https://colab.research.google.com/drive/1W2Lmk2PKEc1MluFSca5qJk6OzARfSSJf?usp=sharing

Homework 4 - Group 92

Feed-Forward Networks

The MNIST dataset contains 70000 instances, from which 10000 belong to the test split and 60000 belong to the train split. Each instance contains a feature in the Image format and a label in the int format. Just by taking this into account, we know that a Convolutional Neural Network is the best suit for this classification problem, since image processing is the main purpose of this kind of network. However, this doesn't necessarily mean that a Feed-Forward Network will perform poorly.

The main aspects that need to be taken into account when defining the ideal Feed-Forward Network is the number of **Neurons per Hidden Layer**, the **number of Hidden Layers** and the **Regularization method.** Other parameters don't really need any testing, since they are obviously the best choice. We're studying a classification problem, so we select **Sparse Categorical Cross-Entropy** for the Loss function, as well as the **Softmax** activation function. Also, we'll be using **Stochastic Gradient Descent** for optimization.

Let's start by analysing the number of neurons of each hidden layer. The thumb rule states that this value should be somewhere **between the input and output size**. In our case, there will be a Flatten Layer before the hidden layers, which means that the input for the first hidden layer will have a size of 784 and the final output a size of 10. With these limits, we can **fix one hidden layer for testing purposes and vary the number of neurons** to verify which one brings the best results.

We verified that testing values with a distance smaller than 100 wouldn't be necessary. As you can see from Figure 1 and 2, the model gives the best results when the **number of neurons is somewhere in the middle of the range (n = 400)**, which makes sense since it's the tested value closest to the average of both limits.

The next step is finding the ideal number of Hidden Layers. Finding the ideal value isn't easy. More hidden layers mean a more complex model with a **tendency to overfit**. Also, keep in mind that in a real world scenario having an exponentially large number of training parameters isn't ideal because **training becomes incredibly slow**. With this in mind, we followed a testing methodology similar to the previous one. We decided to **range the number of hidden layers from 1 to 6, all with 400 neurons as seen in the previous step**, given that more layers than that would increase too much the complexity of the model. If you take a look at Figure 3 and 4, you can easily verify that **the best results are obtained with 4 hidden layers**.

The final step is to find the best regularization method in order to avoid overfitting. In this final stage there are two parameters that we need to find: the Regularization Methodology (L1 regularization or L2 regularization) and the regularization factor. A **factor of 0 would have the same effect as not having any regularization at all**: We're back to the original loss function. Also, **a factor of 1 would really penalize the weights** and they become close to 0. Due to these reasons, we try to find a factor between 0 and 1 using the same testing methodology as in the previous steps. Using a network with 4 hidden layers and 400 neurons in each layer, **we start by testing larger intervals with a length of 0.1**. Figure 5 shows that **as soon as the regularization factor starts to increase, the model's accuracy drops drastically in both methods**. For that reason, we decided to **test smaller intervals with a length of 0.01 between 0 and 0.1 in L2**. Figure 6 shows that even with these smaller values, the model's performance is still lackluster. However, L2 performs slightly better than L1. We can conclude that regularization doesn't help improving our model.

With all these analysis completed, we can conclude that the Feed-Forward Network that achieves the best result has the following properties: **4 hidden layers**, **400 neurons** in each layer and **no regularization** applied. This model provides us with an accuracy of about 97.305% and a loss of about 0.092. Let's now examine the behaviour of Convolutional networks and compare it with the previous model.

Convolutional Neural Networks

Just like before, when defining the Convolutional Neural Network, it is important to find the best number of **neurons per hidden layer**, the **number of hidden layers** and the **regularization method**. However, convolutional networks also have other important factors such as the **number of filters**, the **kernel size** and the **pool size**. Since we are dealing with 28x28 images, we decided to go with **16 filters**, a **kernel size of 4** and a **pool size of 2**. We still use the **Sparse Categorical Cross-Entropy** for the loss function and the **Softmax activation function** for the output layer. For the hidden layers we use the **Rectified Linear Unit** activation function (ReLU).

We began by analysing the number of neurons per hidden layer in a network with a single hidden layer, just like we had done for the feed-forward network. Since neither the goal nor the input size had changed, we tested the same values as for the previous model. However, this time, we found **the ideal number of neurons to be 700** (Figures 7 and 8), yielding an accuracy of approximately 98.28%.

Then we explored the possibilities regarding the number of hidden layers, each with 700 neurons. We tested, once more, between 1 and 6 hidden layers, finding **6 to be the best fit** (Figures 9 and 10), with a slightly increased accuracy of 98.4%.

For the regularization analysis, we first tried L1 and L2 regularization and then explored **dropout**. We decided to explore L1 and L2 regularization again due to the fact that neither method had positive results previously. We began with intervals of length 0.1 and then tried finer intervals, of length 0.01. Comparing both types of regularization, although **L2 seems to yield better results overall**, the fact is that, again, **none of the two seemed to have significant impact in the accuracy of the model** (Figure 11). In fact, we concluded that the regularization factor with better outcomes is 0 for both types of regularization.

Regarding dropout, as we can see from Figure 12, we can see that **the best results are achieved with a dropout factor of 0.1**. When testing, we used a smaller patience factor of 5 for the early stop and a training set of only 5000 instances due to the long execution time.

Conclusions

With the full analysis complete, we conclude that the best results for this dataset are obtained using Convolutional Neural Networks, as predicted and explained in the previous sections. We also denoted some interesting remarks during testing. We verified that regularization and dropout methods should not be applied in every hidden layer because it will produce too much generalization. Regarding the number of neurons and hidden layers, our predictions were correct. Increasing this value too much will produce a model that is too complex and lead to overfitting.

Appendix

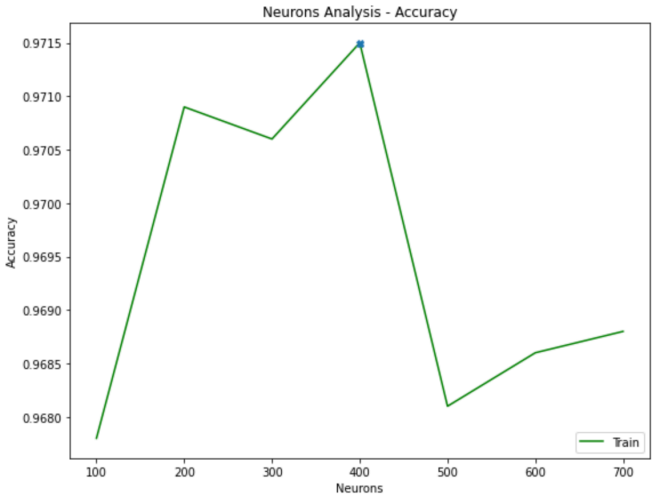

Fig 1: Model Accuracy with Varying Neurons on a Fixed Hidden Layer.

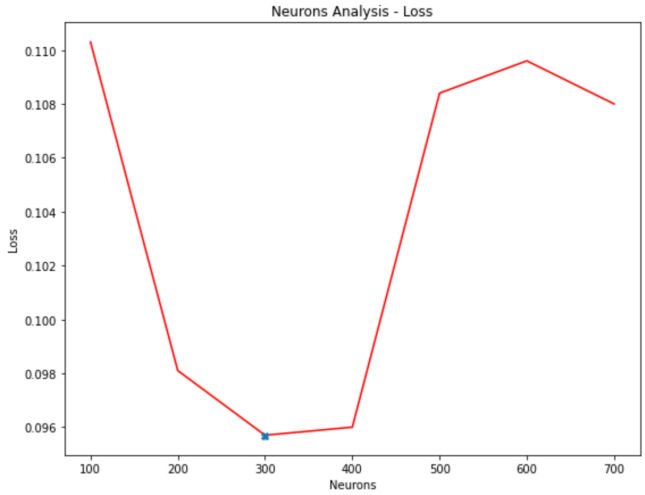

Fig 2: Model Loss with Varying Neurons on a Fixed Hidden Layer.

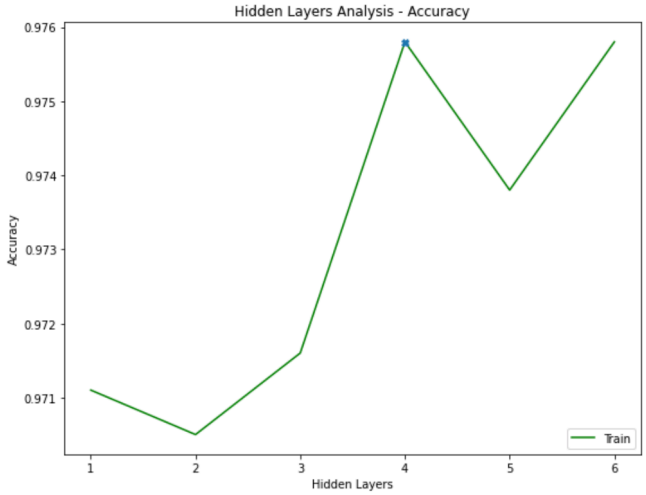

Fig 3: Model Accuracy with Varying Number of Hidden Layers

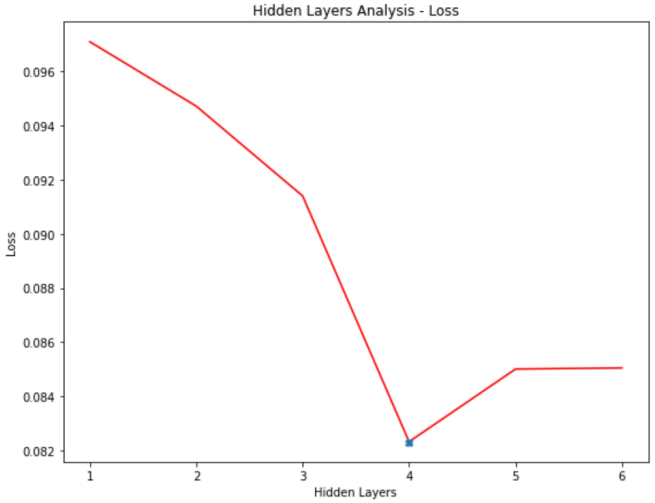

Fig 4: Model Loss with Varying Number of Hidden Layers

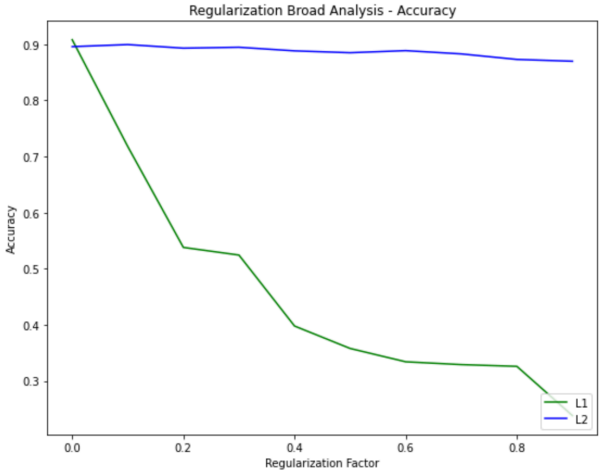

Fig 5: Model Accuracy with Varying Regularization Factor in Large Intervals.

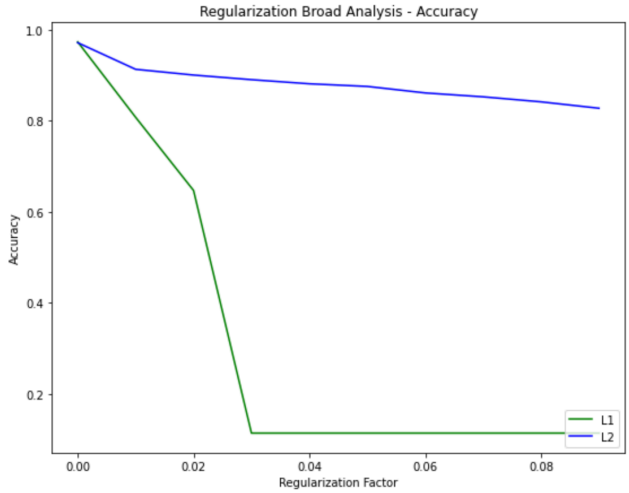

Fig 6: Model Accuracy with Varying Regularization Factor in Narrow Intervals.

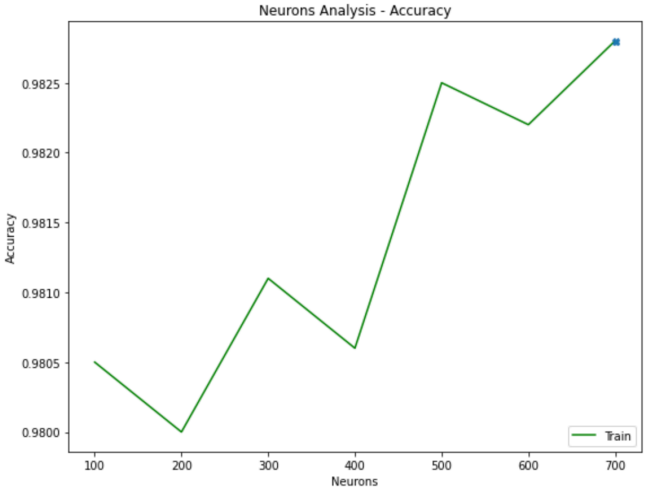

Fig 7: Model Accuracy with Varying Neurons on a Fixed Hidden Layer.

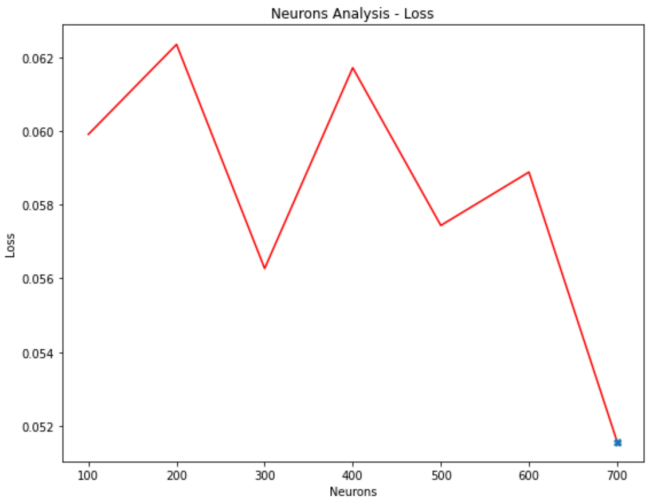

Fig 8: Model Loss with Varying Neurons on a Fixed Hidden Layer.

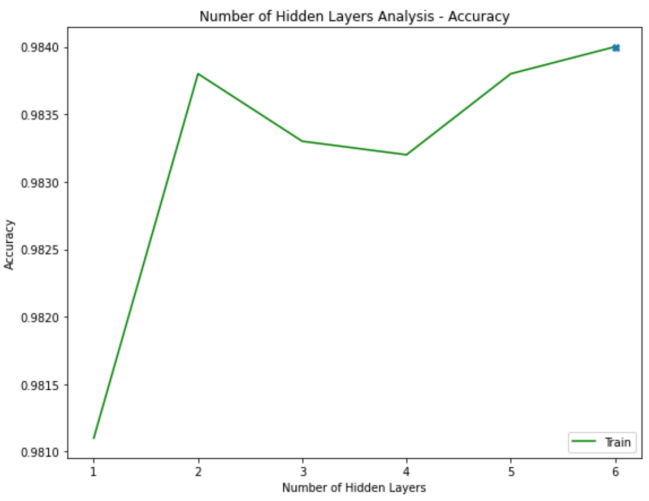

Fig 9: Model Accuracy with Varying Number of Hidden Layers

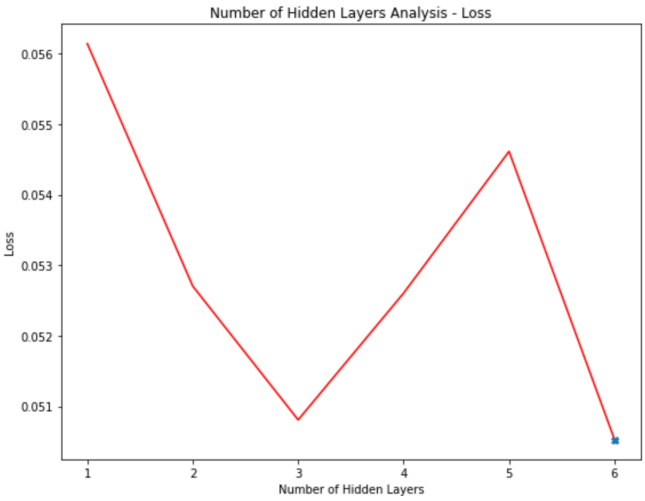

Fig 10: Model Loss with Varying Number of Hidden

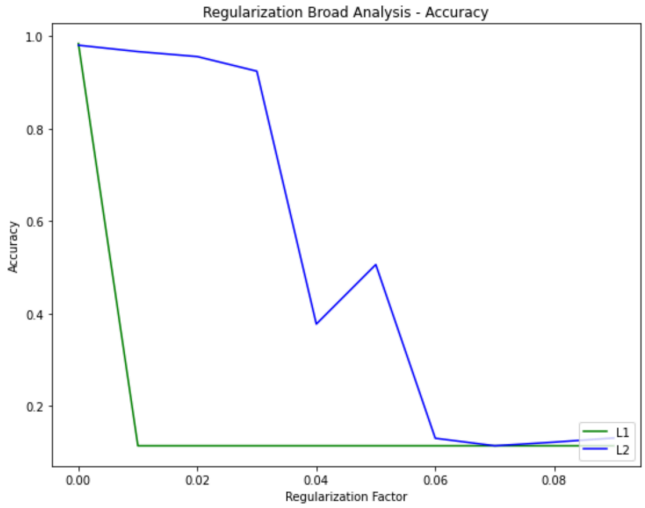

Fig 11: Model Accuracy with Varying Regularization

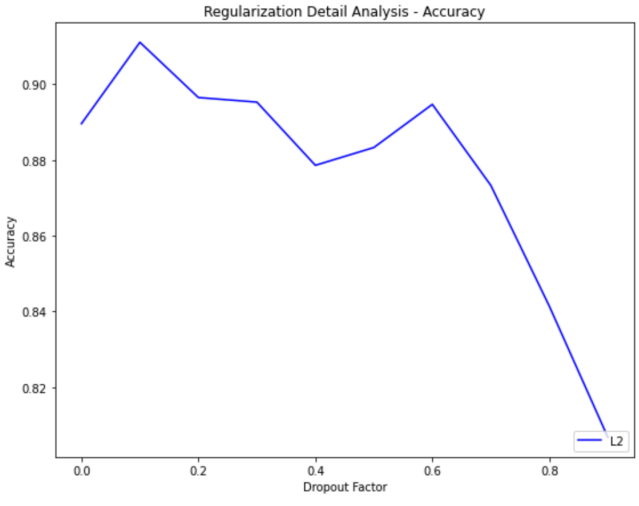

Fig 12: Model Accuracy with Varying Dropout Factor.