Mete Uz

Homework

1. **How did you make the input size compatible with AlexNet network?**

Our dataset already consists of 256x256 images, which is the standard input size for the AlexNet so I did not make any changes to input size to make the model work.

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

I have used transforms.Normalize(mean, std) to normalize the data. I have calculated the mean and standard deviation for all sets individually. By using the calculated means and stds, the mentioned function calculates the normalization operation in which the mean is subtracted from the data and then it is divided by the std.

3. **What parts of the AlexNet architecture did you modify? How did you modify the last layer?**

```
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace=True)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=4096, out_features=1024, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=1024, out_features=3, bias=True)
  )
)
```

Figure 1: Summary of the model

Above figure shows the architecture after I have made the changes. I have updated the out_features of the final classifier (6) to 3 because in this classification problem we have 3 classes. I have also decreased the number of nodes in the dense layer because the original AlexNet is used for 1000 class classification task but we only have 3.

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

I have used CrossEntropy loss because this is a classification task.

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

I have used SGD as my optimizer, because it performs well in classification tasks. I have taken initial learning rate as 0.01 (it is a good choice to start with a learning rate between 0.1 to 0.01 in a classification task like this). I have chosen the momentum as 0.9 again for similar reasons. In order to explore larger spaces in the beginning and making it smaller until convergence, I have added a learning rate scheduler which multiplies the learning rate by 0.1 every 30 iteration. I have trained the model for 100 iterations because it converges after that point.

6. **What loss function did you use in backpropogation?**

I tried to weight scaling for class imbalance problem but my predictions did not change with it.

	Training portion of the training set				Validation portion of the training set				Test set			
	Class 1	Class 2	Class 3	Overall	Class 1	Class 2	Class 3	Overall	Class 1	Class 2	Class 3	Overall
With input normalization and with addressing the class-imbalance problem	100%	100%	100%	100%	100%	100%	66.6%	92.2%	100%	84.6%	90.9%	91.1%
With input normalization and without addressing the class-imbalance problem	100%	100%	100%	100%	100%	100%	66.6%	92.2%	100%	84.6%	90.9%	91.1%
Without input normalization and with addressing the class-imbalance problem	73.9%	77%	94.1%	80%	61.5%	81%	100%	79%	78.7%	73%	87%	78.1%

Figure 2: Results