Deep Learning Project 1

Fabian Kropfhamer

December 18, 2022

1 Introduction

The goal of this project is to compare different architectures of neural networks. The different Architectures are evaluated on the same task. This task is given two images of one-digit numbers to predict which of the two numbers has a higher value. The Images thereby are 14 by 14 pixels. Therefore the input for the neural networks is a 2x14x14 tensor. The output and also the label for the images is 1 if the first digit is higher than the second digit and 0 otherwise. The training dataset and also the test dataset consists of 1000 samples each. In the following, the different architectures are introduced and then compared.

2 Architectures

2.1 Feed Forward

The Feed Forward neural network is the simplest of the investigated architectures. It consists of 3 fully connected layers. Between the 2 two hidden layers are two activation functions. These are activation functions are ReLU functions. Also before the input layer, there is a flatten function, which transforms the 2x14x14 input tensor into a 2x196 tensor. As the loss function binary cross entropy loss is used. The neural network is optimized with the Adam optimization algorithm.

2.2 Convolution

The second architecture is a convolutional neural network. This network includes 2 convolution 2d layers with ReLU as their activation function. Also, between the 2 convolution layers, a max pooling layer is used. These convolution layers are followed by three fully connected layers, which also use the ReLU activation function. For the training of the network, Adam is used as the optimizer and binary cross entropy for the loss function.

2.3 Shared Weights

This architecture uses convolutional layers, too. But in this neural network, the two images are not fed into the convolution layer as one tensor. The 2x14x14 tensor first split up into the two images, leaving two 14x14 tensors. These separate tensors are then fed one after another into the convolution layers of the network. These convolution layers are very similar to the ones used in the convolution architecture. They hereby consist also of 2 convolution 2d layers again with max pooling and the ReLU activation function. Followed by 2 fully connected layers also with RelU as the activation function. This first part forms the shared part of the neural network. That means the weights are shared for the two images. After this shared part, the output for the two images is concatenated and then fed into a last fully connected layer. This fully connected layer takes the two outputs and produces the prediction and output of the network.

2.4 Auxiliary Loss

In this approach, additional intermediary results are used. The network predicts first which digit a picture shows. This prediction is then used to calculate an additional loss, which is then used to train the model. Furthermore, the prediction then feeds into fully the connected layer to form the final prediction. As in the shared weights architecture, the input tensor is split up into the two images. These two images are then forwarded to the convolutional layers. Hereby the same convolutional layers and activation functions are used as for the shared weights network. The only difference is that the weights are separated and not shared. Another difference is in the training of the network. Because there are three predictions in total, one being the binary classification as in the other architectures. The others are the

prediction of the digits in the image as multiclass classification. Therefore two different loss functions have to be used. For the binary classification, the binary cross entropy is used and for the digit classification, cross entropy loss is used. These losses are then used to optimize the weights of the model. Stochastic gradient descent is used as the optimization algorithm.

2.5 Pretraining

The last architecture is very similar to the shared weight architecture. The layers of the model are herby the same and the only difference is the training. It combines basically the auxiliary loss architecture with the shared weight architecture. Thereby, the weights are shared for both images and the prediction of the class of the image is used in training. This is accomplished by splitting the training into two steps. The first step is to train the first part of the network including all the convolution layers to predict the digits in the images. The second step is then to take these trained weights and use them to predict the original task. In the neural network, these pre-trained weights are again shared for the two images. Also, cross entropy loss is used for the first training step, and binary cross entropy loss for the second. Adam is again used as the optimization algorithm in both steps.

3 Results

These are the results from 30 runs for each architecture. The models are trained for 20 epochs. The table shows the best result from each run and epoch.

Architecture	Max accuracy	Min accuracy	Standard deviation
Feed Forward	0.8270	0.7910	0.0089
Convolution	0.8130	0.8130	0.0085
Shared Weights	0.8780	0.8380	0.0118
Auxiliary Loss	0.7090	0.4450	0.0703
Pretraining	0.8770	0.8410	0.0098

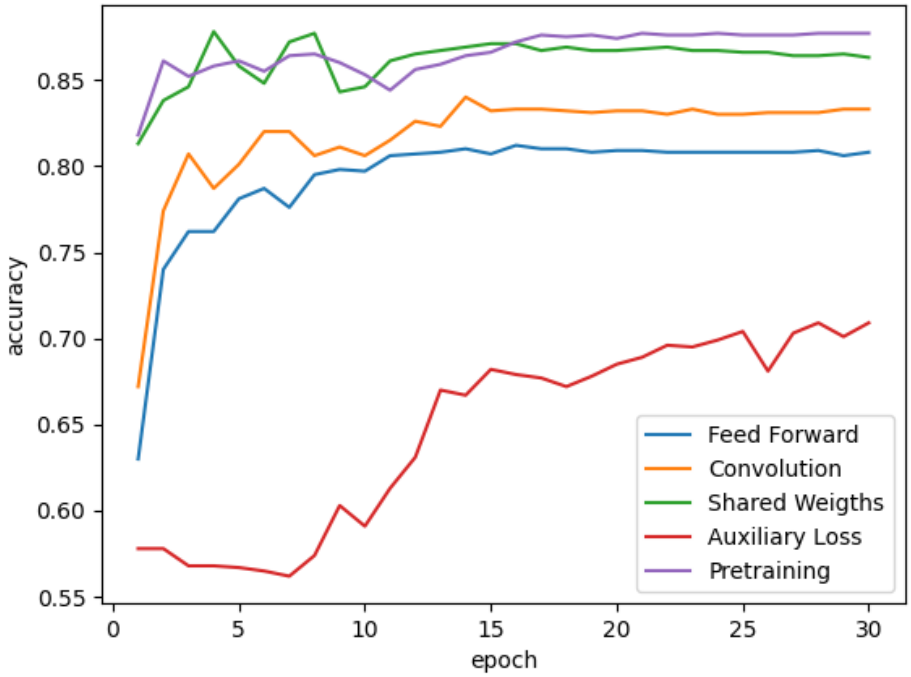

Figure 1: Plot of accuracies

4 Discussion

The results also show that the auxiliary model performs the worst. It also has the highest difference in performance between the runs.

The feed forward performs reasonably well despite it being the simplest model of the presented. This could have to do with the fact that it has a lot of parameters. Although it achieves low loss the fastest its accuracy is not the best.

All models get to a low less within 15 epochs, the auxiliary architecture being the exception. Also, for all but the auxiliary model, the accuracy does not improve significantly after 15 epochs.

The results indicate that the pretraining architecture has the highest accuracy. This could be because it combines the shared weights and the auxiliary loss.

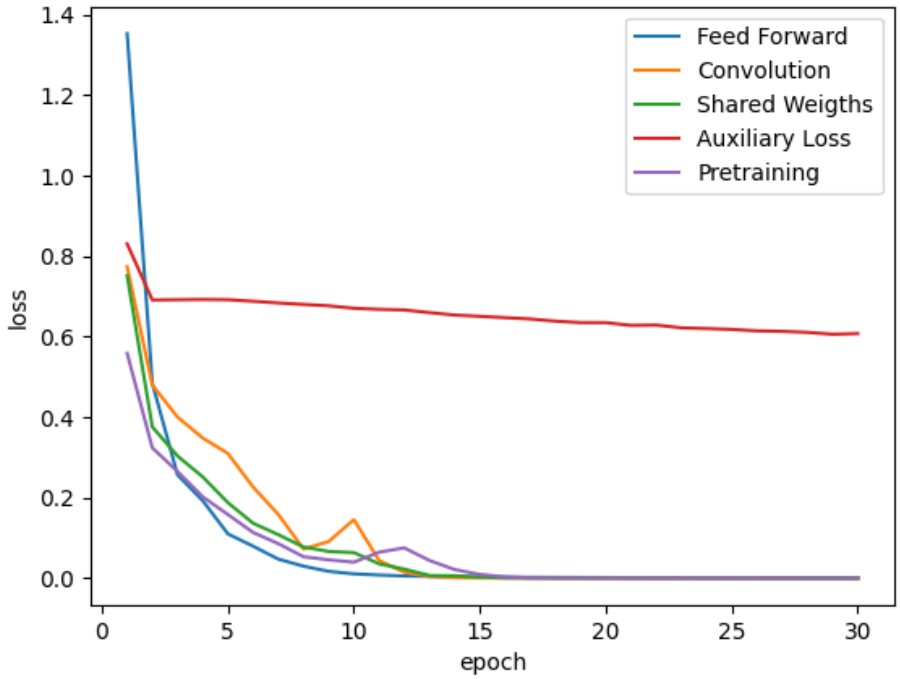

Figure 2: Plot of losses

In conclusion, different architectures can perform differently well on the given task. The project shows that even simple architectures can outperform more advanced architectures. Also across the project sharing weights performed the best for this specific data, with pretraining and without.