A PROJECT REPORT ON

Image Recognition using Convolution Neural Network

Submitted by

Pranav Kalikate

A deep learning CNN model that can be used to classify images into different classes is built.

Number of epochs and input_shape

The number of epochs and input_shape are taken to be very low in order to compensate for the low computing power.

#Part 1 - Building the CNN

#Importing the Keras libraries and packages

```
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
```

# Initialising the CNN

```
classifier = Sequential()
```

# Step 1 - Convolution

```
classifier.add(Conv2D(32, (3, 3), input_shape = (64, 64, 3), activation = 'relu'))
```

# Step 2 - Pooling

```
classifier.add(MaxPooling2D(pool_size = (2, 2)))
```

# Adding a second convolutional layer

```
classifier.add(Conv2D(32, (3, 3), activation = 'relu'))

classifier.add(MaxPooling2D(pool_size = (2, 2)))
```

# Step 3 - Flattening

```
classifier.add(Flatten())
```

# Step 4 - Full connection

```
classifier.add(Dense(units = 128, activation = 'relu'))

classifier.add(Dense(units = 1, activation = 'sigmoid'))
```

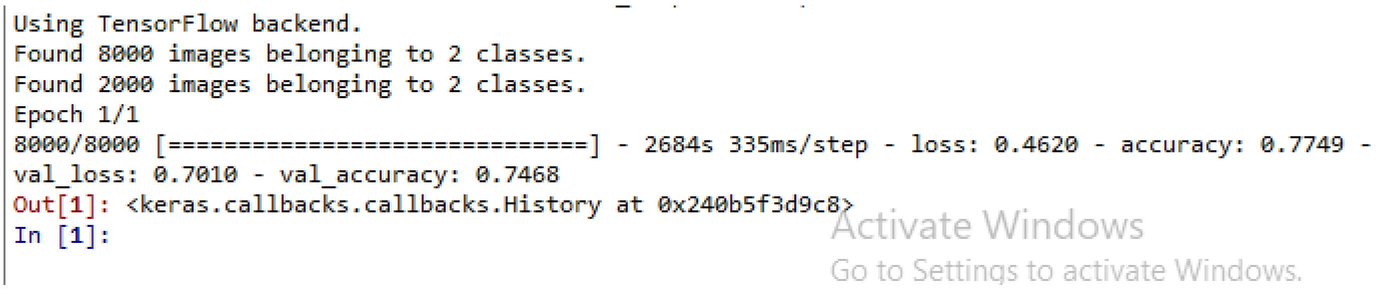
# Compiling the CNN

```
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

<u>**# Part 2 - Fitting the CNN to the images**</u>

```
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale = 1./255)

training_set = train_datagen.flow_from_directory('dataset/training_set',
                                                 target_size = (64, 64),
                                                 batch_size = 32,
                                                 class_mode = 'binary')
test_set = test_datagen.flow_from_directory('dataset/test_set',
                                            target_size = (64, 64),
                                            batch_size = 32,
                                            class_mode = 'binary')
classifier.fit_generator(training_set,
                         steps_per_epoch = 8000,
                         epochs = 1,
                         validation_data = test_set,
                         validation_steps = 2000)
```



```
Using TensorFlow backend.
Found 8000 images belonging to 2 classes.
Found 2000 images belonging to 2 classes.
Epoch 1/1
8000/8000 [==============================] - 2684s 335ms/step - loss: 0.4620 - accuracy: 0.7749 -
val_loss: 0.7010 - val_accuracy: 0.7468
Out[1]: <keras.callbacks.callbacks.History at 0x240b5f3d9c8>
In [1]:
```

Accuracy on training set = 77%

Accuracy on test set = 74%

Increase the number of epochs so that our model will learn to better detect features in an image.

# Part 3 - Making new predictions

```
import numpy as np

from keras.preprocessing import image

test_image = image.load_img('dataset/single_prediction/cat_or_dog_2.jpg', target_size = (64,
64))

test_image = image.img_to_array(test_image)

test_image = np.expand_dims(test_image, axis = 0)

result = classifier.predict(test_image)

training_set.class_indices

if result[0][0] == 1:

    prediction = 'dog'

else:

    prediction = 'cat'
```

This block of code can be used to assess whether the object in an image belongs to dog class or cat class.

Accuracy of the CNN model can be improved by:

- Adding more convolution layers or
- Adding another full connection layer or
- By increasing the target size

Image Augmentation:

Image Augmentation is used in the following model to avoid the overfitting since the training set consists of only 8000 images. Image Augmentation enriches the dataset without adding more images. It performs some random transformations such as rotating, flipping, shifting, and shearing on the images in our dataset.