1.2 Architecture

1 A Single Perceptron

A fundamental component of neural network is single perceptron, which can classify objects based on its input data. Its mode of operation is based on taking input values and multiplying them by their respective weights, which are then summing them up to produce an output value. When the output is input into the activation function, the final value is produced, on which the decision is based.

A single perceptron has one significant limitation, namely, it can only learn linearly separable problems. So, while it can learn the AND and OR functions, the XOR operation is too complex for it, as its decision boundary is not linear. Below, we can see the results of computing those operations by our single perceptron model.

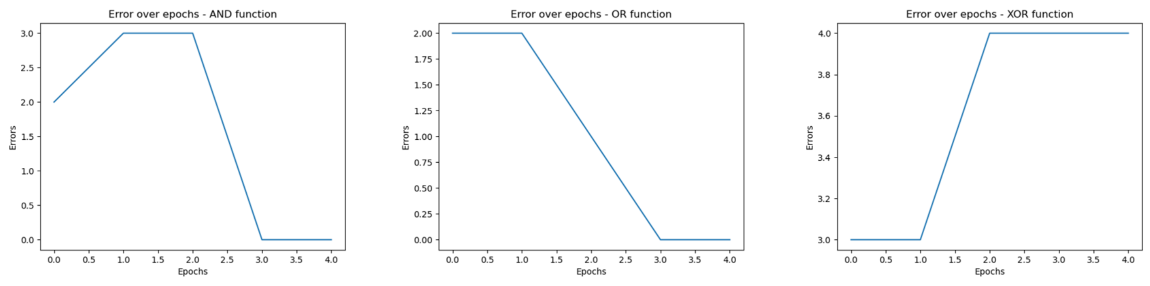


Figure 1: AND, OR, XOR functions processed by single perceptron model

2 Number of Input Neurons

The aim of this assignment is to classify objects into specific grocery products based on ten features. In order to do so, we decided that our input layer will have a size of ten, so that every node corresponds to one feature.

3 Number of Output Neurons

As we have seven destination classes, our ANN model has seven nodes in the output layer, each one corresponding to one class.

4 Hidden Layer

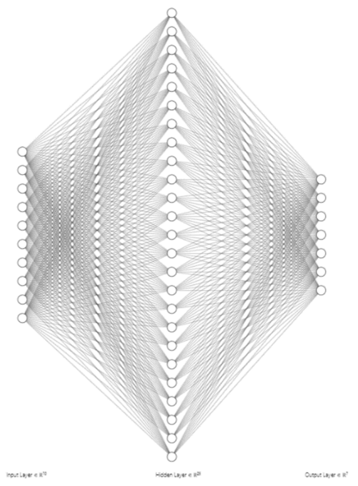
Our initial guess assumes one hidden layer with 25 neurons. In our opinion, two or more layers will lead to overfitting and slowing down our model. However, we will have to evaluate whether our guess was correct in further computations.

5 Activation Functions

After prior reviewing the format of input data, we decided to use the sigmoid function as the activation function of the hidden layer. It both takes into account the possibility of negative values occurring in the data as well as provides interpretable output.

For the output layer, we settled for the softmax function, as it provides the best result for multiclass classification with mutually exclusive labels.

6 Schematic Diagram of Network


Input Layer $\in \mathbb{R}^{10}$
Hidden Layer $\in \mathbb{R}^{25}$
Output Layer $\in \mathbb{R}^{7}$

1.3 Training

7 Division of Data

To ensure an unbiased estimate of performance, it is important to randomly shuffle the data before splitting it into training, validation, and test sets. This ensures that the data is representative of the entire dataset and that each set has a similar distribution of samples. Additionally, it is important to ensure that the splits are IID (independent and identically distributed), meaning that each set has samples that are representative of the entire dataset.

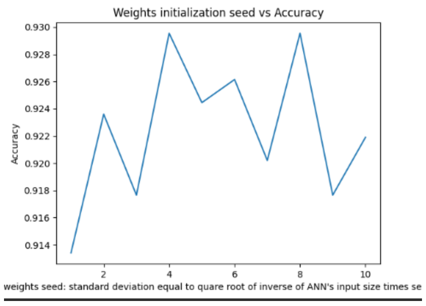
8 Evaluation of performance

The performance of our network is evaluated using the accuracy metric, which measures the number of correct classifications made by the network divided by the total number of classifications. This value is expressed as a percentage and provides an indication of how well the data is recognized by the network. The highest accuracy we achieved was done using 1 hidden layer with 29 neurons, 35 epochs, 20 mini-batches and a 0.1 learning rate.

9 End of Training

The decision to end the training process was made based on the performance of the network on the validation set. To avoid overfitting, we monitored the validation loss and stopped training when it stopped improving or started to increase. Continuing training beyond this point could lead to overfitting, where the model becomes too specialized to the training data and fails to generalize well to new data. Therefore, it is generally not advised to continue training for too many epochs.

10 Impact of Weights Initialization

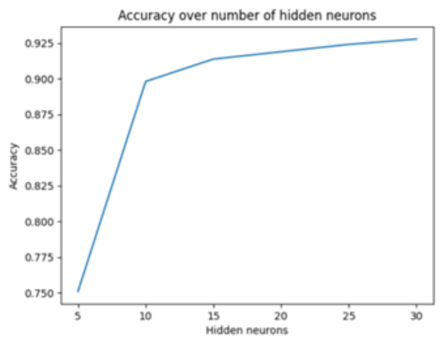


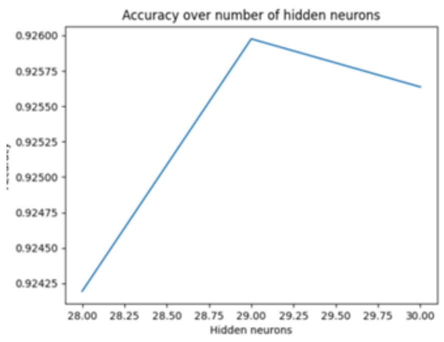
Above you can see the depiction of the relationship between weight initialization and the accuracy of the model. As you can see the choice of the correct initialization is crucial for the proper performance of the model.

1.4 Optimization

11 Different Amounts of Hidden Neurons

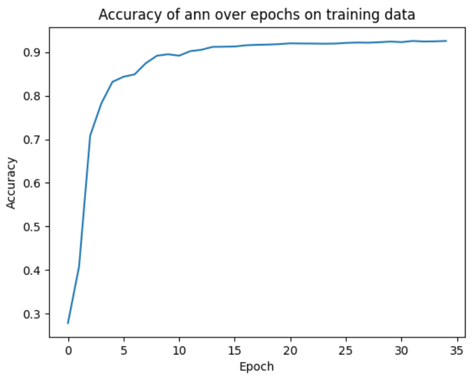
We performed cross-validation, averaging 10 times for each number of neurons, to ensure the reliability of the results (cross-validation was conducted on the validation set). We conducted two sets of experiments. Firstly, we tested 5, 10, 15, 25, and 30 hidden neurons to determine the range where the highest accuracy occurred. Based on the initial experiment, we concluded that the optimal number of hidden neurons was between 25 and 30. Subsequently, we conducted another cross-validation experiment within the range of [25, 30]. As a result, we determined that the most optimal number of hidden neurons for our network was 29 in a single hidden layer:

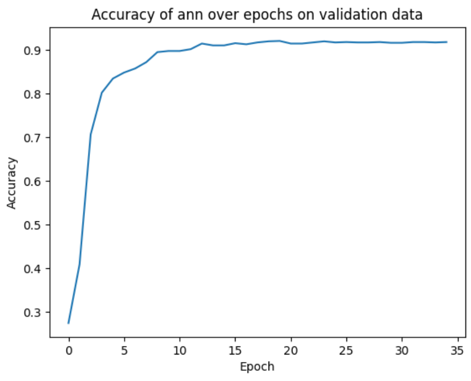




12 Best Architecture

We picked the architecture with 29 hidden neurons spread across 1 hidden layer. We set the number of epochs to 35 and trained the ANN. The figures below display the changes of accuracy over epochs during training process over training data and validation data. We can see great improvements in the first 10 epochs for both data sets, then the gradient becomes small but still positive. It seems like the accuracy can be portrayed as logarithmic function of number of epochs.





1.5 Evaluation

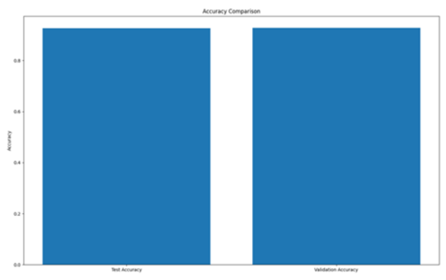
13 Success Rate

The difference between the success rate measured by accuracy on the test set and the validation set lies in the purpose of the usage of these two sets in the training and evaluation process of a machine learning model.

The validation set is used during the training process to find optimal hyperparameters for our model and prevent overfitting. We use a validation set to evaluate the performance of the model on data that is not used for training, and the hyperparameters are adjusted based on the validation set performance. In general, the purpose of the validation set is to provide some kind of estimation of how well the model generalizes to unseen data. In our case, we used this set in our cross-validation method to find the most optimal number of neurons in the hidden layer.

On the other hand, the test set is used to evaluate the final performance of the model after the hyperparameters have been decided using the validation set. We completely separate the test set from the training and validation sets, and we use it only once to evaluate the performance of the model on unseen data. The goal of the test set is to provide an unbiased estimate of the performance of the model on new, unseen data.

We used the Accuracy metric to evaluate the performance of our classification model. As expected, the accuracy on the Validation set was a bit higher than the one obtained on the Test set. However, in both cases the accuracy was satisfactory.



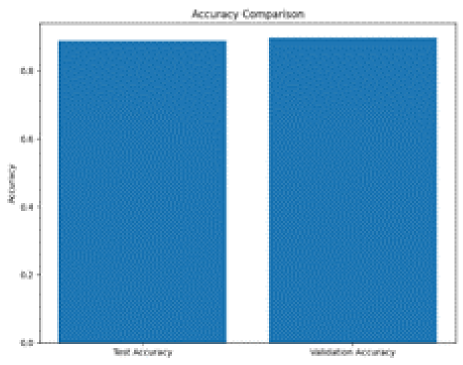


Figure 2: 35 epochs - Test Set Accuracy = 0.9261, Validation Set Accuracy = 0.9280
5 epochs - Test Set Accuracy = 0.8879, Validation Set Accuracy = 0.8961

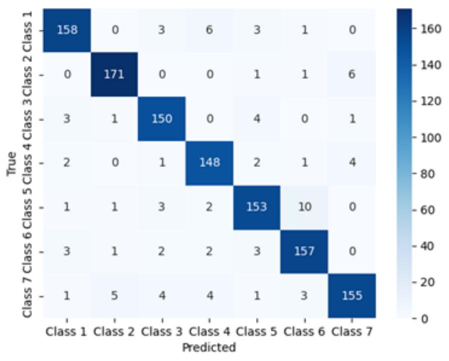
14 Confusion Matrix

A confusion matrix is a useful tool for evaluating the performance of a multiclass classification model, such as a multilayer perceptron (MLP). It shows the number of correctly and incorrectly classified instances for each class, allowing us to identify where the model is making the most mistakes.

The confusion matrix is a 7x7 matrix, where the rows correspond to the true classes and the columns correspond to the predicted classes. Each element in the matrix represents the number of instances that belong to the true class and was predicted to belong to the predicted class. To interpret the confusion matrix, we can look at the diagonal elements, which represent the number of correctly classified instances for each class. The off-diagonal elements represent the number of incorrectly classified instances for each class. By examining these elements, we can identify which classes the model is having the most difficulty with.

We can see that the model is making the most mistakes in predicting class 5 as there are some instances of off-diagonal elements in that row/column, especially for misclassification to classes 1 and 7. But the model seems to be performing well overall making only several mistakes for each class. Also, the best accuracy we obtain in predicting class 2 (second row, sixth column), as 171 out of 180 instances were correctly classified.

Overall, the confusion matrix allows us to identify where the model is making the most mistakes and can provide insights into how to improve the model's performance.



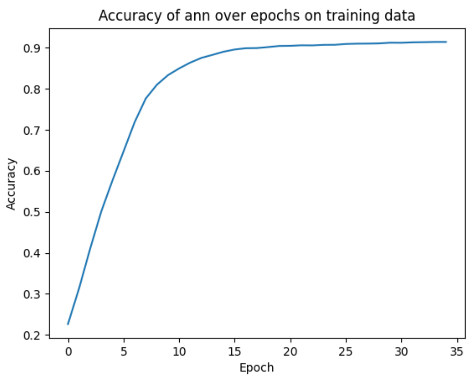
1.6 Scikit-Learn

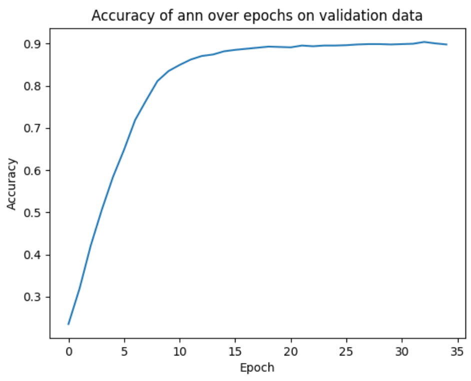
16 Optimized Parameters

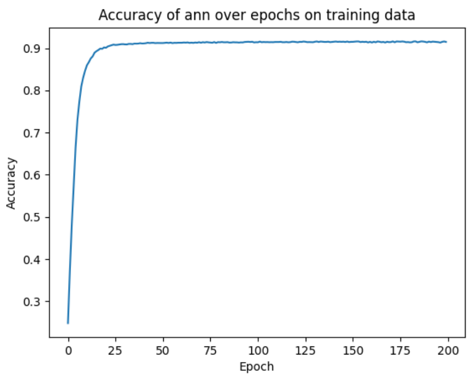
There are several differences between Scikit-Learn's choice of hyper-parameters and the choices made in our model. The main differences include activation function, number of iterations, learning rate and batch size. Changing the learning rate and the number of iterations introduced the most severe consequences, as an immediate change occurred in the speed of the training. Whereas, adjusting the activation function did not have much effect on the model as our problem has linear decision boundaries.

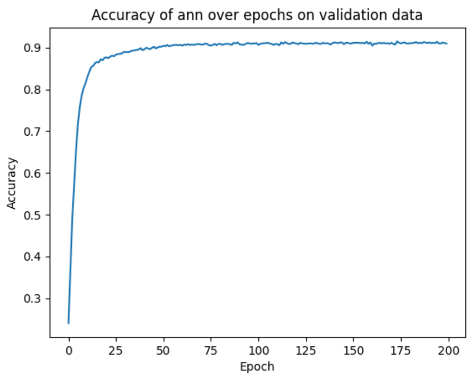
17 Trying best parameters

After adding changing all parameters in our model to the ones obtained form grid search that is activation function to identity, iterations to 200 (we didn't include early stop logic), learning rate to 0.01 and size of a mini batch to 200 we plotted the process of training on training and validation data similar to what we did in task 1.4.12. In case of 200 epochs we noticed some anomalies and non linear behaviour in the accuracy starting from 100th epoch this may suggest some overfitting to the training data. To better understand and visualize the changes we also decided to plot the training with Scikit-Learn parameters but for 35 epochs. When compared to the ones from exercise 1.4.12 we can see that the accuracy of model trained on our parameters increases quicker but not that smoothly. Both graphs look similar and can describe logarithmic relationship. The accuracy of the test set for 200 epochs we achieved was 0.925, the difference to our being smaller than 0.001. In summary we think that the parameters are better in case of controlling the behaviour and if indeed the stopping criterion stops the training after 35 epochs the overfitting and the difference in training time is comparable to our model.









1.7 Reflection

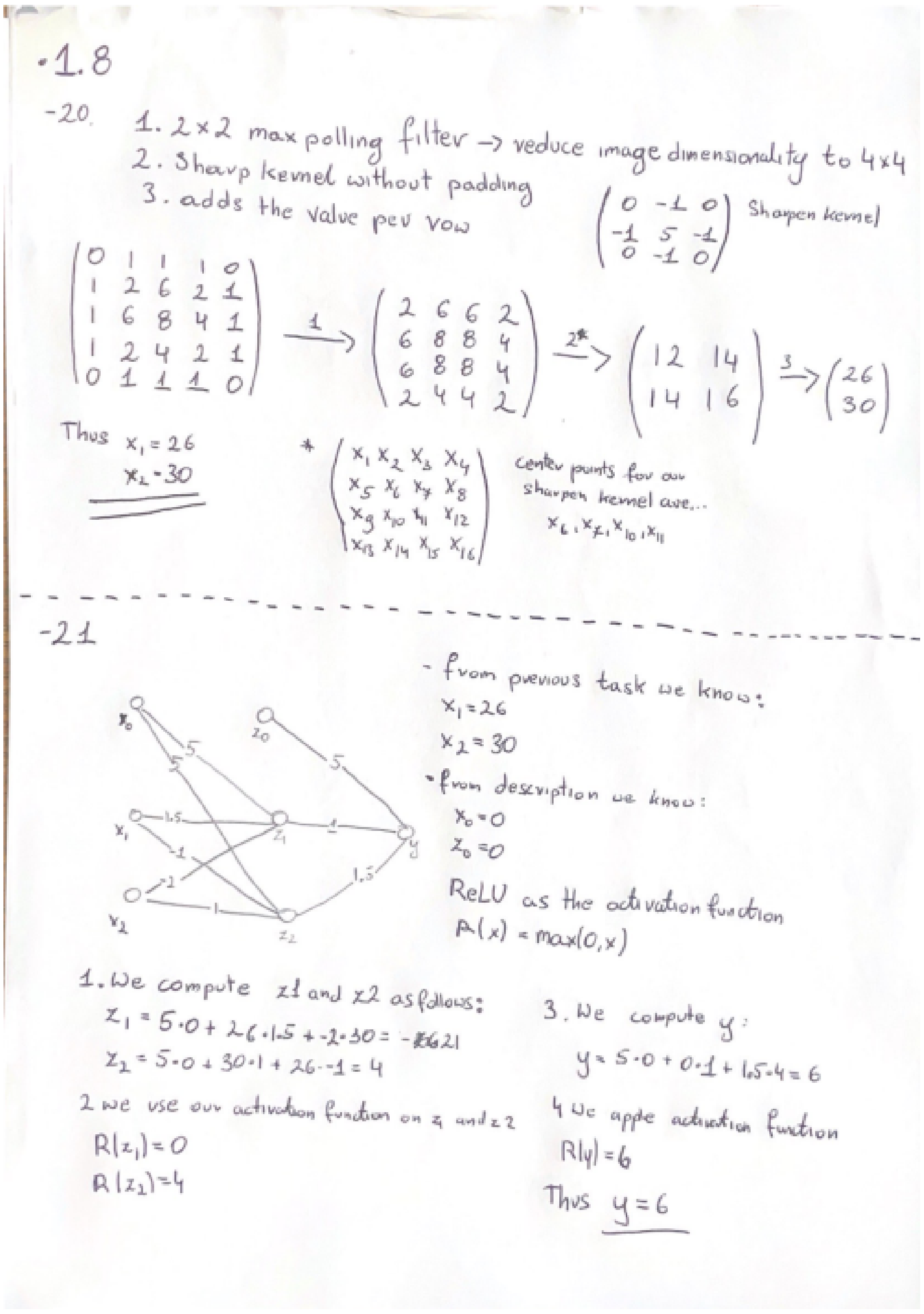
18 Missclassification and Harmful Consequences

One example of misclassification with harmful consequences is the use of facial recognition technology that disproportionately misidentifies individuals from certain racial or ethnic groups. This has resulted in wrongful arrests, surveillance, and even violence against innocent people. In some cases, this technology has been used to target marginalized communities and perpetuate systemic biases. These consequences can have long-lasting impacts on individuals and communities.

19 Ways to Mitigate the Harm

To mitigate the harm caused by unjust classification, one approach is to increase diversity and representation in the development and deployment of these technologies. This includes involving individuals from diverse backgrounds in decision-making processes, and auditing algorithms for biases. Another approach is to regulate the use of these technologies and ensure that they are used ethically and transparently. However, these solutions may not fully solve the problem and require ongoing effort and monitoring to ensure they are effective.

1.8 Pen and Paper



·1.8

-20. 1. 2×2 max polling filter → reduce image dimensionality to 4×4
2. Sharp kernel without padding
3. adds the value per row

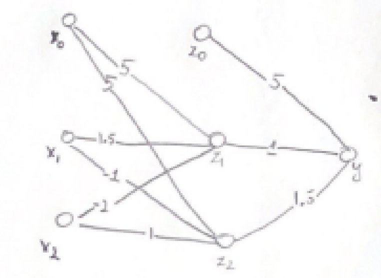
$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$ Sharpen kernel

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 2 & 6 & 2 & 1 \\ 1 & 6 & 8 & 4 & 1 \\ 1 & 2 & 4 & 2 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix} \xrightarrow{1} \begin{pmatrix} 2 & 6 & 6 & 2 \\ 6 & 8 & 8 & 4 \\ 6 & 8 & 8 & 4 \\ 2 & 4 & 4 & 2 \end{pmatrix} \xrightarrow{2*} \begin{pmatrix} 12 & 14 \\ 14 & 16 \end{pmatrix} \xrightarrow{3} \begin{pmatrix} 26 \\ 30 \end{pmatrix}$$

Thus $x_1 = 26$
$\underline{\underline{x_2 = 30}}$

$$* \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} \end{pmatrix}$$ center points for our sharpen kernel are... x_6, x_7, x_{10}, x_{11}

-21



- from previous task we know:
$x_1 = 26$
$x_2 = 30$
- from description we know:
$x_0 = 0$
$z_0 = 0$

ReLU as the activation function
$R(x) = \max(0, x)$

1. We compute z1 and z2 as follows:
$z_1 = 5 \cdot 0 + 26 \cdot 1.5 + -2 \cdot 30 = -$[illegible]21
$z_2 = 5 \cdot 0 + 30 \cdot 1 + 26 \cdot -1 = 4$

2 we use our activation function on z1 and z2
$R(z_1) = 0$
$R(z_2) = 4$

3. We compute y:
$y = 5 \cdot 0 + 0 \cdot 1 + 1.5 \cdot 4 = 6$

4 We apple activation function
$R(y) = 6$

Thus $\underline{y = 6}$