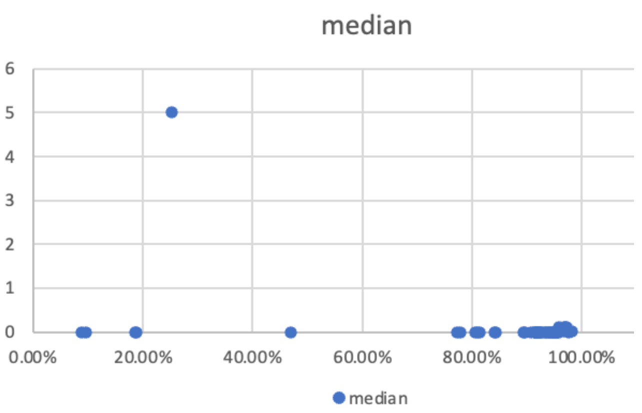
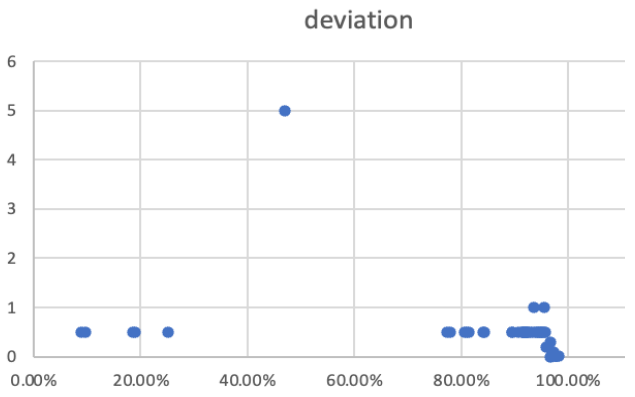
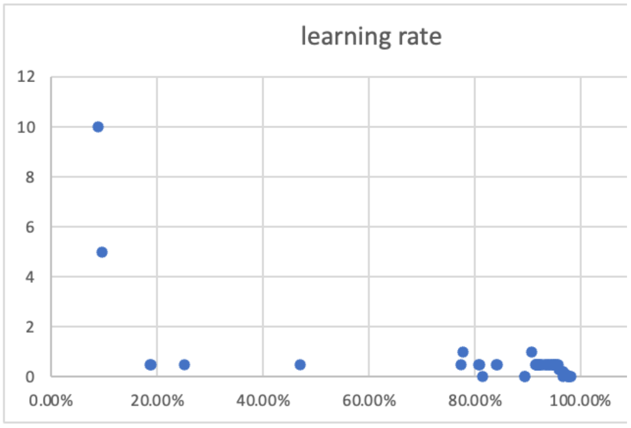
Pear/Thitapa Saelee saeleethit@myvuw.ac.nz ID : 300622594

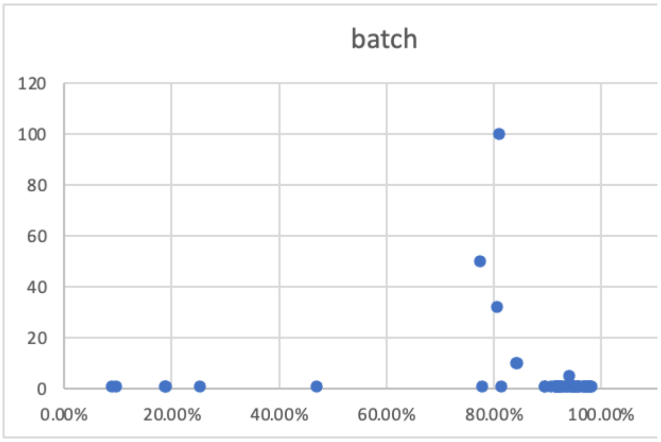
Reflection

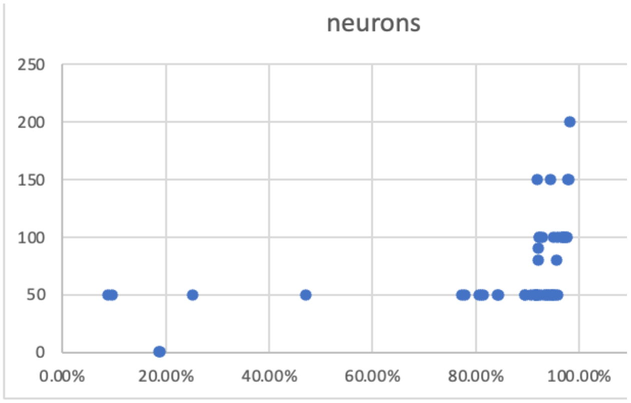
1. **Include all combinations of network meta-parameters you tried. To save space and to make it more clear use graphs of recognition rate (%) vs parameters.**

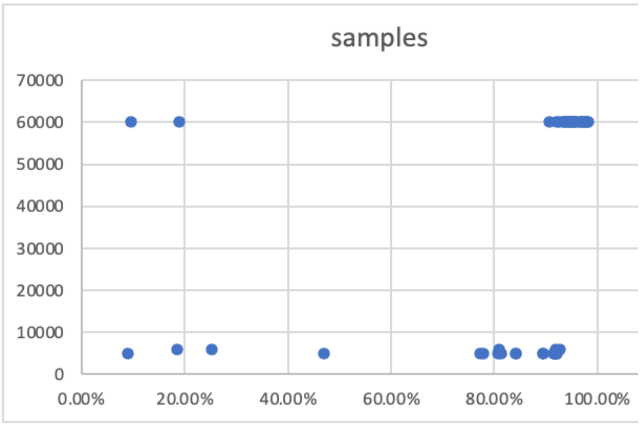












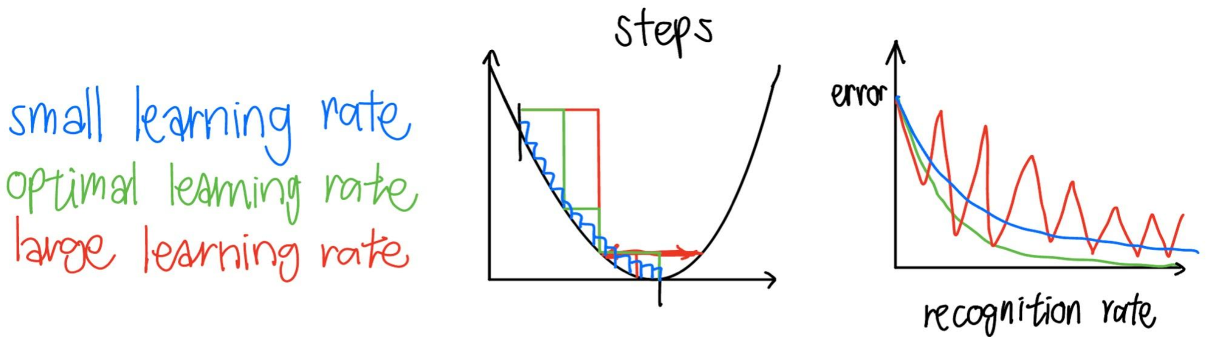
2. **Explain what was your strategy for tuning network parameters and why you decided to follow it. Something along the lines:" We started with these values because.... After that we decided that now we can try...".**

I started with smaller n_samples. Since there are 60,000 training data samples. I started with 1/12 of the total training set, which is 5000 samples so we know some of the data inputs' patterns and a rough recognition rate. Then started in_median, in_deviation and learning_rate values of 0 then increased them. After that, I started with 1 of n_hidden and n_epoch then adjusting them to see the changes in the training set. If the recognition rate is high, then I come back to increase n_samples and eventually increase n_samples to 60,000, so I know the pattern of the total training set.

3. **What was the final value of the learning rate and how did you arrive to that?**

- The small value of learning rate steps really small, but eventually it will reach the optimal region, where the error is at 0, although it takes a longer training time. By using the small value of learning rate, it demonstrates slow training time along with low error values making stably accurate (blue line) and lead to pretty high recognition rate.
- On the other hand, the large value of learning rate steps really big, causing it to bounce back and forth and not reach the optimal region at the end. By using the large value of learning rate, it demonstrates fast training time along with high error values making unstable accuracy (red line) and lead to low recognition rate.
- The optimal (medium) learning rate steps not too big and not too small, reaches the optimal region faster than small learning rate, but also does not bounce back and forth like the large learning rate does. By using the optimal value of learning rate, it demonstrates quite fast training time along with low error values making stably accurate (green line) and lead to high recognition rate.

- As the reasons above, the facts match with my trials, I have tried the learning rate of 10 and the recognition rate is significantly lower than **my final learning rate of 0.01**.



4. **How the number of hidden neurons affected network performance. What is the reason?**

- 28 x 28 pixels of an image (input 28 x 28 = 784) and 10 output digits (0,1,2,3,4,5,6,7,8,9). One neuron is one section of input data and two neurons are two sections of input data and so on. In the pixels, some of the pixels are black. By adding several neurons, it creates a lot of regions inside the black pixels, so that a group of neurons will be fired and the input images are likely to be shown, which identifies the shape of the certain numbers' characteristics. In general, the more neurons, the better the recognition rate of network performance. Although, it significantly prolongs the training process. The number of hidden layers affects the training ability of the network performance. The presence of few hidden neurons prevents the capacity from learning enough patterns between inputs. However, too many neurons can lead to "overfitting", which is tuning them to all peculiarities of input images. Since images are different, if we are tuning them too nicely to particular images and particular training sets. It leads to missing things in the training set, which causes them to not work as well as they are supposed to.

5. **Same for initial weights value deviation? Why did you select this particular value?**

- The input data of median 0 multiply to weights then add them all together to the activation function. Activation function is when the neuron learns fastest when its input is around 0. If the input of the neuron is very high, it will never learn anything because the slope is too small as it basically kills the neuron, if one of the weights is too high. We want the number before it goes to the activation function to be around 0 because if the median of input data is 0, no matter if we multiply it to positive number(s) or negative number(s), the median will always be 0.
- Deviation cannot be 0 because then the weights are all the same, like assignment 2, it does not work in a multi-layer network. Deviation can not be too large because some of the value of the weight will also be too big, which leads to killing the neuron (in the area where it will never learn anything). Optimum deviation value is so much less than 1. I pick 0.01 because in this project, we have a lot more inputs compared to the assignment. On the other hand, in the assignment, the optimum deviation value is 1 because we only have two inputs in the assignment.

6. **Did increasing the number of samples significantly change optimum values of other parameters? Explain why.**

- I started with 5000 samples, however, it only picks up a small part of the samples, so only some of the patterns and not all patterns. I notice that the more training data samples, the better the recognition rate because it is learning more samples. But from my 55 trials, increasing the number of the samples does not significantly change recognition rate nor the optimum values of other parameters. For example, 5000 samples to 60000 samples only increase 2% of the recognition rate in my trials.

7. **What performed better: on-line training or mini-batch training? What does it say about our input data?**

- On-line training performed better as they do the tests, change the weights and biases and do the test, which steps immediately for each weight and bias.
- Mini-batch training: do the tests, do not change the weights and biases and do the test again, so it is faster for the amount of images done, but it averages them out, so the performance is worse in this data set.
- When we draw error changes, like perfect parabola shape, but in reality, it is far from perfect. As it has lots of local minimum areas, we need to move a lot to get out of the local minimum areas, if we get stuck there. So, for mini-batch training, steps can be not enough to get out of local minimums and in this project, there are a lot of local minimums. So, for this training set, the mini batch training does not work really well.
- Also, in this data set, each image looks different as they are different digits. For example 0 has no similarity to 1. So mini-batch is not ideal, because different digits are tuning/pulling into different directions and steps. However, in other cases, if the images are very similar then it can speed up, it is better to use mini-batch training.
- **It says about our input data** that we do not have a perfect smooth error like parabola shape, but we have wiggle changing input data.