CS 763/CS 764

COMPUTER VISION

REFLECTION ESSAY
LAB 07-Part B

Submitted by

Group 05

193079001, 193079005, 193079015

Contents

List of Tables

1 Training Details

1.1 Dataloader:

We have created custom datasetloader class, which reads the input data_train.pkl file from given base directory, this was done using pickle class, and have made a dictionary of target data consisting of 2D points of 2nd view and the rotation and translation vectors and focal length to get to the same. This along with input data (2D points of 1st view) are then passed to a Dataloader class from torch.utils.data to make a batched dataset for training, along with a random sampler of data.

1.2 Model:

The model uses 2 residual blocks sequentially. Each residual block contains two sequential layers of linear layer for feature extraction, batch normalization layer, ReLU activation layer, and dropout layer randomly dropping neurons from output of ReLU layer. Also to take the input to our model input, a linear layer is used for input size of 15x2, and output size of 1024. To get the output from our model, a linear layer is used with input size as 1024 and output size as 15x3.

1.3 Training script:

Initialization of optimizer and scheduler for learning rate was done. In the training loop, for each batch of data, we predict the 3D points (inputsize = batchsize*15*2) for the given corresponding 2D points (outputsize = batchsize*15*3) for view 1. These 3D world coordinate points are projected in the 2nd camera coordinate system, by rotating and translating the points by given rotation and translation input data. Next we project them in the image coordinate system using the focal length given for 2nd camera. These 2D coordinates of view 2 that we obtained from the predicted 3D world coordinates of pose is then compared to the actual 2D coordinates of view 2, a loss is computed between them, and is back propagated for the model weights to learn.

2 Q. Loss Function used:

The loss function used is mean square error between the predicted 2D points of view2 and the real 2D points of view2 . The average over all the points and all the data over a batch is provided as loss which is then propagated through the model. This is indicated by the train loss and validation loss given in Table 1. As a evaluation metric, mean square error is used between the 3D point predicted and given 3D point of world coordinate system. This is indicated by the test loss given in Table 1.

3 Experiments and Results

The experimentation's were conducted on batch size (64 and 128), dropout rate (0.5, 0.2, and 0.9), learning rate initial (10^{-3}, 10^{-2}), and different schedulers (ExponentialLR and stepLR).

Initially over model was overfitting on the train data, as our validation loss was high compared to our training loss, so we started implementing more regularization on our model, we tried increasing dropout rate for the same.

We have run the following experiments for 20 epochs.

Batch Size	Dropout	Learning Rate	Scheduler	Decay or Step size	Train Loss	Validation loss	Test loss
64	0.5	10^{-3}	ExponentialLR()	decay = 0.9	0.66	0.38	2.305
128	0.5	10^{-3}	ExponentialLR()	decay = 0.9	2.55	3.9	2.075
128	0.5	10^{-3}	ExponentialLR()	decay = 0.5	2.3	0.912	1.157
128	0.9	10^{-3}	ExponentialLR()	decay = 0.9	9.15	0.698	1.153
128	0.2	10^{-3}	ExponentialLR()	decay = 0.9	0.782	0.471	1.149
128	0.5	10^{-2}	ExponentialLR()	decay = 0.9	9.15	0.698	1.148
128	0.2	10^{-2}	StepLR()	step size = 15	0.281	0.44	1.154
128	**0.2**	10^{-3}	**StepLR()**	**stepsize = 20**	**0.375**	**0.258**	**1.147**

Table 1: Experiments and Results

4 Conclusion

The best results were found on using constant learning rate of 10^{-3}, with a dropout of 0.2 and batchsize of 128 on 50 epochs . This was the best learning rate we found for our model learning. We also saw bigger batch gave better results, because it helps generalizing more (reduce over fitting).

The best result as said above were, **train_loss = 0.035, validation_loss = 0.177, test_loss = 1.147**.

5 References

[1] Referred to Task 7A