CENG 499 HW1 Report

Sinan Talha KOŞAR - e2099190

For this hyperparameter tuning part, I have configured with,

- 750 nodes
- 0.1 ratio
- 1000 batch size
- SGD

Important!: In task2.py, all layers runs consecutively once the code run.

For each k-layer, I have created k layer. My results are below in table.
Ps 1. Please note that, after having 50% at earliest epoch in any config, other epochs are truncated since we are focusing on the best on each layer
Ps 2. Please also note that activation functions are arranged in order.
Ps 3. The best configurations that are chosen are highlighted in the tables.

To decide best hyperparameters,

- Number of layers. I just keep adding layers until the test error does not improve anymore.
- Batch size: Started with 10 but after increasing it, our system gave more accurate results
- Number of nodes: Started with 50 and if I increased it too much like 5000 the system is really slowed down, 750 decided as optimal.

For 1-layer				
Activation Functions	***Learning Rate***	***Epoch***	***Train Loss***	***Percent Correct***
ReLU	0.1	12	1.3919	50.0%
ReLU	0.03	12	1.6187	45.0%
ReLU	0.01	12	1.8037	39.0%
ReLU	0.003	12	2.0115	33.0%
ReLU	0.001	12	2.1689	27.0%
GeLU	0.01	12	1.4525	48.0%
GeLU	0.03	12	1.6830	43.0%
SeLU	0.1	12	1.6514	42.0%
SeLU	0.03	12	1.7193	41.0%

Table 1: 1-layered trainings

For 2-layer				
Activation Functions	***Learning Rate***	***Epoch***	***Train Loss***	***Percent Correct***
GeLU + GeLU	0.1	13	1.5060	48%
GeLU + GeLU	0.03	13	1.8012	38%
GeLU + GeLU	0.01	13	2.0664	29.0%
GeLU + GeLU	0.003	13	2.2690	23.0%
GeLU + GeLU	0.001	13	2.3290	18.0%
ReLU + ReLU	0.1	13	1.4113	50.0%
ReLU + ReLU	0.03	13	1.6788	42.0%
ReLU + ReLU	0.01	13	1.9568	33.0%
ReLU + ReLU	0.003	13	2.220	27.0%
ReLU + ReLU	0.001	13	2.3158	19.0%
SeLU + SeLU	0.1	13	1.6211	44.0%
SeLU + SeLU	0.03	13	1.7025	41.0%
SeLU + SeLU	0.01	13	1.8006	39.0%
SeLU + SeLU	0.003	13	1.9267	36.0%
SeLU + SeLU	0.001	13	2.0808	30.0%

Table 2: 2-layered trainings

For 3-layer				
Activation Functions	**Learning Rate**	**Epoch**	**Train Loss**	**Percent Correct**
ReLU + ReLU + ReLU	0.1	11	1.5144	46.0%
ReLU + ReLU + ReLU	0.03	11	1.8821	35.0%
ReLU + ReLU + ReLU	0.01	11	2.2838	21.0%
ReLU + ReLU + ReLU	0.003	11	2.3447	15.0%
ReLU + ReLU + ReLU	0.001	11	2.3536	12.0%
GeLU + GeLU + GeLU	0.1	11	1.6947	41.0%
GeLU + GeLU + GeLU	0.03	11	2.0955	26.0%
GeLU + GeLU + GeLU	0.01	11	2.3351	17.0%
GeLU + GeLU + GeLU	0.003	11	2.3526	11.0%
GeLU + GeLU + GeLU	0.001	11	2.3567	9.0%
SeLU + SeLU + SeLU	0.1	11	1.6027	44.0%
SeLU + SeLU + SeLU	0.03	11	1.7259	41.0%
SeLU + SeLU + SeLU	0.01	11	1.8547	37.0%
SeLU + SeLU + SeLU	0.003	11	2.0060	31.0%
SeLU + SeLU + SeLU	0.001	11	2.1622	25.0%
ReLU + ReLu + GeLU	0.1	11	1.5466	46.0%
ReLU + ReLu + GeLU	0.03	11	1.9493	32.0%
ReLU + ReLu + GeLU	0.01	11	2.3118	20.0%
ReLU + ReLu + GeLU	0.003	11	2.3495	14.0%
ReLU + ReLu + GeLU	0.001	11	2.3557	10.0%
ReLU + ReLu + SeLU	0.1	11	1.4335	50.0%
ReLU + ReLu + SeLU	0.03	11	1.7380	40.0%
ReLU + ReLu + SeLU	0.01	11	2.0589	29.0%
ReLU + ReLu + SeLU	0.003	11	2.2945	21.0%
ReLU + ReLu + SeLU	0.001	11	2.3416	13.0%
ReLU + GeLu + GeLU	0.1	11	1.6051	45.0%

ReLU + GeLu + GeLU	0.03	11	2.0374	28.0%
ReLU + GeLu + GeLU	0.01	11	2.3310	19.0%
ReLU + GeLu + GeLU	0.003	11	2.3519	11.0%
ReLU + GeLu + GeLU	0.001	11	2.3564	10.0%
ReLU + GeLu + ReLU	0.1	11	1.5712	46.0%
ReLU + GeLu + ReLU	0.03	11	1.9586	32.0%
ReLU + GeLu + ReLU	0.01	11	2.3157	19.0%
ReLU + GeLu + ReLU	0.003	11	2.3471	15.0%
ReLU + GeLu + ReLU	0.001	11	2.3543	11.0%
ReLU + GeLu + SeLU	0.1	11	1.4657	49.0%
ReLU + GeLu + SeLU	0.03	11	1.7926	38.0%
ReLU + GeLu + SeLU	0.01	11	2.1148	26.0%
ReLU + GeLu + SeLU	0.003	11	2.3137	20.0%
ReLU + GeLu + SeLU	0.001	11	2.3471	11.0%
ReLU + SeLu + SeLU	0.1	11	1.3725	51.0% (*)
ReLU + SeLu + SeLU	0.03	11	1.6187	44.0%
ReLU + SeLu + SeLU	0.01	11	1.8788	36.0%
ReLU + SeLu + SeLU	0.003	11	2.1487	27.0%
ReLU + SeLu + SeLU	0.001	11	2.2805	21.0%

Table 3: 3-layered trainings

(*) There is also 50% at 11th epoch, but 51% chosen as it is more accurate.

For each k-layer network, the best performing hyperparameter configurations and its training and validation losses on the same graph are below

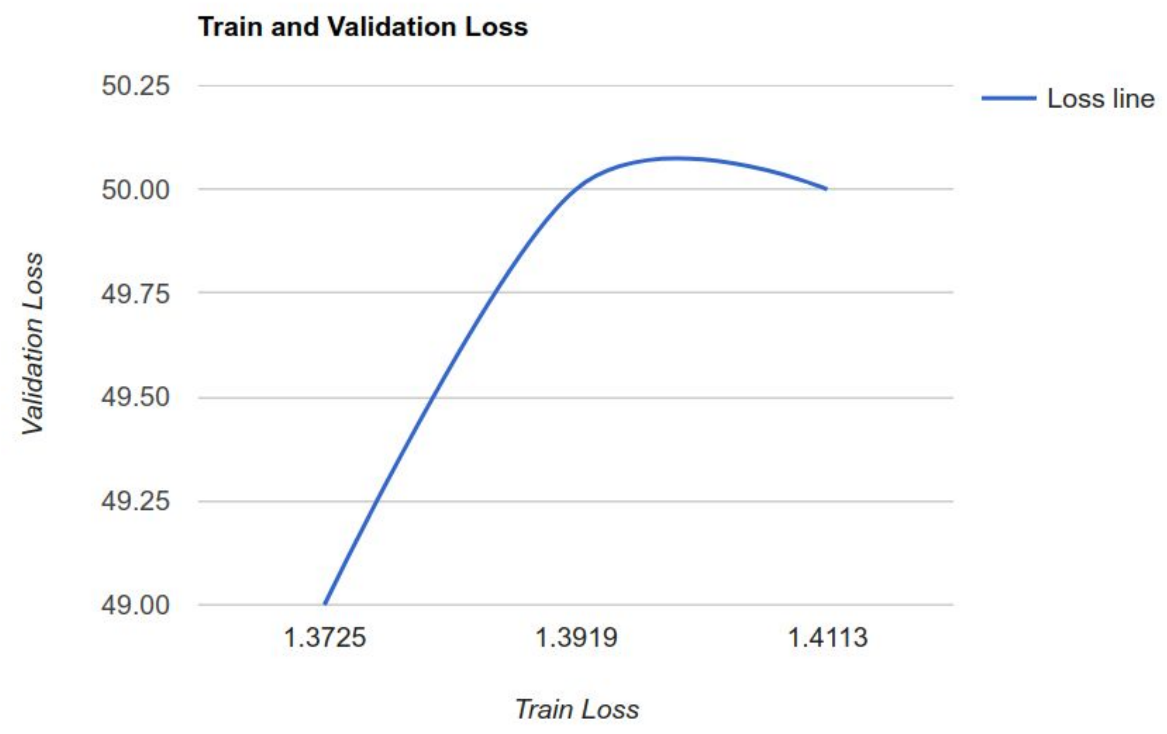

Figure 1: Training and Validation Loss

(See validation loss is opposite of percent correct) We cannot say that when best configurations selected and layer count increases there is an increase or decrease in validation loss but just for looking graph we can say that when layer count increased, the training accuracy increased, but we cannot make a general comment with only 1% difference. Also we can say that validation loss increase when configurations are more worse.

For 1-layer ReLU worked best with learning rate 0.1 and 0.03.
I can say that ReLU is more accurate.
For 2-layer ReLU + ReLU worked best.
I can say that ReLU is more accurate.
For 3-layer ReLU + SeLU + SeLU worked best.
I cannot make any comments since I haven't tried all combinations.

What countermeasure did you take against overfitting?

To avoid overfitting, we can do;

1. collect more data
2. use ensembling methods that "average" models
3. choose simpler models or penalize complexity

Generally (comment from table), the fewer parameters that I have to tuned the better.

So, how may one understand when a network starts to overfit during training?

We can understand overfitting better by looking at the opposite problem, underfitting.Underfitting occurs when a model is too simple – informed by too few features or regularized too much – which makes it inflexible in learning from the dataset.

What method did you use to decide where to end the training procedure?

In homework pdf, it was stated that, the best configuration should achieve at least 50% accuracy in the test set. So when on any epoch reach 50%, I have stopped the process and decrease the epoch count for the rest of trainings by doing that I am able to catch the accuracies on same epoch count and make comment on how any configurations are more accurate compared to others.

Whether accuracy is a suitable metric to measure the performance of a network for this specific dataset.

To decide whether the accuracy is a suitable metric to measure the performance of a network for this specific dataset, just look at the tables. On every record, when learning rate decreases, the train loss and percent correct decreased. We can say that learning rate is proportional with train loss, but not with percent correct. Because on every epoch increase there is decrease in train loss but percent correct values are irrelevant. On every epochs sometimes percent correct increased sometimes not. I have tried 65 configurations but since I have not tried all of activation functions, I cannot say that there is any relation between chosen activation function and accuracy. But I can say that Relu > Gelu > Selu by looking Relu + Relu, Gelu + Gelu, Selu + Selu for 2-layer, and Relu > Selu > Gelu by looking Relu + Relu + Relu, Relu + Relu + Gelu, Relu + Relu + Selu from 3-layer (each > denotes that left hand side is more accurate than right hand side).