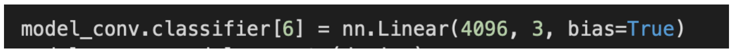
COMP 448/548 – Medical Image Analysis Homework #3

Explain how you made use of the pretrained AlexNet to design your ***own*** *classifier:*

1. ***How did you make the input size compatble with the AlextNet network?***

   I changed last layer of neural network as indicated below. Since there is 3 classes, the output of that layer should be 3.

   ```
   model_conv.classifier[6] = nn.Linear(4096, 3, bias=True)
   ```

2. ***How did you normalize the input?***

   Since the model is pretrained, mean and standard deviation is obtained from the website https://pytorch.org/hub/pytorch_vision_alexnet/.

   I created a data_transforms function which transforms the train test and validation data. If the function is called by parameter is_normalized = true, than the function normalizes the image_dataset (train, validation and test) else function transforms the image dataset without normalization. Afterwards, I passed it to data_transforms function and in compose function I called normalize method by passing mean and standard deviation of AlexNet.

3. W***hat parts of the AlextNet architecture did you modify? How did you modify the last layer?***

   I selected 6th layer, I modified that layer as Linear(4096, 3). I chose the loss function as CrossEntropy(). To say more, I selected optimizer as torch.optim.SGD(model_conv.parameters(), lr= LEARNING_RATE)

   LEARNING_RATE = 0.1 is set as below in the beginning. I set epoch number as 4.

4. ***What loss function did you use in backpropagation?***

   I used l CrossEntropy() as loss function since it is used in AlexNet.

5. ***How did you select the parameters related to backpropagation? For example, did you use any optimizer? If so, what were the parameters of this optimizer and how did you select their values?***

   model_conv.parameters(), learning_rate and momentum was parameters of optimizer.

   I used optimizer. I chose momentum as 0.9 since it gives the best accuracy score and set learning_rate as 0.1. There is no significant change by the decrease of learning rate.

6. ***How did you address the class-imbalance problem?***

   I balanced the images by oversampling. I oversampled every training class to have the size 88 since the biggest size is 88 by class 2. After that I spare 20% of samples in classes for validation.

Note 1: *I added model.eval() line to cancel drop out layer in the evaluation phase, however the accuracy results still change for every run.*

Note 2 : *In order to run the code, insert the attached database to your home path of drive and access that drive with the /content/drive/MyDrive/COMP448-HW3 colab directory. If your colab path differs somehow, adjust path_dataset accordingly.*

	Training portion of the training set				Validation portion of training set				Test set			
	1	2	3	Overall	1	2	3	Overall	1	2	3	Overall
With input normalization and with addressing the class-imbalance problem	.99	.98	.98	.98	.98	.94	1.0	.97	.93	.94	.76	.89
With input normalization and without addressing the class-imbalance problem	.98	.96	.92	.96	.91	.79	1.0	.86	.93	.91	.84	.90
Without input normalization and with addressing the class-imbalance problem	.97	.66	.94	.83	1.0	.82	.94	1.0	.93	.91	.84	.90

References

- https://www.geeksforgeeks.org/adjusting-learning-rate-of-a-neural-network-in-pytorch/
- https://medium.com/analytics-vidhya/pytorch-directly-use-pre-trained-alexnet-for-image-classification-and-visualization-of-the-dea0de3eade9
- https://towardsdatascience.com/pytorch-tabular-multiclass-classification-9f8211a123ab
- https://analyticsindiamag.com/implementing-alexnet-using-pytorch-as-a-transfer-learning-model-in-multi-class-classification/
- https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html
- https://medium.com/walmartglobaltech/tackling-multi-class-image-classification-problem-with-transfer-learning-using-pytorch-50150b215fb6
- https://towardsdatascience.com/debugging-neural-networks-with-pytorch-and-w-b-2e20667baf72