Aarhus University

Deep Learning

PROJECT

Author:

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

Supervisor:

Supervisor

December 20, 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 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 CIFAR-10

The dataset used to train and test the convolutional network on in this project is chosen to be the CIFAR-10 dataset. This dataset is a labeled subset of the much larger tiny image dataset (which contains 80 million tiny images). The collecting has been done by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.

The CIFAR-10 dataset consists of 60,000 images spread among 10 classes. The classes are the following: airplane, automobile, bird, car, deer, dog, frog,, horse, ship and truck. The images are of size 32×32 pixels and are colour images.

The dataset is split into a training part and a test part. The training part of the dataset is 50,000 images split into five training batches which contains 10,000 random images each. The test part of the dataset contains exactly 1,000 images from each class which results in a total number of 10,000 images.

The dataset can be downloaded from Alex Krizhevsky's home page[2].

4.3 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.

4.3.1 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..

[2]https://www.cs.toronto.edu/ kriz/cifar.html

INDST FIGUR

4.3.2 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 hence alleviating computations needed when training the network. This essentially means that the number of parameters in the network decreases. Thus pooling layers also contribute to control overfitting.

Parameter reduction is achieved by performing downsampling. This is done by applying filters along the spatial dimension of a volume effectively discarding many of the activations. The filter size, or spatial extend F, and stride S are hyperparameters that has to be chosen at time the model is being constructed. As stated in [**?**] most commonly seen pooling layers are either non-overlapping with $F = 2, S = 2$ or overlapping with $F = 3, S = 2$.

Various downsampling strategies exists. The ones that are of interest in this project are max and average pooling. Historically average pooling has been used however recently max pooling has shown to perform better in practice. To understand the intuition of this one might consider this rough analogy: Say a city has four districts and you know whether or not it rains in each district. When you are asked if it rains in the city, you would look at each district and if it rains in any of these, it rains in the city. This process resembles max pooling. You would probably not count the raining cities and divide by 4 as average pooling proposes.

We can transfer this analogy to our ConvNet. If a feature is detected in a small region it is less likely to appear multiple times. Calculating the mean of this region would result in a value that is lower than the actual value. In that sense average pooling is not ideal.

4.3.3 Fully-Connected Layers

Softmax...

4.4 Training and testing

De forskellige hyperparametre, regularization, mini ba, epoches,

4.5 Networks

The network from the tutorial described earlier in this report is reffered to as the initial 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.

4.5.1 The initial network

The initial network is a six layer network. The network consists of three convolutional layers and three fully connected layers. The layers are as follows:

1. 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 ReLU. It also consists of max pooling with a kernel size of three and a stride of two.

2. The second layer consists of the same three parts with the same values of strides and kernel sizes.

3. 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.

4. The fourth layer is a normal fully connected layer.

5. The fifth layer is also a normal fully connected layer.

6. The sixth layer is a fully coonected Softmax layer.

For a detailed description of each type/part of the layers see section 4.3.

The initial network architecture file called: `cifar10__quick_traintest.prototx` can be found in the examples folder under cifar10 in Caffe[**?**]. This file has been used as a starting point for all our networks which are explaind later in this section.

For running the network on a dataset it is also necessary to edit the configuration file: `cifar10_quick_solver.prototxt` which can be found in the same folder as the network architecture file. In this file a number of hyperparameters for the network are specified such as the number of iterations (epochs) and the value of the momentum parameter. The hyberparameters influence on the convolution networks are described in section 4.4.

The only hyperparameter changed in the different networks is the number of iterations (epochs). 1 epoch is specified as when all images in a dataset has been through the entire network once. In Caffe the number of epochs are specified as interations where 4000 iterations is equivalent to 8 epochs. In our networks the number of iterations used is 4000 and/or 8000 (8 and/or 16 epochs). The initial network were trained with both 4000 and 8000 iterations in order to see the performance difference.

Results

Test result with 4000 iterations:

Accuracy: 0.7164 / 0.7227

Test result with 8000 iterations:

Accuracy: 0.7191 /

4.5.2 Customized AlexNet

We have tried to mimic the architecture of AlexNet[**?**] in order to test its performance on the CIFAR-10 dataset. The original AlexNet consists of 8 layers and are using delineation of responsibilities between two GPUs. Since we did not have multiple GPUs avaliable we focused on designing the network for a single GPU as in [**?**].

AlexNet was trained, validated and tested on the ImageNet dataset which contain much larger images than the CIFAR-10 dataset. This meant that we had to change the kernel size, the stride and the pooling in order for our network to be compatible with the CIFAR-10 images.

Our AlexNet configuration for Caffe is a 8 layer network with five convolutional layers and three fully connected layers. The layers are as follows:

1. 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 ReLU and last in the layer max pooling is applied with a kernel size of three and a stride of two.

2. The second layer is equivalent to the first layer

3. The third layer is also equivalent to the first layer but without any pooling in it.

4. The fourth layer is equivalent to the third layer

5. The fifth layer is equivalent to the first layer with max pooling

6. The sixth layer is a normal fully connected layer

7. The seventh layer is also a normal fully connected layer

8. The eight layer is a fully coonected Softmax layer

For a detailed description of each type/part of the layers see section 4.3.

The differences from the original AlexNet is the change in kernel size for both convolution and for max pooling which is scaled down to fit the smaller images in CIFAR-10.

Also the stride is minimized to one in the convolutions but not in the max pooling. Another argument for changing the value to smaller numbers besides fitting the network to the dataset is as explained in [**?**] where Zeiler and Fergus conclude that "the overall depth of the model is important for obtaining good performance". Zeiler and Fergus[**?**] reduce the size of the kernel size from 11 to 7 and the stride from 4 to 2 in the first layer and their reduction in the first layer allows the network to retain more information which improved their overall performance.

[// TODO: Maaske mere her..]

Beskrivelse af arkitekturen og implementationsfilerne

Results

Beskrivelse af resultaterne

Test result with 8000 iterations:

Accuracy: 0.6992

4.5.3 Another network

Beskrivelse af arkitekturen og implementationsfilerne

Beskrivelse af resultaterne

4.6 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

4.7 Discussion

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

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

4.8 Conclusion

We have..

[?] [?]