11310CS460200 Intro 2 ML Lab 5

Name	Date
110020007 施淙綸	2024/11/14

Explain why ReLU is typically preferred over Sigmoid as the activation function in the convolutional block?

- First of all, ReLU is definitely easier for calculation. Also, because ReLU doesn't provide nagetive value which can prevent vanishing gradient problem.

Describe how you design the CNN architecture and your findings in choosing parameters such as filter_size and pool_size for each layer?

- At first, I observed the loss and accuracy of training dataset, if the training loss (accuracy) do not decrease (increase), I would try deeper structure of layer stacking. Though sometimes there's too many parameters to train, the model failed to predict image label and always output the same value. In such case, I reduced the layers.
- Using the approach above, I finally discover the best architecture for my hyperparameters and use it to fine-tune the final model.
- When I used the larger `filter_size` in `Conv` layers, the model always present a bad performance in loss and accuracy. While in contrast, the larger `pool_size` in `MaxPool` layers provide slightly higher performance in my case.
- This is my observation in my case, I'm not very sure about the relationship of `filter_size`, `pool_size` and the performance of model in theoretical inference.

Calculate and compare the number of learnable parameters between the CNN model and the NN model you designed for binary classification in Lab4. For simplicity, omit the bias parameters and calculate only the weights.

- In Lab 4 and 5, only `Dense` and `Conv` layers have learnable parameters, `Activation`, `MaxPool` and `Flatten` have no learnable parameters.

```
# Lab 4, using Lab 5 layer representation
Dense(784, 16)
Dense(16, 32)
Dense(32, 1)
```

- Each node in fully connected layer from A nodes to B nodes will provide $N_L = A * B$ learnable parameters. So in Lab 4, there're $N_{L,4} = 784 * 16 + 16 * 32 + 32 * 1 = 13088$ learnable parameters.

```
# Lab 5
Conv(filter_size=3, input_channel=1, output_channel=4, pad=1, stride=1)
Conv(filter_size=3, input_channel=4, output_channel=8, pad=1, stride=1)
Conv(filter_size=3, input_channel=8, output_channel=8, pad=1, stride=1)
Dense(72, 1)
```

- Each `Conv` layer has $N_L = input_channel * filter_size * filter_size * output_channel$ when omit the bias parameters. So in Lab 5, there're $N_{L,5} = (1 * 3 * 3 * 4 + 4 * 3 * 3 * 8 + 8 * 3 * 3 * 8) + (72 * 1) = (36 + 288 + 576) + 72 =$ learnable parameters.