# Yutong Dai report

September 26, 2018

## 1 Model Configuration

**CNN Architecture:** The same as described in the lecture notes.

- convolution(channel=64, filter_size=4, stride=1, padding=2) -> Relu -> Batch normalization
- convolution(channel=64, filter_size=4, stride=1, padding=2) -> Relu -> Maxpooling(stride =2. kernel_size = 2)
- Dropout(probability=0.25)
- convolution(channel=64, filter_size=4, stride=1, padding=2) -> Relu -> Batch normalization
- convolution(channel=64, filter_size=4, stride=1, padding=2) -> Maxpooling(stride =2. kernel_size = 2)
- Dropout(probability=0.25)
- convolution(channel=64, filter_size=4, stride=1, padding=2) -> Relu -> Batch normalization
- convolution(channel=64, filter_size=3, stride=1, padding=0) -> Relu
- Dropout(probability=0.25)
- convolution(channel=64, filter_size=3, stride=1, padding=0) -> Relu -> Batch normalization
- convolution(channel=64, filter_size=3, stride=1, padding=0) -> Relu -> Batch normalization
- Dropout
- Fully connected layer(hidden_units=500)
- Fully connected layer(hidden_units=500)
- Softmax

**Batch Size:**100
**Optimizer:** Adam
**Data Augmentation:**

- With 0.5 probability employ the data augmentation.
  - Randomly flip 60% of mini-batch training samples either horizontally or vertically.
  - Randomly rotate 60% of mini-batch training samples 90 degrees.

## 2 Experiment Results

**Note my code supports `resume` functionality**, which means you can training your cnn based previous stored results.

First, I trained my model on 30 epochs, which has 80% test accuracy. In order to get higher test accuracy, I resumed training 3 times, each with additional 5 epochs. So finally:

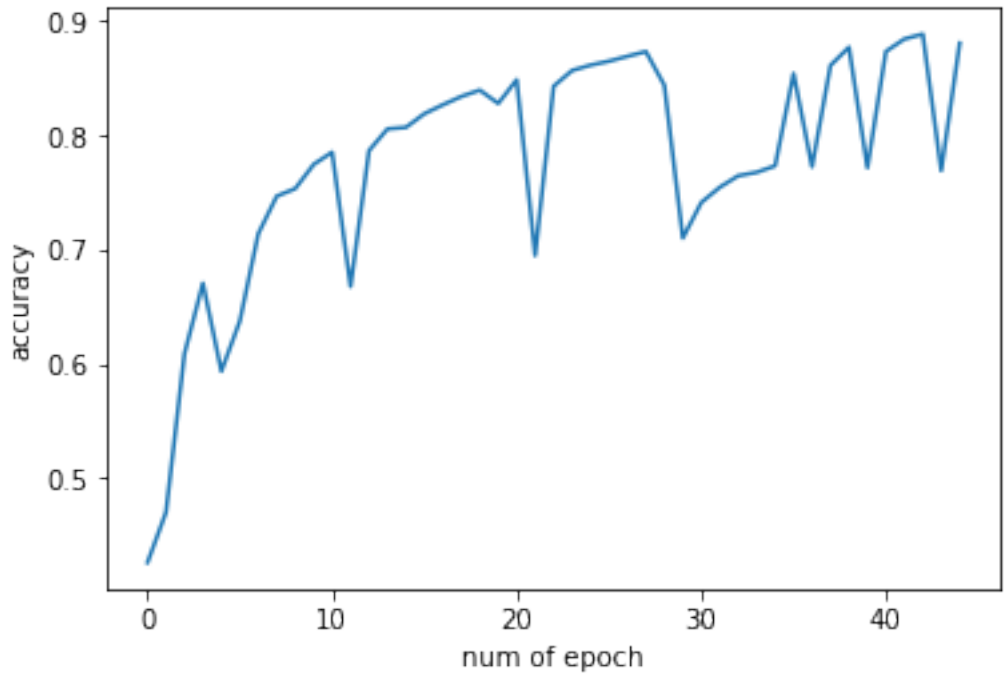
**For 45 epochs, I obtain 85.43% training accuracy.** The training log is given below.

```
training...
=> no checkpoint found at './checkpoint.pth.tar'
2018-09-26 00:40:33,767 - [INFO] - torch version: 0.3.0
2018-09-26 00:40:33,784 - [INFO] - loading data...
2018-09-26 00:41:38,176 - [INFO] - Epoch: 1 | Loss: 1.260013461112976 | Accuracy::0.42578
2018-09-26 00:41:38,310 - [INFO] - Best parameters is updated!
2018-09-26 00:42:28,968 - [INFO] - Epoch: 2 | Loss: 1.296244502067566 | Accuracy::0.4698
2018-09-26 00:42:29,051 - [INFO] - Best parameters is updated!
2018-09-26 00:43:18,118 - [INFO] - Epoch: 3 | Loss: 0.8326022624969482 | Accuracy::0.60864
2018-09-26 00:43:18,222 - [INFO] - Best parameters is updated!
2018-09-26 00:44:07,194 - [INFO] - Epoch: 4 | Loss: 1.0159904956817627 | Accuracy::0.670539999
2018-09-26 00:44:07,294 - [INFO] - Best parameters is updated!
2018-09-26 00:44:57,909 - [INFO] - Epoch: 5 | Loss: 1.0425361394882202 | Accuracy::0.59338
2018-09-26 00:45:48,699 - [INFO] - Epoch: 6 | Loss: 1.0083378553390503 | Accuracy::0.63738
2018-09-26 00:46:37,809 - [INFO] - Epoch: 7 | Loss: 0.8012216687202454 | Accuracy::0.7139
2018-09-26 00:46:37,913 - [INFO] - Best parameters is updated!
2018-09-26 00:47:26,905 - [INFO] - Epoch: 8 | Loss: 0.7867785692214966 | Accuracy::0.746819999
2018-09-26 00:47:27,008 - [INFO] - Best parameters is updated!
2018-09-26 00:48:17,627 - [INFO] - Epoch: 9 | Loss: 0.9155967831611633 | Accuracy::0.7534
2018-09-26 00:48:17,734 - [INFO] - Best parameters is updated!
2018-09-26 00:49:08,430 - [INFO] - Epoch: 10 | Loss: 0.5476292967796326 | Accuracy::0.774620000
2018-09-26 00:49:08,553 - [INFO] - Best parameters is updated!
2018-09-26 00:49:59,315 - [INFO] - Epoch: 11 | Loss: 0.7429620623588562 | Accuracy::0.78534
2018-09-26 00:49:59,403 - [INFO] - Best parameters is updated!
2018-09-26 00:50:50,160 - [INFO] - Epoch: 12 | Loss: 0.7952980995178223 | Accuracy::0.66768
2018-09-26 00:51:40,980 - [INFO] - Epoch: 13 | Loss: 0.6046001911163333 | Accuracy::0.78676
2018-09-26 00:51:41,146 - [INFO] - Best parameters is updated!
2018-09-26 00:52:30,233 - [INFO] - Epoch: 14 | Loss: 0.5061242580413818 | Accuracy::0.80578
2018-09-26 00:52:30,343 - [INFO] - Best parameters is updated!
2018-09-26 00:53:21,051 - [INFO] - Epoch: 15 | Loss: 0.6744173169136047 | Accuracy::0.80708
2018-09-26 00:53:21,605 - [INFO] - Best parameters is updated!
2018-09-26 00:54:10,660 - [INFO] - Epoch: 16 | Loss: 0.5296404957771301 | Accuracy::0.81908
2018-09-26 00:54:10,769 - [INFO] - Best parameters is updated!
2018-09-26 00:54:59,785 - [INFO] - Epoch: 17 | Loss: 0.5702775716781616 | Accuracy::0.82688
2018-09-26 00:54:59,877 - [INFO] - Best parameters is updated!
2018-09-26 00:55:48,908 - [INFO] - Epoch: 18 | Loss: 0.5297756791114807 | Accuracy::0.834159999
2018-09-26 00:55:49,022 - [INFO] - Best parameters is updated!
2018-09-26 00:56:38,040 - [INFO] - Epoch: 19 | Loss: 0.44035109877586365 | Accuracy::0.83974
2018-09-26 00:56:38,155 - [INFO] - Best parameters is updated!
2018-09-26 00:57:28,850 - [INFO] - Epoch: 20 | Loss: 0.5621123313903809 | Accuracy::0.828059999
2018-09-26 00:58:17,992 - [INFO] - Epoch: 21 | Loss: 0.6654871106147766 | Accuracy::0.84884
2018-09-26 00:58:18,106 - [INFO] - Best parameters is updated!
2018-09-26 00:59:08,746 - [INFO] - Epoch: 22 | Loss: 0.7074993848800659 | Accuracy::0.6948
2018-09-26 00:59:57,873 - [INFO] - Epoch: 23 | Loss: 0.4395642876625061 | Accuracy::0.843119999
2018-09-26 01:00:46,992 - [INFO] - Epoch: 24 | Loss: 0.3320717513561249 | Accuracy::0.85696
2018-09-26 01:00:47,094 - [INFO] - Best parameters is updated!
2018-09-26 01:01:36,155 - [INFO] - Epoch: 25 | Loss: 0.4970552325248718 | Accuracy::0.86164
2018-09-26 01:01:36,274 - [INFO] - Best parameters is updated!
```

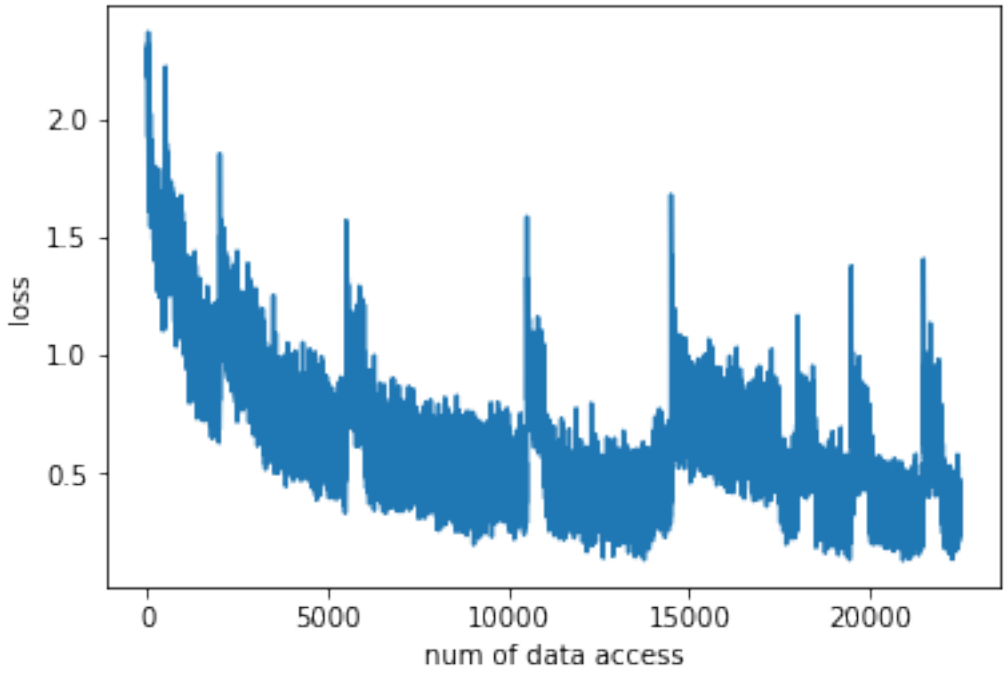
```
2018-09-26 01:02:25,312 - [INFO] - Epoch: 26 | Loss: 0.48153769969940186 | Accuracy::0.86512
2018-09-26 01:02:25,416 - [INFO] - Best parameters is updated!
2018-09-26 01:03:14,437 - [INFO] - Epoch: 27 | Loss: 0.392272412776947 | Accuracy::0.869379999
2018-09-26 01:03:14,559 - [INFO] - Best parameters is updated!
2018-09-26 01:04:03,589 - [INFO] - Epoch: 28 | Loss: 0.3902410566806793 | Accuracy::0.87354
2018-09-26 01:04:03,711 - [INFO] - Best parameters is updated!
2018-09-26 01:04:54,393 - [INFO] - Epoch: 29 | Loss: 0.46228259801864624 | Accuracy::0.8439800
2018-09-26 01:05:45,239 - [INFO] - Epoch: 30 | Loss: 0.7465217709541321 | Accuracy::0.710119999
2018-09-26 01:05:48,100 - [INFO] - trained on [30] epoch, with test accuracy [0.800000000000000
=> checkpoint found at './checkpoint.pth.tar'
2018-09-26 01:07:43,765 - [INFO] - torch version: 0.3.0
2018-09-26 01:07:43,765 - [INFO] - loading data...
2018-09-26 01:08:48,078 - [INFO] - Epoch: 31 | Loss: 0.8729916214942932 | Accuracy::0.74126
2018-09-26 01:09:38,858 - [INFO] - Epoch: 32 | Loss: 0.7604179978370667 | Accuracy::0.75458
2018-09-26 01:10:29,618 - [INFO] - Epoch: 33 | Loss: 0.6580194234848022 | Accuracy::0.764679999
2018-09-26 01:11:20,380 - [INFO] - Epoch: 34 | Loss: 0.5148293972015381 | Accuracy::0.7677
2018-09-26 01:12:11,116 - [INFO] - Epoch: 35 | Loss: 0.7237148880958557 | Accuracy::0.77336
2018-09-26 01:12:14,044 - [INFO] - trained on [35] epoch, with test accuracy [0.8082]
=> checkpoint found at './checkpoint.pth.tar'
2018-09-26 01:12:48,311 - [INFO] - torch version: 0.3.0
2018-09-26 01:12:48,315 - [INFO] - loading data...
2018-09-26 01:13:51,021 - [INFO] - Epoch: 36 | Loss: 0.5348299145698547 | Accuracy::0.854279999
2018-09-26 01:14:41,852 - [INFO] - Epoch: 37 | Loss: 0.5350968837738037 | Accuracy::0.772579999
2018-09-26 01:15:31,053 - [INFO] - Epoch: 38 | Loss: 0.4586135149002075 | Accuracy::0.861420000
2018-09-26 01:16:20,174 - [INFO] - Epoch: 39 | Loss: 0.3627488613128662 | Accuracy::0.877339999
2018-09-26 01:16:20,292 - [INFO] - Best parameters is updated!
2018-09-26 01:17:10,958 - [INFO] - Epoch: 40 | Loss: 0.5626611113548279 | Accuracy::0.77176
2018-09-26 01:17:13,840 - [INFO] - trained on [40] epoch, with test accuracy [0.823699999999999
=> checkpoint found at './checkpoint.pth.tar'
2018-09-26 01:18:08,354 - [INFO] - torch version: 0.3.0
2018-09-26 01:18:08,358 - [INFO] - loading data...
2018-09-26 01:19:11,541 - [INFO] - Epoch: 41 | Loss: 0.199357807636261 | Accuracy::0.873579999
2018-09-26 01:20:00,656 - [INFO] - Epoch: 42 | Loss: 0.33019065856933594 | Accuracy::0.88454
2018-09-26 01:20:00,777 - [INFO] - Best parameters is updated!
2018-09-26 01:20:49,802 - [INFO] - Epoch: 43 | Loss: 0.35046303272247314 | Accuracy::0.8888400
2018-09-26 01:20:49,938 - [INFO] - Best parameters is updated!
2018-09-26 01:21:40,676 - [INFO] - Epoch: 44 | Loss: 0.5082605481147766 | Accuracy::0.7692
2018-09-26 01:22:29,865 - [INFO] - Epoch: 45 | Loss: 0.2712952792644501 | Accuracy::0.880940000
2018-09-26 01:22:32,733 - [INFO] - trained on [45] epoch, with test accuracy [0.8543]
```

```
In [1]: import torch
        import matplotlib.pyplot as plt
        resume = '../checkpoint.pth.tar'
        checkpoint = torch.load(resume, map_location='cpu')
        start_epoch = checkpoint['epoch']
        best_train_acc = checkpoint['best_train_acc']
        train_loss = checkpoint['train_loss']
        train_accuracy_epoch = checkpoint['train_accuracy_epoch']
```

In [2]:
```
plt.plot(train_accuracy_epoch)
plt.xlabel("num of epoch")
plt.ylabel("accuracy")
plt.show()
```




In [3]:
```
plt.plot(train_loss)
plt.xlabel("num of data access")
plt.ylabel("loss")
plt.show()
```




Here `1 num of data access` means that 100 figures are used to train the model.

### 2.1 Some notes on experiments

1. Why we have sudden decrease in training accuracy? Because in my training setting, with 0.5 probability I apply the data augmentation in the current epoch. And for each batch of data, only 60% of the pictures will be randomly flipped either horizontally or vertically. The drop may occur when the data augmentation takes place and produce some new patterns that are unseen to the current cnn network or it is possible the batch size is small. But test accuracy is still increasing.

2. I previously made a mistake when writing the dropout. Indeed we need to use `nn.Droput2d` instead of `nn.functional.droput2d`. If you use the latter, you need specifically set the `training=True`. Otherwise, it won't take effect! That's the reason why I saw 97% training accuracy in my previous implementation. Obviously, it is over-fitted. But, I can still get around 83% test accuracy.

### 2.2 Extra credit problem

Following code snippet is the key part of the monte carlo method for estimating testing accuracy. For details, please see the attached code `mc_test_acc.py`.

```
for i in range(0, len(y_test), batch_size):
    x_test_batch = torch.FloatTensor(x_test[i:i+batch_size, :])
    y_test_batch = torch.LongTensor(y_test[i:i+batch_size])
    if use_cuda:
```

```
        data, target = Variable(x_test_batch).cuda(), Variable(y_test_batch).cuda()
    else:
        data, target = Variable(x_test_batch), Variable(y_test_batch)
    output = model.forward(data)
    mc_output = output
    for i in range(99):
        output = model.forward(data)
        mc_output += output
    mc_output = mc_output / 100
    prediction = mc_output.data.max(1)[1]
    accuracy = (float(prediction.eq(target.data).sum()) / float(batch_size))
    test_accuracy.append(accuracy)
accuracy_test = np.mean(test_accuracy)
```

I get 100 replicates of the output of the softmax function and average them to get the predicted value. Finally, with 100 i.i.d relization of the mask $R$, my monte carlo test accuracy is **83.78%**, which is smaller than the heuristic test accuracy **85.43%**.