CS 8803 DL Assigment 3

Jingdao Chen

March 14

Part 1: Building your own CNN

The Convolutional Neural Network architecture is implemented as follows, with the input as a 224x224x3 image and the output as 20 classes.

```
nn.Sequential {
[input -> (1) -> (2) -> output]
(1): nn.Sequential {
[input -> (1) -> (2) -> (3) -> (4) -> (5) -> (6) -> (7) -> (8) -> (9) -> (10) -> output]
(1): nn.SpatialConvolutionMM(3 -> 16, 5x5)
(2): nn.ReLU
(3): nn.SpatialMaxPooling(4,4,4,4)
(4): nn.SpatialConvolutionMM(16 -> 32, 7x7)
(5): nn.ReLU
(6): nn.SpatialMaxPooling(4,4,4,4)
(7): nn.Reshape(4608)
(8): nn.Linear(4608 -> 100)
(9): nn.ReLU
(10): nn.Linear(100 -> 20)
```

```
}
(2): nn.Sequential {
[input -> (1) -> output]
(1): nn.LogSoftMax
}
}
```

Each section below presents the training and testing accuracies depending on the modifications made:

ReLU

Tanh

Dropout

Data augmentation

Discussion

As shown in the training and testing accuracies, ReLU is more effective than Tanh since it achieves better performance in the training and testing stage. Dropout is also an effective form of regularization since it improved the testing accuracy. Data augmentation however did not improve the training accuracy but did increase the testing accuracy slightly.

Filter Visualization

The weights in the first two convolutional layers are also visualized below:

Part 2: Fine-tuning

In this section, a pre-trained network is used with the last fully connected layer replaced to fit the number of desired classes. During the training stage, all the layers are frozen except the last layer. The training and testing accuracies are shown below:

Part 3: Class model Visualization

To generate a class model image, the backpropogation technique is used where gradients are propogated down from the top layer given a target class. An input image is initialized to zero and incremented with the input gradient of the first convolution layer. The generated image for class 131 is shown below:

assignment3/results/131.png