Homework 3

ConvNet for MNIST Classification

2022403086 Bendik H. Haugen

03 22, 2022

80240743 Deep Learning

Introduction

For this project, I have been tasked with using logistic- and SoftMax-regression on the MNIST data-set. The data-set contains hand-written digits from 0 to 9. When I used logistic regression, I only used a subset containing only 3 and 6. In the second part, I were tasked with classifying the entire data-set.

For this project, I have been tasked with creating a classifyer for the MNIST dataset, by creating a convolution neural network. The task also wants us to implement dropout in the network. The dataset contains digits from 0 to 9.

Convolutional neural network

A concolutional neural network (CNN) is a DL approach that is typically used to process images. The neural network (NN) tries to automate the feature extraction part, by using filters that glides over the network, before we use fully connected layers in the end to classify based on the features extracted by the filters. Between the conv-filters we use pooling layers to reduse the spatial size of the conv layers.

The Filter

The filter, or kernel, works as a pre-processing tool for the fully connected layer. The idea is that the filters should be able to extract different features from the image, while also reducing the images into a form which is easier to process without losing any information. Typically the conv-layers reduce the spatial information of the image, but increases the depth. The filters can be of different sizes, but typically we use 3x3 or 5x5. The filters also have depth, but since we are working with grey-scale images, the depth is 1. Had we used RGB pictures, we typically would have a depth of 3, and the images would have one slice per color. The Kernels shifts over the input-image, shifting sideways with a distance called stride. In this implementation we have been tasked with using stride = 1, but this could in theory be changed. Each element of the kernel consist of a value, or a weight, and that weight is multiplied with the input value at the same position. This is then summed up to produce an output. So at each position, the kernel produces only one output value, independent of the size of the Kernel.

Pooling layers

The pooling layer is another tool that is used to reduce the spatial size of the convolved features. This help with reducing the computational cost of the network. Pooling layers are also good at extracting dominant features of the input image. There are different types of pooling, but in this task we have been asked to implement Max-pooling layers, meaning that

the kernel of size NxN sets the output equal to the highest value of the input image that the kernel is covering.

The Fully connected layer

The fully connected layer is used as the classifier, and works similarly to the MLP used in the previous homework. The output of the convolutional layers needs to be flattened before it can be processed by the FC-layers. This is typicaly done by rolling out the input-image, creating a (nxn, 1) vector from a (nxn) matrix.

Dropout

For the second part of the exercise, we were tasked with implementing dropout in the network. Dropout is a regularization technique. The dropout layers works by ignoring certain neurons during training, to avoid the network becoming to dependent on certain compositions of neurons. If the network develops to many strong co-dependencies among the neurons during training, the network will overfit the training data. The dropout layer will, at random, select neurons to drop during training. This is done by using the dropout rate, which is the probability p that any given neuron should be set to not-active.

Results

During training, I've tried multiple different hyperparameters, but for the implementation set by the homework, I used vary few epochs. This was so that I could try more combinations of hyperparameters, but this does of course affect the result. I ended up with a test accuracy of 78%. This is of course not the best result, but I chose not to spend more time trying to get better results, so that I could spend more time on the dropout network, as it is more complex. See below for the Loss and Accuracy per epoch. The network was clearly still learning, as both loss and accuracy was decreasing for each epoch.

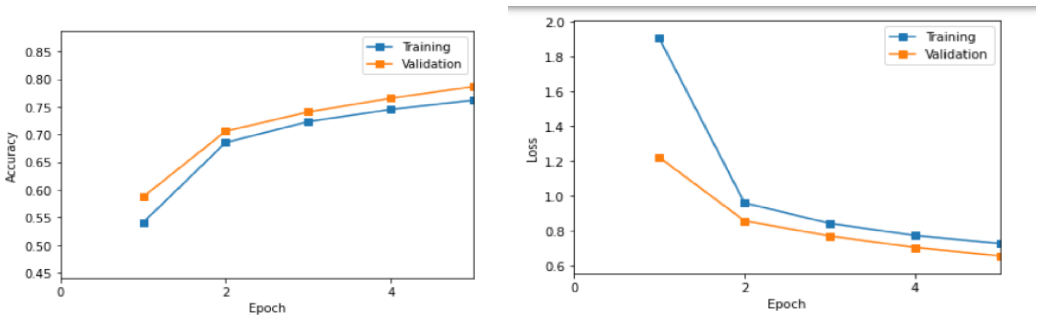

Figure 1: Accuracy and loss of the final round of the basic CNN implementation

Dropout-implementation

Whereas I changed I couple of hyperparameters, increasing both learning rate and epochs, for the first run I only added a singe dropout-layer in the classification part of the network. What I noticed, is that both loss and accuracy was better on the validation data compared to the training data, but the test-data gave the best results. So I figured that since the network clearly isn't overfitting, I could add some complexity to the network. Hence I added another fully connected layer with 32 nodes, ReLu-activation and dropout. This should allow the network to find more complex patterns.

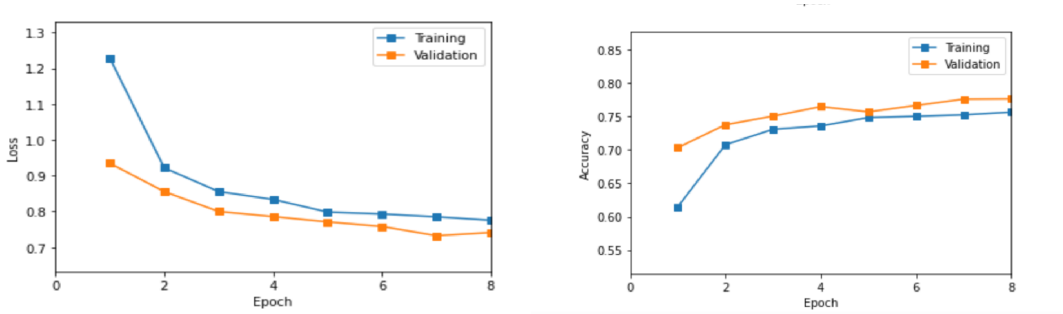

Figure 2: The loss and accuracy of the first network with dropout implemented

After adding another hidden layer in the network, I noticed some changes. The first, being that the network learns a lot slower. This is probably due to the increase in the number of weights, meaning that more parameters has to be adjusted. The second being that again the network seems to underfit the data, as it performs better on validation than on training dataset. For future implementations, it would be an idea to add more complexity to the model, as there is still improvements to be made w.r.t both Loss and Accuracy.

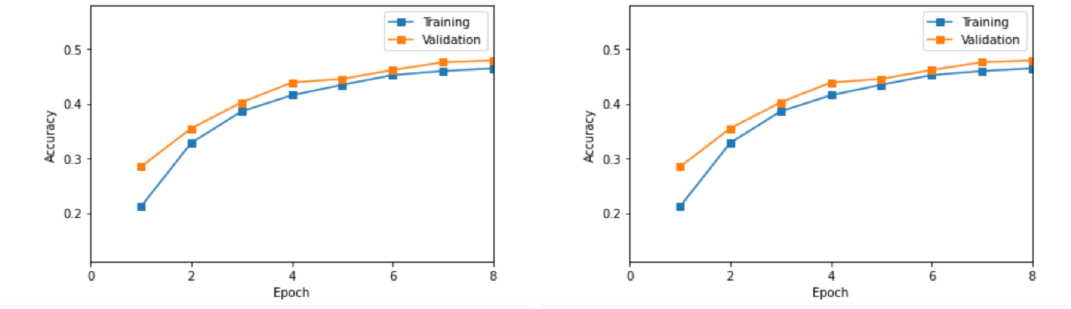

Figure 3: The loss and accuracy of the first network with dropout implemented, and with another hidden layer

The model could probably gain a higher accuracy and lower loss if I had used more filters in the Conv layers, but time would not allow me to try.