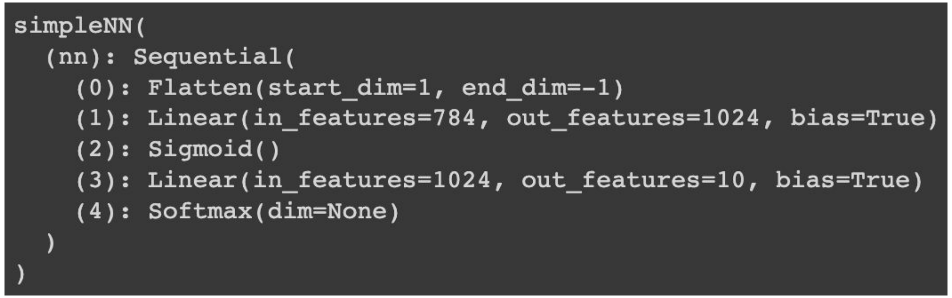
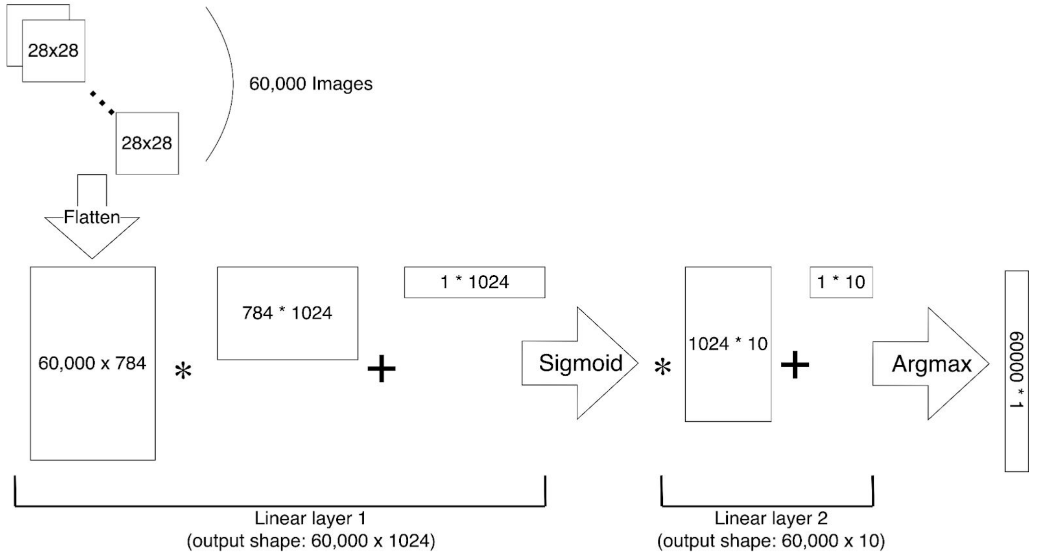
Lab 5: OpenACC

DNN Model

- 2 fully-connected layers
- Recognize handwritten digits (0~9)
- Input dimension: 28x28
- Batch size: 60,000



```
simpleNN(
  (nn): Sequential(
    (0): Flatten(start_dim=1, end_dim=-1)
    (1): Linear(in_features=784, out_features=1024, bias=True)
    (2): Sigmoid()
    (3): Linear(in_features=1024, out_features=10, bias=True)
    (4): Softmax(dim=None)
  )
)
```



DNN Model

- 3 types of calculation:
  - Single Layer ($y = aX + b$)
  - Sigmoid
  - Argmax

Task

- The sequential code requires about 30~40 seconds
- Please parallelize the code using OpenACC (GPU).

Template Files

- CPU sequential code: `lab5.cpp`
- Pre-trained model weights
- Makefile

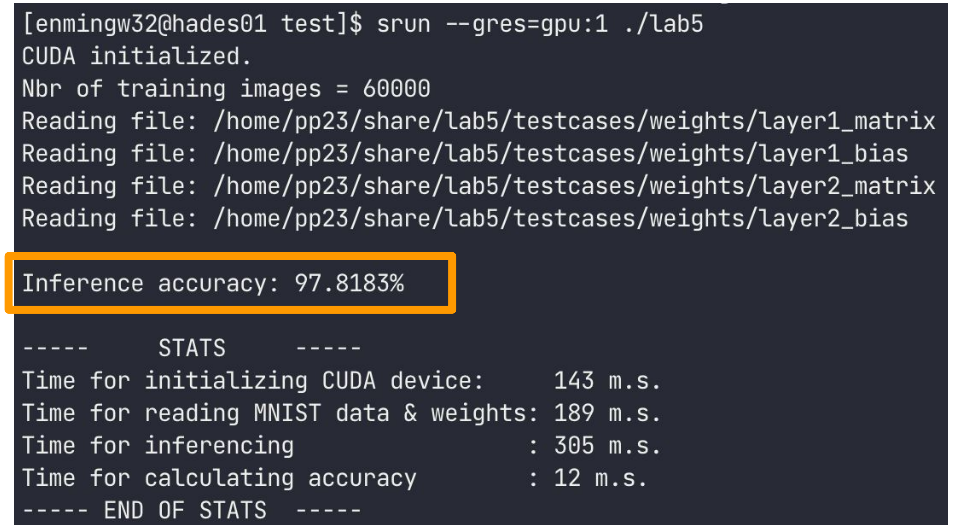
Files are located at: `/home/pp23/share/lab5/`

Only `my_nn()` and functions invoked inside `my_nn()` can be modified.

Compile & Execute

- Load module (NVIDIA HPC SDK): `module load nvhpc`
- Compile: `make`
- Execute: `srun --gres=gpu:1 ./lab5`
- Judge: `lab5-judge`

The expected inference accuracy is 97.8183%



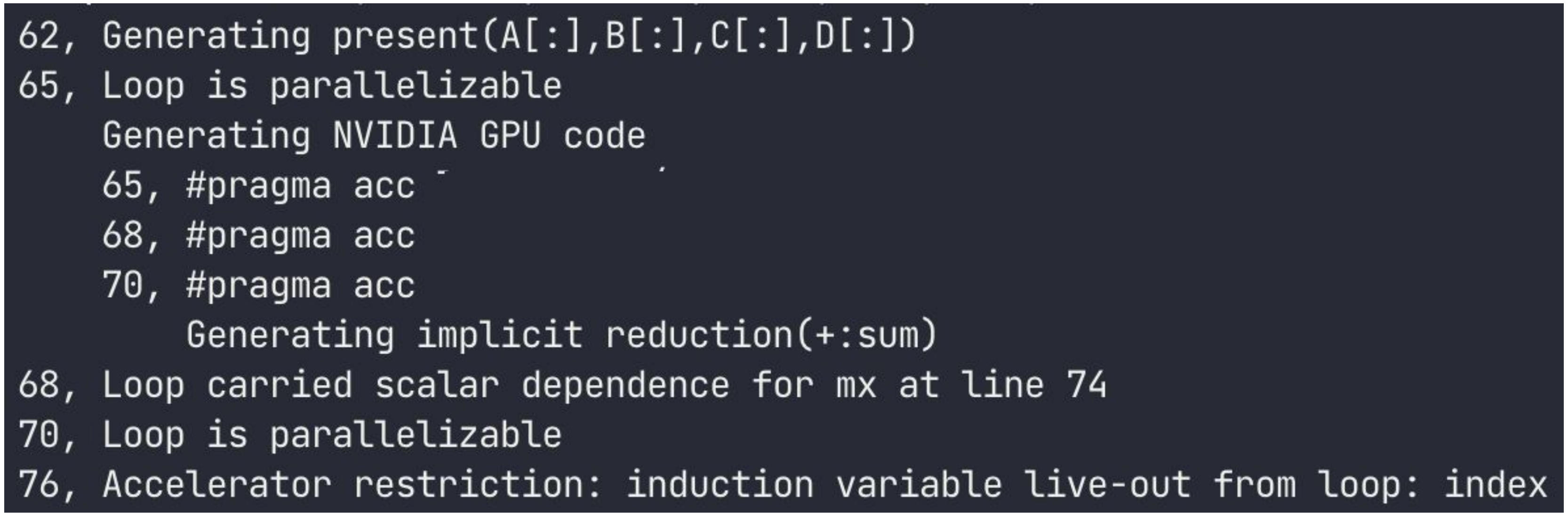
```
[enmingw32@hades01 test]$ srun --gres=gpu:1 ./lab5
CUDA initialized.
Nbr of training images = 60000
Reading file: /home/pp23/share/lab5/testcases/weights/layer1_matrix
Reading file: /home/pp23/share/lab5/testcases/weights/layer1_bias
Reading file: /home/pp23/share/lab5/testcases/weights/layer2_matrix
Reading file: /home/pp23/share/lab5/testcases/weights/layer2_bias

Inference accuracy: 97.8183%

-----     STATS     -----
Time for initializing CUDA device:     143 m.s.
Time for reading MNIST data & weights: 189 m.s.
Time for inferencing                 : 305 m.s.
Time for calculating accuracy        : 12 m.s.
----- END OF STATS  -----
```

Compilation details

- `nvc++` will print the loops that is compiled successfully for running on GPU.
- The file extension should be `.cpp`, do not modify it to `.cu`. Otherwise, `nvc++` would not compile it.



```
62, Generating present(A[:],B[:],C[:],D[:])
65, Loop is parallelizable
    Generating NVIDIA GPU code
    65, #pragma acc
    68, #pragma acc
    70, #pragma acc
        Generating implicit reduction(+:sum)
68, Loop carried scalar dependence for mx at line 74
70, Loop is parallelizable
76, Accelerator restriction: induction variable live-out from loop: index
```