Assignment 2

Machine Learning KU, SS 2025

Team Member		
Last name	First name	Matriculation Number
Forster	Greg	12406972
Donohue	Joseph	12443763

1.1

The variance of our data was preserved at $\sim 65.78\%$ from our PCA.

1.2

Layer size	Best Loss	Final Loss	Training Accuracy	Test Accuracy
(2,)	0.8973	0.8973	0.625	0.519
(8,)	0.4779	0.4779	0.843	0.709
(64,)	0.0953	0.0953	0.993	0.747
(256,)	0.0169	0.0169	0.999	0.766
(1024,)	0.0045	0.0045	1.000	0.756
(128,256,128)	0.0019	0.0041	1.000	0.731

Table 1: Validation scores for our different layer sizes.

1.1.3

A model will (generally) begin to overfit if the training accuracy becomes higher than the test set accuracy. If both are low, the model is most likely underfit.

Underfitting occurs with (2) neurons, mild underfitting occurs with (8) neurons. Overfitting occurs with the rest, at (64), (256), (1024), (128, 256, 128) neurons, with most having close to a training accuracy of 100% and test set accuracy around 75%. That being said, we would prefer the 256 neuron model as it has the highest test set accuracy. Additional neurons does not seem to improve it.

1.1.4

What better way to make a decision than with data? Yes, it improves the results from the previous step. Bumping the default alpha of 0.0001 to 0.1 proved to have the best results. Enabling early stopping caused a lower accuracy and higher loss across the board.

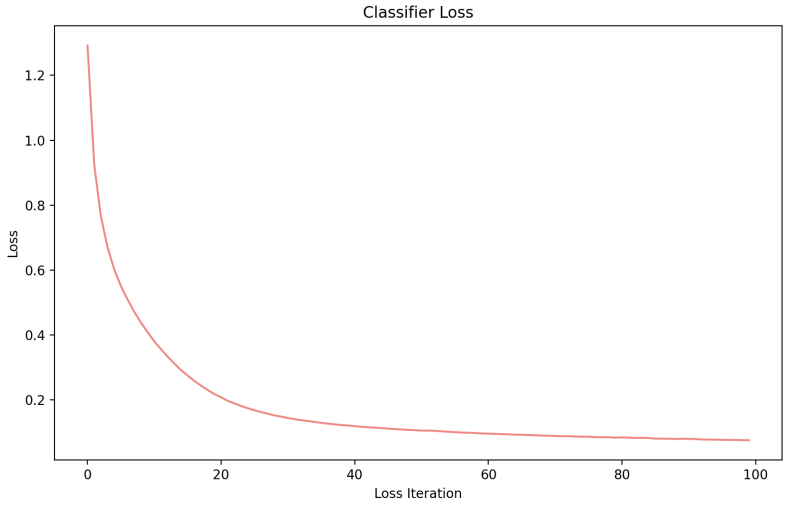
Increasing the alpha to **0.1** helps with regularization of the weights. This in turn helps prevent the model from creating large weights, which in the case of our overfitting models, leads to a higher test accuracy.

Case	Layer size	Best Loss	Final Loss	Training Accuracy	Test Accuracy
(a)	(2,)	0.9011	0.9011	0.627	0.519
(a)	(8,)	0.4917	0.4917	0.841	0.703
(a)	(64,)	0.1589	0.1589	0.992	0.744
(a)	(256,)	0.0987	0.0987	0.998	0.772
(a)	(1024,)	0.0763	0.0763	1.000	**0.781**
(a)	(128,256,128)	0.0777	0.0777	1.000	0.744
(b)	(2,)	1.1224	1.1224	0.452	0.403
(b)	(8,)	0.6293	0.6293	0.770	0.659
(b)	(64,)	0.4790	0.4790	0.805	0.684
(b)	(256,)	0.1711	0.1711	0.920	0.719
(b)	(1024,)	0.1988	0.1988	0.835	0.747
(b)	(128,256,128)	0.0201	0.0201	0.941	0.728
(c)	(2,)	1.1430	1.1430	0.440	0.394
(c)	(8,)	0.6385	0.6385	0.767	0.663
(c)	(64,)	0.4980	0.4980	0.810	0.681
(c)	(256,)	0.2393	0.2393	0.919	0.722
(c)	(1024,)	0.2797	0.2797	0.834	0.747
(c)	(128,256,128)	0.1548	0.1548	0.922	0.728

Table 2: Validation scores for different layer sizes and cases (a-c).

1.1.5

We had a test set accuracy of 78.125% for our best model. The loss curve for this model is shown below. Note, the loss for this model is higher than the one without the alpha modification (because it has a higher penalty when calculating the loss). However, test accuracy is king.



1.2.1

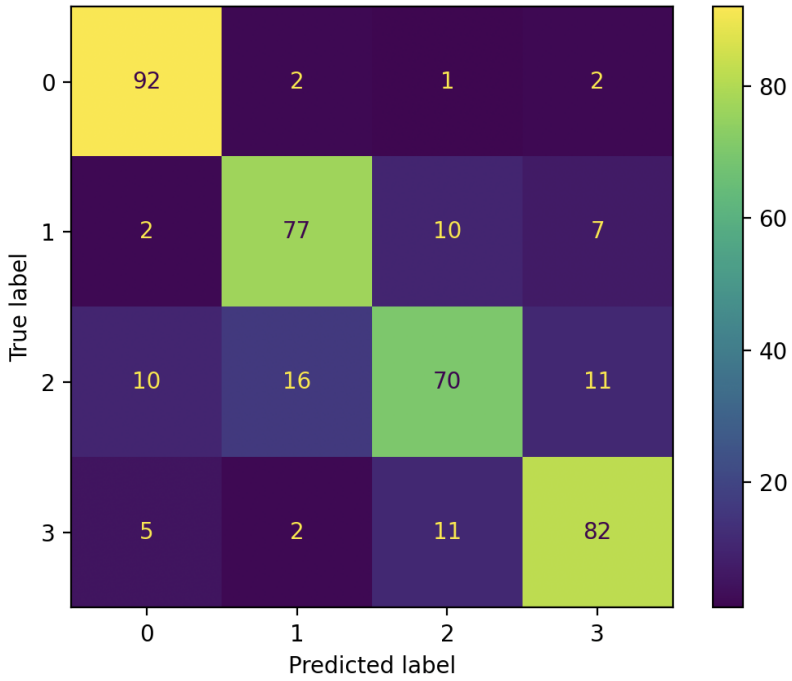
We calculated the number of architectures by finding the total number of combinations possible. With three possible alpha values, two batch sizes and two hidden layer sizes, the math came out to be:
$3 * 2 * 2 = 12$ total combinations or architectures.

1.2.3

The best architecture we found was (128), alpha 0.1, and batch size 32. It has a score (average across the 5 folds) of 76.5%.

1.2.4

We chose the model from the grid search, since it has a final test accuracy of **80.25%** while the regularization model only had a final test accuracy of 74%.



	precision	recall	f1-score	support
0	0.84	0.95	0.89	97
1	0.79	0.80	0.80	96
2	0.76	0.65	0.70	107
3	0.80	0.82	0.81	100
accuracy			0.80	400
macro avg	0.80	0.81	0.80	400
weighted avg	0.80	0.80	0.80	400

1.2.5

Recall is the fraction that were positive, positively identified of the dataset.
Precision is the fraction to which the model's positive classifications are actually positive.
The true label 2 was misclassified the most often, as it had 37 missed predictions.

1.2.6

Since the parameters are continuously updated, these values learn the patterns from the data. Unlike parameters, hyperparameters stay constant, dictating the way the model is training, as they are specified before training. In terms of neural networks, learning rate, activation function, number of hidden layers and epochs are examples of hyperparameters, as they stay constant throughout the learning process. Weights and biases are an example of a parameter, as they can change throughout the learning process and are impacted by the preset hyperparameters.

2.5

The final accuracies are: Train accuracy: 72.89%. Validation accuracy: 68.75%. Test accuracy: 67.25%

2.6

We tested alpha = 0.1 and alpha = 0.01
Accuracies for alpha 0.1 are: Train accuracy: 74.21%. Validation accuracy: 70%. Test accuracy: 68%
Accuracies for alpha 0.01 are: Train accuracy: 72.89%. Validation accuracy: 68.75%. Test accuracy: 67.50%

2.7

(a)

$\frac{\partial f}{\partial f} = 1$

$\frac{\partial f}{\partial s} = \frac{\partial f}{\partial f} * \frac{\partial f}{\partial s} = 1 * 2s = 2s$

$\frac{\partial f}{\partial r} = \frac{\partial f}{\partial s} * \frac{\partial s}{\partial r} = 2s * -1 = -2s$

$\frac{\partial f}{\partial \hat{y}} = \frac{\partial f}{\partial s} * \frac{\partial s}{\partial \hat{y}} = 1 * 2s = 2s$

$\frac{\partial f}{\partial o} = \frac{\partial f}{\partial r} * \frac{\partial r}{\partial o} = 1 * -2s = -2s$

$\frac{\partial f}{\partial q} = \frac{\partial f}{\partial r} * \frac{\partial r}{\partial q} = 1 * -2s = -2s$

$\frac{\partial f}{\partial n} = \frac{\partial f}{\partial o} * \frac{\partial o}{\partial n} = -2s * \cos(n)$

$\frac{\partial f}{\partial p} = \frac{\partial f}{\partial q} * \frac{\partial q}{\partial p} = -2s * \exp(p)$

$\frac{\partial f}{\partial x} = (\frac{\partial f}{\partial n} * \frac{\partial n}{\partial x}) + (\frac{\partial f}{\partial p} * \frac{\partial p}{\partial x}) = [(-2s * \cos(mx)) * m] + [(-2s * \exp(ax)) * a]$

$\frac{\partial f}{\partial m} = \frac{\partial f}{\partial n} * \frac{\partial n}{\partial m} = -2sx * \cos(mx) * x$

$\frac{\partial f}{\partial a} = (\frac{\partial f}{\partial p} * \frac{\partial p}{\partial a}) + (\frac{\partial f}{\partial m} * \frac{\partial m}{\partial a}) = [(-2s * \exp(a*x) * x] + [(-2sx * \cos(a^2x) * 2a)$

Where $s = \hat{y} - r$

(b) Linear functions aren't often representative of real world datasets. Nonlinear functions are required

because if a linear function is used, no matter how many hidden layers in the neural network, the result would be the same as a single perceptron. Summing these layers together would be concluding in another linear function.

(c) Typically, when the number of hidden layers in a neural network increases, it becomes harder to train. Another error that can occur is the gradient either dropping to 0 or becoming far too large. This leads to the training error rate and the test error rate to increase.

(d) L2 regularization significantly decreases the impact of weights, especially larger ones.

3.2

Modifying the alpha to 0.1 did not have an effect on the final test accuracy, only a minor improvement on the train accuracy.

The final results: Train accuracy: 96.72%. Validation accuracy: 91.88%. Test accuracy: 92.5%.

3.3

Due to the subject matter of this particular case, accuracy can be a misleading metric. Precision is a more useful metric when dealing with a binary classification of brain cancer. This takes into account the number of patients who are incorrectly diagnosed as non-cancerous, while accuracy purely looks at the number of correct predictions vs. total predictions.