Saurabh Hinduja

Neural Network

Homework # 4

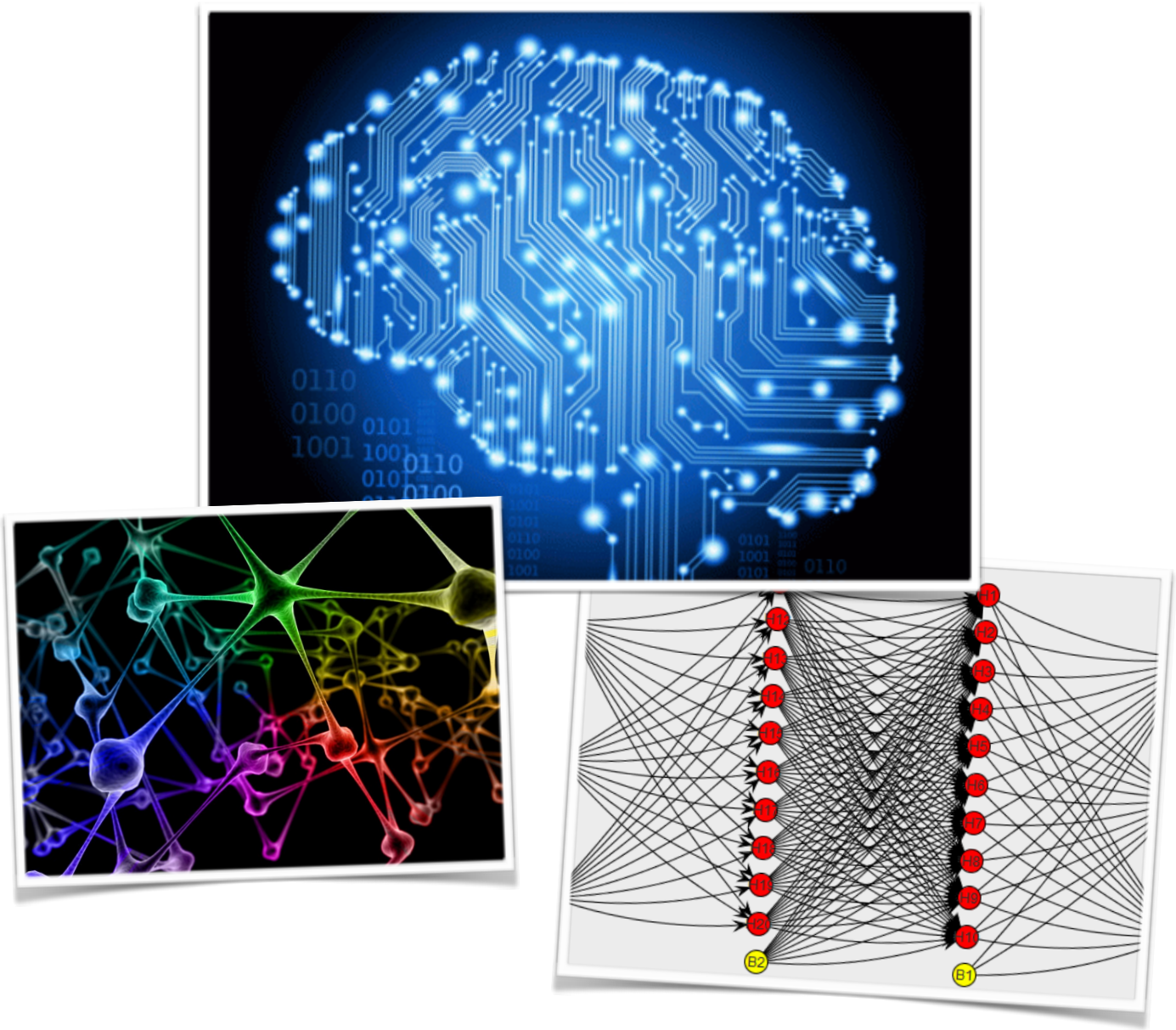
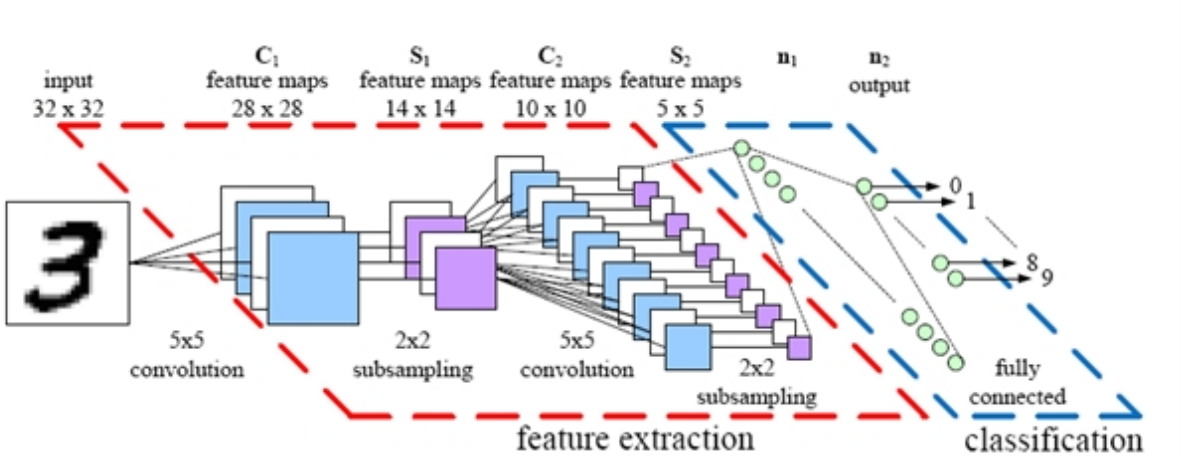


Table of Contents

Introduction

A Convolutional Neural Network (CNN) is comprised of one or more convolutional layers (often with a subsampling step) and then followed by one or more fully connected layers as in a standard multilayer neural network. The architecture of a CNN is designed to take advantage of the 2D structure of an input image (or other 2D input such as a speech signal). This is achieved with local connections and tied weights followed by some form of pooling which results in translation invariant features. Another benefit of CNNs is that they are easier to train and have many fewer parameters than fully connected networks with the same number of hidden units. In this article we will discuss the architecture of a CNN and the back propagation algorithm to compute the gradient with respect to the parameters of the model in order to use gradient based optimization.

In this project CNN was implemented and tested on the MNIST dataset. In first step only one convolution layer was implemented and in second step two convolution layers were implemented. The CNN can be visualized as below:



One 2D Convolution Network:

First, only one 2D convolution network was implemented, in this the output was pooled and softmax was applied before being used as the input for a fully connected Neural network.

Configuration:

Number of filters: 10

Filter size: 5 x 5

Pooling: 4 x 4

Stride: 2

Padding: 0

Neurons in first layer of fully connected NN: 100

Neurons in second layer of fully connected NN: 10

Parameters trained:

Number of parameter in convolution network:
Number of filter in each filter: 5 x 5 = 25
Number of bias in each filter: 1
Number of filters: 10
Total number of parameters in CNN: (25 + 1) x 10 = **260**

Output parameter after max pooling for each later: 3 x 3 = 9
Total number of output parameters after max pooling (input for fully connected layer):
10 x 9 = 90
Number of parameters in first layer of NN: (90+ 1) x 100 = **9100**
Number of parameter in second layer of NN: (100+1) x 10 = **1010**

Total Number of parameters trained: 10370

Note: Lasagne trains different bias for each neuron (not each layer)

Two layer 2D Convolution Network:

For this, another layer of CNN was added. The output of max pooling from first layer is taken as input for the second layer and the output of second CNN is used as input for the fully connected NN.

Configuration:

Layer 1:

Number of filters: 10

Filter size: 5 x 5

Stride: 1

Pooling: 4 x 4

Padding: 0

Layer 2:

Number of filters: 10

Filter size: 5 x 5

Pooling: 2 x 2

Stride: 1

Padding: 0

Neurons in first layer of fully connected NN: 100

Neurons in second layer of fully connected NN: 10

Parameters trained:

Number of parameter in layer 1 of convolution network:
- Number of filter in each filter: 5 x 5 = 25
- Number of bias in each filter: 1
- Number of filters: 10

Total number of parameters in CNN: (25 + 1) x 10 = **260**

Number of parameter in layer 2 of convolution network:
- Number of filter in each filter: 5 x 5 = 25
- Number of bias in each filter: 1
- Number of filters: 10

Total number of parameters in CNN: 10 x 25 x 10 + 10 = **2510**

Output parameter after max pooling for each layer = 1
Total number of output parameters after max pooling (input for fully connected layer):
10 x 1 = 10
Number of parameters in first layer of NN: (10+ 1) x 100 = **1100**
Number of parameter in second layer of NN: (100+1) x 10 = **1010**

Total Number of parameters trained: 4880

Note: Visualization of filter is done in output of code. The value of filters is printed.