CNN on RGBD images

Kaiwen Zhou

Due 12/14

1 Method

1.1 Input

I tried several kinds of inputs

- All img0
- All img1
- All img2
- depth: 3 channel 1 images
- img0+img1: 6 channels or 3 channels but double samples.
- img0+depth: 4 channels
- img1+depth: 4 channels

1.2 Models

1.2.1 Model 1:

```
Conv(
    conv2D(input_size, conv_feature, kernel_size=5)
    relu()
    maxpool2d(kernel_size=2)
    conv2d(conv_feature, conv_feature, kernel_size=5)
    relu()
    maxpool2d(kernel_size=2)
)

FC(
    linear_feature(conv_feature*53*53, fc_feature),
    relu(),
    linear_feature(fc_feature, output_size),
)
```

1.2.2 Model 2:

```
Conv(
    conv2D(input_size, conv_feature, kernel_size=5)
    relu()
    maxpool2d(kernel_size=2)
    conv2d(conv_feature, conv_feature, kernel_size=5)
    relu()
    maxpool2d(kernel_size=2)
    conv2d(conv_feature, conv_feature, kernel_size=6)
    relu()
    maxpool2d(kernel_size=2)
)

FC(
    linear_feature(conv_feature*24*24, fc_feature),
    relu(),
    linear_feature(conv_feature*12*12, conv_feature*6*6),
    relu(),
    linear_feature(conv_feature*6*6, fc_feature),
    relu(),
    linear_feature(fc_feature, output_size),
)
```

1.2.3 Model 3:

```
Conv(
    conv2d(input_size, conv_feature, kernel_size=5)
    relu()
    maxpool2d(kernel_size=2)
    conv2d(conv_feature, conv_feature, kernel_size=5)
    relu()
    maxpool2d(kernel_size=2)
    conv2d(conv_feature, 16, kernel_size=6)
    relu()
    maxpool2d(kernel_size=2)
    conv2d(16, 24, kernel_size=5)
    relu()
    maxpool2d(kernel_size=2)
)

FC(
    linear_feature(10*10*24, fc_feature),
    relu(),
    linear_feature(fc_feature, output_size),
)
```

1.2.4 Model 4:

```
Conv(
    conv2d(input_size, conv_feature, kernel_size=5)
    relu()
    maxpool2d(kernel_size=2)
    conv2d(conv_feature, conv_feature, kernel_size=5)
    relu()
    maxpool2d(kernel_size=2)
    conv2d(conv_feature, 16, kernel_size=6)
    relu()
    maxpool2d(kernel_size=2)
)

FC(
    linear_feature(16*24*24, fc_feature),
    relu(),

    linear_feature(fc_feature, output_size),
)
```

1.2.5 Similarities and Differences

In the 4 models I constructed

- Model 1 is the standard model in HW5 but with the last log_softmax() function removed, since here we are not working on a classification problem.
- Model 2 adds an extra convolutional layer and 2 FCN layers. Also log_softmax() function is removed.Therefore, this one is more balanced.
- Model 3 adds 2 extra convolutional layer. Also log_softmax() function is removed. Therefore, this one is more convolutional focused.
- Model 4 adds 1 extra convolutional layer. Also log_softmax() function is removed. Therefore, this a simplified version of Model 3.

1.3 Tuning Parameters

There are several parameters that are involved in tuning:

- Means and variance for the normalization on rgb images
- batch size for the train data loader
- number of neurons in the middle layer of FCN portion of the whole network.
- learning rate and momentum in SGD.
- number of convolutional features

2 Experimental Results

lr = 0.007, conv_feature = 6, fc_features = 2000, input = img0, batch_size = 64

momentum/ batch size	0.5/64	0.9/64	1.0/64	0.5/2	0.9/2	1.0/2
Model 1	0.011	0.008	0.022	0.007	0.005	0.016
Model 2	0.016	0.022	0.017	0.016	0.015	0.019
Model 3	0.021	0.015	0.024	0.011	0.009	0.014
Model 4	0.011	0.013	0.012	0.008	0.0059	0.007

lr = 0.007, input = img0, batch_size = 2, momentum = 0.9,

conv_feature/ fc_features	6/50	15/50	6/2000	15/2000
Model 1	0.009	0.007	0.008	0.0059
Model 2	0.013	0.011	0.012	0.008
Model 3	0.011	0.007	0.007	0.005
Model 4	0.008	0.006	0.007	0.004

lr = 0.007, batch_size = 64, momentum = 0.9, conv_feature = 6 fc_features = 2000

input	img0	img2	depth	img0+depth
Model 1	0.009	0.016	0.011	0.013
Model 2	0.020	0.024	0.022	0.019
Model 3	0.007	0.0 12	0.017	0.022
Model 4	0.008	0.010	0.014	0.016

3 Discussion

- Making the model's FCN portion intensive (more layers in FCN potion) does not do any good. I think this has to do with the focus of the FCNs, since they tend to find a solution where the "big" mistakes are excluded but bear some mind error. And that's why the output of Model 2 is very stable but not as great as the other 3.
- Smaller batch size is definitely better. I think this has to with how SGD works. Since we know SGD is not only foucusing on finding the minimum. Some times it will sort of explore other places to find the potential local minimum, and a lower batch size kind of gives it more chances to explore the global minimum.
- Similar to the reason above, the larger the momentum the higher chance SGD will find a lower minimum. And a smaller learning rate makes the process finer, but you have to always consider the computation time of the whole network. From my experience, the optimal learning rate is around $[0.01, 0.05]$, and the optimal momentum is around $[0.85, 0.95]$.
- More input data does not give you better result. Especially, I found that img3s and their corresponding depth images are not consistent. And that will just introduce more noise to the whole system. From my experience using all img0 works the best, I think this has to do with the normalizing means and variances I chose. I didn't normalize them one by one, instead I used a standard normalizing factor provided by ImageNet, and probably img0 just works the best given that.

- From my experience, the more convolutional features, the better. I think this is because the CNN can therefore extract and maintain more characteristic, or critical values embedded in the given images. However, like always, you have to take computation complexity into consideration.
- Always normalize the input images. This avoids the situation where the computation of gradient might explode and other tiny but annoying problems.

4 Future Work

I haven't tried Augmentation in this project. I think in the future I will first try augmentation and then design more CNN networks to see which one works the best so that I can have a clearer intuition about it.