Pointers

- The overlap area factor is mentioned as the *Generalization Factor* in my code.
- I have used **sin(x)** as the **1-D function**
- The 70:30 training-to-testing split is generated randomly so the training and testing data are different for every run. I did this because I wanted to see a general trend for the effects of overlap area and time of convergence.
- Due to random generation of training and testing data, you might get different results upon running my code.
- All of the code is available on my GitHub repository, named, cmac-neural-network.

Algorithm

- Distribute the input and true output of a function over 100 evenly-spaced points.
- Split the data into 7:3 training to testing ratio. This split is random for each full execution.
- Calculate the output using training values, then calculate the absolute error to update the weights.
- The above step is followed for testing data.
- Iterate 35 times to vary the generalization factor from 1 to 35 and set final weights for each iteration.

Question 1

We will discuss the performance of the discrete CMAC after exploring the effect of overlap area on generalization and convergence time. Figures 3 and 4 show the worst and best performance of the discrete CMAC on the testing data.

Overlap Area

We observe that the training accuracy drastically decreases and then tries to reach its original highest value (See Figure 1) as the overlap area increases on generalization. This is because a generalization factor of 1 is essentially 1-to-1 mapping of data points. As the generalization increases, it becomes 1-to-many mapping of data points that decreases the accuracy during training. However, the decrease in training accuracy tunes the network to deal with errors and provide us with better results on the testing data. Thus, testing accuracy increases upon increase in overlap area on generalization (See figure 2).

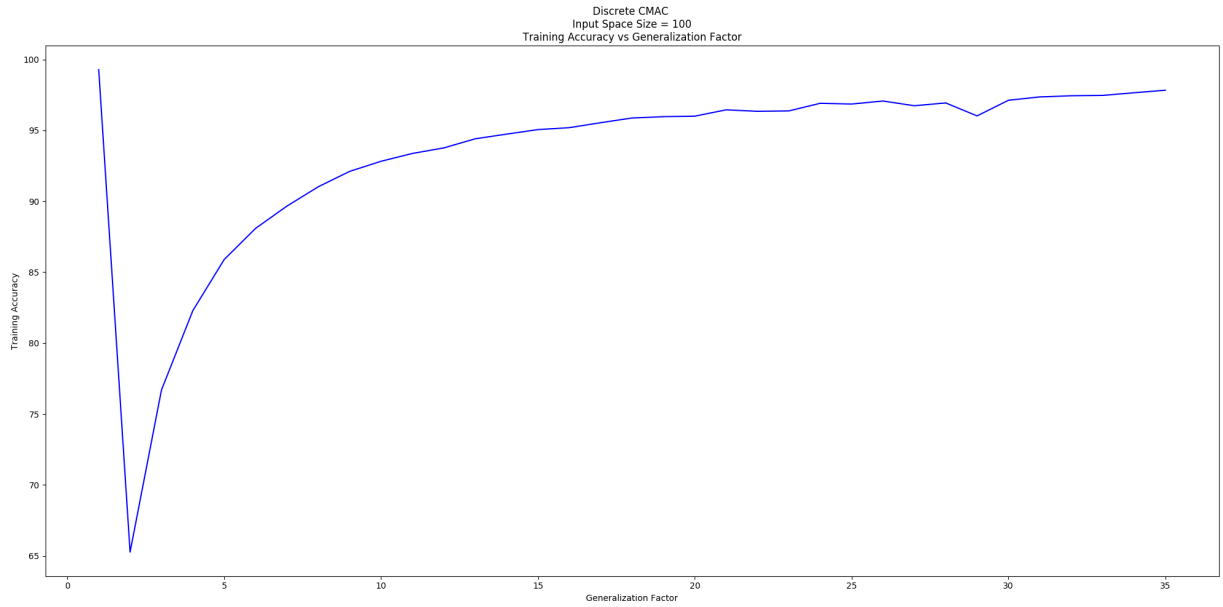

Figure 1: Training Accuracy with increase in Overlap Area

Time of Convergence

From Figure 3, it can be seen that accuracy increases with increase in time of convergence. A more accurate sense of convergence time can be gained from Figure 9. Figure 9 represents a graph of convergence time versus generalization factor. It can be seen that as generalization factor increases, convergence time also increases. As explained in the previous section, a generalization factor of n denotes 1-to-n mapping. Therefore, more complex mapping indicates more time will taken for global convergence.

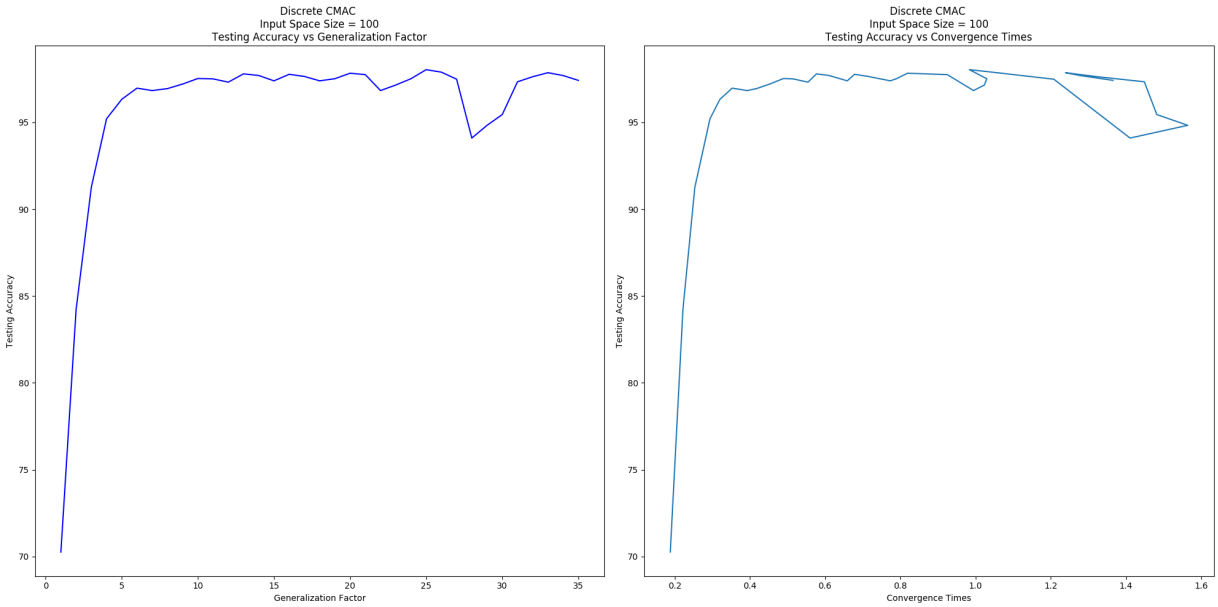

Figure 2: Performance of Discrete CMAC

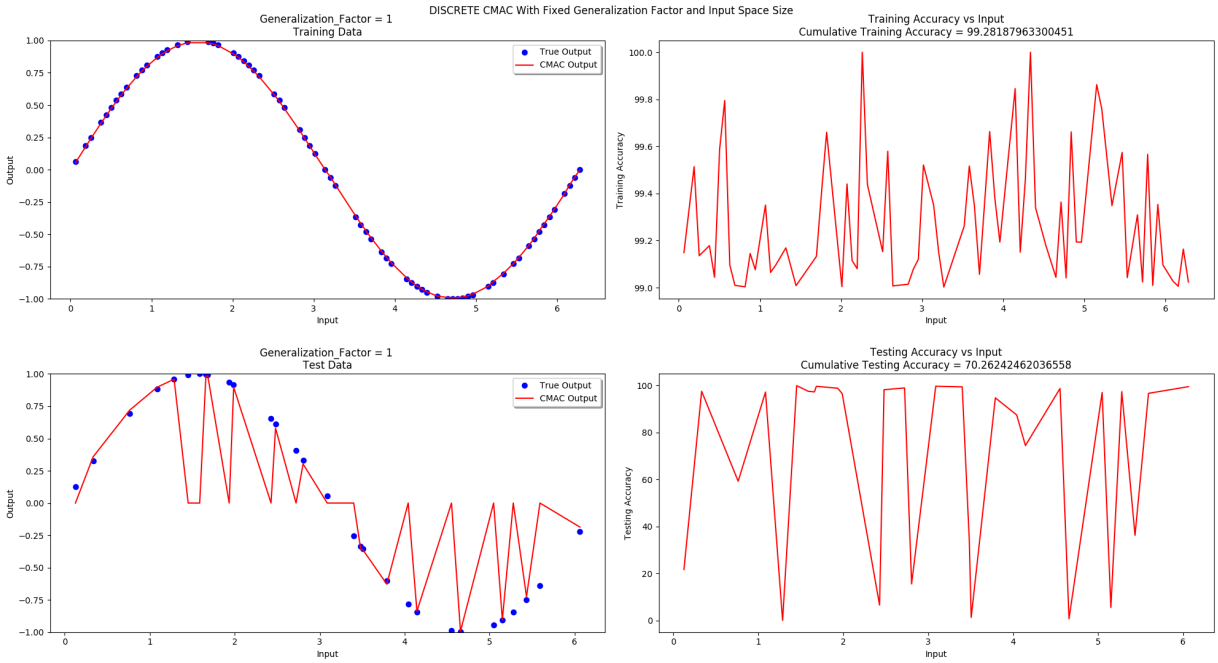

Figure 3: Worst Performance of Discrete CMAC on Testing Data

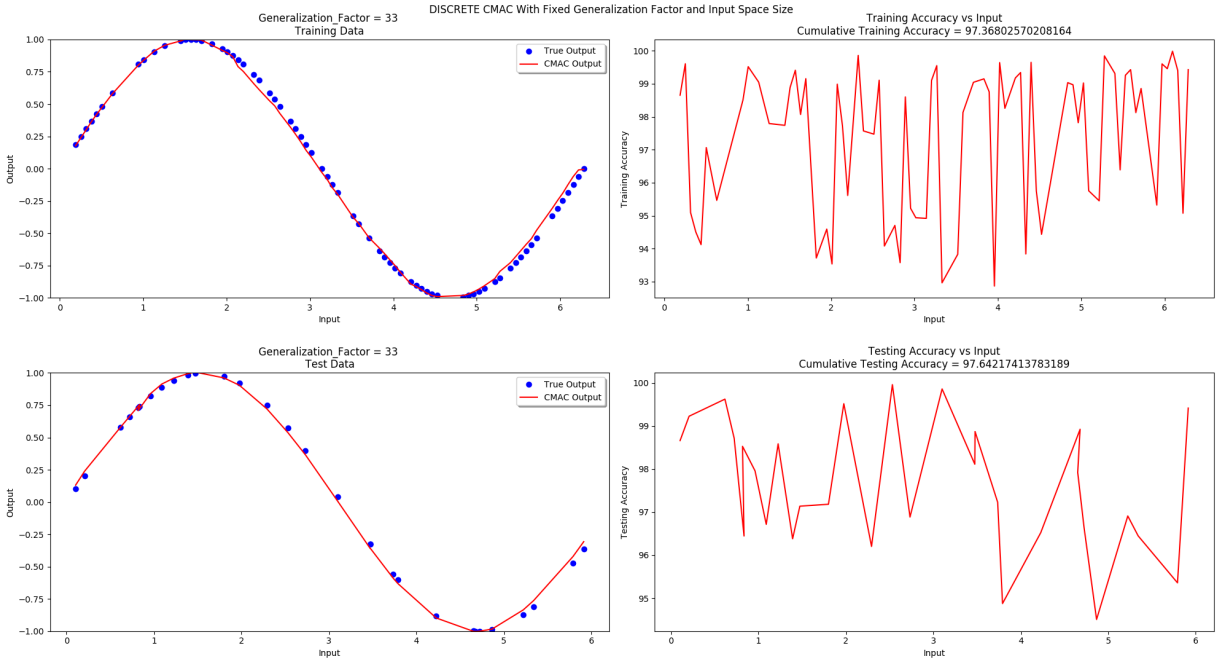

Figure 4: Best Performance of Discrete CMAC on Testing Data

Question 2

The conclusions made for effects of overlap area on generalization and time of convergence remain the same for a continuous CMAC. The only difference is testing accuracy and time of convergence are higher than that of a discrete CMAC. The convergence time is higher because a continuous CMAC employs more number of iterations for global convergence of data due to partial overlap. Please look at figures 5-8 for training-testing accuracies and performance of the continuous CMAC.

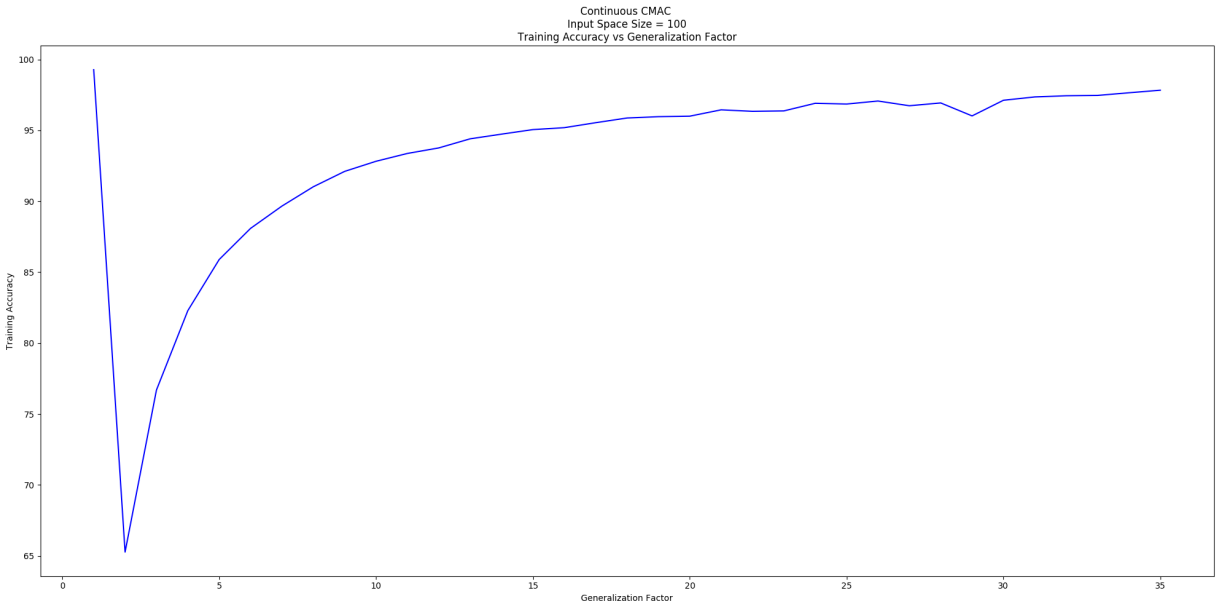

Figure 5: Training Accuracy with increase in Overlap Area

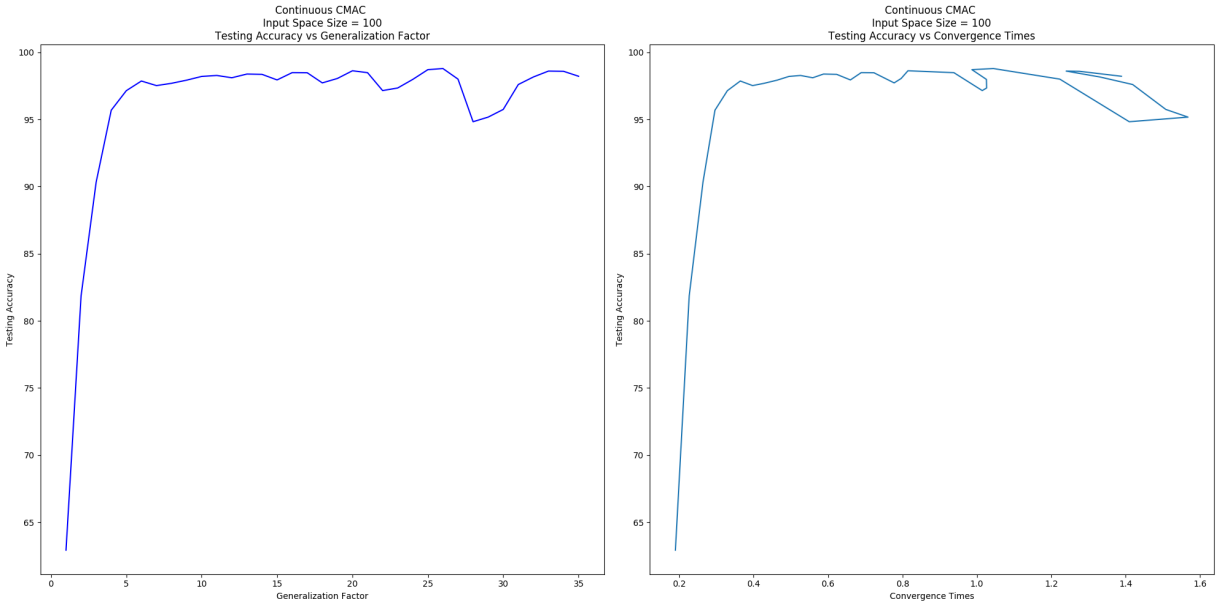

Figure 6: Performance of Continuous CMAC

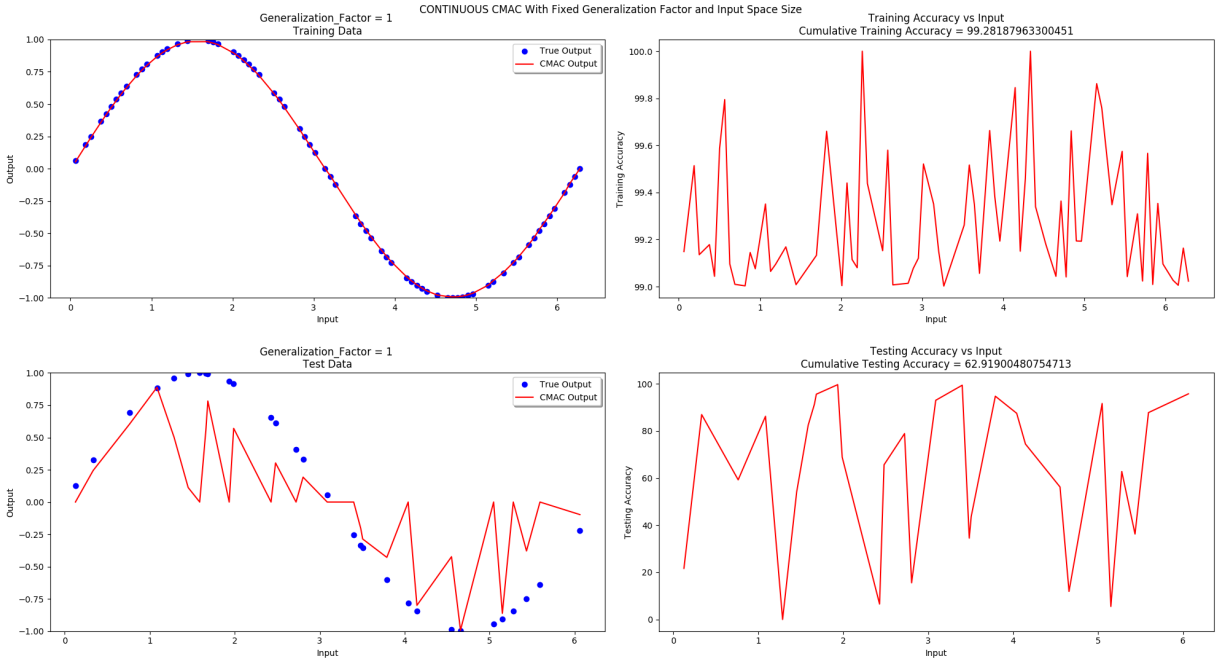

Figure 7: Worst Performance of Continuous CMAC on Testing Data

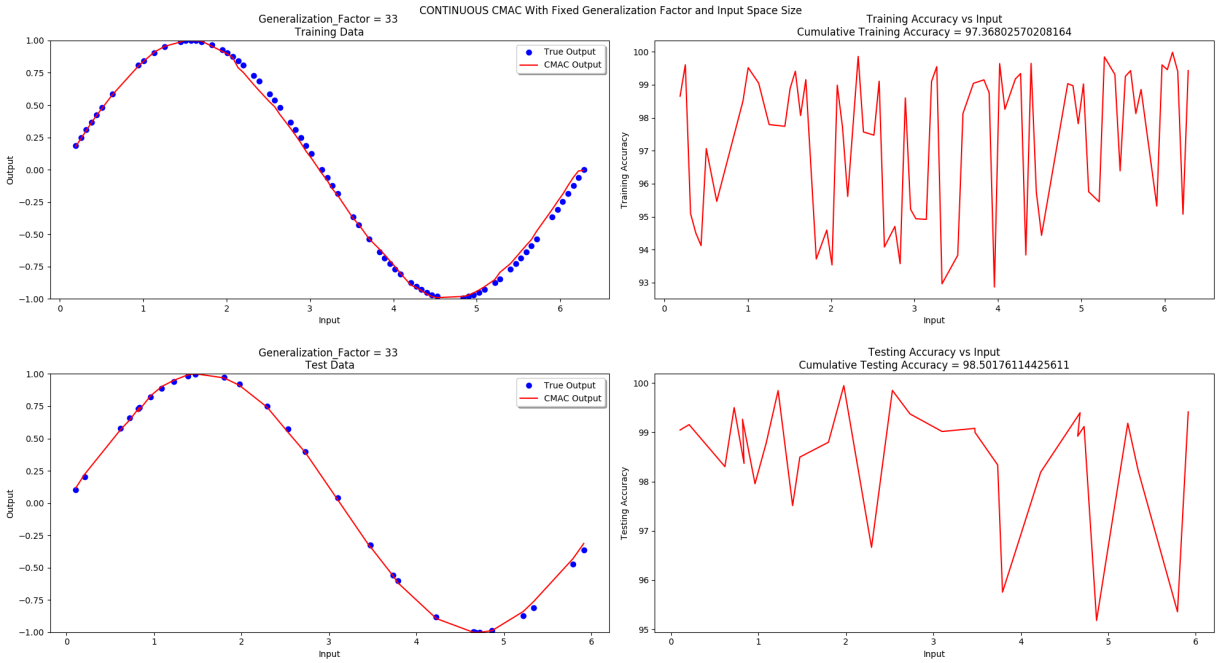

Figure 8: Best Performance of Continuous CMAC on Testing Data

Comparison of Discrete and Continuous CMACs

The accuracy and convergence time of continuous CMAC are slightly higher than that of discrete CMAC. If we want to choose one over the other, it depends on our application. The application will justify the trade-offs we can make. If the application can afford a delay a few milliseconds in convergence time for a 1-2% gain in accuracy, then we can employ a continuous CMAC for the application. Figure 9 compares testing accuracy and time of convergence of both the CMACs with respect to the generalization factor.

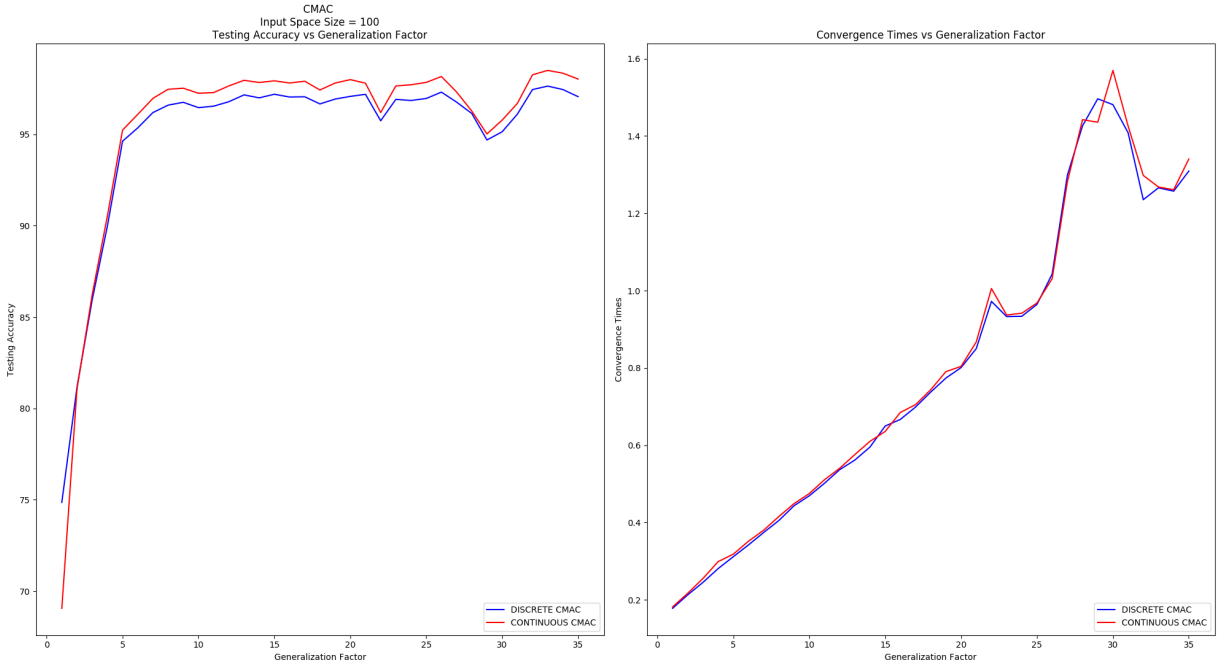

Figure 9: Comparison of Discrete and Continuous CMACs

Question 3

Recurrent connections are mainly used another type of neural network known as Recurrent Neural Network (RNN). In order to avoid the usage of time as an input, we can employ the logic of Finite State Machines (FSMs) to output the desired trajectory.

In an FSM, the next output of the system is decided based on the current state and condition of the system. This is really helpful in scenarios where the the network has to learn to draw trajectories that intersects itself such as learning to write in cursive letters or drawing the infinity symbol. A simple flow diagram of an FSM is given in figure 10. In the figure, S_i represents each state and the fraction along each arrow reprsents the condition for transitioning from one state to another.

We will have to decide upon some general conditions that acknowledge the change of state and alter the weights function to take conditions into consideration while update the weights.

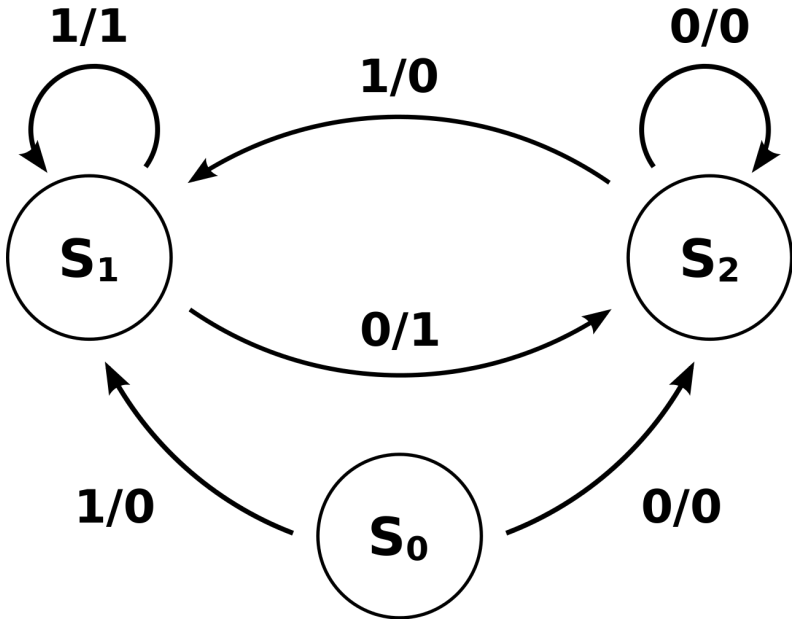

Figure 10: A Simple Flow Diagram of an FSM