Report

Emil Shikhaliyev

26.04.2021

1 Sanity Checks

Since we know that our dataset is balanced, we can do the following checks.

1.1 Loss

Initially we should expect the loss as $-log(0.1) = 2.30$, because we have 10 output classes, that why we have 0.1 probability to find the true class. Since I got 1.90, which is close to theoretical value, which we expected. So, we can say the initial weights were initialized better than the expected value, 2.30.

1.2 Accuracy

Since we have 10 classes as a result, we should expect the accuracy as $1/10 = 0.1$. But, in my model after random initialization I got 0.2876, which is a reason for saying that, initial weights are initialized better than we expected initially.

2 Separating Validation Set

I separated validation set from train set via the use of randomsplit function from torch.util.data library. I used 5000 of photos as validation set and 25000 as train set.

3 Hyperparameter optimization

3.1 1-layer (0-hidden-layer) network

There's 0 hidden layer in my network.
Hyperparameters, which I used:
Learning rates: 0.01 0.003, 0.001, 0.0003, 0.0001, 0.00003
Layer Sizes: 256, 512, 1024

	Learning Rate					
	0.01	0.003	0.001	0.0003	0.0001	0.00003
	0.2600	0.3228	0.3618	0.3693	0.3674	0.3096

Table 1: 1-layer network accuracy rates

3.2 2-layer (1-hidden-layer) network

There's 1 hidden-layer in my network architecture. Hyperparameters:
Activation functions: Relu, sigmoid, tanh
Layer Sizes: 256, 512, 1024,
Learning Rate: 0.01, 0.003, 0.001, 0.0003, 0.0001, 0.00003)

Layer	Learning Rate					
Activations	0.01	0.003	0.001	0.0003	0.0001	0.00003
S, 256	0.099	0.0944	0.4736	0.4942	0.4060	0.3572
S, 512	0.0996	0.1102	0.4776	0.5246	0.3934	0.3666
S, 1024	0.0996	0.1012	0.4296	0.5338	0.4266	0.3706
T, 256	0.1102	0.1026	0.1726	0.5216	0.4676	0.3822
T, 512	0.0972	0.0944	0.2610	0.5194	0.4726	0.3888
T, 1024	0.1102	0.0944	0.2818	0.5344	0.4868	0.3964
R, 256	0.1024	0.3142	0.4682	0.4582	0.3948	0.3995
R, 512	0.0996	0.3794	0.4852	0.4992	0.4758	0.3966
R, 1024	0.0944	0.3262	0.5222	0.5085	0.5020	0.4048

Table 2: 2-layer network accuracy rates

3.3 3-layer (2-hidden-layer) network

There's 1 hidden-layer in my network architecture. Hyperparameters:
Activation functions: Relu, sigmoid, tanh
Layer Sizes: 256, 512, 1024
Learning Rate: 0.1, 0.003, 0.001, 0.0001, 0.00003

AF, Layer	Learning Rate				
Count	0.1	0.03	0.001	0.0001	0.00003
S, 256	0.1102	0.1012	0.3762	0.3850	0.3418
S, 512	0.1102	0.0968	0.3586	0.3746	0.3468
S, 1024	0.0992	0.0968	0.3152	0.5144	0.3510
T, 256	0.0972	0.0992	0.1786	0.4966	0.3642
T, 512	0.0996	0.1102	0.1778	0.4650	0.3948
T, 1024	0.1012	0.1970	0.2184	0.4868	0.3970
R, 256	0.1026	0.3278	0.5438	0.4944	0.4238
R, 512	0.0944	0.3998	0.4910	0.5448	0.4612
R, 1024	0.1026	0.1970	0.5438	0.5528	0.5218

Table 3: 3-layer network accuracy rate

4 The best hyperparameter

4.1 Results

Draw its training and validation loss graph (both loss should appear in the same graph so that we can compare). I got the best validation loss and best accuracy rate with corresponding hyperparameters:
Activation function = 'relu'
Learning rate = 0.0001
Hidden layers = 2
Layer size = 1024
Using these parameters, I got the test accuracy as 0.103457. You can see graph the validation and training loss graph below. I decided the count of epochs as 30. Because I think it should be enough for getting sufficient results.

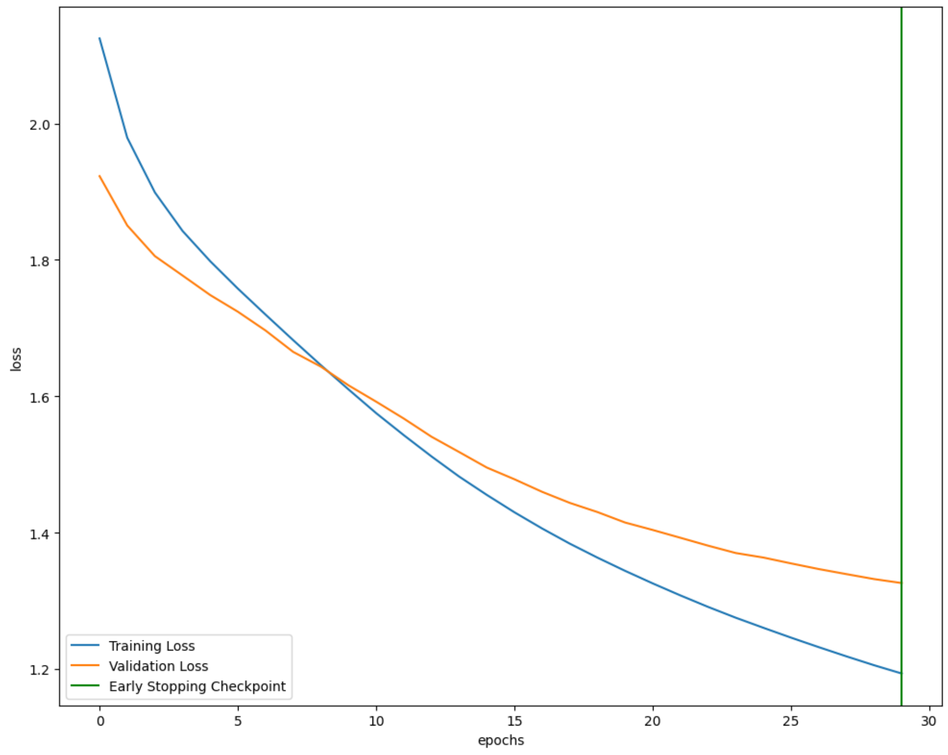

Figure 1: Training and Validation loss graph in the best case

4.2 Overfitting countermeasures

Because of overfitting case I implemented the algorithm which is given by the our assistant, Artun, in our homework pdf, which is about save the minimum value of validation loss and return it as loss in every epochs.

5 Comments