Marshall Lindsay
CS 4770
Lab1PartB

1)

For convergence testing I used the MSE and not the SSE. After each EPOC the data was shuffled. Figure 1 shows the MSE per EPOC for the initial values of alpha = 0.7, beta = 0.3.

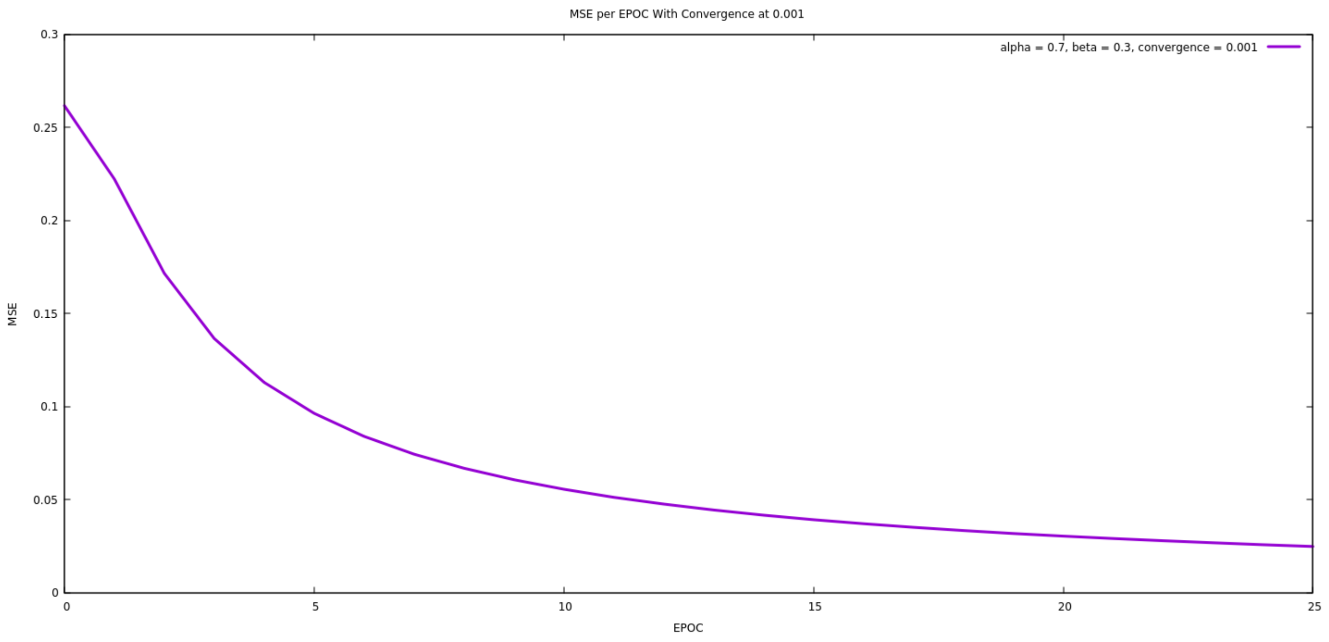

Figure 1: MSE per EPOC for alpha = 0.7, beta = 0.3 and a convergence factor of 0.001

We then tested this trained network against the square [-2.1,2.1] X [-2.1,2.1] with a granularity of 0.01 and plotted the predicted labels for each point. Label prediction was decided by:

$$Label = \begin{cases} 0 & output(0) > output(1) \\ 1 & output(0) < output(1) \end{cases}$$

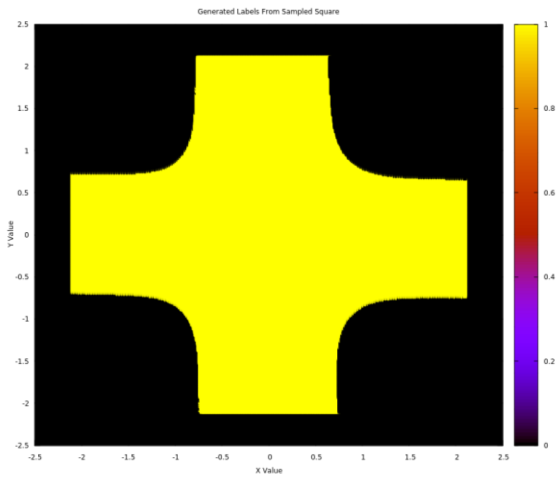

Figure 2: Predicted labels of the sampled square.

We found that removing the third value of the input data had little effect on the classification of the data set. The MSE per EPOC without the third data point is shown in Figure 3 and the boundary is shown in Figure 4. This independence is due to the ability to separate the data in 2-space and not needing to go to the third.

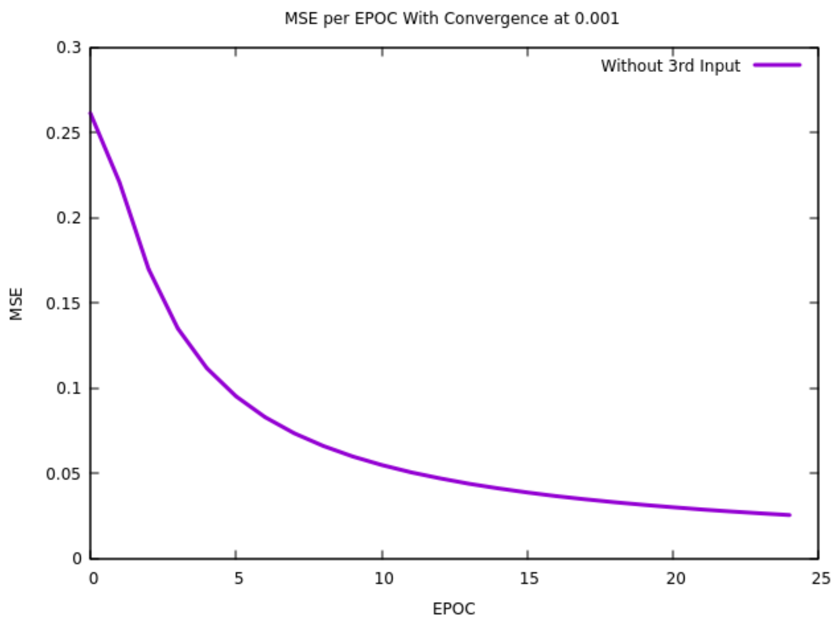

Figure 3 : MSE per EPOC without using the third point.

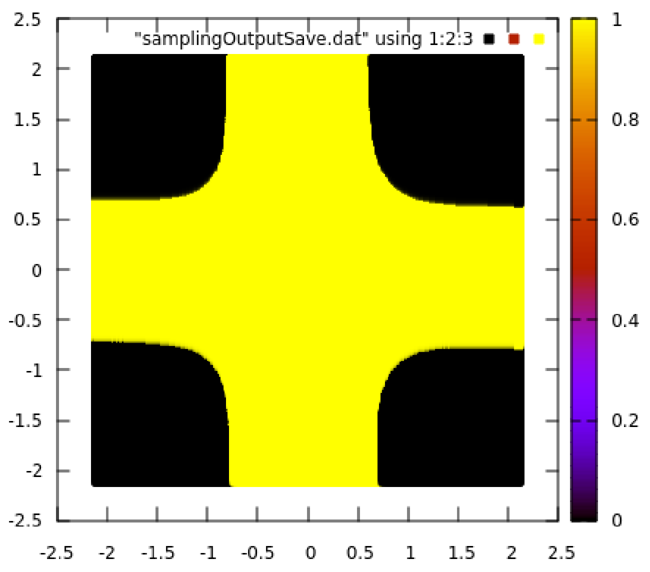

Figure 4: Predicted labels of the sampled square from the network trained without the third input.

2)

The network was then trained using different values for alpha (learning rate). The MSE was plotted per EPOC and a comparison is shown in Figure 5.

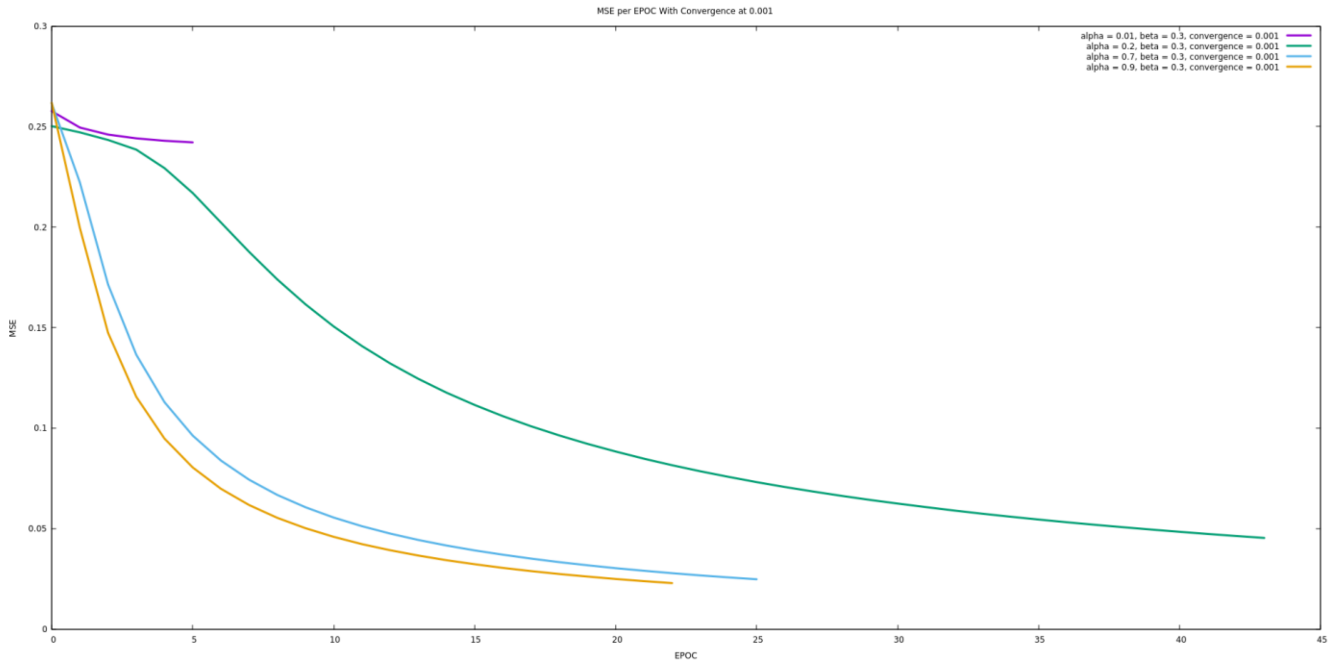

Figure 5: Comparison of the MSE per EPOC given different learning rates.

It's clear from the comparison graph that larger learning rates cause the network to converge faster. However, it was noted that for alpha = 0.01, the network was never "fully trained" i.e. the final MSE is much larger for alpha = 0.01 than the other values. We assumed this was because of our definition of convergence being that the delta MSE > 0.001. For small values of alpha, it's possible that the delta MSE is very small throughout the training process and the program quits before its properly trained. To test this theory, we changed our definition of convergence to delta MSE > 0.0001 and ran the tests again. The results confirmed our suspicion and the results are shown in Figure 6.

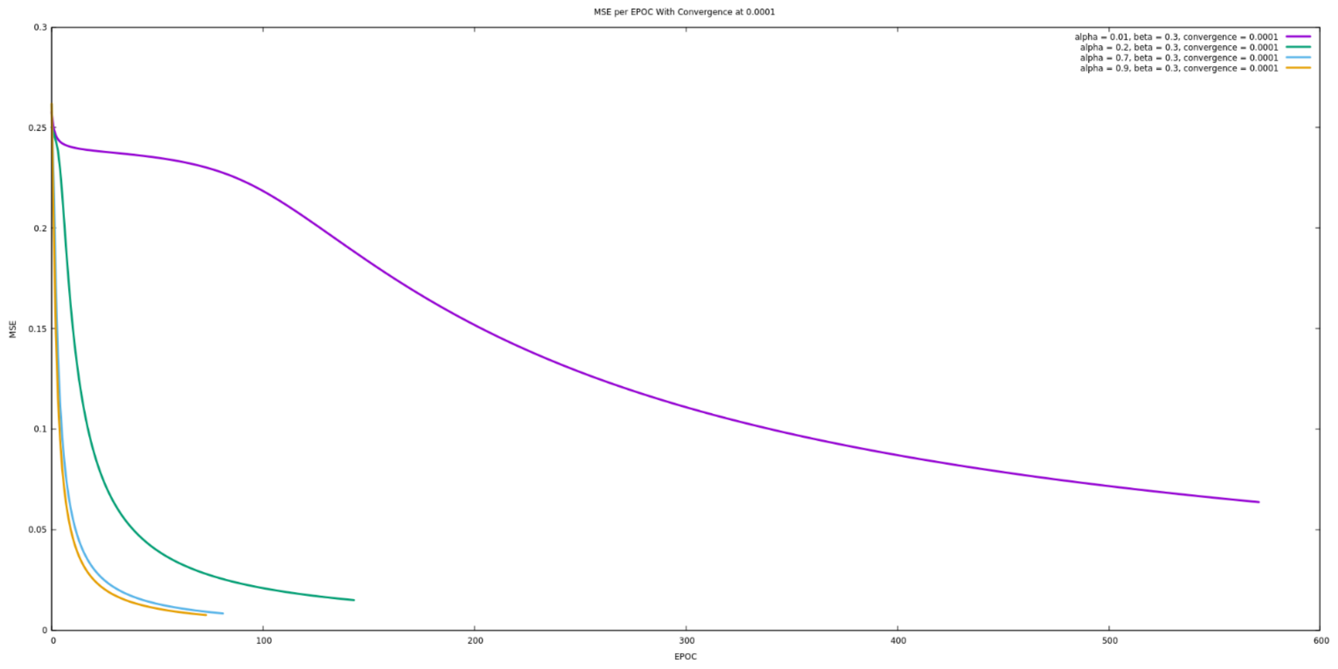

Figure 6: Comparison of different learning rates with a definition of convergence of delta MSE > 0.0001. It’s clear from this graph that with a learning rate of 0.01, the network takes more time to finally converge on a solution, but eventually fully trains.

3)

The network was retrained with variable beta (momentum rate) values and an alpha = 0.01. The results from these tests are shown in Figure 7.

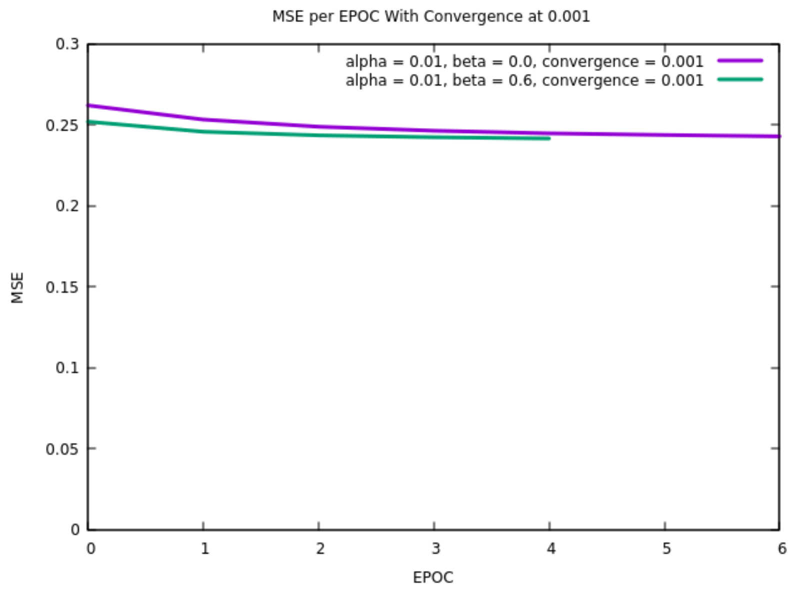

Figure 7: Comparison of MSE per EPOC for variable beta values.

Figure 7 shows terrible convergence on a solution. We assume this is partly due to the value of alpha and its associated problems from 2) but also on the beta values. We note that the as beta increases, the network trains to a better solution. We believe that the network would train to a smaller error if the alpha value was increased or if our definition of convergence changed to give the network a better chance.

4)

For this part we set up a 5-fold cross validation experiment against the gaussian data. This experiment was done for N = 2, N = 8, and N = 14. Random weights between (-2,2) were used to initialize the network. Results from training each configuration and the corresponding MSE per EPOC are shown in Figures 8 – 10. We define convergence for the training as $\Delta MSE \leq 0.001$.

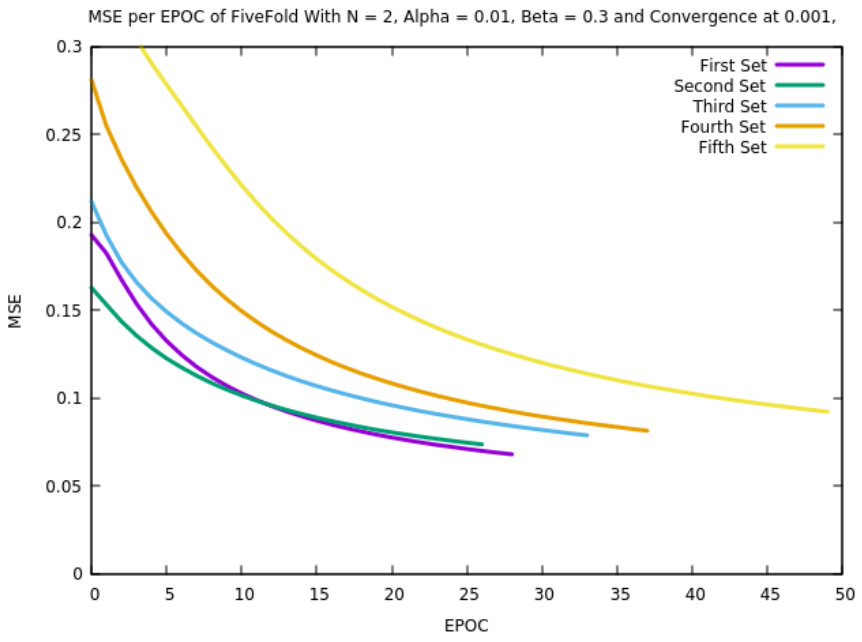

Figure 8: MSE per EPOC for the cross-validation experiment where N = 2.

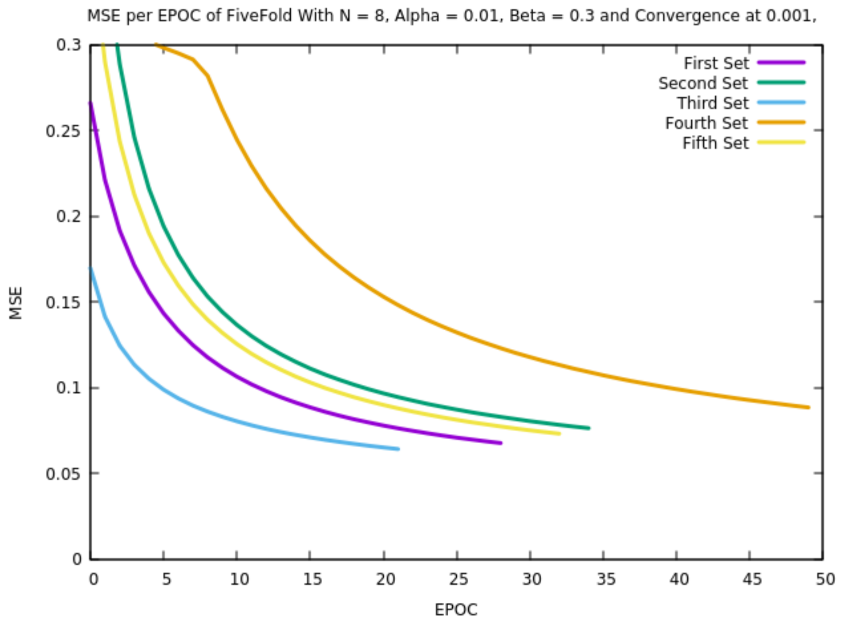

Figure 9: MSE per EPOC for the cross-validation experiment where N = 8.

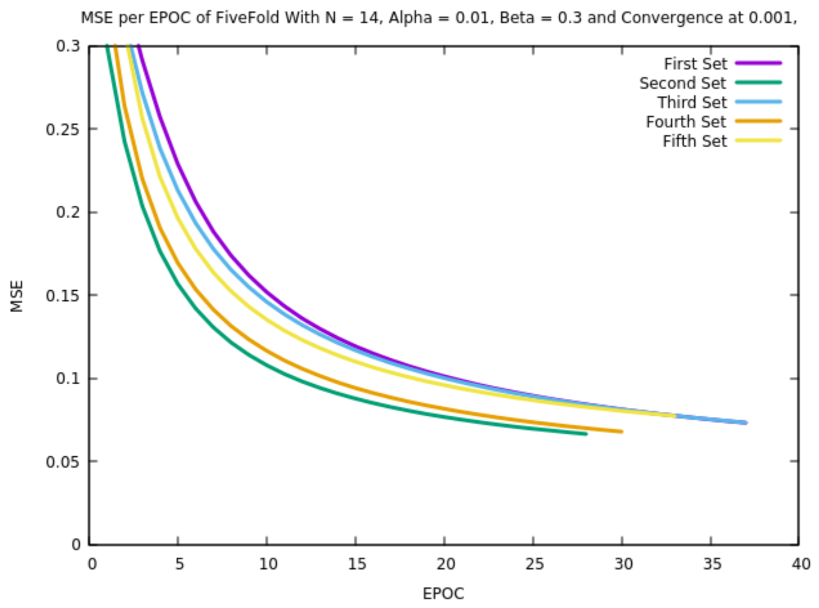

Figure 10: MSE per EPOC for the cross-validation experiment where N = 14.

After each configuration was trained, the networks were tested on the remaining data to produce confusion matrices. These matrices give an indication on how well the network is trained. The matrices and corresponding statistics are shown below.

$$\begin{bmatrix} 92 & 8 \\ 7 & 92 \end{bmatrix}$$

Figure 11: Confusion matrix for N = 2, first round of 5-fold cross-validation.

Measure	Value	Derivations
Sensitivity	0.9293	TPR = TP / (TP + FN)
Specificity	0.9200	SPC = TN / (FP + TN)
Precision	0.9200	PPV = TP / (TP + FP)
Negative Predictive Value	0.9293	NPV = TN / (TN + FN)
False Positive Rate	0.0800	FPR = FP / (FP + TN)
False Discovery Rate	0.0800	FDR = FP / (FP + TP)
False Negative Rate	0.0707	FNR = FN / (FN + TP)
Accuracy	0.9246	ACC = (TP + TN) / (P + N)
F1 Score	0.9246	F1 = 2TP / (2TP + FP + FN)
Matthews Correlation Coefficient	0.8493	TP*TN - FP*FN / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))

Table 1: Computed statistics for the confusion matrix corresponding to N = 2, first round of 5-fold cross validation.

$$\begin{bmatrix} 96 & 4 \\ 3 & 96 \end{bmatrix}$$

Figure 12: Confusion matrix for N = 2, second round of 5-fold cross-validation.

Measure	Value	Derivations
Sensitivity	0.9697	TPR = TP / (TP + FN)
Specificity	0.9600	SPC = TN / (FP + TN)
Precision	0.9600	PPV = TP / (TP + FP)
Negative Predictive Value	0.9697	NPV = TN / (TN + FN)
False Positive Rate	0.0400	FPR = FP / (FP + TN)
False Discovery Rate	0.0400	FDR = FP / (FP + TP)
False Negative Rate	0.0303	FNR = FN / (FN + TP)

Measure	Value	Derivations
Accuracy	0.9648	ACC = (TP + TN) / (P + N)
F1 Score	0.9648	F1 = 2TP / (2TP + FP + FN)
Matthews Correlation Coefficient	0.9297	TP*TN - FP*FN / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))

Table 2: Computed statistics for the confusion matrix corresponding to N = 2, second round of 5-fold cross validation.

$$\begin{bmatrix} 87 & 0 \\ 12 & 100 \end{bmatrix}$$

Figure 13: Confusion matrix for N = 2, third round of 5-fold cross-validation.

Measure	Value	Derivations
Sensitivity	0.8788	TPR = TP / (TP + FN)
Specificity	1.0000	SPC = TN / (FP + TN)
Precision	1.0000	PPV = TP / (TP + FP)
Negative Predictive Value	0.8929	NPV = TN / (TN + FN)
False Positive Rate	0.0000	FPR = FP / (FP + TN)
False Discovery Rate	0.0000	FDR = FP / (FP + TP)
False Negative Rate	0.1212	FNR = FN / (FN + TP)
Accuracy	0.9397	ACC = (TP + TN) / (P + N)
F1 Score	0.9355	F1 = 2TP / (2TP + FP + FN)
Matthews Correlation Coefficient	0.8858	TP*TN - FP*FN / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))

Table 3: Computed statistics for the confusion matrix corresponding to N = 2, third round of 5-fold cross validation.

$$\begin{bmatrix} 90 & 4 \\ 9 & 96 \end{bmatrix}$$

Figure 14: Confusion matrix for N = 2, fourth round of 5-fold cross-validation.

Measure	Value	Derivations
Sensitivity	0.9091	TPR = TP / (TP + FN)
Specificity	0.9600	SPC = TN / (FP + TN)
Precision	0.9574	PPV = TP / (TP + FP)
Negative Predictive Value	0.9143	NPV = TN / (TN + FN)
False Positive Rate	0.0400	FPR = FP / (FP + TN)
False Discovery Rate	0.0426	FDR = FP / (FP + TP)
False Negative Rate	0.0909	FNR = FN / (FN + TP)
Accuracy	0.9347	ACC = (TP + TN) / (P + N)
F1 Score	0.9326	F1 = 2TP / (2TP + FP + FN)
Matthews Correlation Coefficient	0.8704	TP*TN - FP*FN / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))

Table 4: Computed statistics for the confusion matrix corresponding to N = 2, fourth round of 5-fold cross validation.

$$\begin{bmatrix} 91 & 0 \\ 8 & 100 \end{bmatrix}$$

Figure 15: Confusion matrix for N = 2, fifth round of 5-fold cross-validation.

Measure	Value	Derivations
Sensitivity	0.9192	TPR = TP / (TP + FN)
Specificity	1.0000	SPC = TN / (FP + TN)

Measure	Value	Derivations
Precision	1.0000	PPV = TP / (TP + FP)
Negative Predictive Value	0.9259	NPV = TN / (TN + FN)
False Positive Rate	0.0000	FPR = FP / (FP + TN)
False Discovery Rate	0.0000	FDR = FP / (FP + TP)
False Negative Rate	0.0808	FNR = FN / (FN + TP)
Accuracy	0.9598	ACC = (TP + TN) / (P + N)
F1 Score	0.9579	F1 = 2TP / (2TP + FP + FN)
Matthews Correlation Coefficient	0.9226	TP*TN - FP*FN / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))

Table 5: Computed statistics for the confusion matrix corresponding to N = 2, fifth round of 5-fold cross validation.

$$\begin{bmatrix} 456 & 16 \\ 39 & 484 \end{bmatrix}$$

Figure 16: Aggregate confusion matrix for N = 2.

Measure	Value	Derivations
Sensitivity	0.9212	TPR = TP / (TP + FN)
Specificity	0.9680	SPC = TN / (FP + TN)
Precision	0.9661	PPV = TP / (TP + FP)
Negative Predictive Value	0.9254	NPV = TN / (TN + FN)
False Positive Rate	0.0320	FPR = FP / (FP + TN)
False Discovery Rate	0.0339	FDR = FP / (FP + TP)
False Negative Rate	0.0788	FNR = FN / (FN + TP)
Accuracy	0.9447	ACC = (TP + TN) / (P + N)
F1 Score	0.9431	F1 = 2TP / (2TP + FP + FN)

Measure	Value	Derivations
Matthews Correlation Coefficient	0.8904	TP*TN - FP*FN / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))

Table 6: Computed statistics for the aggregate of confusion matrices corresponding to N = 2.

$$\begin{bmatrix} 461 & 17 \\ 34 & 483 \end{bmatrix}$$

Figure 17: Aggregate confusion matrix for N = 8.

Measure	Value	Derivations
Sensitivity	0.9313	TPR = TP / (TP + FN)
Specificity	0.9660	SPC = TN / (FP + TN)
Precision	0.9644	PPV = TP / (TP + FP)
Negative Predictive Value	0.9342	NPV = TN / (TN + FN)
False Positive Rate	0.0340	FPR = FP / (FP + TN)
False Discovery Rate	0.0356	FDR = FP / (FP + TP)
False Negative Rate	0.0687	FNR = FN / (FN + TP)
Accuracy	0.9487	ACC = (TP + TN) / (P + N)
F1 Score	0.9476	F1 = 2TP / (2TP + FP + FN)
Matthews Correlation Coefficient	0.8980	TP*TN - FP*FN / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))

Table 7: Computed statistics for the aggregate of confusion matrices corresponding to N = 8.

$$\begin{bmatrix} 457 & 17 \\ 38 & 483 \end{bmatrix}$$

Figure 18: Aggregate confusion matrix for N = 14.

Measure	Value	Derivations
Sensitivity	0.9232	TPR = TP / (TP + FN)
Specificity	0.9660	SPC = TN / (FP + TN)
Precision	0.9641	PPV = TP / (TP + FP)
Negative Predictive Value	0.9271	NPV = TN / (TN + FN)
False Positive Rate	0.0340	FPR = FP / (FP + TN)
False Discovery Rate	0.0359	FDR = FP / (FP + TP)
False Negative Rate	0.0768	FNR = FN / (FN + TP)
Accuracy	0.9447	ACC = (TP + TN) / (P + N)
F1 Score	0.9432	F1 = 2TP / (2TP + FP + FN)
Matthews Correlation Coefficient	0.8902	TP*TN - FP*FN / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))

Table 8: Computed statistics for the aggregate of confusion matrices corresponding to N = 14.

From the confusion matrices and the computed stats, we note that there isn't a noticeable difference in the final performance of the trained network for different values of N. However, when we take a look at the MSE per EPOC, we see that for larger values of N we produce more consistent training patterns. This can be observed in the clumping of MSE lines in Figures 8 – 10.

The main thing I learned from this lab was how variable these networks can be. We can *think* we have a trained network based on our definition of "trained" when in reality it's nowhere near a solution. In order to properly use neural nets, you must pay attention to all of the adjustable variables, and the calculated outputs to see if it is behaving how you assume it should. Also, there seems to be some dumb luck with initialization. If I initialize and guess well, I have a better chance at converging than if I guess bad.