COS 314 Project 2

Training a Neural Network - Option 2

Tobias Jeremiah Bester - u14041368

This project involved training a feedforward neural network along with the use of backpropagation to change the training weights to optimize classification accuracy.

The dataset on which the network was trained and validated was a set of 20000 patterns, each of which containing features of one instance of the 26 English capital letters, represented in one of 20 fonts.
Each feature described a certain aspect of the pattern, and was represented by an integer value ranging from 0 to 15. Each pattern also contained a class label, namely the letter it represented. This data was split 60%/20%/20% into training, validation, and generalization sets, respectively.

Running the Program

The following process is used to compile and run the program on a Linux system.

1. Run "**javac *.java**" in the "code" folder to compile all the java files.
2. Run the respective experiment program by using "**java experiment?**" where ? is replaced with the corresponding experiment number.
3. The output of each epoch number, training error, and generalization error can be found in **output.txt**.

Data Pre-processing

Before splitting the data into the different sets, data pre-processing was applied to each pattern. This was done so that the data will be in the necessary format for the neural network. The exact pre-processing took place in two steps:

1. Firstly, each pattern's features, which ranged from 0 to 15, needed to be scaled to a range of [0.0, 1.0]. The reason for this is that the activation functions used in both the hidden and output units were sigmoid functions, which output to a range of [0.0, 1.0]. Since the output range of the output units had to be the same as the input values, the feature values were divided by 15 in order to get them in the range of [0.0, 1.0].
2. Secondly, for the same reason of the output range being [0.0, 1.0], the classification labels ranging from A to Z had to be processed to fit into the appropriate range. This was done by first converting the ASCII code of the pattern's classification letter into an integer in the range of [1, 26], corresponding to its index in the alphabet. This was followed by scaling this integer into the range of [0.0, 1.0] by dividing by 26. This resulted in each pattern's classification label being in a range of [0.0, 1.0], appropriate for the output range of the neural network's output units.

Stopping Conditions

During the training process, there are a few variables and factors that change as the process continues. When some of these variables reach a certain criteria, the training process needs to come to a halt for a certain reason. The following stopping conditions are used for this project's training process.

- Stop if **Epoch Num > Epoch Max**: This stopping condition is generally the final check to be made when deciding whether or not to carry on training the model. An epoch is a single iteration or run of the entire training set through the neural network. This stopping condition entails stopping the training process when a maximum number of epochs have been reached. The main reason for this stopping condition is to prevent training from continuing after a certain length of time has been reached, especially after other stopping conditions have not been met yet, which would generally indicate that a lack of change or improvement to the network has been made. Further details on the specific Max Epoch value used for each experiment can be found in the respective experiment's section.
- Stop if **Generalization Accuracy >= Desired Accuracy**: This stopping condition is used to ensure that no overfitting occurs with the given dataset. Although a high prediction accuracy is wanted, when the accuracy surpasses a certain threshold, it can be considered that the training is overfitting the given examples. Therefore, the training should end when this threshold is reached. In this project, the desired accuracy chosen was 99%, which was deemed high enough for the network to be good at generalization, but not so that it overfits the dataset. Any higher accuracy value was considered to both be overfitting and take too long to reach.
- Stop if **Validation Error > Average validation error - 5 * Standard Deviation of validation error**: This stopping condition also prevents the training process to overfit the given dataset, and also to preserve a good validation accuracy. The point of overfitting was determined to be when the validation error decreased below a certain value. Originally, this value was (the average validation error) minus (the standard deviation), and then later two and three deviations, but, as will be shown in the individual experiment sections, a trend occurred where during the training process, the prediction accuracy would dip for a few epochs before increasing sharply. Therefore, five deviations from the average was chosen to be the threshold to account for this dip and increase, as it lead to a generally high accuracy.

In the following sections, the different experiments will be detailed.

Experiment 1

In this experiment, the prediction that needed to be made was whether the output was a specific letter or not. This specific letter can be chosen by the user. Only one output unit is used, which should output 1.0 (or > 0.7) if the current patteren's class is the letter chosen, and 0.0 (or > 0.3) if it is not.

As mentioned previously, a trend that was noticed during the training process was that the prediction accuracy would momentarily, and only once per training session, dip and then increase again. Sometimes, this dip would be large enough to trigger the third stopping condition, especially when there was no change in the accuracy for a number of epochs. Strangely, this phenomenon only occurred for a few specific letters, such as "G", whereas with other letters, there was always some change in the accuracy after a few epochs, so the dip was never too large. Whether this oddity was due to the dataset or a flaw in the implementation was yet to be determined.

Three parameters were tuned via trial-and-error to achieve different accuracies and results from the neural network. The three parameters were the number of hidden units, the learning rate, and the momentum. By default, the following values were used:

- Num. Hidden Units: For this default value, the average of the number of input and output units was used. Since this experiment used one output unit, the default value was (16 + 1)/ 2 = **8**.
- Learning Rate: The default value used for this parameter was the inverse of the fanin for each neuron. Therefore, the learning rate for the hidden unit weight change was ⅛ = **0.125** and the learning rate for the input weight change was 1/16 = **0.0625**.
- Momentum: The default value for this parameter was chosen to be the 0.05, which just felt like a good default momentum value.

Tuning the Number of Hidden Units

Below is a table of ten training sessions where the number of hidden units was set to the default, as discussed above. All subsequent values chosen all remain in the range of (num. input units, num. output units).

Letter : A		**Hidden Units: 8**		**Learning Rate: 0.0625**		**Momentum: 0.05**	
Training Run	Epoch Reached	Training Accuracy	Validation Accuracy	Generalization Accuracy	Max Epoch	Desired Accuracy	Overfitting
1	31	98.775	99	99	0	1	0
2	26	98.5833	99	99	0	1	0
3	89	99.0583	98.9	99	0	1	0
4	37	98.8416	98.675	99	0	1	0
5	100	99.1916	98.925	98.875	1	0	0

6	70	98.9833	99.25	99	0	1	0
7	21	98.575	98.825	99	0	1	0
8	40	98.8917	98.775	99	0	1	0
9	100	99.075	98.95	98.825	1	0	0
10	100	99.1083	99.175	98.9	1	0	0
	61.4	98.90831	98.9475	98.96	3	7	0

Below is a similar table using a lower number of hidden units, 4.

Letter : A		**Hidden Units: 4**		**Learning Rate: 0.0625**		**Momentum: 0.05**	
Training Run	Epoch Reached	Training Accuracy	Validation Accuracy	Generalization Accuracy	Max Epoch	Desired Accuracy	Overfitting
1	35	99.0333	98.8501	99	0	1	0
2	26	98.7667	98.925	99.0825	0	1	0
3	17	98.5083	98.625	99.0245	0	1	0
4	26	98.775	98.925	99	0	1	0
5	100	99.1	99.2	98.95	1	0	0
6	24	98.8583	98.7	99	0	1	0
7	21	98.8	98.675	99	0	1	0
8	27	98.95	98.625	99.05	0	1	0
9	73	99.15	98.775	99	0	1	0
10	35	99.0083	98.9	99	0	1	0
	38.4	98.89499	98.82001	99.0107	1	9	0

Below is a similar table using a higher number, 12.

Letter : A		**Hidden Units: 12**		**Learning Rate: 0.0625**		**Momentum: 0.05**	
Training Run	Epoch Reached	Training Accuracy	Validation Accuracy	Generalization Accuracy	Max Epoch	Desired Accuracy	Overfitting
1	58	98.9833	98.95	99	0	1	0
2	47	98.9416	98.925	99	0	1	0
3	49	98.8583	99.125	99	0	1	0
4	55	99.0167	98.95	99	0	1	0
5	26	98.5167	98.7	99	0	1	0
6	52	99	98.95	99	0	1	0
7	25	98.6	98.625	99	0	1	0
8	100	99.0583	99.2	98.925	1	0	0
9	40	98.8166	98.75	99	0	1	0

10	52	98.9332	98.95	99	0	1	0
	50.4	98.87247	98.9125	98.9925	1	9	0

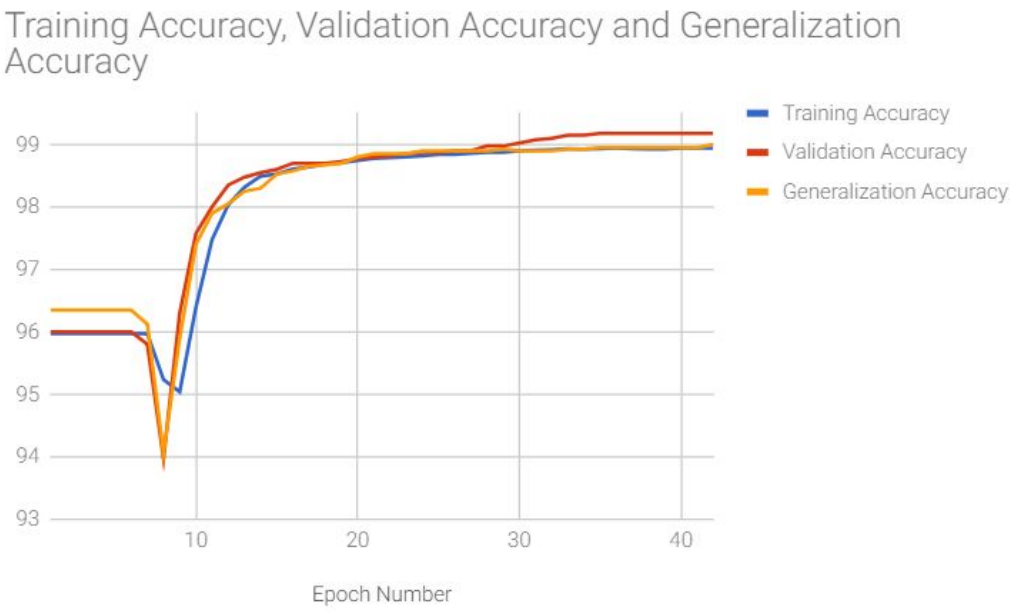
A summary of these results can be found in the following table:

Hidden Unit	Avg. Epoch Reached	Training Accuracy	Validation Accuracy	Generalization Accuracy	Max Epoch	Desired Accuracy	Overfitting
8	61.4	98.90831	98.9475	98.96	3	7	0
4	38.4	98.89499	98.82001	99.0107	1	9	0
12	50.4	98.87247	98.9125	98.9925	1	9	0

As can be seen, a lower number of hidden units results in fewer epochs need to get the desired accuracy. This could be as a result of the lower value also resulting in the learning rate being higher. This concept of a lower number being better can be confirmed when looking at the following table, where **two** hidden units were used.

Training Run	Epoch Reached
1	18
2	22
3	15
4	23
5	23
6	40
7	33
8	77
9	21
10	79
	35.1

A learning profile of a single training run with four hidden units used can be found below.



The dip in accuracy can also be seen in the graph above. Upon further investigation, the reason for this dip was determined to be that it was the point in which the output values were increased towards 1.0 for the patterns with the chosen letter classification, and the decreasing towards 0.0 for patterns which were not in the chosen letter class. A crossing point with these output values exist in which both produce incorrect predictions, which results in the dip in accuracy.

Another note is that the maximum number of epochs was set to 100, as a majority of training sessions reached the desired accuracy of 99% before epoch 100.

Tuning the Learning Rate

Although the learning rate was changed according to how the number of hidden units were tuned in the previous section, in this section, the learning rate will be specifically tuned, starting with the default values as discussed below. A learning rate of 0.0625 is used for the input weight changes and 0.125 for the hidden weight changes. Also note that the letter V was the chosen letter, which resulted in more epochs required.

Letter : V		**Hidden Units: 8**		**Learning Rate: 0.0625 / 0.125**		**Momentum: 0.05**	
Training Run	Epoch Reached	Training Accuracy	Validation Accuracy	Generalization Accuracy	Max Epoch	Desired Accuracy	Overfitting
1	100	97.5333	97.45	97.475	1	0	0
2	100	97.7667	97.95	97.575	1	0	0
3	20	96.0666	95.45	95.05	0	0	1
4	100	97.1883	96.7	96.675	1	0	0
5	100	97.725	97.5	97.7	1	0	0

6	25	95.8833	94.325	94.5	0	0	1
7	100	97.475	97.475	97.275	1	0	0
8	21	96.275	95.925	95.825	1	0	0
9	100	98.0333	98.1	97.95	1	0	0
10	100	97.6833	97.975	97.95	1	0	0
	76.6	97.16298	96.885	96.7975	8	0	2

Below is a similar table with both learning rates set at 0.05, a relatively low value.

Letter : V		**Hidden Units: 8**		**Learning Rate: 0.05 / 0.05**		**Momentum: 0.05**	
Training Run	Epoch Reached	Training Accuracy	Validation Accuracy	Generalization Accuracy	Max Epoch	Desired Accuracy	Overfitting
1	30	96.0333	96.15	96.5	0	0	1
2	21	96.1166	96.25	96.1	0	0	1
3	31	96.2083	96.225	95.95	0	0	1
4	22	96.2916	96.175	95.75	0	0	1
5	27	96.0833	96	95.975	0	0	1
6	24	96.1417	95.7	95.925	0	0	1
7	24	96.125	96.375	96.05	0	0	1
8	29	96.125	96.025	95.95	0	0	1
9	27	95.8083	95.675	95.7	0	0	1
10	36	96.1166	95.625	95.225	0	0	1
	27.1	96.10497	96.02	95.9125	0	0	10

Below is a similar table with a slightly higher learning rate of 0.5.

Letter : V		**Hidden Units: 8**		**Learning Rate: 0.5 / 0.5**		**Momentum: 0.05**	
Training Run	Epoch Reached	Training Accuracy	Validation Accuracy	Generalization Accuracy	Max Epoch	Desired Accuracy	Overfitting
1	54	99.1667	99.075	99	0	1	0
2	49	99.05	99.25	99.05	0	1	0
3	45	98.975	99.05	99.025	0	1	0
4	100	99.375	98.8	98.875	1	0	0
5	71	99.2416	99.15	99.025	0	1	0
6	61	99.1916	99	99	0	1	0
7	100	99.325	99.025	98.85	1	0	0
8	87	99.2083	99.4	99	0	1	0
9	51	99.2167	99.05	99	0	1	0

10	45	99.0584	98.75	99	0	1	0
	66.3	99.18083	99.055	98.9825	2	8	0

Below is a summary of all three tuned values.

Learning Rate	Epoch Reached	Training Accuracy	Validation Accuracy	Generalization Accuracy	Max Epoch	Desired Accuracy	Overfitting
1 / fanin	76.6	97.16298	96.885	96.7975	8	0	2
0.05	27.1	96.10497	96.02	95.9125	0	0	10
0.5	66.3	99.18083	99.055	98.9825	2	8	0

Using a low learning rate resulted in a much slower training improvement rate, and at its lowest, when set to 0.05, it resulted in the stopping condition being reached in all sessions. This was due to the initial improvement and weight changes being too slow, that when the accuracy dip occurred, it was determined to be too much of a decrease in the validation accuracy and thus an overfit. A higher learning rate resulted in a much faster training improvement rate. This corresponds to the first section, where a lower number of hidden units also resulted in a better improvement rate, especially since the learning rate was inversely proportional to this number (fanin).

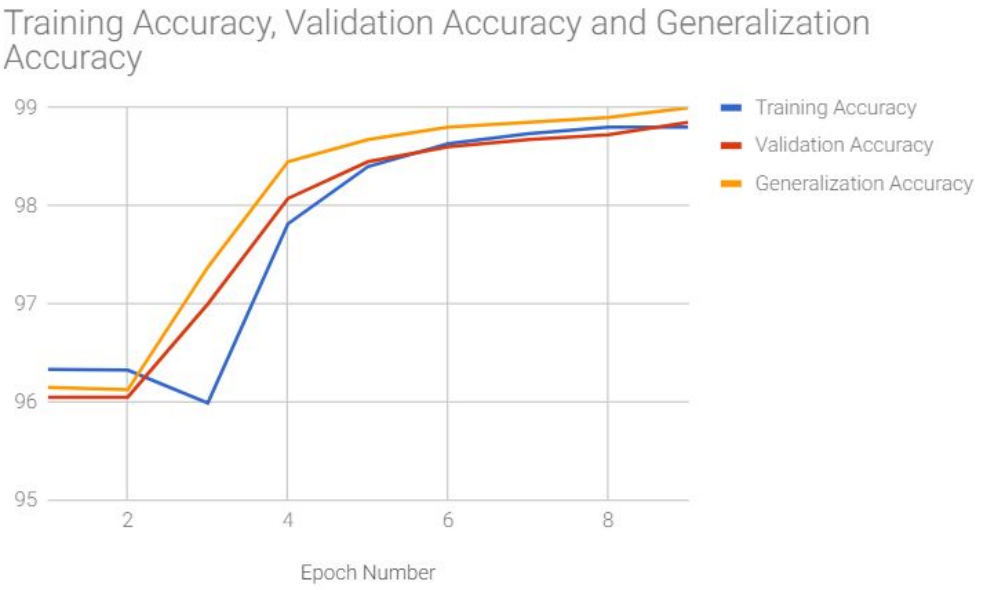
Tuning the Momentum

The final parameter that can be tuned is the momentum, which smooths the search trajectory and ensures that the search path is in the average downhill direction.
Two values were tested as parameters, a low value of 0.05 and a high value of 0.5.

Letter : D		**Hidden Units: 8**		**Learning Rate: 0.5 / 0.5**		**Momentum: 0.05**	
Training Run	Epoch Reached	Training Accuracy	Validation Accuracy	Generalization Accuracy	Max Epoch	Desired Accuracy	Overfitting
1	100	97.7917	97.325	97.725	1	0	0
2	100	98.0333	98.3	98	1	0	0
3	100	98.1833	96.65	97.475	1	0	0
4	100	98.25	97.675	98.05	1	0	0
5	100	98.1916	98.3	97.925	1	0	0
6	100	97.7833	98.025	97.35	1	0	0
7	100	97.7667	97.575	98	1	0	0
8	100	98	96.8	95.625	1	0	0
9	100	97.9083	97.7	97.899	1	0	0
10	100	97.2833	97.55	97.875	1	0	0
	100	97.91915	97.59	97.5924	10	0	0

Letter : D		**Hidden Units: 8**		**Learning Rate: 0.5 / 0.5**		**Momentum: 0.5**	
Training Run	Epoch Reached	Training Accuracy	Validation Accuracy	Generalization Accuracy	Max Epoch	Desired Accuracy	Overfitting
1	100	97.7333	98.05	97.8999	1	0	0
2	100	98.1166	98.1	97.975	1	0	0
3	100	98.0544	97.5	96.85	1	0	0
4	100	97.8666	98.425	97.8	1	0	0
5	100	98.0167	97.55	97.2	1	0	0
6	100	98.25	97.95	98.05	1	0	0
7	100	97.7667	97.6	97.5	1	0	0
8	100	97.725	97.1	96.975	1	0	0
9	100	97.8999	97.45	98.125	1	0	0
10	100	97.575	98.075	97.7	1	0	0
	100	97.90042	97.78	97.60749	10	0	0

Comparing the two results, it can be seen that a large difference in values **did not make much of a difference** to the learning accuracy and improvement rate. Perhaps the one outcome of this section is the aforementioned inconsistency with different letters being able to be predicted accurately more quickly than others. This can be proven by comparing the classification of the letter D (from above) to that of the letter W. Below is a learning profile of one training session where letter W was used. The desired generalization accuracy was reached in nine epochs, whereas the letter D would have only reached the desired accuracy after 100 epochs.



From this, it could be determined that some letters are more easily classified than others.

From training sessions run on other letters, it was determined that a higher momentum value resulted in a faster improvement rate.

Experiment 2

In this experiment, the classification needed to be made was whether or not the given pattern represented a vowel letter or a non-vowel letter. Once again, a single output unit was used in the neural network, and if the pattern was a vowel class, then the output should've been 1.0, and if a non-vowel class, then the output should've been 0.0.

Since the neural network architecture is exactly the same as experiment 1, the only difference between the two experiments is that more patterns in the dataset needed to be fine-tuned, since five classes need to be taken into account as opposed to one.

Because a lower rate of accuracy improvement is expected, the maximum number of epochs is raised to 500, and the desired accuracy reduced to 95%.

The same three parameters from experiment 1 will be fine-tuned in this experiment. The default values used for this experiment will be those which resulted in the best performance in experiment 1, so a low number of hidden units and a high learning rate and momentum.

Tuning the Number of Hidden Units

Once again, the number of hidden units needs to be in the range of (1, 16). Below is a table showing eight training sessions where four hidden units were used.

		Hidden Units: 4		Learning Rate: 0.5		Momentum: 0.05	
Training Run	Epoch Reached	Training Accuracy	Validation Accuracy	Generalization Accuracy	Max Epoch	Desired Accuracy	Overfitting
1	500	87.5666	86.85	86.325	1	0	0
2	500	87.1916	86.675	85.2	1	0	0
3	500	85.9083	86.175	86.0249	1	0	0
4	500	84.9583	81.825	81.825	1	0	0
5	500	87.5083	87	87.45	1	0	0
6	500	88.5249	86.55	86.2	1	0	0
7	500	87.8166	87.1	85.85	1	0	0
8	500	86.5666	85.9	85.925	1	0	0
	500	87.00515	86.009375	85.5999875	8	0	0

Below is a similar table with eight hidden units used, the average of the number of input and output values.

		Hidden Units: 8		Learning Rate: 0.5		Momentum: 0.05	
Training Run	Epoch Reached	Training Accuracy	Validation Accuracy	Generalization Accuracy	Max Epoch	Desired Accuracy	Overfitting
1	500	90.8166	90.4	90.025	1	0	0
2	500	90.0416	89.025	88.6499	1	0	0
3	500	91.3333	90.9	90.75	1	0	0
4	500	83.425	89.3	88.325	1	0	0
5	500	89.8666	90.125	90.225	1	0	0
6	500	89.8666	89.45	89.075	1	0	0
7	500	89.966	89.85	89.5	1	0	0
8	500	92.7	92.225	92.5	1	0	0
	500	89.7519625	90.159375	89.8812375	8	0	0

Then an even higher number of hidden units will be used. The result can be seen below.

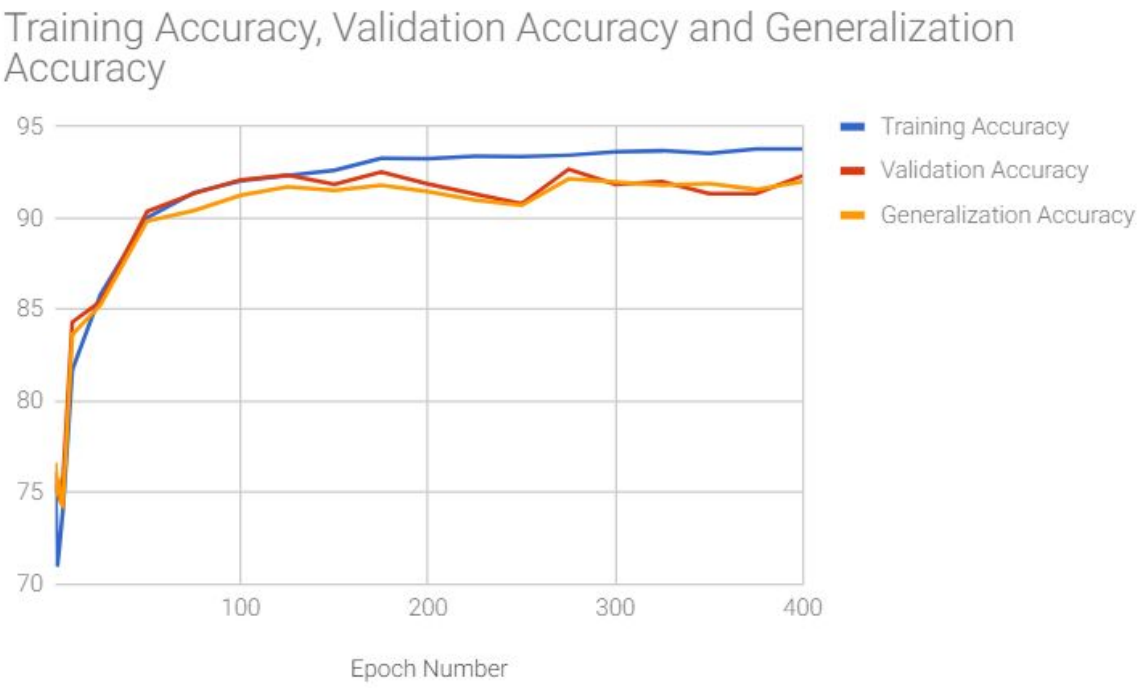
		Hidden Units: 12		Learning Rate: 0.5		Momentum: 0.05	
Training Run	Epoch Reached	Training Accuracy	Validation Accuracy	Generalization Accuracy	Max Epoch	Desired Accuracy	Overfitting
1	500	93.1166	91.1	91.525	1	0	0
2	500	93.9083	93.55	92.7	1	0	0
3	500	94.375	93.3	92.975	1	0	0
4	500	94.0833	93.175	93.325	1	0	0
5	500	93.6166	93	93.15	1	0	0
6	500	93.9916	93.8	93.05	1	0	0
7	500	93.3083	92.925	92.475	1	0	0
8	500	94.5916	94.225	93.8	1	0	0
	500	93.8739125	93.134375	92.875	8	0	0

The following is a summary of the three values of the number of hidden units.

Hidden Units	Avg. Epoch Reached	Training Accuracy	Validation Accuracy	Generalization Accuracy	Max Epoch	Desired Accuracy	Overfitting
4	500	87.00515	86.009375	85.5999875	8	0	0
8	500	89.7519625	90.159375	89.8812375	8	0	0
12	500	93.8739125	93.134375	92.875	8	0	0

It can be seen that a lower number of hidden units results in a lower classification accuracy compared to a higher number. This is in contrast to experiment 1, where a higher number resulted in a lower accuracy. This is as a result of a fixed learning rate in this experiment, meaning the number of hidden units is not related to the learning rate. This suggests that the

learning rate has a higher impact on the rate of training improvement compared to the number of hidden units.
Below is a learning profile of a single training session of the experiment using 12 hidden units.



The accuracy takes a huge dip within the first few epochs but rises sharply before steadying after a certain amount of time, despite the generalization and validation accuracy fluctuating heavily.

Tuning the Learning Rate

As mentioned earlier, the learning rate possibly has a bigger impact on the training accuracy compared to the number of hidden units. The results of a higher learning rate like 0.5 has already been seen from the previous section. The following table shows the results when a lower learning rate of 0.05 is used.

		Hidden Units: 12		**Learning Rate: 0.05**		Momentum: 0.05	
Training Run	Epoch Reached	Training Accuracy	Validation Accuracy	Generalization Accuracy	Max Epoch	Desired Accuracy	Overfitting
1	500	91.5416	90.9	89.925	1	0	0
2	500	90.1333	89.3	88.3	1	0	0
3	500	90.7749	90.075	89.925	1	0	0
4	500	90.7166	89.925	89.05	1	0	0
5	500	91.0416	89.475	90.125	1	0	0
6	500	90.1	89.925	90.4	1	0	0
7	500	90.5166	90.35	89.8	1	0	0

8	500	90.1916	89.725	89.925	1	0	0
	500	90.627025	89.959375	89.68125	8	0	0

Next, the inverse of the fanin will be used as the learning rate. Since this actual value will be similar to 0.05 (1/12 = 0.08333), it can be assumed that the results will be too similar to see a significant difference. A summary of the two values can be seen below.

Learning Rate	Avg. Epoch Reached	Training Accuracy	Validation Accuracy	Generalization Accuracy	Max Epoch	Desired Accuracy	Overfitting
0.5	500	93.8739125	93.134375	92.875	8	0	0
0.05	500	90.627025	89.959375	89.68125	8	0	0

A higher learning rate is seen to result in a higher accuracy, which corresponds to experiment 1. However, the slight difference in accuracy does not linearly correlate to the difference in learning rate, because a vast difference in learning rate does not significantly decrease the rate of classification improvement.

Tuning the Momentum

The previous sections have displayed the results when using a low momentum value. The results of using a higher momentum value can be seen below.

		Hidden Units: 12		Learning Rate: 0.5		**Momentum: 0.5**	
Training Run	Epoch Reached	Training Accuracy	Validation Accuracy	Generalization Accuracy	Max Epoch	Desired Accuracy	Overfitting
1	500	93.9833	92.5	92.2	1	0	0
2	500	93.4	91.6499	91.375	1	0	0
3	500	94.666	93.925	93.875	1	0	0
4	500	94.7916	93.4	92.9	1	0	0
5	500	93.4083	92.125	92.75	1	0	0
6	500	93.9333	93	93.025	1	0	0
7	500	94.3	93.75	93.675	1	0	0
8	500	94.1333	94.2749	93.825	1	0	0
	500	94.076975	93.0781	92.953125	8	0	0

The summary of the lower and higher momentum values and their set of results can be seen below.

Momentum	Avg. Epoch Reached	Training Accuracy	Validation Accuracy	Generalization Accuracy	Max Epoch	Desired Accuracy	Overfitting
0.05	500	93.8739125	93.134375	92.875	8	0	0
0.5	500	94.076975	93.0781	92.953125	8	0	0

Similarly to experiment 1, the change in momentum does not heavily influence the training accuracy or the improvement rate.
The main outcomes from this experiment is that the training accuracy increases as the number of hidden units increases and the learning rate increases. Another outcome is that the validation and generalization accuracy fluctuates heavily past a certain number of epochs, whereas the training accuracy steadily increases, as seen in the learning profile graph.

Experiment 3

In this experiment, the appropriate letter should be classified by the neural network, according to the pattern's exact letter class. For example, if the pattern's class is 'E', then the fourth output unit's value should be 1.0, where every other output unit's value should be 0.0. 26 output units are used, one for each letter.

This experiment will be the most difficult to generalize, since each pattern's exact class needs to be predicted. Therefore, a maximum number of 1000 epochs is used, and the desired accuracy is 90%.

Five training samples will be used. The default parameters used are ones that have been most successful in the previous experiments: a learning rate of 0.5, a momentum of 0.5, and a number of hidden units equal to the average of the number of input units and output units. In this case, (16 + 26)/2 = 21 hidden units are used. The results of five training sessions using the mentioned values are found below.

		Hidden Units: 21		Learning Rate: 0.5		Momentum: 0.5	
Training Run	Epoch Reached	Training Accuracy	Validation Accuracy	Generalization Accuracy	Max Epoch	Desired Accuracy	Overfitting
1	1000	71.1833	67.55	66.85	1	0	0
2	1000	71.9416	68.7	67.25	1	0	0
3	1000	72.15	68.225	67.125	1	0	0
4	1000	70.2833	68.225	67.525	1	0	0
5	1000	71.0833	67.95	66.55	1	0	0
	1000	71.3283	68.13	67.06	5	0	0

Tuning the Number of Hidden Units

In the previous set of training sessions, 21 hidden units were used. The results of a lower number of hidden units, 18, are below.

		Hidden Units: 18		Learning Rate: 0.5		Momentum: 0.5	
Training Run	Epoch Reached	Training Accuracy	Validation Accuracy	Generalization Accuracy	Max Epoch	Desired Accuracy	Overfitting

1	1000	70.94166	67.4	66.75	1	0	0
2	1000	68.475	66.4	66.725	1	0	0
3	1000	70.2416	66.4	66.375	1	0	0
4	1000	70.1	66.025	66.375	1	0	0
5	1000	69.6583	66.2	65.3	1	0	0
	1000	69.883312	66.485	66.305	5	0	0

In contrast, the results of a higher number of hidden units, 24, is shown below. A single training run was done.

		Hidden Units: 24		Learning Rate: 0.5		Momentum: 0.5	
Training Run	Epoch Reached	Training Accuracy	Validation Accuracy	Generalization Accuracy	Max Epoch	Desired Accuracy	Overfitting
1	1000	74.4666	71.375	70.85	1	0	0

From the results gathered, it can be seen that a higher number of hidden units used results in a higher training accuracy. This is in accordance with experiment 2.

Tuning the Learning Rate

In the previous section, a large learning rate was used (0.05). The following table shows a single training run where a smaller learning rate of 0.05 is used.

		Hidden Units: 24		**Learning Rate: 0.05**		Momentum: 0.5	
Training Run	Epoch Reached	Training Accuracy	Validation Accuracy	Generalization Accuracy	Max Epoch	Desired Accuracy	Overfitting
1	1000	70.9166	69.175	68.025	1	0	0

This result is enough to confirm the previous outcome that a lower learning rate results in a lower training accuracy.

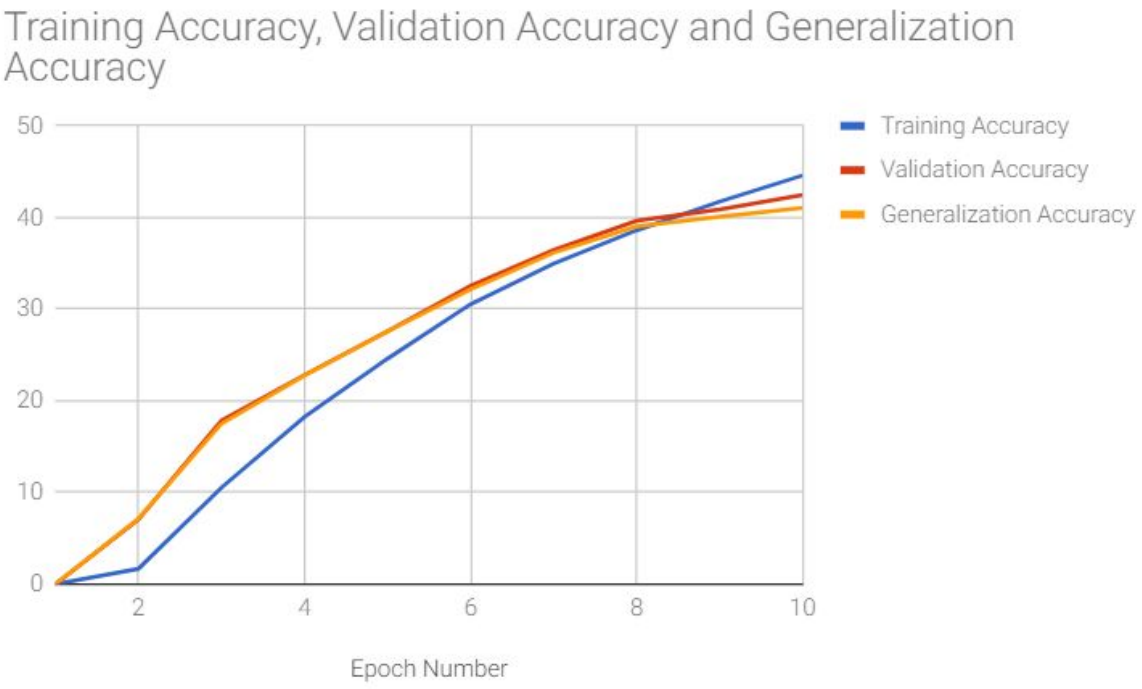
Tuning the Momentum

The results of a large momentum value (0.5) was already established. The results of a momentum of 0.05 can be seen below.

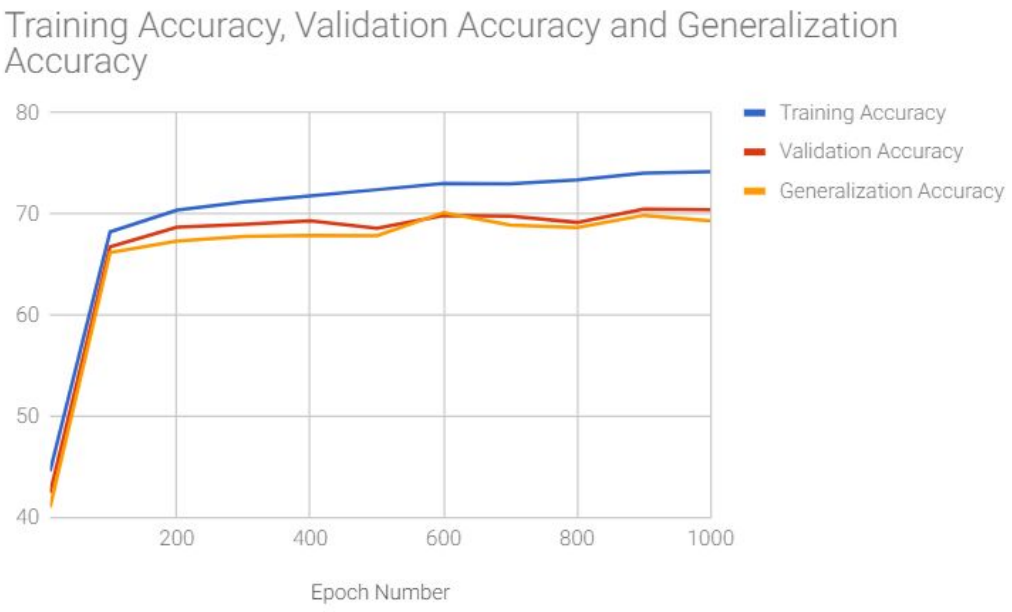
		Hidden Units: 24		Learning Rate: 0.5		**Momentum: 0.05**	
Training Run	Epoch Reached	Training Accuracy	Validation Accuracy	Generalization Accuracy	Max Epoch	Desired Accuracy	Overfitting
1	1000	74.1666	70.425	69.325	1	0	0

Once again, this result confirms the takeaway from previous experiments that the momentum has little to no impact on the training accuracy and improvement rate of the neural network.

When looking at the learning profiles of the experiment using the previous training run, it can be seen that it is slightly different to the learning profile graphs of the previous experiments.



This graph shows the accuracy achieved in the first 10 epochs. It can be seen that, unlike the other experiments, an accuracy of 0% is reached after the first epoch, suggesting that the network was unable to predict a single pattern initially. This is understandable since there is a low probability that an output will correspond to the pattern's letter class. Another thing to notice is that there is no dip in classification accuracy as there was in the previous experiments. This is possibly as a result of all output values being incorrect initially, and all output units could be wrong, therefore there is no "crossing over" point. As expected, the training accuracy surpasses the validation and generalization accuracy after a few epochs.



From the above graph of the entire training session, it can be seen how initially the training accuracy rises sharply before constantly rising. The validation and generalization accuracy fluctuates, similarly to experiment 2. The reason for this is possibly that the weights are updated according to the training dataset, and not the validation and generalization datasets.