Aarhus University

Deep Learning

# PROJECT

*Author:*

Peter Holst - 20104144
Line Aggerbo - 201302451
Nikolas Bram - 20104278

*Supervisor:*

Supervisor

December 17, 2015

# TABLE OF CONTENTS


**1 Introduction** **1**

**2 Exercise 1** **2**

2.1 Neural Network layers . . . . . 2

**3 Exercise 2** **3**

3.1 Neural Network layer extension . . . . . 3

**4 Project** **4**

4.1 Caffe . . . . . 4

4.2 Convolutional Network architecture . . . . . 5

4.2.1 Convolutional layers . . . . . 5

4.2.2 Pooling layers . . . . . 5

4.2.3 Fully-Connected Layers . . . . . 5

4.3 Training and testing . . . . . 6

4.4 The networks . . . . . 6

4.4.1 The original network . . . . . 6

4.4.2 Another network . . . . . 7

4.5 Results . . . . . 7

4.6 Discussion . . . . . 7

4.7 Conclusion . . . . . 7

# 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 Neural Network layers

The first exercise is two folded. The first part is a walkthrough of implementing a two-layer neural network and using it on the CIFAR-10 dataset. In order for the network to work we implemented the 1st and 2nd order methods for parameter updates such that the weights and bias could be updated.

The second part of the exercise requires implementation of three modules: The forward and backward passes of an affine layer and a Rectified Linear Unit (ReLU) layer.

The solution to exercise 1 can be found in the Appendix.

# 3

# EXERCISE 2

## 3.1 Neural Network layer extension

In the second exercise of the Deep Learning course we extend our Neural network implementation from the first exercise with three modules. The modules implemented are a Dropout module, a convolution module and a pooling layer module.

In order to make the network more robust and agile towards data variations we implemented different types of data augmentation. The overall idea of using data augmentation is to artificially introduce variations in the dataset. We have implemented 4 types of augmentation: horizontal flip, random crops, change in contrast and brightness. Another reason for doing data augmentation is to reduce overfitting and we have tried to do this by combining the data augmentation and dropout.

The solution to exercise 2 can be found in the Appendix.

# 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.

Der skal nok skrives lidt om her ..

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

Pooling layers are common in-between successive convolutional layers. The primary purpose of introducing pooling layers is to reduce the spatial dimensions of a volume thus alleviating computations needed when training the network. This parameter reduction is achieved by performing a downsampling. Different downsampling strategies exists, e.g. max pooling, average pooling and L2 pooling.

more fun to come...

overfitting..

### 4.2.3 Fully-Connected Layers

## 4.3 Training and testing

De forskellige hyperparametre, regularization, mini ba, epoches,

## 4.4 The networks

The network from the tutorial described earlier in this report is reffered to as the original network. The orther networks which are tried out in this project is based on this network with some changes in the architecture. All networks are described in this section.

### 4.4.1 The original network

The original network is a six layer network. The network consists of three convolutional layers and three fully connected layers. The layers are as following

- The first layer consists of a convolution with a kernel size of five, a stride of one and a padding of two. The activation function is the ReLU. It also consists of max pooling with a kernel size of three and a stride of two.
- The second layer consists of the same three parts with the same values of strides and kernel sizes.
- The third layer also consists of three parts. The convolutional part and the ReLU is the same as in the earlier layers. But instead of using max pooling, average pooling is used.
- The fourth layer is a normal fully connected layer.
- The fifth layer is also a normal fully connected layer.
- The sixth layer is a fully coonected softmax layer.

section 4.4

Beskrivelse af arkitekturen og implementationsfilerne

Beskrivelse af resultaterne

AlexNet: (8 layers, 7 hidden layers)
Conv, Max pool — Conv, Max pool — Conv — Conv — Conv, Max pool — Fully con. — Fully con. — Fully con. (output)

### 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

We have experimented with different configurations in the Caffe framework for convolutional neural networks as explained in section 4.4.

- Sammenlign med CIFAR fra exercise 1 Q1, “Train a network” -

## 4.7 Conclusion

We have..

[?] [?]