**BME445 NEURAL BIOELECTRICITY**
# Lab 5: ARTIFICIAL NEURAL NETWORKS (ANNs)
**November 22, 2023 - PRA0102**

Liza Babaoglu - 1006026532
Deniz Uzun - 1006035005
Nicholas Heeralal - 1005867475

## Purpose

In this lab, we aim to train an Artificial Neural Network (ANN) to detect and classify brain states (seizure vs non-seizure) based on a training set from electroencephalogram (EEG) recordings. We will investigate the effects of hidden units and hidden layer numbers, as well as the effects of learning rate and momentum parameters on the ANN models, by running controlled experiment sets on MATLAB.

## Results/Discussion

***1) Attach the MSE plot for each network trained. There should be 12 figures in total (3 figures per each case).***

### Experiment 1A
### Network 1:

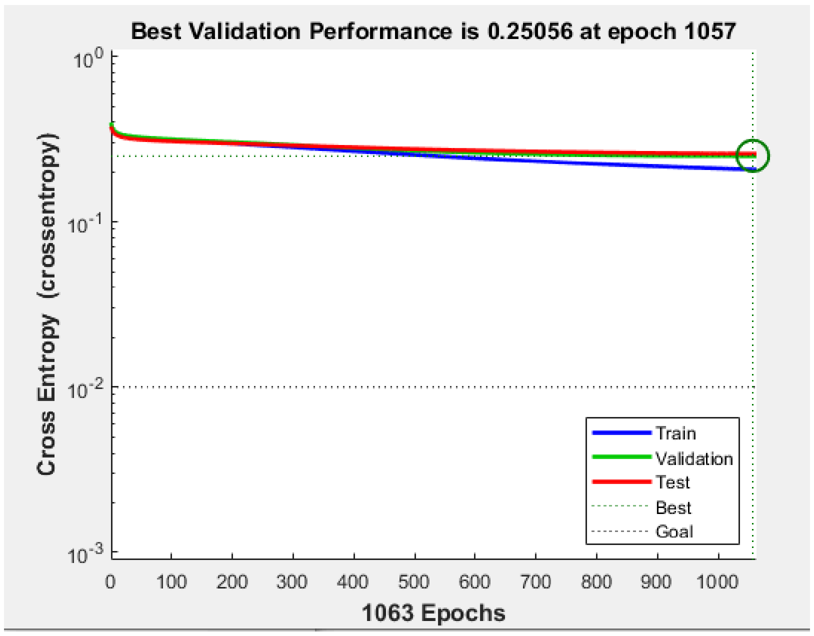

*Figure 1: ANN Performance corresponding to Experiment 1A, Network 1*

**Network 2:**

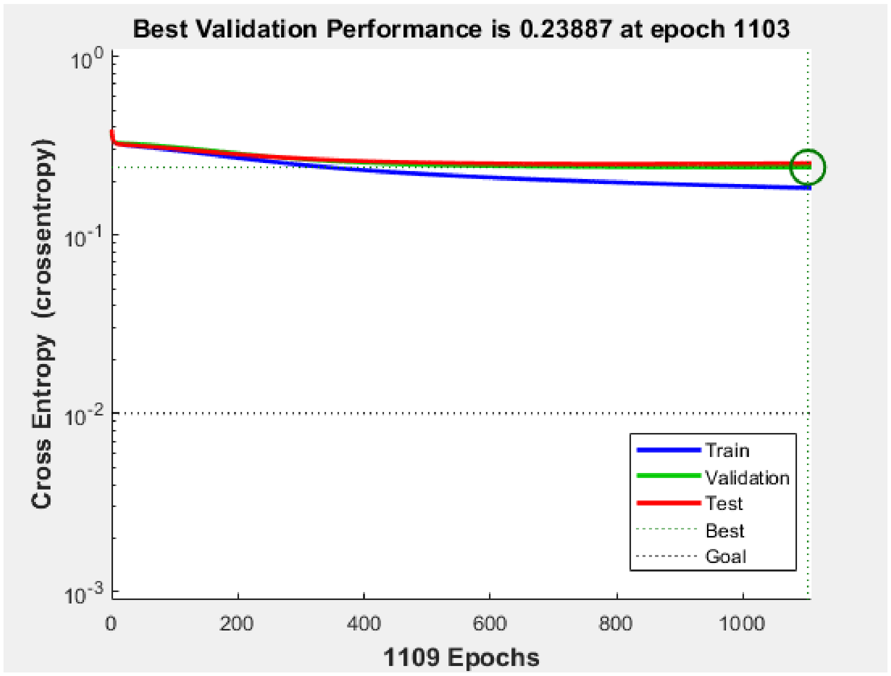

*Figure 2: ANN Performance corresponding to Experiment 1A, Network 2*

**Network 3:**

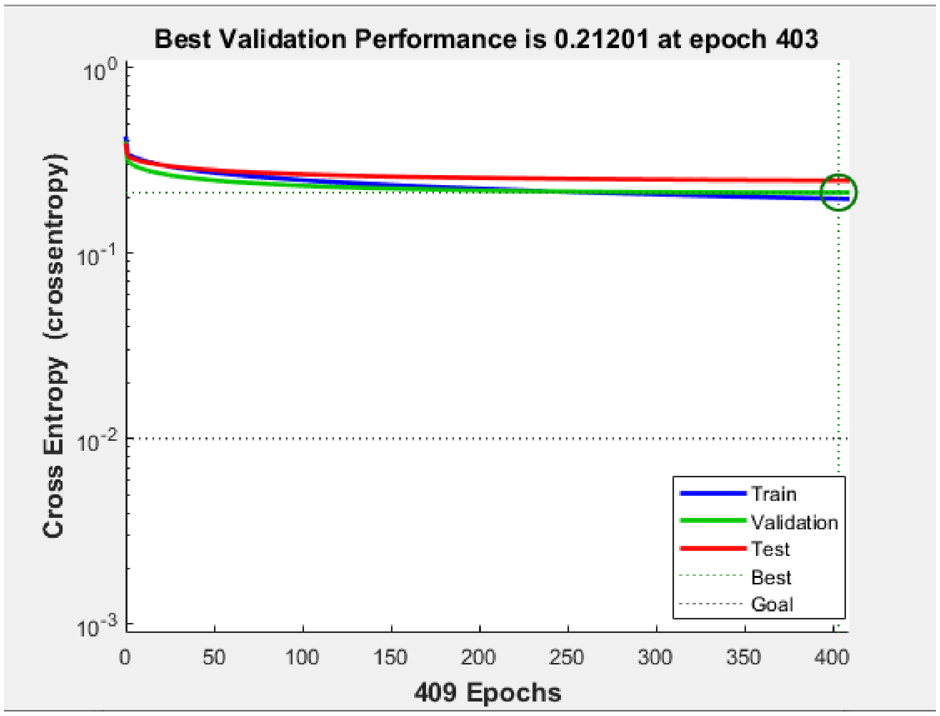

*Figure 3: ANN Performance corresponding to Experiment 1A, Network 3*

**Experiment 1B**
**Network 1**

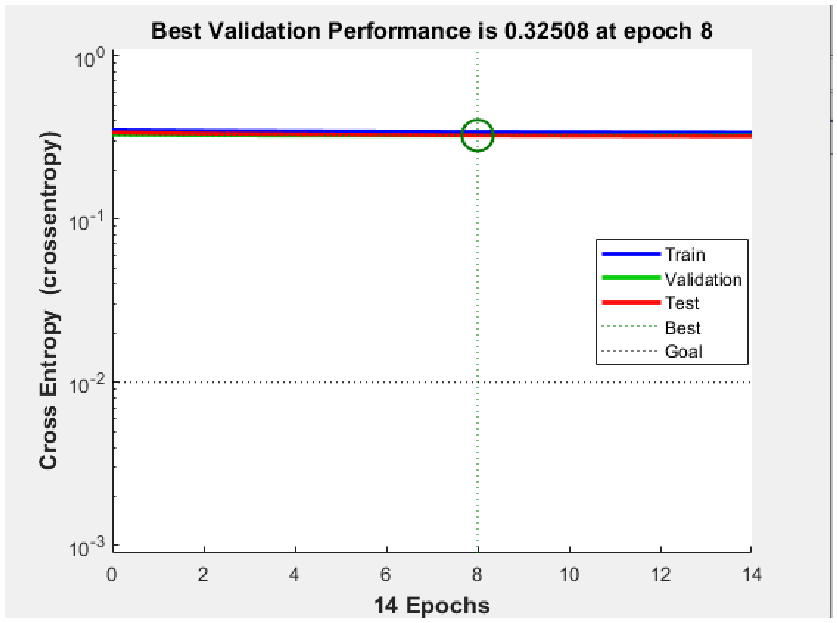

*Figure 4: ANN Performance corresponding to Experiment 1B, Network 1*

**Network 2**

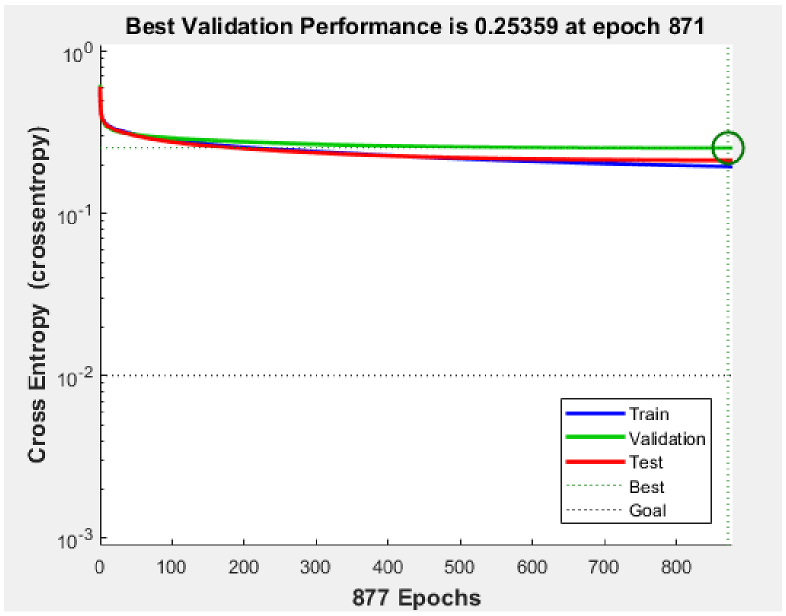

*Figure 5: ANN Performance corresponding to Experiment 1B, Network 2*

**Network 3**

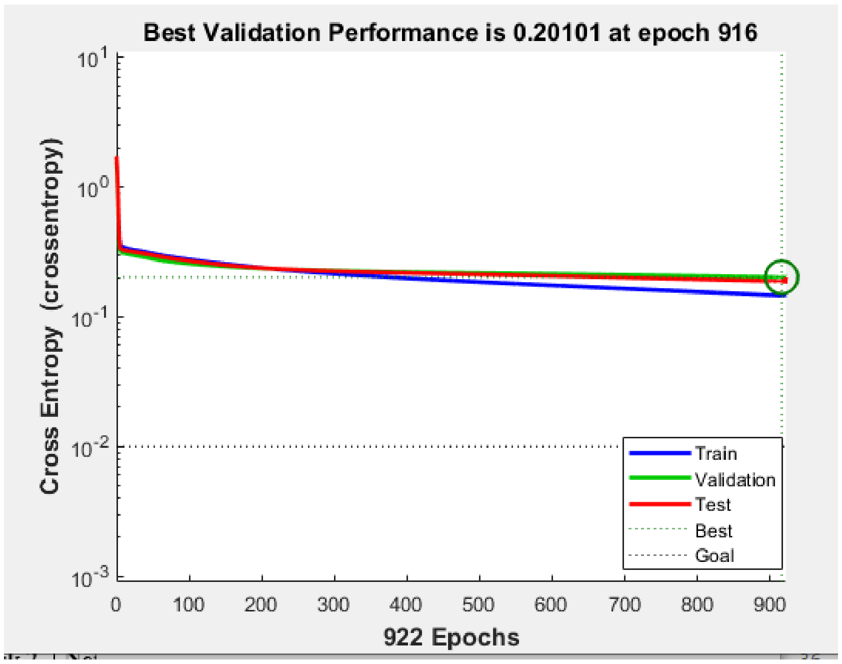

*Figure 6: ANN Performance corresponding to Experiment 1B, Network 3*

**Experiment 1C**
**Network 1**

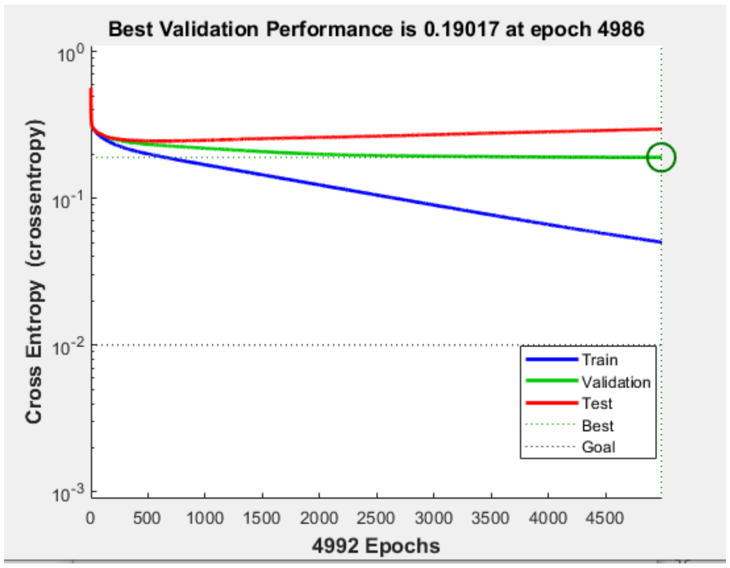

*Figure 7: ANN Performance corresponding to Experiment 1C, Network 1*

**Network 2**

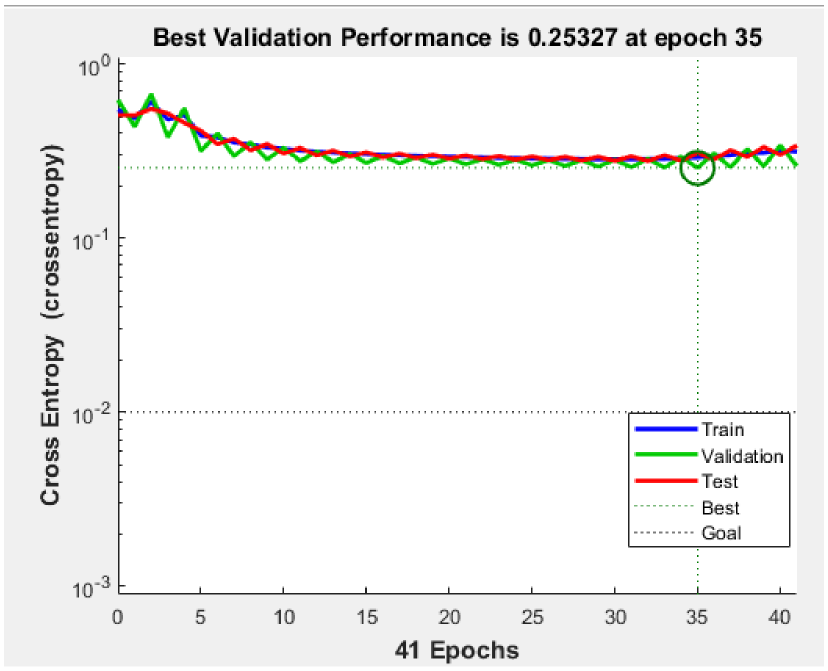

*Figure 8: ANN Performance corresponding to Experiment 1C, Network 2*

**Network 3**

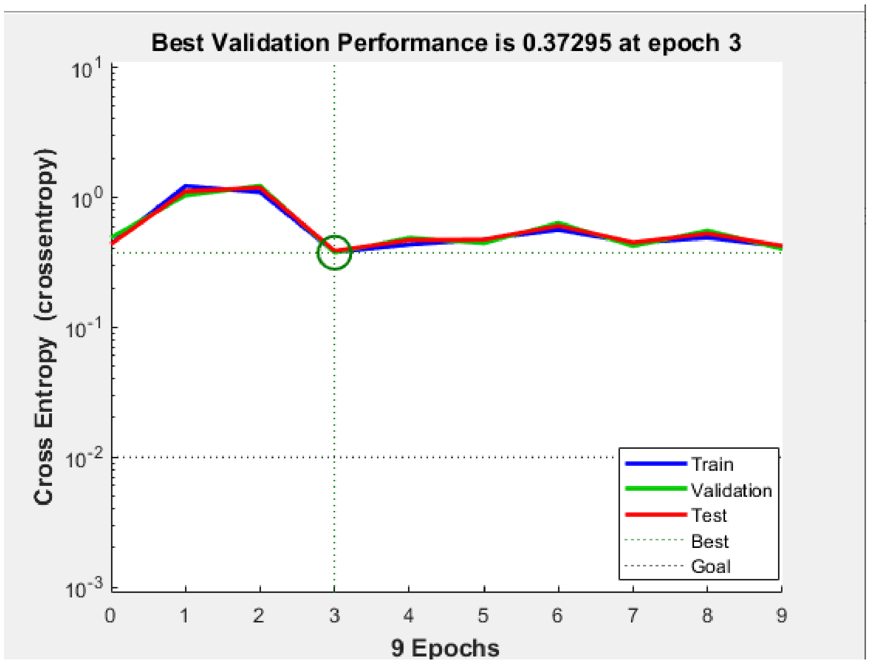

*Figure 9: ANN Performance corresponding to Experiment 1C, Network 3*

**<u>Experiment II</u>**
**<u>Network 1</u>**

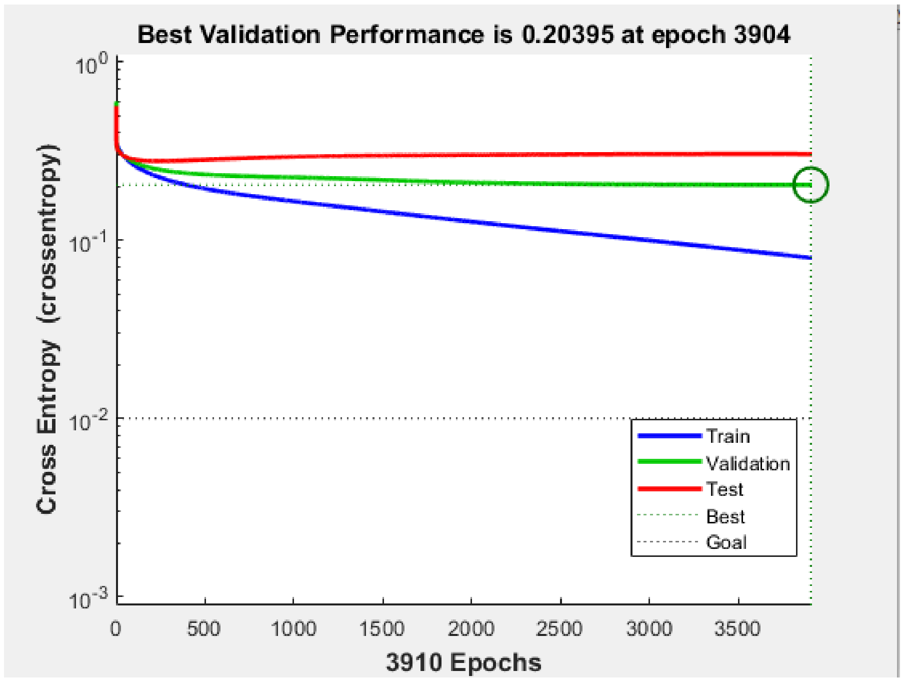

*Figure 10: ANN Performance corresponding to Experiment II, Network 1*

**<u>Network 2</u>**

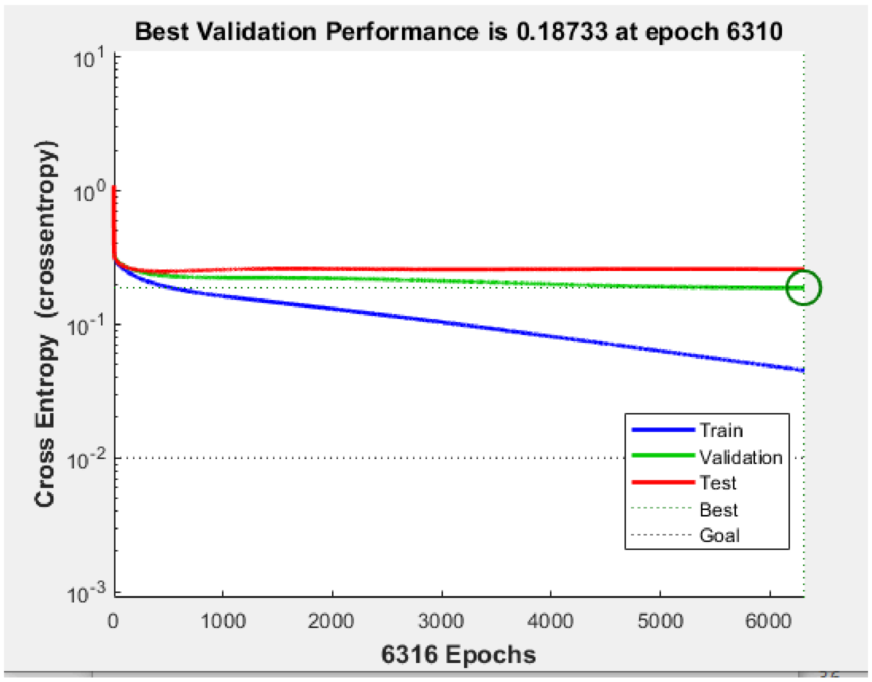

*Figure 11: ANN Performance corresponding to Experiment II, Network 2*

**Network 3**

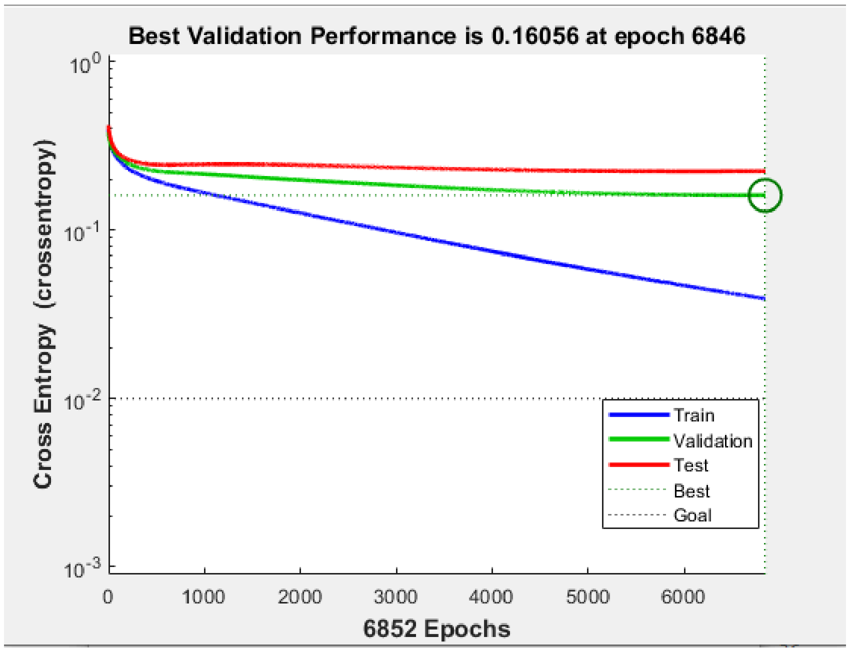

*Figure 12: ANN Performance corresponding to Experiment II, Network 3*

***2) Summarize the accuracy of each network on the test set. After the training MSE has reached the desired level, simulate the trained network on the test set and evaluate the percentage of correct network response for each brain state, as well as the overall accuracy. Also, indicate the MSE goal used to train the network (i.e. 0.01 or 0.05).***

An MSE goal of 0.1 was used to train all networks.

| | **Experiment IA** | | | **Experiment IB** | | |
|---|---|---|---|---|---|---|
| | **Network 1** | **Network 2** | **Network 3** | **Network 1** | **Network 2** | **Network 3** |
| **Non-seizure state** | 38.22 % | 33.05 % | 31.61 % | 31.32 % | 37.36 % | 39.94 % |
| **Seizure state** | 33.62 % | 37.36 % | 34.77 % | 33.33 % | 36.49 % | 36.78 % |
| **Overall Accuracy** | 73.28 % | 70.40 % | 66.38 % | 64.66 % | 73.85 % | 76.72 % |

| | Experiment IC | | | Experiment II | | |
|---|---|---|---|---|---|---|
| | **Network 1** | **Network 2** | **Network 3** | **Network 1** | **Network 2** | **Network 3** |
| **Non-seizure state** | 40.23 % | 15.52 % | 3.74 % | 39.08 % | 39.66 % | 37.36 % |
| **Seizure state** | 38.51 % | 42.53 % | 50.00 % | 39.08 % | 38.22 % | 40.23 % |
| **Overall Accuracy** | 78.74 % | 58.05 % | 53.74 % | 78.16 % | 77.87 % | 77.59 % |

**Experiment IA**
**Network 1:** Achieved an overall accuracy of 73.28%, with 38.22% for non-seizure states and 33.62% for seizure states.
**Network 2:** Had an overall accuracy of 70.40%, with 33.05% for non-seizure states and 37.36% for seizure states.
**Network 3:** Showed the lowest overall accuracy among the three in this experiment at 66.38%, with 31.61% for non-seizure states and 34.77% for seizure states.
**Experiment IB**
**Network 1:** Had an overall accuracy of 64.66%, with 31.32% for non-seizure states and 33.33% for seizure states.
**Network 2:** Improved to an overall accuracy of 73.85%, with 37.36% for non-seizure states and 36.49% for seizure states.
**Network 3:** Achieved the highest overall accuracy in this set at 76.72%, with 39.94% for non-seizure states and 36.78% for seizure states.
**Experiment IC**
**Network 1:** Achieved an overall accuracy of 78.74%, with 40.23% for non-seizure states and 38.51% for seizure states.
**Network 2:** Had a significant drop in overall accuracy to 58.05%, with 15.52% for non-seizure states and 42.53% for seizure states.
**Network 3:** The overall accuracy was 53.74%, with 3.74% for non-seizure states and a higher rate of 50.00% for seizure states.
**Experiment II**
**Network 1:** Had an overall accuracy of 78.16%, with 39.08% for both non-seizure and seizure states.
**Network 2:** Showed a similar overall accuracy to Network 1, at 77.87%, with 39.66% for non-seizure states and 38.22% for seizure states.
**Network 3:** Had a slightly lower overall accuracy of 77.59%, with 37.36% for non-seizure states and 40.23% for seizure states.

## <u>**EFFECT OF HIDDEN UNITS:**</u>

In Experiment 1A, we only change the number of neurons in one hidden layer to observe the effects of hidden units (size of the hidden layer).

***I) Does an increased number of hidden units always imply better performance on testing? Why or why not?***
As seen from the overall accuracy results, as well as validation performance results in Experiment 1A, the performance on testing does not necessarily increase with increasing the number of hidden units in a layer. In fact, we do see a decrease in performance and accuracy. This is primarily due to overtraining the data on a given training set, where the network model overfits, performing very well on the training data but poorly on the unseen testing data. Increasing the number of hidden units decreases generality. In our case, we had the best results with the lowest number of hidden units (5), this may be due to the relative simplicity of our problem, where a simpler model with fewer neurons captured the predominant underlying trends necessary without overfitting to any training-data specific noise.

***II) If a classification problem can be solved using an ANN with 10 hidden units, would an ANN with 40 units also solve the same problem? How about vice versa?***
A classification problem solvable with an ANN of 10 hidden units may be addressed by an ANN with 40 units, however, this larger and more complex network, is also more susceptible to overfitting, where the model learns the specificities of the training data rather than the underlying pattern, negatively impacting its performance on new data. Additionally, the increased complexity may necessitate more advanced training methods and greater computational resources. Conversely, the smaller network with 10 hidden units might lack the complexity and sophistication, potentially failing to recognize all relevant patterns essential to accurately classify. There has to be a balance between model complexity and generalization.

***III) What are the pros and cons of having more hidden units?***
The advantage of more hidden units is that increased model capacity captures complex patterns in the data. This may improve generality if these additional units can help the network generalize better to unseen data by building a more sophisticated model based on the input data. However, an increased number of hidden units may also hinder generalization if this sophistication results in overfitting. Other disadvantages include optimization challenges, larger data requirements and increased computational cost, if applicable.

## EFFECT OF NUMBER OF HIDDEN LAYERS:

In Experiment 1B, we only change the number of hidden layers to observe the effects of hidden layers on the performance of our network.

***I) Does increased number of hidden layers always imply better performance on testing? Why or why not?***

Based on our data, as seen from the table, the overall accuracy increased as we increased the number of hidden layers. However, the validation performance results in Experiment 1B slightly decreased as we increased the number of hidden layers. Increasing the number of hidden layers does not always imply better performance on testing. In Network 1 (with no hidden layer), we had the lowest accuracy result. This means that classifying seizures may not be a simple linear problem where we can only classify based on input and output data. However, we did reach the highest validation performance in earlier epochs in Network 1, which may be because it was easier to find a good solution in a space without hidden layers as there are fewer parameters to optimize. Generally, deciding on the number of hidden layers is data/problem-specific. There again needs to be a balance between model complexity and generalization.

***II) If a classification problem can be solved using an ANN without any hidden layers, would an ANN having one hidden layer also solve the same problem? How about vice versa?***

If a classification problem can be resolved using an ANN without any hidden layers, implying the problem is linearly separable, then an ANN with one hidden layer should also be able to solve it. The addition of a hidden layer provides the network with the ability to learn not only linear but also non-linear relationships, enhancing its versatility. However, one disadvantage might be overfitting, as the more complex model has the potential to over-train and over-learn the training data. Conversely, if a problem is solvable with an ANN that has a hidden layer, it might not be solvable with an ANN without hidden layers, particularly if the problem is non-linear. The absence of hidden layers restricts the network to linear decision boundaries, which may not be sufficient for more complex problems that require the hidden layers.

***III) Can an artificial neural network without any hidden layers be trained to perform the "AND" logic? How about the "OR" logic?***

An ANN with no hidden layer can be trained to perform "AND" and "OR" logic. These are simple functions that are linearly separable, meaning that a decision boundary can be drawn with a straight line to separate the outputs. This is a single-layer perceptron, where the perceptron learns the weights and bias.

**<u>STEP SIZE/ LEARNING RATE CONSIDERATIONS:</u>**
In Experiment 1C, we only change the learning rate, by keeping the number of hidden layers constant at 1, and the number of hidden layer units constant at 20. The learning rates are 0.1, 0.5, and 1, respectively, for Networks 1, 2, and 3.

***I) Is a larger step size always better? If you are using only a simple gradient descent search algorithm, what potential problems would you run into if the step size is too large? Too small?***
Based on our data, as seen from the table, the overall accuracy significantly decreased as we increased the step size/learning rate, in Experiment 1C. Therefore, it is clear that a larger step size is not always better. A too large step size may result in the algorithm overshooting the optimal solution. A large step size can also result in oscillation around a minimum or diverge entirely. A too small step size will take a long time to converge, but more significantly it may get stuck in a local minimum in the gradient descent search.

**<u>EFFECT OF MOMENTUM:</u>**
In Experiment II, we only change the momentum value, by keeping the number of hidden layers constant at 1, and the number of hidden layer units constant at 40. The momentum values are 0.1, 0.5, and 0.9, respectively, for Networks 1, 2, and 3.

***I) What is the motivation in using momentum with a simple gradient descent algorithm?***
The motivation for using momentum with a simple gradient descent algorithm is to escape the local minima. The momentum value builds up enough velocity ("momentum") to escape the local minima and navigate plateaus, especially in non-convex loss functions.

***II) What potential problems are there if you use a momentum term that is too large?***
A momentum term that is excessively high can result in numerical instability within the gradient descent optimization process. This is due to the momentum term excessively amplifying the influence of past gradients, which may not only be irrelevant but could also be harmful to the current direction of descent. It can also lead to a failure in convergence as there may be oscillations or diversion especially if the learning rate is fixed and too large to compensate for the momentum's effects. In Experiment II, we do see a slight decrease in the overall accuracy and performance as we increase the momentum values. However, overall, they are promising, and haven't resulted in diversion since our learning rate was reasonably small, 0.1.

***4) For you particular ANNs, are there any specific test cases that are more difficult to classify? Can you speculate why?***

The momentum term in gradient descent helps accelerate the optimization in the direction of consistent gradients, potentially leading to faster convergence. However, its effectiveness can vary depending on the network configuration and the nature of the problem. Networks with too few neurons or layers (underfitting) may not capture the complexity of the data, leading to poor performance. Conversely, overly complex models (overfitting) can memorize the training data and perform poorly on new, unseen data. Different network configurations (like the number of layers and neurons) will respond differently to momentum. Networks with more layers or neurons might benefit more from higher momentum as they might be dealing with more complex error landscapes. Test cases with a lot of noise in the EEG can be difficult for ANNs to classify accurately. Noise can obscure or distort the underlying patterns in the data, making it harder for the network to make accurate predictions.

## Summary of Results/Discussion

In this lab, the training of ANNs for the classification of brain states has revealed several key insights. The number of hidden units and layers in an ANN impacts its ability to capture the complex relationships within the data, with a balance required to avoid overfitting. Our experiments demonstrate that while additional hidden units and layers can enhance the network's capacity, they also heighten the risk of overfitting and hinder generalization. We also observed the importance of tuning learning rate and momentum parameters. The utilization of correct learning rates and momentum values aids in overcoming local minima and accelerating convergence, whereas excessive values might result in instability and divergence. In conclusion, we learned the importance of tuning parameters, and their dependence on the type of data we are working with.