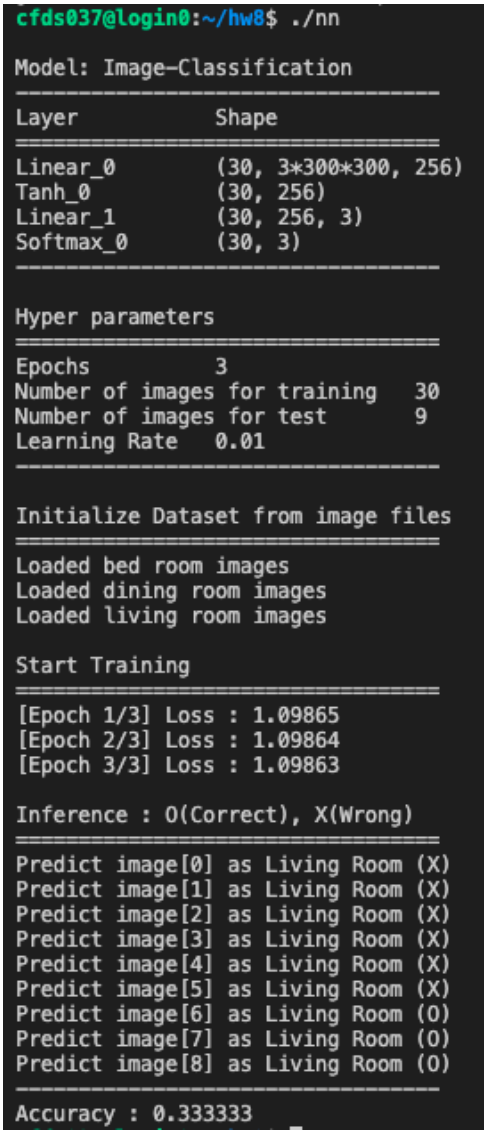
CFDS Homework 8

2022-24790 박성우 (CFDS037)

Report

1. 먼저, 주어진 대로 구현을 완료했을 때의 실행결과는 아래 그림과 같다.



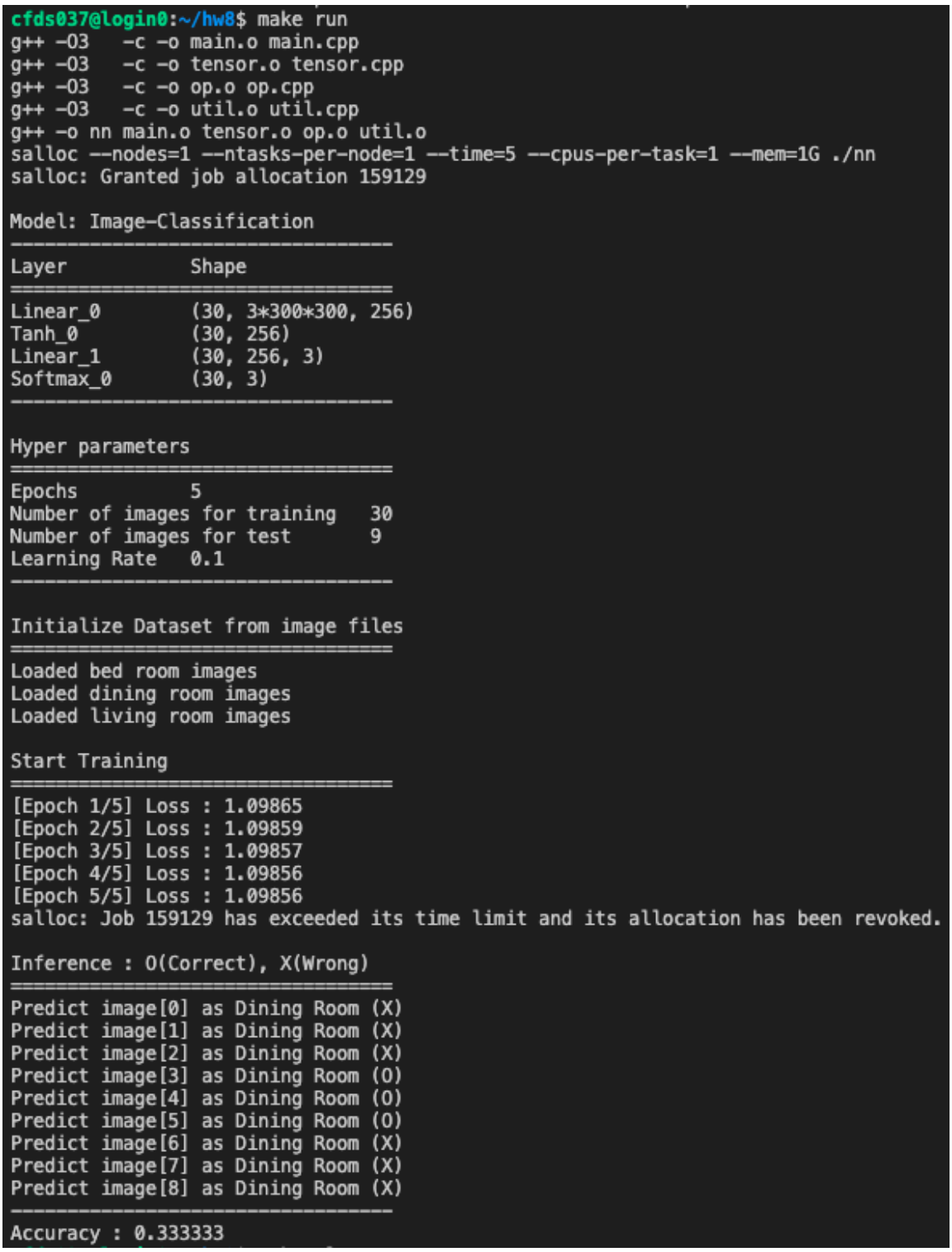
   - 그림에서 알 수 있듯이 Epoch이 진행됨에 따라서 loss가 미약하게나마 줄어드는 것을 확인할 수 있다.

2. Epoch을 3에서 10으로 늘렸을 때의 실행결과는 아래 그림과 같다.

```
cfds037@login0:~/hw8$ make run
g++ -O3   -c -o main.o main.cpp
g++ -O3   -c -o tensor.o tensor.cpp
g++ -O3   -c -o op.o op.cpp
g++ -O3   -c -o util.o util.cpp
g++ -o nn main.o tensor.o op.o util.o
salloc --nodes=1 --ntasks-per-node=1 --time=5 --cpus-per-task=1 --mem=1G ./nn
salloc: Granted job allocation 159125

Model: Image-Classification
---------------------------------
Layer           Shape
=================================
Linear_0        (30, 3*300*300, 256)
Tanh_0          (30, 256)
Linear_1        (30, 256, 3)
Softmax_0       (30, 3)
---------------------------------

Hyper parameters
=================================
Epochs          10
Number of images for training   30
Number of images for test       9
Learning Rate   0.01
---------------------------------

Initialize Dataset from image files
=================================
Loaded bed room images
Loaded dining room images
Loaded living room images

Start Training
=================================
[Epoch 1/10] Loss : 1.09865
[Epoch 2/10] Loss : 1.09864
[Epoch 3/10] Loss : 1.09863
[Epoch 4/10] Loss : 1.09863
[Epoch 5/10] Loss : 1.09862
salloc: Job 159125 has exceeded its time limit and its allocation has been revoked.
[Epoch 6/10] Loss : 1.09862
[Epoch 7/10] Loss : 1.09861
[Epoch 8/10] Loss : 1.09861
[Epoch 9/10] Loss : 1.0986
[Epoch 10/10] Loss : 1.0986

Inference : O(Correct), X(Wrong)
=================================
Predict image[0] as Dining Room (X)
Predict image[1] as Dining Room (X)
Predict image[2] as Dining Room (X)
Predict image[3] as Dining Room (O)
Predict image[4] as Dining Room (O)
Predict image[5] as Dining Room (O)
Predict image[6] as Dining Room (X)
Predict image[7] as Dining Room (X)
Predict image[8] as Dining Room (X)
---------------------------------
Accuracy : 0.333333
```

- Learning rate을 0.01로 고정했음에도, epoch 수를 늘려줌에 따라서 training loss가 epoch이 3이었던 경우보다 training loss가 더 많이 감소했음을 확인할 수 있다.
- 다만 여전히 정확도의 측면에서는 이전과 비슷한 성능을 보였다. 이는 개인적으로 image classification에서 자주 사용되는 CNN 대신 Fully-connected layer를 사용했기 때문으로 여겨진다.

3. Learning rate을 0.01에서 0.1로 늘렸을 때의 결과는 아래와 같다.



```
cfds037@login0:~/hw8$ make run
g++ -O3   -c -o main.o main.cpp
g++ -O3   -c -o tensor.o tensor.cpp
g++ -O3   -c -o op.o op.cpp
g++ -O3   -c -o util.o util.cpp
g++ -o nn main.o tensor.o op.o util.o
salloc --nodes=1 --ntasks-per-node=1 --time=5 --cpus-per-task=1 --mem=1G ./nn
salloc: Granted job allocation 159129

Model: Image-Classification
---------------------------------
Layer           Shape
=================================
Linear_0        (30, 3*300*300, 256)
Tanh_0          (30, 256)
Linear_1        (30, 256, 3)
Softmax_0       (30, 3)
---------------------------------

Hyper parameters
=================================
Epochs          5
Number of images for training   30
Number of images for test       9
Learning Rate   0.1
---------------------------------

Initialize Dataset from image files
=================================
Loaded bed room images
Loaded dining room images
Loaded living room images

Start Training
=================================
[Epoch 1/5] Loss : 1.09865
[Epoch 2/5] Loss : 1.09859
[Epoch 3/5] Loss : 1.09857
[Epoch 4/5] Loss : 1.09856
[Epoch 5/5] Loss : 1.09856
salloc: Job 159129 has exceeded its time limit and its allocation has been revoked.

Inference : O(Correct), X(Wrong)
=================================
Predict image[0] as Dining Room (X)
Predict image[1] as Dining Room (X)
Predict image[2] as Dining Room (X)
Predict image[3] as Dining Room (O)
Predict image[4] as Dining Room (O)
Predict image[5] as Dining Room (O)
Predict image[6] as Dining Room (X)
Predict image[7] as Dining Room (X)
Predict image[8] as Dining Room (X)
---------------------------------
Accuracy : 0.333333
```

- Learning rate이란 weight을 update할 때 gradient가 반영되는 비중을 결정하는 hyper-parameter이다. 이때 learning rate을 늘릴수록 weight의 update 속도가 빨라지기 때문에 loss가 더욱 빠르게 감소할 수 있으며, 위의 실행 화면은 그러한 결과를 잘 보여준다.

- 다만 여전히 accuracy는 33%의 수준을 벗어나지 못하는데, 이는 상기했듯이 fully-connected layer에 따른 근본적인 한계라고 생각된다.