# CATS vs DOGS CNN Prediction

## Importing Libraries

In [1]:

```
import keras
import PIL
```

In [2]:

```
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense, Dropout
from keras.layers import Activation, BatchNormalization
```

# Initializing and Building the CNN

In [3]:

```
model = Sequential()
```

In [4]:

```
model.add(Conv2D(32, (3,3) ,input_shape=(64, 64, 3), activation='relu')) #Convolution
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size= (2,2))) #Pooling
model.add(Dropout(0.25))
```

In [5]:

```
model.add(Conv2D(32, (3, 3), activation='relu')) #Convolution
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size= (2,2))) #Pooling
model.add(Dropout(0.25))
```

In [6]:

```
model.add(Flatten())
```

In [7]:

```
model.add(Dense(units=64, activation='relu', kernel_initializer='uniform'))
model.add(BatchNormalization())
model.add(Dropout(0.6))
model.add(Dense(units=2, activation='softmax'))
```

# Compiling the model

In [8]:

```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

In [9]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896

 batch_normalization (Batch  (None, 62, 62, 32)        128
 Normalization)
 max_pooling2d (MaxPooling2  (None, 31, 31, 32)        0
 D)

 dropout (Dropout)           (None, 31, 31, 32)        0

 conv2d_1 (Conv2D)           (None, 29, 29, 32)        9248

 batch_normalization_1 (Bat  (None, 29, 29, 32)        128
 chNormalization)

 max_pooling2d_1 (MaxPoolin  (None, 14, 14, 32)        0
 g2D)

 dropout_1 (Dropout)         (None, 14, 14, 32)        0

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 64)                401472

 batch_normalization_2 (Bat  (None, 64)                256
 chNormalization)

 dropout_2 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 2)                 130

=================================================================
Total params: 412258 (1.57 MB)
Trainable params: 412002 (1.57 MB)
Non-trainable params: 256 (1.00 KB)
_________________________________________________________________
```

# Initializing EarlyStopping & Reduce-LR-On-Plateau

In [10]:

```python
from keras.callbacks import EarlyStopping, ReduceLROnPlateau
```

In [11]:

```python
earlystop = EarlyStopping(patience=10)
learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy',
                                            patience=2,
                                            verbose=1,
                                            factor=0.75,
                                            min_lr=0.00005)

callbacks = [earlystop, learning_rate_reduction]
```

# Fitting images in the CNN

In [12]:

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

training_datagen = ImageDataGenerator(rescale=1./255,
                                      shear_range=0.1,
                                      zoom_range=0.1,
                                      horizontal_flip=True)

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

In [13]:

```python
# Training Dataset
training_set = training_datagen.flow_from_directory('C:/Users/LEGION/Downloads/dataset/d
                                                     target_size=(64,64),
                                                     batch_size=32,
                                                     class_mode='categorical')
```

```
Found 8000 images belonging to 2 classes.
```

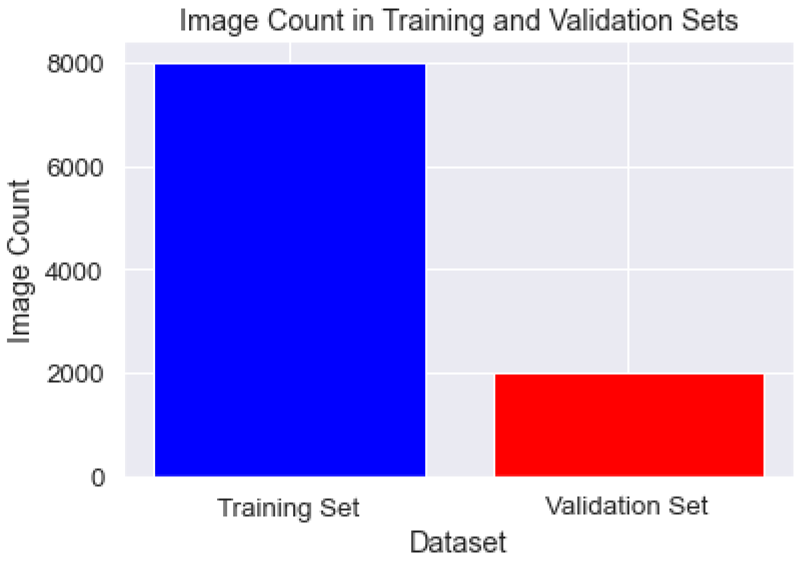
In [14]:

```
# Testing Dataset
validation_set = validation_datagen.flow_from_directory('C:/Users/LEGION/Downloads/datas
                                                 target_size=(64,64),
                                                 batch_size=32,
                                                 class_mode='categorical')
```

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

In [44]:

```
import matplotlib.pyplot as plt

# Assuming you have already loaded your training and validation sets
# using ImageDataGenerator and flow_from_directory.

# Get the counts of images in the training and validation sets
training_count = len(training_set.filenames)
validation_count = len(validation_set.filenames)

# Create labels for the sets
set_labels = ['Training Set', 'Validation Set']

# Create a list of image counts
image_counts = [training_count, validation_count]

# Create a bar plot
plt.bar(set_labels, image_counts, color=['blue', 'red'])
plt.xlabel('Dataset')
plt.ylabel('Image Count')
plt.title('Image Count in Training and Validation Sets')
plt.show()
```



In [15]:

```
FAST_RUN = False
epochs=5 if FAST_RUN else 30
```

In [16]:

```
history = model.fit_generator(training_set,
                              steps_per_epoch=8000//32,
                              epochs=epochs,
                              validation_data=validation_set,
                              validation_steps=2000//32,
                              callbacks=callbacks)
```

```
Epoch 1/30
C:\Users\LEGION\AppData\Local\Temp\ipykernel_31160\3544215541.py:1: UserWarning: `Model.
```

```
fit_generator` is deprecated and will be removed in a future version. Please use `Model.
fit`, which supports generators.
  history = model.fit_generator(training_set,
250/250 [==============================] - 35s 135ms/step - loss: 0.8560 - accuracy: 0.5
915 - val_loss: 0.8132 - val_accuracy: 0.5287 - lr: 0.0010
Epoch 2/30
250/250 [==============================] - 32s 127ms/step - loss: 0.6652 - accuracy: 0.6
289 - val_loss: 0.6166 - val_accuracy: 0.6689 - lr: 0.0010
Epoch 3/30
250/250 [==============================] - 33s 130ms/step - loss: 0.5917 - accuracy: 0.6
824 - val_loss: 0.6271 - val_accuracy: 0.6613 - lr: 0.0010
Epoch 4/30
250/250 [==============================] - ETA: 0s - loss: 0.5744 - accuracy: 0.6955
Epoch 4: ReduceLROnPlateau reducing learning rate to 0.0007500000356230885.
250/250 [==============================] - 32s 127ms/step - loss: 0.5744 - accuracy: 0.6
955 - val_loss: 0.6962 - val_accuracy: 0.6038 - lr: 0.0010
Epoch 5/30
250/250 [==============================] - 32s 128ms/step - loss: 0.5320 - accuracy: 0.7
347 - val_loss: 0.5403 - val_accuracy: 0.7268 - lr: 7.5000e-04
Epoch 6/30
250/250 [==============================] - 32s 128ms/step - loss: 0.5307 - accuracy: 0.7
319 - val_loss: 0.5912 - val_accuracy: 0.6880 - lr: 7.5000e-04
Epoch 7/30
250/250 [==============================] - 32s 129ms/step - loss: 0.5133 - accuracy: 0.7
490 - val_loss: 0.5091 - val_accuracy: 0.7490 - lr: 7.5000e-04
Epoch 8/30
250/250 [==============================] - 32s 127ms/step - loss: 0.4924 - accuracy: 0.7
617 - val_loss: 0.4833 - val_accuracy: 0.7707 - lr: 7.5000e-04
Epoch 9/30
250/250 [==============================] - 32s 127ms/step - loss: 0.4887 - accuracy: 0.7
596 - val_loss: 0.5672 - val_accuracy: 0.7056 - lr: 7.5000e-04
Epoch 10/30
250/250 [==============================] - ETA: 0s - loss: 0.4797 - accuracy: 0.7694
Epoch 10: ReduceLROnPlateau reducing learning rate to 0.0005625000048894435.
250/250 [==============================] - 32s 127ms/step - loss: 0.4797 - accuracy: 0.7
694 - val_loss: 0.5966 - val_accuracy: 0.6976 - lr: 7.5000e-04
Epoch 11/30
250/250 [==============================] - 32s 127ms/step - loss: 0.4618 - accuracy: 0.7
809 - val_loss: 0.4976 - val_accuracy: 0.7631 - lr: 5.6250e-04
Epoch 12/30
250/250 [==============================] - 32s 127ms/step - loss: 0.4522 - accuracy: 0.7
909 - val_loss: 0.4912 - val_accuracy: 0.7727 - lr: 5.6250e-04
Epoch 13/30
250/250 [==============================] - 32s 127ms/step - loss: 0.4485 - accuracy: 0.7
896 - val_loss: 0.4959 - val_accuracy: 0.7535 - lr: 5.6250e-04
Epoch 14/30
250/250 [==============================] - ETA: 0s - loss: 0.4504 - accuracy: 0.7874
Epoch 14: ReduceLROnPlateau reducing learning rate to 0.0004218749818392098.
250/250 [==============================] - 32s 128ms/step - loss: 0.4504 - accuracy: 0.7
874 - val_loss: 0.5039 - val_accuracy: 0.7525 - lr: 5.6250e-04
Epoch 15/30
250/250 [==============================] - 31s 123ms/step - loss: 0.4299 - accuracy: 0.8
011 - val_loss: 0.5047 - val_accuracy: 0.7666 - lr: 4.2187e-04
Epoch 16/30
250/250 [==============================] - 31s 125ms/step - loss: 0.4330 - accuracy: 0.8
015 - val_loss: 0.4285 - val_accuracy: 0.8014 - lr: 4.2187e-04
Epoch 17/30
250/250 [==============================] - 31s 122ms/step - loss: 0.4121 - accuracy: 0.8
115 - val_loss: 0.4320 - val_accuracy: 0.8095 - lr: 4.2187e-04
Epoch 18/30
250/250 [==============================] - 31s 124ms/step - loss: 0.4126 - accuracy: 0.8
074 - val_loss: 0.4195 - val_accuracy: 0.8054 - lr: 4.2187e-04
Epoch 19/30
250/250 [==============================] - ETA: 0s - loss: 0.4055 - accuracy: 0.8159
Epoch 19: ReduceLROnPlateau reducing learning rate to 0.00031640623637940735.
250/250 [==============================] - 31s 123ms/step - loss: 0.4055 - accuracy: 0.8
```

```
159 - val_loss: 0.4212 - val_accuracy: 0.8029 - lr: 4.2187e-04
Epoch 20/30
250/250 [==============================] - 31s 125ms/step - loss: 0.4077 - accuracy: 0.8
174 - val_loss: 0.4190 - val_accuracy: 0.8070 - lr: 3.1641e-04
Epoch 21/30
250/250 [==============================] - 30s 120ms/step - loss: 0.3929 - accuracy: 0.8
235 - val_loss: 0.4139 - val_accuracy: 0.8100 - lr: 3.1641e-04
Epoch 22/30
250/250 [==============================] - 30s 122ms/step - loss: 0.3941 - accuracy: 0.8
223 - val_loss: 0.4572 - val_accuracy: 0.7863 - lr: 3.1641e-04
Epoch 23/30
250/250 [==============================] - 30s 121ms/step - loss: 0.3805 - accuracy: 0.8
305 - val_loss: 0.4094 - val_accuracy: 0.8135 - lr: 3.1641e-04
Epoch 24/30
250/250 [==============================] - 29s 114ms/step - loss: 0.3737 - accuracy: 0.8
313 - val_loss: 0.4193 - val_accuracy: 0.8140 - lr: 3.1641e-04
Epoch 25/30
250/250 [==============================] - 31s 123ms/step - loss: 0.3839 - accuracy: 0.8
316 - val_loss: 0.4424 - val_accuracy: 0.7989 - lr: 3.1641e-04
Epoch 26/30
250/250 [==============================] - 31s 123ms/step - loss: 0.3735 - accuracy: 0.8
328 - val_loss: 0.4185 - val_accuracy: 0.8276 - lr: 3.1641e-04
Epoch 27/30
250/250 [==============================] - 28s 112ms/step - loss: 0.3811 - accuracy: 0.8
279 - val_loss: 0.3976 - val_accuracy: 0.8271 - lr: 3.1641e-04
Epoch 28/30
250/250 [==============================] - ETA: 0s - loss: 0.3654 - accuracy: 0.8396
Epoch 28: ReduceLROnPlateau reducing learning rate to 0.00023730468819849193.
250/250 [==============================] - 31s 124ms/step - loss: 0.3654 - accuracy: 0.8
396 - val_loss: 0.4149 - val_accuracy: 0.8105 - lr: 3.1641e-04
Epoch 29/30
250/250 [==============================] - 7239s 29s/step - loss: 0.3649 - accuracy: 0.8
374 - val_loss: 0.4439 - val_accuracy: 0.7984 - lr: 2.3730e-04
Epoch 30/30
250/250 [==============================] - ETA: 0s - loss: 0.3538 - accuracy: 0.8404
Epoch 30: ReduceLROnPlateau reducing learning rate to 0.00017797851614886895.
250/250 [==============================] - 32s 129ms/step - loss: 0.3538 - accuracy: 0.8
404 - val_loss: 0.4034 - val_accuracy: 0.8175 - lr: 2.3730e-04
```

In [17]:

```
history.history.keys()
```

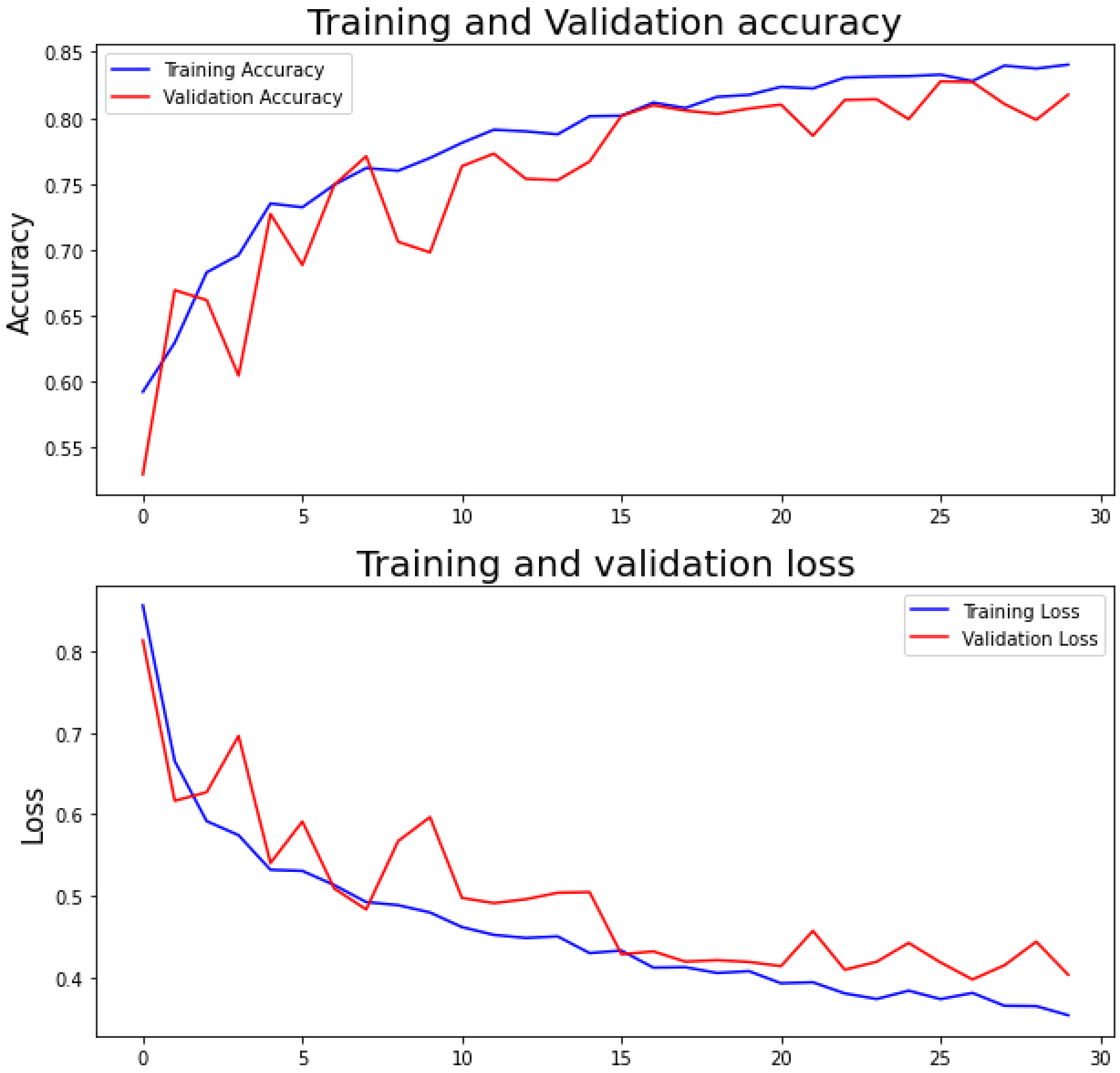
Out[17]:

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy', 'lr'])
```

In [18]:

```python
import matplotlib.pyplot as plt

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

fig = plt.figure(figsize=(10,10),edgecolor='Black')
ax1 = fig.add_subplot(2,1,1)
ax2 = fig.add_subplot(2,1,2)

ax1.plot(accuracy, label='Training Accuracy', color='Blue')
ax1.plot(val_accuracy, label='Validation Accuracy', color='Red')
ax1.set_title("Training and Validation accuracy", fontsize=20)
ax1.set_ylabel("Accuracy", fontsize=15)
ax1.legend()

ax2.plot(loss, label='Training Loss', color='Blue')
ax2.plot(val_loss, label='Validation Loss', color='Red')
ax2.set_title("Training and validation loss", fontsize=20)
ax2.set_ylabel("Loss", fontsize=15)
ax2.legend()
```

```
plt.show()
```



In [43]:

```
print("Training Accuracy:",round(val_accuracy[-1]*100))
print("Accuracy Score:" , round(accuracy[-1]*100))
```

```
Training Accuracy: 82
Accuracy Score: 84
```

In [20]:

```
model.save('MODEL')
```

```
INFO:tensorflow:Assets written to: MODEL\assets
```

INFO:tensorflow:Assets written to: MODEL\assets

# Loding images

In [21]:

```
from keras.preprocessing.image import ImageDataGenerator
from keras.models import load_model
from matplotlib import pyplot as plt
```

In [22]:

```
import cv2
import numpy as np
```

In [23]:

```
model = load_model('MODEL')
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896       
                                                                 
 batch_normalization (Batch  (None, 62, 62, 32)        128       
 Normalization)                                                  
                                                                 
 max_pooling2d (MaxPooling2  (None, 31, 31, 32)        0         
 D)                                                              
                                                                 
 dropout (Dropout)           (None, 31, 31, 32)        0         
                                                                 
 conv2d_1 (Conv2D)           (None, 29, 29, 32)        9248      
                                                                 
 batch_normalization_1 (Bat  (None, 29, 29, 32)        128       
 chNormalization)                                                
                                                                 
 max_pooling2d_1 (MaxPoolin  (None, 14, 14, 32)        0         
 g2D)                                                            
                                                                 
 dropout_1 (Dropout)         (None, 14, 14, 32)        0         
                                                                 
 flatten (Flatten)           (None, 6272)              0         
                                                                 
 dense (Dense)               (None, 64)                401472    
                                                                 
 batch_normalization_2 (Bat  (None, 64)                256       
 chNormalization)                                                
                                                                 
 dropout_2 (Dropout)         (None, 64)                0         
                                                                 
 dense_1 (Dense)             (None, 2)                 130       
                                                                 
=================================================================
Total params: 412258 (1.57 MB)
Trainable params: 412002 (1.57 MB)
Non-trainable params: 256 (1.00 KB)
_________________________________________________________________
```

In [35]:

```
pred_datagen = ImageDataGenerator(rescale=1./255)
pred_set = pred_datagen.flow_from_directory('C:/Users/LEGION/Desktop/prediction',
                                             target_size=(64,64),
                                             class_mode='categorical',
                                             shuffle = False)
```

```
Found 7 images belonging to 2 classes.
```

In [36]:

```
pred_prob = model.predict_generator(pred_set)
pred_prob = np.round(pred_prob*100,2)
```

```
C:\Users\LEGION\AppData\Local\Temp\ipykernel_31160\3110469594.py:1: UserWarning: `Model.
predict_generator` is deprecated and will be removed in a future version. Please use `Mo
del.predict`, which supports generators.
  pred_prob = model.predict_generator(pred_set)
```

In [37]:

```
print(pred_prob)
```

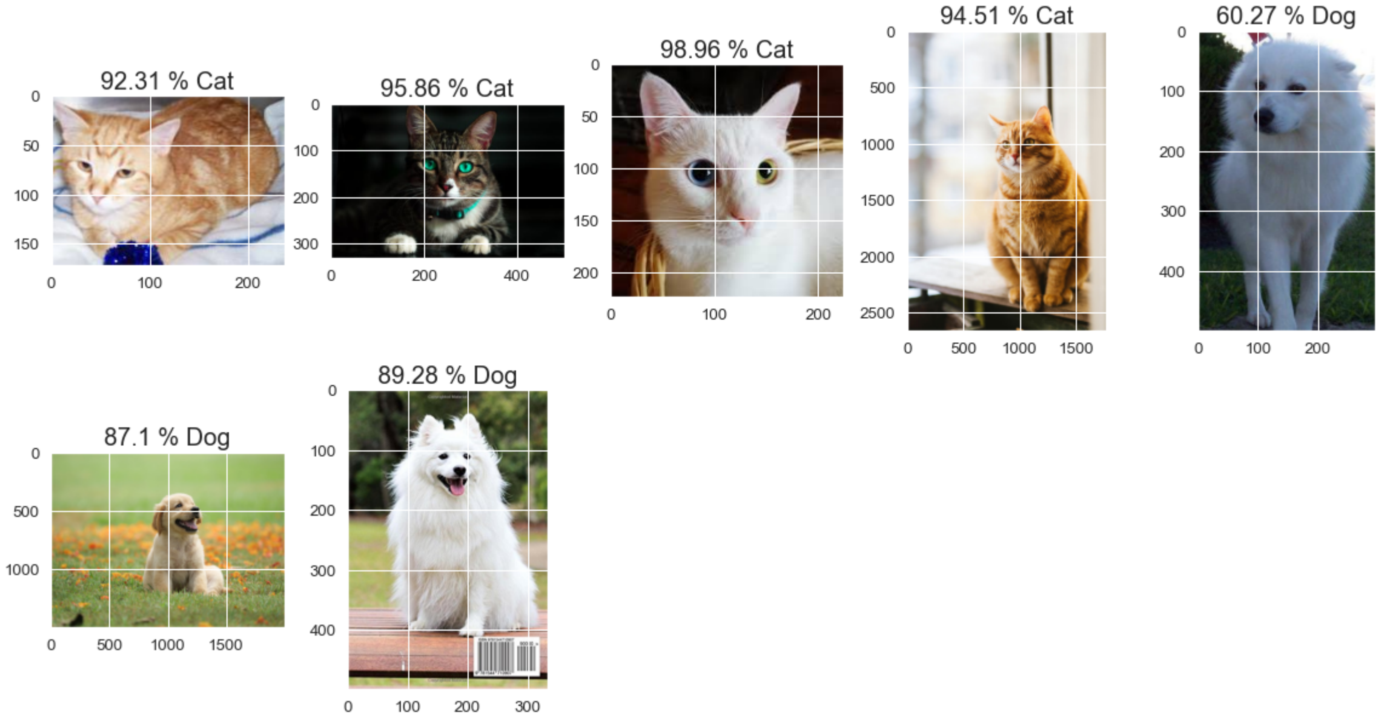
```
[[92.31  7.69]
 [95.86  4.14]
 [98.96  1.04]
 [94.51  5.49]
 [39.73 60.27]
 [12.9   87.1 ]
 [10.72 89.28]]
```

In [38]:

```
image1 = cv2.imread('C:/Users/LEGION/Desktop/prediction//cats/cat_sample_1.jpg')
image2 = cv2.imread('C:/Users/LEGION/Desktop/prediction/cats/cat_sample_2.jpg')
image3 = cv2.imread('C:/Users/LEGION/Desktop/prediction/cats/cat_sample_3.jpg')
image4=cv2.imread('C:/Users/LEGION/Desktop/prediction/cats/cat_sample_4.jpg')


image5 = cv2.imread('C:/Users/LEGION/Desktop/prediction/dogs/dog_sample_1.jpg')
image6 = cv2.imread('C:/Users/LEGION/Desktop/prediction/dogs/dog_sample_2.jpg')
image7 = cv2.imread('C:/Users/LEGION/Desktop/prediction/dogs/dog_sample_3.jpg')



sample1 = image1[:,:,::-1]
sample2 = image2[:,:,::-1]
sample3 = image3[:,:,::-1]


sample4= image4[:,:,::-1]
sample5= image5[:,:,::-1]
sample6 = image6[:,:,::-1]
sample7 = image7[:,:,::-1]


fig = plt.figure(figsize=(20,10))
ax1 = fig.add_subplot(2,5,1)
ax2 = fig.add_subplot(2,5,2)
ax3 = fig.add_subplot(2,5,3)
ax4 = fig.add_subplot(2,5,4)
ax5 = fig.add_subplot(2,5,5)
ax6 = fig.add_subplot(2,5,6)
ax7 = fig.add_subplot(2,5,7)


ax1.imshow(sample1)
ax2.imshow(sample2)
ax3.imshow(sample3)
ax4.imshow(sample4)
ax5.imshow(sample5)
ax6.imshow(sample6)
ax7.imshow(sample7)

#ax10.imshow(sample10)

axis = [ax1, ax2, ax3, ax4, ax5, ax6, ax7]
print(pred_prob)

for i in range(8):
    if pred_prob[i][0] > 50 :
        axis[i].set_title(str(pred_prob[i][0]) +' % Cat',fontsize =20)
    else:
        axis[i].set_title(str(pred_prob[i][1]) +' % Dog',fontsize =20)




plt.show()
```

```
[[92.31  7.69]
 [95.86  4.14]
 [98.96  1.04]
 [94.51  5.49]
 [39.73 60.27]
 [12.9   87.1 ]
 [10.72 89.28]]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Input In [38], in <cell line: 47>()
     45 print(pred_prob)
     47 for i in range(8):
---> 48     if pred_prob[i][0] > 50 :
     49         axis[i].set_title(str(pred_prob[i][0]) +' % Cat',fontsize =20)
     50     else:

IndexError: index 7 is out of bounds for axis 0 with size 7
```



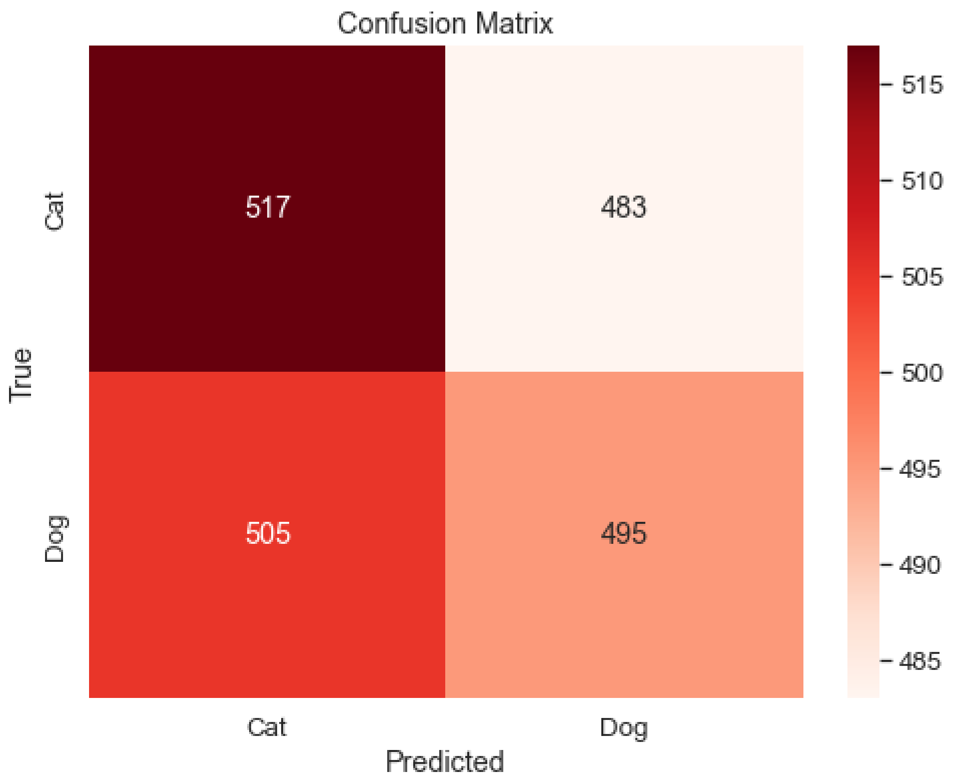
# Confusion Matrix

In [46]:

```
from sklearn.metrics import confusion_matrix
import numpy as np

# Make predictions on the validation set
validation_predictions = model.predict(validation_set)

# Convert predicted probabilities to class labels (0 or 1)
predicted_labels = np.argmax(validation_predictions, axis=1)

# Get true labels from the validation set
true_labels = validation_set.classes

# Create the confusion matrix
confusion = confusion_matrix(true_labels, predicted_labels)

# Display the confusion matrix
print("Confusion Matrix:")
print(confusion)

import seaborn as sns
import matplotlib.pyplot as plt

# Create a confusion matrix (as described in the previous response)

# Set the class names for your labels
```

```
class_names = ['Cat', 'Dog']  # Replace with your class names

# Create a heatmap
plt.figure(figsize=(8, 6))
sns.set(font_scale=1.2)  # Adjust the font size if needed
sns.heatmap(confusion, annot=True, fmt='d', cmap='Reds', xticklabels=class_names, ytickl
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
print("Training Accuracy:",round(val_accuracy[-1]*100))
print("Accuracy Score:" , round(accuracy[-1]*100))
```

```
63/63 [==============================] - 2s 31ms/step
Confusion Matrix:
[[517 483]
 [505 495]]
```



```
Training Accuracy: 82
Accuracy Score: 84
```

# Accuracy , Precision and Recall report

In [48]:

```
from sklearn.metrics import accuracy_score, precision_score, recall_score

# Calculate accuracy
accuracy = accuracy_score(true_labels, predicted_labels)

# Calculate precision
precision = precision_score(true_labels, predicted_labels)

# Calculate recall
recall = recall_score(true_labels, predicted_labels)

print(f'Accuracy: {accuracy:.2f}')
print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
```

```
Accuracy: 0.51
Precision: 0.51
```

```
Recall: 0.49
```