EX5 REPORT

About the model

After understanding what spectrograms are - a visual representation of sound signals, I have chosen to try out CNN based models.

I have started with the following simple CNN model:

- Convolution layer `in_channels=1, out_channels=12, kernel_size=5, stride=1, padding=1`
- Convolution layer `in_channels=12, out_channels=12, kernel_size=5, stride=1, padding=1`
- Max pool layer `kernel_size=2, stride=2`
- Convolution layer `in_channels=12, out_channels=24, kernel_size=5, stride=1, padding=1`
- Convolution layer `in_channels=24, out_channels=24, kernel_size=5, stride=1, padding=1`
- fully connected layer of input 2400 and output 30
- All convolutional layers followed by batch normalization and a relu activation function

For that model, the best result I achieved were validation accuracy of 82% with hyperparameters of 0.001 learning rate, 10 epochs, ADAM optimizer, batch size of 100.
I tried adding another fully connected layer and a dropout layer, which lead to improved validation accuracy of 85%. Then I tried adding another fully connected layer but have experienced degradation in validation accuracy of 83%.

Then I have investigated some other CNN models, i.e., ReSnet and Google Net which seemed a bit complex, but also VGG which seemed relatively simple and quite like my starting point model. I have tried out the VGG-13 architecture. Because of the increasing number of convolutions, run time took twice longer but I have received 92% of validation accuracy. Hyperparameters remains like what I have mentioned above except this time 12 epochs.

VGG-13:

- Convolution layer `in_channels=1, out_channels=64, kernel_size=3, stride=1, padding=1`
- Convolution layer `in_channels=64, out_channels=64, kernel_size=3, stride=1, padding=1`
- Max pool layer `kernel_size=2, stride=2`

- Convolution layer `in_channels=64, out_channels=128, kernel_size=3, stride=1, padding=1`
- Convolution layer `in_channels=128, out_channels=128, kernel_size=3, stride=1, padding=1`
- Max pool layer `kernel_size=2, stride=2`

- Convolution layer `in_channels=128, out_channels=256, kernel_size=3, stride=1, padding=1`
- Convolution layer `in_channels=256, out_channels=256, kernel_size=3, stride=1, padding=1`
- Max pool layer `kernel_size=2, stride=2`

- Convolution layer `in_channels=256, out_channels=512, kernel_size=3, stride=1, padding=1`
- Convolution layer `in_channels=512, out_channels=512, kernel_size=3, stride=1, padding=1`
- Max pool layer `kernel_size=2, stride=2`

- Convolution layer `in_channels=512, out_channels=512, kernel_size=3, stride=1, padding=1`
- Convolution layer `in_channels=512, out_channels=512, kernel_size=3, stride=1, padding=1`
- Max pool layer `kernel_size=2, stride=2`

- Adaptive average pool layer of output size 7X7

- fully connected layer of input 512 * 7 * 7 and output 4096

- fully connected layer of input 4096 and output 4096

- fully connected layer of input 4096 and output 30

- All convolutional layers followed by batch normalization and a relu activation function
- fully connected layers 1 and 2 followed by relu activation function and a dropout layer

How to run the model

1. Prerequisites:

   - On the same directory you should have:
     - My submitted files: ex5.py, gcommand_dataset.py
     - The data in a folder named gcommands

   - gcommands should be with the following content:
     - test
       - file named as you wish
         - 6836.wav
         - …
         - …
     - train
       - bed
       - bird
       - …
       - …
     - valid
       - bed
       - bird
       - …
       - …

2. After this, just run: `python3 ex5.py`