Conv2  (None, 19, 19, 512)          66048     ['conv3_block4_2_
relu[0][0]']
 D)

 conv3_block4_3_bn (BatchNo  (None, 19, 19, 512)          2048      ['conv3_block4_3_
conv[0][0]']
 rmalization)

 conv3_block4_add (Add)      (None, 19, 19, 512)          0         ['conv3_block3_ou
t[0][0]',
                                                                     'conv3_block4_3_

bn[0][0]']

 conv3_block4_out (Activati  (None, 19, 19, 512)          0         ['conv3_block4_ad
d[0][0]']
 on)

 conv4_block1_1_conv (Conv2  (None, 10, 10, 256)          131328    ['conv3_block4_ou
t[0][0]']
 D)

 conv4_block1_1_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block1_1_
conv[0][0]']
 rmalization)

 conv4_block1_1_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block1_1_
bn[0][0]']
 ation)

 conv4_block1_2_conv (Conv2  (None, 10, 10, 256)          590080    ['conv4_block1_1_
relu[0][0]']
 D)

 conv4_block1_2_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block1_2_
conv[0][0]']
 rmalization)

 conv4_block1_2_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block1_2_
bn[0][0]']
 ation)

 conv4_block1_0_conv (Conv2  (None, 10, 10, 1024)         525312    ['conv3_block4_ou
t[0][0]']
 D)

 conv4_block1_3_conv (Conv2  (None, 10, 10, 1024)         263168    ['conv4_block1_2_
relu[0][0]']
 D)
 conv4_block1_0_bn (BatchNo  (None, 10, 10, 1024)         4096      ['conv4_block1_0_
conv[0][0]']
 rmalization)

 conv4_block1_3_bn (BatchNo  (None, 10, 10, 1024)         4096      ['conv4_block1_3_
conv[0][0]']
 rmalization)

 conv4_block1_add (Add)      (None, 10, 10, 1024)         0         ['conv4_block1_0_
bn[0][0]',
                                                                     'conv4_block1_3_

bn[0][0]']

 conv4_block1_out (Activati  (None, 10, 10, 1024)         0         ['conv4_block1_ad
d[0][0]']
 on)

 conv4_block2_1_conv (Conv2  (None, 10, 10, 256)          262400    ['conv4_block1_ou
t[0][0]']
 D)

 conv4_block2_1_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block2_1_
conv[0][0]']
 rmalization)

 conv4_block2_1_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block2_1_
bn[0][0]']
 ation)

 conv4_block2_2_conv (Conv2  (None, 10, 10, 256)          590080    ['conv4_block2_1_
relu[0][0]']
 D)

 conv4_block2_2_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block2_2_
conv[0][0]']
 rmalization)

 conv4_block2_2_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block2_2_
bn[0][0]']
 ation)

 conv4_block2_3_conv (Conv2  (None, 10, 10, 1024)         263168    ['conv4_block2_2_
relu[0][0]']
 D)

 conv4_block2_3_bn (BatchNo  (None, 10, 10, 1024)         4096      ['conv4_block2_3_
conv[0][0]']
 rmalization)

 conv4_block2_add (Add)      (None, 10, 10, 1024)         0         ['conv4_block1_ou
t[0][0]',
                                                                     'conv4_block2_3_

bn[0][0]']

 conv4_block2_out (Activati  (None, 10, 10, 1024)         0         ['conv4_block2_ad
d[0][0]']
 on)

 conv4_block3_1_conv (Conv2  (None, 10, 10, 256)          262400    ['conv4_block2_ou
t[0][0]']
 D)

 conv4_block3_1_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block3_1_
conv[0][0]']
 rmalization)

 conv4_block3_1_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block3_1_
bn[0][0]']
 ation)

 conv4_block3_2_conv (Conv2  (None, 10, 10, 256)          590080    ['conv4_block3_1_
relu[0][0]']
 D)

 conv4_block3_2_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block3_2_
conv[0][0]']
 rmalization)

 conv4_block3_2_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block3_2_
bn[0][0]']
 ation)

 conv4_block3_3_conv (Conv2  (None, 10, 10, 1024)         263168    ['conv4_block3_2_
relu[0][0]']
 D)

 conv4_block3_3_bn (BatchNo  (None, 10, 10, 1024)         4096      ['conv4_block3_3_
conv[0][0]']
 rmalization)

 conv4_block3_add (Add)      (None, 10, 10, 1024)         0         ['conv4_block2_ou
t[0][0]',
                                                                     'conv4_block3_3_
bn[0][0]']

 conv4_block3_out (Activati  (None, 10, 10, 1024)         0         ['conv4_block3_ad
d[0][0]']
 on)

 conv4_block4_1_conv (Conv2  (None, 10, 10, 256)          262400    ['conv4_block3_ou
t[0][0]']
 D)

 conv4_block4_1_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block4_1_
conv[0][0]']
 rmalization)

 conv4_block4_1_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block4_1_
bn[0][0]']
 ation)

 conv4_block4_2_conv (Conv2  (None, 10, 10, 256)          590080    ['conv4_block4_1_
relu[0][0]']
 D)

 conv4_block4_2_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block4_2_
conv[0][0]']
 rmalization)
 conv4_block4_2_relu (Activ  (None, 10, 10, 256)          0           ['conv4_block4_2_
bn[0][0]']
 ation)

 conv4_block4_3_conv (Conv2  (None, 10, 10, 1024)         263168      ['conv4_block4_2_
relu[0][0]']
 D)

 conv4_block4_3_bn (BatchNo  (None, 10, 10, 1024)         4096        ['conv4_block4_3_
conv[0][0]']
 rmalization)

 conv4_block4_add (Add)      (None, 10, 10, 1024)         0           ['conv4_block3_ou
t[0][0]',
                                                                       'conv4_block4_3_
bn[0][0]']

 conv4_block4_out (Activati  (None, 10, 10, 1024)         0           ['conv4_block4_ad
d[0][0]']
 on)

 conv4_block5_1_conv (Conv2  (None, 10, 10, 256)          262400      ['conv4_block4_ou
t[0][0]']
 D)

 conv4_block5_1_bn (BatchNo  (None, 10, 10, 256)          1024        ['conv4_block5_1_
conv[0][0]']
 rmalization)

 conv4_block5_1_relu (Activ  (None, 10, 10, 256)          0           ['conv4_block5_1_
bn[0][0]']
 ation)

 conv4_block5_2_conv (Conv2  (None, 10, 10, 256)          590080      ['conv4_block5_1_
relu[0][0]']
 D)

 conv4_block5_2_bn (BatchNo  (None, 10, 10, 256)          1024        ['conv4_block5_2_
conv[0][0]']
 rmalization)

 conv4_block5_2_relu (Activ  (None, 10, 10, 256)          0           ['conv4_block5_2_
bn[0][0]']
 ation)

 conv4_block5_3_conv (Conv2  (None, 10, 10, 1024)         263168      ['conv4_block5_2_
relu[0][0]']
 D)

 conv4_block5_3_bn (BatchNo  (None, 10, 10, 1024)         4096        ['conv4_block5_3_
conv[0][0]']
 rmalization)

 conv4_block5_add (Add)      (None, 10, 10, 1024)         0           ['conv4_block4_ou
t[0][0]',
                                                                       'conv4_block5_3_
bn[0][0]']

 conv4_block5_out (Activati  (None, 10, 10, 1024)         0           ['conv4_block5_ad
d[0][0]']
```

```
 on)

 conv4_block6_1_conv (Conv2  (None, 10, 10, 256)          262400    ['conv4_block5_ou
t[0][0]']
 D)

 conv4_block6_1_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block6_1_
conv[0][0]']
 rmalization)

 conv4_block6_1_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block6_1_
bn[0][0]']
 ation)

 conv4_block6_2_conv (Conv2  (None, 10, 10, 256)          590080    ['conv4_block6_1_
relu[0][0]']
 D)

 conv4_block6_2_bn (BatchNo  (None, 10, 10, 256)          1024      ['conv4_block6_2_
conv[0][0]']
 rmalization)

 conv4_block6_2_relu (Activ  (None, 10, 10, 256)          0         ['conv4_block6_2_
bn[0][0]']
 ation)

 conv4_block6_3_conv (Conv2  (None, 10, 10, 1024)         263168    ['conv4_block6_2_
relu[0][0]']
 D)

 conv4_block6_3_bn (BatchNo  (None, 10, 10, 1024)         4096      ['conv4_block6_3_
conv[0][0]']
 rmalization)

 conv4_block6_add (Add)      (None, 10, 10, 1024)         0         ['conv4_block5_ou
t[0][0]',
                                                                     'conv4_block6_3_
bn[0][0]']

 conv4_block6_out (Activati  (None, 10, 10, 1024)         0         ['conv4_block6_ad
d[0][0]']
 on)

 conv5_block1_1_conv (Conv2  (None, 5, 5, 512)            524800    ['conv4_block6_ou
t[0][0]']
 D)

 conv5_block1_1_bn (BatchNo  (None, 5, 5, 512)            2048      ['conv5_block1_1_
conv[0][0]']
 rmalization)

 conv5_block1_1_relu (Activ  (None, 5, 5, 512)            0         ['conv5_block1_1_
bn[0][0]']
 ation)

 conv5_block1_2_conv (Conv2  (None, 5, 5, 512)            2359808   ['conv5_block1_1_
relu[0][0]']
 D)

 conv5_block1_2_bn (BatchNo  (None, 5, 5, 512)            2048      ['conv5_block1_2_
conv[0][0]']
 rmalization)

 conv5_block1_2_relu (Activ  (None, 5, 5, 512)            0          ['conv5_block1_2_
bn[0][0]']
 ation)

 conv5_block1_0_conv (Conv2  (None, 5, 5, 2048)           2099200    ['conv4_block6_ou
t[0][0]']
 D)

 conv5_block1_3_conv (Conv2  (None, 5, 5, 2048)           1050624    ['conv5_block1_2_
relu[0][0]']
 D)

 conv5_block1_0_bn (BatchNo  (None, 5, 5, 2048)           8192       ['conv5_block1_0_
conv[0][0]']
 rmalization)

 conv5_block1_3_bn (BatchNo  (None, 5, 5, 2048)           8192       ['conv5_block1_3_
conv[0][0]']
 rmalization)

 conv5_block1_add (Add)      (None, 5, 5, 2048)           0          ['conv5_block1_0_
bn[0][0]',
                                                                      'conv5_block1_3_
bn[0][0]']

 conv5_block1_out (Activati  (None, 5, 5, 2048)           0          ['conv5_block1_ad
d[0][0]']
 on)

 conv5_block2_1_conv (Conv2  (None, 5, 5, 512)            1049088    ['conv5_block1_ou
t[0][0]']
 D)

 conv5_block2_1_bn (BatchNo  (None, 5, 5, 512)            2048       ['conv5_block2_1_
conv[0][0]']
 rmalization)

 conv5_block2_1_relu (Activ  (None, 5, 5, 512)            0          ['conv5_block2_1_
bn[0][0]']
 ation)

 conv5_block2_2_conv (Conv2  (None, 5, 5, 512)            2359808    ['conv5_block2_1_
relu[0][0]']
 D)

 conv5_block2_2_bn (BatchNo  (None, 5, 5, 512)            2048       ['conv5_block2_2_
conv[0][0]']
 rmalization)

 conv5_block2_2_relu (Activ  (None, 5, 5, 512)            0          ['conv5_block2_2_
bn[0][0]']
 ation)

 conv5_block2_3_conv (Conv2  (None, 5, 5, 2048)           1050624    ['conv5_block2_2_
relu[0][0]']
 D)
 conv5_block2_3_bn (BatchNo  (None, 5, 5, 2048)           8192      ['conv5_block2_3_
conv[0][0]']
 rmalization)

 conv5_block2_add (Add)      (None, 5, 5, 2048)           0         ['conv5_block1_ou
t[0][0]',
                                                                     'conv5_block2_3_
bn[0][0]']

 conv5_block2_out (Activati  (None, 5, 5, 2048)           0         ['conv5_block2_ad
d[0][0]']
 on)

 conv5_block3_1_conv (Conv2  (None, 5, 5, 512)            1049088   ['conv5_block2_ou
t[0][0]']
 D)

 conv5_block3_1_bn (BatchNo  (None, 5, 5, 512)            2048      ['conv5_block3_1_
conv[0][0]']
 rmalization)

 conv5_block3_1_relu (Activ  (None, 5, 5, 512)            0         ['conv5_block3_1_
bn[0][0]']
 ation)

 conv5_block3_2_conv (Conv2  (None, 5, 5, 512)            2359808   ['conv5_block3_1_
relu[0][0]']
 D)

 conv5_block3_2_bn (BatchNo  (None, 5, 5, 512)            2048      ['conv5_block3_2_
conv[0][0]']
 rmalization)

 conv5_block3_2_relu (Activ  (None, 5, 5, 512)            0         ['conv5_block3_2_
bn[0][0]']
 ation)

 conv5_block3_3_conv (Conv2  (None, 5, 5, 2048)           1050624   ['conv5_block3_2_
relu[0][0]']
 D)

 conv5_block3_3_bn (BatchNo  (None, 5, 5, 2048)           8192      ['conv5_block3_3_
conv[0][0]']
 rmalization)

 conv5_block3_add (Add)      (None, 5, 5, 2048)           0         ['conv5_block2_ou
t[0][0]',
                                                                     'conv5_block3_3_
bn[0][0]']

 conv5_block3_out (Activati  (None, 5, 5, 2048)           0         ['conv5_block3_ad
d[0][0]']
 on)

 flatten_3 (Flatten)         (None, 51200)                0         ['conv5_block3_ou
t[0][0]']

 dense_6 (Dense)             (None, 256)                  1310745   ['flatten_3[0]
[0]']

                                                          6
 dense_7 (Dense)             (None, 1)                    257       ['dense_6[0][0]']

=====================================================================================
=============
Total params: 36695425 (139.98 MB)
Trainable params: 13107713 (50.00 MB)
Non-trainable params: 23587712 (89.98 MB)
_____________________________________________________________________________________
_____________
```

Training The Model

In [58]:

```
# Train the model
history = model.fit(
    train_generator,
    steps_per_epoch=2000 // batch_size,  # 1500 images in total (1000 cat + 1000 dog)
    epochs=30,
    validation_data=validation_generator,
    validation_steps=validation_samples // batch_size
)
```

```
Epoch 1/30
100/100 [==============================] - 165s 2s/step - loss: 0.8609 - accuracy: 0.
5345 - val_loss: 0.6573 - val_accuracy: 0.6020
Epoch 2/30
100/100 [==============================] - 145s 1s/step - loss: 0.7291 - accuracy: 0.
5520 - val_loss: 0.6674 - val_accuracy: 0.5900
Epoch 3/30
100/100 [==============================] - 160s 2s/step - loss: 0.6793 - accuracy: 0.
5765 - val_loss: 0.8207 - val_accuracy: 0.5000
Epoch 4/30
100/100 [==============================] - 144s 1s/step - loss: 0.6859 - accuracy: 0.
5805 - val_loss: 0.6616 - val_accuracy: 0.5820
Epoch 5/30
100/100 [==============================] - 145s 1s/step - loss: 0.6657 - accuracy: 0.
5950 - val_loss: 0.6332 - val_accuracy: 0.6500
Epoch 6/30
100/100 [==============================] - 161s 2s/step - loss: 0.6943 - accuracy: 0.
5825 - val_loss: 0.6593 - val_accuracy: 0.5920
Epoch 7/30
100/100 [==============================] - 161s 2s/step - loss: 0.6828 - accuracy: 0.
5645 - val_loss: 0.6378 - val_accuracy: 0.6240
Epoch 8/30
100/100 [==============================] - 161s 2s/step - loss: 0.6694 - accuracy: 0.
5970 - val_loss: 0.6572 - val_accuracy: 0.6020
Epoch 9/30
100/100 [==============================] - 160s 2s/step - loss: 0.6719 - accuracy: 0.
5800 - val_loss: 0.6640 - val_accuracy: 0.5960
Epoch 10/30
100/100 [==============================] - 147s 1s/step - loss: 0.6649 - accuracy: 0.
5895 - val_loss: 0.6390 - val_accuracy: 0.6160
Epoch 11/30
100/100 [==============================] - 161s 2s/step - loss: 0.6637 - accuracy: 0.
6055 - val_loss: 0.6293 - val_accuracy: 0.6560
Epoch 12/30
100/100 [==============================] - 161s 2s/step - loss: 0.6512 - accuracy: 0.
6195 - val_loss: 0.6279 - val_accuracy: 0.6520
Epoch 13/30
100/100 [==============================] - 160s 2s/step - loss: 0.6521 - accuracy: 0.
6215 - val_loss: 0.6223 - val_accuracy: 0.6700
Epoch 14/30
100/100 [==============================] - 161s 2s/step - loss: 0.6533 - accuracy: 0.
6110 - val_loss: 0.6396 - val_accuracy: 0.6300
Epoch 15/30
100/100 [==============================] - 146s 1s/step - loss: 0.6534 - accuracy: 0.
6045 - val_loss: 0.6335 - val_accuracy: 0.6300
Epoch 16/30
100/100 [==============================] - 145s 1s/step - loss: 0.6577 - accuracy: 0.
5965 - val_loss: 0.6232 - val_accuracy: 0.6400
Epoch 17/30
100/100 [==============================] - 145s 1s/step - loss: 0.6457 - accuracy: 0.
6255 - val_loss: 0.6212 - val_accuracy: 0.6440
Epoch 18/30
100/100 [==============================] - 161s 2s/step - loss: 0.6383 - accuracy: 0.
6400 - val_loss: 0.6420 - val_accuracy: 0.6260
Epoch 19/30
100/100 [==============================] - 145s 1s/step - loss: 0.6390 - accuracy: 0.
6400 - val_loss: 0.6300 - val_accuracy: 0.6400
Epoch 20/30
100/100 [==============================] - 165s 2s/step - loss: 0.6579 - accuracy: 0.
5980 - val_loss: 0.6247 - val_accuracy: 0.6420
```

```
Epoch 21/30
100/100 [==============================] - 162s 2s/step - loss: 0.6585 - accuracy: 0.
6050 - val_loss: 0.6844 - val_accuracy: 0.5520
Epoch 22/30
100/100 [==============================] - 162s 2s/step - loss: 0.6594 - accuracy: 0.
6075 - val_loss: 0.6125 - val_accuracy: 0.6640
Epoch 23/30
100/100 [==============================] - 147s 1s/step - loss: 0.6459 - accuracy: 0.
6175 - val_loss: 0.6181 - val_accuracy: 0.6460
Epoch 24/30
100/100 [==============================] - 147s 1s/step - loss: 0.6506 - accuracy: 0.
6105 - val_loss: 0.6107 - val_accuracy: 0.6740
Epoch 25/30
100/100 [==============================] - 162s 2s/step - loss: 0.6404 - accuracy: 0.
6305 - val_loss: 0.6447 - val_accuracy: 0.6140
Epoch 26/30
100/100 [==============================] - 162s 2s/step - loss: 0.6465 - accuracy: 0.
6395 - val_loss: 0.6016 - val_accuracy: 0.6760
Epoch 27/30
100/100 [==============================] - 163s 2s/step - loss: 0.6405 - accuracy: 0.
6395 - val_loss: 0.6164 - val_accuracy: 0.6580
Epoch 28/30
100/100 [==============================] - 162s 2s/step - loss: 0.6360 - accuracy: 0.
6425 - val_loss: 0.6056 - val_accuracy: 0.6700
Epoch 29/30
100/100 [==============================] - 147s 1s/step - loss: 0.6448 - accuracy: 0.
6150 - val_loss: 0.6462 - val_accuracy: 0.6180
Epoch 30/30
100/100 [==============================] - 162s 2s/step - loss: 0.6349 - accuracy: 0.
6310 - val_loss: 0.6313 - val_accuracy: 0.6420
```

In [59]:

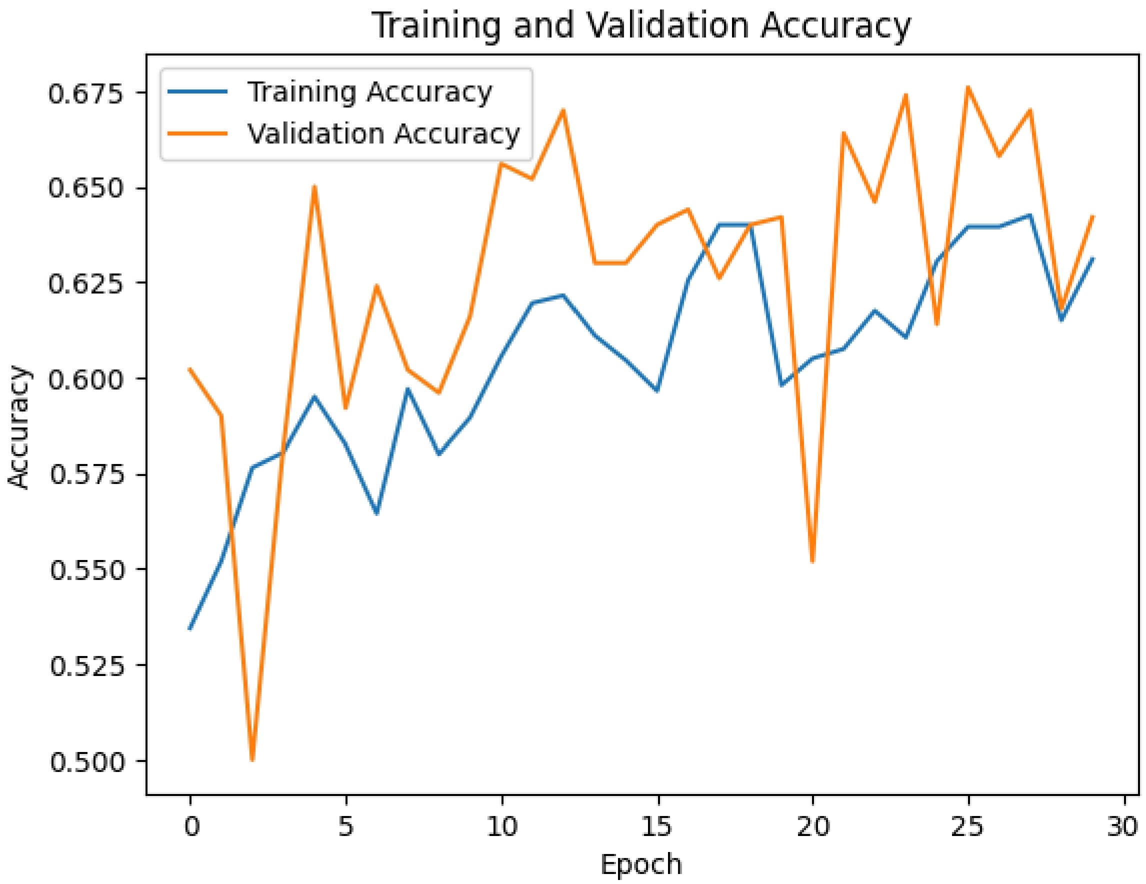
```python
# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(test_generator, steps=test_samples // batch_
print("Test accuracy:", test_accuracy)
```

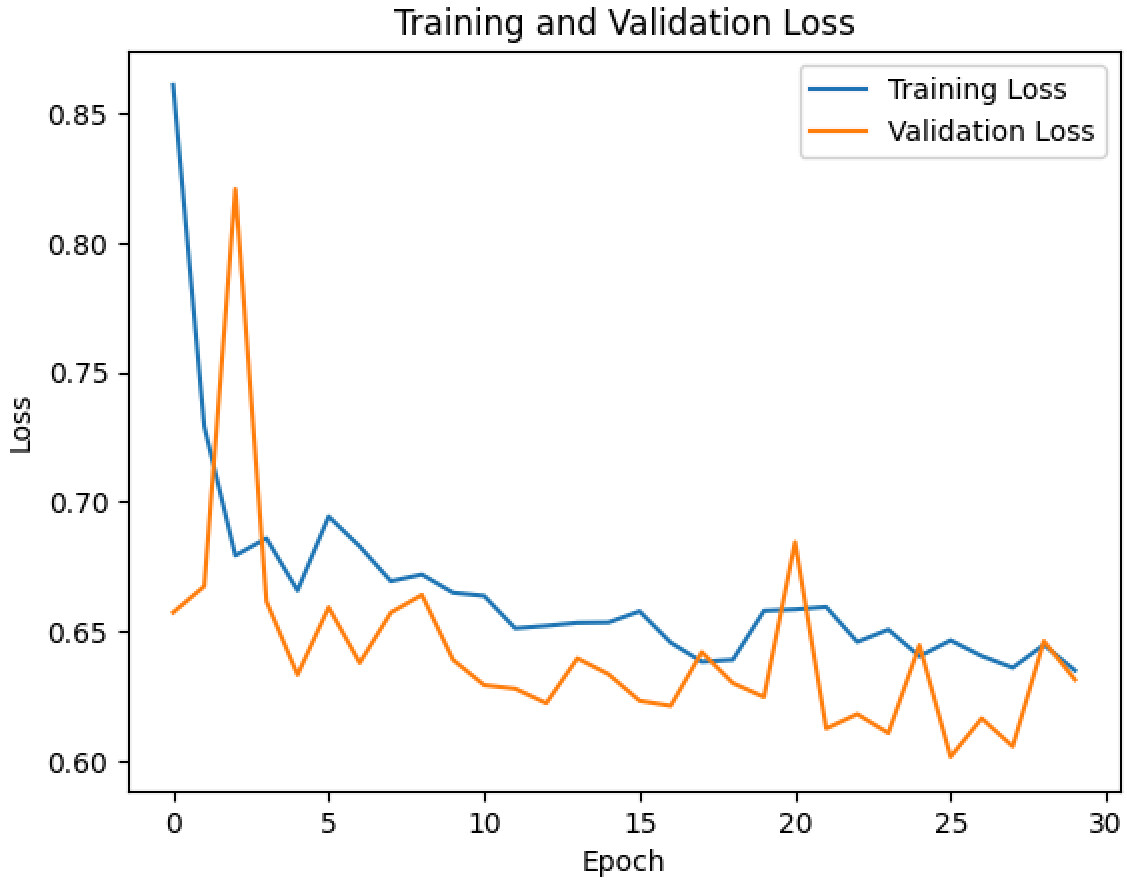
```
25/25 [==============================] - 26s 1s/step - loss: 0.6642 - accuracy: 0.614
0
Test accuracy: 0.6140000224113464
```

Perfomace Metrics

In [60]:

```python
# Plot training and validation accuracy
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
# Plot training and validation loss
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```


Training and Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy
0.675
0.650
0.625
0.600
0.575
0.550
0.525
0.500
0
5
10
15
20
25
30
Epoch


Training and Validation Loss
Training Loss
Validation Loss
Loss
0.85
0.80
0.75
0.70
0.65
0.60
0
5
10
15
20
25
30
Epoch