Results and Analysis

Jin Hyun Park 633001823

- Final result

Model / Epochs	25 or 30	90
Standard Model	**30 epochs: 74.36%**	**79.05%**
Bottleneck Model	**25 epochs: 71.26%**	**80.65%**

- Please see the next page for more details

1. **Running Report & Result and Analysis**

A. `DataReader.py`

i. `def load_data`

1. There is a total of 6 batches: 5 for training and 1 for testing. I downloaded the CIFAR-10 dataset from the given link and converted them to (`x_train`, `y_train`) and (`x_test`, `y_test`).

B. `ImageUtils.py`

i. `def preprocess_image`

1. Added four extra pixels on each side by using `np.pad`
2. Randomly cropped a (32, 32) section of the image using `np.random.randint`
3. Randomly flip the image by using `np.random.randint` and `np.flip`
4. Normalize the image for each channel by using `np.mean` and `np.std`

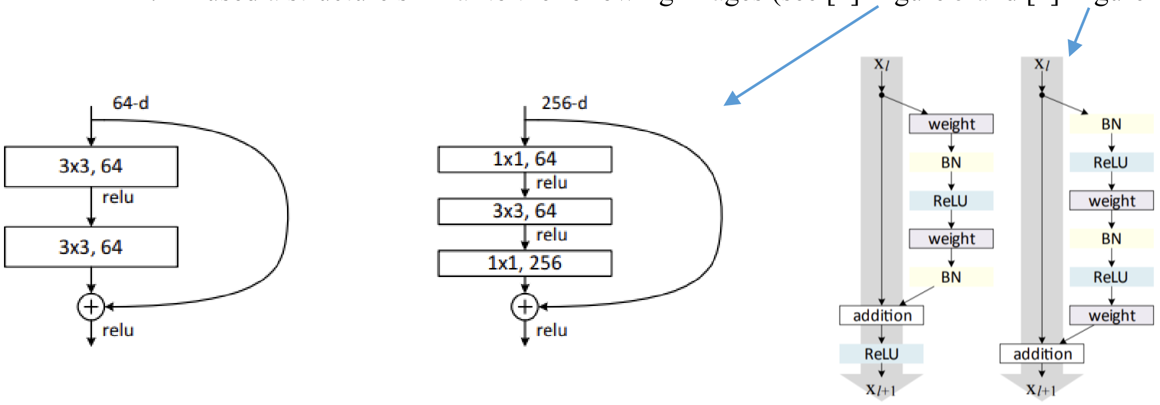
C. `Network.py`

i. `class batch_norm_relu_layer`

1. `nn.BatchNorm2d()` followed by `nn.ReLU()`

ii. `class standard_block` and `def bottleneck_block`

1. I used a structure similar to the following images (see [1]-Figure 5 and [2]-Figure 1)



(a) original (b) proposed

iii. `class stack_layer`

1. `projection_shortcut` is defined in this method/function
2. Created a stack of `standard_blocks` and `bottleneck_blocks` where

`projection_shortcut` is only applied in the first block.

3. Stack `standard_block` or `bottleneck_block` by using `resnet_size`

iv. `class output_layer`

1. Applied pooling, fully-connected layer, and Softmax

D. `Model.py`

i. `def train`

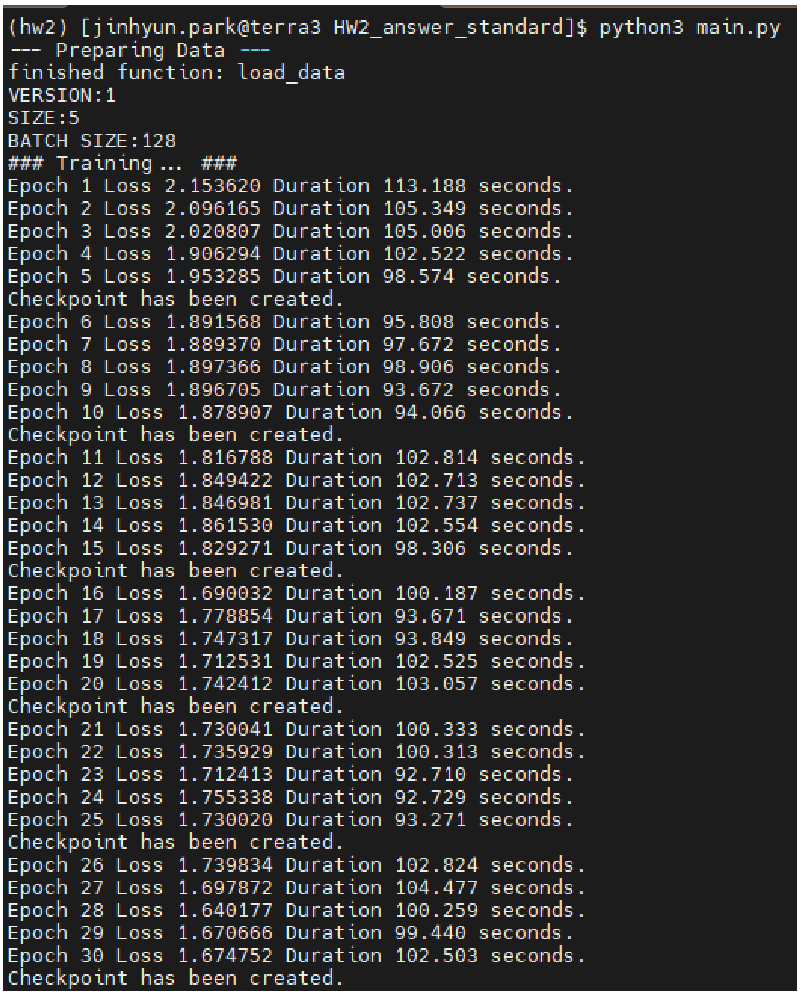
1. Split data into `batch_size` to train models.
2. Set an appropriate learning rate and scheduling algorithm.

ii. `def test_or_validate`

1. Used this function to validate and test the model (standard or bottleneck)

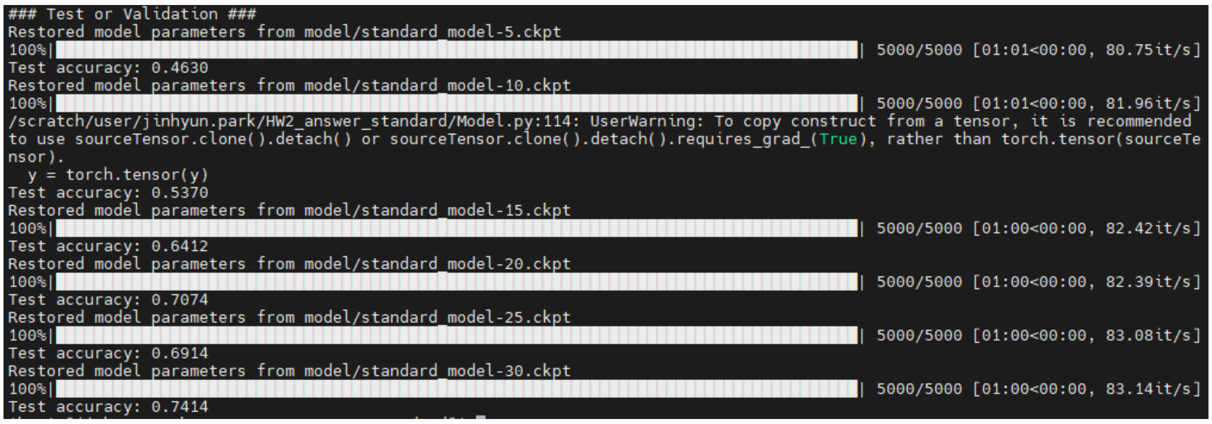
E. Standard Model: Screenshots of training loss, validation/test accuracy (network size = 5).

i. Training on `x_train_new` and `y_train_new`



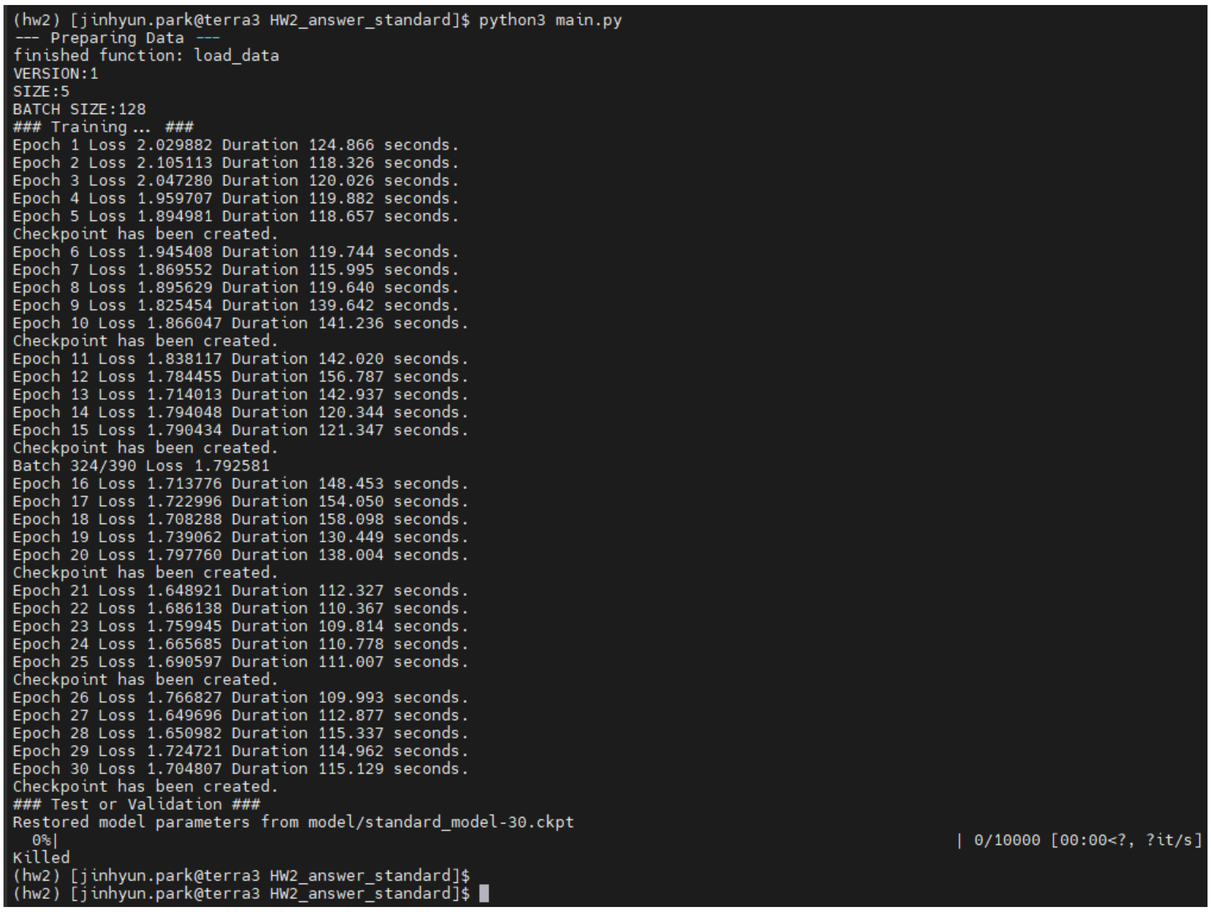
```
(hw2) [jinhyun.park@terra3 HW2_answer_standard]$ python3 main.py
--- Preparing Data ---
finished function: load_data
VERSION:1
SIZE:5
BATCH SIZE:128
### Training... ###
Epoch 1 Loss 2.153620 Duration 113.188 seconds.
Epoch 2 Loss 2.096165 Duration 105.349 seconds.
Epoch 3 Loss 2.020807 Duration 105.006 seconds.
Epoch 4 Loss 1.906294 Duration 102.522 seconds.
Epoch 5 Loss 1.953285 Duration 98.574 seconds.
Checkpoint has been created.
Epoch 6 Loss 1.891568 Duration 95.808 seconds.
Epoch 7 Loss 1.889370 Duration 97.672 seconds.
Epoch 8 Loss 1.897366 Duration 98.906 seconds.
Epoch 9 Loss 1.896705 Duration 93.672 seconds.
Epoch 10 Loss 1.878907 Duration 94.066 seconds.
Checkpoint has been created.
Epoch 11 Loss 1.816788 Duration 102.814 seconds.
Epoch 12 Loss 1.849422 Duration 102.713 seconds.
Epoch 13 Loss 1.846981 Duration 102.737 seconds.
Epoch 14 Loss 1.861530 Duration 102.554 seconds.
Epoch 15 Loss 1.829271 Duration 98.306 seconds.
Checkpoint has been created.
Epoch 16 Loss 1.690032 Duration 100.187 seconds.
Epoch 17 Loss 1.778854 Duration 93.671 seconds.
Epoch 18 Loss 1.747317 Duration 93.849 seconds.
Epoch 19 Loss 1.712531 Duration 102.525 seconds.
Epoch 20 Loss 1.742412 Duration 103.057 seconds.
Checkpoint has been created.
Epoch 21 Loss 1.730041 Duration 100.333 seconds.
Epoch 22 Loss 1.735929 Duration 100.313 seconds.
Epoch 23 Loss 1.712413 Duration 92.710 seconds.
Epoch 24 Loss 1.755338 Duration 92.729 seconds.
Epoch 25 Loss 1.730020 Duration 93.271 seconds.
Checkpoint has been created.
Epoch 26 Loss 1.739834 Duration 102.824 seconds.
Epoch 27 Loss 1.697872 Duration 104.477 seconds.
Epoch 28 Loss 1.640177 Duration 100.259 seconds.
Epoch 29 Loss 1.670666 Duration 99.440 seconds.
Epoch 30 Loss 1.674752 Duration 102.503 seconds.
Checkpoint has been created.
```

ii. Validation on `x_valid and y_valid`



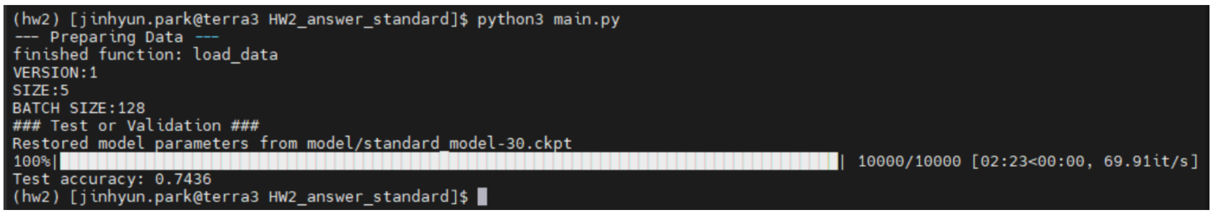
```
### Test or Validation ###
Restored model parameters from model/standard_model-5.ckpt
100%|████████████████████████████████████████████████████████████| 5000/5000 [01:01<00:00, 80.75it/s]
Test accuracy: 0.4630
Restored model parameters from model/standard_model-10.ckpt
100%|████████████████████████████████████████████████████████████| 5000/5000 [01:01<00:00, 81.96it/s]
/scratch/user/jinhyun.park/HW2_answer_standard/Model.py:114: UserWarning: To copy construct from a tensor, it is recommended
to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(True), rather than torch.tensor(sourceTe
nsor).
  y = torch.tensor(y)
Test accuracy: 0.5370
Restored model parameters from model/standard_model-15.ckpt
100%|████████████████████████████████████████████████████████████| 5000/5000 [01:00<00:00, 82.42it/s]
Test accuracy: 0.6412
Restored model parameters from model/standard_model-20.ckpt
100%|████████████████████████████████████████████████████████████| 5000/5000 [01:00<00:00, 82.39it/s]
Test accuracy: 0.7074
Restored model parameters from model/standard_model-25.ckpt
100%|████████████████████████████████████████████████████████████| 5000/5000 [01:00<00:00, 83.08it/s]
Test accuracy: 0.6914
Restored model parameters from model/standard_model-30.ckpt
100%|████████████████████████████████████████████████████████████| 5000/5000 [01:00<00:00, 83.14it/s]
Test accuracy: 0.7414
```

iii. Training on `x_train` and `y_train`



```
(hw2) [jinhyun.park@terra3 HW2_answer_standard]$ python3 main.py
--- Preparing Data ---
finished function: load_data
VERSION:1
SIZE:5
BATCH SIZE:128
### Training... ###
Epoch 1 Loss 2.029882 Duration 124.866 seconds.
Epoch 2 Loss 2.105113 Duration 118.326 seconds.
Epoch 3 Loss 2.047280 Duration 120.026 seconds.
Epoch 4 Loss 1.959707 Duration 119.882 seconds.
Epoch 5 Loss 1.894981 Duration 118.657 seconds.
Checkpoint has been created.
Epoch 6 Loss 1.945408 Duration 119.744 seconds.
Epoch 7 Loss 1.869552 Duration 115.995 seconds.
Epoch 8 Loss 1.895629 Duration 119.640 seconds.
Epoch 9 Loss 1.825454 Duration 139.642 seconds.
Epoch 10 Loss 1.866047 Duration 141.236 seconds.
Checkpoint has been created.
Epoch 11 Loss 1.838117 Duration 142.020 seconds.
Epoch 12 Loss 1.784455 Duration 156.787 seconds.
Epoch 13 Loss 1.714013 Duration 142.937 seconds.
Epoch 14 Loss 1.794048 Duration 120.344 seconds.
Epoch 15 Loss 1.790434 Duration 121.347 seconds.
Checkpoint has been created.
Batch 324/390 Loss 1.792581
Epoch 16 Loss 1.713776 Duration 148.453 seconds.
Epoch 17 Loss 1.722996 Duration 154.050 seconds.
Epoch 18 Loss 1.708288 Duration 158.098 seconds.
Epoch 19 Loss 1.739062 Duration 130.449 seconds.
Epoch 20 Loss 1.797760 Duration 138.004 seconds.
Checkpoint has been created.
Epoch 21 Loss 1.648921 Duration 112.327 seconds.
Epoch 22 Loss 1.686138 Duration 110.367 seconds.
Epoch 23 Loss 1.759945 Duration 109.814 seconds.
Epoch 24 Loss 1.665685 Duration 110.778 seconds.
Epoch 25 Loss 1.690597 Duration 111.007 seconds.
Checkpoint has been created.
Epoch 26 Loss 1.766827 Duration 109.993 seconds.
Epoch 27 Loss 1.649696 Duration 112.877 seconds.
Epoch 28 Loss 1.650982 Duration 115.337 seconds.
Epoch 29 Loss 1.724721 Duration 114.962 seconds.
Epoch 30 Loss 1.704807 Duration 115.129 seconds.
Checkpoint has been created.
### Test or Validation ###
Restored model parameters from model/standard_model-30.ckpt
  0%|                                                                      | 0/10000 [00:00<?, ?it/s]
Killed
(hw2) [jinhyun.park@terra3 HW2_answer_standard]$
(hw2) [jinhyun.park@terra3 HW2_answer_standard]$
```

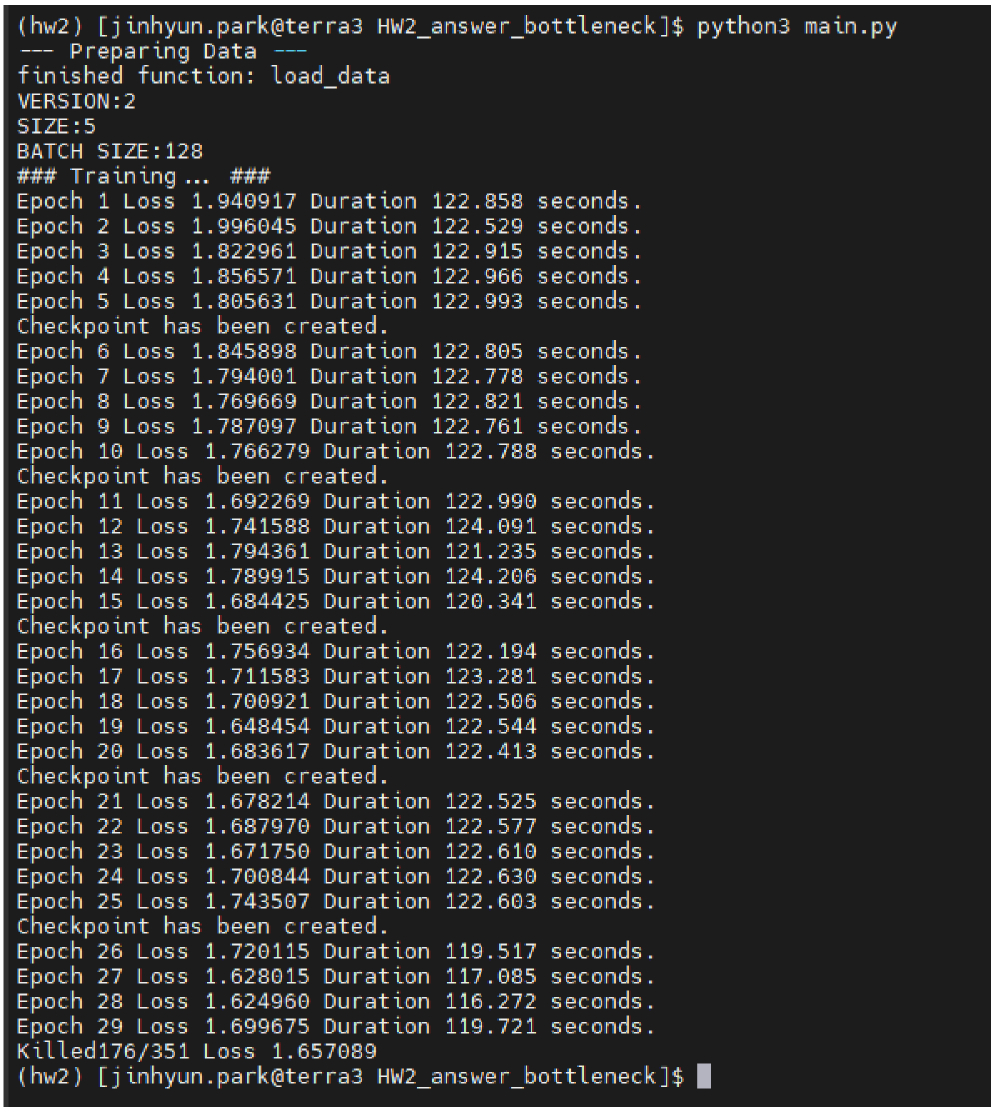
iv. Testing on `x_test` and `y_test` (30 epochs – I could not run more than this because the server automatically *killed* my process): 74.36%



```
(hw2) [jinhyun.park@terra3 HW2_answer_standard]$ python3 main.py
--- Preparing Data ---
finished function: load_data
VERSION:1
SIZE:5
BATCH SIZE:128
### Test or Validation ###
Restored model parameters from model/standard_model-30.ckpt
100%|█████████████████████████████████████████████████████████| 10000/10000 [02:23<00:00, 69.91it/s]
Test accuracy: 0.7436
(hw2) [jinhyun.park@terra3 HW2_answer_standard]$
```

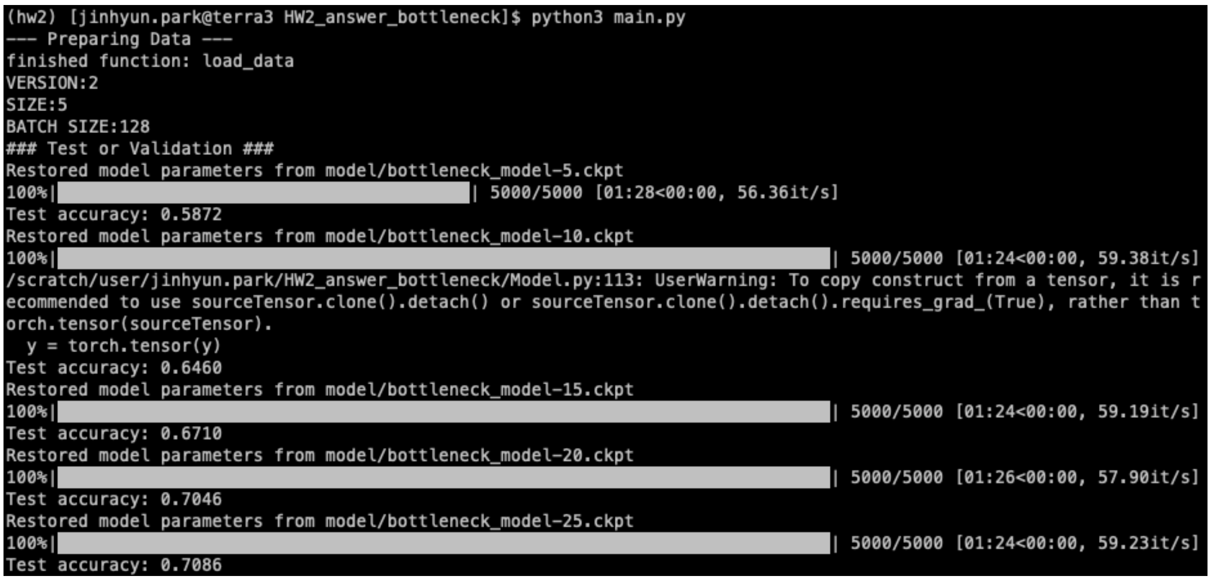
F. Bottleneck Model: Screenshots of training loss, validation/test accuracy (network size = 5).

i. Training on `x_train_new` and `y_train_new`



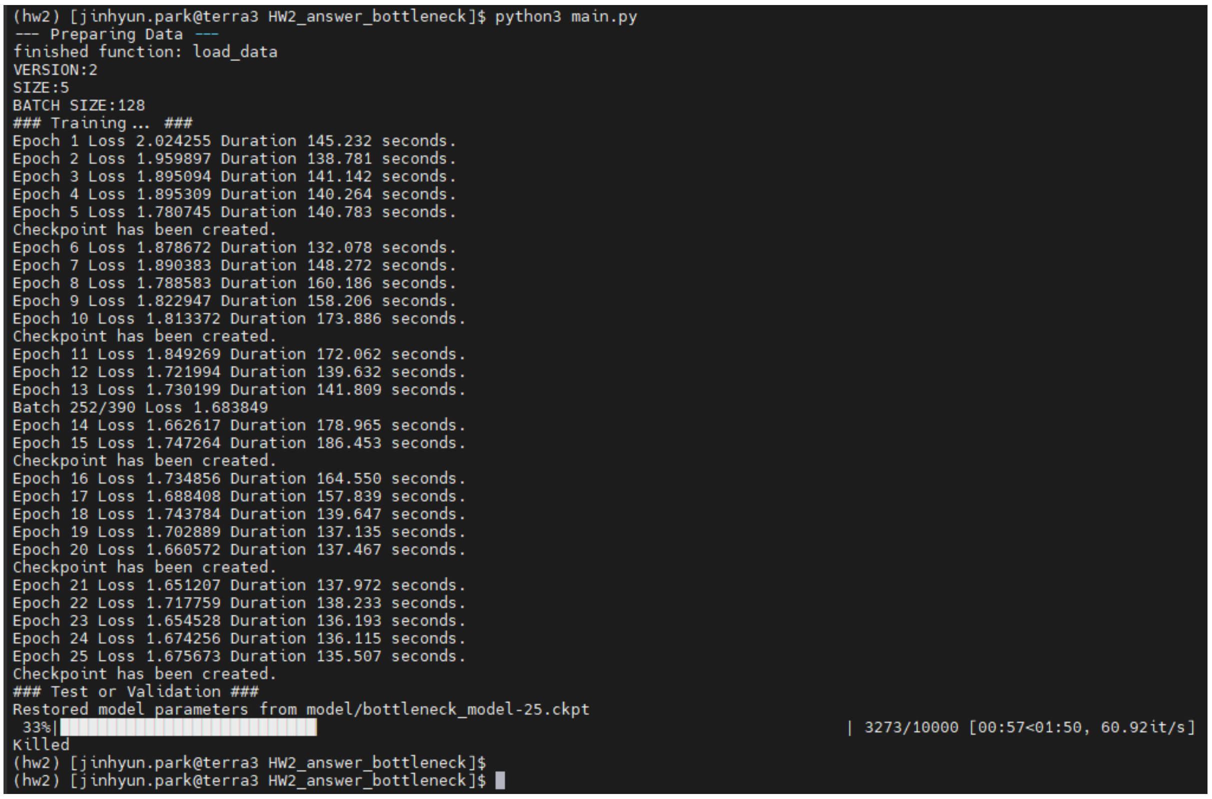
```
(hw2) [jinhyun.park@terra3 HW2_answer_bottleneck]$ python3 main.py
--- Preparing Data ---
finished function: load_data
VERSION:2
SIZE:5
BATCH SIZE:128
### Training... ###
Epoch 1 Loss 1.940917 Duration 122.858 seconds.
Epoch 2 Loss 1.996045 Duration 122.529 seconds.
Epoch 3 Loss 1.822961 Duration 122.915 seconds.
Epoch 4 Loss 1.856571 Duration 122.966 seconds.
Epoch 5 Loss 1.805631 Duration 122.993 seconds.
Checkpoint has been created.
Epoch 6 Loss 1.845898 Duration 122.805 seconds.
Epoch 7 Loss 1.794001 Duration 122.778 seconds.
Epoch 8 Loss 1.769669 Duration 122.821 seconds.
Epoch 9 Loss 1.787097 Duration 122.761 seconds.
Epoch 10 Loss 1.766279 Duration 122.788 seconds.
Checkpoint has been created.
Epoch 11 Loss 1.692269 Duration 122.990 seconds.
Epoch 12 Loss 1.741588 Duration 124.091 seconds.
Epoch 13 Loss 1.794361 Duration 121.235 seconds.
Epoch 14 Loss 1.789915 Duration 124.206 seconds.
Epoch 15 Loss 1.684425 Duration 120.341 seconds.
Checkpoint has been created.
Epoch 16 Loss 1.756934 Duration 122.194 seconds.
Epoch 17 Loss 1.711583 Duration 123.281 seconds.
Epoch 18 Loss 1.700921 Duration 122.506 seconds.
Epoch 19 Loss 1.648454 Duration 122.544 seconds.
Epoch 20 Loss 1.683617 Duration 122.413 seconds.
Checkpoint has been created.
Epoch 21 Loss 1.678214 Duration 122.525 seconds.
Epoch 22 Loss 1.687970 Duration 122.577 seconds.
Epoch 23 Loss 1.671750 Duration 122.610 seconds.
Epoch 24 Loss 1.700844 Duration 122.630 seconds.
Epoch 25 Loss 1.743507 Duration 122.603 seconds.
Checkpoint has been created.
Epoch 26 Loss 1.720115 Duration 119.517 seconds.
Epoch 27 Loss 1.628015 Duration 117.085 seconds.
Epoch 28 Loss 1.624960 Duration 116.272 seconds.
Epoch 29 Loss 1.699675 Duration 119.721 seconds.
Killed176/351 Loss 1.657089
(hw2) [jinhyun.park@terra3 HW2_answer_bottleneck]$
```

ii. Validation on `x_valid and y_valid`



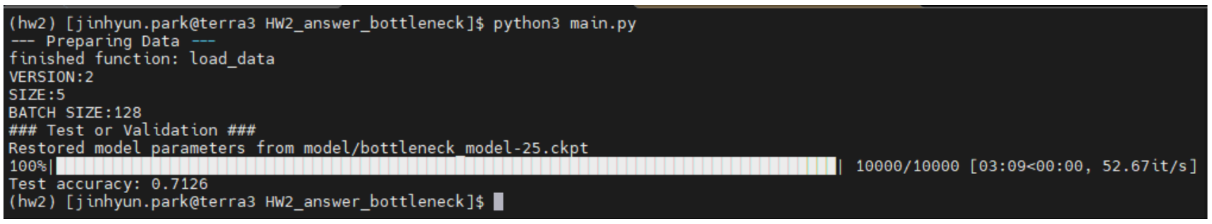
```
(hw2) [jinhyun.park@terra3 HW2_answer_bottleneck]$ python3 main.py
--- Preparing Data ---
finished function: load_data
VERSION:2
SIZE:5
BATCH SIZE:128
### Test or Validation ###
Restored model parameters from model/bottleneck_model-5.ckpt
100%|████████████████████████████████| 5000/5000 [01:28<00:00, 56.36it/s]
Test accuracy: 0.5872
Restored model parameters from model/bottleneck_model-10.ckpt
100%|████████████████████████████████████████████████████████████| 5000/5000 [01:24<00:00, 59.38it/s]
/scratch/user/jinhyun.park/HW2_answer_bottleneck/Model.py:113: UserWarning: To copy construct from a tensor, it is r
ecommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(True), rather than t
orch.tensor(sourceTensor).
  y = torch.tensor(y)
Test accuracy: 0.6460
Restored model parameters from model/bottleneck_model-15.ckpt
100%|████████████████████████████████████████████████████████████| 5000/5000 [01:24<00:00, 59.19it/s]
Test accuracy: 0.6710
Restored model parameters from model/bottleneck_model-20.ckpt
100%|████████████████████████████████████████████████████████████| 5000/5000 [01:26<00:00, 57.90it/s]
Test accuracy: 0.7046
Restored model parameters from model/bottleneck_model-25.ckpt
100%|████████████████████████████████████████████████████████████| 5000/5000 [01:24<00:00, 59.23it/s]
Test accuracy: 0.7086
```

iii. Training on `x_train` and `y_train`



```
(hw2) [jinhyun.park@terra3 HW2_answer_bottleneck]$ python3 main.py
--- Preparing Data ---
finished function: load_data
VERSION:2
SIZE:5
BATCH SIZE:128
### Training... ###
Epoch 1 Loss 2.024255 Duration 145.232 seconds.
Epoch 2 Loss 1.959897 Duration 138.781 seconds.
Epoch 3 Loss 1.895094 Duration 141.142 seconds.
Epoch 4 Loss 1.895309 Duration 140.264 seconds.
Epoch 5 Loss 1.780745 Duration 140.783 seconds.
Checkpoint has been created.
Epoch 6 Loss 1.878672 Duration 132.078 seconds.
Epoch 7 Loss 1.890383 Duration 148.272 seconds.
Epoch 8 Loss 1.788583 Duration 160.186 seconds.
Epoch 9 Loss 1.822947 Duration 158.206 seconds.
Epoch 10 Loss 1.813372 Duration 173.886 seconds.
Checkpoint has been created.
Epoch 11 Loss 1.849269 Duration 172.062 seconds.
Epoch 12 Loss 1.721994 Duration 139.632 seconds.
Epoch 13 Loss 1.730199 Duration 141.809 seconds.
Batch 252/390 Loss 1.683849
Epoch 14 Loss 1.662617 Duration 178.965 seconds.
Epoch 15 Loss 1.747264 Duration 186.453 seconds.
Checkpoint has been created.
Epoch 16 Loss 1.734856 Duration 164.550 seconds.
Epoch 17 Loss 1.688408 Duration 157.839 seconds.
Epoch 18 Loss 1.743784 Duration 139.647 seconds.
Epoch 19 Loss 1.702889 Duration 137.135 seconds.
Epoch 20 Loss 1.660572 Duration 137.467 seconds.
Checkpoint has been created.
Epoch 21 Loss 1.651207 Duration 137.972 seconds.
Epoch 22 Loss 1.717759 Duration 138.233 seconds.
Epoch 23 Loss 1.654528 Duration 136.193 seconds.
Epoch 24 Loss 1.674256 Duration 136.115 seconds.
Epoch 25 Loss 1.675673 Duration 135.507 seconds.
Checkpoint has been created.
### Test or Validation ###
Restored model parameters from model/bottleneck_model-25.ckpt
 33%|███████████████████████████                                                      | 3273/10000 [00:57<01:50, 60.92it/s]
Killed
(hw2) [jinhyun.park@terra3 HW2_answer_bottleneck]$
(hw2) [jinhyun.park@terra3 HW2_answer_bottleneck]$
```

iv. Testing on `x_test` and `y_test` (25 epochs – I could not run more than this because the server automatically *killed* my process): 71.26%



```
(hw2) [jinhyun.park@terra3 HW2_answer_bottleneck]$ python3 main.py
--- Preparing Data ---
finished function: load_data
VERSION:2
SIZE:5
BATCH SIZE:128
### Test or Validation ###
Restored model parameters from model/bottleneck_model-25.ckpt
100%|████████████████████████████████████████████████████████████████████████████████| 10000/10000 [03:09<00:00, 52.67it/s]
Test accuracy: 0.7126
(hw2) [jinhyun.park@terra3 HW2_answer_bottleneck]$
```

Hyper-parameter tuning

I used to the same hyper-parameter setting as the ResNet paper (See page 7 in [1]). They used a weight decay of 0.0001, a momentum of 0.9, no dropout, a mini-batch size of 128, and a learning rate 0.1. I used exactly the same hyper-parameter setting with network size of 5.

IMPORTANT NOTE

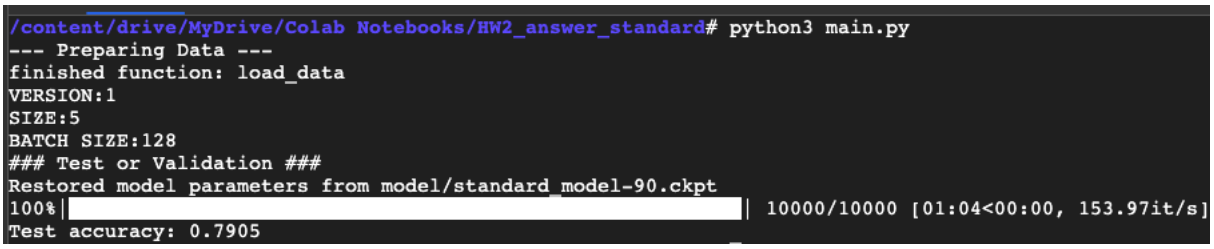
I used *@terra3.tamu.edu* for GPU processing. However, the server killed my GPU processing after +/- 30 epochs so I could not run more epochs than that. I used other programs such as *tmux* to avoid this problem but it did not work. You can check the message "*Killed*" from the figures above.

To avoid this problem, I used Google Colab Pro GPU. However, the problem was I could not run my program with Python 3.8 and Pytorch 1.7. This is because, as you know, if I use Colab, they not only assign a random GPU server but also flush the programs I have installed when I restart the session.

I did not change a single line of code to run my code in Google Colab. In other words, I used the exactly same code in both *@terra3.tamu.edu* and Google Colab.

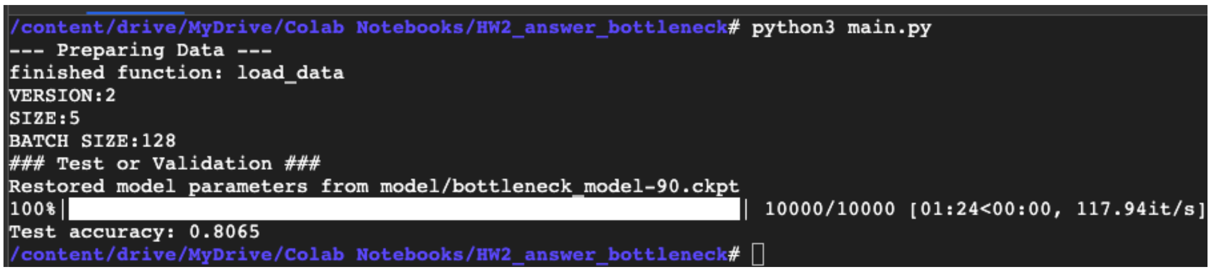
Colab enabled me to run more epochs and I got way better accuracy with long epochs. Please see below (90 epochs).

1. Standard Model Test accuracy: 79.05%



```
/content/drive/MyDrive/Colab Notebooks/HW2_answer_standard# python3 main.py
--- Preparing Data ---
finished function: load_data
VERSION:1
SIZE:5
BATCH SIZE:128
### Test or Validation ###
Restored model parameters from model/standard_model-90.ckpt
100%|██████████████████████████████████████████████████| 10000/10000 [01:04<00:00, 153.97it/s]
Test accuracy: 0.7905
```

2. Bottleneck Model Test accuracy: 80.65%



```
/content/drive/MyDrive/Colab Notebooks/HW2_answer_bottleneck# python3 main.py
--- Preparing Data ---
finished function: load_data
VERSION:2
SIZE:5
BATCH SIZE:128
### Test or Validation ###
Restored model parameters from model/bottleneck_model-90.ckpt
100%|██████████████████████████████████████████████████| 10000/10000 [01:24<00:00, 117.94it/s]
Test accuracy: 0.8065
/content/drive/MyDrive/Colab Notebooks/HW2_answer_bottleneck# 
```

3. You can check more details in the following URL (I have all training and validation history): https://drive.google.com/drive/folders/1YQZw6AYmJY5XMJOzuHX48lz9vGR9i8_c?usp=sharing

Google Colab also did not allow me to run more than +/- 100 epochs. However, if it is possible to run 200 epochs, I expect to observe over-fitting. This implies that I would get a certain accuracy for the training phase but get lower accuracy than the certain training accuracy for the testing phase.

Please check the *Appendix* Folder for more details.