# MSBD 5012 HA3 report

Student Name: Zuo Yifan
Student ID: 20876522
Assignment #: Assignment HA3
Student Email: yzuoah@connect.ust.hk
Course Name: MSBD5012

---

- Baseline:

| Loss | Accuracy |
|---|---|
| <pre>[1,  2000] loss: 2.153<br>[1,  4000] loss: 1.814<br>[1,  6000] loss: 1.642<br>[1,  8000] loss: 1.552<br>[1, 10000] loss: 1.516<br>[1, 12000] loss: 1.440<br>[2,  2000] loss: 1.371<br>[2,  4000] loss: 1.333<br>[2,  6000] loss: 1.346<br>[2,  8000] loss: 1.295<br>[2, 10000] loss: 1.265<br>[2, 12000] loss: 1.281<br>Finished Training</pre> | `Accuracy of the network on the 10000 test images: 54 %` |

- Alter number of hidden layer experiment:
    - Add two more hidden layer:

      We add nn.Linear(84, 60) and nn.Linear(60, 30) change fc3 to nn.Linear(30, 10)

| Loss | Accuracy |
|---|---|
| <pre>[1,  2000] loss: 2.305<br>[1,  4000] loss: 2.281<br>[1,  6000] loss: 2.034<br>[1,  8000] loss: 1.932<br>[1, 10000] loss: 1.849<br>[1, 12000] loss: 1.739<br>[2,  2000] loss: 1.665<br>[2,  4000] loss: 1.625<br>[2,  6000] loss: 1.576<br>[2,  8000] loss: 1.523<br>[2, 10000] loss: 1.468<br>[2, 12000] loss: 1.469</pre> | `Accuracy of the network on the 10000 test images: 46 %` |

`Compare to the baseline result, we can see that the loss decreases slower and the accuracy decreases which is due to the difficulty of gradients to propagate back to the lower layers since we add two more hidden layers.`

  - Drop one hidden layer:

    We drop fc2.

| Loss | Accuracy |
|---|---|
| <pre>[1,  2000] loss: 2.074<br>[1,  4000] loss: 1.742<br>[1,  6000] loss: 1.589<br>[1,  8000] loss: 1.502<br>[1, 10000] loss: 1.452<br>[1, 12000] loss: 1.417<br>[2,  2000] loss: 1.347<br>[2,  4000] loss: 1.325<br>[2,  6000] loss: 1.299<br>[2,  8000] loss: 1.268<br>[2, 10000] loss: 1.264<br>[2, 12000] loss: 1.262</pre> | `Accuracy of the network on the 10000 test images: 56 %` |

    `Compare to the baseline result, we can see that the loss decreases slightly faster and the accuracy increases, because the less hidden layer we have, the easier the gradients can propagate back to the lower layers.`

- Alter number of filters experiment:
  - Increase number of filters:

    We increase conv1 filters from 6 to 15 and conv2 filters from 16 to 30.

| Loss | Accuracy |
|---|---|
| <pre>[1,  2000] loss: 2.188<br>[1,  4000] loss: 1.815<br>[1,  6000] loss: 1.612<br>[1,  8000] loss: 1.500<br>[1, 10000] loss: 1.433<br>[1, 12000] loss: 1.367<br>[2,  2000] loss: 1.289<br>[2,  4000] loss: 1.259<br>[2,  6000] loss: 1.219<br>[2,  8000] loss: 1.191<br>[2, 10000] loss: 1.183<br>[2, 12000] loss: 1.139</pre> | `Accuracy of the network on the 10000 test images: 59 %` |

    `Compare to the baseline result, we can see that the loss decrease slightly faster and the accuracy increases, because we increase the number of filters on convolutional layers increases the number of units in fc1 from 16*5*5 to 30*5*5, since the number of filters increases helps the modules learn more about different features from images which helps classify images.`

  - Decrease number of filters:

    We increase conv1 filters from 6 to 4 and conv2 filters from 16 to 6.

| Loss | Accuracy |
|---|---|
| `[1,  2000] loss: 2.218` | `Accuracy of the network on the 10000 test images: 48 %` |
| `[1,  4000] loss: 1.940` | |
| `[1,  6000] loss: 1.777` | |
| `[1,  8000] loss: 1.660` | |
| `[1, 10000] loss: 1.600` | |
| `[1, 12000] loss: 1.568` | |
| `[2,  2000] loss: 1.521` | |
| `[2,  4000] loss: 1.497` | |
| `[2,  6000] loss: 1.499` | |
| `[2,  8000] loss: 1.477` | |
| `[2, 10000] loss: 1.474` | |
| `[2, 12000] loss: 1.474` | |

`Compare to the baseline result, we can see that the loss decrease slower and the accuracy decreases, because decreasing the number of filters on convolutional layers decreases the number of units in fc1 from 16*5*5 to 6*5*5, since the number of filters decreases, module get less features about images and higher loss decreases the accuracy.`

- Alter learning rate experiment:
  - Decrease learning rate:

    Learning rate from 0.001 to 0.0001

| Loss | Accuracy |
|---|---|
| `[1,  2000] loss: 2.304` | `Accuracy of the network on the 10000 test images: 36 %` |
| `[1,  4000] loss: 2.300` | |
| `[1,  6000] loss: 2.294` | |
| `[1,  8000] loss: 2.274` | |
| `[1, 10000] loss: 2.208` | |
| `[1, 12000] loss: 2.129` | |
| `[2,  2000] loss: 2.039` | |
| `[2,  4000] loss: 1.977` | |
| `[2,  6000] loss: 1.926` | |
| `[2,  8000] loss: 1.883` | |
| `[2, 10000] loss: 1.829` | |
| `[2, 12000] loss: 1.789` | |

`Compare to the baseline result, we can see that the loss decrease slower and the accuracy decreases a lot, becrease in gradient descent process, decreasing learning rate means shrinking the step moved in the opposite direction of the derivative which leads to slow learning and low accuracy comparing with same amount of learning.`

  - Increase learning rate:

    Learning rate from 0.001 to 0.01

| Loss | Accuracy |
|---|---|
| [1,  2000] loss: 2.099 | Accuracy of the network on the 10000 test images: 23 % |
| [1,  4000] loss: 1.965 | |
| [1,  6000] loss: 1.963 | |
| [1,  8000] loss: 1.960 | |
| [1, 10000] loss: 1.948 | |
| [1, 12000] loss: 1.986 | |
| [2,  2000] loss: 1.965 | |
| [2,  4000] loss: 1.996 | |
| [2,  6000] loss: 2.023 | |
| [2,  8000] loss: 1.996 | |
| [2, 10000] loss: 2.097 | |
| [2, 12000] loss: 2.049 | |

```
Compare to the baseline result, we can see that the loss nearly remain the same over
mini-batches and the accuracy decreases a lot, becrease in gradient descent process,
we rising learning rate too much which leads to the step moved in the opposite
direction of the derivative too much to miss the minima which leads to slow learning
and low accuracy comparing with same amount of learning.
```

- Alter optimization method experiment:
  - AdaGrad:

| Loss | Accuracy |
|---|---|
| [1,  2000] loss: 2.124 | Accuracy of the network on the 10000 test images: 35 % |
| [1,  4000] loss: 1.954 | |
| [1,  6000] loss: 1.917 | |
| [1,  8000] loss: 1.883 | |
| [1, 10000] loss: 1.866 | |
| [1, 12000] loss: 1.848 | |
| [2,  2000] loss: 1.839 | |
| [2,  4000] loss: 1.803 | |
| [2,  6000] loss: 1.820 | |
| [2,  8000] loss: 1.810 | |
| [2, 10000] loss: 1.804 | |
| [2, 12000] loss: 1.774 | |

```
Compare to the baseline result, we can see that the loss decreases a little faster
then nearly remain the smae and the accuracy decreases a lot. This is because at the
beginning stage the learning is fast and historical gradients is large, AdaGrad make
the learning rate become small, then the learning is really slow which leads to low
accuracy comparing with smae amount of learning.
```

  - Adam:

| Loss | Accuracy |
|---|---|
| [1,  2000] loss: 1.863 | Accuracy of the network on the 10000 test images: 55 % |
| [1,  4000] loss: 1.616 | |
| [1,  6000] loss: 1.506 | |
| [1,  8000] loss: 1.457 | |
| [1, 10000] loss: 1.425 | |
| [1, 12000] loss: 1.381 | |
| [2,  2000] loss: 1.319 | |
| [2,  4000] loss: 1.314 | |
| [2,  6000] loss: 1.289 | |
| [2,  8000] loss: 1.266 | |
| [2, 10000] loss: 1.272 | |
| [2, 12000] loss: 1.228 | |

`Compare to the baseline result, we can see that the loss decreases dramatically faster than baseline and the accuracy increase a little since Adam combines RMSProp and Momentum which is the best optimizer over others, Adam dramatically increase learning speed.`

- RMSProp:

| Loss | Accuracy |
|---|---|
| [1,  2000] loss: 1.847 | Accuracy of the network on the 10000 test images: 53 % |
| [1,  4000] loss: 1.627 | |
| [1,  6000] loss: 1.529 | |
| [1,  8000] loss: 1.482 | |
| [1, 10000] loss: 1.434 | |
| [1, 12000] loss: 1.390 | |
| [2,  2000] loss: 1.319 | |
| [2,  4000] loss: 1.310 | |
| [2,  6000] loss: 1.332 | |
| [2,  8000] loss: 1.313 | |
| [2, 10000] loss: 1.289 | |
| [2, 12000] loss: 1.297 | |

`Compare to the baseline result, we can see that the loss decreases dramatically faster then baseline and the accuracy is nearly the same. Since RMSProp use the same idea of AdaGrad except it uses an exponentially decaying average to discard history gradients from the extreme past. This optimizer perform better than Adagrad but no better than Adam.`

- Use batch normalization experiment:

| Loss | Accuracy |
|---|---|
| `[1,  2000] loss: 1.925` | |
| `[1,  4000] loss: 1.640` | |
| `[1,  6000] loss: 1.526` | |
| `[1,  8000] loss: 1.477` | |
| `[1, 10000] loss: 1.444` | |
| `[1, 12000] loss: 1.407` | `Accuracy of the network on the 10000 test images: 57 %` |
| `[2,  2000] loss: 1.324` | |
| `[2,  4000] loss: 1.287` | |
| `[2,  6000] loss: 1.271` | |
| `[2,  8000] loss: 1.258` | |
| `[2, 10000] loss: 1.253` | |
| `[2, 12000] loss: 1.223` | |

Compare to the baseline result, we can see that the loss decreases dramatically faster then baseline and the accuracy increases. This is because batch normalization makes different layers more independent and avoids covariate shift which accelerate training.

## Best Performance

- By dropping some hidden layers and increasing number of filters with Adam optimization and batch normalization, the best performance I can get is:

| Loss | Accuracy |
|---|---|
| `[1,  2000] loss: 1.732` | |
| `[1,  4000] loss: 1.483` | |
| `[1,  6000] loss: 1.374` | |
| `[1,  8000] loss: 1.310` | |
| `[1, 10000] loss: 1.242` | |
| `[1, 12000] loss: 1.211` | `Accuracy of the network on the 10000 test images: 63 %` |
| `[2,  2000] loss: 1.107` | |
| `[2,  4000] loss: 1.099` | |
| `[2,  6000] loss: 1.067` | |
| `[2,  8000] loss: 1.069` | |
| `[2, 10000] loss: 1.035` | |
| `[2, 12000] loss: 1.021` | |