CONVOLUTIONAL NEURAL NETWORKS AND FASHION MNIST

SAMIP KARKI

Applied Mathematics, University of Washington, Seattle, WA
karki1@uw.edu

Abstract. This work is an introduction to convolutional neural networks (CNNs). I use the benchmark task of Fashion-MNIST classification to compare and contrast CNNs to fully connected networks (FCNs). I investigate how network size, optimizers, and the addition of convolution layers impact metrics like testing accuracy and runtime.

1. Introduction and Overview

In a previous work [1], I explored fully connected networks (FCNs) which take a flattened vector as an input. In this work, I explore convolutional neural networks(CNNs), which instead can take 2D or 3D tensors as inputs and extract general patterns and relationshps in the data. Because CNNs are specifically designed for processing grid-like data, they are particularly effective in tasks like image recognition.

2. Theoretical Background

The core building block of CNNs are the convolutional layers. These apply learnable filters (also called kernels) onto input data that can be 2D or 3D tensors. Each filter slides across the input data, computing the dot product between the filter weights and the corresponding input values. This operation captures local patterns or features in the data.

CNNs also make use of pooling layers. These reduce the dimension of the convolution layers while retaining important information. Common pooling operations include max pooling or average pooling, which take the maximum or average value within a given pooling region.

Date: August 30, 2024.

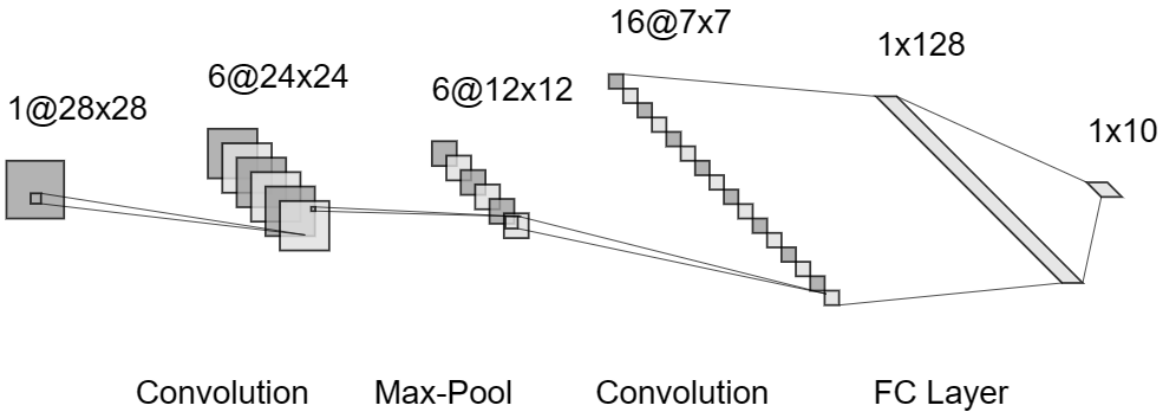

Figure 1. Diagram of the 100K convolution network. Created with [2]

FCN Network	Optimizer	Testing Accuracy [%]	Training Time (25 Epochs) [s]
100K	AdaM	86.85 ± 2.31	408
50K	AdaM	86.45 ± 1.78	431
20K	AdaM	87.21 ± 2.03	399
10K	AdaM	86.06 ± 3.21	410
10K	SGD	85.05 ± 1.92	393

FIGURE 2. Table for FCNs performance summary.

CNN Network	Optimizer	Testing Accuracy [%]	Training Time (25 Epochs) [s]
100K	AdaM	88.96 ± 1.84	609
50K	AdaM	88.70 ± 2.74	618
20K	AdaM	89.41 ± 2.21	617
10K	AdaM	88.85 ± 1.75	612
10K	SGD	87.30 ± 1.80	611

FIGURE 3. Table for CNNs performance summary.

Fully connected layers are used to connect the convolutional layers to the output, which is typically a vector. Similarly to FCN, activation function like relu and optimization algorithms like gradient descent are also used. In CNNs, the number of weights in the convolution layers are neglibible compared to the number of weights in fully connected layers, which makes CNNs capable of performing well with a relatively small number of saved parameters. However, CNNs have a higher order of computational complexity than FCNs, so a trade off comes between having a lower number of parameters and the runtime.

3. Algorithm Implementation and Development

This project makes extensive use of `torch` and related packages. The Fashion MNIST dataset was taken from the torchvision package. Neural network model layers, relu activation functions, cross entropy loss, optimizers, was done with `torch` commands. The splitting of the training and testing data was done with `sklearn`.

4. Computational Results

In this work I compare FCN and CNN of various sizes on the Fashion-MNIST classification task. In a previous work [1], I saw how FCN preformed at the Fashion-MNIST task while exploring the effect of hyper-parameter tuning and different shapes of networks. I found suitable results with the batch normalization on the 2nd hidden layer and using the AdaM optimizer, and therefore will continue using these for most of the networks in this work.

The Fashion MNIST dataset of 28×28 pixel images has $60,000$ images in the training set and $10,000$ images. The images has 10 different classes corresponding to different articles of clothes. Furthermore, the images in the training dataset are split into batches of size 512, where 51 of these were used for validation. The test batches are size 256.

An explanation of the different networks are given here:

- **FCNs**: All the fully connected networks have similar structure, the only difference being the number of nodes in a single hidden layer. The input layer has 784 nodes, corresponding to the vector of the 28×28 pixel image. This is followed by the hidden layer with a variable number of nodes and the output with 10 nodes. The 100K network had a hidden layer with 128 nodes, which resulted in a network with $101,632$ total adjustable weights. The 50K, 20K, and 10K networks had hidden layers of size $64, 32$, and 16 respectively.

- **CNNs**: An illustration of the 100K CNN is given in figure(1). The CNNs take the 28×28 pixel image as input. Using 6 filters with length 5, the first convolution layer has 6 channels of 24×24. Layer 2 is a pool layer made with max-pooling filter of length 2 and stride 2, resulting in layer 2 having 6 channels of 12×12. Using 16 filters of size 6, layer 3 is a convolution layer with 16 channels of 7. If the values in layer 3 were flattened, there would be $16 * 7 * 7 = 784$, the same number as the input layer in the FCNs. This convolution layer is connected to a hidden layer with variable number of nodes and then an output layer with 10 nodes. Like in the FCN, the number of total weights is controlled by the number of nodes in the hidden layer (the total number of weights in the convolution and pool layers is 726, which is small compared to the number of number of weights in the fully connected layers). The 100K, 50K, 20K, and 10K have hidden layers of size $128, 62, 32$, and 16 respectively.

The networks were each trained using 25 epochs of the training data and then their accuracy on classifying the testing data was evaluated in batches. The results of FCN networks are given in figure(2) and the CNN network results are given in figure(3). The error on the testing accuracy is given by the standard deviation of the accuracy on all the testing batches.

Interestingly, reducing the size of the hidden layer did not have much effect on the either FCNs or CCNs, as both types of networks performed similarly when the hidden layer was reduced in size. The CNNs in general performed $1 - 2\%$ better than their FCN counterparts, but at the cost of taking longer to train. At first, I was expecting much greater performance from the CNNs, but then I realized $1 - 2\%$ increase in accuracy corresponds to $7 - 15\%$ fewer incorrect classification, which seems more significant. Another interesting point is that reducing that reducing the network sizes did not seem to have any effect on the training time for both FCNs and CNNs. This is especially confusing to me because for FCN, a smaller number of weights should mean faster training (using Google Collab to run all my code). The only explanation I can think is that for both CNN and FCN, the updating of the weights was actually a negligible part in the runtime and instead most of the computation time was taken by the forward pass of the data (although even in this explanation, the smaller FCNs should train faster than larger FCNs). Maybe, if both types of networks had even larger number of weights, up to millions or tens of millions, we would see start to see what we expect, that larger FCNs take longer to train.

I was somewhat confused about the similarity in performance of all the networks, and was wondering if choosing AdaM optimizer made the results of the FCN too strong to see meaningful trends. I did some additional tests with stochastic gradient decent as my optimizer instead on the 10K networks, which can be seen from the last row on figure(2) and figure(3). Like with AdaM, the CNN 10K network was able to perform a couple of percents better than its FCN counter part.

5. Summary and Conclusions

I explored how CNNs hold up to their FCN counter parts in an image classification task which should favor CNNs. Although relationship between network size, accuracy, and runtime still is unclear, there was a clear increase in accuracy from the CNNs from their FCN counterparts which was expected.

References

[1] S. Karki. Amath 482/582: Homework 4: Deep neural networks and fashion-mnist.
[2] A. Lenail. Nn-svg. `https://alexlenail.me/NN-SVG/LeNet.html`.