CAPSTONE PROJECT MLND-Vidit Shah

Subject : Classifying INDIA and USA FLAGS

We will use Convolutional Neural Networks(CNN) for distinguishing the flags

CNN is used mainly for image classification.

Domain Background:

This is an Image classification task.
Image classification involves processing images and extracting features out of it.
There are many ways to approach Image processing one can use software like matlab to detect features from it.
We use CNN to detect features which are learnt in weight formats(Architecture is Explained later).

Problem Statement

We are given images of flag(USA and India).
We need to develop a network which can classify whether the image contains flag on India or USA
This is a binary Classification problem(2 outputs)

Dataset:

I have created my own Dataset of Flag images of India and USA
I have created this Dataset by downloading it from google.
They Vary in sizes .
We use Open CV library to size them 55 by 55 for every input data.
The color channel is RGB so the total Input size is :
55 by 55 by 3
Few Images from Data Set:

I have total of 200 images of each India and USA.

I have splitted them in 150 training and 50 testing for both India and USA

Few sample Images are as follows

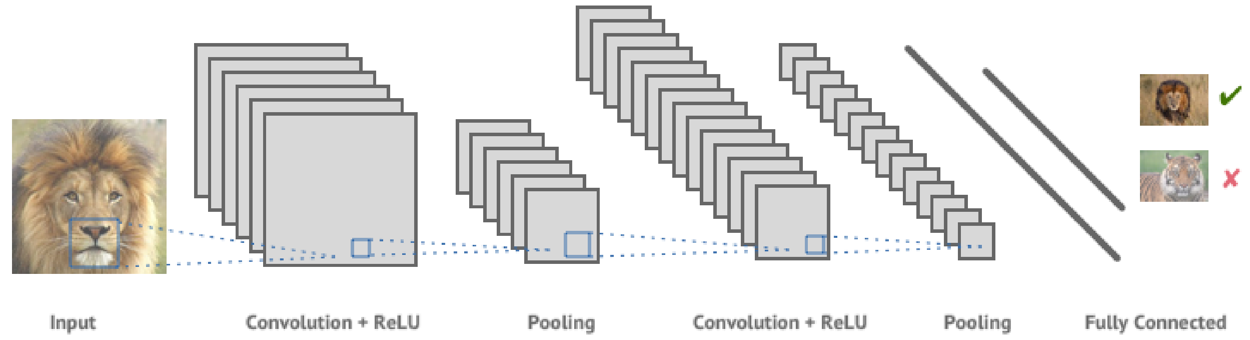
Metrics:

A loss function can be defined in many different ways but a common one is MSE (mean squared error), which is ½ times (actual - predicted) squared.

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

CNN Architecture:(Design)-Workflow

The Figure below shows Basic CNN Architecture



Let's look at the architecture of the convolutional Neural network.

Preprocessing.
We need to clean the incoming data with appropriate size.
I have created a helper class which takes care of the following process.
It resizes the image to feed into the network

The BAsic Architecture of CNN looks as the block diagram shown below:

Input -> Conv -> ReLU -> Conv -> ReLU -> Pool -> ReLU -> Conv -> ReLU -> Pool ->Fully Connected

It consists of the following steps:

1. Input Image
2. Convolutional Layer
3. Pooling
4. Activation Function
5. Fully Connected
6. Output -Classification.

Now Let’s Understand Each step clearly.

Note:We can use steps 2-3-4 how many time we wish in order to increase the Accuracy but it comes with cost of training time(High Computation)

Input Image:

First step,involved in CNN is the input image.We fix the Size of the image(Height and width).

We resize the input image with the help of openCV library to a fix size say 35 by 35 so to feed into our network

Next comes the Convolutional Layer;

Convolutional Layer(CNN)

Imagine one holding a torch and going through image row by row.

Consider the output from the torch as patch which moving over the image from left to right and shifting by certain amount of space.

Each patch stores some data and moves to the next patch .This happen till the patch has received from the top left of the image to bottom right of the image.

The features which the patch gathers are stored in a stack over the other(Increasing the depth)

One can run how many patches he/she wishes to run on the network.

Now Comes the maxPooling

MaxPooling

Maxpooling takes the highest value from the matrix of the given patch into consideration.

This process is very beneficial as only useful features are passed on to next task.

Consider a 2 by 2 matrix

2   4

5   8

Assume this matrix as one of the patch.Only the highest value from patch will be taken into consideration.

For the above example value 8 will be taken under Consideration.

Activation Layer(Relu).

We use Activation layer as a threshold for a neuron to be allowed to proceed to next stage or not.

Here we use RELU(Rectified Linear Unit) as a activation function.

A rectified linear unit has output 0 if the input is less than 0, and raw output otherwise. That is, if the input is greater than 0, the output is equal to the input. ReLUs' machinery is more like a real neuron in your body.

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

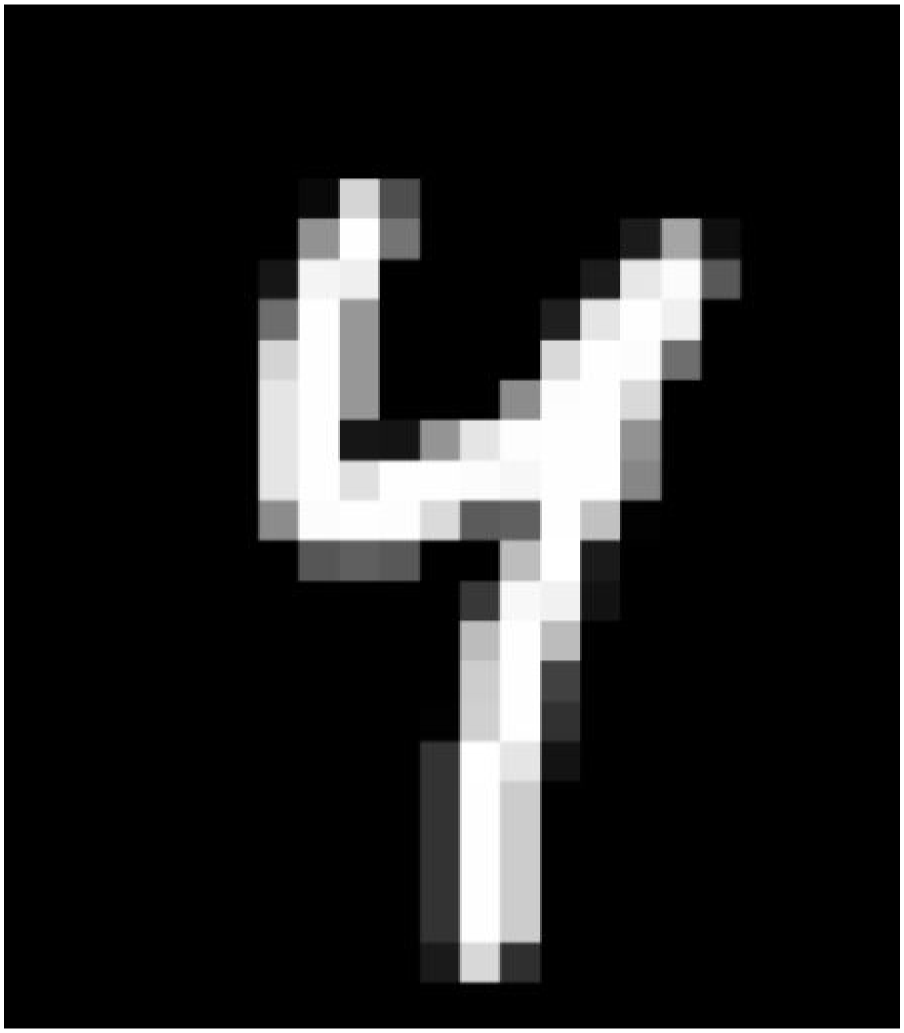
ReLU activations are the simplest non-linear activation function one can use, obviously. When you get the input is positive, the derivative is just 1, so there isn't the squeezing effect you meet on backpropagated errors from the sigmoid function.

Fully Connected Layer.

Here ,output from the activation layer is Flatten(reduced to output classification dimension.

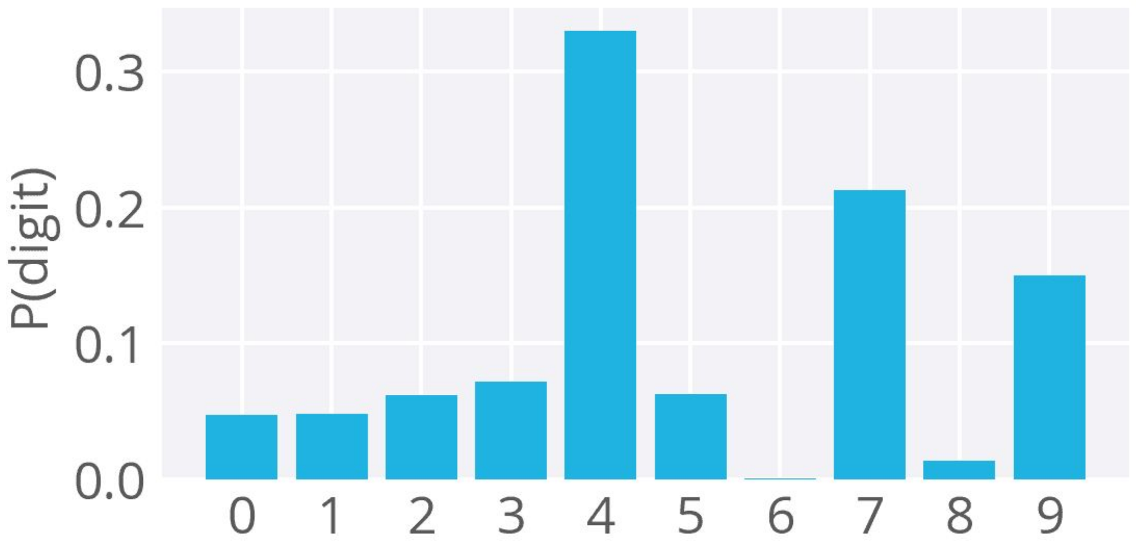
The fully Connected layer is connected to a softmax Activation function which gives probability distribution of the classification with the range of 1.

To understand this better, think about training a network to recognize and classify handwritten digits from images. The network would have ten output units, one for each digit 0 to 9. Then if one fed it an image of a number 4 (see below), the output unit corresponding to the digit 4 would be activated.



Building a network like this requires 10 output units, one for each digit. Each training image is labeled with the true digit and the goal of the network is to predict the correct label. So, if the input is an image of the digit 4, the output unit corresponding to 4 would be activated, and so on for the rest of the units.

For the example image above, the output of the softmax function might look like:



The image looks the most like the digit 4, so you get a lot of probability there. However, this digit also looks somewhat like a 7 and a little bit like a 9 without the loop completed. So, you get the most probability that it's a 4, but also some probability that it's a 7 or a 9.

The softmax can be used for any number of classes. It's also used for hundreds and thousands of classes, for example in object recognition problems where there are hundreds of different possible objects.

Benchmark

We can use SVM,Naive Bayes as an alternative to CNN.

These were used previously for this purpose and I find them Suitable

References

1. http://cs231n.github.io/convolutional-networks/
2. https://www.youtube.com/watch?v=2pQOXjpO_u0
3. https://in.udacity.com/course/deep-learning-nanodegree-foundation--nd101/?