ENPM673 Perception for Autonomous Robots

PROJECT 6

Image Classifier Using Convolution Neural Network

Nalin Das (116698290)
Aditya Khopkar (116911627)
Nidhi Bhojak (116787529)

Contents

List of Figures

1 Introduction

In this project, we are provided with the classic Dog Vs Cat Dataset. The dataset is divided into 2 parts i.e Training and Testing.The training dataset contains 25,000 images and the testing dataset contains 12,500 images. Given an input image, we need to classify whether it is a Dog image or Cat image. The aim is to implement a Convolution Neural Network (CNN) to perform the image classification task.

2 What is Convolution Neural Network?

A Convolution Neural Network(CNN) or ConvNet is a feed-forward artificial neural network. The algorithm takes in an input image, assigns importance(weights) to various features of the image so that the model is capable enough to differentiate features from one another. The pre-processing required in a ConvNet is much lower as compared to other classification algorithm.
A ConvNet is successfully able to capture the spatial and temporal dependencies in an image through the application of relevant filters. In short, the role of ConvNet is to reduce the images into a form which is easier to process, without loosing features which are critical for the image.

3 Building the CNN

There are 3 main components of a CNN,

1. Convolution Layer
2. Pooling Layer
3. Fully Connected Layer

Convolution layer comes first. The input image is entered into it and algorithm then selects a suitable smaller size matrix called as filter. The filter moves along the input image and multiplies its values by the original pixel values. All these multiplications are summed up and one number is obtained in the end. After passing the filter across whole image, a matrix is obtained, but smaller then the input matrix. This matrix is called the feature map. Fig.[1] demonstrates the convolution operation.

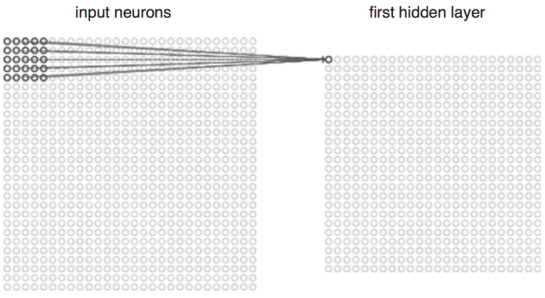

Figure 1: Convolution Operation

The above mentioned operation identifies boundaries and simple contours of the image. In order to recognize the properties of a higher level we need to build a network. This network will consist of several convolutional layer mixed with additional pooling and fully connected layers. When the image passes through one convolution layer, the output of the first layer becomes input for the second layer.

An additional operation called ReLU (Rectified Linear Unit) is used after every Convolution operation. It is a an activation function which introduces non-linearity to the network. The function is a decision making function that determines the presence of a particular neural feature. Mathematically, it is defined as,

$$y = max(0, x)$$

If $x > 0$, the volume of the array of pixels remains the same, and if $x < 0$ it cuts off the particular neuron.

Pooling Layer follows afterwards. It basically works with width and height of the image and performs downsampling operation on them. As a result, the volume of the image is reduced. This means that, if some of the features have already been identified in the previous convolution operation, then the detailed image is no longer needed for further processing and hence it has been compressed to less detailed images.

After series of convolution and pooling layers, we attach a fully connected layer at the end. The objective of using this layer is to take the results of previous layers and use them to classify image into a label. The output is first flattened into a single vector of values,each representing a probability that a certain feature belongs to a label. This fully connected layer goes through its own back propagation process to determine the most accurate weights. Each neuron receives weights that prioritize the most appropriate label. Finally the neurons vote on each of the labels and the maximum value label determines the result of classification. Below Fig.[2] shows the working of Fully Connected Layer.

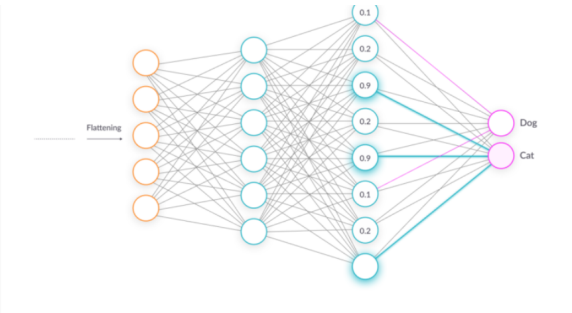

Figure 2: Fully Connected Layer

4 Our Approach

The flow for task implementation is as follows,

1. Model Construction
2. Model Training
3. Model Testing
4. Model Evaluation

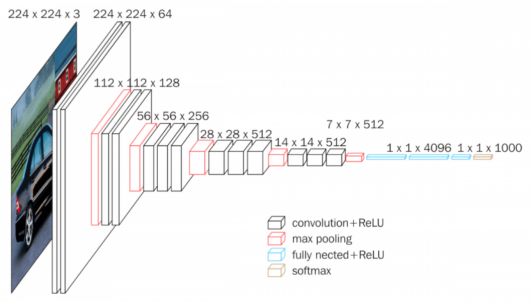

Figure 3: VGG 16 Architecture

4.1 Model Parameters

The architecture we used for this project is VGG-16 which is shown in the Fig.[6]. It has 16 layers in total. There are 13 convolutional layers, 5 max pooling layers and 3 Dense layers which sums upto 21 layers out of which only 16 are weighted. We have implemented the model using TensorFlow.

Looking at the convolution layer, we have a 3 $\times$3 matrix which corresponds to the height and width of the kernel window/filter. As described previously, the activation function used here is ReLU. MaxPooling2D layer is pooling operation for spatial data. The pool size we have used is 2.

Next step is model compiling. This model has a **cross entropy** loss function which shows sum of all individual losses. The optimizer function utilized is **Stochastic Gradient Descent (SGD)**. The rest of the hyperparameters are mentioned below,

Learning rate = 0.01
Decay steps = 10000
Decay Rate =0.9
Epochs = 50
Steps per epoch = 261
Batch Size = 64
Here, Batch size refers to the number of training samples utilized inorder to make a single update to the model parameters.

5 Results

The training dataset was further divided into 2 parts. The model was trained for 67% of the training data and then tested using the rest 33% . This approach is used to get the idea about Validation Accuracy and Validation Loss. Below figures show the results we obtained by setting the hyperparameters according to the previous section. Also, the Output csv file created for Testing Dataset can be found from the attached zip folder. Here in the output file, Label Value "1" is for Dog Image and Label Value "0" is for Cat Image.

The above testcase was tested with batch size as 32. But it was observed that the validation accuracy obtained was only 73%. With the increase in Batch size to 64, the validation accuracy increased to 78.5%.

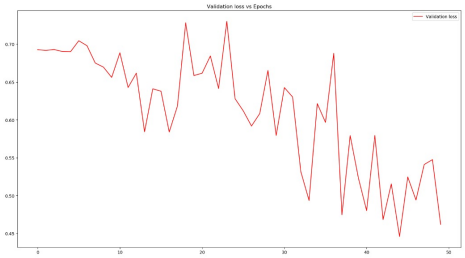

Figure 4: Validation Loss Vs. Epoch- Obtained Validation Loss-0.4620

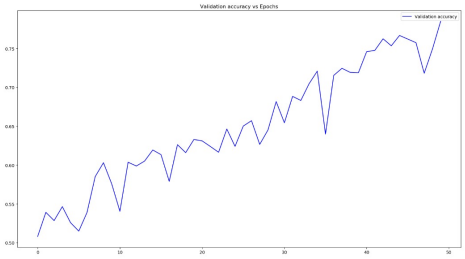

Figure 5: Validation Accuracy Vs. Epoch- Obtained Validation Accuracy-78.3%

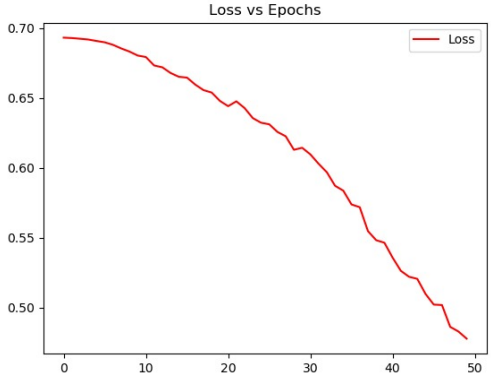

Figure 6: Loss Vs. Epoch- Obtained Minimum Loss-0.4778

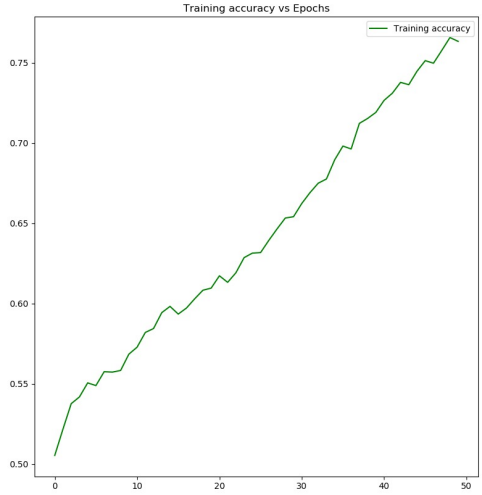

Figure 7: Accuracy Vs. Epoch- Obtained Maximum Accuracy- 76.33%