University Of Maryland

College Park

ENPM - 673 Perception of Autonomous Robots

Project 6

May 10, 2020

Govind Ajith Kumar - 116699488
Rajeshwar N S - 116921237
Ashwin Kumar Prabhakaran - 117030402

Libraries used :

```
import numpy as np
import pandas as pd
import sys
from keras.preprocessing.image import ImageDataGenerator, load_img
from keras.utils import to_categorical
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import random
from IPython.display import display
from PIL import Image
import tensorflow.compat.v1 as tf
import os
import PIL
from tensorflow.keras.callbacks import ModelCheckpoint
import seaborn as sns
from keras.models import Sequential
from keras import layers
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense, Activation,GlobalMaxPooling2D
from keras import applications
from keras import optimizers
from keras.applications import VGG16
from keras.models import Model
```

Github Link:
magenta`https://github.com/govindak-umd/ENPM673/tree/master/Project%206`

Aim:
The main of aim of this project is to classify the given data-set of images as dogs or cats. The data-set consists of 25000 pictures in which this classification must be carried out.

1 Answer

Convolution Neural Network
The Convolution Neural Network(CNN) is a deep learning algorithm which is used to take input as an image and study the different aspects of the same and then differentiating the image from other types of images. In this project the images are that of a dog and cat.

The structure of CNN is similar to that of the human brain Visual Cortex. Unlike other methods, which are hand engineered, CNN has the ability to learn the characteristics of the images.

CNN is a type of feed froward neural network but differs in its spatial arrangement.Each layer of the the CNN is arranged in the form of 2D grids called as featured maps. Unlike other methods in which the input to an unit is given from a layer below it, in this method, the feature maps are a convolution filter that has been performed on a layer below it. The CNN can easily capture the spatial and temporal dependencies of an image using the appropriate filters.In general the network can be made to learn about the image sophistication better.

There are many color spaces in which the image can exists such as RGB, Grayscale, HSV, etc. What CNN does best is that it reduces the image form such that it is easy to process them without losing many of its features. In order to get more detail on this we need to look into convolution layers.

Convolution Layers
The main use of using convolution layers is to get the high level features of the image to be extracted. The first convolution layer is used to get the low level features of the image such as color, orientation, edges, etc. With more and more convolution layer added to the network it is much more easier to get more high level features of the image.

The element which is used to carry out the convolution is known as a Kernel. the kernel is made to traverse through the entire width and depth of the image to give a squashed one-depth channel Convoluted Feature Output.

After this there can be two operations that can be carried out on the convoluted image obtained, one is increasing the dimensionality and the other increasing or keeping it the same compared to the input image.This process is known as applying paddling,i.e. either valid paddling or same paddling.

Pooling Layers
Pooling layer is something which is similar to that of convolution layer which helps in reducing the spatial size of that of the convolved image.There are two types of pooling which can be applied,

1. **Max Pooling**
   Max Pooling gives back the maximum value that has been covered by the kernel in the image. It also helps in denoising the image by performing noise suppressants and also reducing the dimensionality of the image.
2. **Average Pooling**
   As the name suggests it returns the average value of the portion covered by the kernel. Also it only reduces the dimensions and no noise reduction. Hence we can say that the max pooling acts better than the average pooling.

These steps that were performed helps in processing the image and understanding the features of the image. Depending on the complexity of the image, the number of convolution and pooling layers required might vary so as to get in more details of the image but the computational cost might become high.

Optimizer

The main objective is to reduce the difference between the predicted output and the input. In order to do so we will have to optimize the results obtained.This will help in predicting data that was not seen before also. So in order to minimize the cost function we find the optimized value for weights. To find the optimized value we run many iterations with different weights which is known as gradient descent.There are three types of gradient descent variants, such as,

1. Batch Gradient Descent
2. Stochastic Gradient Descent
3. Mini-Batch Gradient Descent

In this project we have implemented Stochastic Gradient Descent (SGD) optimizer on the images.

Stochastic Gradient Descent

In general gradient descent is a way to minimize the objective cost $J(\theta)$ for a model where $\theta \in \mathbb{R}^d$ by upgrading the parameters in the opposite direction of the gradient objective function $\Delta_\theta J(\theta)$ with respect to the parameters.The size of the steps that has to be taken to reach the local minimum is determined by the learning rate (η).

The SGD function is represented mathematically as,

$$\theta = \theta - \eta * \Delta_\theta J(\theta; x^{(i)}; y^{(i)}) \tag{1}$$

Unlike other gradient descents, in SGD the redundant computations are done one update at a time and hence it is more widely used as it is faster compared to other methods.The learning rate is an important factor to obtain the correct minimum required. Hence in order to reduce the fluctuations, the learning rate must be decreased slowly there by giving the minimum more precisely without much fluctuations.

Activation Function

In neural network the activation function is used to transform the weighted input into the required output, i.e. the output for the input. One such function is the Rectified Linear Activation(ReLu) function. It is a function which will give the output for the input if it is positive and if zero no output.

In order to use the SGD we need a activation function which linearly transform the required function but in reality is a nonlinear function which helps in learning the complex relationships between the data. The ReLu function does exactly the same in addition to providing more sensitivity and avoiding the saturation of the input.

The node in which this function is implemented is known as rectified linear activation unit.The function has a lot of the desirable properties of a linear activation function when training a neural network using back propagation and gives a linear output for values greater than zero but acts as a nonlinear function when the value is in negative giving the output as zero. This function is also known as piece wise linear function because for one half of the input it acts as liner function and nonlinear function for other half domain.

The other activation function is the Sigmond Activation. Comparing to Relu there are a few disadvantages because of which sigmond activation is not mostly used. The problem of saturation occurs in the sigmond activation function. The saturation occurs regardless of whether the input information is useful or not.Once saturated, it becomes challenging for the learning algorithm to continue to adapt the weights to improve the performance of the model.

The main advantages of the ReLu function are,

- Computational Simplicity
  This function unlike other functions does not require to compute the exponential factor.
- Linear Behaviour
  Network which is close to a linear function helps in avoiding the problem of vanishing gradient as this helps in maintaining the proportionality of the node activation.
- Representational Sparsity
  In this function even negative inputs provide a output, which is zero, which is called as sparse representation. this property is very much required as it simplifies the model and also can boost the learning process.
- Train Deep Networks
  This function can be made to explore improvements and made to applied in multi layered networks and hence is more preferred.

VGG-16

VGG-16 is a convolution neural network configuration architecture which consists of 16 layers such as Convolutional layers, Max Pooling layers, Activation layers, Fully connected layers.In total there are actually 21 layers - 13 convolution layers, 5 max pooling layers, 3 dense layers, but it accounts to only 16 weighted layers.The width of convolution layers is small, starting from 64 in the first layer and then increasing by a factor of 2 after each max-pooling layer, until it reaches 512.

The usage of VGG-16 has certain drawbacks due to its large size. It is very slow and takes a lot of time to run.The architecture of the network is weighted very large.Due to more depth and number of fully connected nodes it is difficult to train this architure.

The various layers of the VGG-16 are:

1. Convolution using 64 filters
2. Convolution using 64 filters + Max pooling
3. Convolution using 128 filters
4. Convolution using 128 filters + Max pooling
5. Convolution using 256 filters
6. Convolution using 256 filters
7. Convolution using 256 filters + Max pooling
8. Convolution using 512 filters
9. Convolution using 512 filters
10. Convolution using 512 filters+Max pooling
11. Convolution using 512 filters
12. Convolution using 512 filters
13. Convolution using 512 filters+Max pooling
14. Fully connected with 4096 nodes
15. Fully connected with 4096 nodes
16. Output layer with Softmax activation with 1000 nodes.

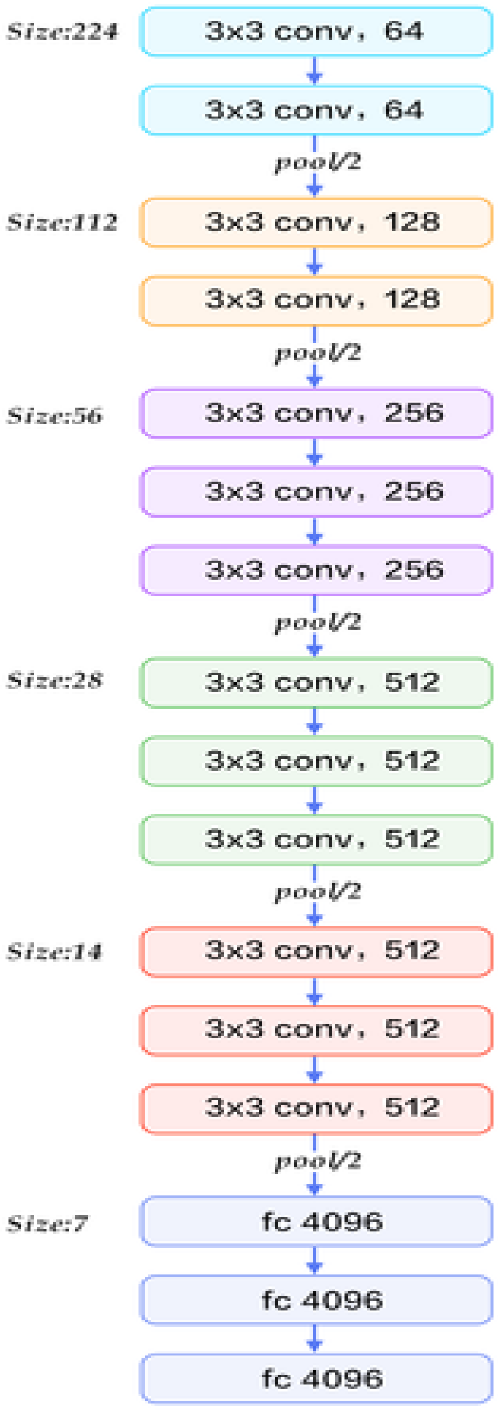


Figure1 -Different Layers of VGG-16

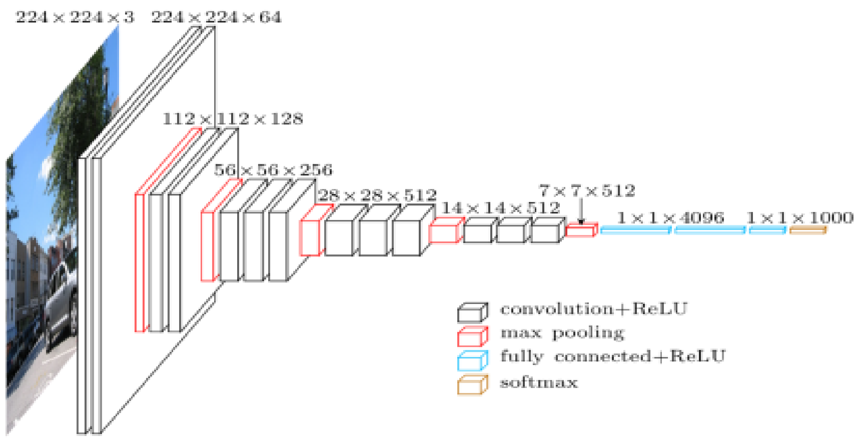


Figure2 -Architecture of VGG-16

Model Summary

Layer (type)	Output Shape	Param
input_1 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
global_max_pooling2d_1 (Glob	(None, 512)	0
dense_1 (Dense)	(None, 512)	262656
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 1)	513

Total params: 14,977,857
Trainable params: 7,342,593
Non-trainable params: 7,635,264

Tensorflow

Tensorflow is a software library which is used for the computation of the numerical using graph data flow.In a graph, the nodes represent the mathematical operations, while the edges of the graph represents the multidimensional data arrays, which is also called as tensor, which passes between them. The main advantage of this architecture is that it can order many computations to one or more GPU or CPU without having to rewrite the code.

Generally, while creating the graph we can specify where the operations must be performed, i.e.,GPU or CPU. If nothing is specified, the system first checks the GPU for free space to perform the operation.The graph can be run using data which can be given from outside also and then running it multiple times for various input.

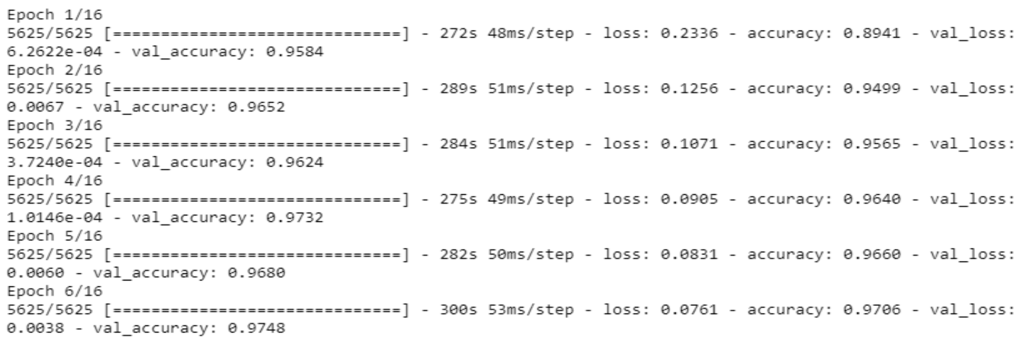
There lies some significant advantage by running the program in GPU rather in CPU. One of the main differences in running the code in the GPU than in the CPU is that for many parallel problems, we can achieve a greater speeds and also many of the programming problems are difficult to formulate as they are and only parallelism is suitable.

CUDA10

CUDA is a development tool chain which is used to run the programs created in the system GPU and also create a controller that helps in controlling such program from the CPU.It is used to run massively parallel SIMD architecture. So the number of operations that can be carried out is more compared to that run in the CPU with same cost but yielding a better result.It uses a general purpose language instead of the pixels and vertex shaders to emulate the general purpose computers.

RESULTS OBTAINED

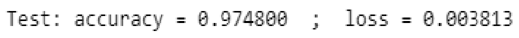
The model was trained on VGG16 architecture and the training was done for 16 epochs. However, early stopping was employed along with a few other callbacks, as seen in the code and the training can be seen as below:



```
Epoch 1/16
5625/5625 [==============================] - 272s 48ms/step - loss: 0.2336 - accuracy: 0.8941 - val_loss:
6.2622e-04 - val_accuracy: 0.9584
Epoch 2/16
5625/5625 [==============================] - 289s 51ms/step - loss: 0.1256 - accuracy: 0.9499 - val_loss:
0.0067 - val_accuracy: 0.9652
Epoch 3/16
5625/5625 [==============================] - 284s 51ms/step - loss: 0.1071 - accuracy: 0.9565 - val_loss:
3.7240e-04 - val_accuracy: 0.9624
Epoch 4/16
5625/5625 [==============================] - 275s 49ms/step - loss: 0.0905 - accuracy: 0.9640 - val_loss:
1.0146e-04 - val_accuracy: 0.9732
Epoch 5/16
5625/5625 [==============================] - 282s 50ms/step - loss: 0.0831 - accuracy: 0.9660 - val_loss:
0.0060 - val_accuracy: 0.9680
Epoch 6/16
5625/5625 [==============================] - 300s 53ms/step - loss: 0.0761 - accuracy: 0.9706 - val_loss:
0.0038 - val_accuracy: 0.9748
```

Model Training

The final accuracy after model testing, that was obtained is as follows:

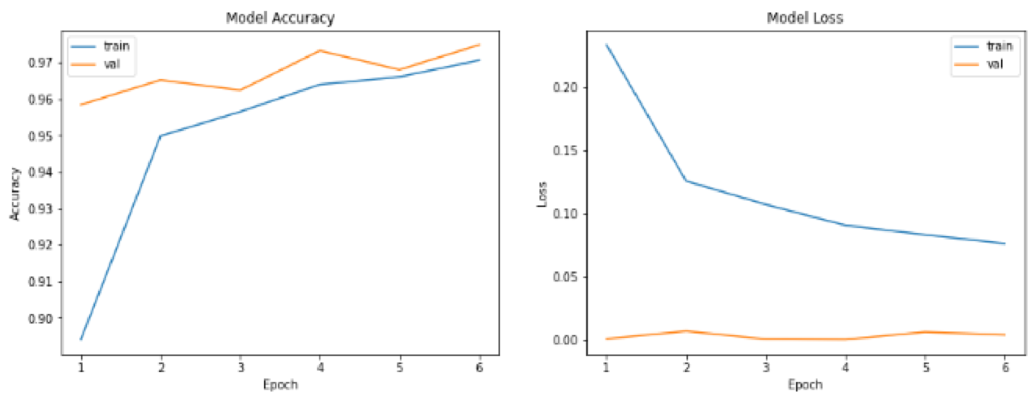


```
Test: accuracy = 0.974800  ;  loss = 0.003813
```

Testing Accuracy

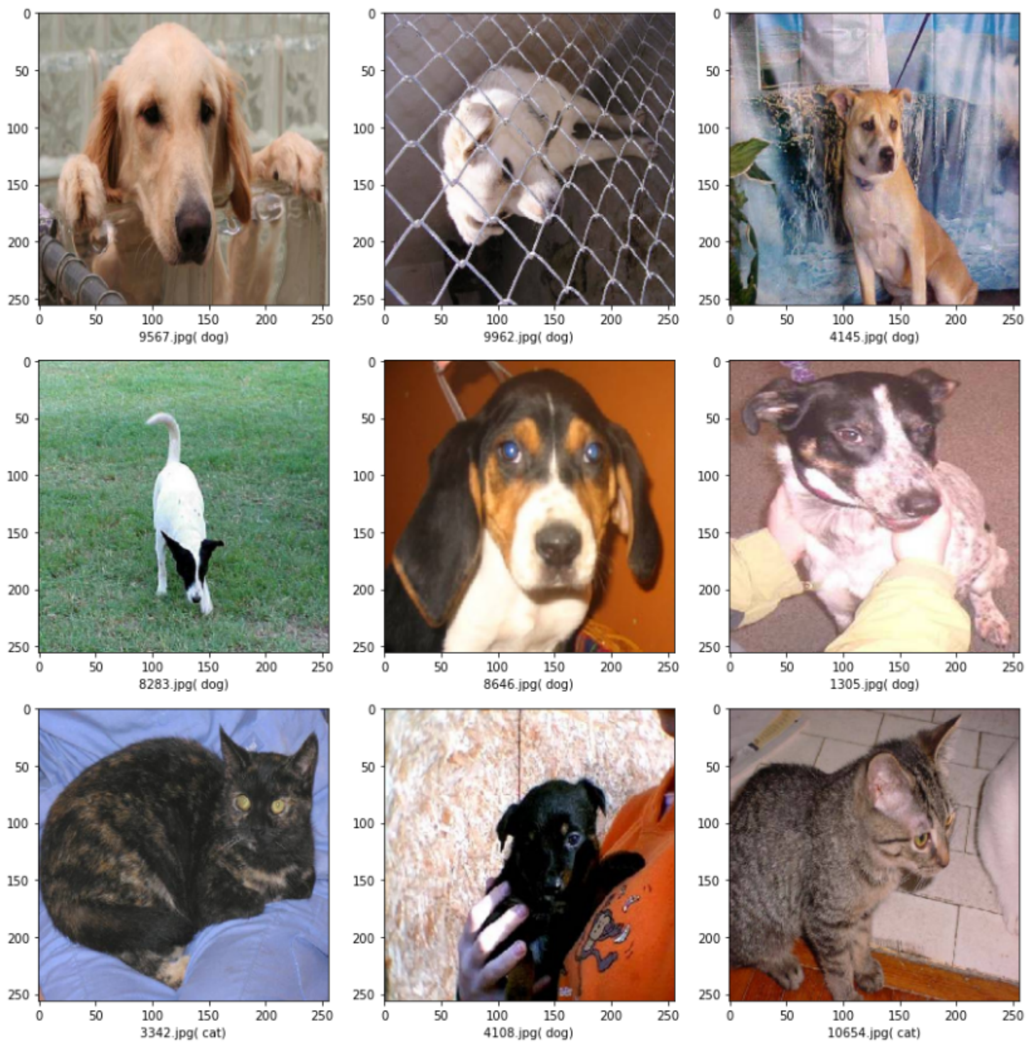
Plots Obtained

The plots for **Accuracy over epoch** and the plots for **Loss over epoch** can be seen below:



Model Training

A random set of 9 images when tested showed the results as shown in the image below:



Validation of testing for random image

Reference
For Figure 1 and Figure 2:
https://iq.opengenus.org/vgg16/