Aarhus University

Deep Learning

PROJECT

Author:

Student
(Student number)

Supervisor:

Supervisor

December 13, 2015

TABLE OF CONTENTS

LIST OF FIGURES

1

INTRODUCTION

This report is written throughout the reading course 'Deep Learning' at Aarhus University Department of Engineering. The reading course is based of parts of the teaching resources from the CS231n 'Convolutional Neural Networks for Visual Recognition' course at Stanford University taught by Andrej Karpathy.

The report consists of three parts. The first two parts are the group's solutions for two given exercises in the reading course - Exercise 1 and Exercise 2. The exercises are solved in iPython and are a part of the CS231n course as well.

The third part is a project which is chosen by the group. The project chosen is to use the deep learning framework Caffe to classify images from the image database CIFAR-10.

2

EXERCISE 1

2.1 section name

3

EXERCISE 2

3.1 section name

4

PROJECT

The project for this reading course is chosen to be carried out using the deep learning framework Caffe. The framework is well known in the deep learning community. It is a rather simple way to obtain a classification convolutional neural network in a very short time, which is the reason why it has been chosen.

The idea of this project is to try out some different neural networks with different values for it's hyperparameters and to see which values will obtain the highest accuracy when training and testing on the CIFAR-10 image data set.

The convolutional network used for inspiration is the one introduced in "Alex's CIFAR-10 tutorial, Caffe style"[1] provided by the team behind Caffe. This convolutional network and the changes made to it will be described later on in this chapter along with the architecture and the different types of layers.

4.1 Caffe

Convolutional Architecture for Fast Feature Embedding (Caffe) is a public available framework which makes it possible to design, train and use a convolutional in a rather simple way. The source code is available online and the team behind the framework encourage other people to note or change any errors which they might find. The framework allows you to choose to use either the CPU or the GPU for calculations. In this project the CPU is used.

In order to use Caffe for image classification three files are needed - a .prototxt file, a _solver.prototxt file and a .caffemodel file. These three files together defines the architecture of the network, the hyperparameters and the training and testing strategy.

The .caffemodel file defines the architecture of the model e.g. the number and types of layers. An example of a defined layer is shown in

The .prototxt file defines ...

The _solver.prototxt file defines ...

[1] http://caffe.berkeleyvision.org/gathered/examples/cifar10.html

4.2 Convolutional Network architecture

A convolutional neural network exists of a number of different layers with different properties. The different layers used in the networks in this project are described in this section.

4.2.1 Convolutional layers

The convolutional layers are often the first layers in the network.

The rectified Linear Unit (ReLU) is one of many activation function to choose among. The ReLU activation function is the most common activation function in the lower layers of convolutional neural netorks. Other activation functions could be the Sigmoid or the Tanh activation functions.

The ReLU differs from the other mentioned activations function in several ways. First of all it converges faster which results in shorter training time. Second of all the ReLU activation function do not saturate which avoids the problem about the gradient getting killed which would lead to the case where the network stops learning during the training (HENVIS TIL MATERIALE OM AKTIVERINGS FUNKTIONER). This problem is known from the use of the other activation functions.

The formula for the ReLU activation function is $f(x) = max(0, x)$ and is illustrated in FIGURE..

INDST FIGUR

4.2.2 Pooling layers

4.2.3 Fully-Connected Layers

4.3 Training and testing

De forskellige hyperparametre, regularization, mini ba, epoches,

4.4 The networks

4.4.1 The original network

Beskrivelse af arkitekturen og implementationsfilerne

Beskrivelse af resultaterne

4.4.2 Another network

Beskrivelse af arkitekturen og implementationsfilerne

Beskrivelse af resultaterne

4.5 Results

Sammendrag af alle resultater

Name	Column heading 1	Column heading 2
Cell text	Cell text	Cell text
Cell text	Cell text	Cell text
Cell text	Cell text 576	Cell text

Table 4.1: Long caption

4.6 Discussion

4.7 Conclusion

Here is an inline equation $f(x) = ax + b$.

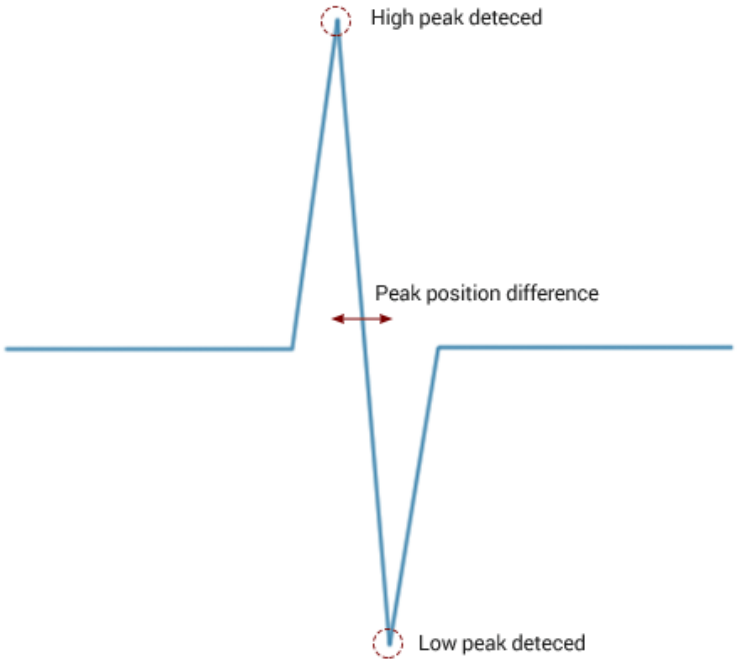

Figure 4.1: Block diagram

$$\frac{n!}{k!(n-k)!} = \binom{n}{k} \tag{4.1}$$

Bibliography