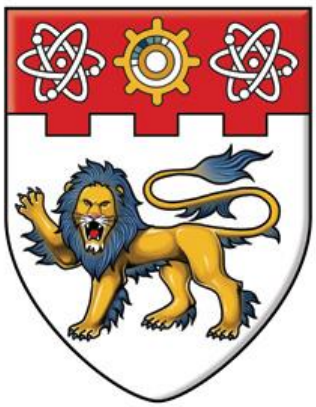

NANYANG TECHNOLOGICAL UNIVERSITY

Project 1 Report

CZ4042: Neural Network & Deep Learning

Student Name: Chua Wen Kai

Matriculation Number: U1722307B

Student Name: Chew Jing Wei

Matriculation Number: U1720459J

1 Table of Contents

2 Part A: Classification Problem

2.1 Introduction

The aim is to build neural networks to classify the Cardiotocography dataset which contains measurements of fetal heart rate (FHR) and uterine contraction (UC) features on 2126 cardiotocograms classified by expert obstetricians.

The cardiotocograms were classified by three expert obstetricians and a consensus classification label with respect to a morphologic pattern and to a fetal state (N: Normal; S: Suspect; P: Pathologic) was assigned to each of them. The resulting model should be able to predict the N, S and P class labels in the test dataset after training the neural network on the training dataset.

Each record is a row of 23 values: 21 input attributes and 2 class labels. For this experiment, all 21 input attributes and only 1 class label (only the NSP label) would be used.

2.2 Methods

Data Pre-processing

For the Cardiotocography classification problem, the data are imported from "ctg_data_cleaned.csv" using the pandas library. Looking at imported dataset, it is evident that the ranges between the features are quite varied from e-03 to e02.

```
[[1.20e+02 0.00e+00 0.00e+00 ... 1.21e+02 7.30e+01 1.00e+00]
 [1.32e+02 6.00e-03 0.00e+00 ... 1.40e+02 1.20e+01 0.00e+00]
 [1.33e+02 3.00e-03 0.00e+00 ... 1.38e+02 1.30e+01 0.00e+00]
 ...
 [1.40e+02 1.00e-03 0.00e+00 ... 1.52e+02 4.00e+00 1.00e+00]
 [1.40e+02 1.00e-03 0.00e+00 ... 1.51e+02 4.00e+00 1.00e+00]
 [1.42e+02 2.00e-03 2.00e-03 ... 1.45e+02 1.00e+00 0.00e+00]]
```

Figure 1A Output of dataset containing the input attributes

Hence, we perform normalization on the data to reduce the data redundancy and improve the integrity of the data using the formula below:

```
#Normalize the training set to range min and max of 0 and 1
def normalize(X, X_min, X_max):
    return (X - X_min) / (X_max - X_min)
```

Figure 1B Formula to perform normalization

For the output label Y, we convert all the 3 possible labels into one-hot encoding to aid the process in matrix multiplication during the training process using the formula below:

```
#Define the one hot encode function
def one_hot_encode(labels):
    n_labels = len(labels)
    n_unique_labels = len(np.unique(labels))
    one_hot_encode = np.zeros((n_labels, n_unique_labels))
    one_hot_encode[np.arange(n_labels), labels-1] = 1
    return one_hot_encode
```

Figure 1C Formula to perform one hot encoding

The dataset is also divided into 70:30 ratio for training and testing purposes. To aid this process, we utilised the library from sklearn to help with the shuffling and splitting of data.

This results in 1488 data belonging to the training set and 638 data belonging to the testing set.

```
def shuffleSplit(X, Y, test_size):
    X, Y = shuffle(X, Y, random_state=1)
    train_X, test_X, train_Y, test_Y = train_test_split(X, Y, test_size = test_size, random_state=2)
    return train_X, test_X, train_Y, test_Y
```

Figure 1D Function to perform shuffling and splitting

Model

To build a 3-layer forward neural network, we use the following parameters:

- All weights are initialized from a truncated normal distribution.
- All biases are set to zero.
- Hidden layer consists of ReLU units.
- Output layer is a softmax layer for multi-class classification.
- Cross-entropy is used as a loss function.
- L2 regularization with decay is applied on the loss.
    - Weight Decay (β) * Σ sqt(weights)/2
- Mini batch gradient descent learning used for training.

The model architecture and hyper-parameters are as below:

Layers	Shape	Activation
Input	(None, 21)	-
Hidden	(21, 10)	ReLU
Output	(10, 3)	softmax
Learning Rate, α =0.01 Batch size = 32 Number of hidden neurons = 10 Weight decay parameter β = 1e-6		

2.3 Experiment & Results

2.3.1 Question 1

a) Use the training dataset to train the model and plot both accuracies on training and testing data against epochs.
b) State the approximate number of epochs where the test error converges.

For question 1, a total of 10,000 epochs of training have been performed on the 1488 records from the randomly selected 70% of the dataset. The model is then evaluated using 638 records from the remaining 30% which will provide us with the test accuracy.

During the process of training, a milestone of 1000 epochs is used to plot the train accuracy, test accuracy and loss of the model. Below are the results of every 1000 epochs in a table format.

Refer to Appendix A Question 1 for the respective visual plots on both accuracies on training and testing data against epochs.

Epochs	Training Loss (4 s.f.)	Training Accuracy (3 s.f.)	Test Accuracy (3 s.f.)
1,000	0.2071	0.910	0.893
2,000	0.1927	0.916	0.903
3,000	0.1851	0.923	0.903
4,000	0.1722	0.927	0.917
5,000	0.1584	0.937	0.918
6,000	0.1496	0.947	0.918
7,000	0.1442	0.943	0.920
8,000	0.1402	0.942	0.926
9,000	0.1373	0.942	0.925
10,000	0.1349	0.942	0.931

Next, the table below shows the difference between the latest number epochs against its previous number of epochs recorded.

Epochs	Training Loss Compared to Previous (4 s.f.)	Training Accuracy Compared to Previous (3 s.f.)	Test Accuracy Compared to Previous (3 s.f.)
1,000	-	-	-
2,000	-0.0144	0.006	0.01
3,000	-0.0076	0.007	0
4,000	-0.0129	0.004	0.014
5,000	-0.0138	0.01	0.001
6,000	-0.0088	0.01	0
7,000	-0.0054	-0.004	0.002
8,000	-0.004	-0.001	0.006
9,000	-0.0029	0	-0.001
10,000	-0.0024	0	0.006

Looking at the training loss compared to the previous epochs, epochs 3000, 6000, 7000, 8000, 9000 and 10000, all have a difference of less than 0.01. If we were to consider the time needed to train more epochs, the difference between the training loss at epochs 10000 and epochs 3000 is about 0.05 only, however the training time required is at least tripled.

Using this observation above, around epochs 3000 should be where the model converges. However, at epochs 4000, the training loss has a sudden spike difference of more than 0.01 which then continues until epochs 6000. It could be at epochs 3000, the gradient just happened to be steeper than the rest hence the training loss difference of less than 0.01.

Therefore, it is determined that the model **convergences at around 5000 to 6000 epochs.**

This is also further supported by the training and testing accuracies where at epoch 3000, there are no improvements to the test accuracy even though the training accuracy has improved. At epoch 6000, the training and testing accuracies have also started to become stagnant and even deteriorated at epochs 7000 onwards.

2.3.2 Question 2

a) *Plot cross-validation accuracies against the number of epochs for different batch sizes. Limit search space to batch sizes {4,8,16,32,64}. Plot the time taken to train the network for one epoch against different batch sizes.*

From Question 1b), it is determined that the model convergences from around 5000 to 6000 epochs hence for this experiment, **5000 epochs** have been used to perform the training for each batch sizes using 5-fold cross validation.

During the process of training, we plot the training loss, cross-validation and training accuracy of the model for different batch sizes at epochs 5000. Below are the results in a table format.

Refer to Appendix A Question 2 for the respective visual plots on cross-validation accuracies against the number epochs for different batch sizes.

Epochs	Fold	Training Accuracy (3 s.f.)	Cross-Validation Accuracy (3 s.f.)
Batch Size 4			
5000	1	0.974	0.899
	2	0.946	0.906
	3	0.962	0.892
	4	0.973	0.940
	5	0.945	0.906
Mean		**0.96**	**0.9086**
Batch Size 8			
5000	1	0.968	0.906
	2	0.951	0.909
	3	0.948	0.899
	4	0.948	0.940
	5	0.962	0.923
Mean		**0.9554**	**0.9154**
Batch Size 16			

5000	1	0.950	0.889
	2	0.951	0.909
	3	0.952	0.916
	4	0.936	0.926
	5	0.960	0.923
Mean		**0.9498**	**0.9126**
Batch Size 32			
5000	1	0.936	0.886
	2	0.930	0.916
	3	0.921	0.899
	4	0.922	0.923
	5	0.930	0.916
Mean		**0.9278**	**0.908**
Batch Size 64			
5000	1	0.915	0.886
	2	0.916	0.913
	3	0.910	0.892
	4	0.911	0.903
	5	0.900	0.903
Mean		**0.9104**	**0.8994**

The time taken to train the network for one epoch against different batch sizes using 5-fold cross validation is shown below.

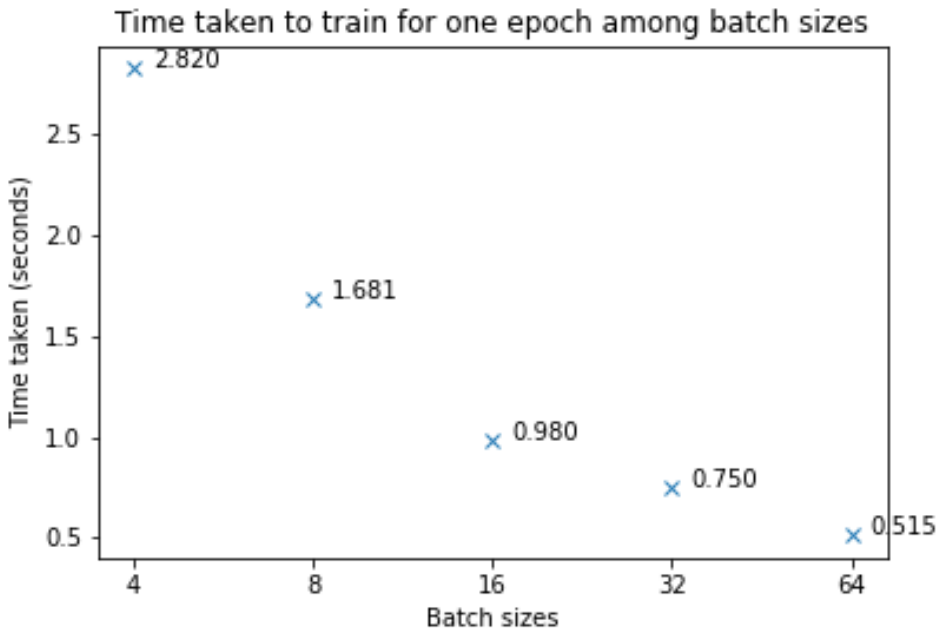

Figure 2.1 Time Taken for one epoch for different batch sizes with 5-fold cross validation

b) Select the optimal batch size and state reasons for your selection.
Using the model and results obtained in 2a), three graphs have been plotted showing the comparison between the training accuracy, cross-validation accuracy and training loss among the different batch sizes.

Figure 2.2: Training Accuracy Among Batch Sizes

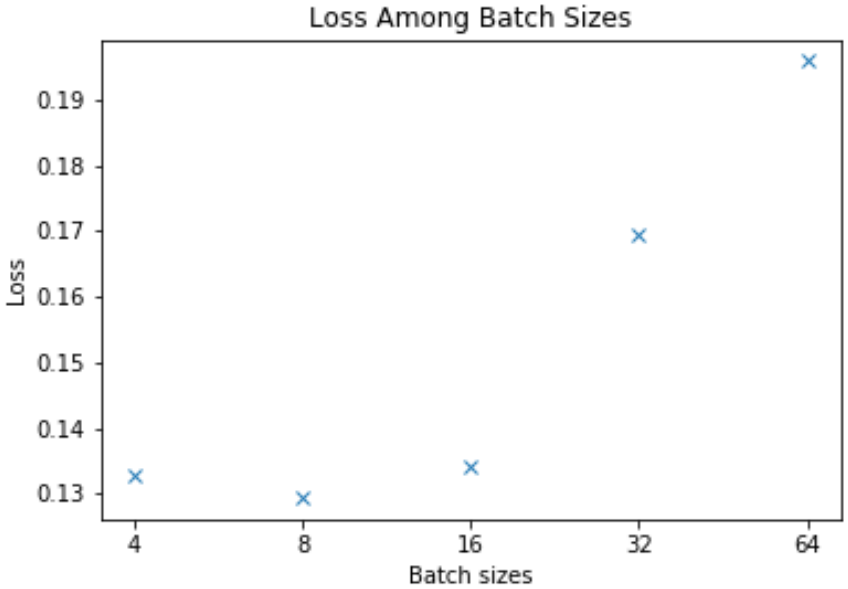

Figure 2.3: Training Loss Among Batch Sizes

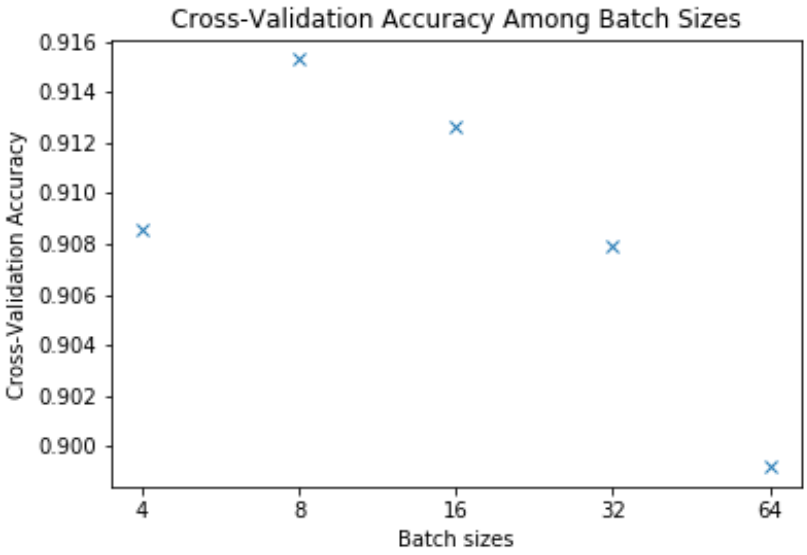

Figure 2.4: Cross-Validation Accuracy Among Batch Sizes

The information depicted by Figures 2.2, 2.3 and 2.4 are consolidated into the table below and ranked based on the optimal value that a batch can attain.

For training loss, the optimal would be the lowest value. For the training and cross-validation accuracy, the optimal would be the highest value.

Batch Size	**Training Accuracy**	**Training Loss**	**Cross-Validation Accuracy**
	Rankings		
4	1	2	3
8	2	1	1
16	3	3	2
32	4	4	4
64	5	5	5

Based on the table above, the **optimal batch size is size of 8.** The reason is that batch size 8 ranked first for having both the lowest training loss and highest cross-validation accuracy in comparison to the rest.

c) Plot the train and test accuracies against epochs for the optimal batch size.

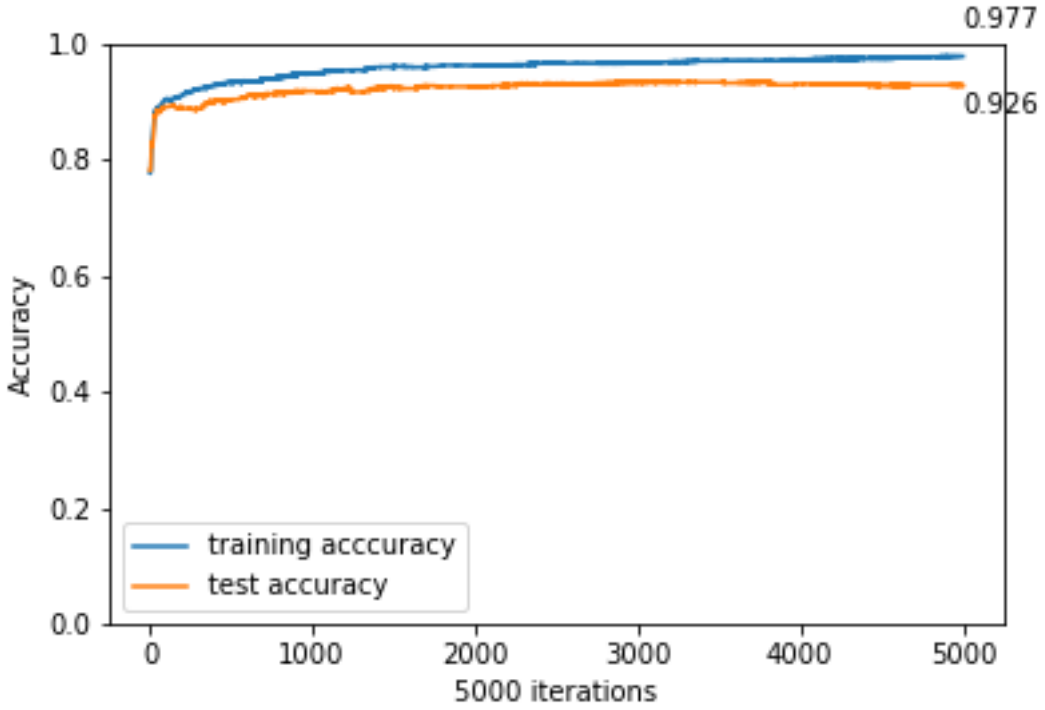

Figure 2.5 Train and Test Accuracies for Batch Size 8

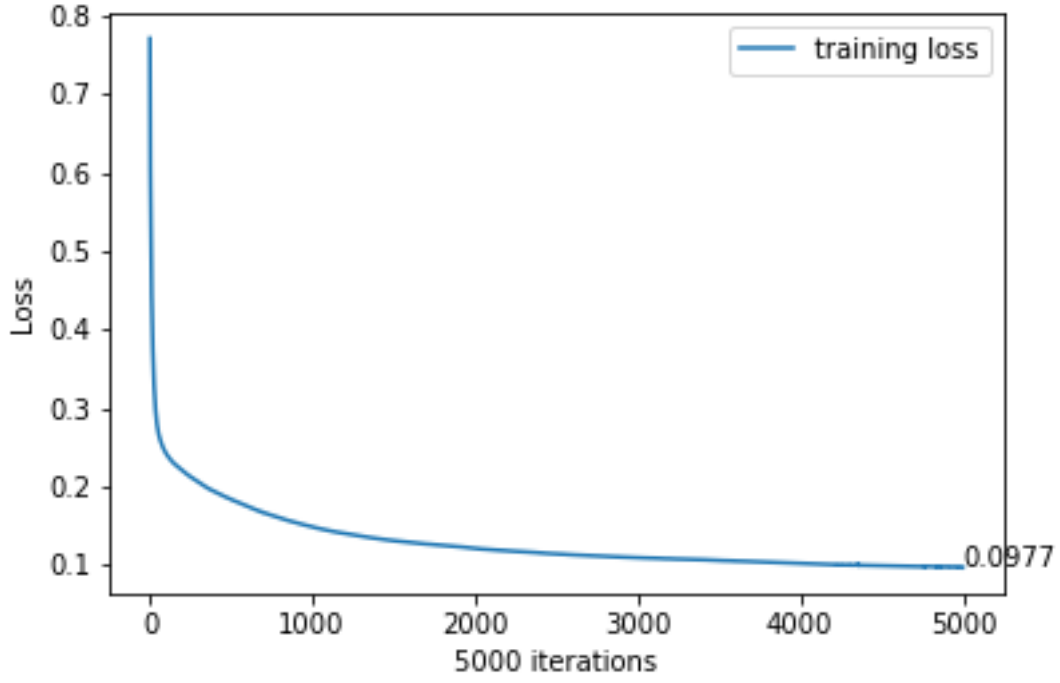

Figure 2.6 Training Loss for Batch Size 8

2.3.3 Question 3

a) Plot the cross-validation accuracies against the number of epochs for different number of hidden-layer neurons. Limit the search space of number of neurons to {5,10,15,20,25}.

Similar to Question 2, 5000 epochs would be used to perform the training for each number of hidden-layer neurons using the optimal batch size of 8 and 5-fold cross validation.

During the process of training, we plot the training loss, cross-validation and training accuracy of the model for different number of hidden-layer neurons at epochs 5000. Below are the results in a table format.

Refer to Appendix A Question 3 for the respective visual plots on cross-validation accuracies against the number epochs for different number of hidden-layer neurons.

Epochs	Fold	Training Accuracy (3 s.f.)	Cross-Validation Accuracy (3 s.f.)
5 Hidden-Layer Neurons			
5000	1	0.950	0.889
	2	0.962	0.906
	3	0.972	0.906
	4	0.960	0.940
	5	0.964	0.919
Mean		0.962	0.912
10 Hidden-Layer Neurons			
5000	1	0.959	0.892
	2	0.938	0.903
	3	0.953	0.919
	4	0.966	0.953
	5	0.965	0.916
Mean		0.956	0.917

15 Hidden-Layer Neurons			
5000	1	0.962	0.909
	2	0.958	0.899
	3	0.933	0.906
	4	0.953	0.936
	5	0.955	0.919
Mean		0.952	0.914
20 Hidden-Layer Neurons			
5000	1	0.968	0.916
	2	0.962	0.906
	3	0.951	0.892
	4	0.946	0.933
	5	0.940	0.909
Mean		0.953	0.911
25 Hidden-Layer Neurons			
5000	1	0.959	0.912
	2	0.967	0.919
	3	0.961	0.906
	4	0.959	0.930
	5	0.943	0.909
Mean		0.957	0.915

b) *Select the optimal number of neurons for the hidden layer. State the rationale for your selection.*

Using the model and results obtained in a), three graphs have been plotted showing the comparison between the training accuracy, cross-validation accuracy and training loss among the different number of hidden-layer neurons.

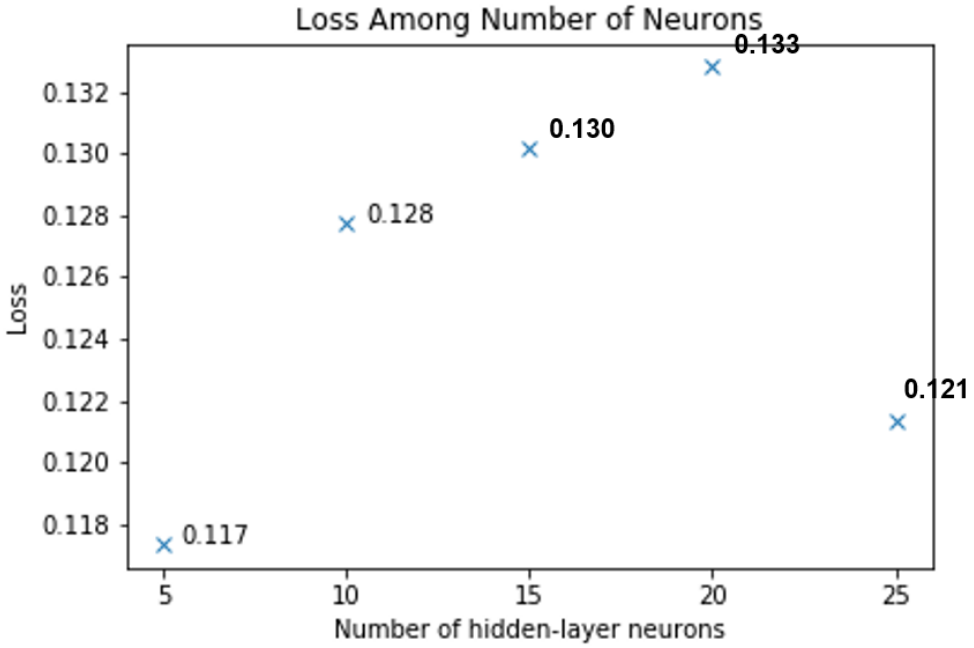

Figure 3.1 Training Loss Among Number of Neurons

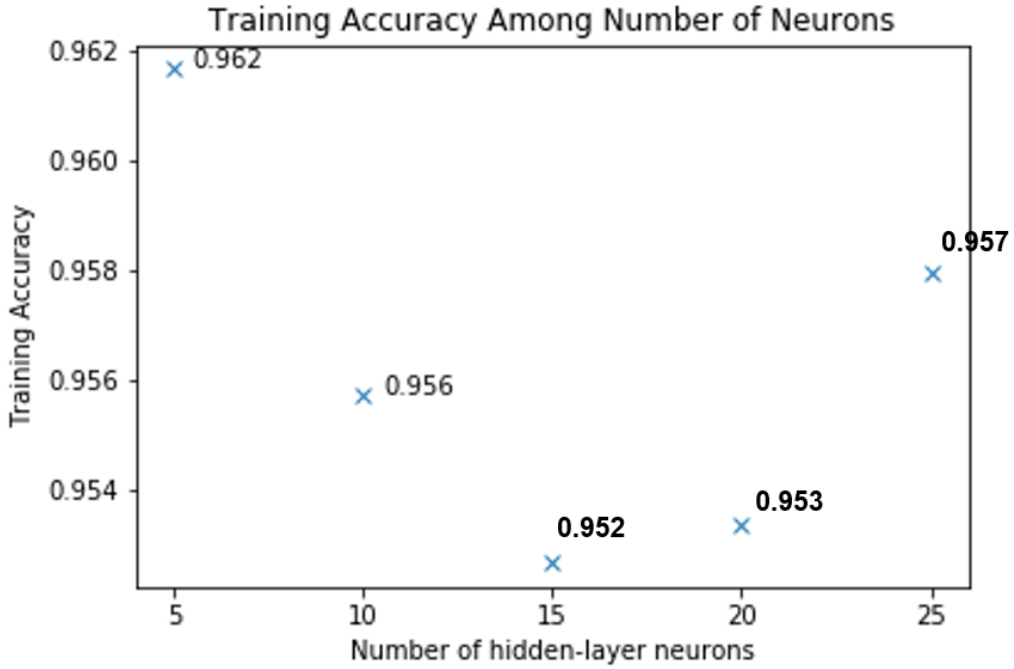

Figure 3.2 Training Accuracy Among Number of Neurons

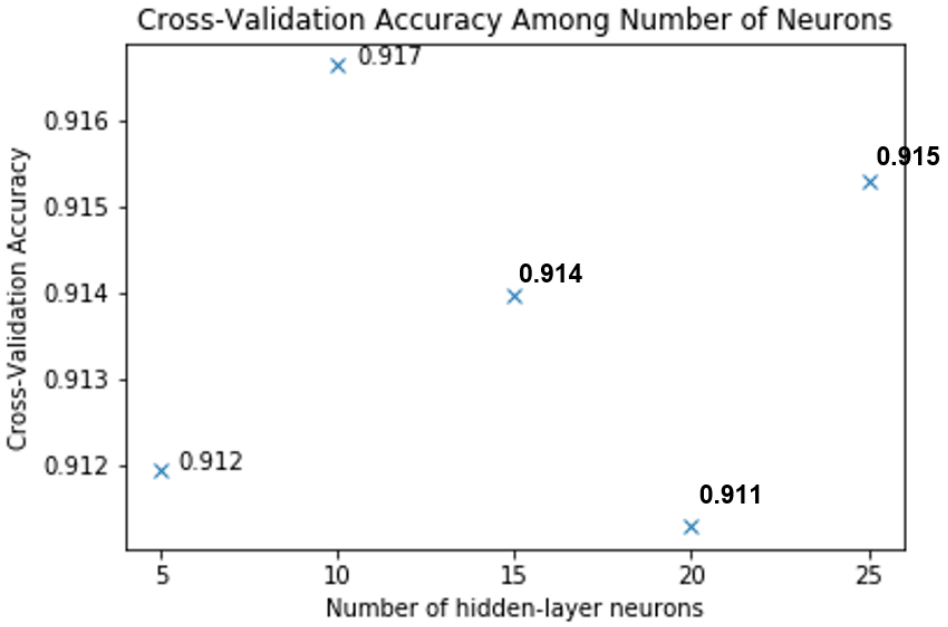

Figure 3.3 Cross-Validation Accuracy Among Number of Neurons

The information depicted by Figures 3.1, 3.2 and 3.3 are consolidated into the table below and ranked based on the optimal value that number of hidden-layer neurons can attain.

For training loss, the optimal would be the lowest value. For the training and cross-validation accuracy, the optimal would be the highest value.

Number of Neurons	**Training Accuracy**	**Training Loss**	**Cross-Validation Accuracy**
	Rankings		
5	1	1	4
10	3	3	1
15	5	4	3
20	4	5	5
25	2	2	2

Based on the table above, I would choose the **optimal number of hidden-layer neurons to be 25.** Based on the ranking itself, 5 neurons seem to be the optimal choice as it ranked first in the training accuracy and training loss. However, as for the

cross-validation accuracy, it ranked fourth which shows that the model might not have generalize well and could be overfitted.

Next, we look at 10 number of neurons, it ranked first for the cross-validation accuracy and third for both the training accuracy and loss. However, in our case, we are interested in using all three categories to determine which is the optimal number of neurons to be used.

As for 25 number of neurons, it ranked second on all categories hence our optimal choice.

To support the statement above, we take the training and cross-validation accuracies of 5 and 25 hidden-layer neurons.

Number of Neurons	**Cross-Validation Accuracy**	**Training Accuracy**	**Difference Between Accuracies**
5	0.912	0.962	0.05
10	0.917	0.956	0.039
25	0.915	0.957	0.042

As shown in the table, 10 number of neurons have the least difference between the cross-validation and training accuracy which comes to show that the model is generalize well. Next would be 25 number of neurons and coming in last is number of neurons.

The difference between the 10 and 25 number of neurons in this case is 0.003 which is a very small number and since we are using all three parameters, training and cross-validation accuracy and training loss, to gauge the optimal parameter, 25 would be our optimal choice.

c) *Plot the train and test accuracies against epochs with the optimal number of neurons.*

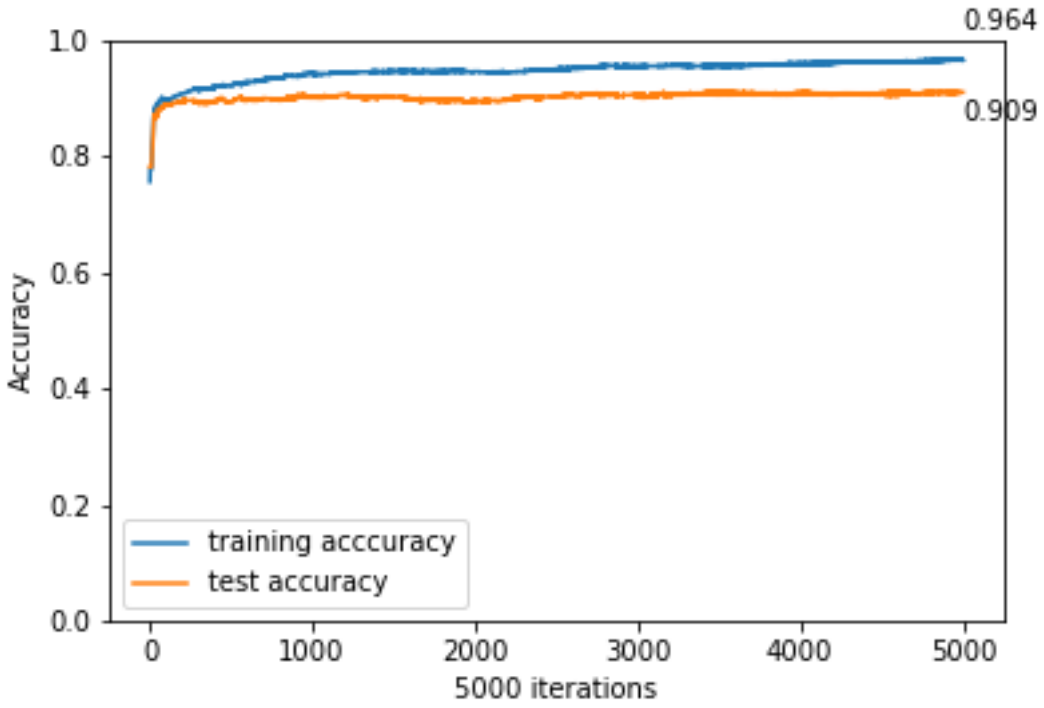

Figure 3.4 Train and Test Accuracies for 25 Hidden Layer Neurons with Batch Size 8

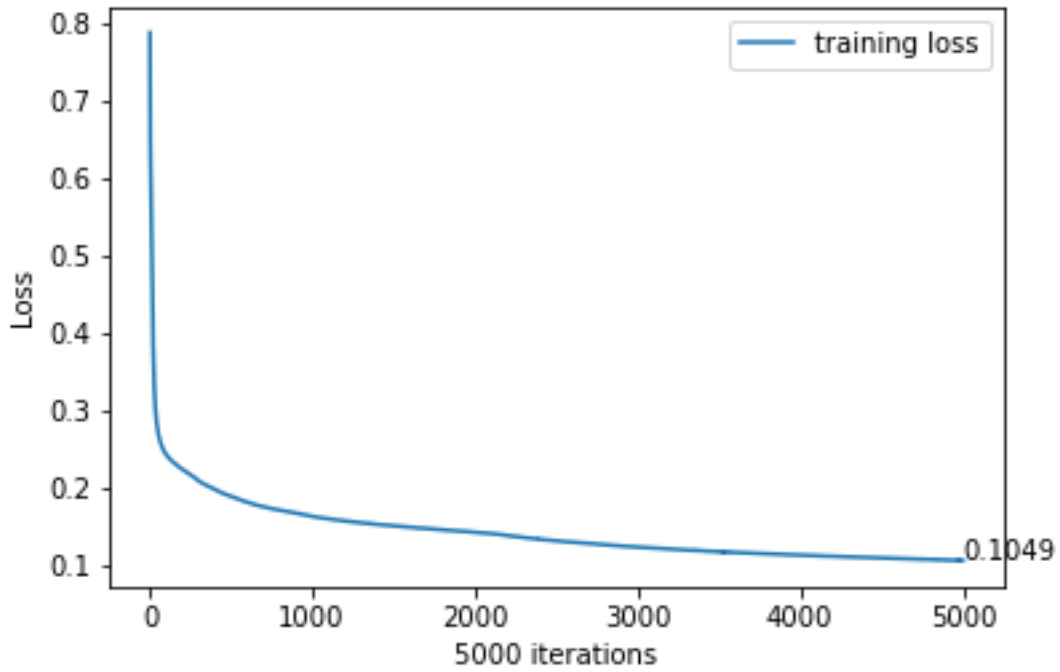

Figure 3.4 Train Loss for 25 Hidden Layer Neurons with Batch Size 8

2.3.4 Question 4

a) *Plot cross-validation accuracies against the number of epochs for the 3-layer network for different values of decay parameters. Limit the search space of decay parameters to {0,10−3,10−6,10−9,10−12}.*

Similar to Question 3, 5000 epochs would be used to perform the training for each values of decay parameter using the optimal 25 hidden-layer neurons, batch size of 8 and 5-fold cross validation.

During the process of training, we plot the training loss, cross-validation and training accuracy of the model for different number of hidden-layer neurons at epochs 5000. Below are the results in a table format.

Refer to Appendix A Question 4 for the respective visual plots on cross-validation accuracies against the number epochs for the different values of decay parameters.

Epochs	Fold	Training Accuracy (3 s.f.)	Cross-Validation Accuracy (3 s.f.)
Decay Parameter: 0			
5000	1	0.976	0.906
	2	0.931	0.916
	3	0.935	0.889
	4	0.971	0.936
	5	0.963	0.933
Mean		0.955	0.916
Decay Parameter: 10^-3			
5000	1	0.954	0.909
	2	0.964	0.909
	3	0.961	0.909
	4	0.965	0.933
	5	0.959	0.919
Mean		0.960	0.916

Decay Parameter: 10^-6			
5000	1	0.949	0.892
	2	0.973	0.916
	3	0.948	0.899
	4	0.951	0.936
	5	0.960	0.909
Mean		0.956	0.911
Decay Parameter: 10^-9			
5000	1	0.975	0.896
	2	0.950	0.919
	3	0.944	0.899
	4	0.957	0.913
	5	0.941	0.919
Mean		0.953	0.909
Decay Parameter: 10^-12			
5000	1	0.948	0.882
	2	0.964	0.913
	3	0.948	0.892
	4	0.938	0.923
	5	0.943	0.916
Mean		0.948	0.905

b) Select the optimal decay parameter. State the rationale for your selection.
Using the model and results obtained in a), three graphs have been plotted showing the comparison between the training accuracy, cross-validation accuracy and training loss among the different decay parameters.

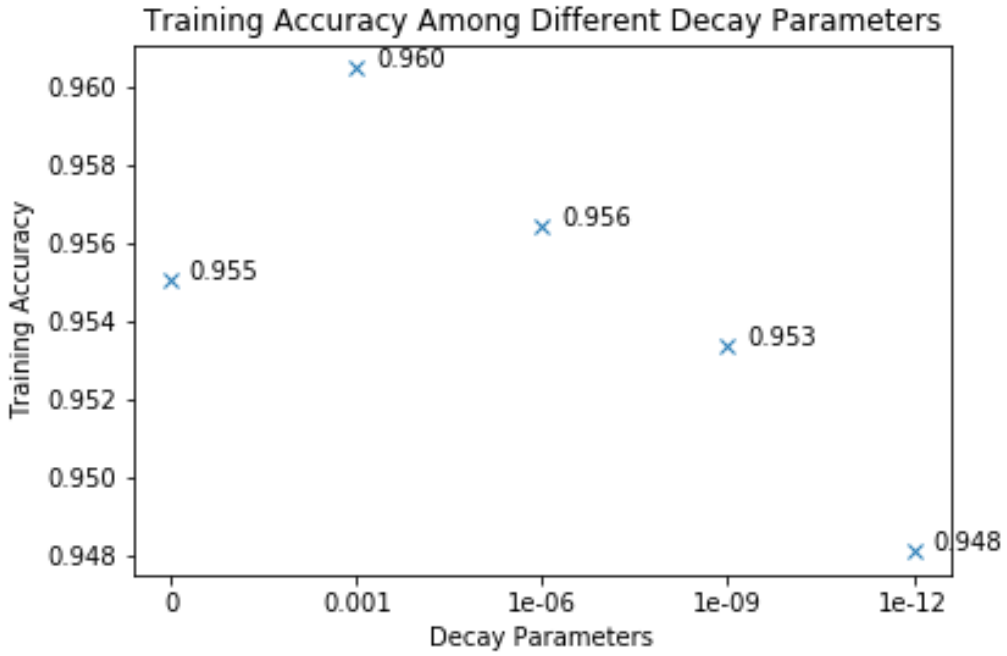

Figure 4.1 Training Accuracy Among Decay Parameters

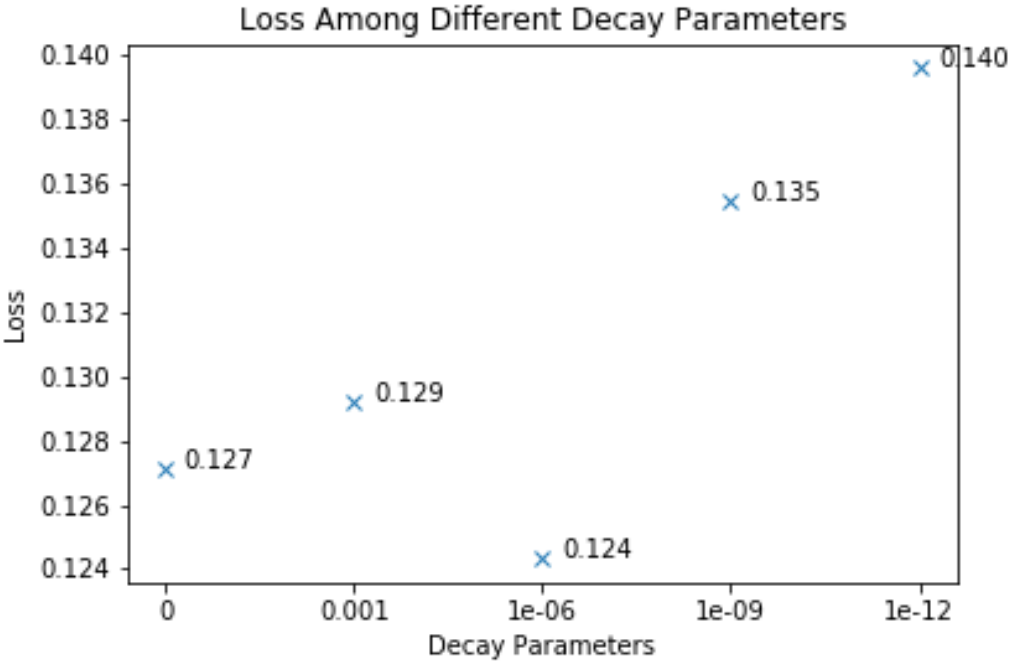

Figure 4.2 Training Loss Among Decay Parameters

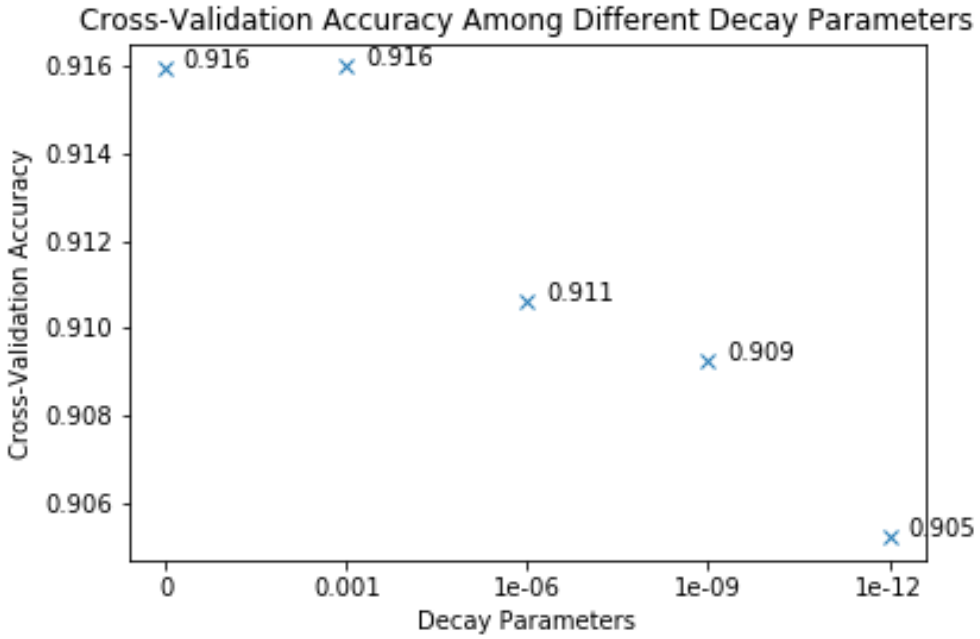

Figure 4.3 Cross-Validation Accuracy Among Decay Parameters

The information depicted by Figures 4.1, 4.2 and 4.3 are consolidated into the table below and ranked based on the optimal value that parameter of decay can attain.

For training loss, the optimal would be the lowest value. For the training and cross-validation accuracy, the optimal would be the highest value.

Decay Parameter	**Training Accuracy**	**Training Loss**	**Cross-Validation Accuracy**
	Rankings		
0	3	2	2
10^-3	1	3	1
10^-6	2	1	3
10^-9	4	4	4
10^-12	5	5	5

Based on the table above, our potential candidates are parameters 0, 10e-3 and 10e-6 as 10e-9 and10e-12 have very clear rankings showing that it will not be the choice of being an optimal parameter.

Hence, we shall plot the table showing the differences between the training and cross validation accuracies of the respective decay parameters and the differences in training loss among these parameters.

The table below shows the differences between the training and cross validation accuracy for the respective parameters.

Decay Parameters	**Cross-Validation Accuracy**	**Training Accuracy**	**Difference Between Accuracies**
0	0.91597	0.955	0.03903
10e-3	0.915985	0.960	0.044015
10e-6	0.911	0.956	0.045

Looking at the table above, the decay parameter of 0 would yield a better result than the rest as the difference between the cross-validation and training accuracy has the least difference. This signifies that model is much generalized and less underfitting compared to the other two parameters.

The table below shows the differences in training loss among the selected parameters.

Differences in Training Loss			
Decay Parameter	**0**	**10e-3**	**10e-6**
0		-0.002	0.003
10e-3	0.002		0.005
10e-6	-0.003	-0.005	

As for the table above, the largest loss difference is only 0.005 which is very insignificant and will not really affect the model's overall performance. Hence, the **most optimal decay parameter would be 0.**

c) *Plot the train and test accuracies against epochs for the optimal decay parameter.*

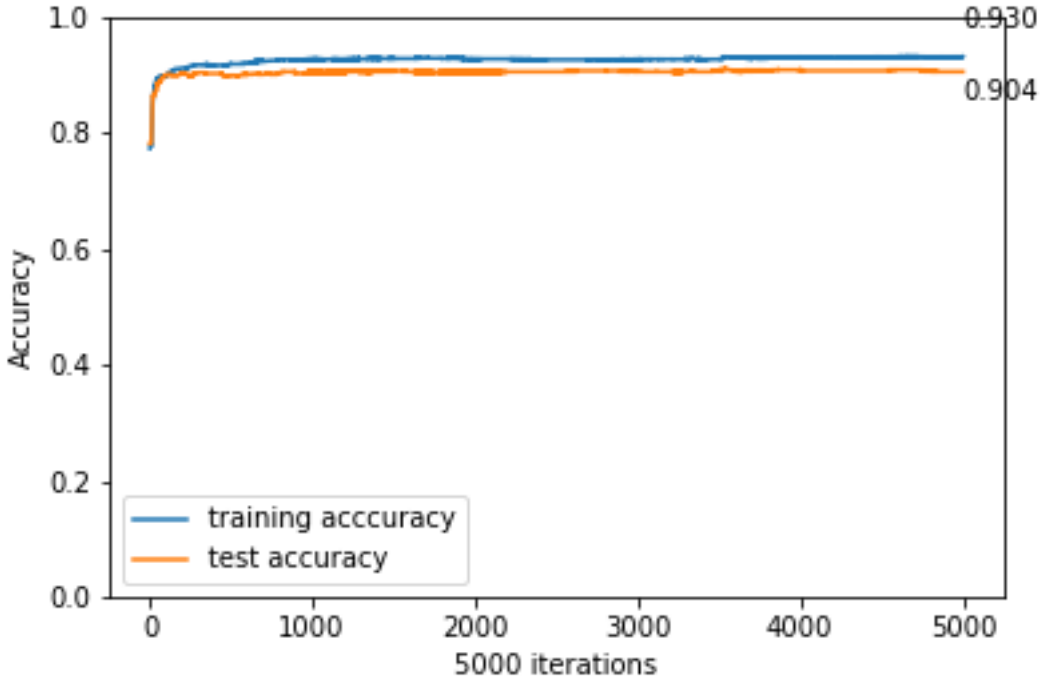

Figure 4.4 Train and Test Accuracies for Decay Parameter 0 of 25 Hidden Layer Neurons with Batch Size 8

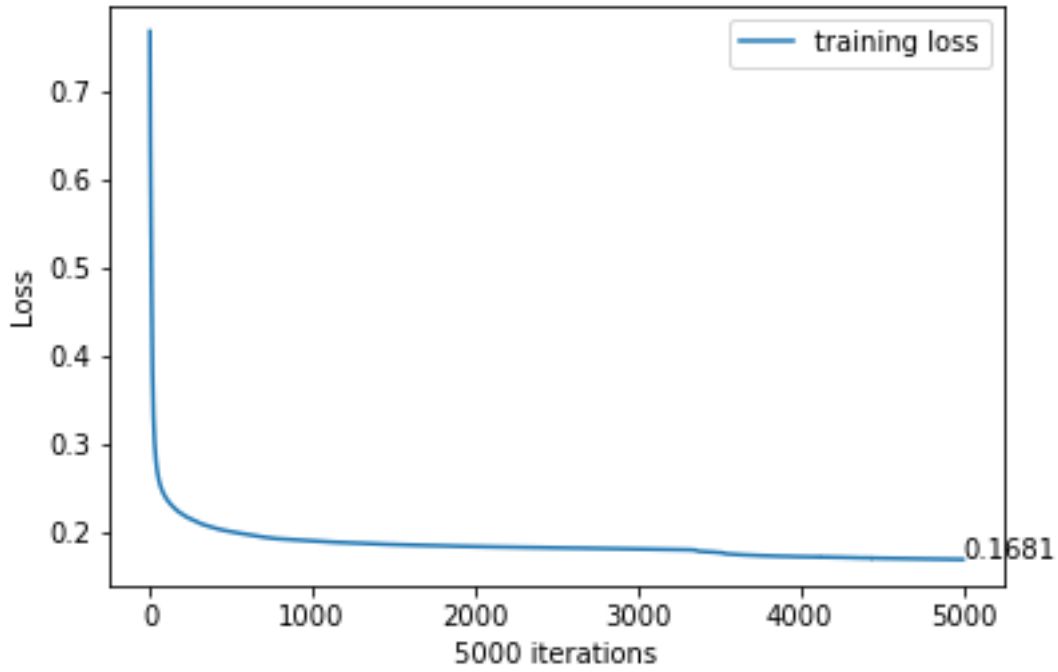

Figure 4.5 Training Loss for Decay Parameter 0 of 25 Hidden Layer Neurons with Batch Size 8

2.3.5 Question 5

a) Plot the train and test accuracy of the 4-layer network.
Similar to question 1, a total of 10,000 epochs of training have been used to perform the training process.

During the process of training, a milestone of 1000 epochs is used to plot the train accuracy, test accuracy and loss of the model. Below are the results of every 1000 epochs in a table format.

<u>Refer to Appendix A Question 5 for the respective visual plots on both accuracies on training and testing data against epochs.</u>

Epochs	Training Loss (4 s.f.)	Training Accuracy (3 s.f.)	Test Accuracy (3 s.f.)
1,000	0.2110	0.908	0.895
2,000	0.1874	0.918	0.887
3,000	0.1691	0.929	0.904
4,000	0.1570	0.936	0.911
5,000	0.1494	0.940	0.904
6,000	0.1430	0.942	0.908
7,000	0.1356	0.946	0.911
8,000	0.1303	0.948	0.909
9,000	0.1264	0.949	0.909
10,000	0.1234	0.947	0.909

Next, the table below shows the difference between the latest number epochs against its previous number of epochs recorded.

Epochs	Training Loss Compared to Previous (4 s.f.)	Training Accuracy Compared to Previous (3 s.f.)	Test Accuracy Compared to Previous (3 s.f.)
1,000	-	-	-
2,000	-0.0236	0.01	-0.008
3,000	-0.0183	0.011	0.017
4,000	-0.0121	0.007	0.007
5,000	-0.0076	0.004	-0.007
6,000	-0.0064	0.002	0.004
7,000	-0.0074	0.004	0.003
8,000	-0.0053	0.002	-0.002
9,000	-0.0039	0.001	0
10,000	-0.003	-0.002	0

b) Compare and comment on the performances of the optimal 3-layer and 4-layer networks.

Below is the training, test accuracies as well as the training loss for the 4-layer network with optimal values such as a decay parameter of 0, 25 neurons inside two of the hidden layers and batch size of 8.

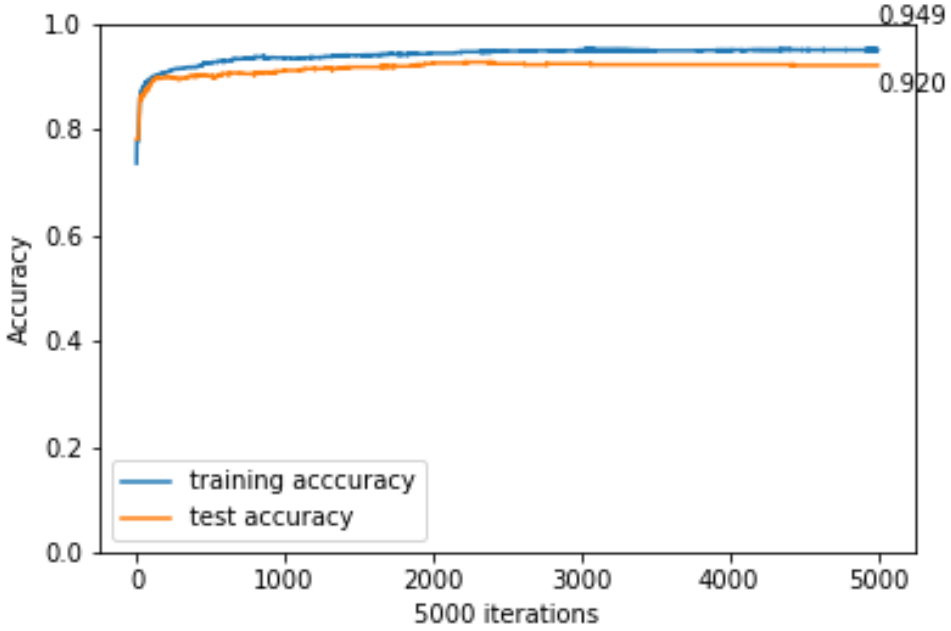

Figure 5.1 Training and Test Accuracies for Optimal 4-Layer Network

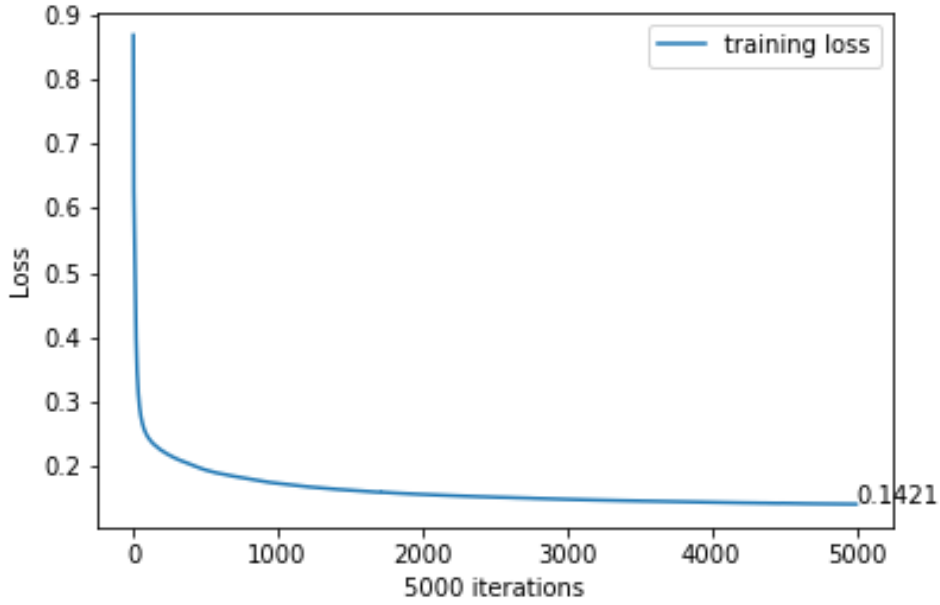

Figure 5.1 Training Loss for Optimal 4-Layer Network

To compare the optimal 3-layer and 4-layer networks, we plot the difference in training loss, training accuracy and test accuracy between the two in a table form.

As seen in 5a), the difference in loss is much stable than the 3-layer neural network with convergence occurring at around 5,000 epochs as the loss is less than 0.01.

Next, we will use the values obtained at 5,000 epochs to compare the two models.

The numbers shown in the final row is the result of taking the values from 3-layer network (from question 4c) and subtracting them to the 4-layer network.

Network	**Training Loss (4 s.f.)**	**Training Accuracy (3 s.f.)**	**Test Accuracy (3 s.f.)**
	3-Layer Network's Value – 4 Layer Network's Value		
3 – Layer Optimal Network (5000 epochs)	0.1681	0.930	0.904
4 – Layer Optimal Network (5000 epochs)	0.1421	0.949	0.920
Difference	0.026	-0.019	-0.016

As seen above, 4-Layer optimal network produces a better result with a lower loss and higher training accuracy and test accuracy. However, if we were to look at each individual generalization score.

- 3-Layer Optimal Network: 0.930 – 0.904 = 0.026
- 4-Layer Optimal Network: 0.949 – 0.920 = 0.029

3-Layer performs slightly better than the 4-Layer by 0.003 accuracy. Given the difference between the training accuracy and test accuracy shown in the table, the difference in generalization score is quite insignificant.

Hence, **4-Layer optimal network performs better than the 3-Layer optimal network** based on having a lower loss, and higher train and test accuracies.

However, the training set provided to the model is around 1488 records which is a very small amount. We believe that if more records were to be used to train the model, a much significant difference would incur and a decision between time needed to train and the accuracy required would have to be chosen.

3 Part B: Regression Problem

3.1 Introduction

The aim is to build neural networks using 400 data from the Graduate Admissions Predication dataset to predict the chances of getting an admit into Master Programs.

The dataset consists of several parameters such as Graduate Record Examinations (GRE) score (out of 340), TOEFL score (out of 120), University Rating (out of 5), Strengths of Statement of Purpose and Letter of Recommendation (LOR) (out of 5), undergraduate GPA (out of 10), research experience (either 0 or 1) which will be used as our input features. The resulting model should be able to predict a chance ranging from 0 to 1.

Each data sample consists of 9 features: 1 serial number (which will be ignored), 7 input attributes and the probability of getting an admit as targets.

3.2 Methods

Data Pre-processing

Input normalization is first performed. The theoretical two-step process for this is:

1. Scale the inputs such that they lie in the range [0, 1]

$$x_i = \frac{x_i - x_{i,min}}{x_{i,max} - x_{i,min}}$$

2. Normalize the inputs so that it can have a standard normal distribution:

$$normalized\_x_i = \frac{x_i - \mu_i}{\sigma_i}$$

In practice, this is implemented in the scikit-learn library using the preprocessing module.

```
scaler = preprocessing.StandardScaler()

trainX = scaler.fit_transform(trainX)

testX = scaler.fit_transform(testX)
```

Truncated normal distribution weights initialization

A truncated normal distribution was used to initialize the weights in the model because it keeps the weights close to zero, hence it can operate in the linear region of the activation function, which is especially useful for neurons using Sigmoid or TanH as their activation functions.

L2 Regularization

Regularization refers to the penalizing of the cost function to prevent some weights from attaining large values, which will reduce training error but results in overfitting. L2 uses the squared value of each weight, thus penalizing values away from the mean in a nonlinear fashion.

Regularization is not applied on biases.

Dropout

The rationale of dropout is to avoid overfitting by only training a fraction of weights in each iteration. So, the core concept is to randomly pick neurons to drop from the network during training. This prevents neurons from co-adapting and thus reduces overfitting. However, it should only be applied during training and not on test data.

Mini-batch gradient descent

The idea behind mini-batch gradient descent is to utilize the more efficient vector operations built into most modern processors today by passing through a ‘mini-batch’ of input data to be processed in parallel, thus speeding up training time. The optimal batch size is strongly correlated to the size of the cache.

Model

To build a 3-layer forward neural network, we use the following parameters:

- All weights are initialized from a truncated normal distribution.
- All biases are set to zero.
- Hidden layer consists of ReLU units.
- Output layer is a linear layer for regression.
- Mean squared error is used as a loss function.
- L2 regularization with decay is applied on the loss.
    - Weight Decay (β) * Σ sqt(weights)/2
- Mini batch gradient descent learning used for training.

The initial model architecture and hyper-parameters are as below:

Layers	Shape	Activation
Input	(8, 7)	-
Hidden	(7, 10)	ReLU
Output	(10, 1)	linear
Learning Rate, α =0.001 Batch size = 8 Number of hidden neurons = 10 Weight decay parameter β = 1e-3		

3.3 Experiment & Results

3.3.1 Question 1

a) Use the train dataset to train the model and plot both the train and test errors against epochs.

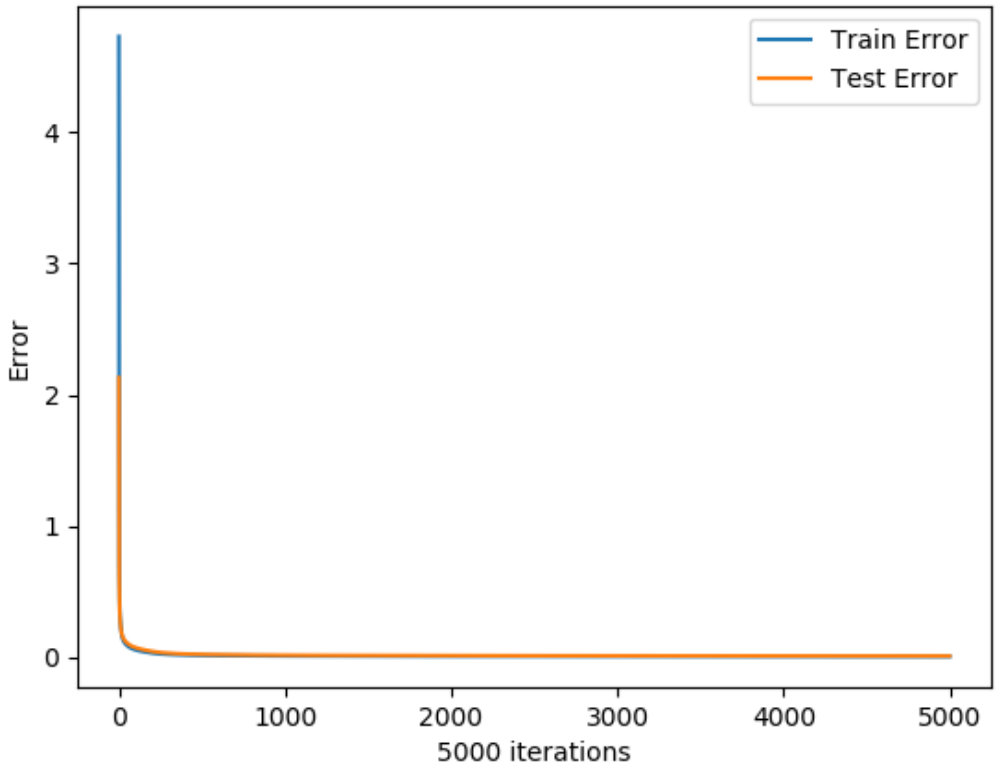

Figure B1. The plot of the train error against test error during training.

b) State the approximate number of epochs where the test error is minimum and use it to stop training.

Epochs	Train Loss	Test Loss	Percentage Decrease In Train Loss	Percentage Decrease in Test Loss
1	4.73095	2.13108	-	-
100	0.0562954	0.073875	98.81%	96.53%
200	0.0317686	0.0448001	43.57%	39.36%
300	0.0210815	0.031303	33.64%	30.13%
400	0.0166757	0.0248407	20.90%	20.64%
500	0.0143022	0.0217213	14.23%	12.56%
600	0.0128963	0.019722	9.83%	9.20%
700	0.0118681	0.0181499	7.97%	7.97%
800	0.0110352	0.0169118	7.02%	6.82%
900	0.010376	0.0160294	5.97%	5.22%
1000	0.0098839	0.0153544	4.74%	4.21%
1100	0.00946596	0.0148925	4.23%	3.01%
1200	0.00909783	0.0144722	3.89%	2.82%
1300	0.00876203	0.0140987	3.69%	2.58%
1400	0.00844779	0.0137714	3.59%	2.32%
1500	0.00819505	0.013475	2.99%	2.15%
1600	0.00797981	0.0132206	2.63%	1.89%
1700	0.00775663	0.0128553	2.80%	2.76%
1800	0.00756245	0.0125285	2.50%	2.54%
1900	0.00740206	0.0123255	2.12%	1.62%
2000	0.00726042	0.0121644	1.91%	1.31%
2100	0.00713171	0.0120649	1.77%	0.82%
2200	0.00699995	0.0119944	1.85%	0.58%

2300	0.00687361	0.0118397	1.80%	1.29%
2400	0.00673707	0.0115869	1.99%	2.14%
2500	0.00662605	0.0114215	1.65%	1.43%
2600	0.00653202	0.0112855	1.42%	1.19%
2700	0.00644288	0.0111378	1.36%	1.31%
2800	0.00635109	0.0110008	1.42%	1.23%
2900	0.00626097	0.0108629	1.42%	1.25%
3000	0.00617998	0.010739	1.29%	1.14%
3100	0.00609555	0.0106253	1.37%	1.06%
3200	0.00601903	0.010551	1.26%	0.70%
3300	0.00594601	0.0104651	1.21%	0.81%
3400	0.00587542	0.0103764	1.19%	0.85%
3500	0.00581252	0.0102853	1.07%	0.88%
3600	0.0057542	0.0101991	1.00%	0.84%
3700	0.0057029	0.0101336	0.89%	0.64%
3800	0.00565673	0.0100666	0.81%	0.66%
3900	0.00561381	0.00999722	0.76%	0.69%
4000	0.00557319	0.00992722	0.72%	0.70%
4100	0.0055348	0.00986469	0.69%	0.63%
4200	0.00549894	0.00980806	0.65%	0.57%
4300	0.0054651	0.0097534	0.62%	0.56%
4400	0.00543339	0.00970221	0.58%	0.52%
4500	0.00540193	0.00965576	0.58%	0.48%
4600	0.00537177	0.00960913	0.56%	0.48%
4700	0.00534209	0.00956795	0.55%	0.43%
4800	0.00531167	0.00950243	0.57%	0.68%

4900	0.00528268	0.00945387	0.55%	0.51%
5000	0.00525459	0.00941233	0.53%	0.44%

Table B2. The data used to plot Figure B1, with the percentage decrease in loss compared to 100 epochs ago.

The cutoff point is taken to be the point where test loss either starts to diverge or converges to a statistically insignificant point. Based on the table above, anything after epoch 3100 will not make test error decrease by more than 1% from its previous error for subsequent epochs and therefore the number of epochs at which to stop training should be 3100.

c) Plot the predicted values and target values for any 50 test samples.

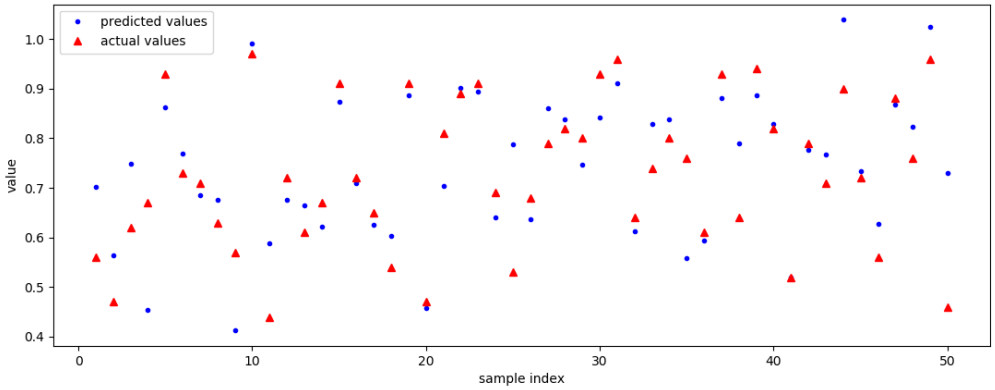

Figure B3. The plot of a random set of 50 predicted values against the actual values. [Mean-Square-Error of this sample = 0.00901713]

3.3.2 Question 2

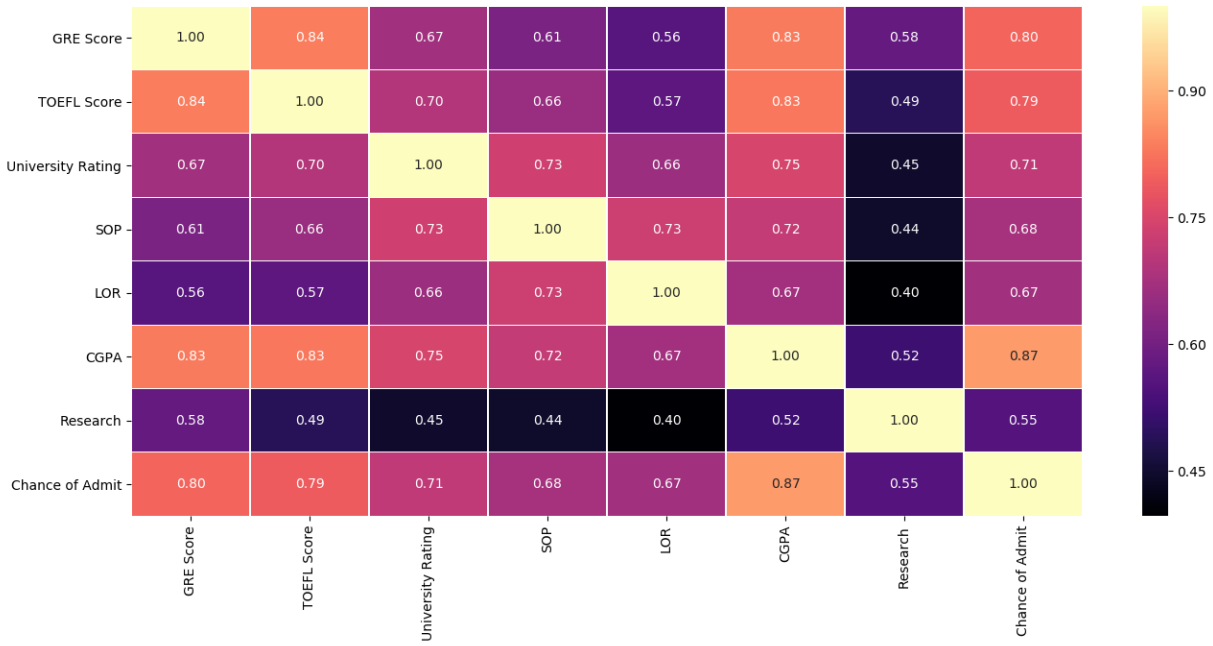

Figure B4. The correlation matrix (excluding serial number) of the dataset

a) Which features are most correlated to each other? Is it justifiable?

Based on the correlation matrix, chance of admission has the highest correlation with the applicant's undegraduate CGPA with a Pearson's Product-Moment Correlation (PPMC) score of 0.87. This is followed by the applicant's Graduate Record Examinations (GRE) score and TOEFL score, which have a PPMC score of 0.84. Finally, the correlation between the applicant's CGPA and their GRE score, and also between their CGPA and TOEFL score are both tied at 0.83.

In summary, the top four correlations are listed below with a possible reasoning behind them:

1. CGPA and Chance of Admission (0.87): the higher an applicant's undegraduate CGPA, the more likely they are perceived to have the cognitive ability and mental fortitude to handle a Masters' program.
1. TOEFL score and GRE score (0.84): the GRE is offered in English, and a higher TOEFL score usually equates to a higher standard of English by the applicant. Therefore, this means they will face less language issues when studying for and taking the GRE.
2. CGPA and GRE score (0.83): an applicant's CGPA shows how exam-smart they are, and the GRE score is used

3. CGPA and TOEFL score (0.83): like the reasoning above, being good at studying usually means that one will excel in an exam.

b) What features have the highest correlations with the chances of admit?

Also, based on the correlation matrix, the top three features (with the PPMC score in brackets) with the highest correlations with the chances of admit are CGPA (0.87), GRE Score (0.80), and TOEFL Score (0.79).

3.3.3 Question 3

a) Compare the accuracy of the model with all input features, with models using 6 input features and 5 input features selected using RFE. Comment on the observations.

Based on the correlation matrix, the first feature to be removed (with the PPMC score in brackets) is whether an applicant has research experience (0.55), and the second feature is the strength of their LOR (0.67).

To compare the accuracy of the model training, both models are trained for 5000 epochs, a valid cut-off number of epochs chosen for each and then their test loss will be compared at those said cut-offs.

Epochs	Train Loss	Test Loss	Percentage Decrease In Train Loss	Percentage Decrease in Test Loss
1	5.1436	2.5276	-	-
100	0.0663	0.0838	98.71%	96.69%
200	0.0411	0.0532	37.96%	36.48%
300	0.0282	0.0385	31.45%	27.67%
400	0.0195	0.0286	30.68%	25.63%
500	0.014	0.0227	28.36%	20.77%
600	0.0106	0.0192	24.43%	15.52%
700	0.0082	0.0162	22.12%	15.30%
800	0.0068	0.0142	17.24%	12.65%
900	0.006	0.0126	12.48%	10.99%
1000	0.0054	0.0115	8.78%	8.47%
1100	0.005	0.0107	8.02%	7.43%
1200	0.0047	0.0101	6.02%	5.62%
1300	0.0045	0.0096	4.67%	4.82%
1400	0.0043	0.0093	3.75%	3.25%
1500	0.0042	0.009	3.11%	3.17%

1600	0.0041	0.0088	2.62%	2.62%
1700	0.004	0.0086	2.41%	2.29%
1800	0.0039	0.0083	2.52%	2.51%
1900	0.0038	0.0081	2.30%	2.51%
2000	0.0037	0.008	2.41%	1.83%
2100	0.0036	0.0078	2.32%	2.06%
2200	0.0035	0.0076	2.05%	2.29%
2300	0.0035	0.0075	1.81%	2.04%
2400	0.0034	0.0074	1.64%	1.75%
2500	0.0034	0.0072	1.57%	1.57%
2600	0.0033	0.0071	1.51%	1.54%
2700	0.0033	0.007	1.69%	1.84%
2800	0.0032	0.0069	1.49%	1.61%
2900	0.0032	0.0068	1.38%	1.84%
3000	0.0031	0.0067	1.31%	1.50%
3100	0.0031	0.0066	1.09%	1.24%
3200	0.0031	0.0065	1.04%	1.16%
3300	0.003	0.0064	0.92%	1.08%
3400	0.003	0.0064	0.79%	1.02%
3500	0.003	0.0063	0.71%	0.95%
3600	0.003	0.0062	0.63%	0.91%
3700	0.0029	0.0062	0.63%	0.85%
3800	0.0029	0.0061	0.55%	0.74%
3900	0.0029	0.0061	0.57%	0.74%
4000	0.0029	0.0061	0.57%	0.76%
4100	0.0029	0.006	0.52%	0.75%

4200	0.0029	0.006	0.51%	0.70%
4300	0.0028	0.0059	0.52%	0.64%
4400	0.0028	0.0059	0.43%	0.62%
4500	0.0028	0.0059	0.46%	0.61%
4600	0.0028	0.0058	0.47%	0.74%
4700	0.0028	0.0058	0.42%	0.74%
4800	0.0028	0.0057	0.40%	0.68%
4900	0.0028	0.0057	0.36%	0.66%
5000	0.0028	0.0057	0.39%	0.58%

Table B5. The training and test loss at each 100 epochs for the model trained using 6 features

Epochs	Train Loss	Test Loss	Percentage Decrease In Train Loss	Percentage Decrease in Test Loss
1	5.49292	2.65363	-	-
100	0.080883	0.0810181	98.53%	96.95%
200	0.0442012	0.0431512	45.35%	46.74%
300	0.0286326	0.0269642	35.22%	37.51%
400	0.0199027	0.0193772	30.49%	28.14%
500	0.0151312	0.01514	23.97%	21.87%
600	0.0122646	0.0127803	18.94%	15.59%
700	0.0103244	0.0113898	15.82%	10.88%
800	0.00891954	0.0105363	13.61%	7.49%
900	0.00794507	0.00977986	10.93%	7.18%
1000	0.00727994	0.00921169	8.37%	5.81%

1100	0.00676772	0.00883401	7.04%	4.10%
1200	0.00636781	0.0085208	5.91%	3.55%
1300	0.00604796	0.00825248	5.02%	3.15%
1400	0.00578862	0.00801732	4.29%	2.85%
1500	0.00557217	0.00784441	3.74%	2.16%
1600	0.00538102	0.00769742	3.43%	1.87%
1700	0.00520611	0.00753255	3.25%	2.14%
1800	0.00504738	0.00738653	3.05%	1.94%
1900	0.0049082	0.0072714	2.76%	1.56%
2000	0.00477767	0.00716266	2.66%	1.50%
2100	0.00465787	0.0070468	2.51%	1.62%
2200	0.00455043	0.0069379	2.31%	1.55%
2300	0.00444986	0.00685153	2.21%	1.24%
2400	0.00434375	0.00675474	2.38%	1.41%
2500	0.0042513	0.00669563	2.13%	0.88%
2600	0.00416483	0.006635	2.03%	0.91%
2700	0.00409089	0.00658037	1.78%	0.82%
2800	0.00402497	0.00652735	1.61%	0.81%
2900	0.00396491	0.0064928	1.49%	0.53%
3000	0.00391013	0.00645845	1.38%	0.53%
3100	0.00386049	0.00642398	1.27%	0.53%
3200	0.0038142	0.00639021	1.20%	0.53%
3300	0.00376873	0.00635758	1.19%	0.51%
3400	0.00371961	0.00631769	1.30%	0.63%
3500	0.00367482	0.00627636	1.20%	0.65%
3600	0.00363324	0.00623028	1.13%	0.73%

3700	0.00359282	0.0061779	1.11%	0.84%
3800	0.00355429	0.00612311	1.07%	0.89%
3900	0.00351924	0.00607708	0.99%	0.75%
4000	0.00348594	0.00603159	0.95%	0.75%
4100	0.0034552	0.00598725	0.88%	0.74%
4200	0.00342659	0.00594563	0.83%	0.70%
4300	0.00339852	0.00589685	0.82%	0.82%
4400	0.00337132	0.0058599	0.80%	0.63%
4500	0.00334373	0.00582944	0.82%	0.52%
4600	0.00331769	0.00580369	0.78%	0.44%
4700	0.00329343	0.00577566	0.73%	0.48%
4800	0.00327028	0.00574976	0.70%	0.45%
4900	0.00324807	0.00572285	0.68%	0.47%
5000	0.00322716	0.0056931	0.64%	0.52%

Table B6. The training and test loss at each 100 epochs for the model trained using 5 features

Based on tables B5 and B6 and based on the heuristic stated in the answer for question 1, the number of epochs to cut training at for the model trained using 6 features should be 3400, which gives us a test loss of 0.0064. On the other hand, the number of epochs to cut training at for the model trained using 5 features should be 2400, which gives us a test loss of 0.00675474.

This could be due to there being too little features to train on if there are only 5 features.

Therefore, the optimum number of features should be 6.

3.3.4 Question 4

a) Compare the performances of all the networks (with and without dropouts) with each other and with the 3-layer network.

To compare the performance of all the networks, I will use the heuristic stated in question 1 to determine their respective cut-off epochs, then compare their test losses at that point in time. Also, 6 features will be used as it was the ideal number chosen in question 3.

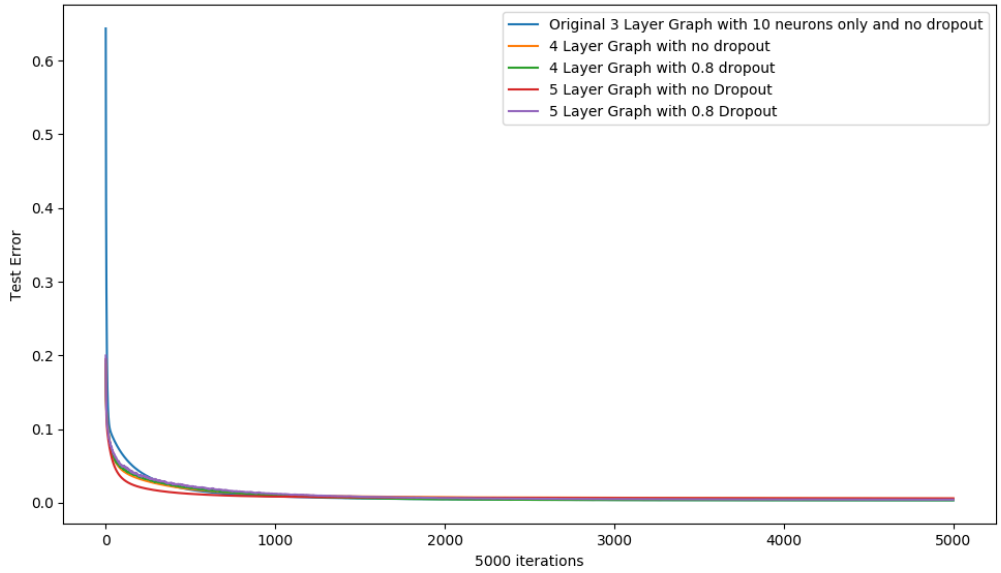

Figure B7. The test loss from each graph

Epochs	Train Loss	Test Loss	Percentage Decrease In Train Loss	Percentage Decrease in Test Loss
1	0.2851	0.1927	-	-
100	0.0637	0.0477	77.67%	75.24%
200	0.0489	0.0348	23.21%	27.09%
300	0.0334	0.0287	31.65%	17.61%
400	0.0263	0.0237	21.16%	17.46%
500	0.0263	0.0201	0.01%	14.90%
600	0.0217	0.0175	17.43%	13.19%
700	0.0181	0.0149	16.75%	14.61%
800	0.0167	0.0135	7.71%	9.75%
900	0.0158	0.0114	5.53%	15.72%
1000	0.0124	0.0104	21.57%	8.45%
1100	0.0131	0.0091	-5.91%	12.48%
1200	0.0104	0.0085	20.69%	7.01%
1300	0.0103	0.0076	0.53%	9.86%
1400	0.0103	0.007	0.07%	8.08%
1500	0.0085	0.0066	18.08%	5.86%
1600	0.0084	0.0061	0.81%	7.11%
1700	0.0073	0.0057	12.69%	7.47%
1800	0.0069	0.0054	5.77%	5.00%
1900	0.007	0.0051	-1.81%	6.07%
2000	0.0066	0.0048	6.35%	5.31%
2100	0.0071	0.0047	-7.40%	2.12%

2200	0.007	0.0046	1.11%	2.65%
2300	0.0062	0.0044	11.86%	3.70%
2400	0.0064	0.0043	-3.48%	3.22%
2500	0.0059	0.0042	7.49%	2.14%
2600	0.0057	0.0041	3.38%	2.11%
2700	0.0054	0.004	5.52%	2.05%
2800	0.0053	0.0039	2.40%	1.45%
2900	0.0062	0.0039	-16.93%	1.79%
3000	0.0053	0.0038	14.46%	0.92%
3100	0.005	0.0038	4.39%	1.23%
3200	0.0049	0.0037	3.30%	0.88%
3300	0.0051	0.0037	-4.25%	-0.01%
3400	0.0053	0.0037	-3.91%	1.04%
3500	0.0053	0.0037	-0.03%	0.77%
3600	0.0052	0.0036	1.27%	1.01%
3700	0.0051	0.0036	1.71%	0.77%
3800	0.0055	0.0036	-6.68%	-0.60%
3900	0.0052	0.0036	5.41%	0.65%
4000	0.0051	0.0036	0.93%	-0.30%
4100	0.0052	0.0036	-2.55%	0.32%
4200	0.0051	0.0036	3.26%	0.74%
4300	0.0044	0.0036	13.28%	-0.34%
4400	0.0047	0.0036	-6.55%	0.84%
4500	0.0046	0.0036	0.91%	-0.39%
4600	0.0045	0.0036	4.05%	0.64%
4700	0.0048	0.0036	-8.62%	0.13%

4800	0.0046	0.0036	4.14%	-0.49%
4900	0.0048	0.0035	-4.31%	0.74%
5000	0.0046	0.0036	4.08%	-0.65%

Table B8. The training and test mean square error at each 100 epochs for the 4-layer model with dropout, trained using 6 features with the selected cutoff epoch highlighted

Epochs	Train Loss	Test Loss	Percentage Decrease In Train Loss	Percentage Decrease in Test Loss
1	0.227859	0.187302	-	-
100	0.0408605	0.0439018	82.07%	76.56%
200	0.029132	0.0325148	28.70%	25.94%
300	0.0226533	0.026257	22.24%	19.25%
400	0.0182987	0.0218552	19.22%	16.76%
500	0.0151314	0.0187187	17.31%	14.35%
600	0.0127368	0.0163761	15.83%	12.51%
700	0.0108991	0.0145922	14.43%	10.89%
800	0.00945697	0.0132042	13.23%	9.51%
900	0.00831417	0.012116	12.08%	8.24%
1000	0.0073901	0.0112364	11.11%	7.26%
1100	0.00663414	0.0105199	10.23%	6.38%
1200	0.00601492	0.00993611	9.33%	5.55%
1300	0.00550826	0.0094437	8.42%	4.96%
1400	0.00508513	0.00904146	7.68%	4.26%
1500	0.00473254	0.00871403	6.93%	3.62%
1600	0.00443229	0.00844086	6.34%	3.13%
1700	0.00417349	0.00820212	5.84%	2.83%
1800	0.00395085	0.00799659	5.33%	2.51%
1900	0.00375746	0.00781966	4.89%	2.21%
2000	0.00358671	0.00766686	4.54%	1.95%
2100	0.00343321	0.00753104	4.28%	1.77%
2200	0.00329298	0.00740957	4.08%	1.61%

2300	0.0031687	0.0072973	3.77%	1.52%
2400	0.00305551	0.00719832	3.57%	1.36%
2500	0.00295135	0.00710383	3.41%	1.31%
2600	0.00285733	0.00702728	3.19%	1.08%
2700	0.00277162	0.00695546	3.00%	1.02%
2800	0.00269419	0.00688945	2.79%	0.95%
2900	0.00262321	0.00682732	2.63%	0.90%
3000	0.00255803	0.00676665	2.48%	0.89%
3100	0.00249742	0.0067008	2.37%	0.97%
3200	0.00244065	0.00663313	2.27%	1.01%
3300	0.0023871	0.00657121	2.19%	0.93%
3400	0.00233842	0.00651358	2.04%	0.88%
3500	0.00229276	0.00646179	1.95%	0.80%
3600	0.0022509	0.00641518	1.83%	0.72%
3700	0.00221111	0.00637303	1.77%	0.66%
3800	0.0021735	0.00633187	1.70%	0.65%
3900	0.0021378	0.006291	1.64%	0.65%
4000	0.00210381	0.00625441	1.59%	0.58%
4100	0.00207124	0.00621774	1.55%	0.59%
4200	0.00203921	0.00617864	1.55%	0.63%
4300	0.00200952	0.00613764	1.46%	0.66%
4400	0.0019813	0.00609898	1.40%	0.63%
4500	0.00195356	0.00605784	1.40%	0.67%
4600	0.00192608	0.0060138	1.41%	0.73%
4700	0.00189953	0.00597262	1.38%	0.68%
4800	0.00187454	0.00593617	1.32%	0.61%

4900	0.00185153	0.00590057	1.23%	0.60%
5000	0.0018288	0.00586551	1.23%	0.59%

Table B9. The training and test mean square error at each 100 epochs for the 4-layer model without dropout, trained using 6 features with the selected cutoff epoch highlighted

Epochs	Train Loss	Test Loss	Percentage Decrease In Train Loss	Percentage Decrease in Test Loss
1	0.27583	0.196949	-	-
100	0.0644682	0.0492359	76.63%	75.00%
200	0.0443687	0.0384489	31.18%	21.91%
300	0.034705	0.0317874	21.78%	17.33%
400	0.0269972	0.0274778	22.21%	13.56%
500	0.0248823	0.023324	7.83%	15.12%
600	0.0204614	0.0197834	17.77%	15.18%
700	0.0181128	0.0172406	11.48%	12.85%
800	0.0159007	0.0153917	12.21%	10.72%
900	0.0134842	0.0137306	15.20%	10.79%
1000	0.0124144	0.0128736	7.93%	6.24%
1100	0.0125436	0.0112817	-1.04%	12.37%
1200	0.0108082	0.010363	13.83%	8.14%
1300	0.0104269	0.00978191	3.53%	5.61%
1400	0.00887875	0.0089462	14.85%	8.54%
1500	0.0100914	0.00848904	-13.66%	5.11%
1600	0.00841296	0.00798404	16.63%	5.95%

1700	0.00911988	0.00756038	-8.40%	5.31%
1800	0.007955	0.00725755	12.77%	4.01%
1900	0.00793801	0.00700663	0.21%	3.46%
2000	0.00671	0.00661353	15.47%	5.61%
2100	0.0072831	0.00637377	-8.54%	3.63%
2200	0.00783297	0.00614893	-7.55%	3.53%
2300	0.00708508	0.00604206	9.55%	1.74%
2400	0.00732682	0.00584665	-3.41%	3.23%
2500	0.00684522	0.0058775	6.57%	-0.53%
2600	0.0064598	0.00550349	5.63%	6.36%
2700	0.00707521	0.00548413	-9.53%	0.35%
2800	0.00624215	0.00536532	11.77%	2.17%
2900	0.00586106	0.00533818	6.11%	0.51%
3000	0.00597452	0.00512768	-1.94%	3.94%
3100	0.00612851	0.00507334	-2.58%	1.06%
3200	0.00574826	0.00498059	6.20%	1.83%
3300	0.00606397	0.00498333	-5.49%	-0.06%
3400	0.0064794	0.00496442	-6.85%	0.38%
3500	0.00615197	0.00483363	5.05%	2.63%
3600	0.00594342	0.00478085	3.39%	1.09%
3700	0.00580967	0.00481708	2.25%	-0.76%
3800	0.00598545	0.00471635	-3.03%	2.09%
3900	0.00503411	0.00469385	15.89%	0.48%
4000	0.00577226	0.00470275	-14.66%	-0.19%
4100	0.00536802	0.00465401	7.00%	1.04%
4200	0.00540028	0.004592	-0.60%	1.33%

4300	0.00532925	0.00454987	1.32%	0.92%
4400	0.00511294	0.00455499	4.06%	-0.11%
4500	0.0053854	0.00454638	-5.33%	0.19%
4600	0.00527662	0.0045083	2.02%	0.84%
4700	0.00599669	0.00447176	-13.65%	0.81%
4800	0.005352	0.00445252	10.75%	0.43%
4900	0.00524555	0.00443413	1.99%	0.41%
5000	0.00554019	0.00437948	-5.62%	1.23%
5100	0.00519857	0.00441552	6.17%	-0.82%
5200	0.00487042	0.00443343	6.31%	-0.41%
5300	0.00539806	0.00436756	-10.83%	1.49%
5400	0.0052508	0.00438654	2.73%	-0.43%
5500	0.00512898	0.00430187	2.32%	1.93%
5600	0.00517807	0.0043527	-0.96%	-1.18%
5700	0.00500472	0.00432688	3.35%	0.59%
5800	0.00552669	0.00435618	-10.43%	-0.68%
5900	0.00477225	0.00425795	13.65%	2.25%
6000	0.00485934	0.00427879	-1.82%	-0.49%
6100	0.00508775	0.00426726	-4.70%	0.27%
6200	0.00485872	0.00428172	4.50%	-0.34%
6300	0.00477014	0.00429001	1.82%	-0.19%
6400	0.00487398	0.00421537	-2.18%	1.74%
6500	0.00461786	0.00425513	5.25%	-0.94%
6600	0.00476407	0.00422758	-3.17%	0.65%
6700	0.00511389	0.00421727	-7.34%	0.24%
6800	0.00479807	0.00423052	6.18%	-0.31%

6900	0.0049124	0.0042363	-2.38%	-0.14%
7000	0.00571809	0.00418341	-16.40%	1.25%
7100	0.00492647	0.00422636	13.84%	-1.03%
7200	0.00467308	0.0041992	5.14%	0.64%
7300	0.00493461	0.00414599	-5.60%	1.27%
7400	0.00489645	0.00416813	0.77%	-0.53%
7500	0.00485981	0.00413951	0.75%	0.69%
7600	0.00482514	0.00414184	0.71%	-0.06%
7700	0.0048626	0.00411174	-0.78%	0.73%
7800	0.00481455	0.00413482	0.99%	-0.56%
7900	0.00440445	0.00413807	8.52%	-0.08%
8000	0.00518592	0.00414909	-17.74%	-0.27%
8100	0.00500852	0.00414018	3.42%	0.21%
8200	0.00483141	0.00411875	3.54%	0.52%
8300	0.0045958	0.00412418	4.88%	-0.13%
8400	0.00492193	0.00411799	-7.10%	0.15%
8500	0.00467635	0.00409104	4.99%	0.65%
8600	0.00461138	0.00405993	1.39%	0.76%
8700	0.00497008	0.00404537	-7.78%	0.36%
8800	0.00501333	0.00407401	-0.87%	-0.71%
8900	0.00481432	0.00411073	3.97%	-0.90%
9000	0.00498191	0.00409196	-3.48%	0.46%

Table B10. The training and test mean square error at each 100 epochs for the 5-layer model with dropout, trained using 6 features with the selected cutoff epoch highlighted

Epochs	Train Loss	Test Loss	Percentage Decrease In Train Loss	Percentage Decrease in Test Loss
1	0.237563	0.195047	-	-
100	0.0339309	0.0329467	85.72%	83.11%
200	0.0217951	0.021515	35.77%	34.70%
300	0.0156884	0.0168882	28.02%	21.50%
400	0.0119481	0.0140952	23.84%	16.54%
500	0.00946856	0.0122629	20.75%	13.00%
600	0.00773437	0.0109888	18.32%	10.39%
700	0.00648847	0.0100544	16.11%	8.50%
800	0.00558287	0.0093704	13.96%	6.80%
900	0.00491208	0.00887039	12.02%	5.34%
1000	0.00440946	0.00848983	10.23%	4.29%
1100	0.0040189	0.00817725	8.86%	3.68%
1200	0.00370696	0.0079236	7.76%	3.10%
1300	0.00344583	0.00772279	7.04%	2.53%
1400	0.00322777	0.00755572	6.33%	2.16%
1500	0.0030417	0.00742806	5.76%	1.69%
1600	0.00288221	0.00731692	5.24%	1.50%
1700	0.00274626	0.00722706	4.72%	1.23%
1800	0.00262933	0.00715089	4.26%	1.05%
1900	0.00252685	0.00707809	3.90%	1.02%
2000	0.00243594	0.00700903	3.60%	0.98%
2100	0.0023544	0.00694938	3.35%	0.85%
2200	0.00228122	0.00689639	3.11%	0.76%

2300	0.00221507	0.00684556	2.90%	0.74%
2400	0.00215488	0.00679937	2.72%	0.67%
2500	0.0020993	0.00675686	2.58%	0.63%
2600	0.00204769	0.00671783	2.46%	0.58%
2700	0.00200022	0.00668155	2.32%	0.54%
2800	0.00195618	0.00664288	2.20%	0.58%
2900	0.00191451	0.00660231	2.13%	0.61%
3000	0.00187578	0.00656431	2.02%	0.58%
3100	0.00183967	0.00652743	1.93%	0.56%
3200	0.00180541	0.00649282	1.86%	0.53%
3300	0.00177359	0.00645916	1.76%	0.52%
3400	0.00174363	0.00642681	1.69%	0.50%
3500	0.00171486	0.00639838	1.65%	0.44%
3600	0.00168809	0.00637329	1.56%	0.39%
3700	0.0016633	0.00634852	1.47%	0.39%
3800	0.0016402	0.00632272	1.39%	0.41%
3900	0.00161822	0.00630051	1.34%	0.35%
4000	0.00159771	0.00627609	1.27%	0.39%
4100	0.0015781	0.00625196	1.23%	0.38%
4200	0.00155922	0.00622657	1.20%	0.41%
4300	0.00154121	0.00620157	1.16%	0.40%
4400	0.001524	0.00617701	1.12%	0.40%
4500	0.00150729	0.00615322	1.10%	0.39%
4600	0.00149126	0.00613146	1.06%	0.35%
4700	0.00147602	0.00610946	1.02%	0.36%
4800	0.00146162	0.00608875	0.98%	0.34%

4900	0.00144794	0.00606826	0.94%	0.34%
5000	0.0014346	0.00604818	0.92%	0.33%

Table B11. The training and test mean square error at each 100 epochs for the 5-layer model without dropout, trained using 6 features with the selected cutoff epoch highlighted

Based on these data gathered, the model configuration which has the lowest test error at their chosen cut-off epochs is the four-layer network with dropout, with a test error of 0.0036. This is even though the dropout models tend to have a higher training loss compared to the non-dropout variants at the same epoch.

4 Conclusions

In this report, we addressed two different types of problems, classification and regression problems. We discussed about the methods and detailed the results of our experiments and is summarized below:

4.1 Part A: Classification Problem

- Number of epochs where test error converges – 5,000 to 6,000
- Optimal batch size – 8
- Optimal number of neurons for hidden layer – 25
- Optimal decay parameter – 0
- Preferred optimal model – 4-Layer network

4.2 Part B: Regression Problem

- Number of epochs where test error is minimum for 3-layer model with no dropout – 3100
- Features that are most correlated to each other – (CGPA, Chance of Admission), (TOEFL score, GRE score), (CGPA, GRE score), (CGPA, TOEFL score)
- Features with the highest correlations with admission chance – CGPA, GRE Score, TOEFL score
- Preferred optimal network – four-layer network with dropout

Finally, we shall discuss on our takeaways and conclusions on findings below:

- Determining a convergence range is important as it is a trade-off between loss and training time. A model can be trained endlessly but only yield minute differences and is a waste of time.
- Choosing a batch size is often a trade-off between accuracy and time. The smaller the batch size, the longer time it takes, but with a higher accuracy gained.
- Increasing the number of neurons within a hidden layer does not necessary improve the performance of the model. It only offers the model with more parameters to use for tuning and learning, but this could in turn cause the model to generalize better or incur more noise.
- Regularization techniques are both problem and model specific. Its purpose is to improve the model when there is overfitting, i.e. a significant difference between training and testing accuracies, as it helps in generalizing the model.
- The datasets provided for both problems are very small (less than 5000) hence most of the results obtained have very low losses and high accuracies.
- It may be better not to train two models to compare at the same number of epochs, as their cut-off epochs may be different.
- A deeper model may not be better, especially since our dataset is small and thus may cause a deeper model to overfit more easily.
- A model with dropout will generally generalize better than a model without.

5 Appendices

5.1 Part A: Classification

5.1.1 Question 1

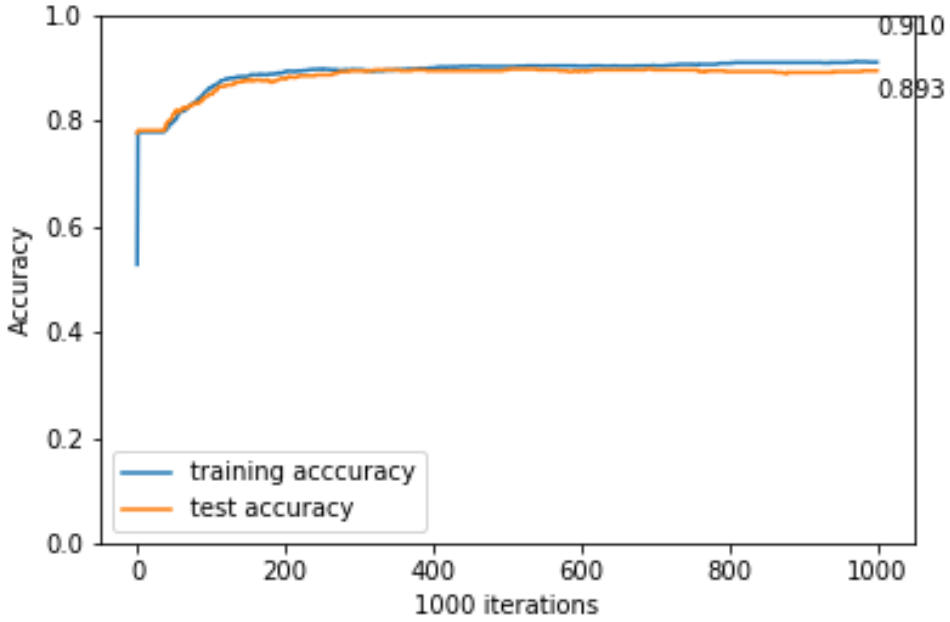

Figure A1.1: Training and Test Accuracy at epochs 1000

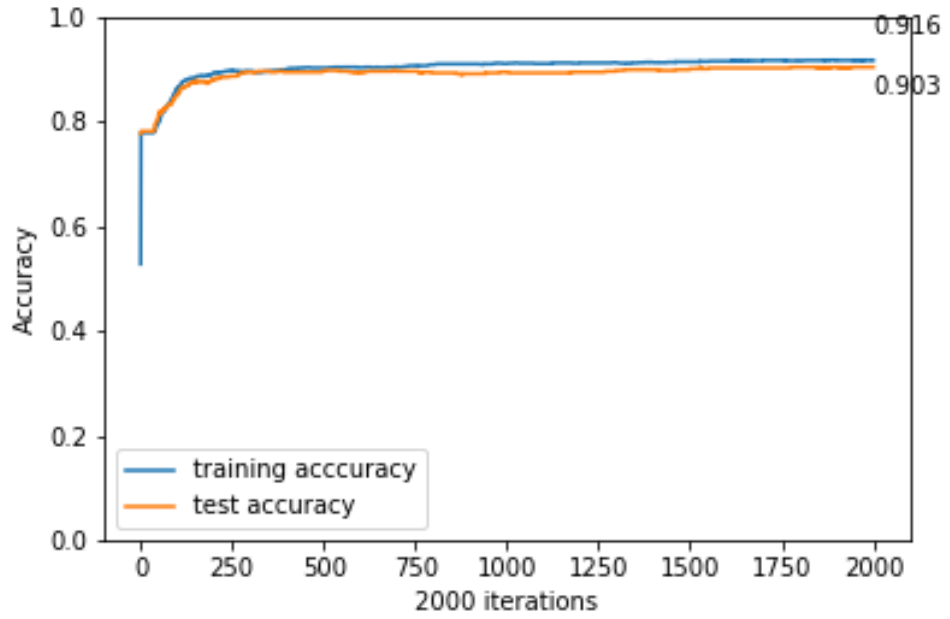

Figure A1.2: Training and Test Accuracy at epochs 2000

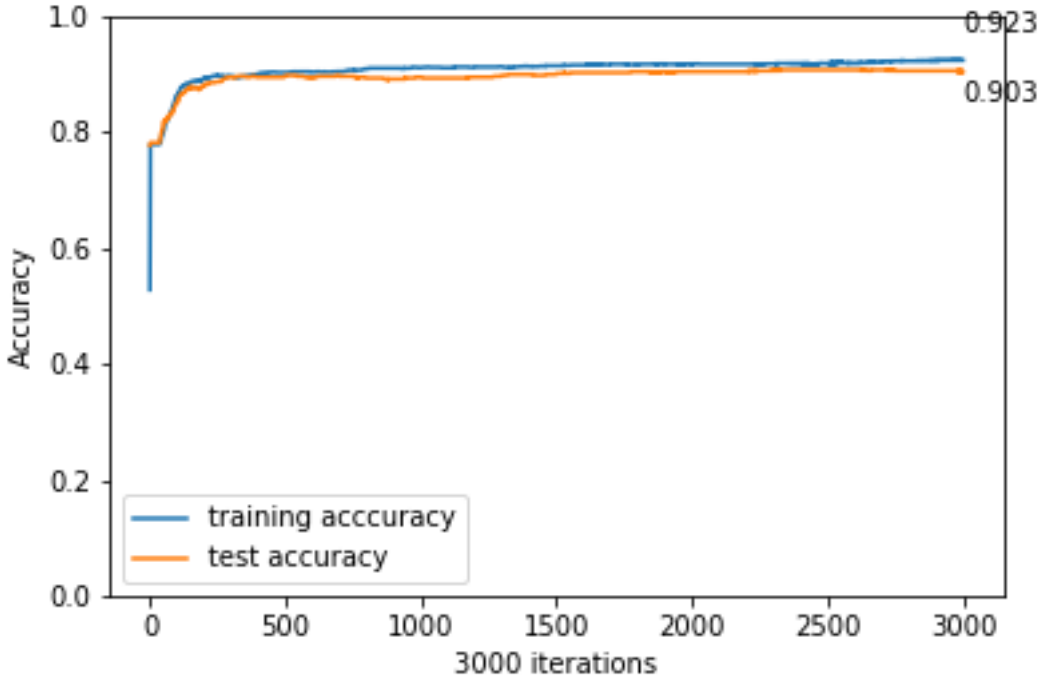

Figure A1.3: Training and Test Accuracy at epochs 3000

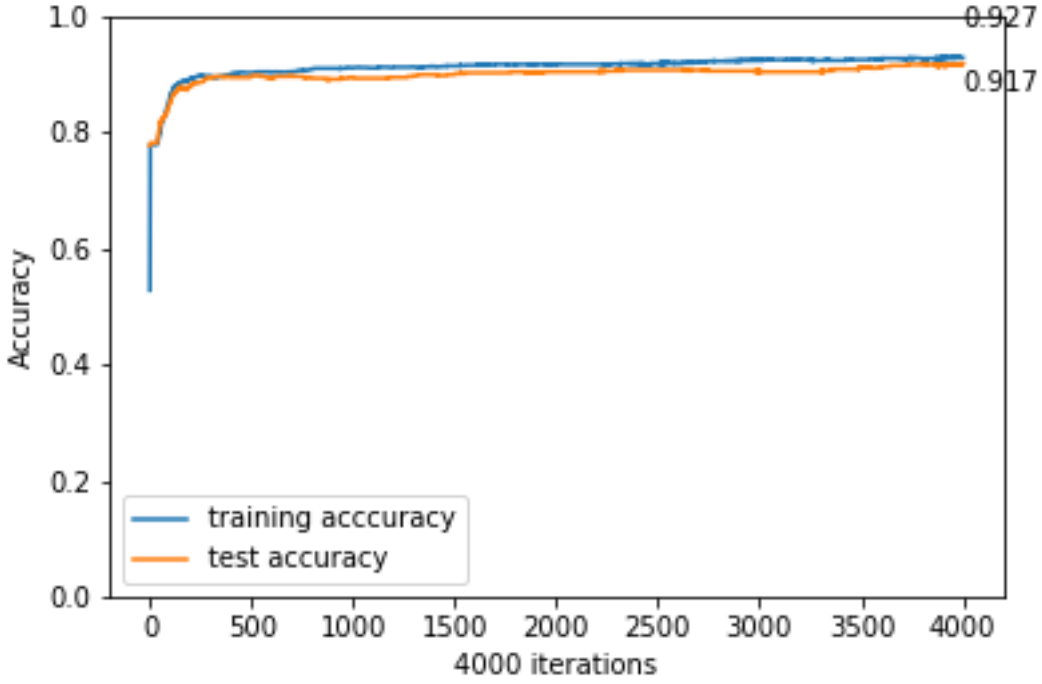

Figure A1.4: Training and Test Accuracy at epochs 4000

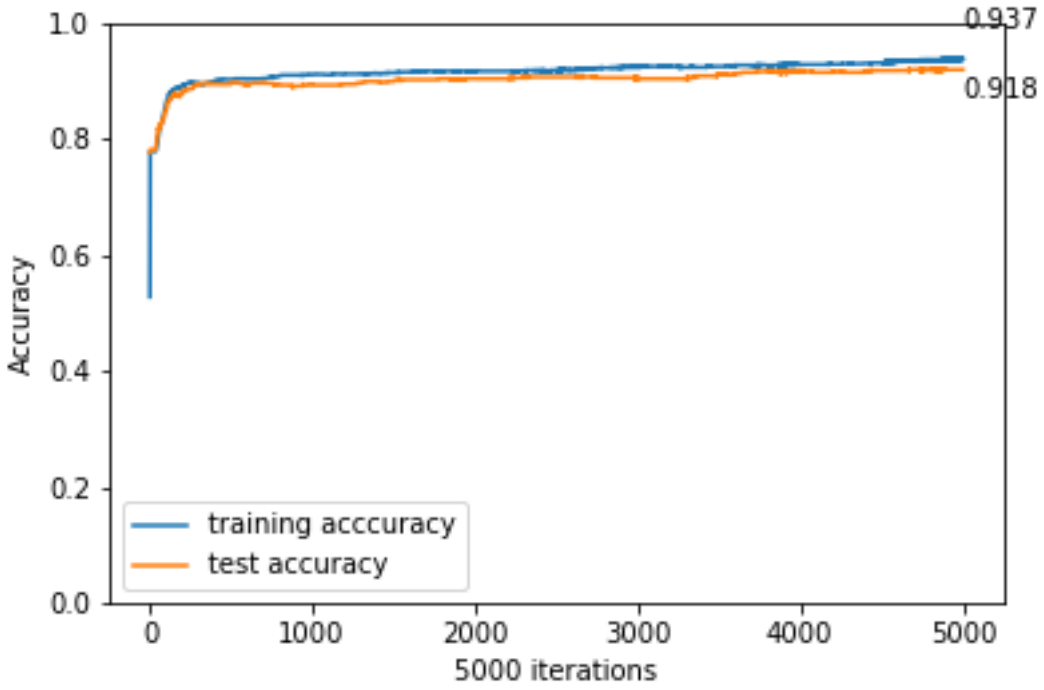

Figure A1.5: Training and Test Accuracy at epochs 5000

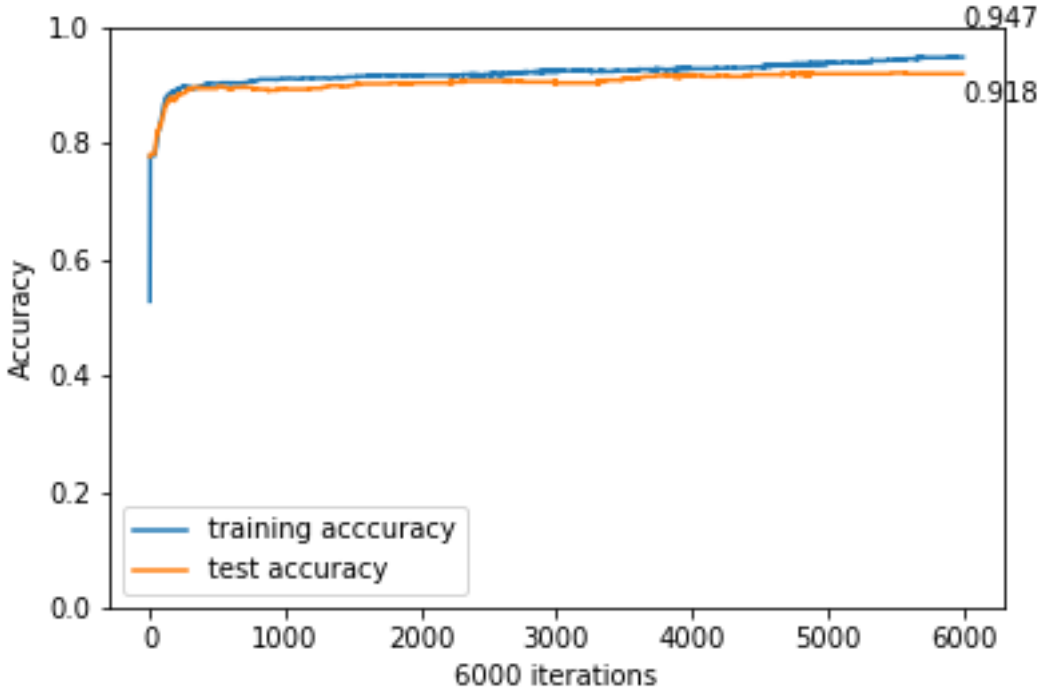

Figure A1.6: Training and Test Accuracy at epochs 6000

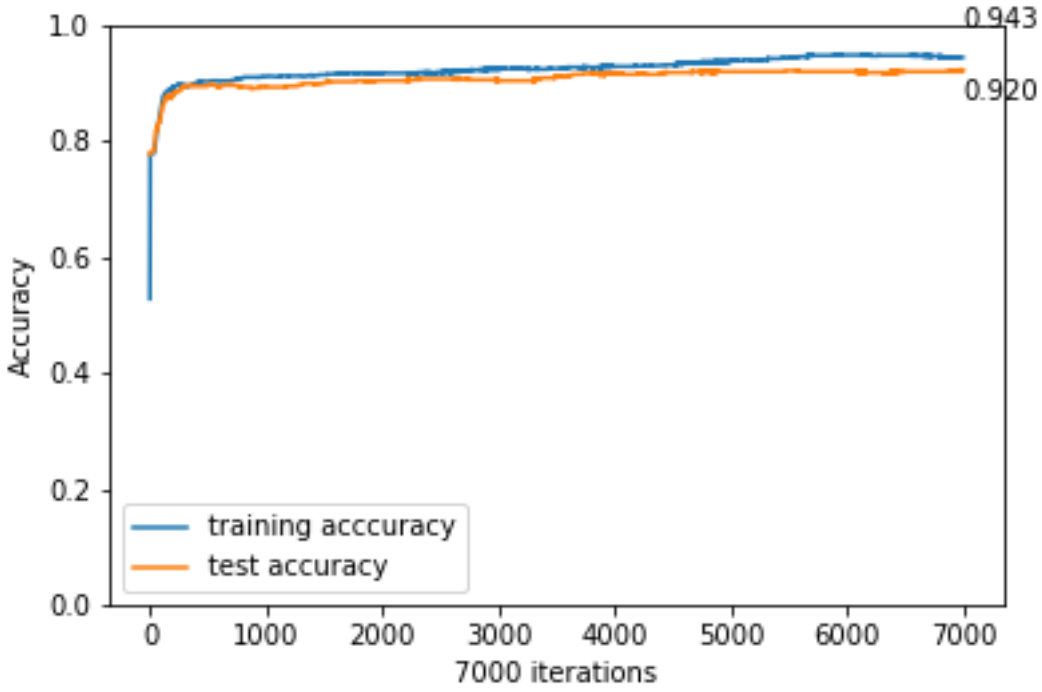

Figure A1.7: Training and Test Accuracy at epochs 7000

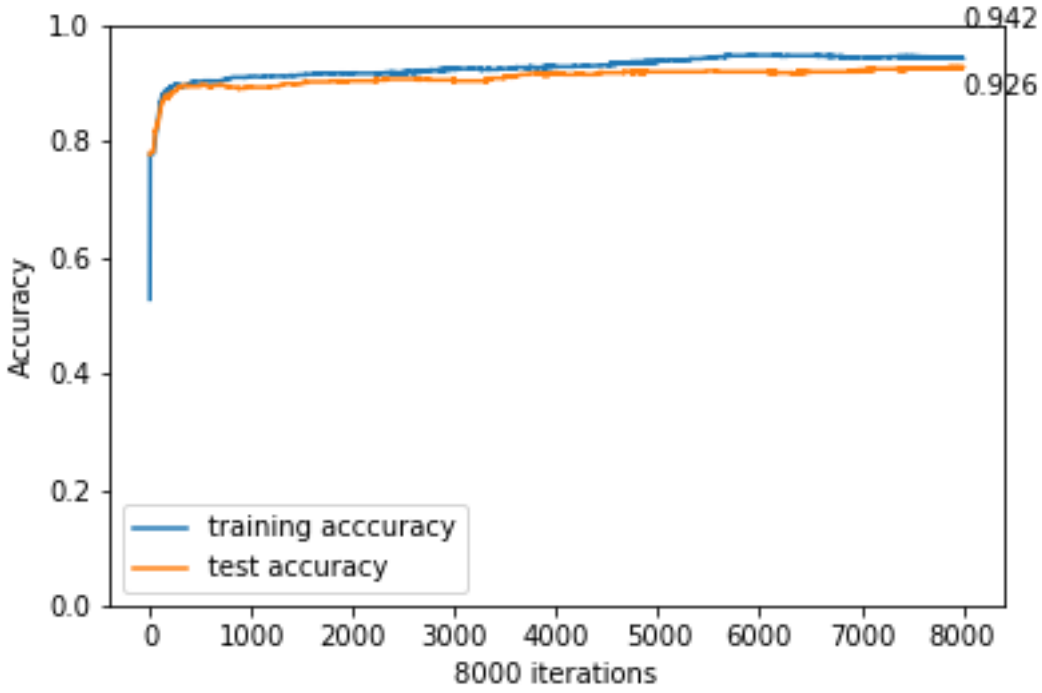

Figure A1.8: Training and Test Accuracy at epochs 8000

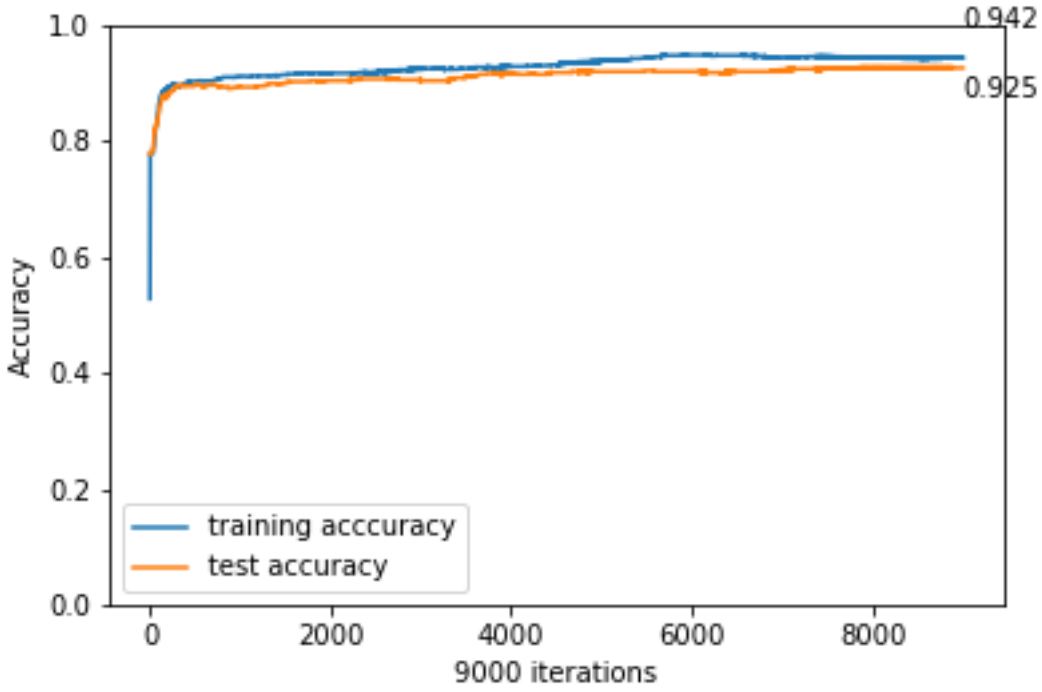

Figure A1.9: Training and Test Accuracy at epochs 9000

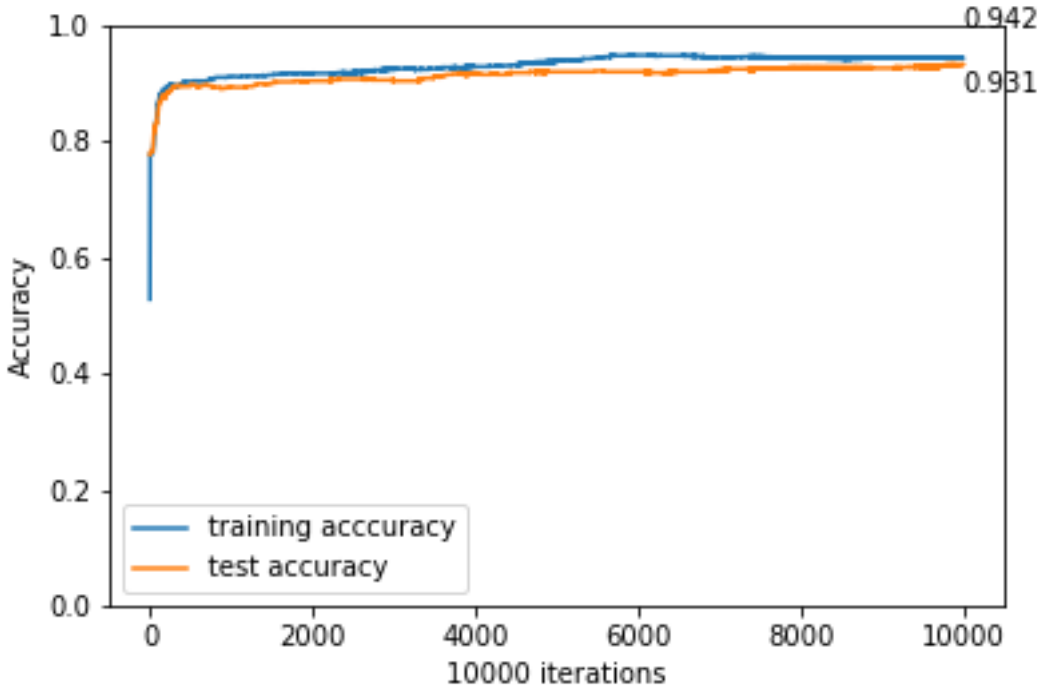

Figure A1.10: Training and Test Accuracy at epochs 10000

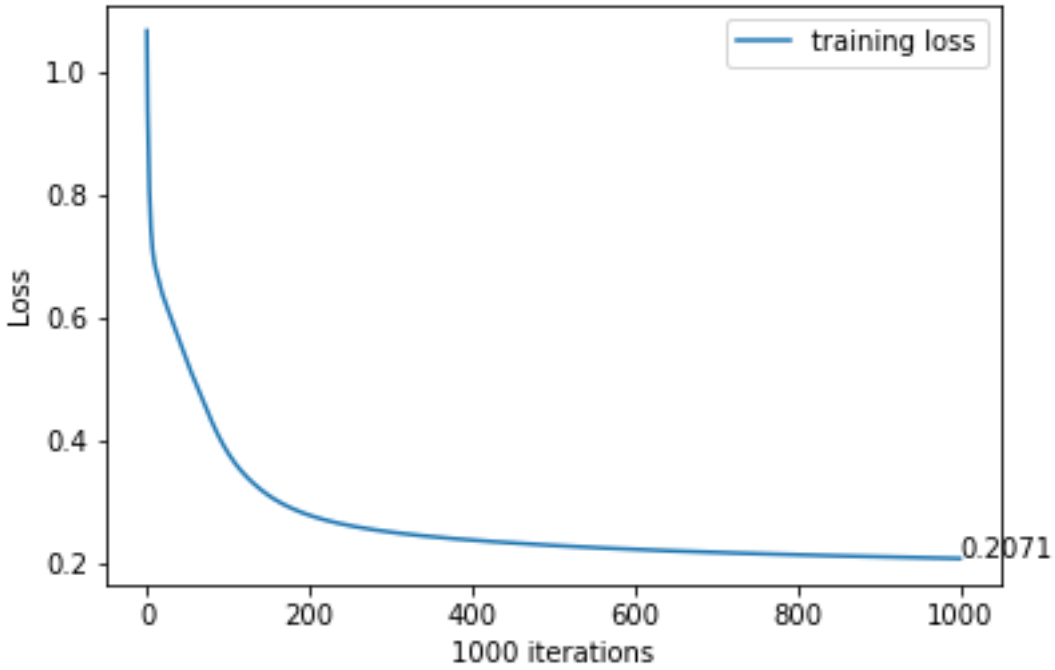

Figure A1.11: Training Loss at epochs 1000

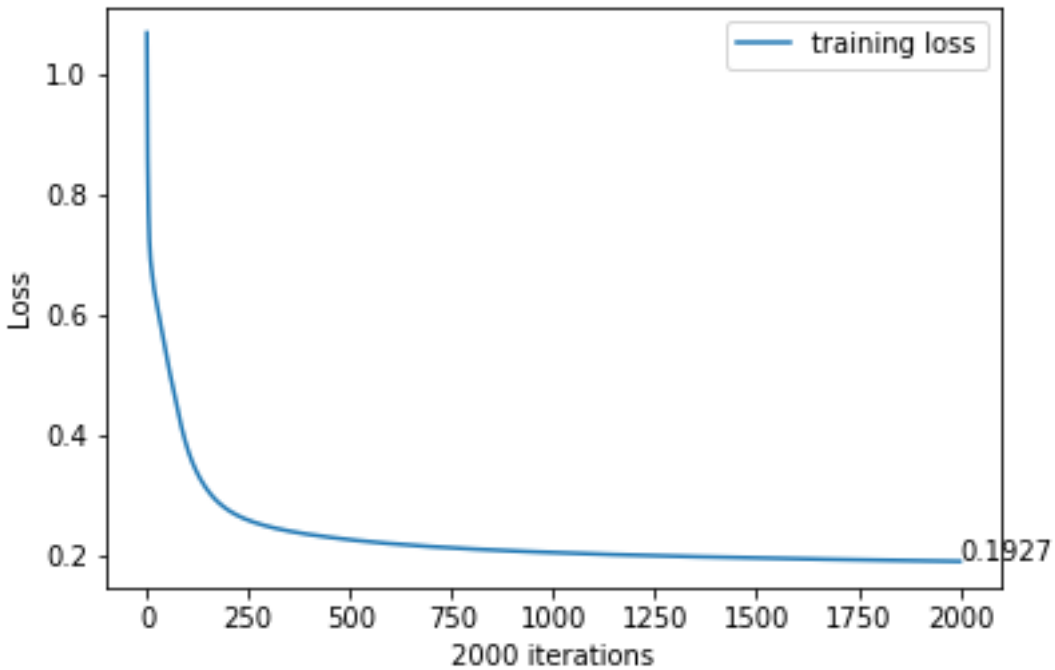

Figure A1.12: Training Loss at epochs 2000

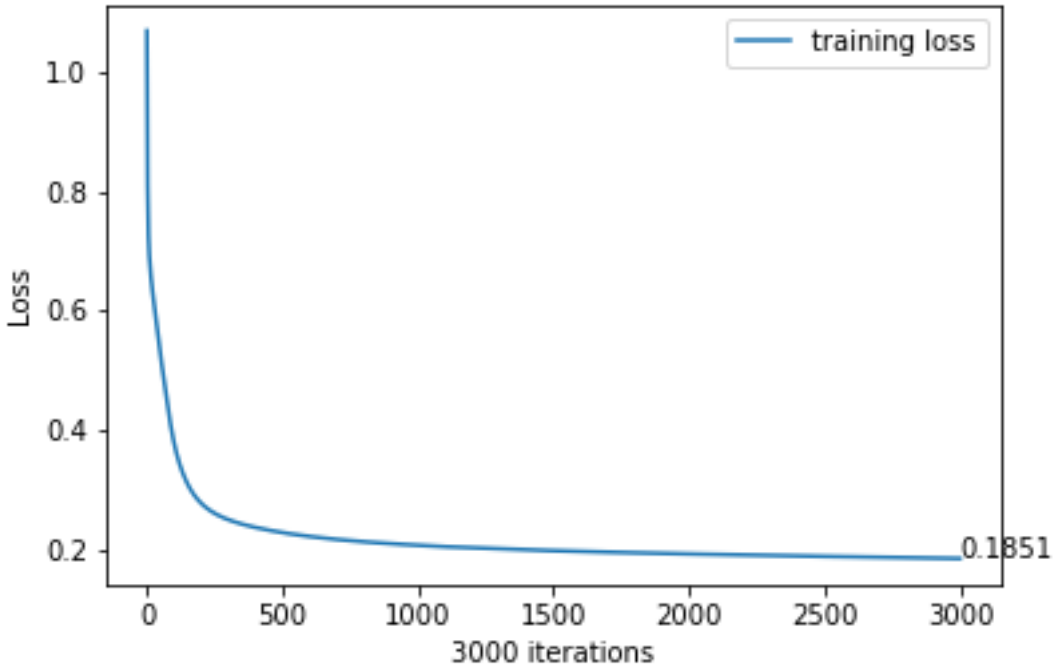

Figure A1.13: Training Loss at epochs 3000

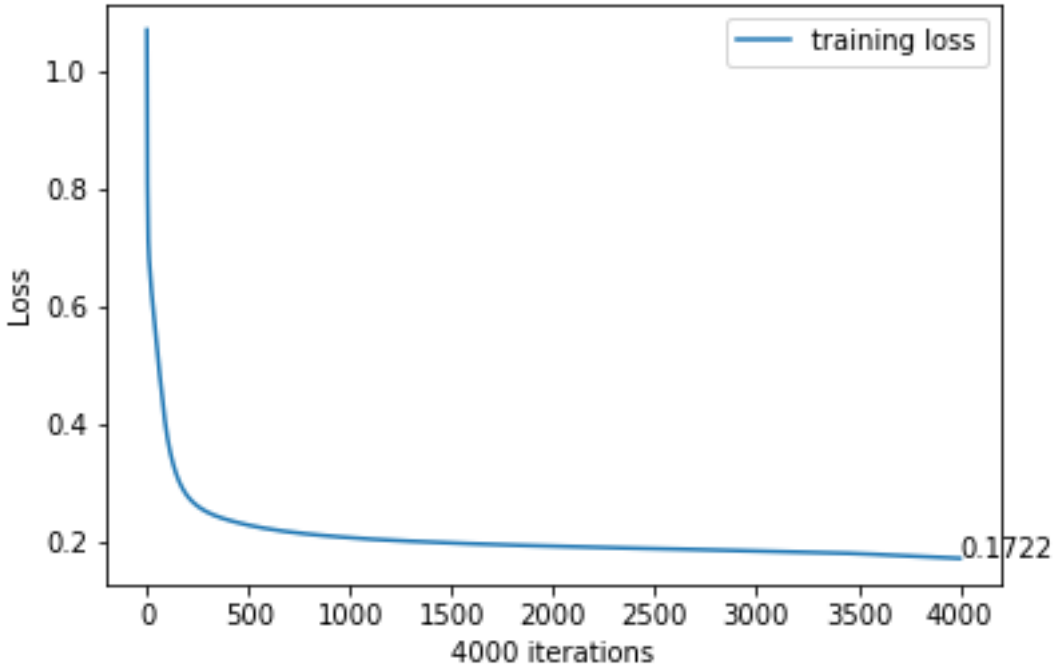

Figure A1.14: Training Loss at epochs 4000

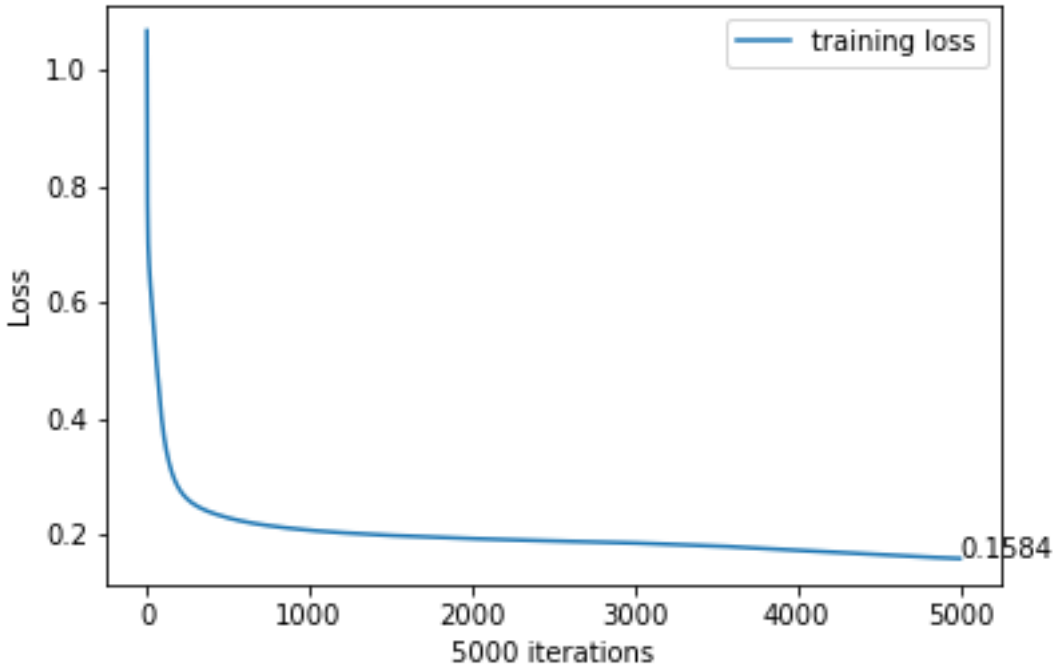

Figure A1.15: Training Loss at epochs 5000

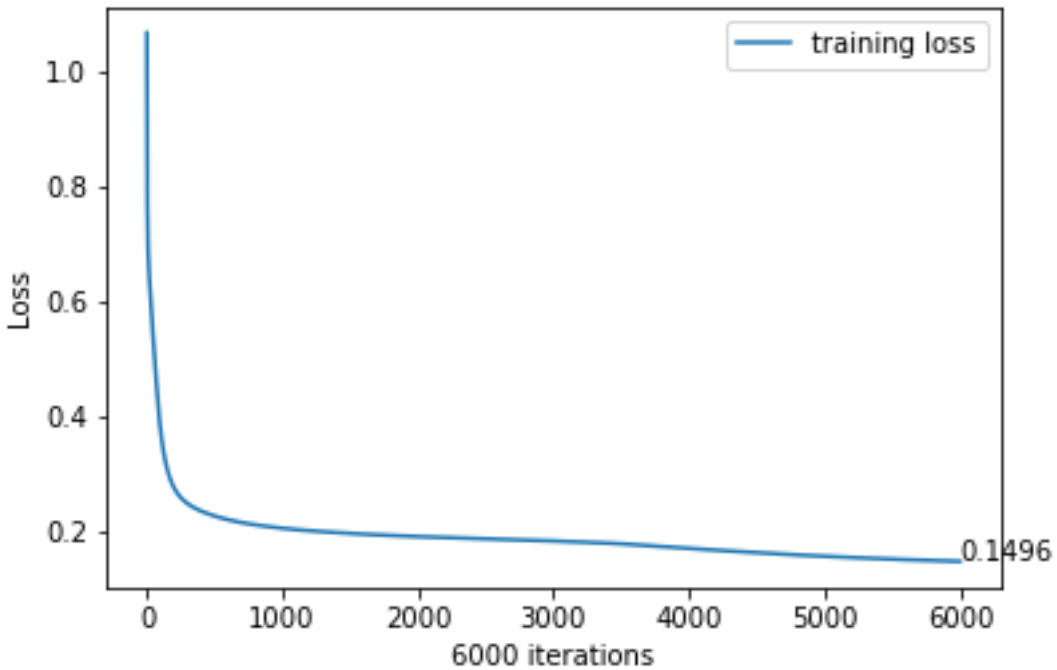

Figure A1.16: Training Loss at epochs 6000

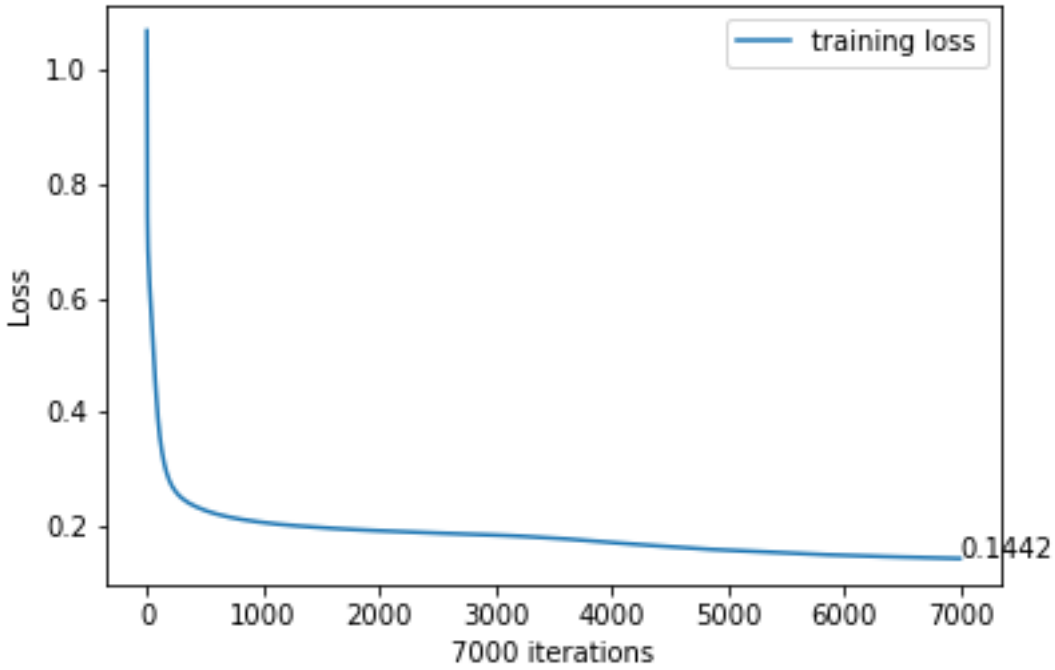

Figure A1.17: Training Loss at epochs 7000

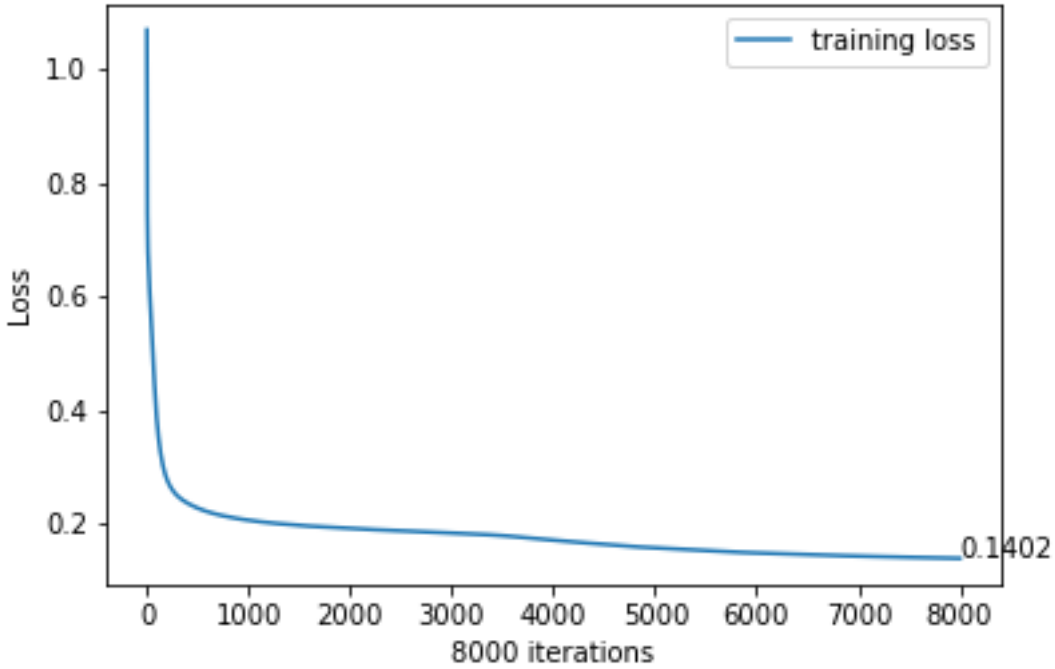

Figure A1.18: Training Loss at epochs 8000

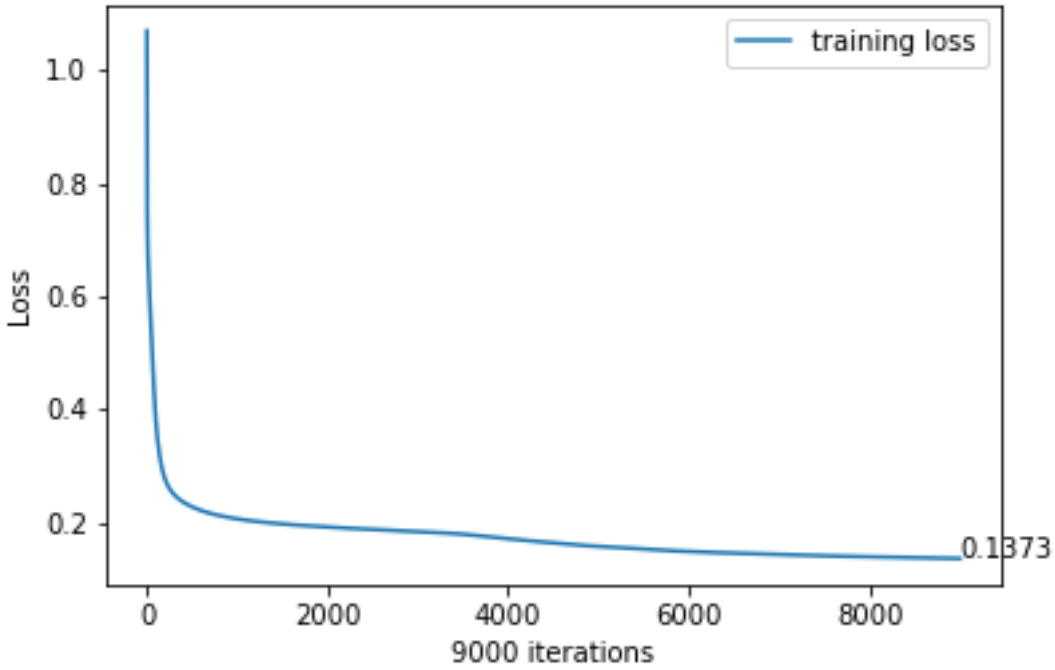

Figure A1.19: Training Loss at epochs 9000

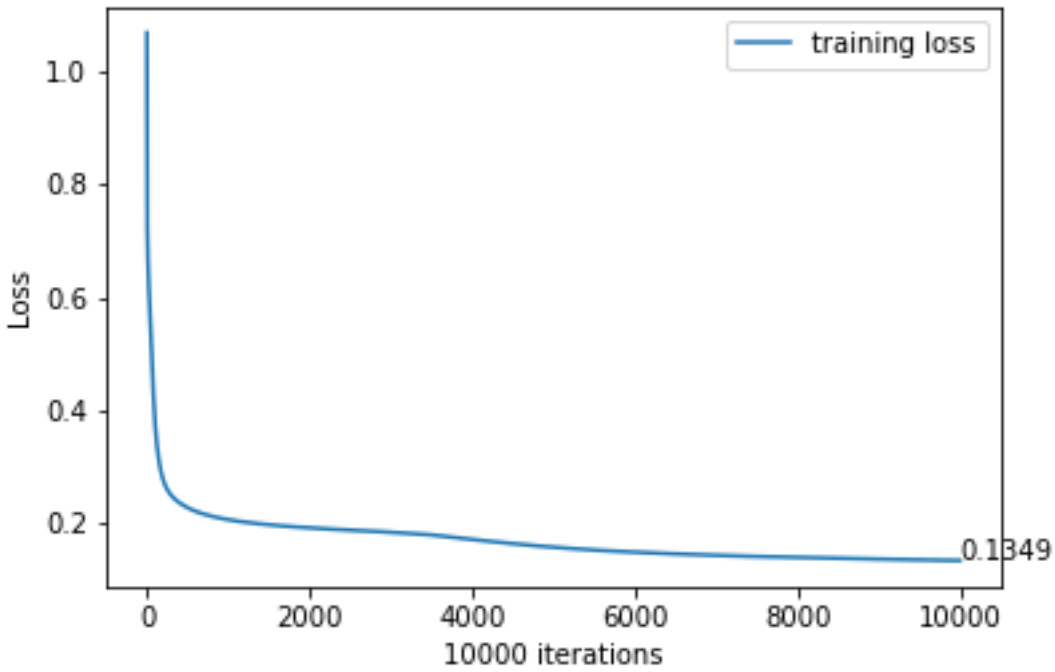

Figure A1.20: Training Loss at epochs 10000

5.1.2 Question 2

Batch size 4

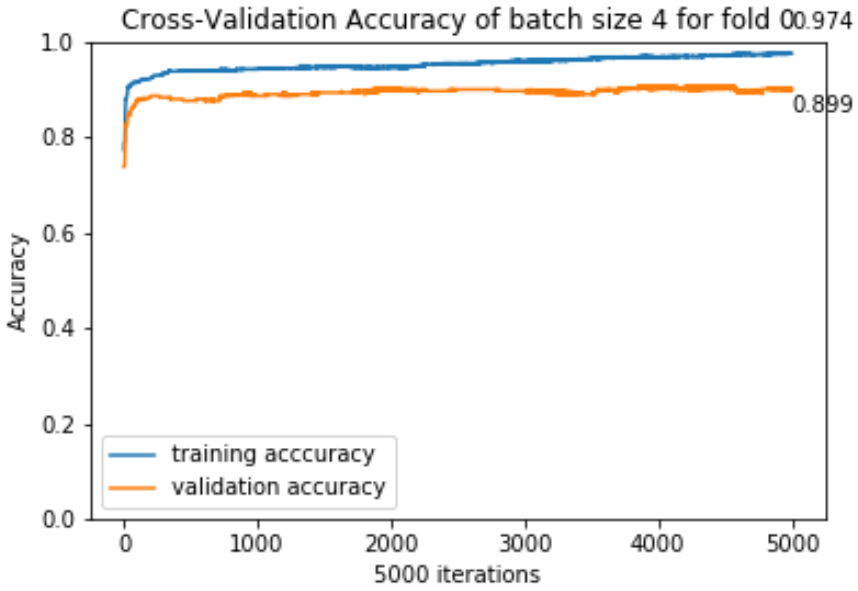

Figure A2.1: Accuracy for Fold 1 (Batch Size 4)

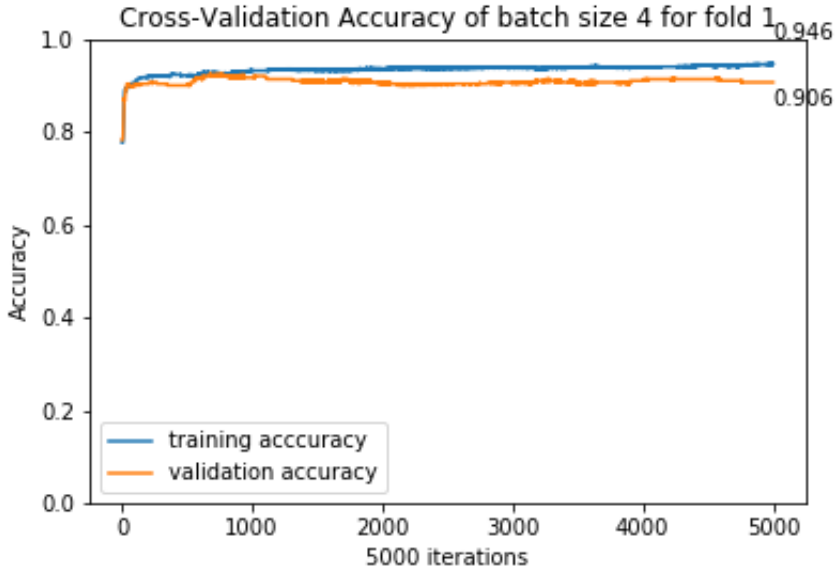

Figure A2.2: Accuracy for Fold 2 (Batch Size 4)

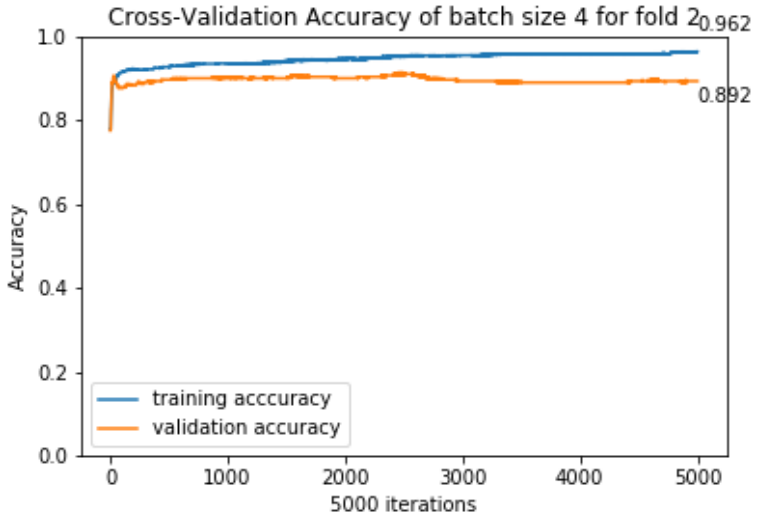

Figure A2.3: Accuracy for Fold 3 (Batch Size 4)

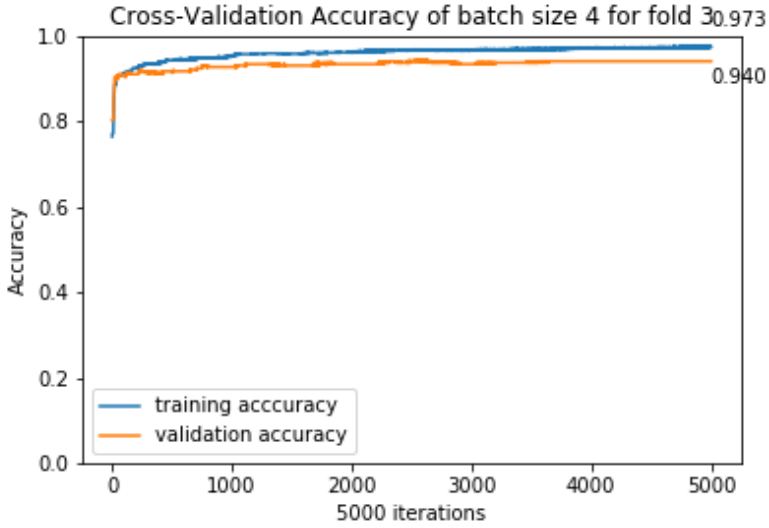

Figure A2.4: Accuracy for Fold 4 (Batch Size 4)

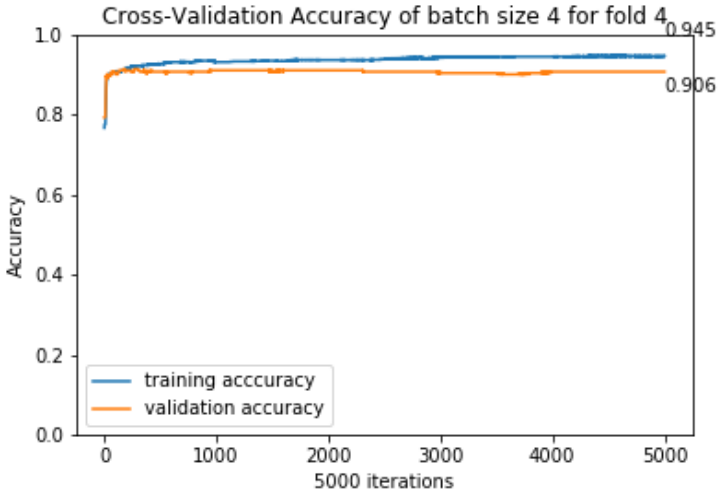

Figure A2.5: Accuracy for Fold 5 (Batch Size 4)

Batch Size 8

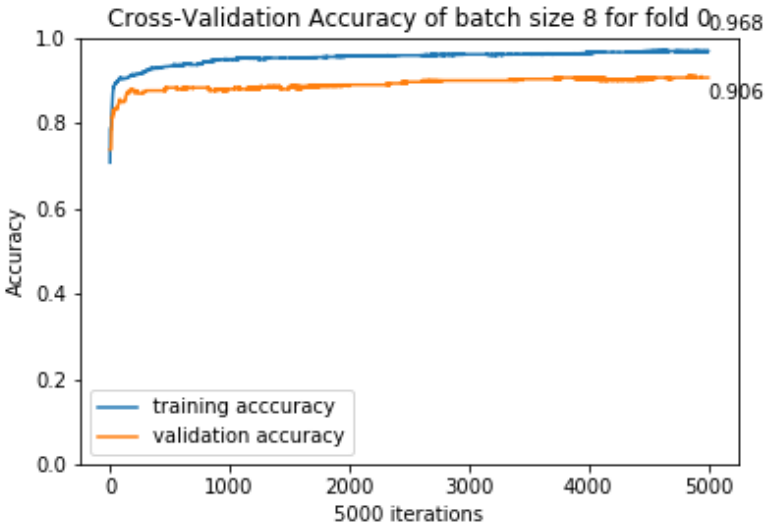

Figure A2.6: Accuracy for Fold 1 (Batch Size 8)

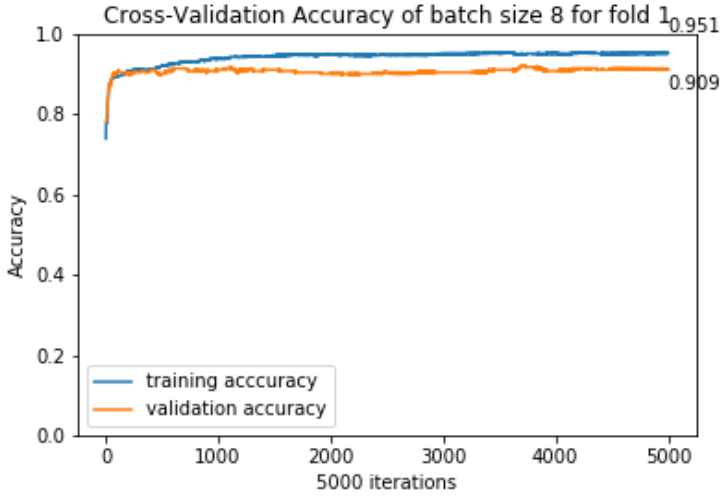

Figure A2.7: Accuracy for Fold 2 (Batch Size 8)

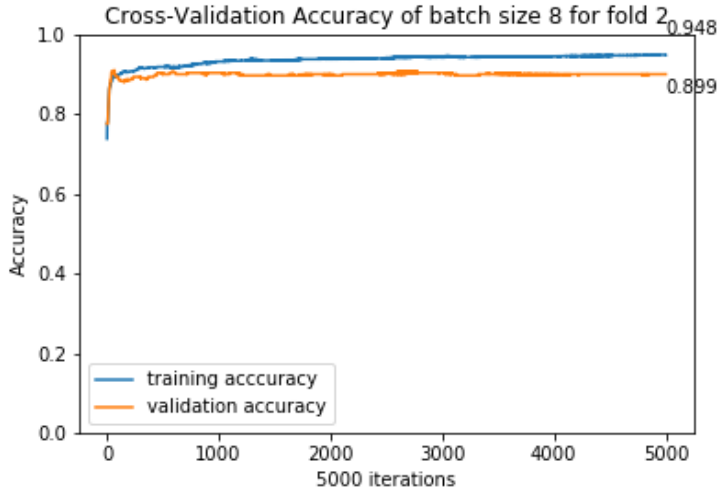

Figure A2.8: Accuracy for Fold 3 (Batch Size 8)

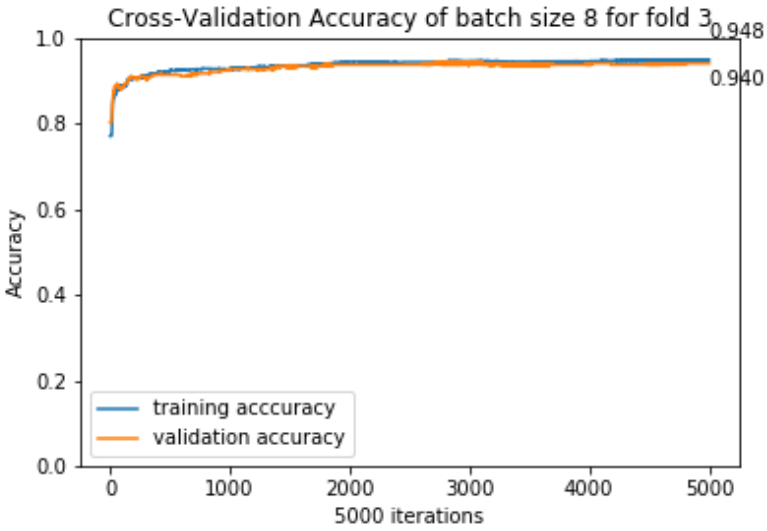

Figure A2.9: Accuracy for Fold 4 (Batch Size 8)

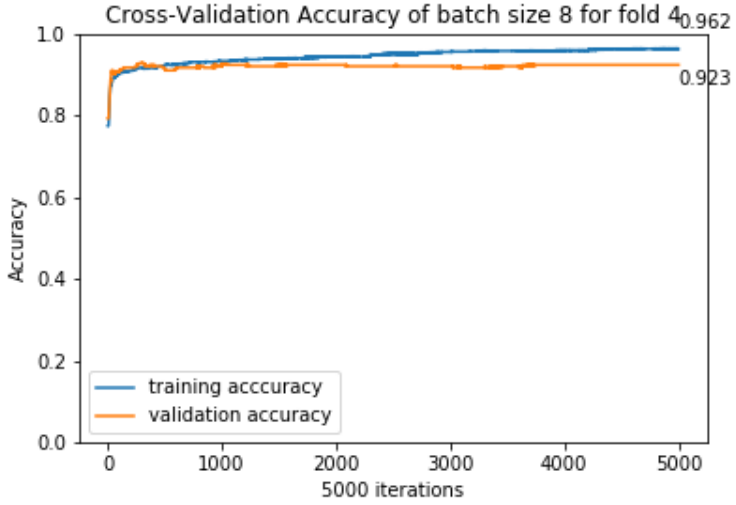

Figure A2.10: Accuracy for Fold 5 (Batch Size 8)

Batch Size 16

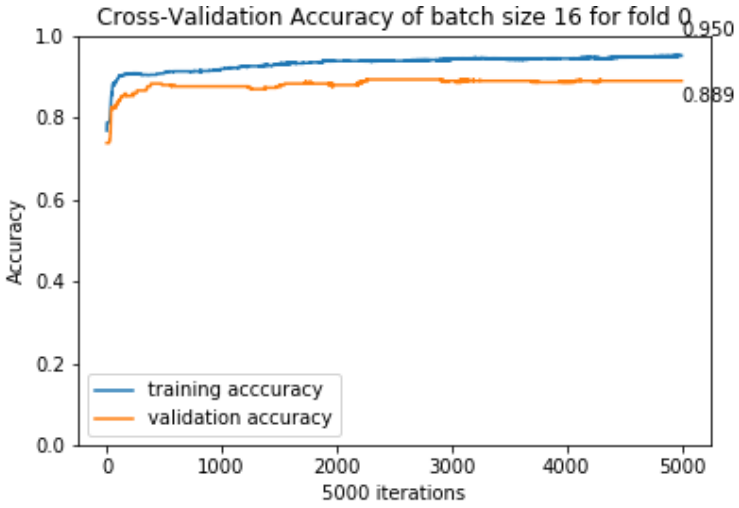

Figure A2.11: Accuracy for Fold 1 (Batch Size 16)

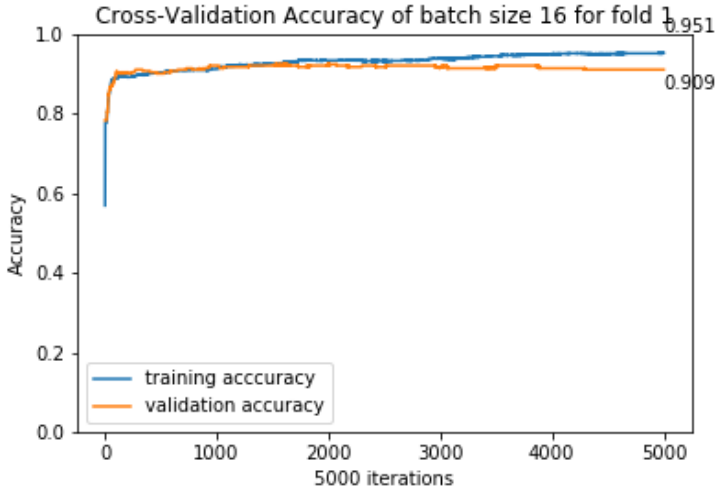

Figure A2.12: Accuracy for Fold 2 (Batch Size 16)

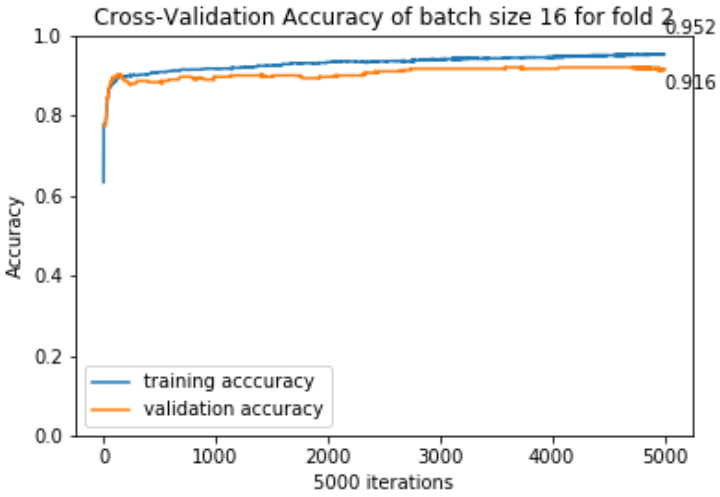

Figure A2.13: Accuracy for Fold 3 (Batch Size 16)

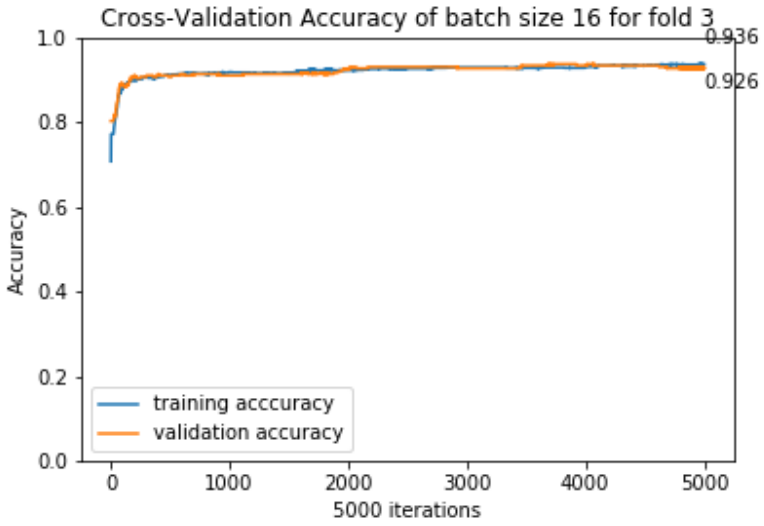

Figure A2.14: Accuracy for Fold 4 (Batch Size 16)

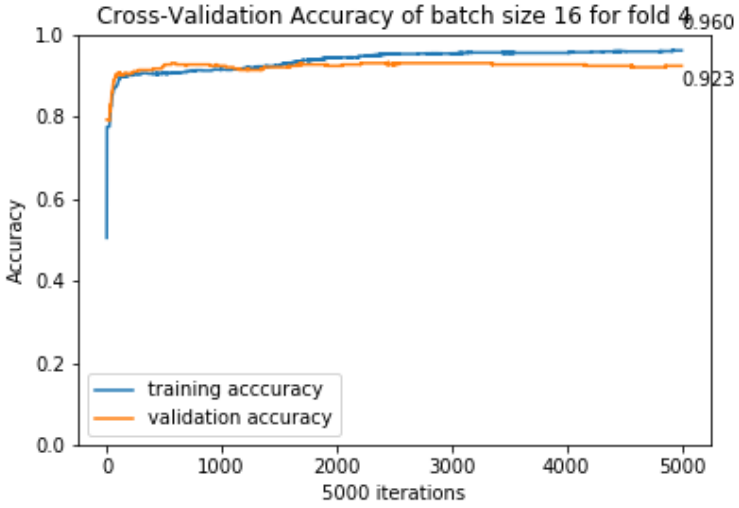

Figure A2.15: Accuracy for Fold 5 (Batch Size 16)

Batch Size 32

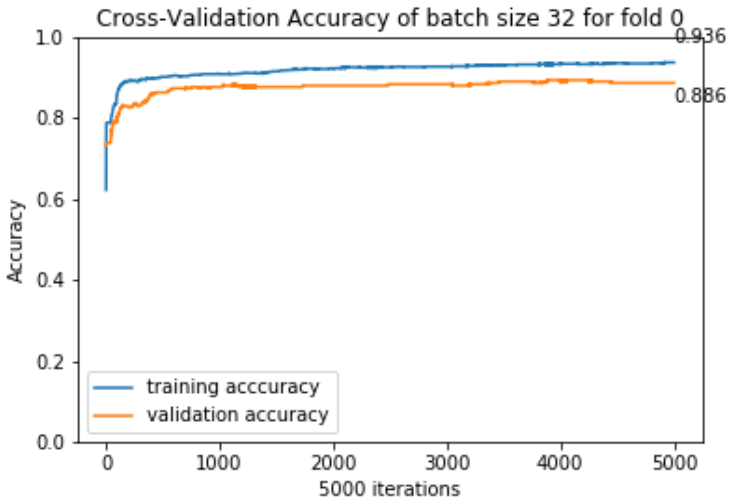

Figure A2.16: Accuracy for Fold 1 (Batch Size 32)

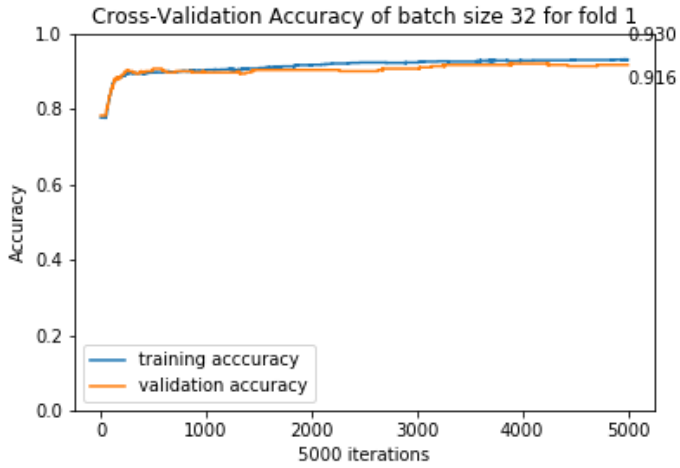

Figure A2.17: Accuracy for Fold 2 (Batch Size 32)

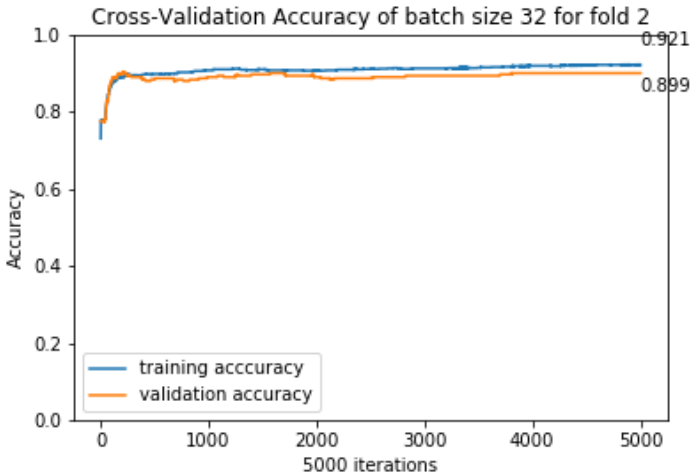

Figure A2.18: Accuracy for Fold 3 (Batch Size 32)

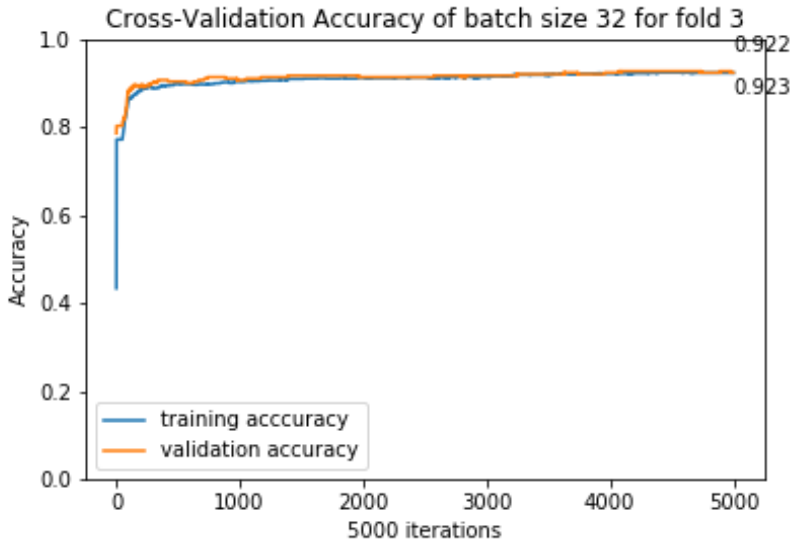

Figure A2.19: Accuracy for Fold 4 (Batch Size 32)

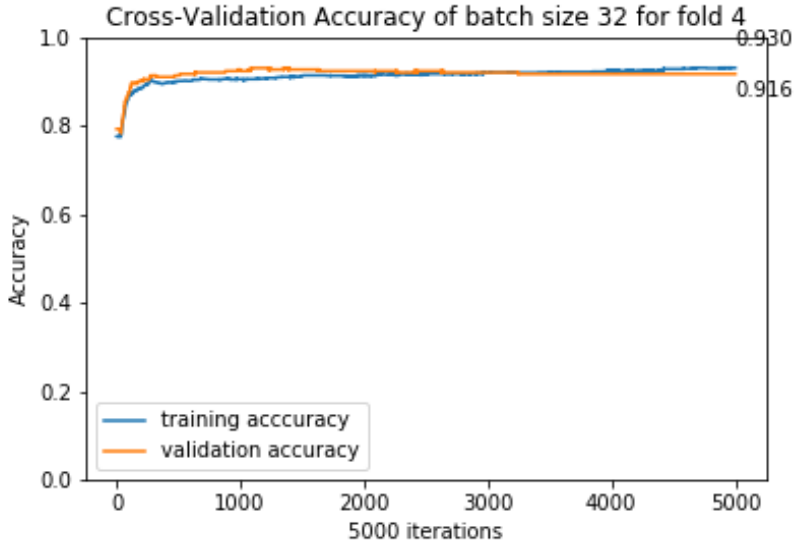

Figure A2.20: Accuracy for Fold 5 (Batch Size 32)

Batch Size 64

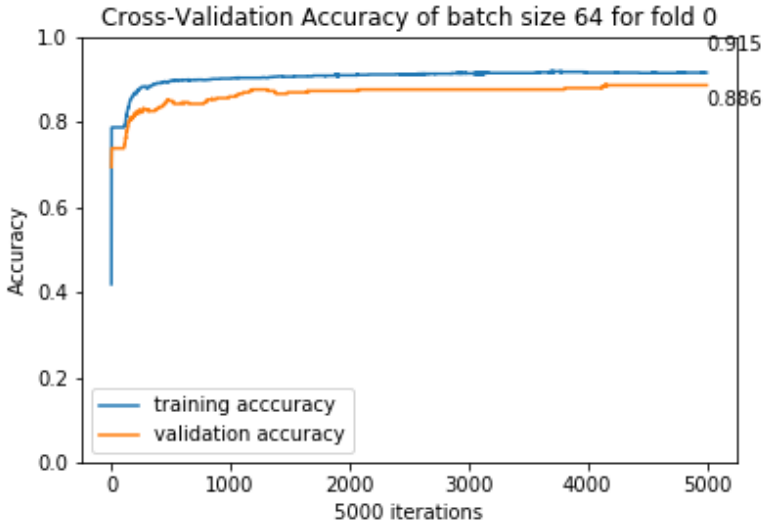

Figure A2.21: Accuracy for Fold 1 (Batch Size 64)

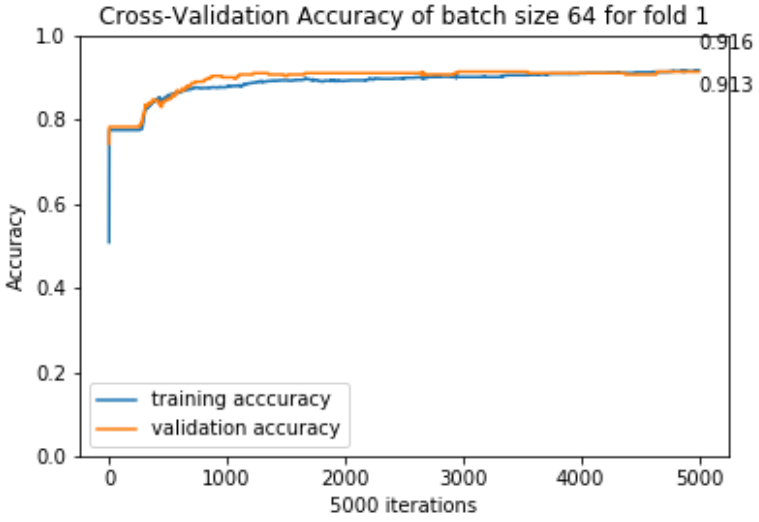

Figure A2.22: Accuracy for Fold 2 (Batch Size 64)

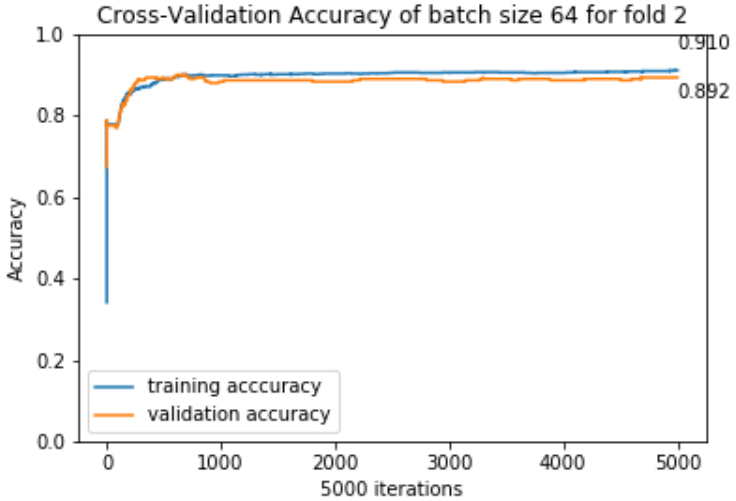

Figure A2.23: Accuracy for Fold 3 (Batch Size 64)

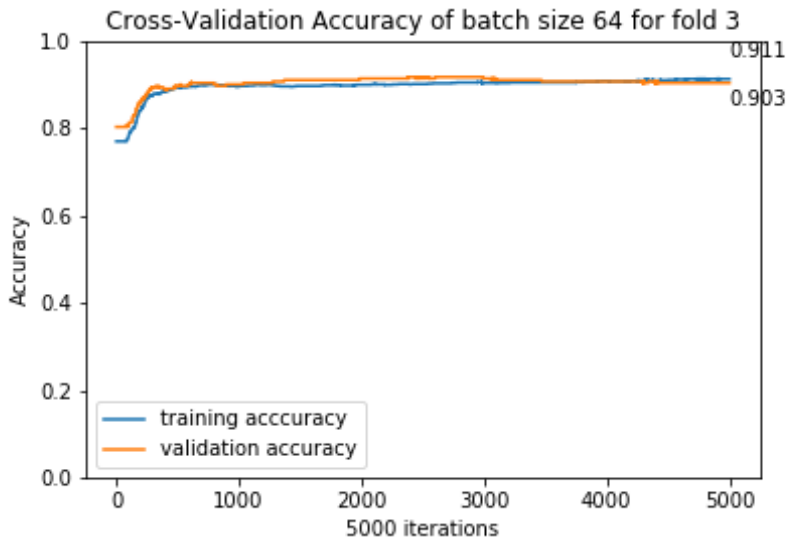

Figure A2.24: Accuracy for Fold 4 (Batch Size 64)

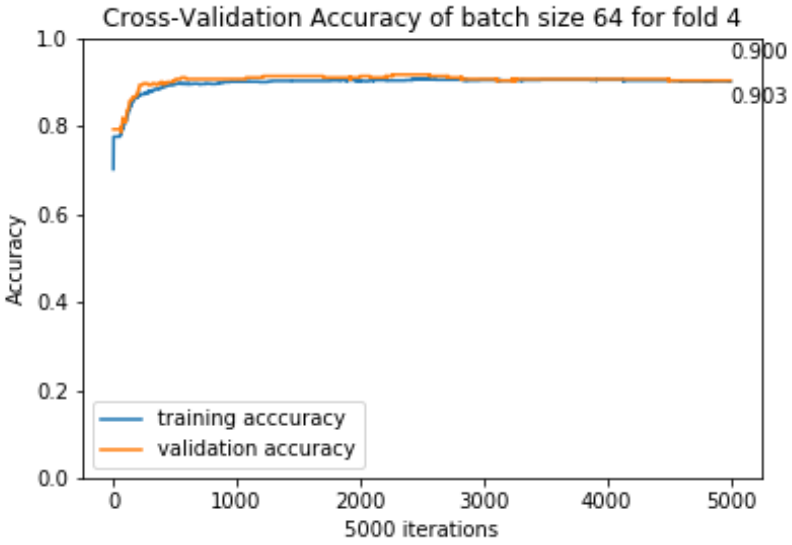

Figure A2.25: Accuracy for Fold 5 (Batch Size 64)

5.1.3 Question 3

5 Hidden-Layer Neurons

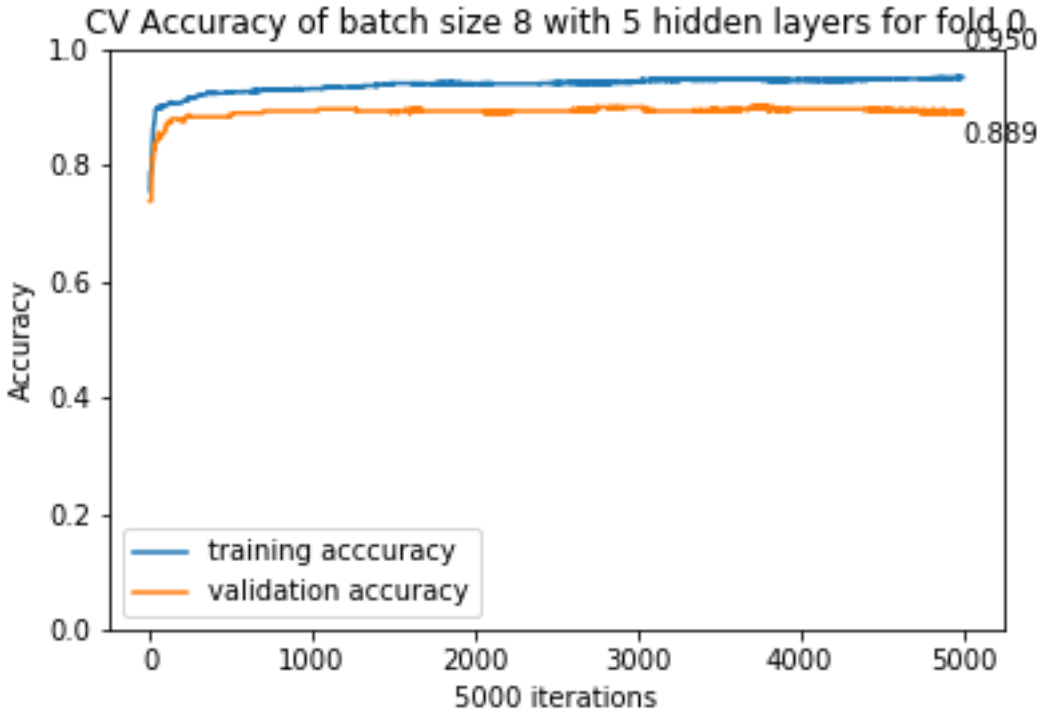

Figure A3.1: Accuracy for Fold 1 (5 Neurons)

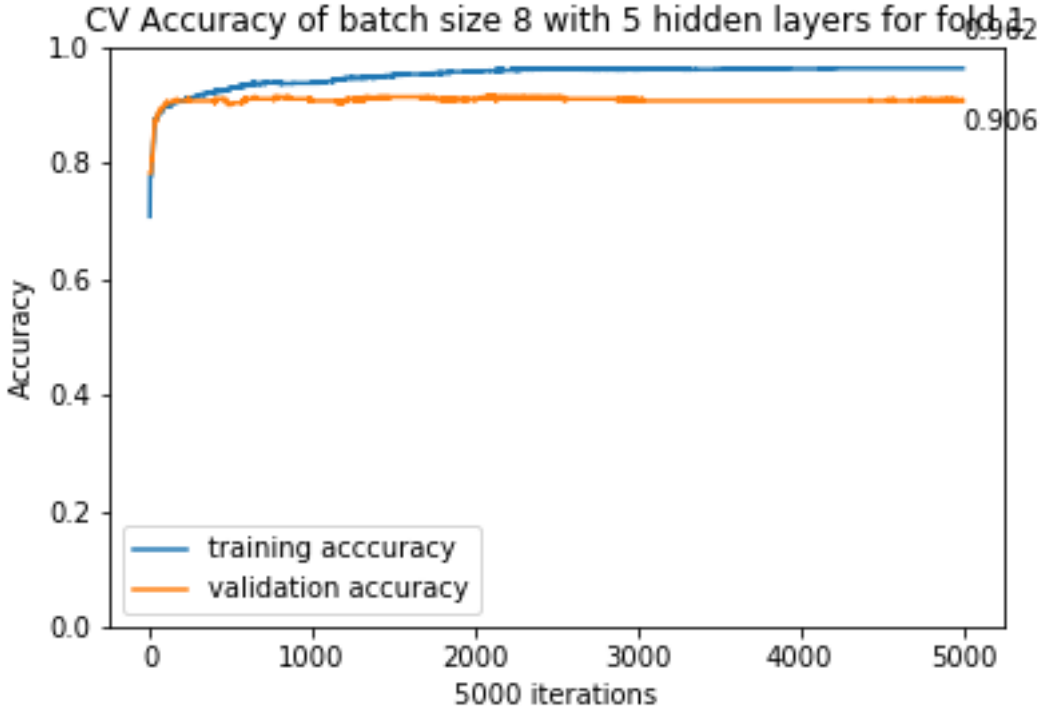

Figure A3.2: Accuracy for Fold 2 (5 Neurons)

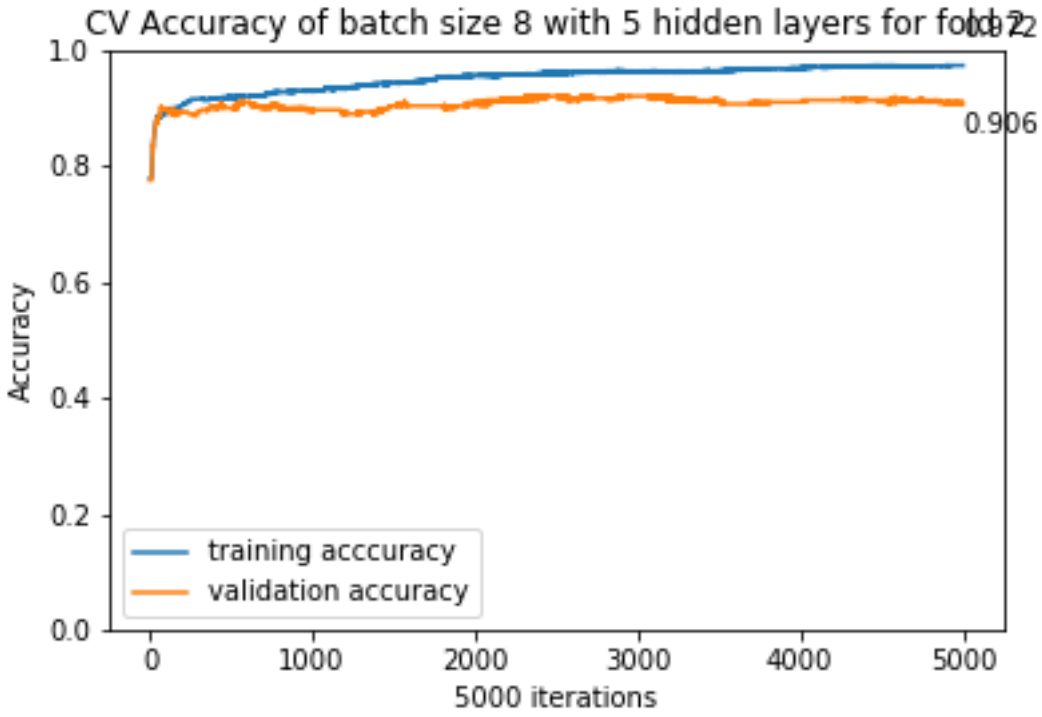

Figure A3.3: Accuracy for Fold 3 (5 Neurons)

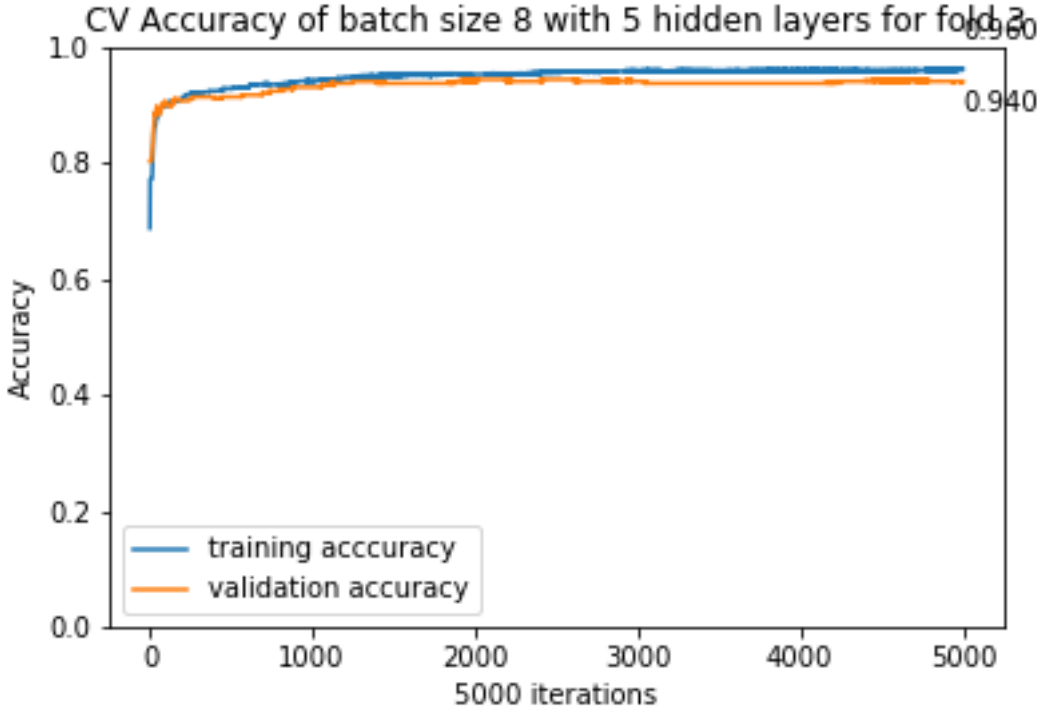

Figure A3.4: Accuracy for Fold 4 (5 Neurons)

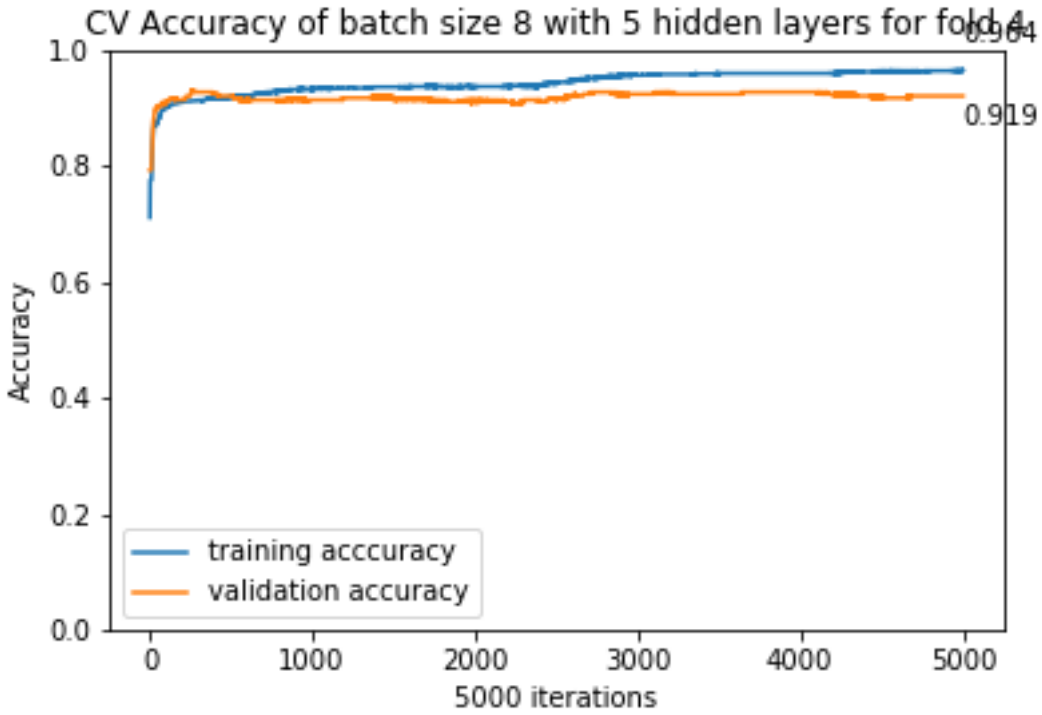

Figure A3.5: Accuracy for Fold 5 (5 Neurons)

10 Hidden-Layer Neurons

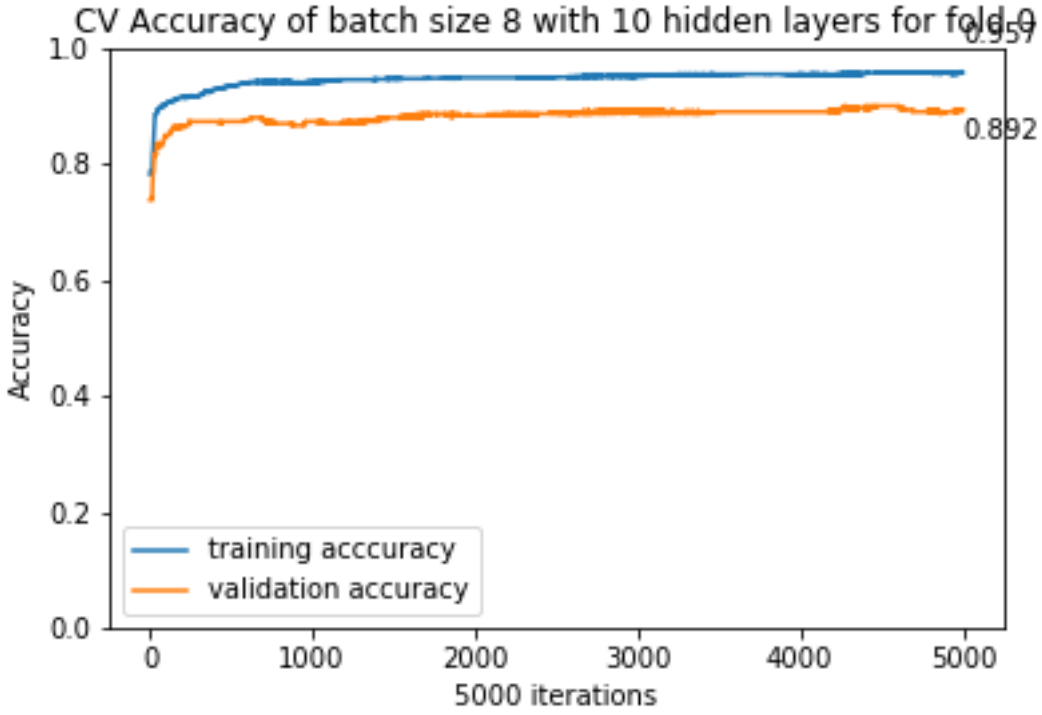

Figure A3.6: Accuracy for Fold 1 (10 Neurons)

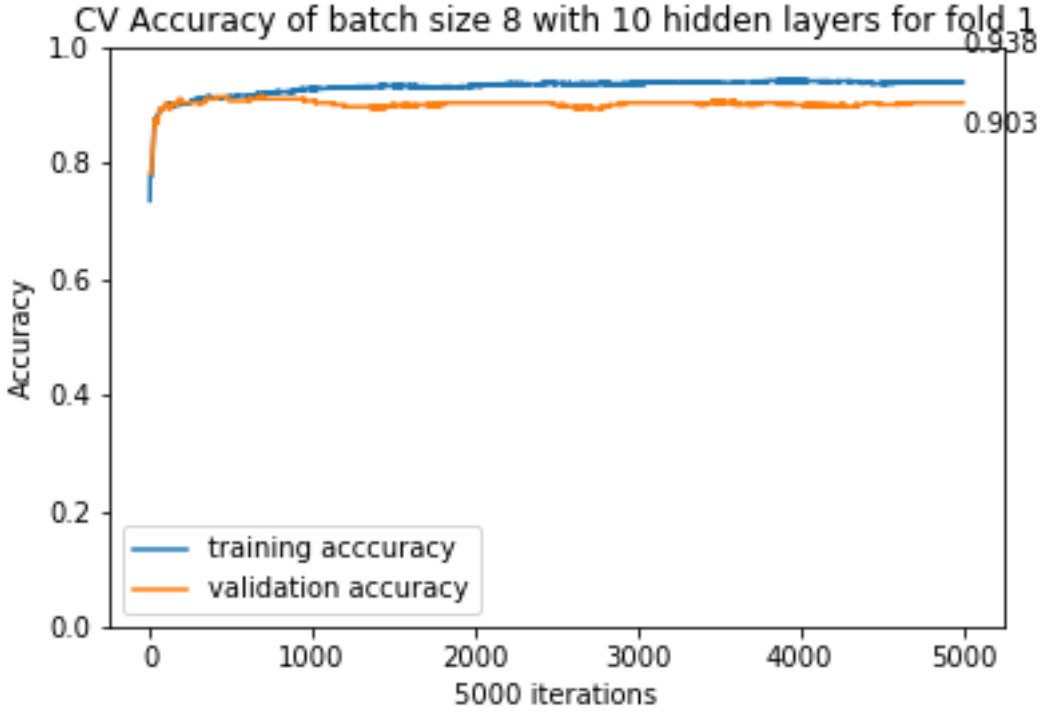

Figure A3.7: Accuracy for Fold 2 (10 Neurons)

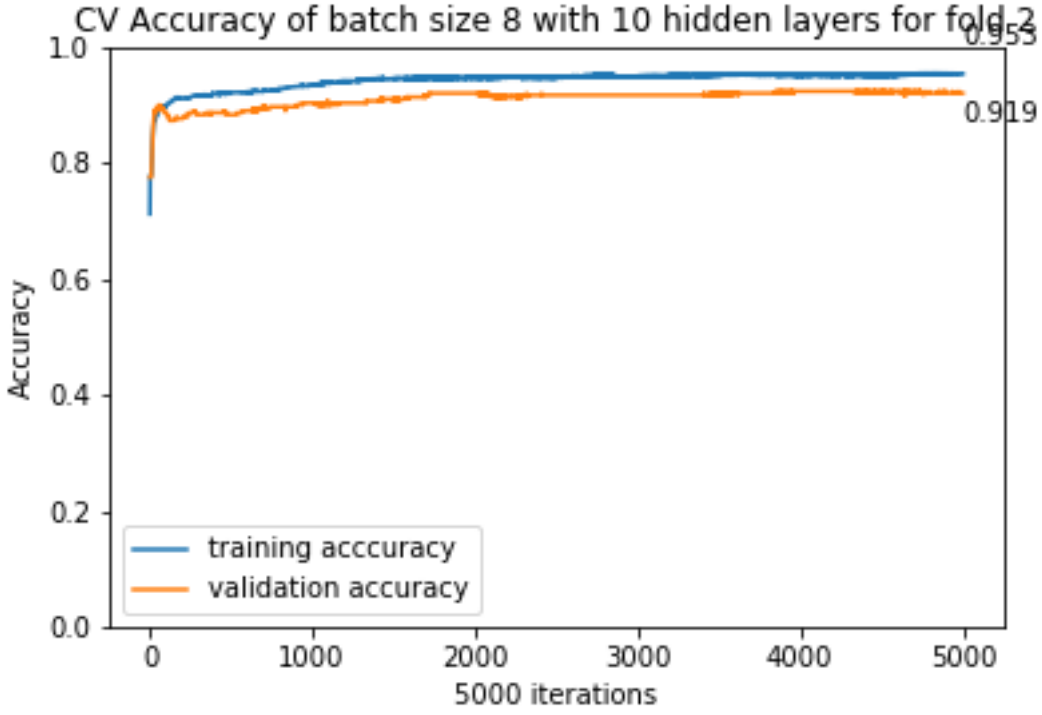

Figure A3.8: Accuracy for Fold 3 (10 Neurons)

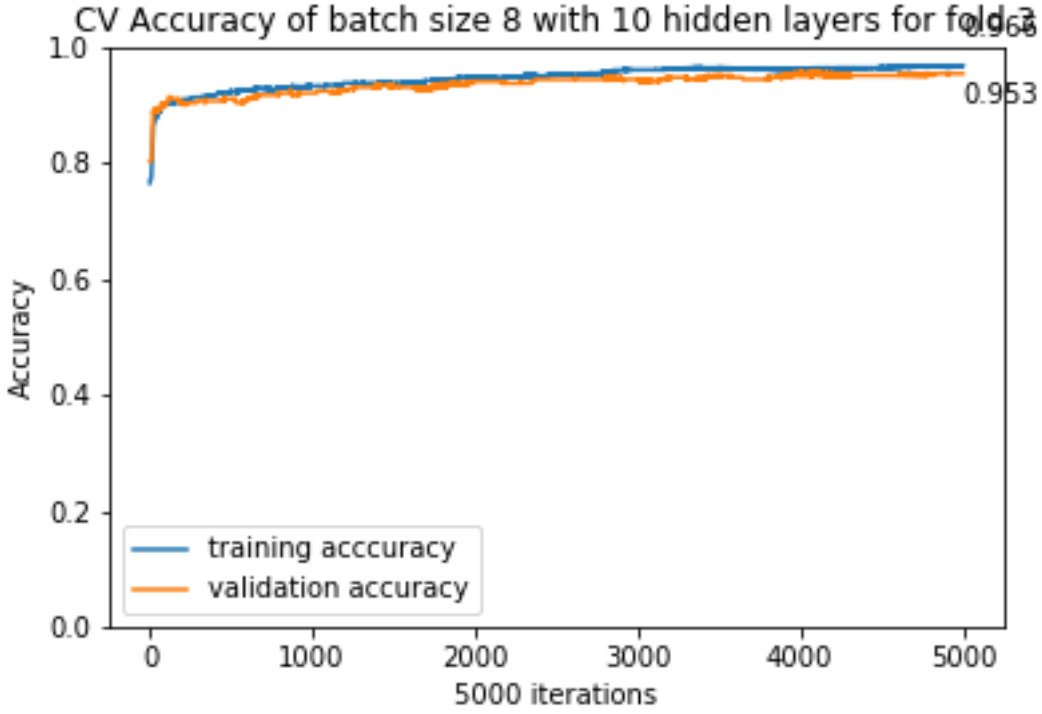

Figure A3.9: Accuracy for Fold 4 (10 Neurons)

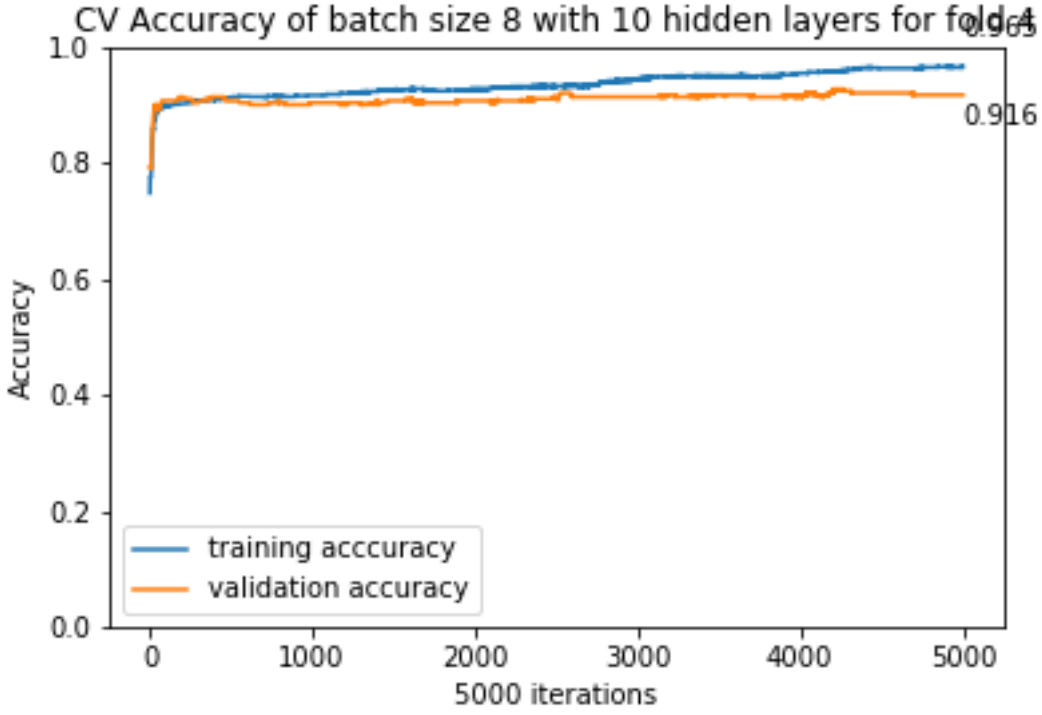

Figure A3.10: Accuracy for Fold 5 (10 Neurons)

15 Hidden-Layer Neurons

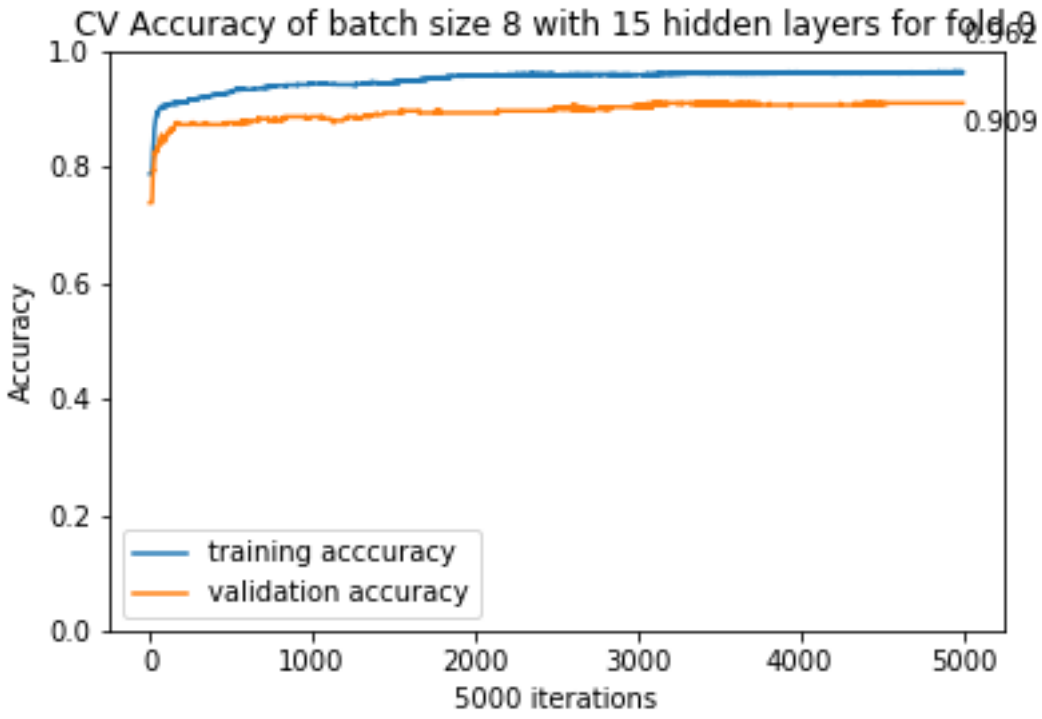

Figure A3.11: Accuracy for Fold 1 (15 Neurons)

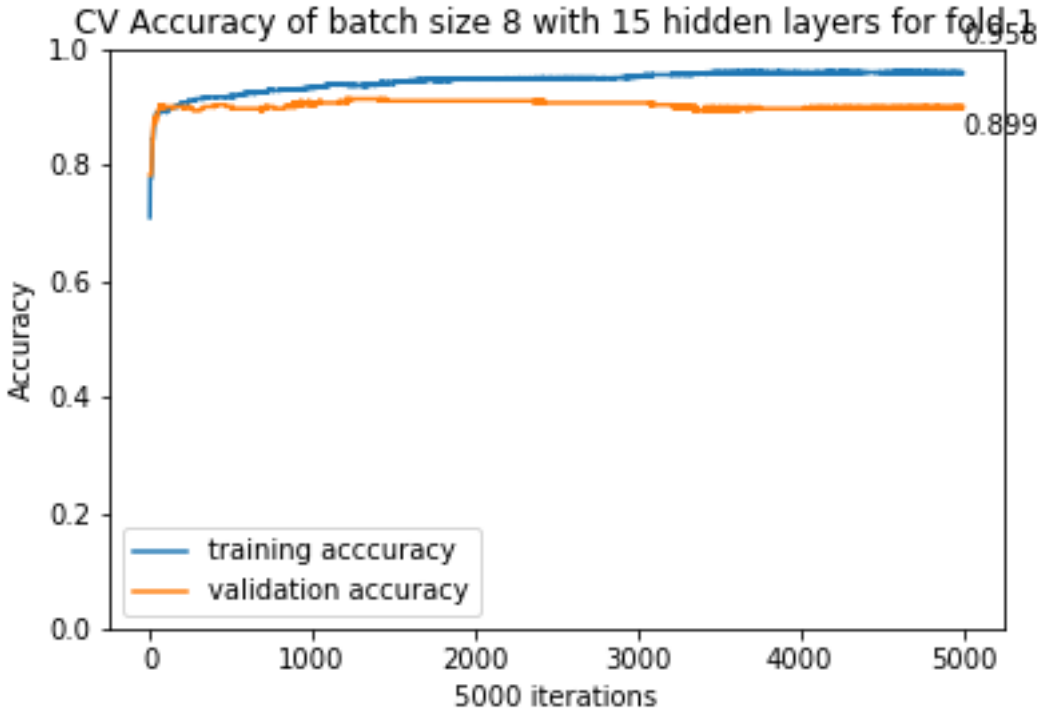

Figure A3.12: Accuracy for Fold 2 (15 Neurons)

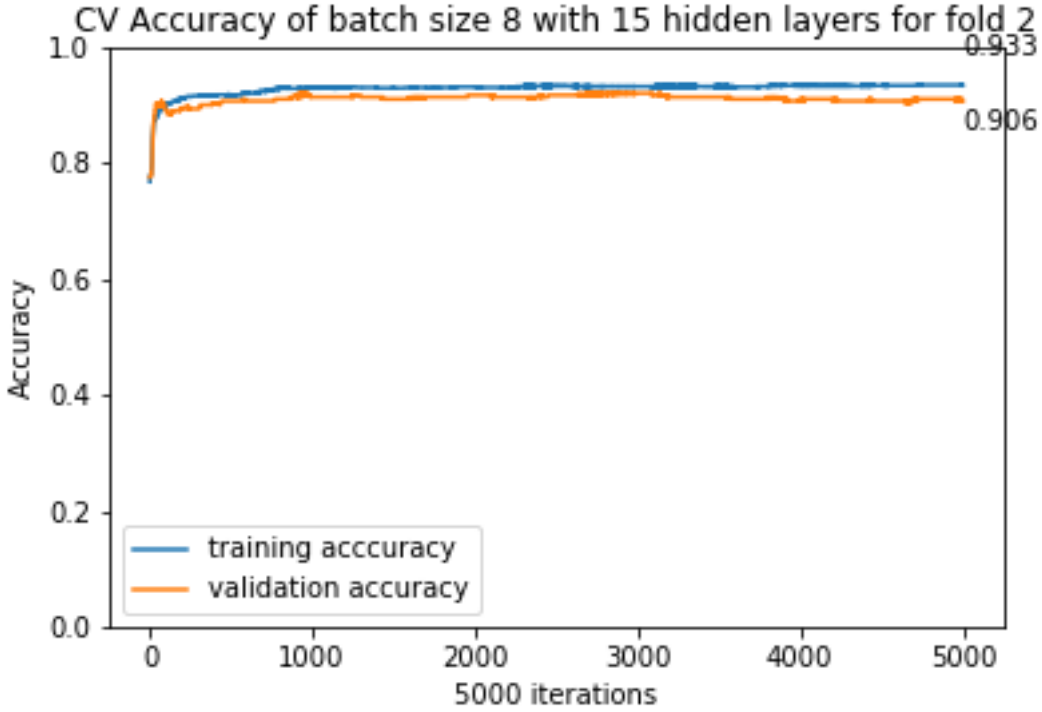

Figure A3.13: Accuracy for Fold 3 (15 Neurons)

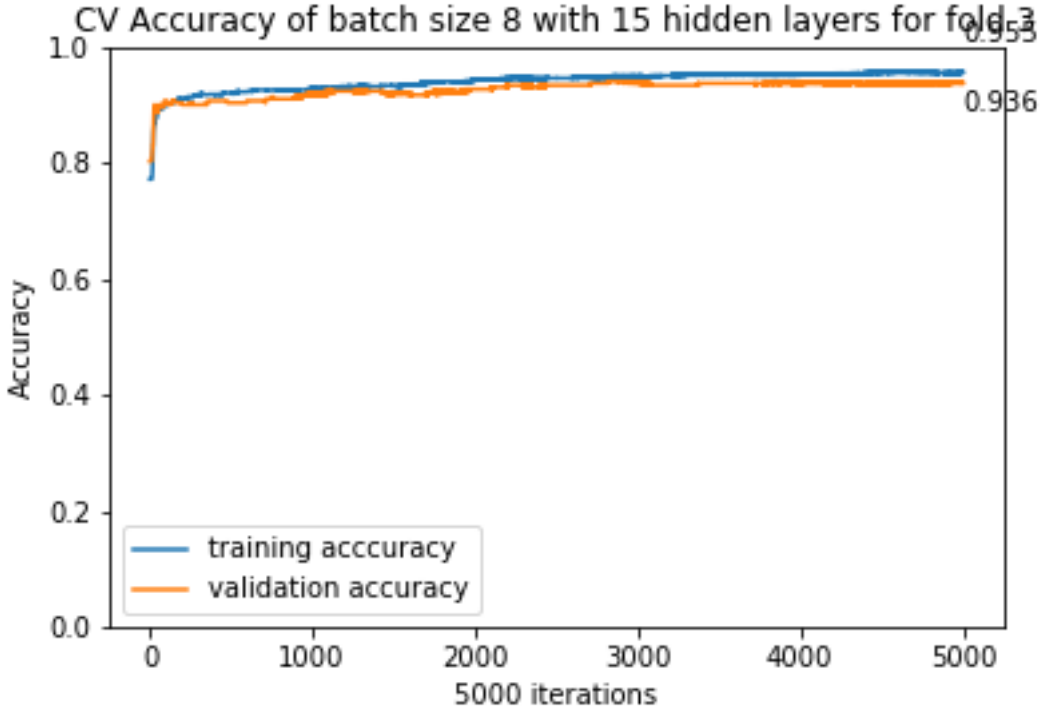

Figure A3.14: Accuracy for Fold 4 (15 Neurons)

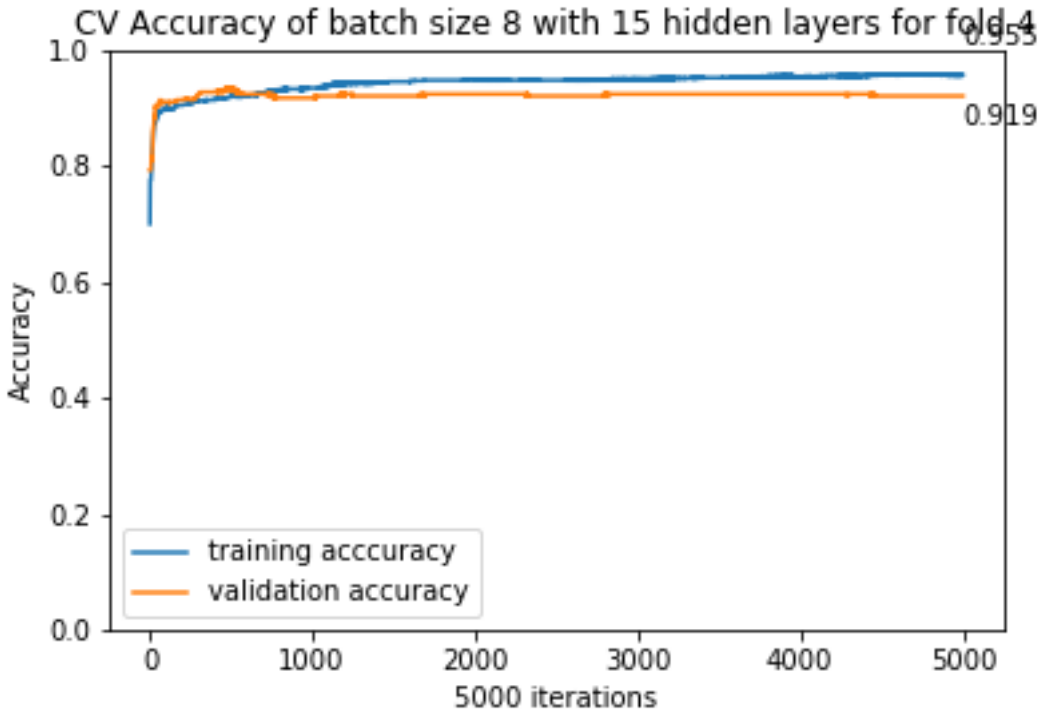

Figure A3.15: Accuracy for Fold 5 (15 Neurons)

20 Hidden-Layer Neurons

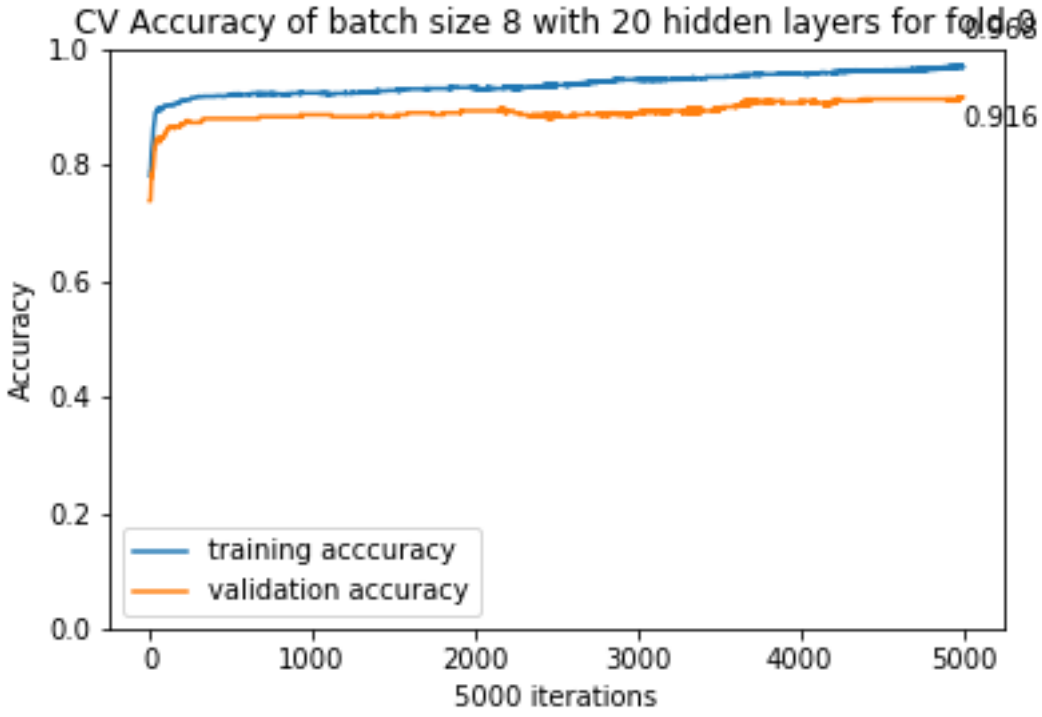

Figure A3.16: Accuracy for Fold 1 (20 Neurons)

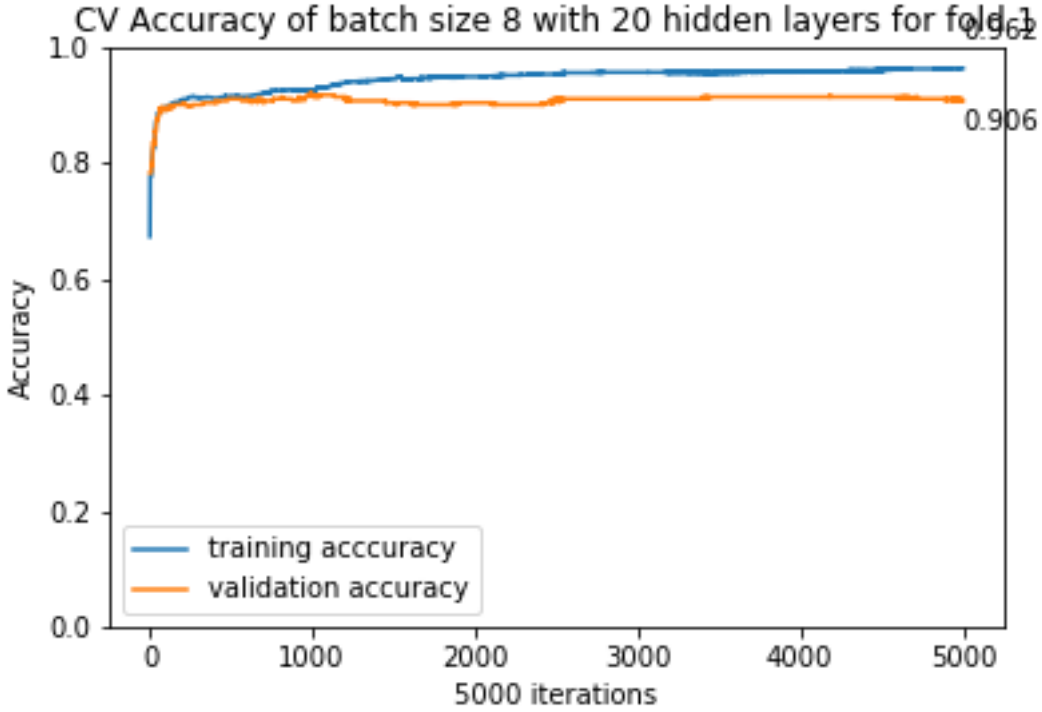

Figure A3.17: Accuracy for Fold 2 (20 Neurons)

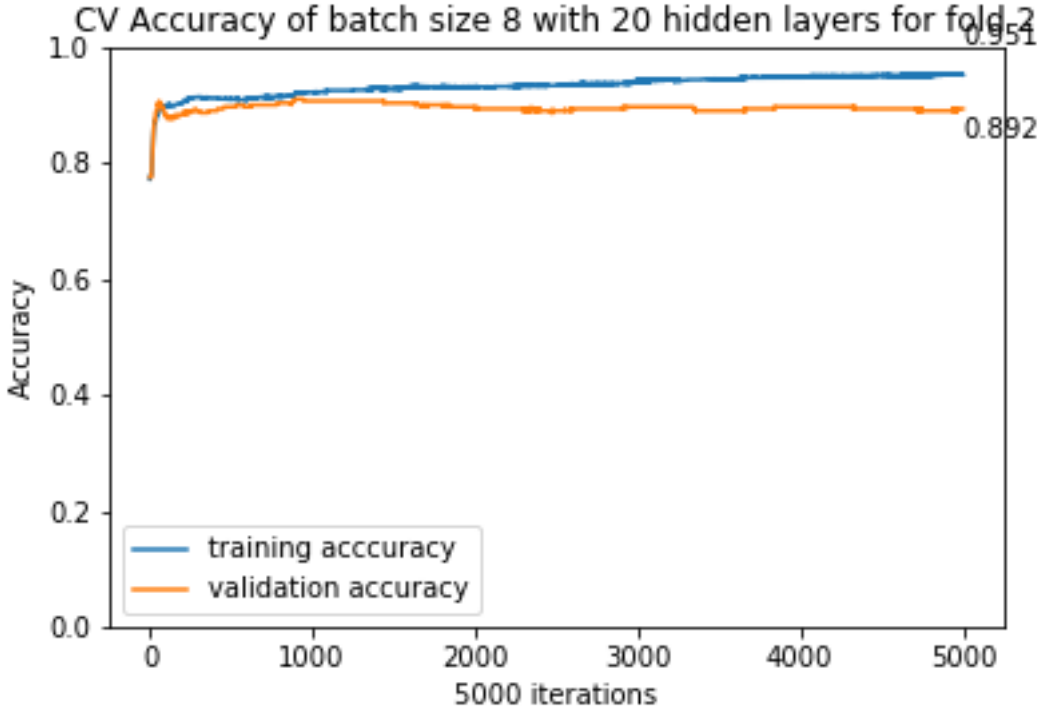

Figure A3.18: Accuracy for Fold 3 (20 Neurons)

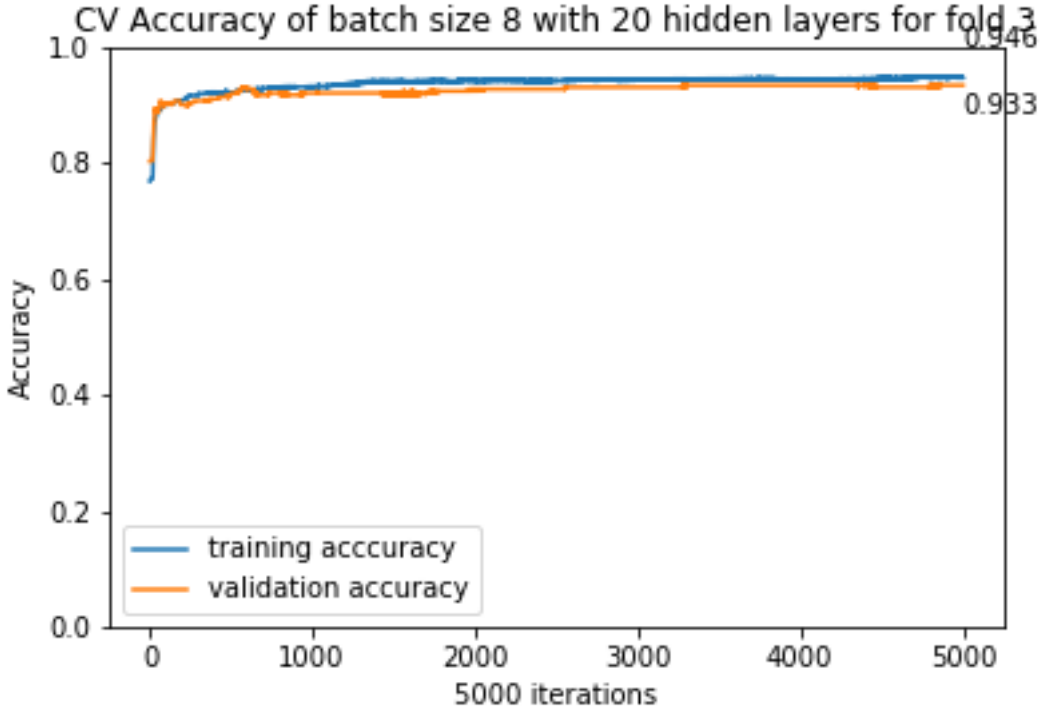

Figure A3.19: Accuracy for Fold 4 (20 Neurons)

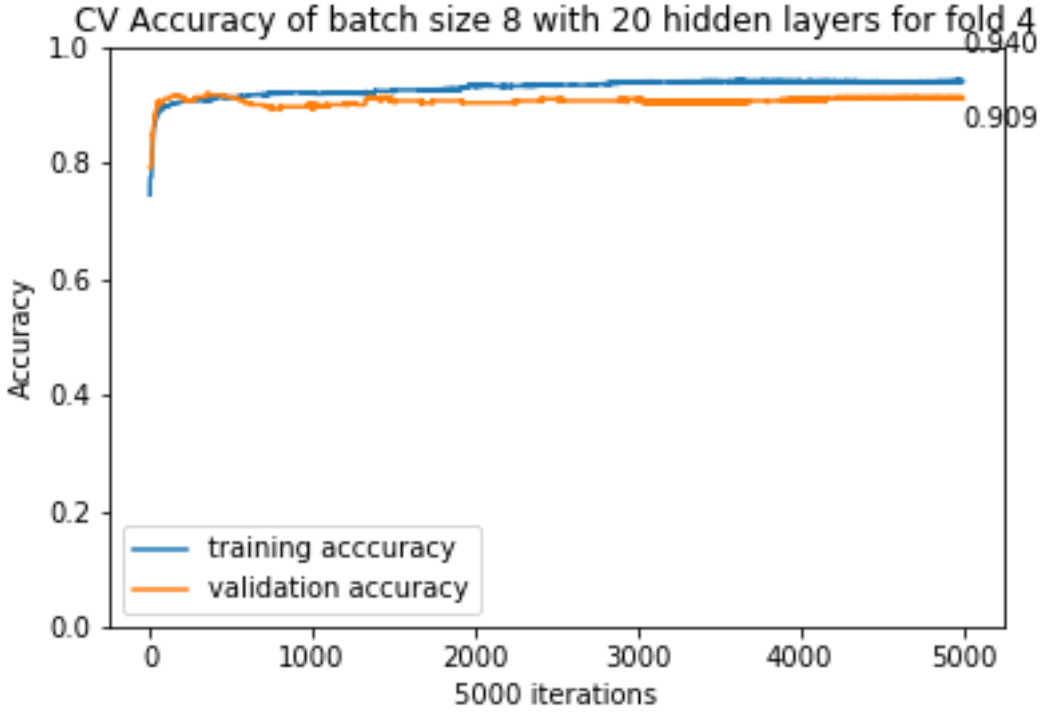

Figure A3.20: Accuracy for Fold 5 (20 Neurons)

25 Hidden-Layer Neurons

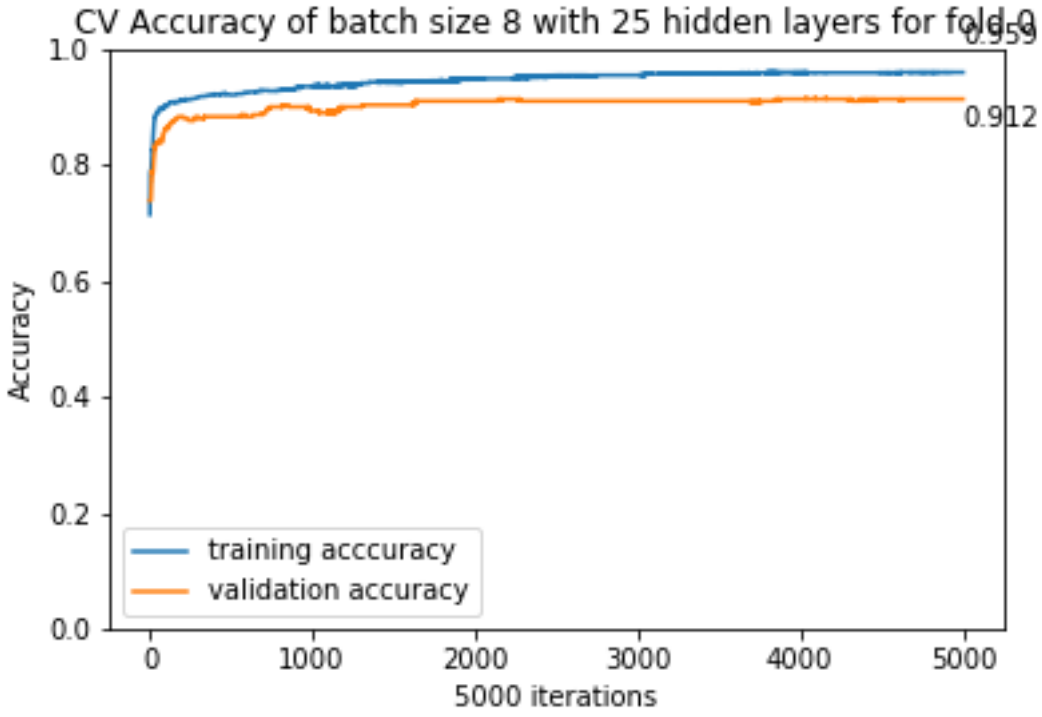

Figure A3.21: Accuracy for Fold 1 (25 Neurons)

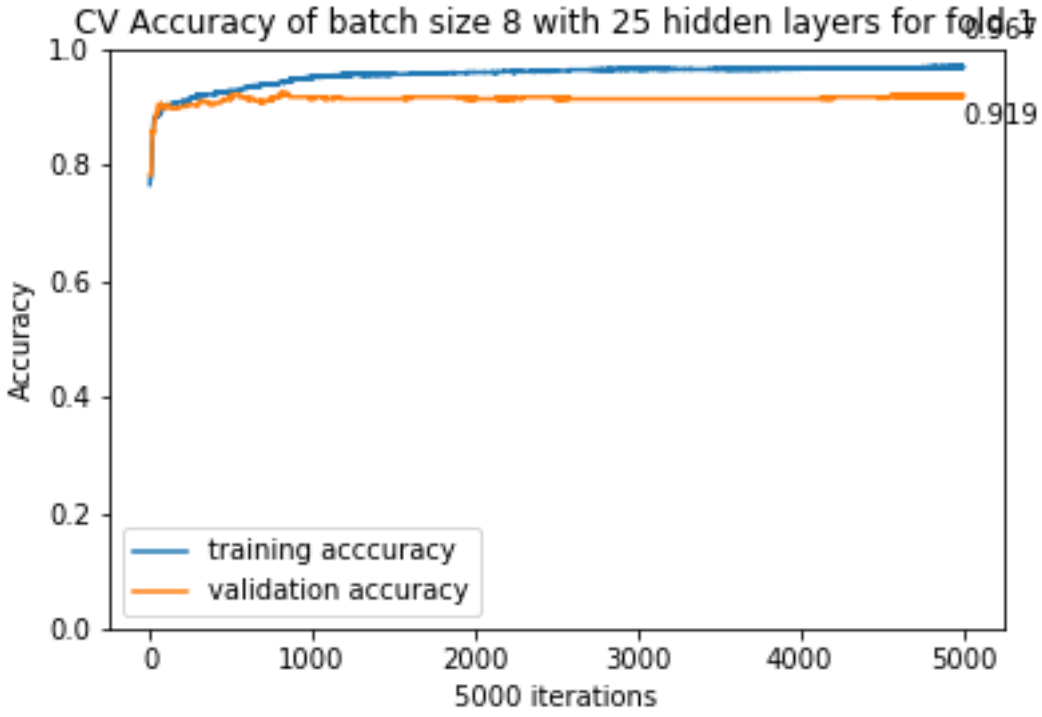

Figure A3.22: Accuracy for Fold 2 (25 Neurons)

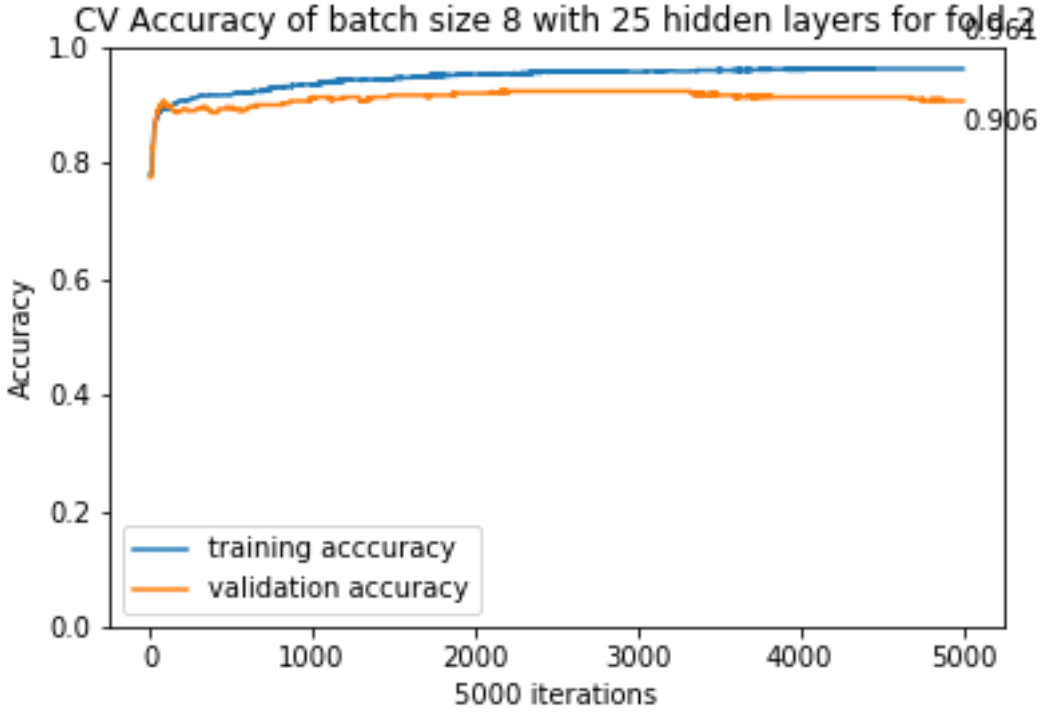

Figure A3.23: Accuracy for Fold 3 (25 Neurons)

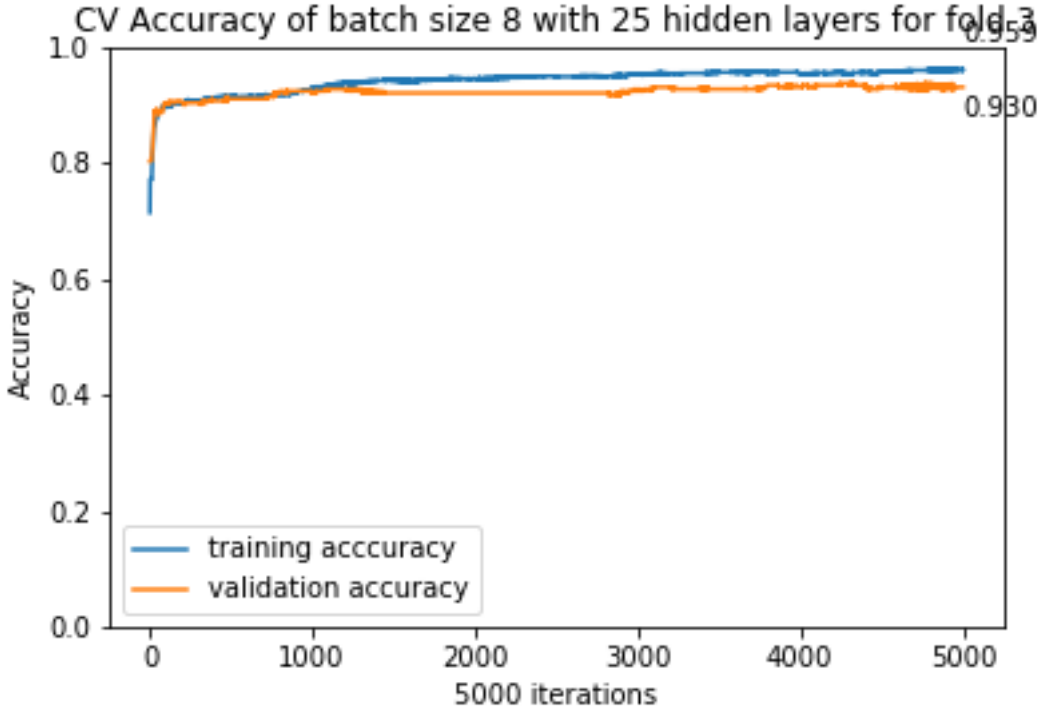

Figure A3.24: Accuracy for Fold 4 (25 Neurons)

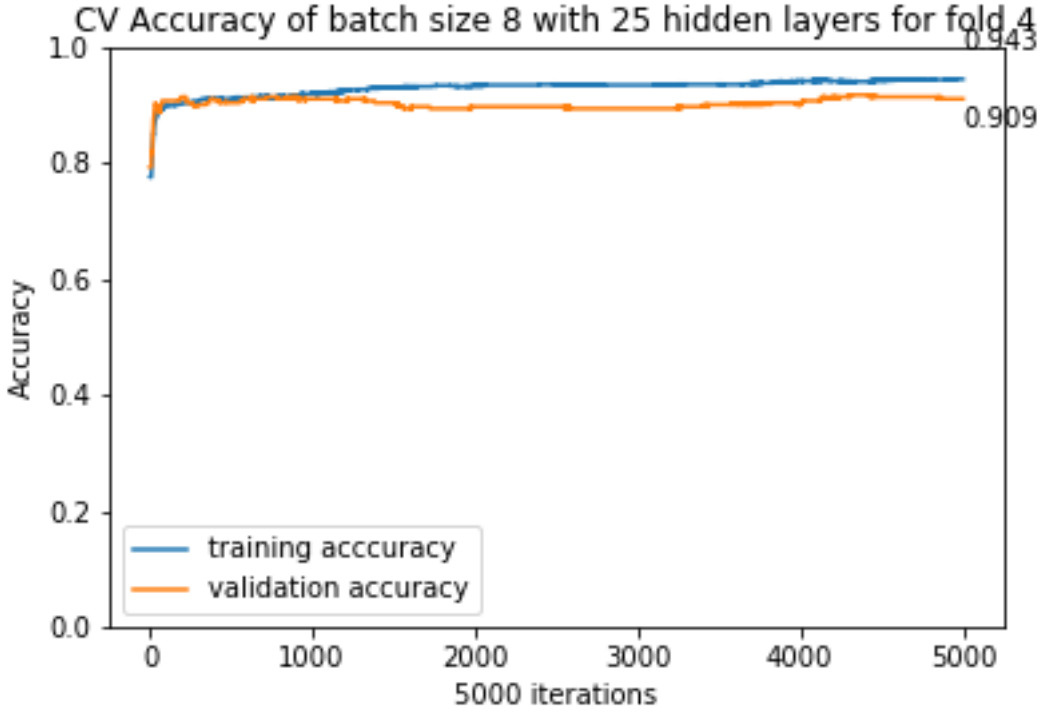

Figure A3.25: Accuracy for Fold 5 (25 Neurons)

5.1.4 Question 4

Decay Parameter 0

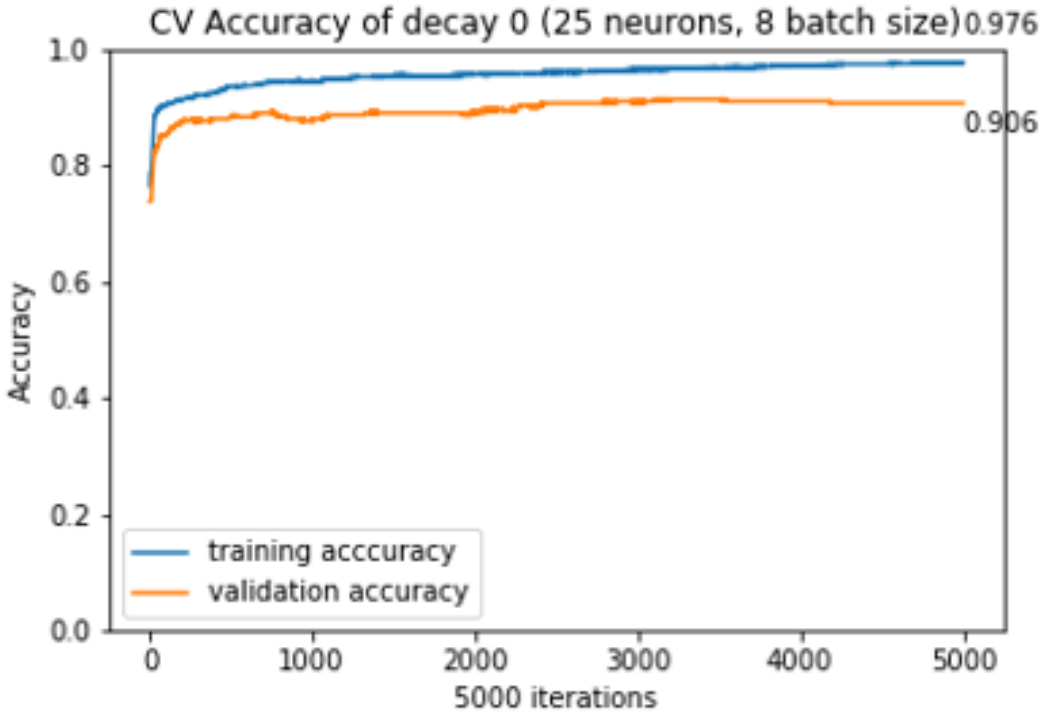

Figure A4.1: Accuracy for Fold 1 (Decay Parameter 0)

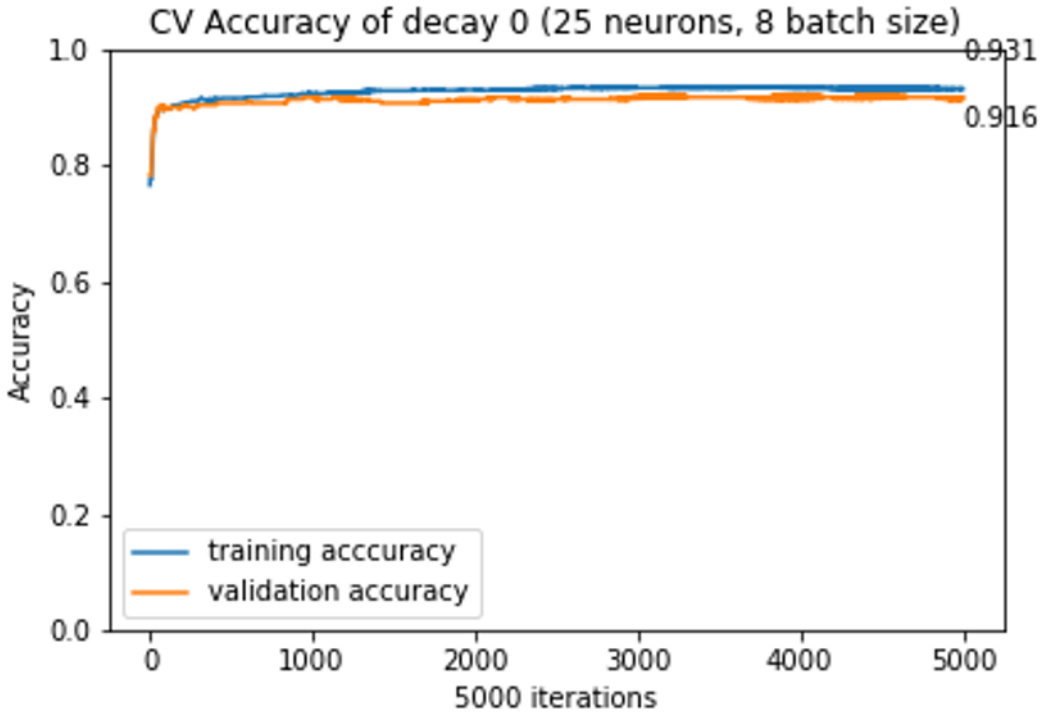

Figure A4.2: Accuracy for Fold 2 (Decay Parameter 0)

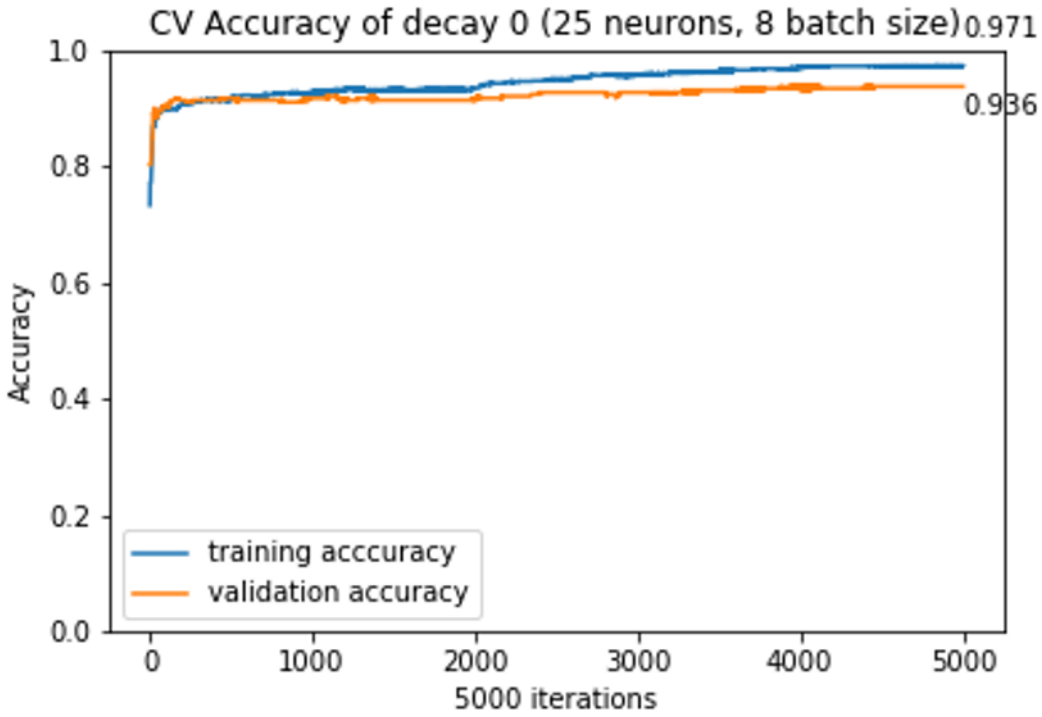

Figure A4.3: Accuracy for Fold 3 (Decay Parameter 0)

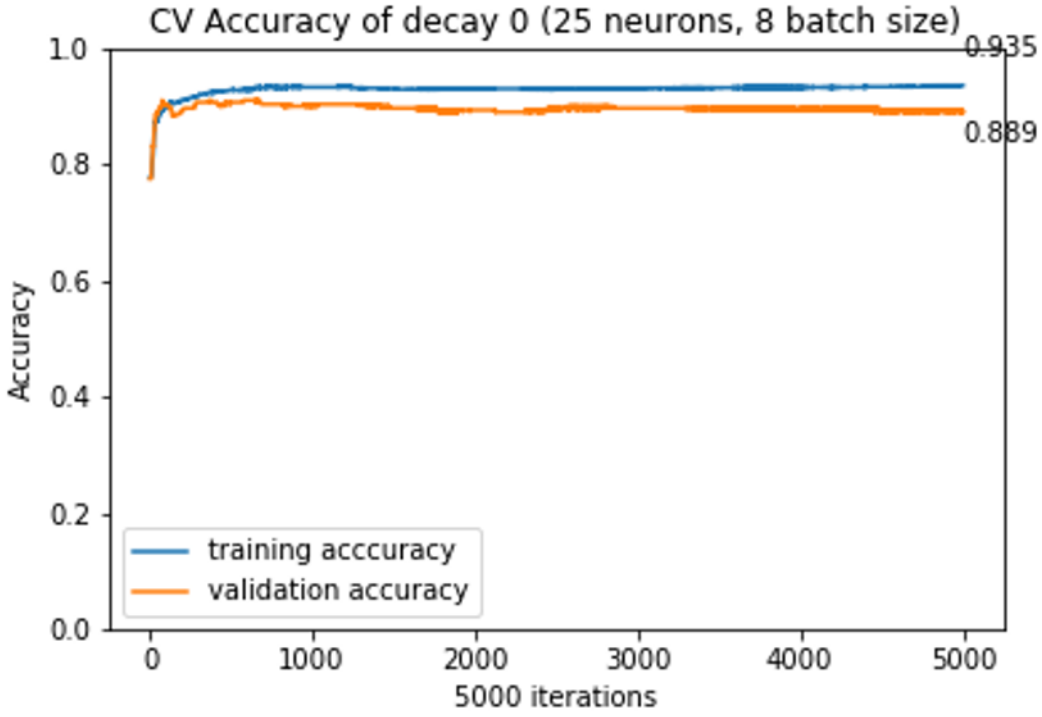

Figure A4.4: Accuracy for Fold 4 (Decay Parameter 0)

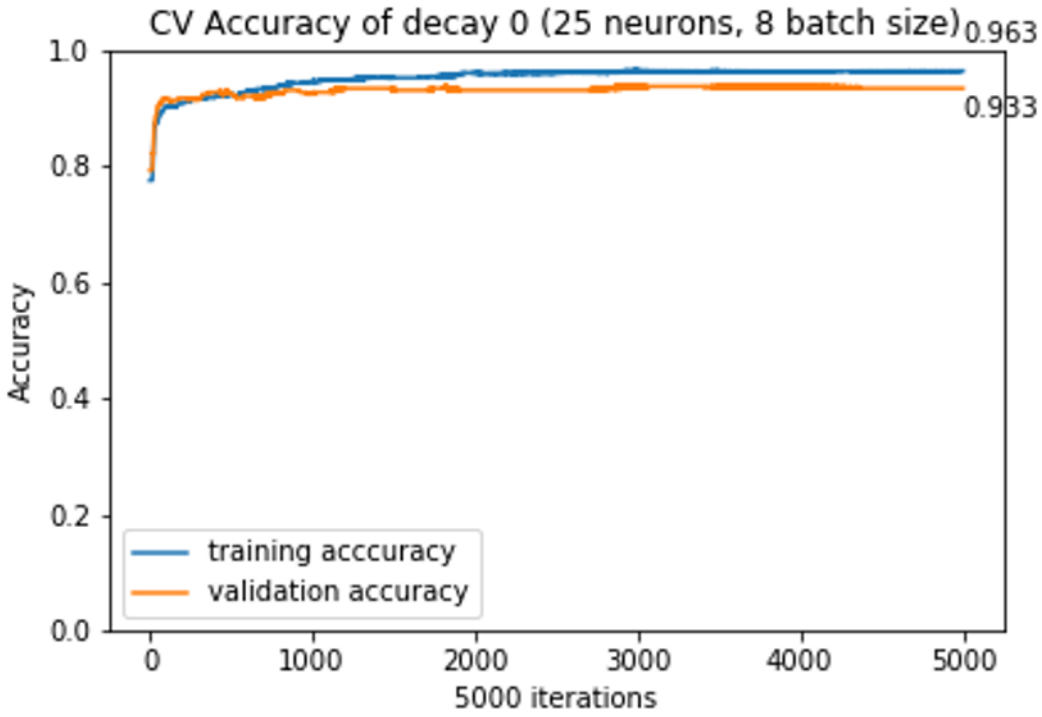

Figure A4.5: Accuracy for Fold 5 (Decay Parameter 0)

Decay Parameter 1e-03

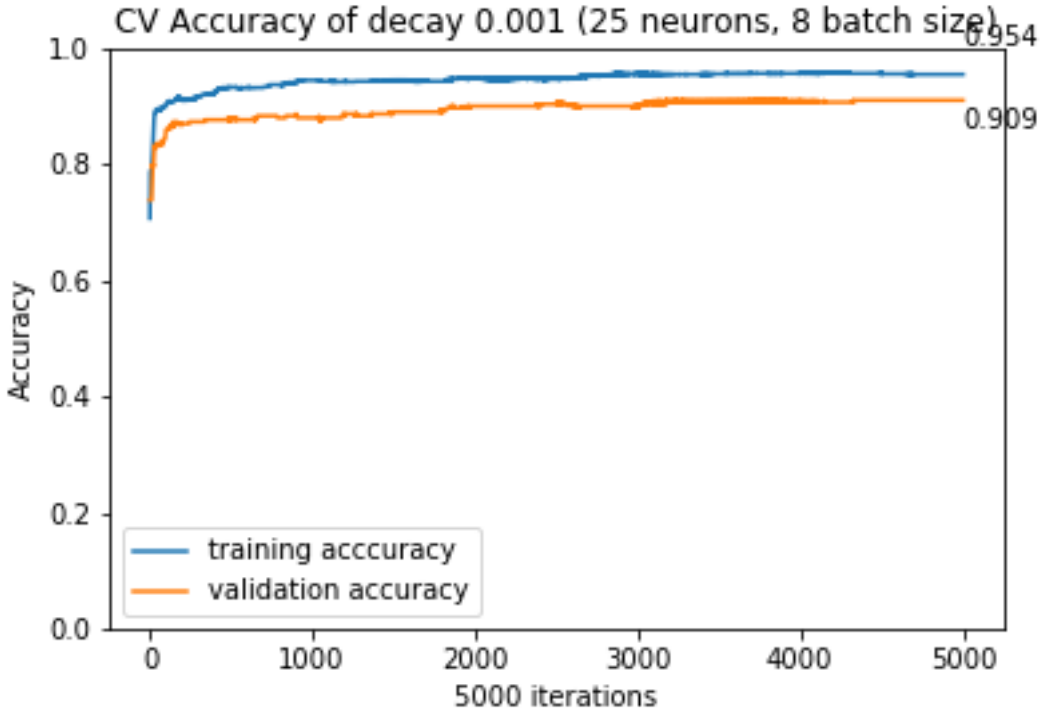

Figure A4.6: Accuracy for Fold 1 (Decay Parameter 10e-3)

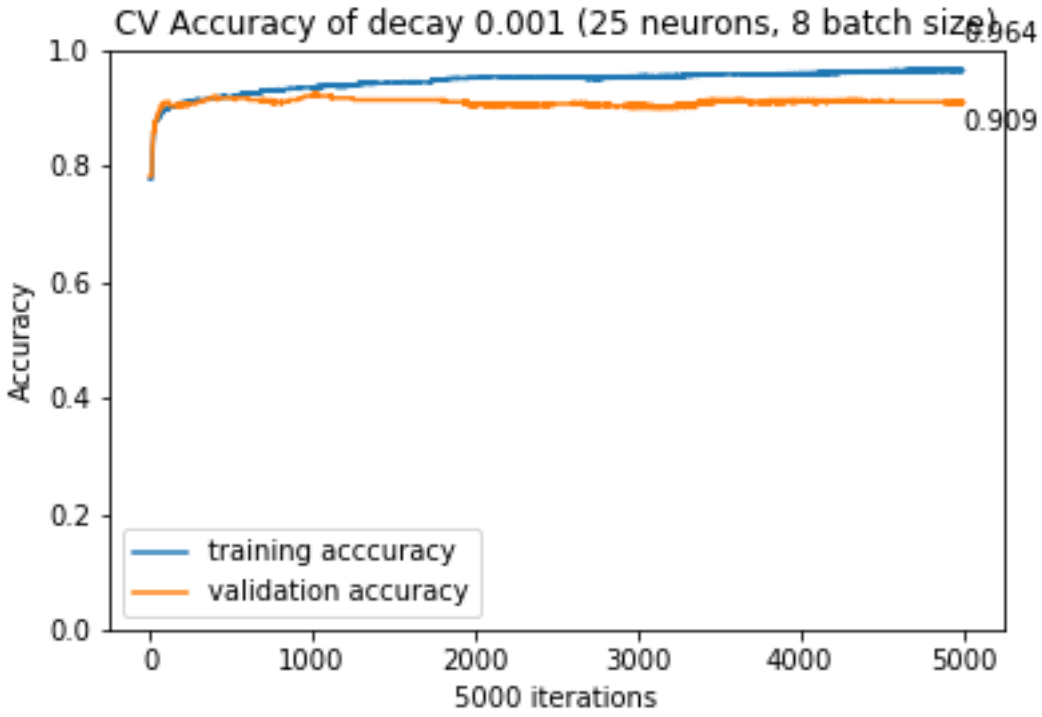

Figure A4.7: Accuracy for Fold 2 (Decay Parameter 10e-3)

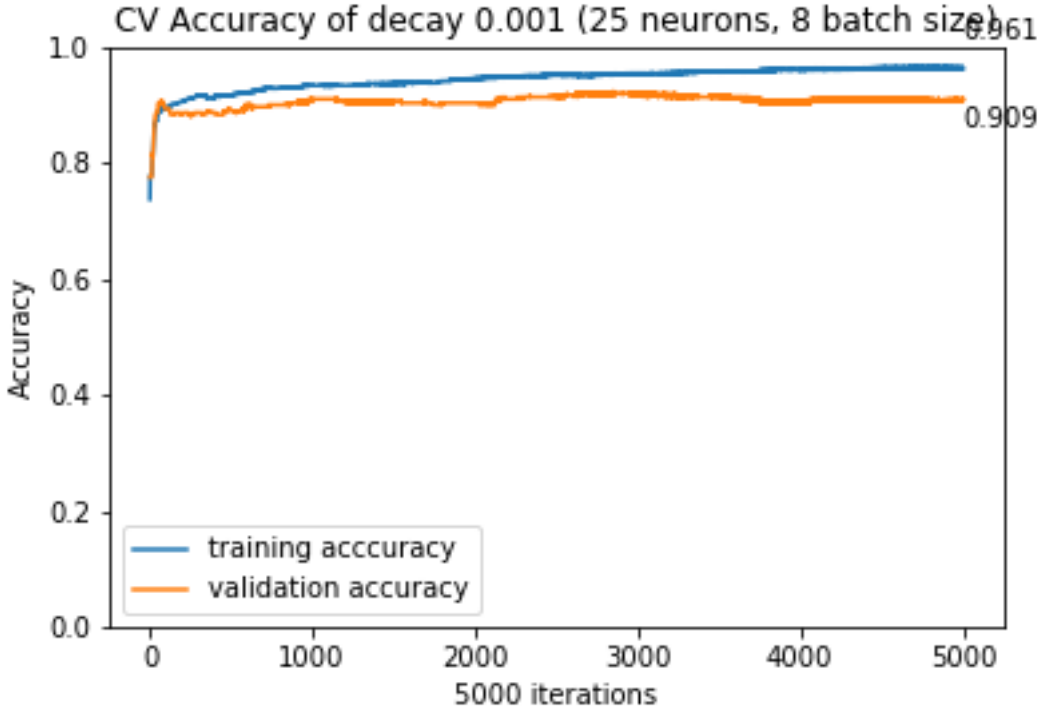

Figure A4.8: Accuracy for Fold 3 (Decay Parameter 10e-3)

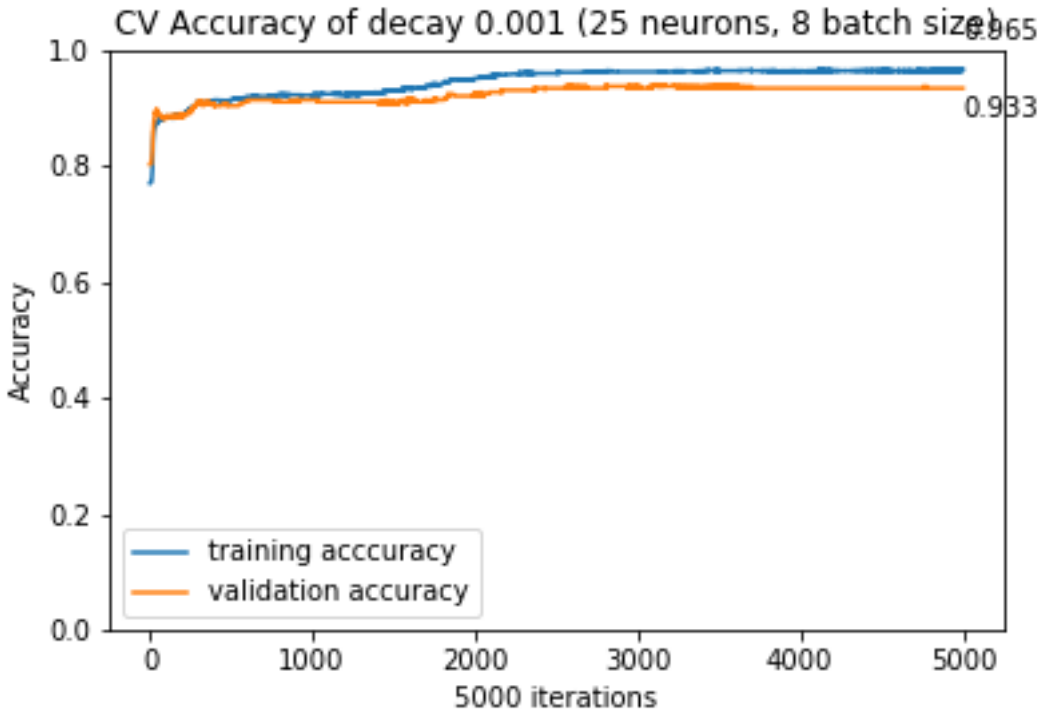

Figure A4.9: Accuracy for Fold 4 (Decay Parameter 10e-3)

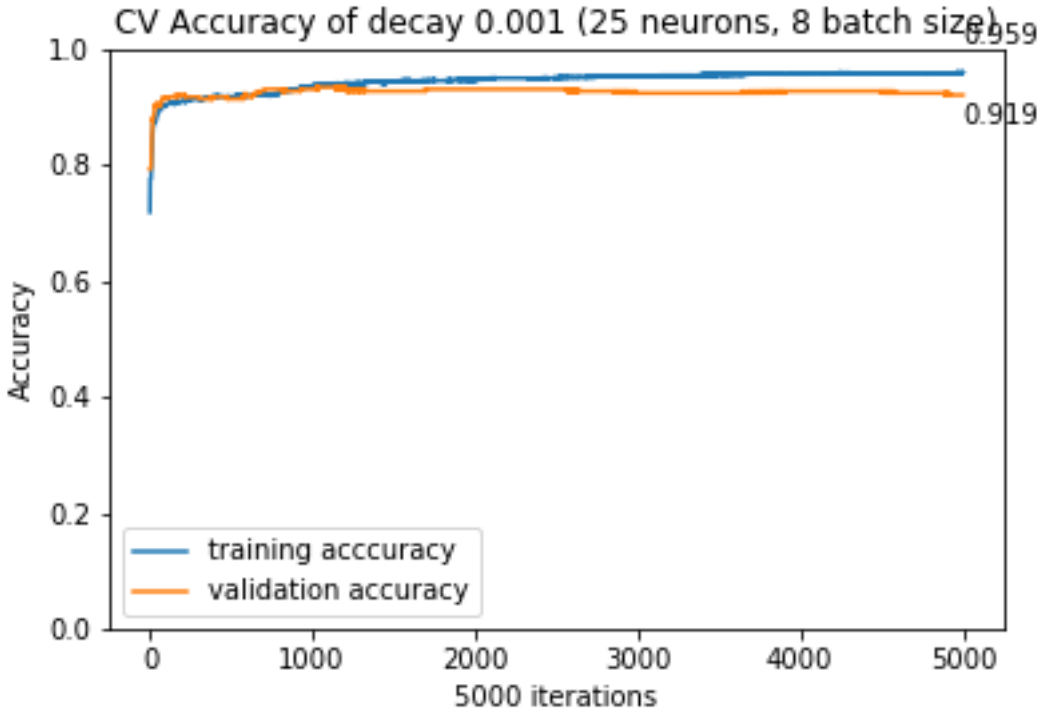

Figure A4.10: Accuracy for Fold 5 (Decay Parameter 10e-3)

Decay Parameter 1e-06

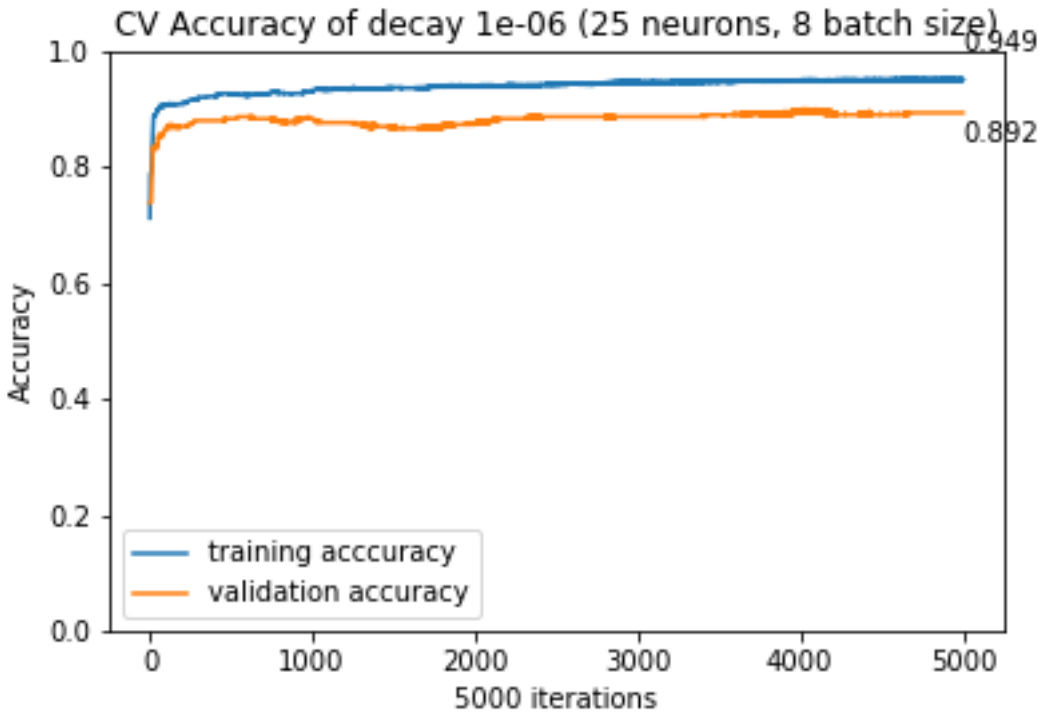

Figure A4.11: Accuracy for Fold 1 (Decay Parameter 10e-6)

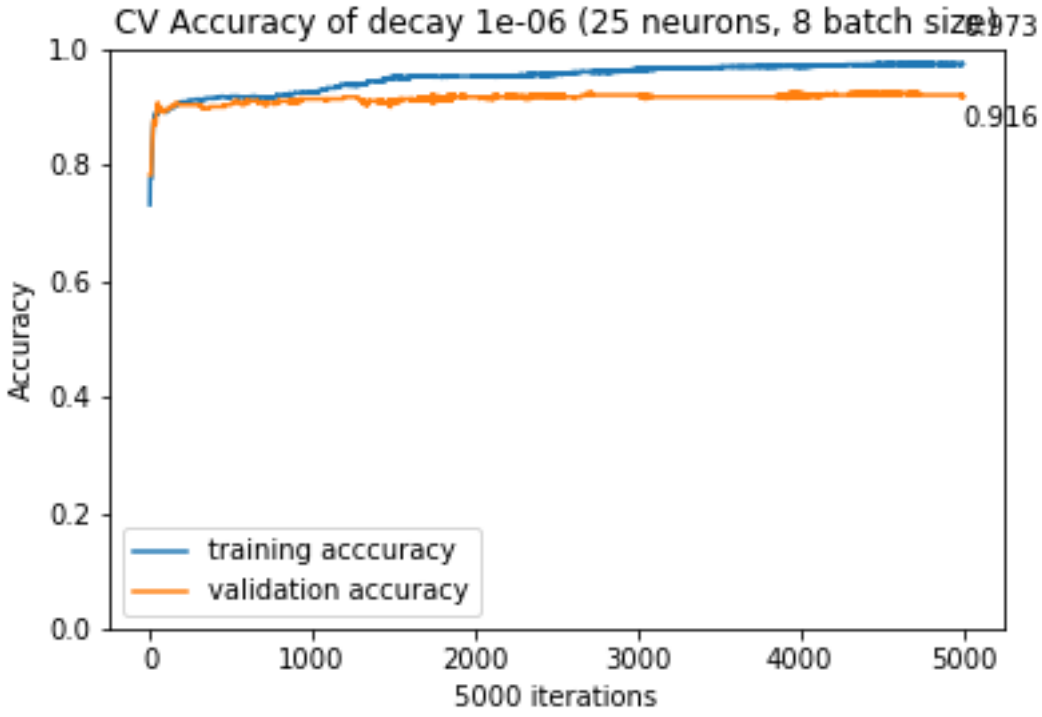

Figure A4.12: Accuracy for Fold 2 (Decay Parameter 10e-6)

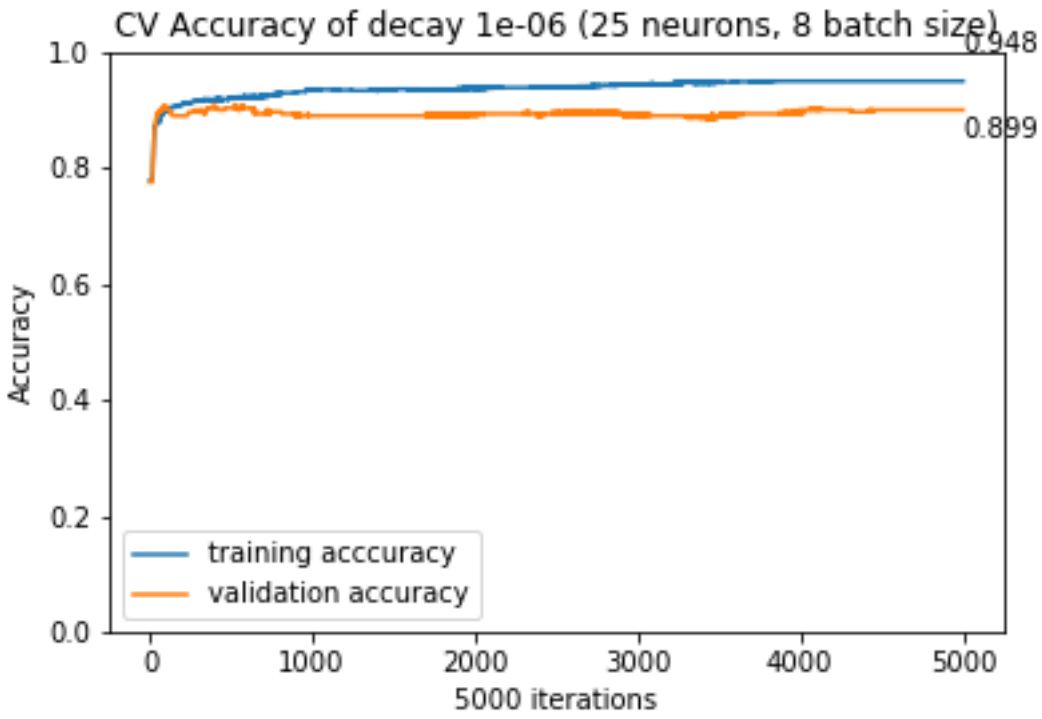

Figure A4.13: Accuracy for Fold 3 (Decay Parameter 10e-6)

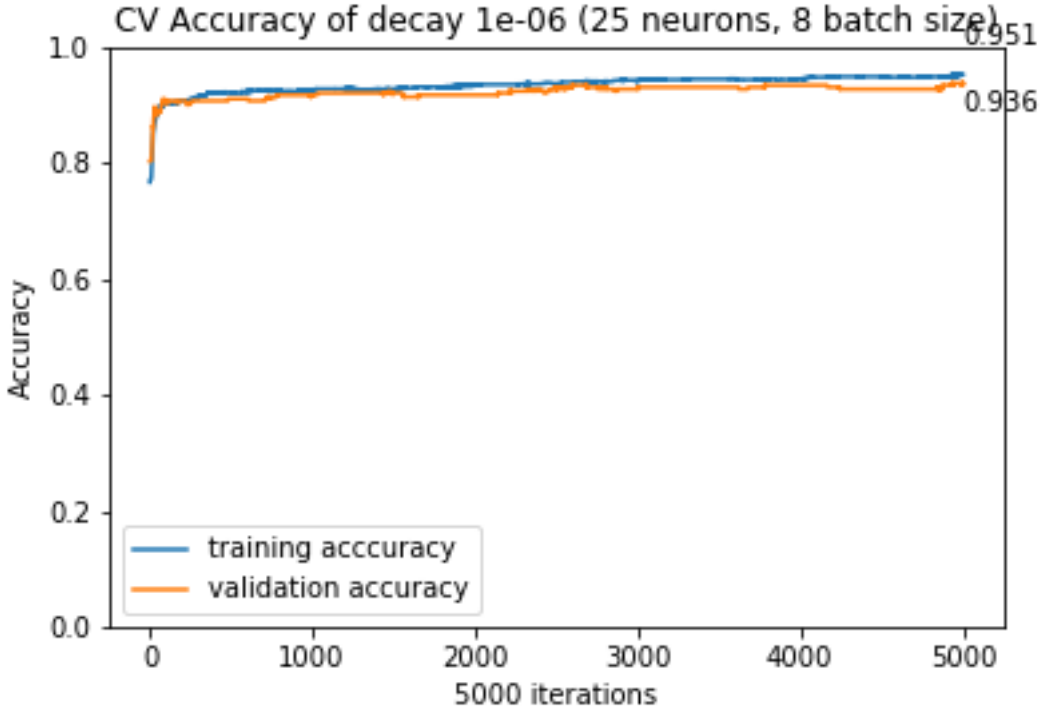

Figure A4.14: Accuracy for Fold 4 (Decay Parameter 10e-6)

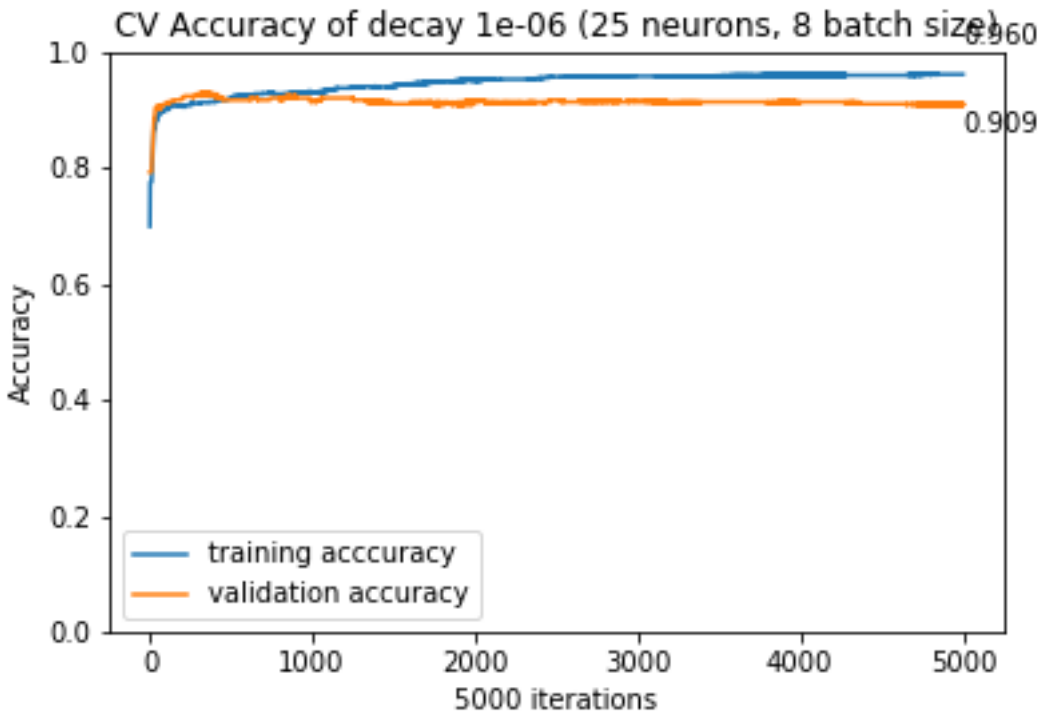

Figure A4.15: Accuracy for Fold 5 (Decay Parameter 10e-6)

Decay Parameter 1e-09

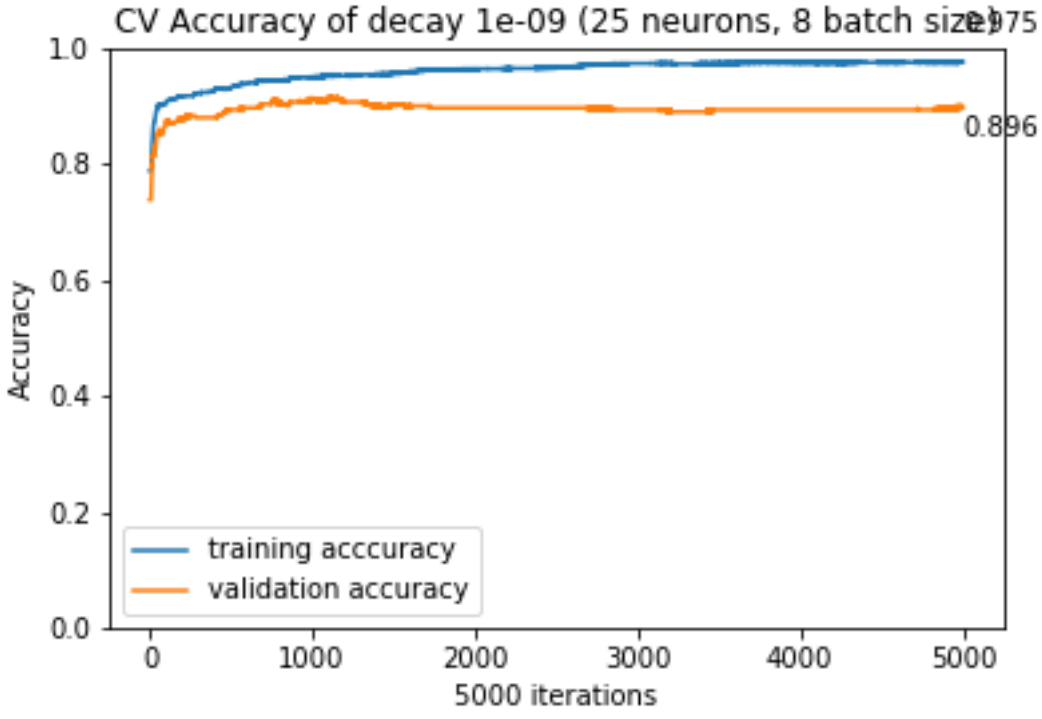

Figure A4.16: Accuracy for Fold 1 (Decay Parameter 10e-9)

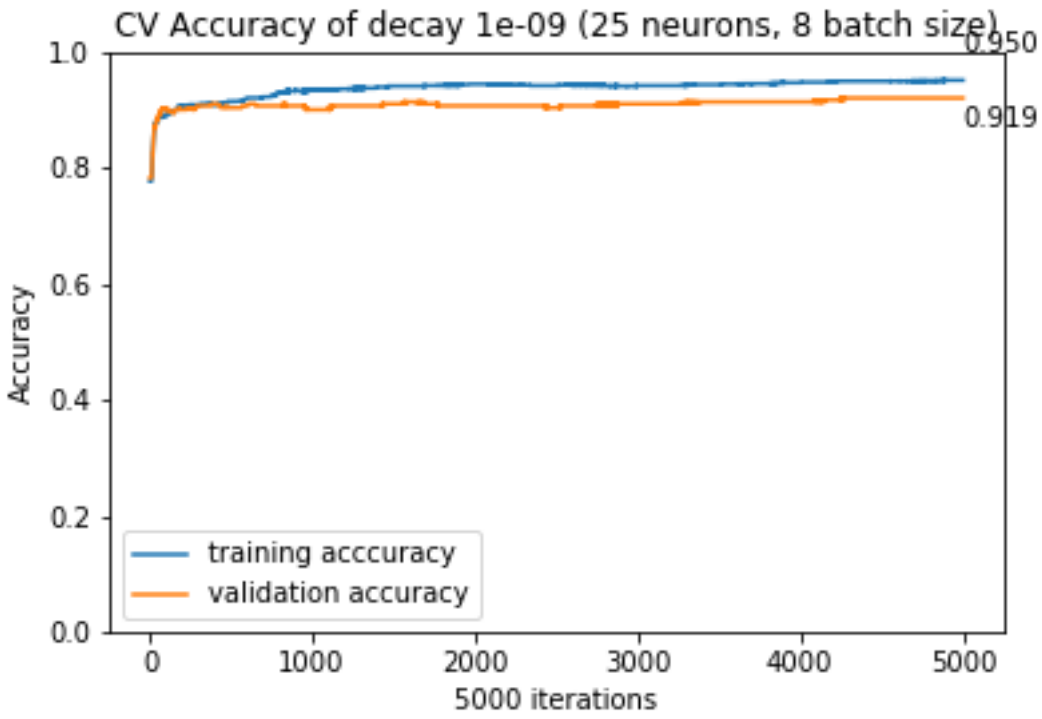

Figure A4.17: Accuracy for Fold 2 (Decay Parameter 10e-9)

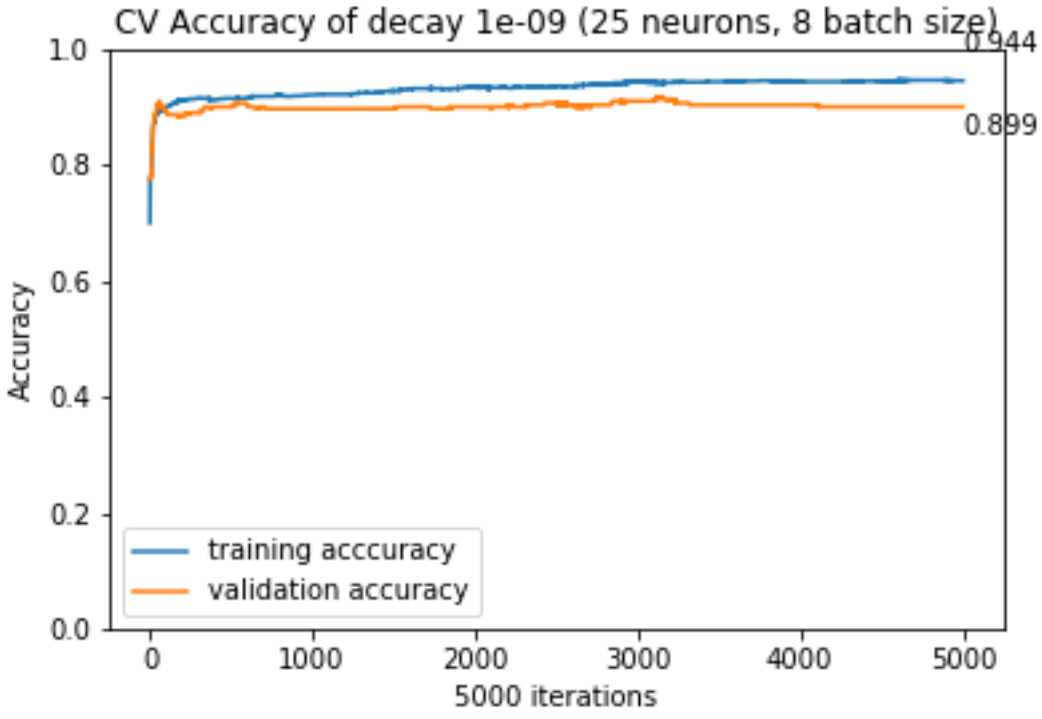

Figure A4.18: Accuracy for Fold 3 (Decay Parameter 10e-9)

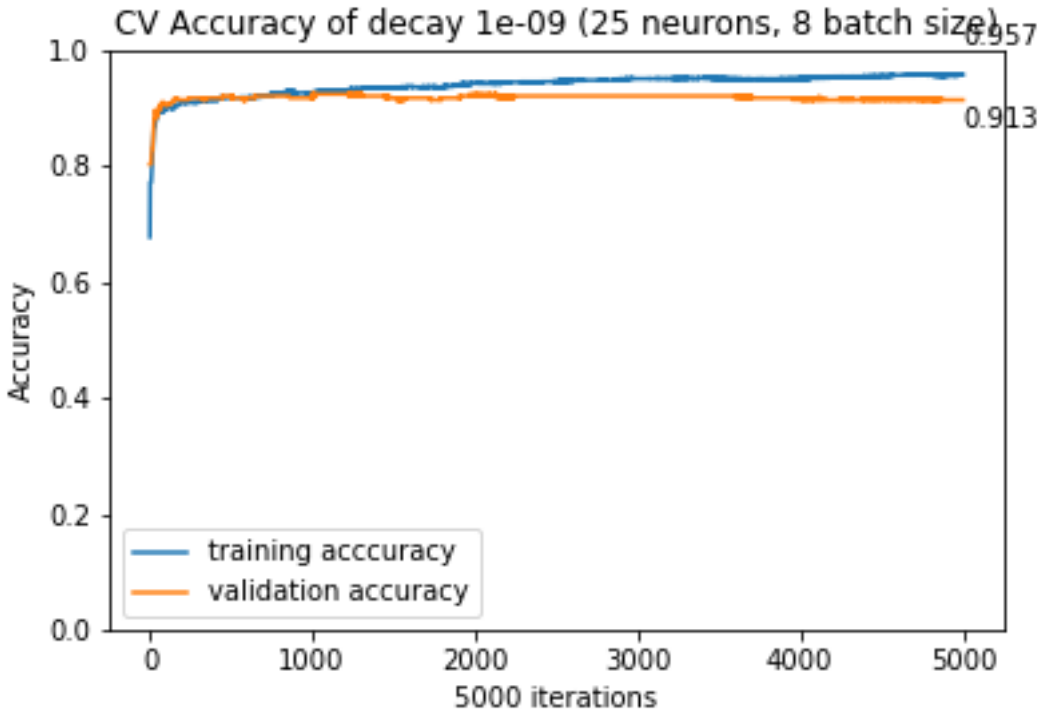

Figure A4.19: Accuracy for Fold 4 (Decay Parameter 10e-9)

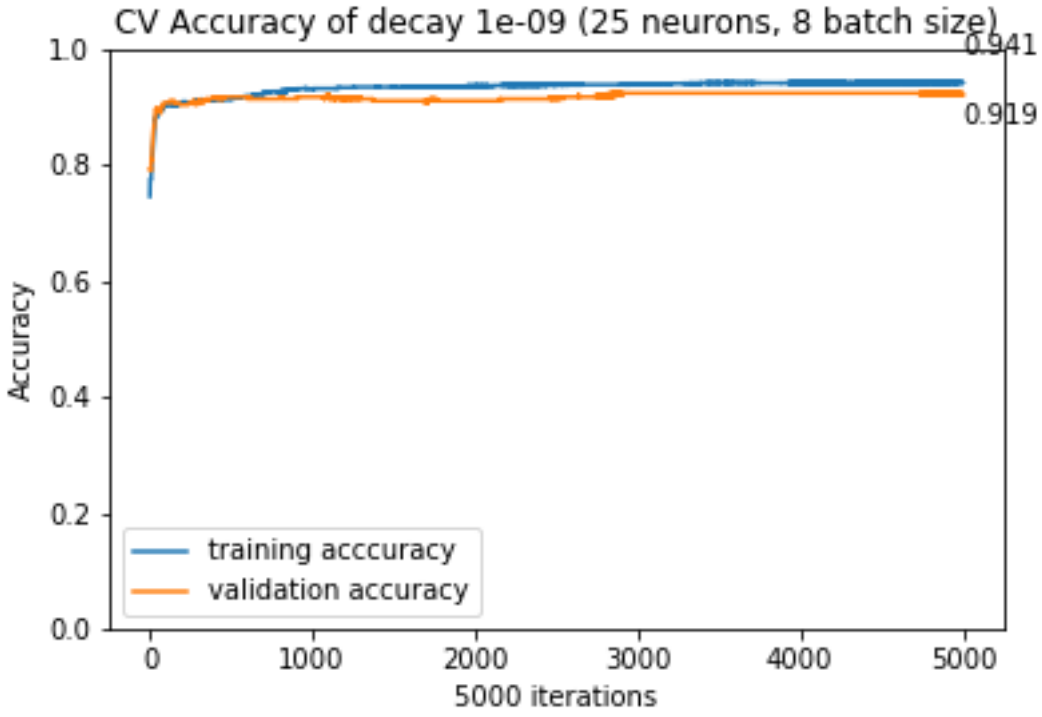

Figure A4.20: Accuracy for Fold 5 (Decay Parameter 10e-9)

Decay Parameter 1e-12

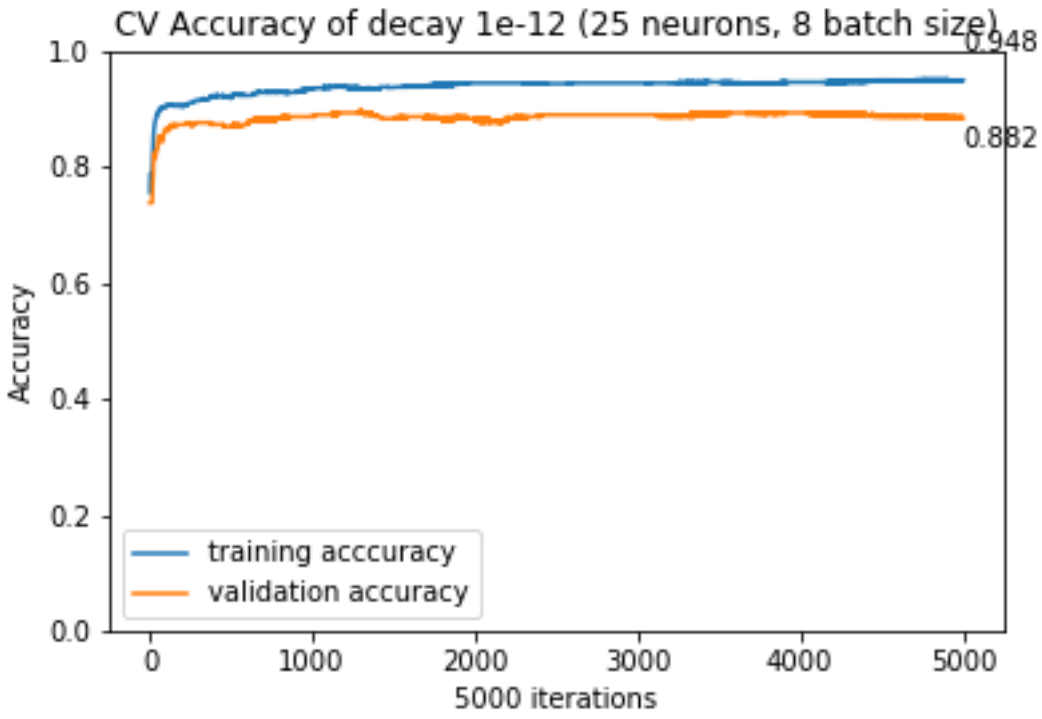

Figure A4.21: Accuracy for Fold 1 (Decay Parameter 10e-12)

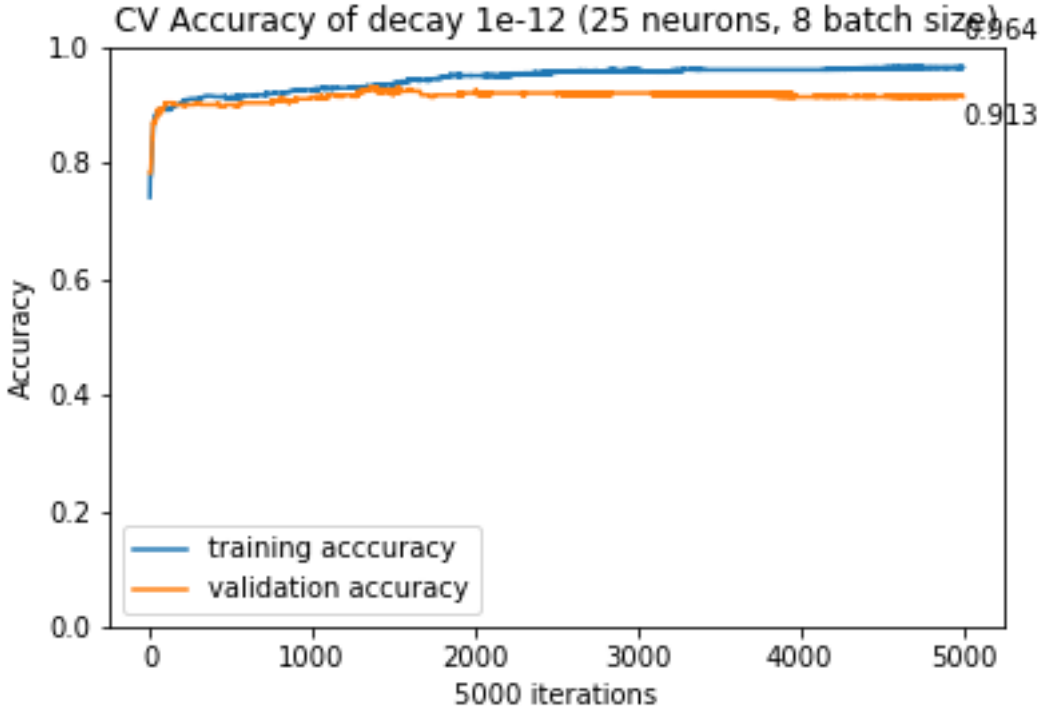

Figure A4.22: Accuracy for Fold 2 (Decay Parameter 10e-12)

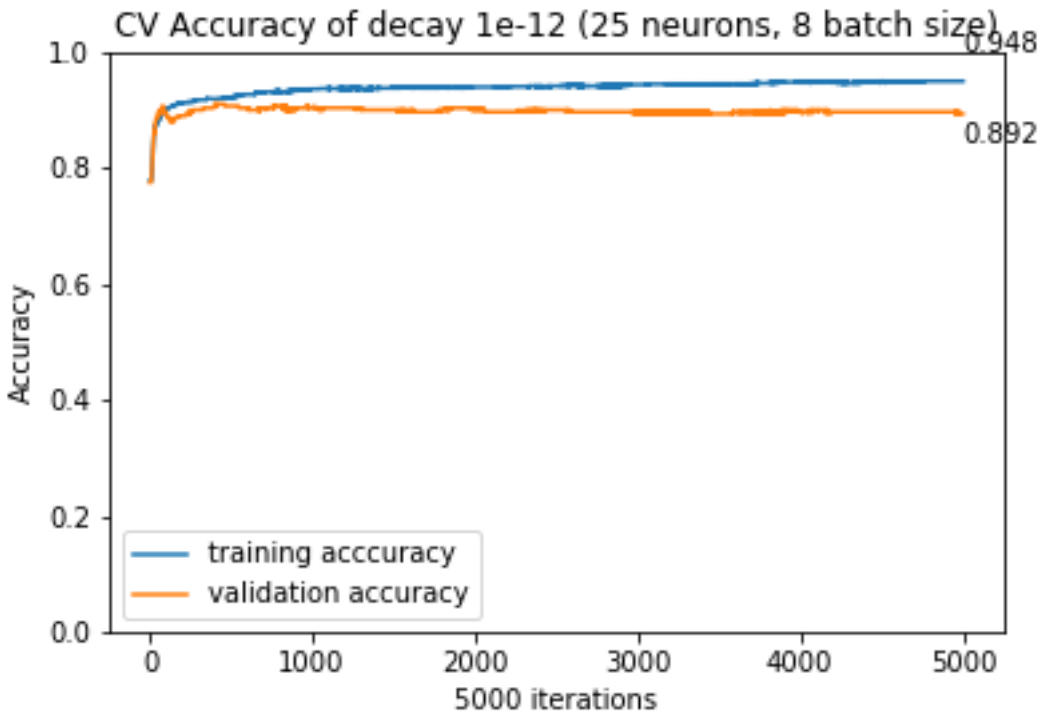

Figure A4.23: Accuracy for Fold 3 (Decay Parameter 10e-12)

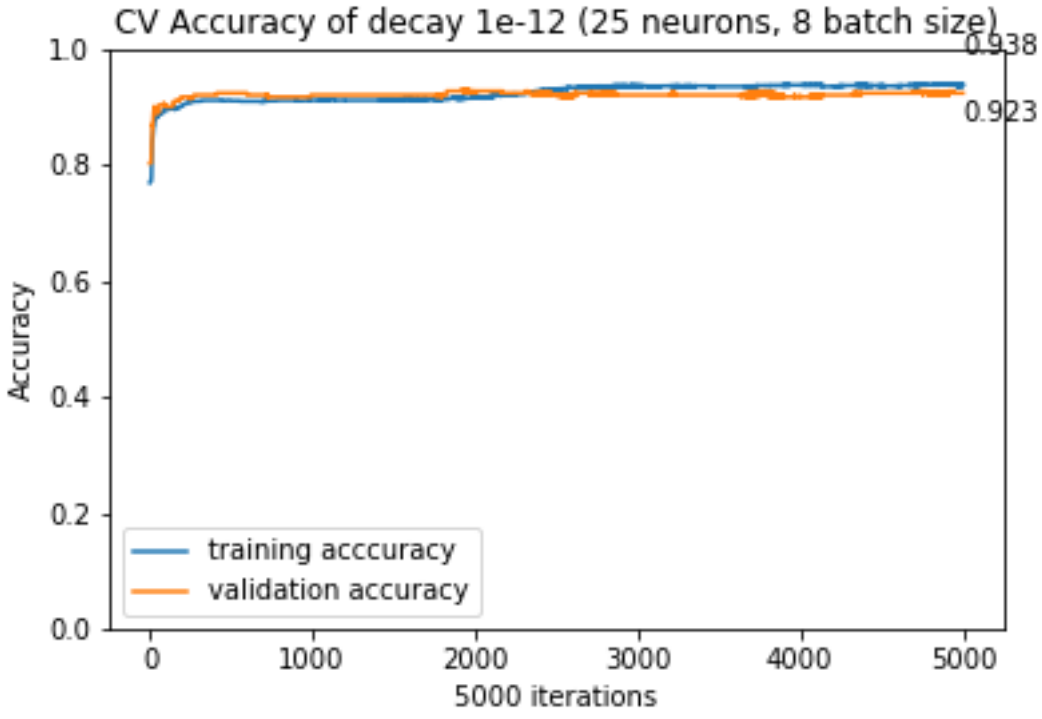

Figure A4.24: Accuracy for Fold 4 (Decay Parameter 10e-12)

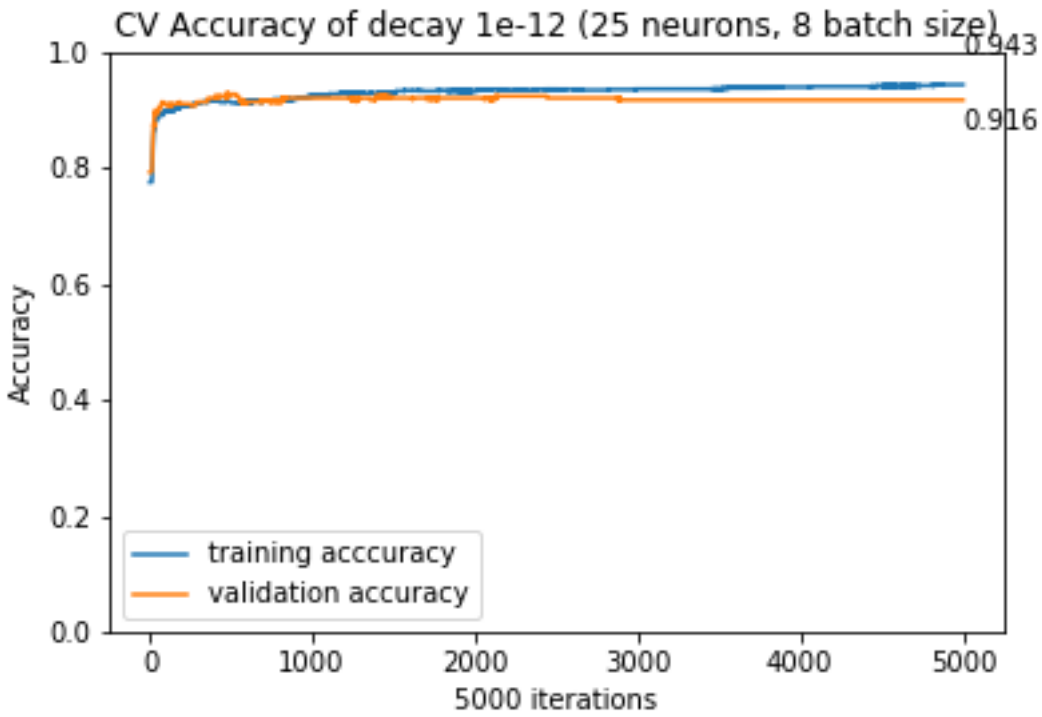

Figure A4.25: Accuracy for Fold 5 (Decay Parameter 10e-12)

5.1.5 Question 5

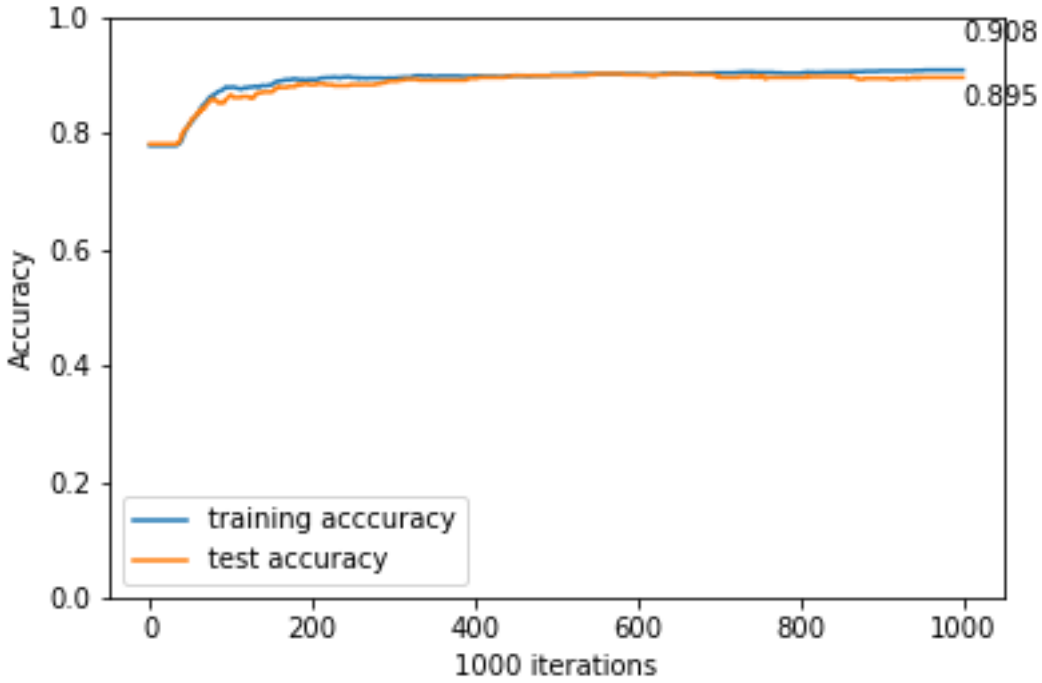

Figure A5.1: Training and Test Accuracy at epochs 1000

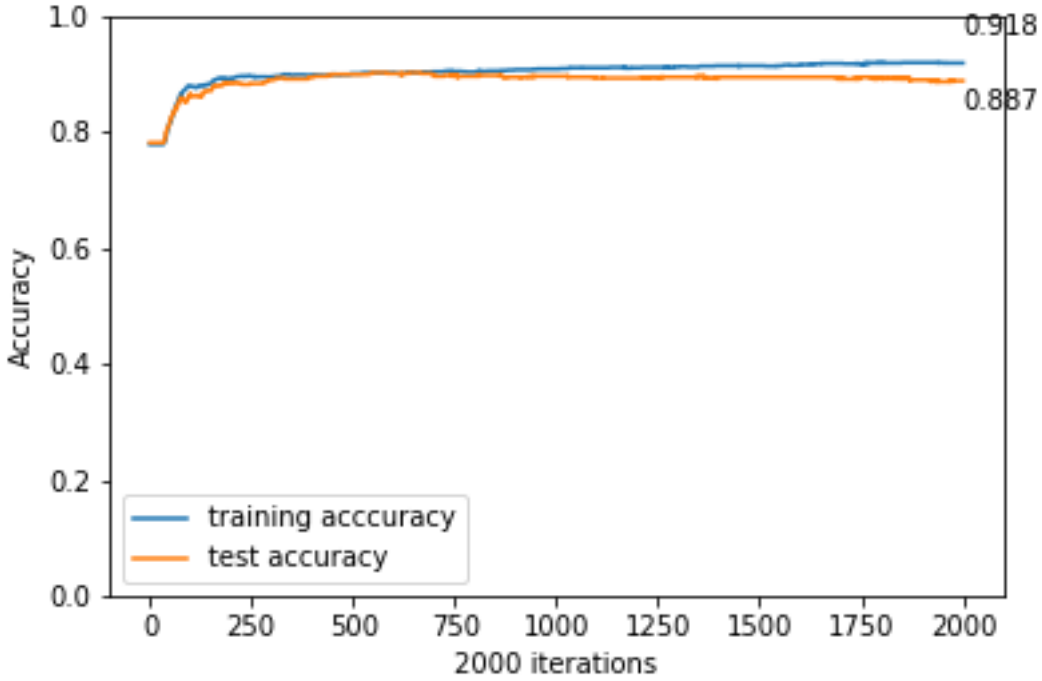

Figure A5.2: Training and Test Accuracy at epochs 2000

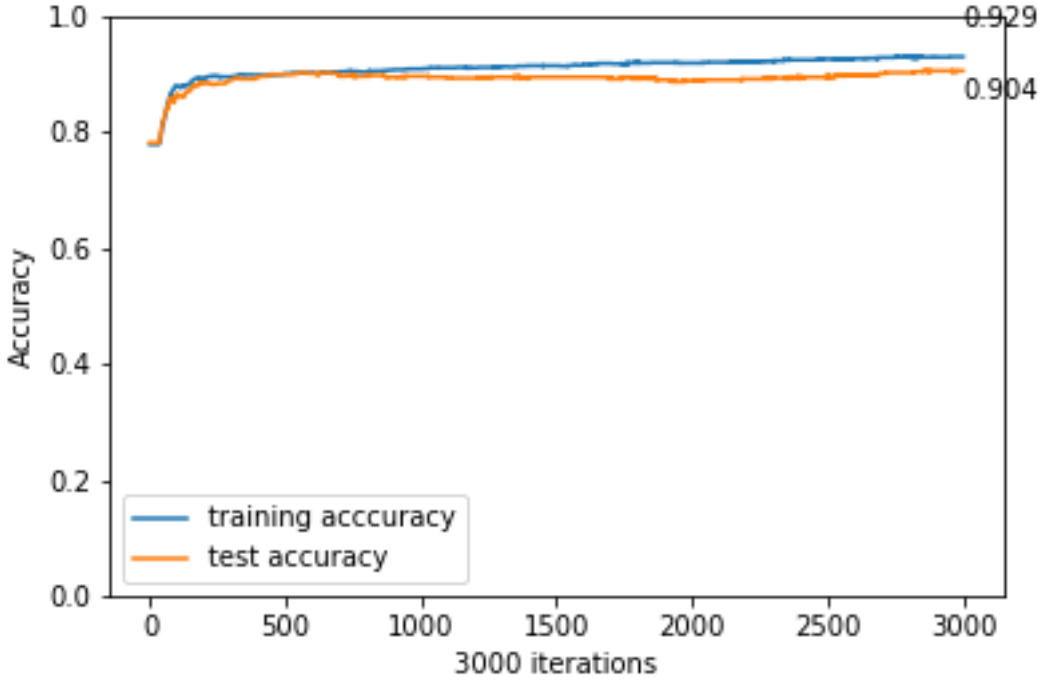

Figure A5.3: Training and Test Accuracy at epochs 3000

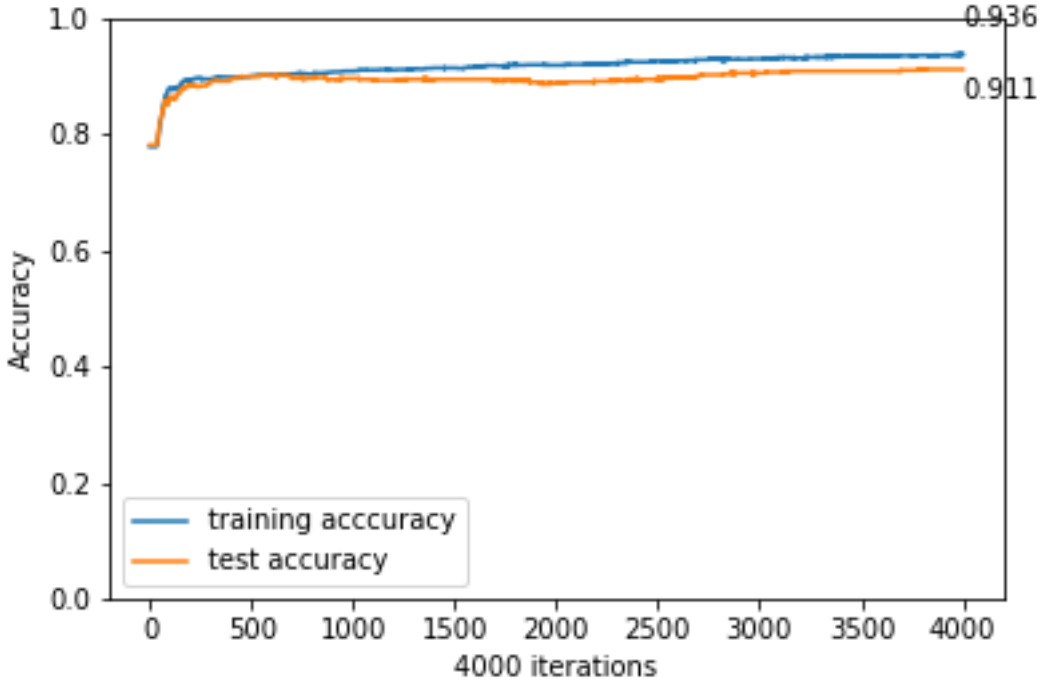

Figure A5.4: Training and Test Accuracy at epochs 4000

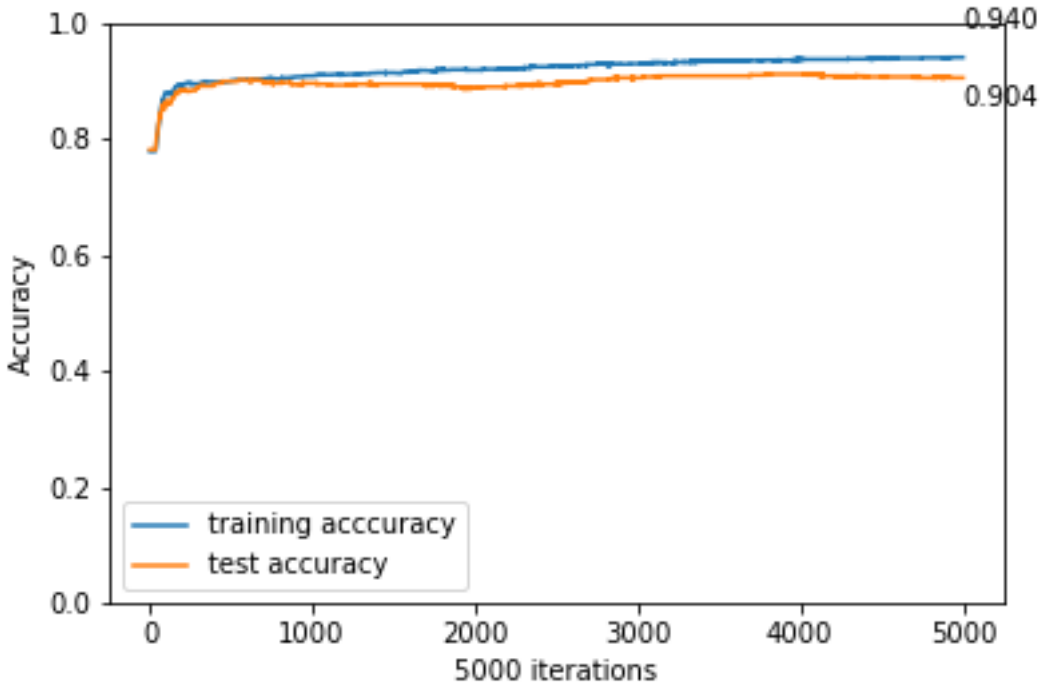

Figure A5.5: Training and Test Accuracy at epochs 5000

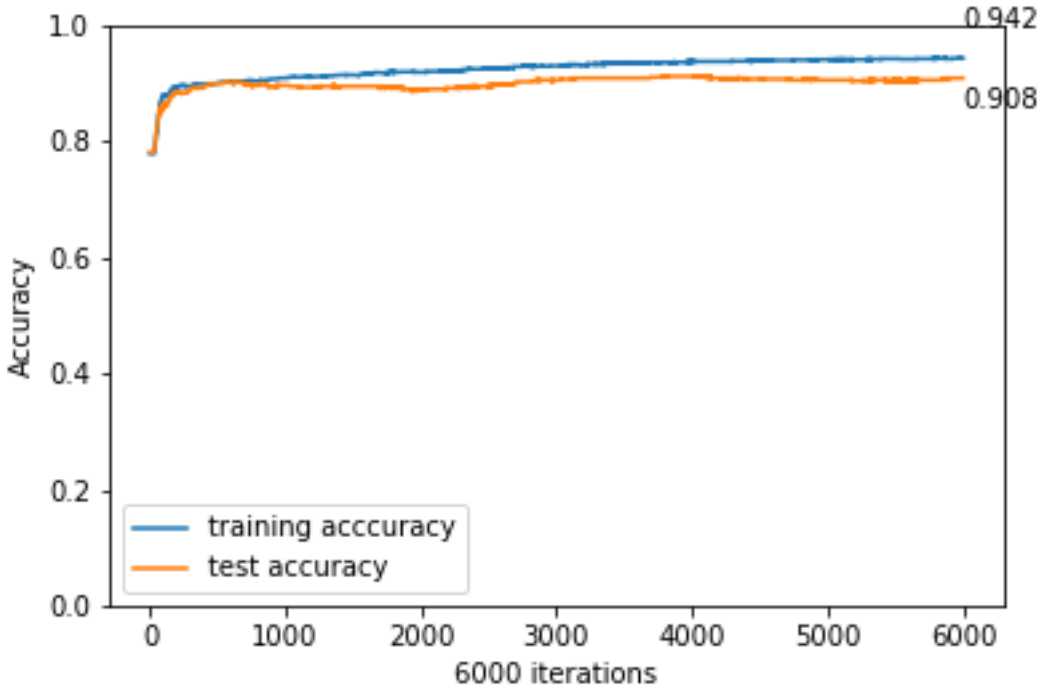

Figure A5.6: Training and Test Accuracy at epochs 6000

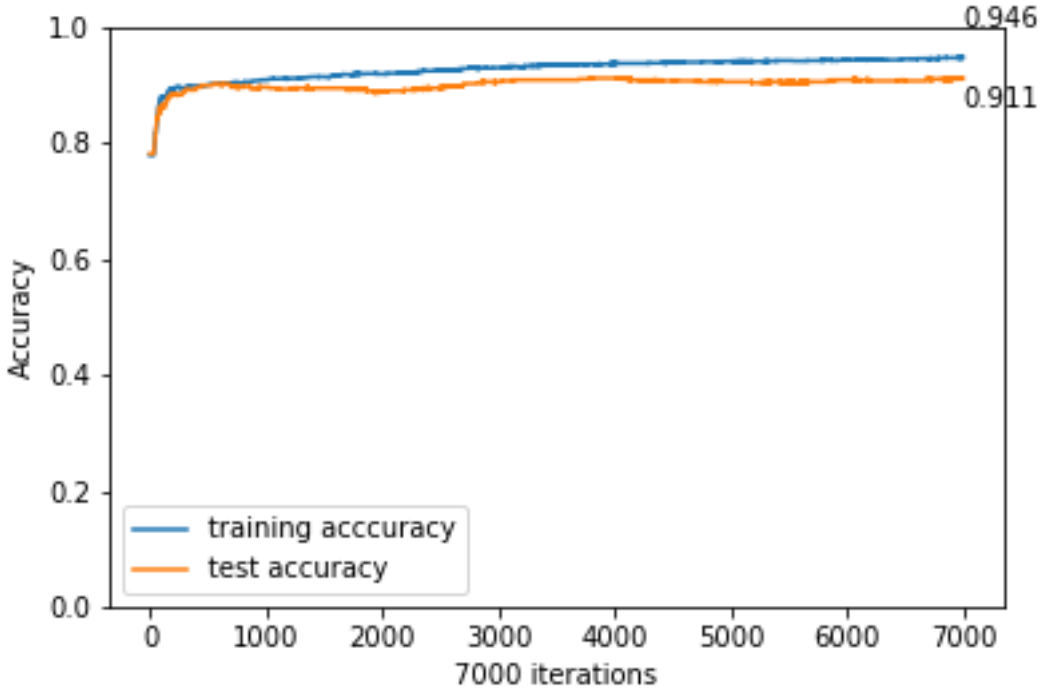

Figure A5.7: Training and Test Accuracy at epochs 7000

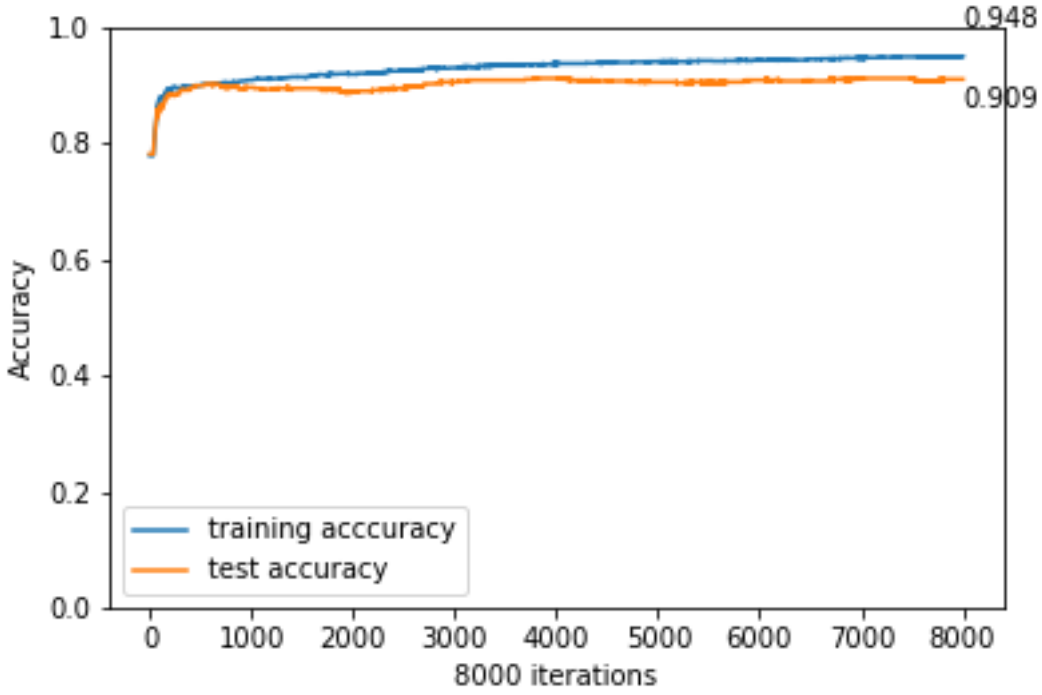

Figure A5.8: Training and Test Accuracy at epochs 8000

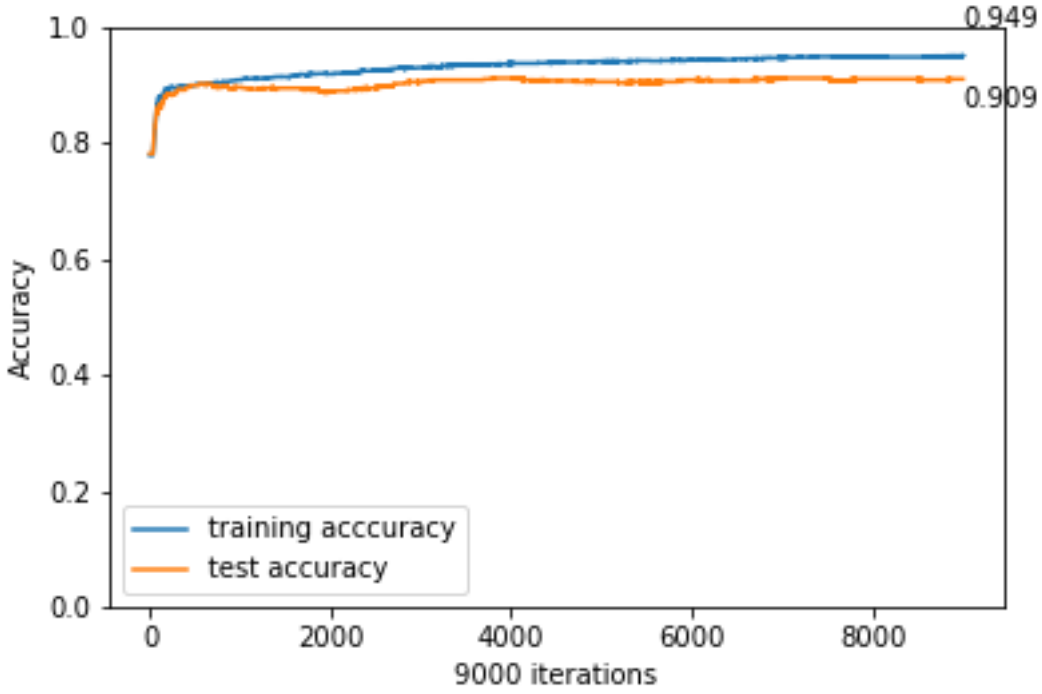

Figure A5.9: Training and Test Accuracy at epochs 9000

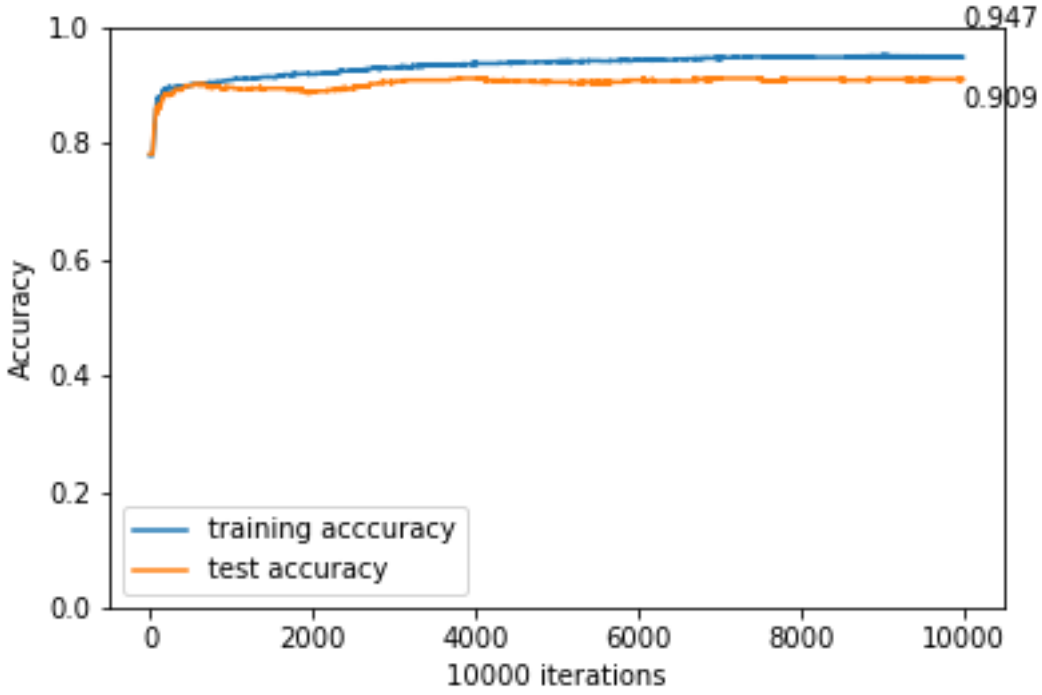

Figure A5.10: Training and Test Accuracy at epochs 10000

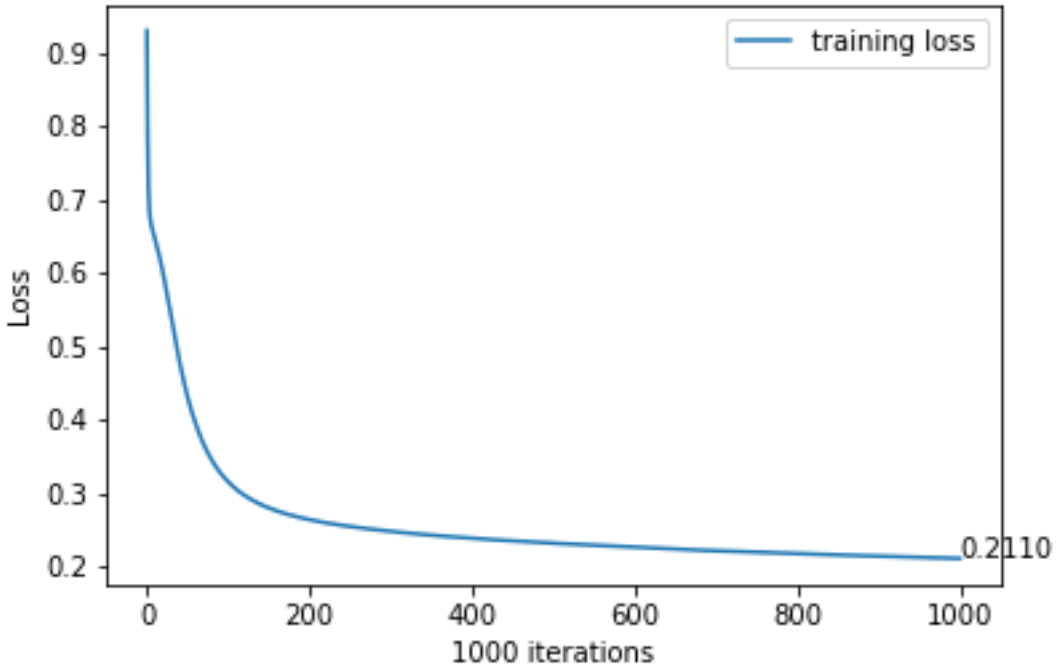

Figure A5.11: Training Loss at epochs 1000

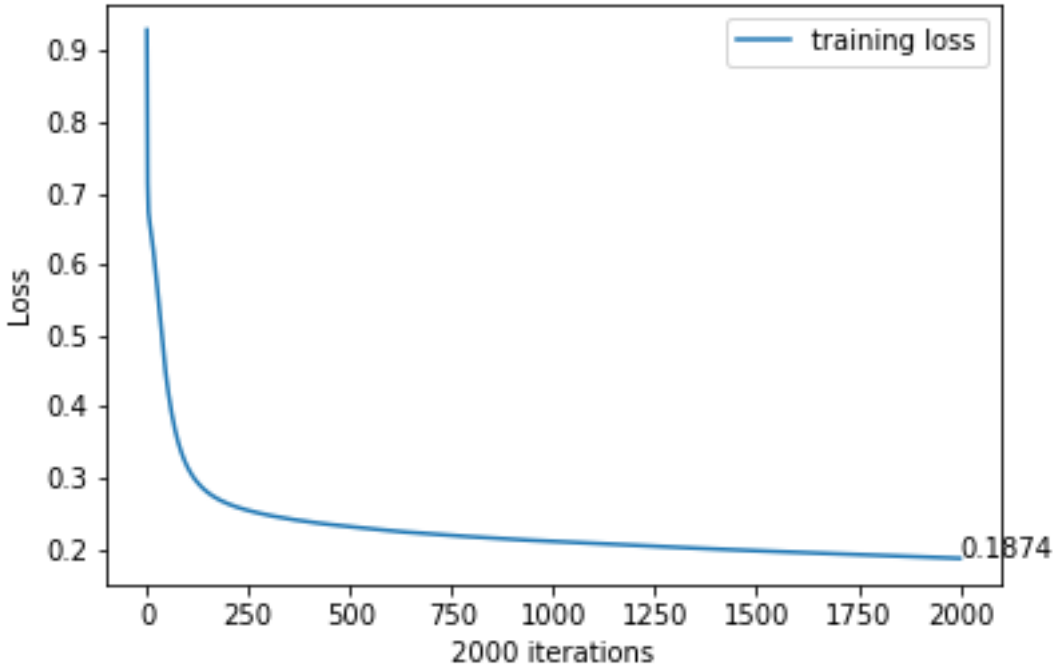

Figure A5.12: Training Loss at epochs 2000

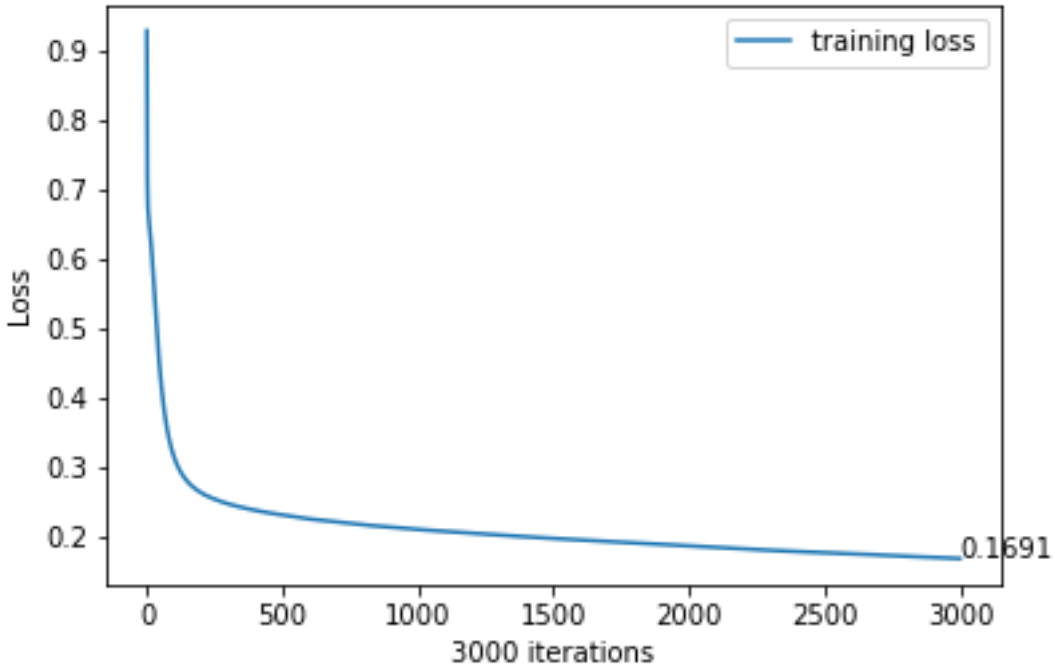

Figure A5.13: Training Loss at epochs 3000

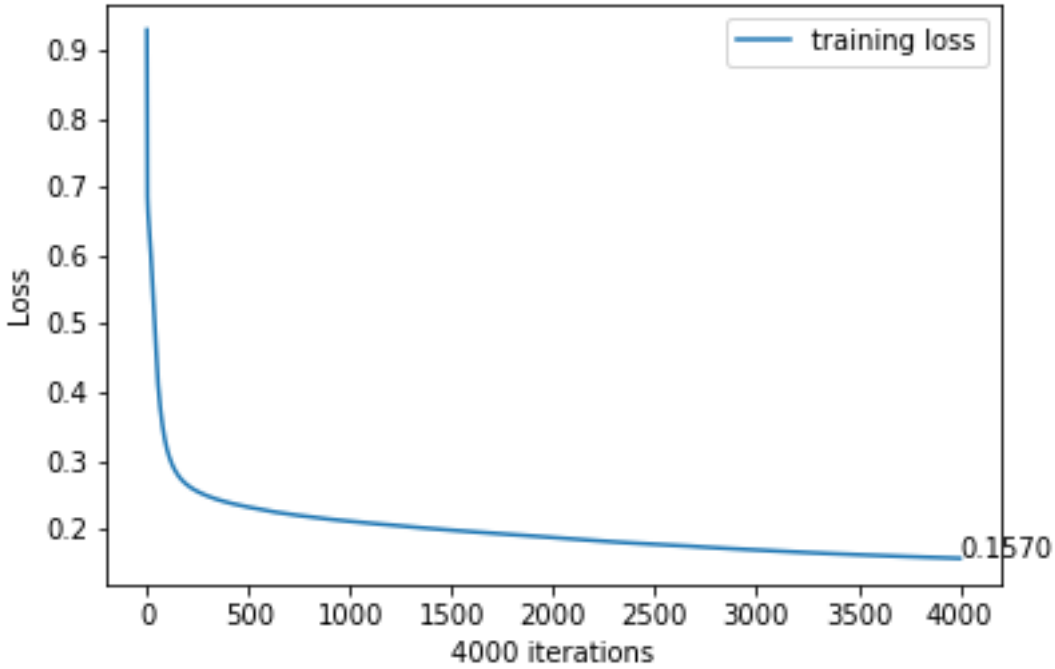

Figure A5.14: Training Loss at epochs 4000

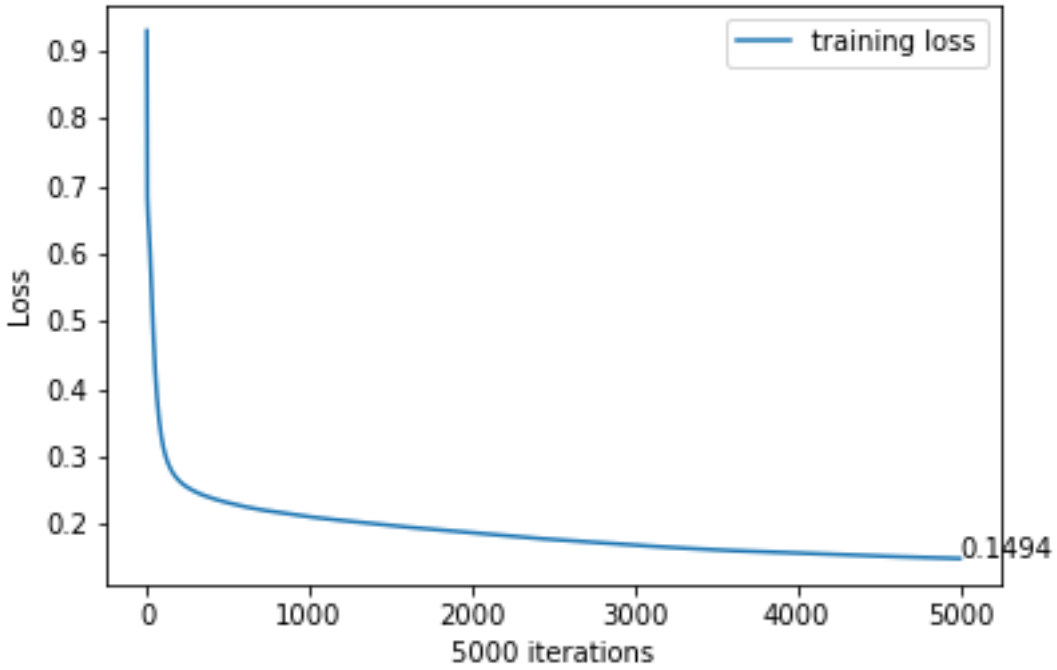

Figure A5.15: Training Loss at epochs 5000

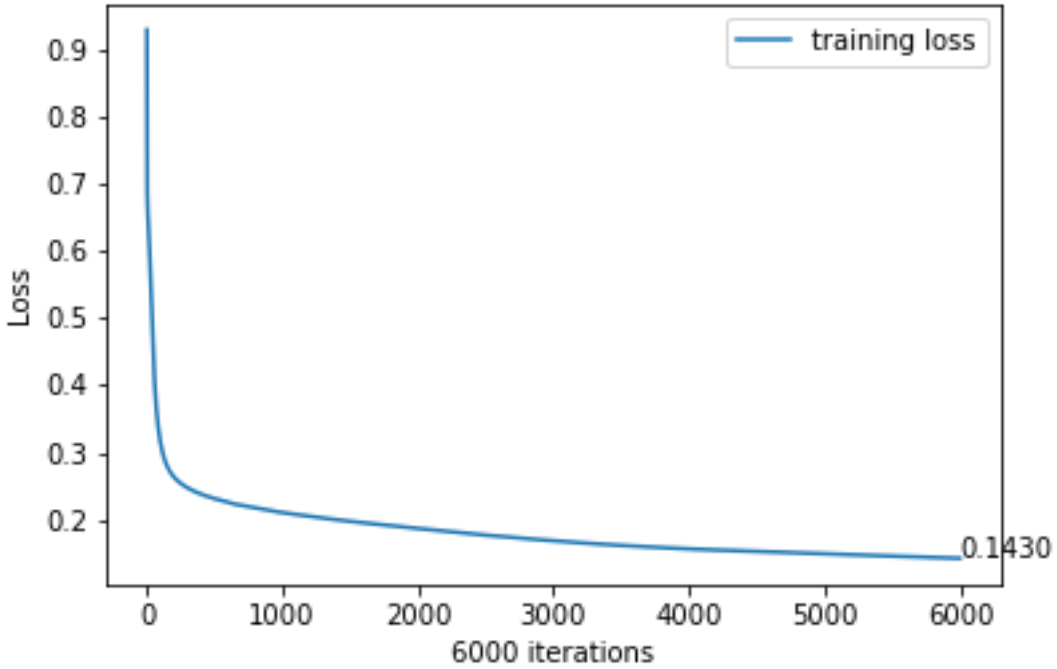

Figure A5.16: Training Loss at epochs 6000

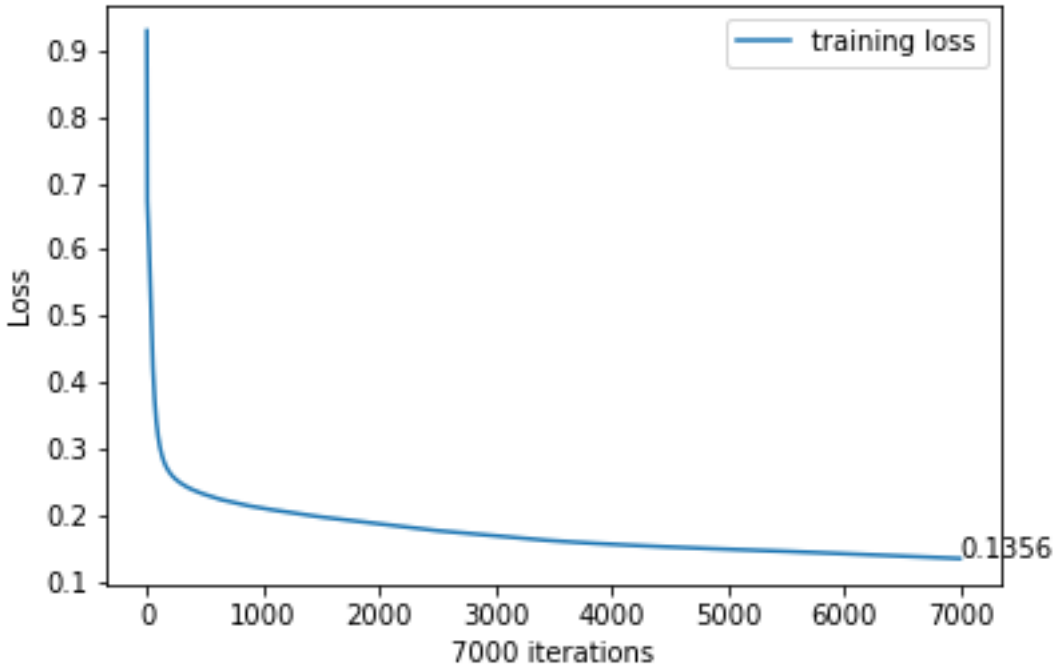

Figure A5.17: Training Loss at epochs 7000

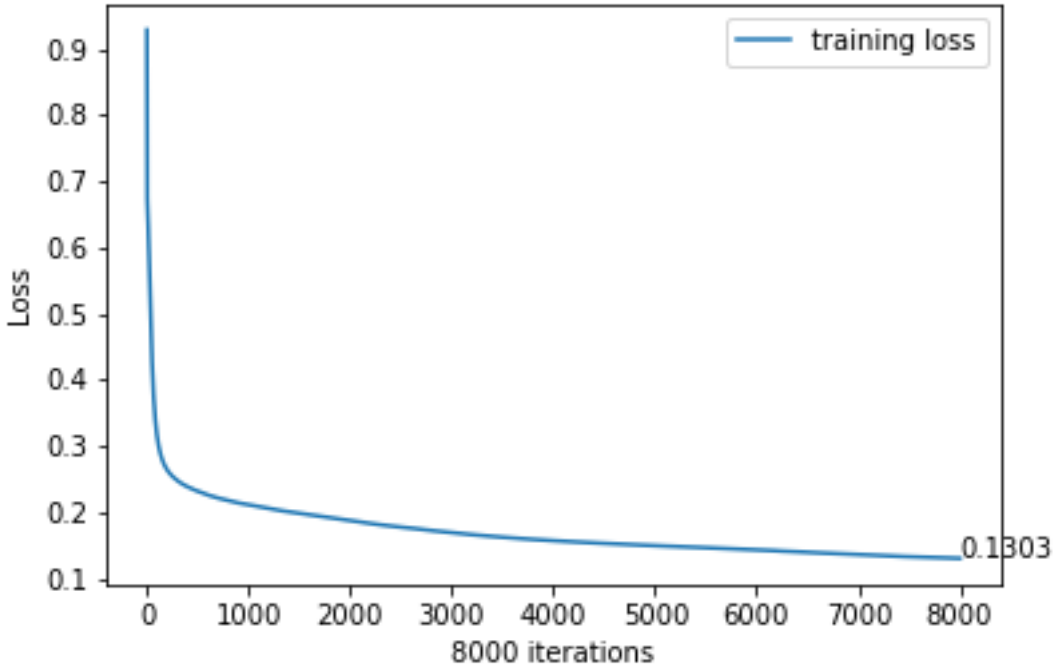

Figure A5.18: Training Loss at epochs 8000

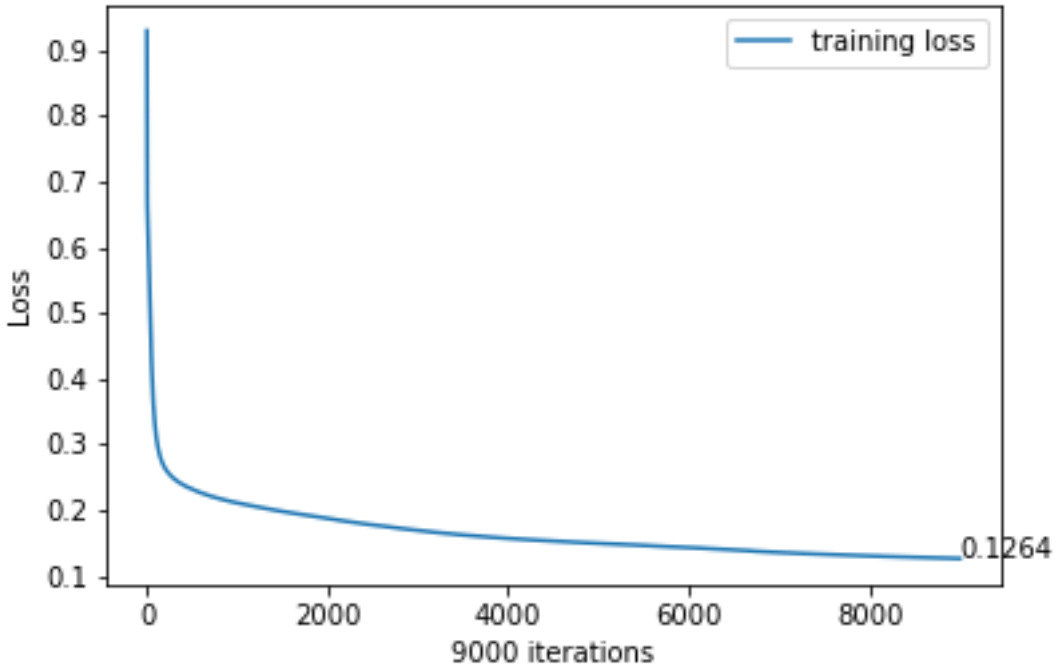

Figure A5.19: Training Loss at epochs 9000

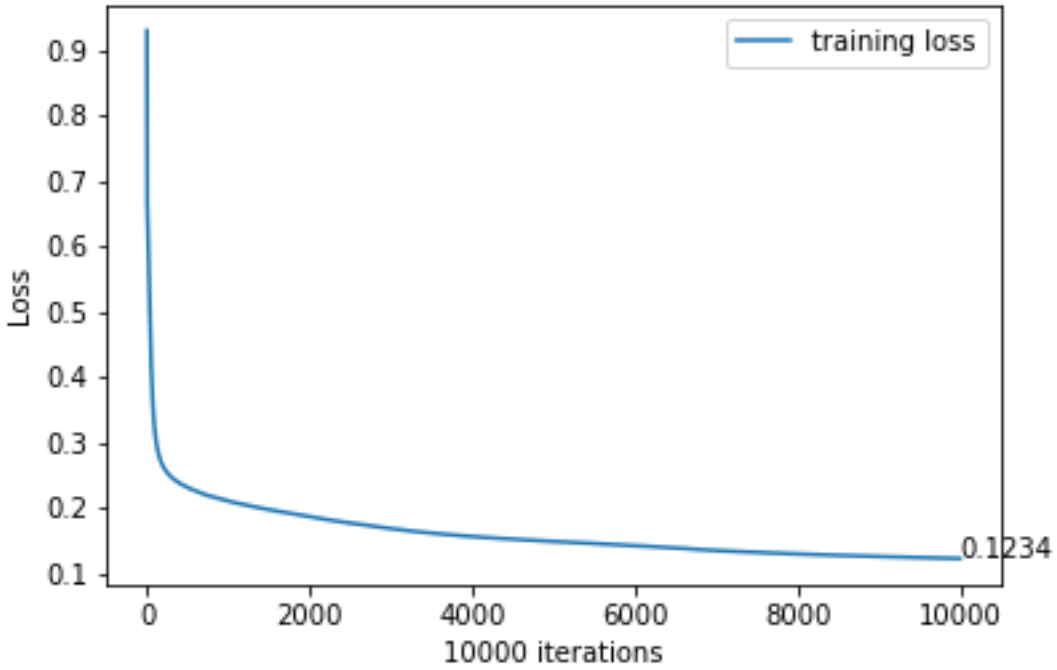

Figure A5.20: Training Loss at epochs 10000