Experiment No. 7

Title: Cat vs Dog Classification Using CNN

Aim: Classify images as either cat or dog using a CNN.

Dataset: Collection of labeled cat and dog images (e.g., Kaggle Dogs vs Cats dataset).

Model Architecture: Convolution Layers: Extract features from the image. Pooling Layers: Reduce dimensionality and retain essential features. Flatten Layer: Convert 2D feature maps to a 1D vector. Dense Layers: Perform classification (with softmax/sigmoid activation). Loss Function: Binary Cross-Entropy (since it's a 2-class problem).

Optimizer: Adam (for faster convergence).

Training: Use ImageDataGenerator for data augmentation and fit the CNN model.

Output: Predict if the input image is a cat or dog.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.callbacks import TensorBoard

from warnings import filterwarnings
filterwarnings('ignore')

classifier = Sequential()
classifier.add(Conv2D(32,(3,3),input_shape=(64,64,3),activation =
'relu'))
classifier.add(MaxPooling2D(pool_size=(2,2),strides=2)) #if stride not
given it equal to pool filter size
classifier.add(Conv2D(32,(3,3),activation = 'relu'))
classifier.add(MaxPooling2D(pool_size=(2,2),strides=2))
classifier.add(Flatten())
classifier.add(Dense(units=128,activation='relu'))
classifier.add(Dense(units=1,activation='sigmoid'))
adam = tensorflow.keras.optimizers.Adam(lr=0.001, beta_1=0.9,
beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
classifier.compile(optimizer=adam,loss='binary_crossentropy',metrics=[
'accuracy'])
#tensorboard = TensorBoard(log_dir="logs/{}".format(time()))
```

Data Augmentation

Using some Data Augmentation techniques for more data and Better results.

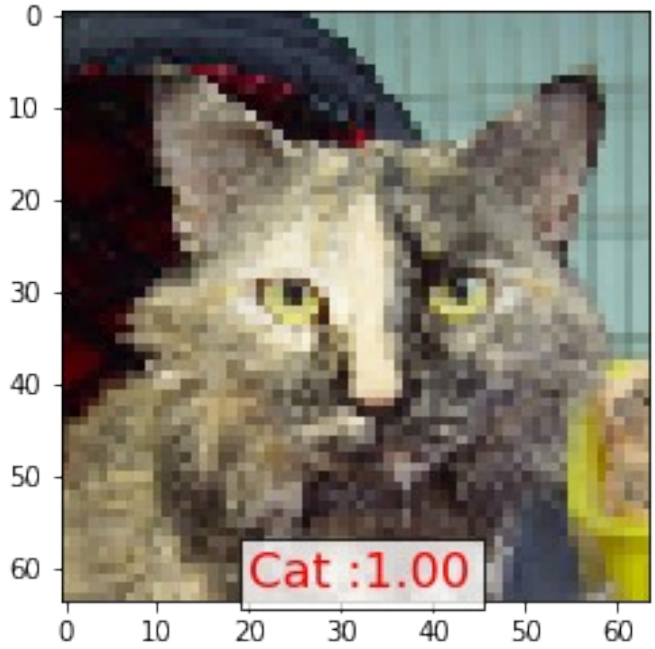
- Shearing of images
- Random zoom

- Horizontal flips

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.1,
                                   zoom_range=0.1,
                                   horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1./255)

#Training Set
train_set = train_datagen.flow_from_directory('train',
                                              target_size=(64,64),
                                              batch_size=32,
                                              class_mode='binary')
#Validation Set
test_set = test_datagen.flow_from_directory('test',
                                            target_size=(64,64),
                                            batch_size = 32,
                                            class_mode='binary',
                                            shuffle=False)
#Test Set /no output available
test_set1 = test_datagen.flow_from_directory('test1',
                                             target_size=(64,64),
                                             batch_size=32,
                                             shuffle=False)


Found 19998 images belonging to 2 classes.
Found 5000 images belonging to 2 classes.
Found 12500 images belonging to 1 classes.

%%capture
classifier.fit_generator(train_set,
                         steps_per_epoch=800,
                         epochs = 200,
                         validation_data = test_set,
                         validation_steps = 20,
                         #callbacks=[tensorboard]
                         );

#Some Helpful Instructions:

#finetune you network parameter in last by using low learning rate
like 0.00001
#classifier.save('resources/dogcat_model_bak.h5')
#from tensorflow.keras.models import load_model
#model = load_model('partial_trained1')
#100 iteration with learning rate 0.001 and after that 0.0001

from tensorflow.keras.models import load_model
classifier = load_model('resources/dogcat_model_bak.h5')
```

Prediction of Single Image

```
#Prediction of image
%matplotlib inline
import tensorflow
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
import numpy as np
img1 = image.load_img('test/Cat/10.jpg', target_size=(64, 64))
img = image.img_to_array(img1)
img = img/255
# create a batch of size 1 [N,H,W,C]
img = np.expand_dims(img, axis=0)
prediction = classifier.predict(img, batch_size=None,steps=1) #gives
all class prob.
if(prediction[:,:]>0.5):
    value ='Dog :%1.2f'%(prediction[0,0])
    plt.text(20,
62,value,color='red',fontsize=18,bbox=dict(facecolor='white',alpha=0.8
))
else:
    value ='Cat :%1.2f'%(1.0-prediction[0,0])
    plt.text(20,
62,value,color='red',fontsize=18,bbox=dict(facecolor='white',alpha=0.8
))

plt.imshow(img1)
plt.show()
```



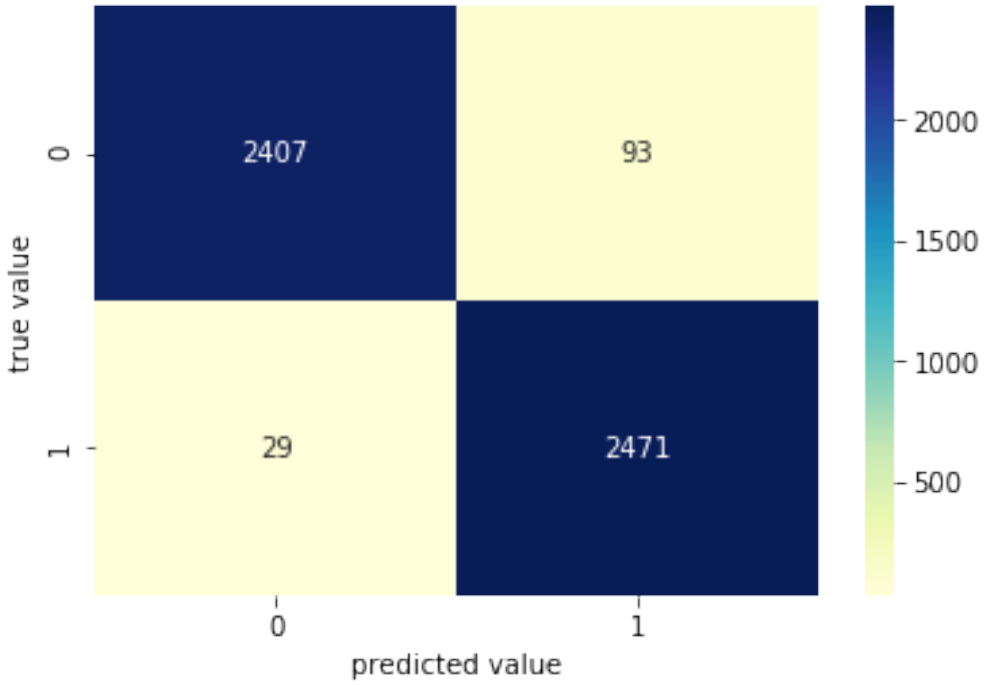
```
import pandas as pd
test_set.reset
ytesthat = classifier.predict_generator(test_set)
df = pd.DataFrame({
    'filename':test_set.filenames,
    'predict':ytesthat[:,0],
    'y':test_set.classes
})

pd.set_option('display.float_format', lambda x: '%.5f' % x)
df['y_pred'] = df['predict']>0.5
df.y_pred = df.y_pred.astype(int)
df.head(10)
```

	filename	predict	y	y_pred
0	Cat/0.jpg	0.00000	0	0
1	Cat/1.jpg	0.00000	0	0
2	Cat/10.jpg	0.00000	0	0
3	Cat/100.jpg	0.99970	0	1
4	Cat/10001.jpg	0.00000	0	0
5	Cat/10009.jpg	0.02340	0	0
6	Cat/1001.jpg	0.00001	0	0
7	Cat/10012.jpg	0.00000	0	0
8	Cat/10017.jpg	0.00000	0	0
9	Cat/10018.jpg	0.00000	0	0

```
misclassified = df[df['y']!=df['y_pred']]
print('Total misclassified image from 5000 Validation images :
%d'%misclassified['y'].count())
```

```
Total misclassified image from 5000 Validation images : 122
```

```
#Prediction of test set
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

conf_matrix = confusion_matrix(df.y,df.y_pred)
sns.heatmap(conf_matrix,cmap="YlGnBu",annot=True,fmt='g');
plt.xlabel('predicted value')
plt.ylabel('true value');
```




```
#Some of Cat image misclassified as Dog.
import matplotlib.image as mpimg

CatasDog = df['filename'][(df.y==0)&(df.y_pred==1)]
fig=plt.figure(figsize=(15, 6))
columns = 7
rows = 3
for i in range(columns*rows):
    #img = mpimg.imread()
    img = image.load_img('test/'+CatasDog.iloc[i], target_size=(64,
64))
    fig.add_subplot(rows, columns, i+1)
    plt.imshow(img)

plt.show()
```

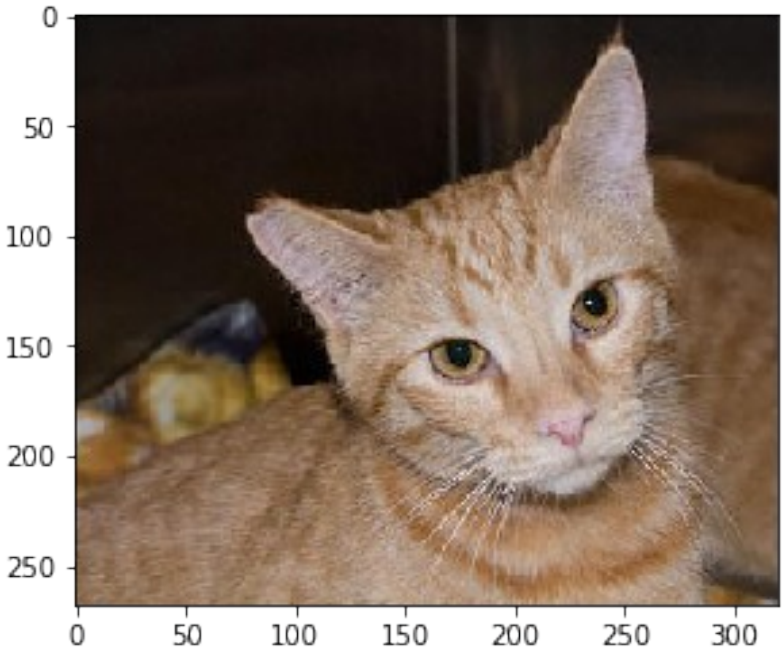
```
#Some of Dog image misclassified as Cat.
import matplotlib.image as mpimg

DogasCat = df['filename'][(df.y==1)&(df.y_pred==0)]
fig=plt.figure(figsize=(15, 6))
columns = 7
rows = 3
for i in range(columns*rows):
    #img = mpimg.imread()
    img = image.load_img('test/'+DogasCat.iloc[i], target_size=(64,
64))
    fig.add_subplot(rows, columns, i+1)
    plt.imshow(img)
plt.show()
```

```
classifier.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_6 (Conv2D)            (None, 62, 62, 32)        896       
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 31, 31, 32)        0         
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 29, 29, 32)        9248      
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 14, 14, 32)        0         
_________________________________________________________________
flatten_3 (Flatten)          (None, 6272)              0         
_________________________________________________________________
dense_6 (Dense)              (None, 128)               802944    
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 129       
=================================================================
Total params: 813,217
Trainable params: 813,217
Non-trainable params: 0
_________________________________________________________________
```

Visualization of Layers Ouptut

```
#Input Image for Layer visualization
img1 = image.load_img('test/Cat/14.jpg')
plt.imshow(img1);
#preprocess image
img1 = image.load_img('test/Cat/14.jpg', target_size=(64, 64))
img = image.img_to_array(img1)
img = img/255
img = np.expand_dims(img, axis=0)
```




```
model_layers = [ layer.name for layer in classifier.layers]
print('layer name : ',model_layers)
```

```
layer name :  ['conv2d_6', 'max_pooling2d_6', 'conv2d_7',
'max_pooling2d_7', 'flatten_3', 'dense_6', 'dense_7']
```

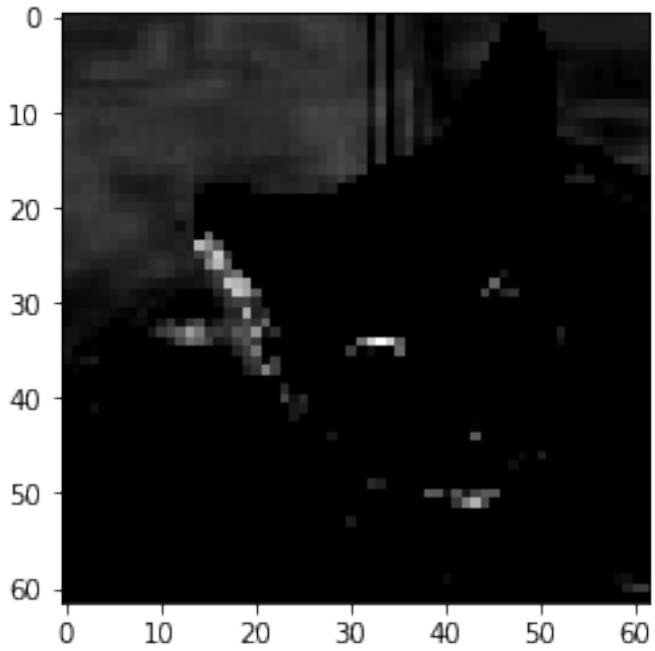
```
from tensorflow.keras.models import Model
conv2d_6_output = Model(inputs=classifier.input,
outputs=classifier.get_layer('conv2d_6').output)
conv2d_7_output =
Model(inputs=classifier.input,outputs=classifier.get_layer('conv2d_7')
.output)

conv2d_6_features = conv2d_6_output.predict(img)
conv2d_7_features = conv2d_7_output.predict(img)
print('First conv layer feature output shape :
',conv2d_6_features.shape)
print('First conv layer feature output shape :
',conv2d_7_features.shape)
```

```
First conv layer feature output shape :  (1, 62, 62, 32)
First conv layer feature output shape :  (1, 29, 29, 32)
```

Single Convolution Filter Output

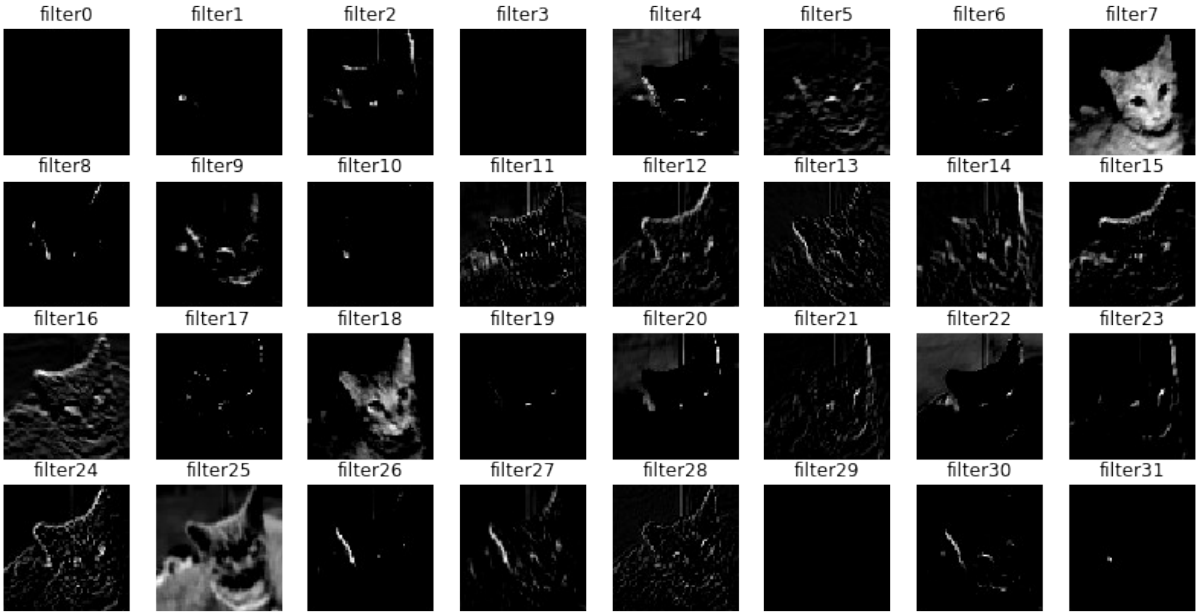
```
plt.imshow(conv2d_6_features[0, :, :, 4], cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f3b1c90f978>
```



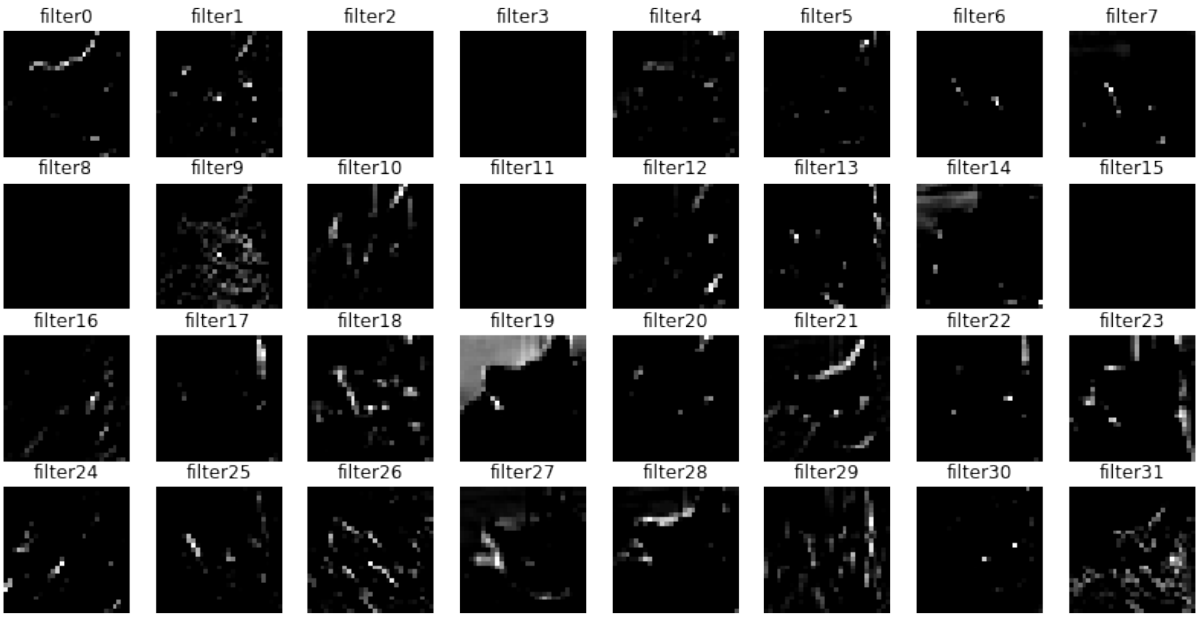
First Covolution Layer Output

```
import matplotlib.image as mpimg

fig=plt.figure(figsize=(14,7))
columns = 8
rows = 4
for i in range(columns*rows):
    #img = mpimg.imread()
    fig.add_subplot(rows, columns, i+1)
    plt.axis('off')
    plt.title('filter'+str(i))
    plt.imshow(conv2d_6_features[0, :, :, i], cmap='gray')
plt.show()
```



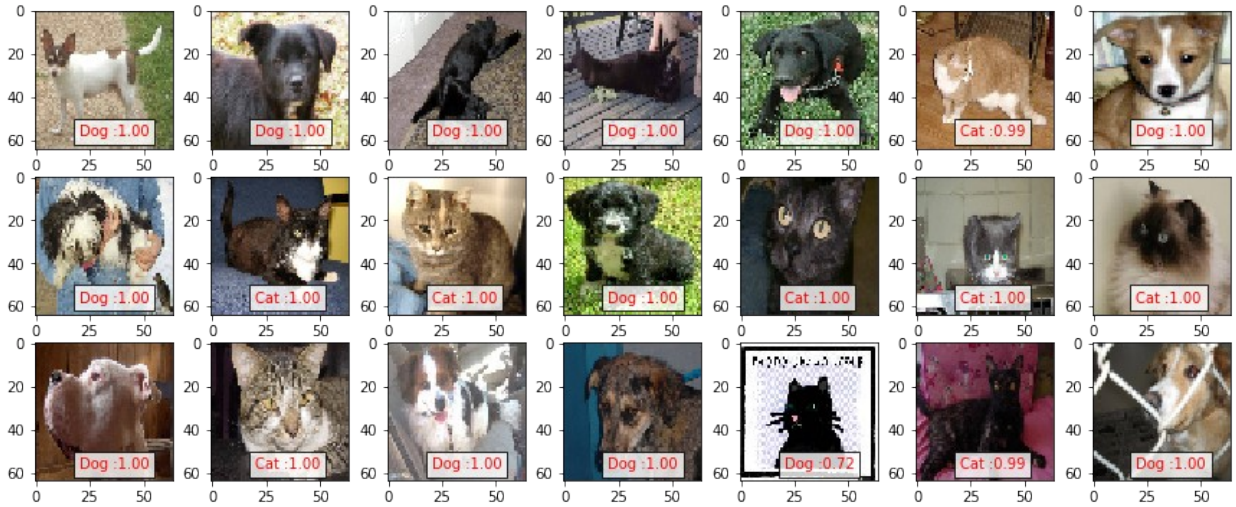
Second Covolution Layer Output

```
fig=plt.figure(figsize=(14,7))
columns = 8
rows = 4
for i in range(columns*rows):
    #img = mpimg.imread()
    fig.add_subplot(rows, columns, i+1)
    plt.axis('off')
    plt.title('filter'+str(i))
    plt.imshow(conv2d_7_features[0, :, :, i], cmap='gray')
plt.show()
```



Model Performance on Unseen Data

```
# for generator image set u can use
# ypred = classifier.predict_generator(test_set)

fig=plt.figure(figsize=(15, 6))
columns = 7
rows = 3
for i in range(columns*rows):
    fig.add_subplot(rows, columns, i+1)
    img1 =
image.load_img('test1/'+test_set1.filenames[np.random.choice(range(125
00))], target_size=(64, 64))
    img = image.img_to_array(img1)
    img = img/255
    img = np.expand_dims(img, axis=0)
    prediction = classifier.predict(img, batch_size=None,steps=1)
#gives all class prob.
    if(prediction[:,:]>0.5):
        value ='Dog :%1.2f'%(prediction[0,0])
        plt.text(20,
58,value,color='red',fontsize=10,bbox=dict(facecolor='white',alpha=0.8
))
    else:
        value ='Cat :%1.2f'%(1.0-prediction[0,0])
        plt.text(20,
58,value,color='red',fontsize=10,bbox=dict(facecolor='white',alpha=0.8
))
    plt.imshow(img1)
```




```
%%capture
# Model Accuracy
x1 = classifier.evaluate_generator(train_set)
x2 = classifier.evaluate_generator(test_set)

print('Training Accuracy  : %1.2f%%     Training loss  : %1.6f'%
(x1[1]*100,x1[0]))
print('Validation Accuracy: %1.2f%%     Validation loss: %1.6f'%
(x2[1]*100,x2[0]))
```

```
Training Accuracy  : 99.96%     Training loss  : 0.002454
Validation Accuracy: 97.56%     Validation loss: 0.102678
```

Conclusion:

The CNN model effectively distinguishes between cats and dogs by automatically extracting key features from images. It achieves high accuracy through proper data preprocessing, tuning, and sufficient labeled data, showcasing CNNs' strength in image classification tasks.