HPML ASSIGNMENT 2

Preetham Rakshith Prakash | Net ID : pp2959 | pp2959@nyu.edu

C1:

The Code submission for C1 are 3 main files that are lab2.py, loader.py and model.py.
All experiments are executed via the lab2.py file using CLI arguments. i.e.

```
python3 lab2.py --device=cuda --num-of-workers=2 --epochs=5 --optimizer=sgd \
--batch-size=128 \
--dataset-path=$DATASET_PATH
```

Or to run code 3 as :

```
python3 lab2.py --device=cuda --num-of-workers=2 --epochs=5 --optimizer=sgd \
--dataset-path=$DATASET_PATH \
--run-code-3=true \
--max-num-of-workers-for-code-3=8 \
--print-epochs=false
```

The model.py and loader.py files consists code for the ResNet18 model and for the data loader. The lab2.py file imports model.py and loader.py for execution.

C2:

The average running times for the data loader, training and overall epoch from code C1 are as follows:

C2.1 : The Average Data Loading time (excluding of time taken to move the data to device) is **0.473 seconds** over each epoch.

C2.2 : The Training time (includes only mini-batch calculation with cuda synchronized) is **17.93 seconds** per epoch.

C2.3 : The total average running time over each epoch is **20.62 seconds**.

Note : The above measurements exclude the time taken during the initial warm-up phase when an epoch starts and also excludes the time taken to parse and calculate loss & accuracy values during each iteration in an epoch.

To run C2 use the CLI arguments with lab2.py file:

```
DATASET_PATH='~/dataset/path'
```

```
python3 lab2.py --device=cuda --num-of-workers=2 --epochs=5 --optimizer=sgd --batch-size=128 \
--dataset-path=$DATASET_PATH
```

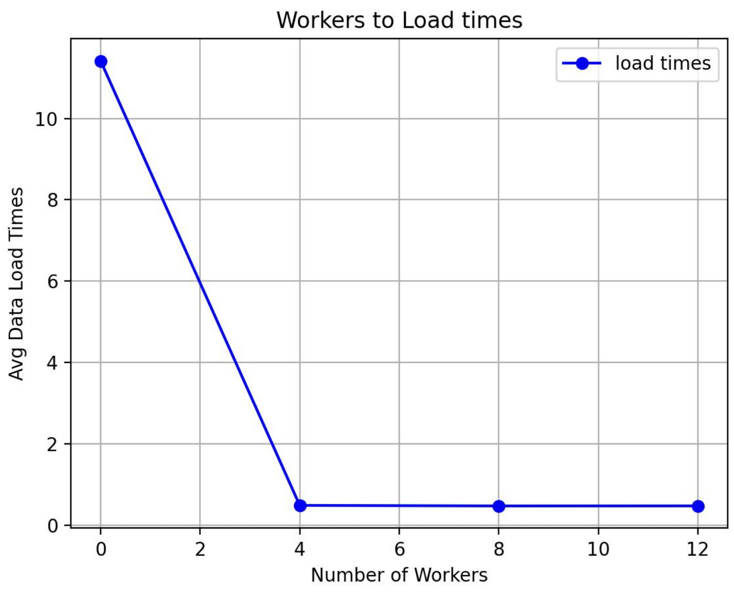
And The output for C2 is as follows:

```
Number of trainable parameters :  11173962
----------------------------------------------
Epoch :  1
Time taken for Epoch :  27.6128 sec
Load Time for epoch :  0.490151 sec with 2 workers  | Train Time :  24.585779 sec
Avg Loss :  1.881  | Avg Accuracy :  33.016 %
Top 1 accuracy is  1 among the individual label accuracies [37.86 46.06 13.42 22.   22.24 30.78
41.84 39.9  39.84 36.22]
----------------------------------------------
Epoch :  2
Time taken for Epoch :  19.1839 sec
Load Time for epoch :  0.485490 sec with 2 workers  | Train Time :  16.615530 sec
Avg Loss :  1.324  | Avg Accuracy :  51.784 %
Top 1 accuracy is  1 among the individual label accuracies [53.54 68.32 33.8  30.04 36.56 46.84
64.78 58.32 62.86 62.78]
----------------------------------------------
Epoch :  3
Time taken for Epoch :  19.2832 sec
Load Time for epoch :  0.478553 sec with 2 workers  | Train Time :  16.615411 sec
Avg Loss :  1.013  | Avg Accuracy :  63.916 %
Top 1 accuracy is  1 among the individual label accuracies [64.62 80.44 51.22 39.66 53.78 55.3
73.36 66.96 75.64 78.18]
----------------------------------------------
Epoch :  4
Time taken for Epoch :  19.3784 sec
Load Time for epoch :  0.475990 sec with 2 workers  | Train Time :  16.607613 sec
Avg Loss :  0.826  | Avg Accuracy :  70.554 %
Top 1 accuracy is  1 among the individual label accuracies [72.62 85.86 58.12 49.02 63.38 60.82
77.02 73.12 82.12 83.46]
----------------------------------------------
Epoch :  5
Time taken for Epoch :  19.4094 sec
Load Time for epoch :  0.474647 sec with 2 workers  | Train Time :  16.606662 sec
Avg Loss :  0.686  | Avg Accuracy :  76.308 %
Top 1 accuracy is  1 among the individual label accuracies [79.1  88.4  65.48 59.16 73.12 65.96
80.98 78.74 85.92 86.22]
Number of gradiants :  11173962
Avg Data Load times with `2` workers is `0.480966` sec and Avg Train Times is `18.206199` sec
Avg epoch time is `20.9735` sec
```

C3 :

C3.1 : The average total times spent by the Data Loaders over an epoch by varying number of workers are as follows:

(a) 0 workers : **11.416 seconds**.
(b) 4 workers : **0.483 seconds**.
(c) 8 workers : **0.469 seconds**.
(d) 12 workers : **0.470 seconds.**



C3.2 : From the measurements above **4 workers** are enough for the best runtime performance, the performance do not change with more than 4 workers.

The CLI arguments to run the program for code 3 is as follows :

```
python3 lab2.py --device=cuda --num-of-workers=2 --epochs=5 --optimizer=sgd \
--dataset-path=$DATASET_PATH --run-code-3=true --max-num-of-workers-for-code-3=8 \
--print-epochs=false
```

(.... OR .... set --print-epochs=true if you would like to print outputs for each epochs while running from 0 to N workers.)

And the Output is as follows:

```
Files already downloaded and verified
-------------------------------------------------------------
Avg Data Load times with `0` workers is `11.416592` sec and Avg Train Times is `16.708047` sec
And Avg epoch time is `30.0914` sec
Files already downloaded and verified
-------------------------------------------------------------
Avg Data Load times with `4` workers is `0.483827` sec and Avg Train Times is `16.612702` sec
And Avg epoch time is `19.2964` sec
```

```
Files already downloaded and verified
-------------------------------------------------------------
Avg Data Load times with `8` workers is `0.469218` sec and Avg Train Times is `16.607552` sec
And Avg epoch time is `19.5119` sec
Files already downloaded and verified
-------------------------------------------------------------
Avg Data Load times with `12` workers is `0.470591` sec and Avg Train Times is `16.604612` sec
And Avg epoch time is `19.7041` sec
```

C4 :

The Data Load times measured between **1 worker at 0.47 sec** and **4 workers at 0.48 sec** are similar because in this case given the batch size of 128 only one worker is sufficient to load the data optimally for training. And the overall computing times are similar at roughly ~19 seconds for both.

To run the program for C4 :

```
python3 lab2.py --device=cuda --num-of-workers=1 --epochs=5 --optimizer=sgd \
--dataset-path=$DATASET_PATH
```

AND

```
python3 lab2.py --device=cuda --num-of-workers=4 --epochs=5 --optimizer=sgd \
--dataset-path=$DATASET_PATH
```

And the results of C4 are as follows:

```
Avg Data Load times with `1` workers is `0.476390` sec and Avg Train Times is `16.666748` sec
Avg epoch time is `19.2428` sec


Avg Data Load times with `4` workers is `0.485539` sec and Avg Train Times is `16.697969` sec
Avg epoch time is `19.3990` sec
```

C5 :

The measured times compared with training the model using CPU vs GPU are as follows:

Avg times over each epoch	CPU Times	GPU Times
Load Times	*0.493 sec*	*0.478 sec*
Train Times	*210.96 sec*	*16.67 sec*
Total Epoch Times	***212.25 sec***	***19.38 sec***

As we can observe from the above measurements the only bottleneck in overall training is from higher mini-batch training times from using CPUs as compared to using a GPU.

The CLI arguments to run C5 :

```
python3 lab2.py --device=cuda --num-of-workers=4 --epochs=5 --optimizer=sgd \
--dataset-path=$DATASET_PATH
```

```
python3 lab2.py --device=cpu --num-of-workers=4 --epochs=5 --optimizer=sgd \
--dataset-path=$DATASET_PATH
```

And the results are as follows:

```
Avg Data Load times with `4` workers is `0.478297` sec and Avg Train Times is `16.673350` sec
Avg epoch time is `19.3820` sec
```

```
Avg Data Load times with `4` workers is `0.493945` sec and Avg Train Times is `210.967605` sec
Avg epoch time is `212.2541` sec
```

C6 :

The training results for the given optimizers measured over 5 epochs using the same hyper parameters are as follows:

Optimizer	**Avg Train times**	**Training Loss**	**Top-1 Accuracy**
SGD	*16.68 sec*	*0.687*	*88.16 %*
SGD Nesterov	*16.77 sec*	*0.780*	*86.48 %*
Ada Grad	*16.88 sec*	*0.923*	*83.80 %*
Ada Delta	*17.15 sec*	*0.499*	*91.44 %*
Adam	*16.94 sec*	*1.802*	*48.86 %*

Comparing with the measurements the model tends to converge faster with **Ada Delta** giving higher training accuracy and better loss over minimal number of epochs.

Whereas the **Adam** optimizer results in poorer performance in converging for this particular model.

The Overall avg mini batch train times for all optimizers are similar, no optimizer improves or degrades the training time.

To run the program for C6 :

```
python3 lab2.py --device=cuda --num-of-workers=4 --epochs=5 \
--dataset-path=$DATASET_PATH  --optimizer=sgd
```

```
python3 lab2.py --device=cuda --num-of-workers=4 --epochs=5 \
--dataset-path=$DATASET_PATH --optimizer=sgd-nesterov
```

```
python3 lab2.py --device=cuda --num-of-workers=4 --epochs=5 \
--dataset-path=$DATASET_PATH --optimizer=adagrad
```

```
python3 lab2.py --device=cuda --num-of-workers=4 --epochs=5 \
--dataset-path=$DATASET_PATH --optimizer=adadelta
```

```
python3 lab2.py --device=cuda --num-of-workers=4 --epochs=5 \
--dataset-path=$DATASET_PATH --optimizer=adam
```

And the resulting outputs are as follows:

```
Using SGD
----------------------------------------------
Epoch :  5
Time taken for Epoch :  19.5341 sec
Load Time for epoch :  0.473417 sec with 4 workers  | Train Time :  16.609284 sec
Avg Loss :  0.687  | Avg Accuracy :  76.168 %
Top 1 accuracy is  1 among the individual label accuracies [78.44 88.16 65.28 58.78 71.9  67.04
81.1  80.   85.6  85.38]
Number of gradiants :  11173962
Avg Data Load times with `4` workers is `0.480175` sec and Avg Train Times is `16.689117` sec
Avg epoch time is `19.4066` sec
```

```
Using SGD-NESTROV
----------------------------------------------
Epoch :  5
Time taken for Epoch :  19.6038 sec
Load Time for epoch :  0.492906 sec with 4 workers  | Train Time :  16.702698 sec
Avg Loss :  0.780  | Avg Accuracy :  72.586 %
Top 1 accuracy is  1 among the individual label accuracies [74.44 86.48 61.7  52.56 67.14 62.6
77.74 74.7  83.86 84.64]
Number of gradiants :  11173962
Avg Data Load times with `4` workers is `0.484722` sec and Avg Train Times is `16.771207` sec
Avg epoch time is `19.4500` sec
```

```
Using ADA-GRAD
----------------------------------------------
Epoch :  5
Time taken for Epoch :  19.5577 sec
Load Time for epoch :  0.498817 sec with 4 workers  | Train Time :  16.745475 sec
Avg Loss :  0.923  | Avg Accuracy :  67.412 %
Top 1 accuracy is  1 among the individual label accuracies [69.2  83.8  54.1  47.16 60.18 56.46 73.9
68.28 80.1  80.94]
Number of gradiants :  11173962
Avg Data Load times with `4` workers is `0.490542` sec and Avg Train Times is `16.882654` sec
Avg epoch time is `19.4799` sec
```

```
Using ADA-DELTA
----------------------------------------------
Epoch :  5
Time taken for Epoch :  20.0976 sec
Load Time for epoch :  0.490570 sec with 4 workers  | Train Time :  17.074933 sec
Avg Loss :  0.499  | Avg Accuracy :  82.652 %
```

```
Top 1 accuracy is  1 among the individual label accuracies [84.54 91.44 75.6  68.38 81.86 72.86
86.62 85.48 90.3  89.44]
Number of gradiants :  11173962
Avg Data Load times with `4` workers is `0.487363` sec and Avg Train Times is `17.156327` sec
Avg epoch time is `20.0314` sec

Using Adam
---------------------------------------------
Epoch :  5
Time taken for Epoch :  19.2781 sec
Load Time for epoch :  0.493564 sec with 4 workers  | Train Time :  16.876524 sec
Avg Loss :  1.802  | Avg Accuracy :  30.614 %
Top 1 accuracy is  1 among the individual label accuracies [33.94 48.86 11.76 22.6  21.02 27.18
38.28 36.74 35.6  30.16]
Number of gradiants :  11173962
Avg Data Load times with `4` workers is `0.498469` sec and Avg Train Times is `16.946046` sec
Avg epoch time is `19.3282` sec
```

C7 :

Training the model without Batch Norm layers in the Convolution Layers results in a lower avg training times that is ***15.00*** **seconds** compared to with Batch Norm layers of **17.93 seconds**, the load times remains unchanged.

This is an improvement in training times of **around ~1.2 seconds** or more.

But the model takes more time to converge, as we can see the loss and accuracies tends to be slow to improve over the epochs as compared to using batch norm layers.

To run the code C7 :

```
python3 lab2.py --device=cuda --epochs=5 --optimizer=sgd --num-of-workers=4 \
--dataset-path=$DATASET_PATH --no-batch-norm=true
```

NOTE : By using the arguments --no-batch-norm=true the program builds the same model but without nn.BatchNorm2D( ) layers for the Convolution layers.

And the Output are as follows :

```
---------------------------------------------
Epoch :  1
Time taken for Epoch :  17.6784 sec
Load Time for epoch :  0.493526 sec with 4 workers  | Train Time :  15.267178 sec
Avg Loss :  1.949  | Avg Accuracy :  26.512 %
Top 1 accuracy is  8 among the individual label accuracies [30.34 32.58  7.46 19.38 17.2  22.32
35.84 31.26 39.62 29.12]
---------------------------------------------
Epoch :  2
Time taken for Epoch :  17.4622 sec
```

```
Load Time for epoch :  0.466602 sec with 4 workers  | Train Time :  14.931943 sec
Avg Loss :  1.599  | Avg Accuracy :  40.852 %
Top 1 accuracy is  1 among the individual label accuracies [41.2  56.1  21.26 25.04 27.4  39.3  48.28
48.16 54.18 47.6 ]
------------------------------------------------
Epoch :  3
Time taken for Epoch :  17.6006 sec
Load Time for epoch :  0.476485 sec with 4 workers  | Train Time :  14.932418 sec
Avg Loss :  1.365  | Avg Accuracy :  50.78 %
Top 1 accuracy is  1 among the individual label accuracies [49.1  68.   37.32 30.98 37.2  40.76 60.66
55.52 67.18 61.08]
------------------------------------------------
Epoch :  4
Time taken for Epoch :  17.6597 sec
Load Time for epoch :  0.486292 sec with 4 workers  | Train Time :  14.935609 sec
Avg Loss :  1.173  | Avg Accuracy :  58.694 %
Top 1 accuracy is  1 among the individual label accuracies [58.94 76.24 46.2  37.1  47.76 50.58 66.8
59.28 72.9  71.14]
------------------------------------------------
Epoch :  5
Time taken for Epoch :  17.7211 sec
Load Time for epoch :  0.466059 sec with 4 workers  | Train Time :  14.932986 sec
Avg Loss :  1.025  | Avg Accuracy :  64.23 %
Top 1 accuracy is  1 among the individual label accuracies [65.14 79.76 52.04 46.18 57.4  54.34
69.86 65.74 77.18 74.66]
Number of gradiants :  11164362
Avg Data Load times with `4` workers is `0.477793` sec and Avg Train Times is `15.000027` sec
Avg epoch time is `17.6244` sec
```

Q1 :

There are **17 Convolutional layers** in the ResNet18 model.
(a) 1 Initial Conv Layer.
(b) Followed by 8 Basic Blocks with 2 Conv Layer per Block
(c) Therefore, in total there are 17 Conv Layers i.e 1 Initial layer + 16 layers in 8 blocks.

Q2 :

The input dimension for the final Linear layer is **512**.
(a) Output dimension for the final conv layer with 512 filters in (512, 4, 4).
(b) Followed by a Average pooling layer with a filter size of (4, 4) results in output dim of (512, 1, 1)
(c) Therefore, dim for the final linear layer is (512, 10) for 512 inputs and 10 outputs.

Q3 :

The Total number of trainable parameters is **11,173,962** and the number of gradients is **11,173,962** when we use the SGD Optimizer.

The code to count the parameters and gradients :

```
def count_parameters(model):
    return sum(p.numel() for p in model.parameters() if p.requires_grad)

def count_gradients(model):
    return sum(p.grad.numel() for p in model.parameters() if p.grad is not None)
```

And the outputs are :

```
Number of trainable parameters :   11173962
Number of gradiants :   11173962
```

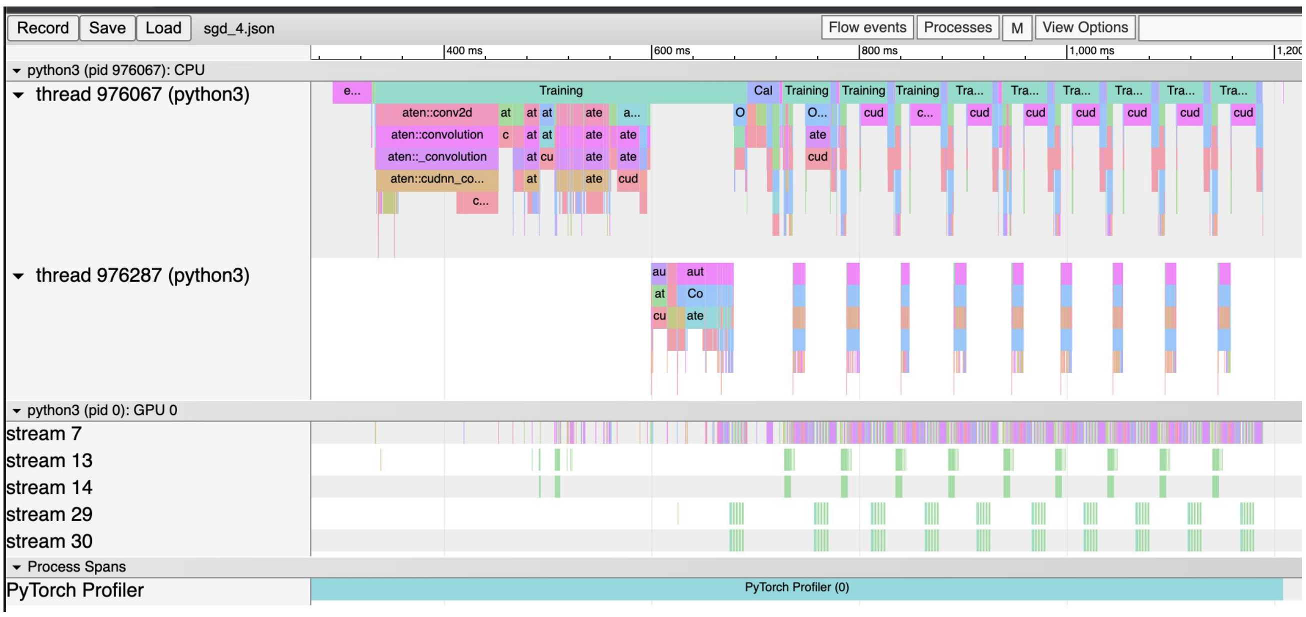
Q4 :

And the total number of parameters and gradients while using Adam optimizer is **11,173,962 .**

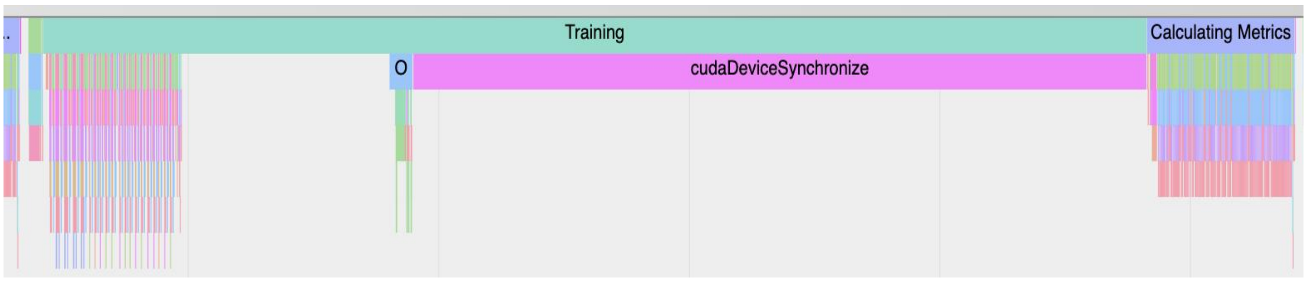
EXTRA CREDITS:

Using the pytorch profiler we can observe from the following codes C1, C2, C3 and C4:

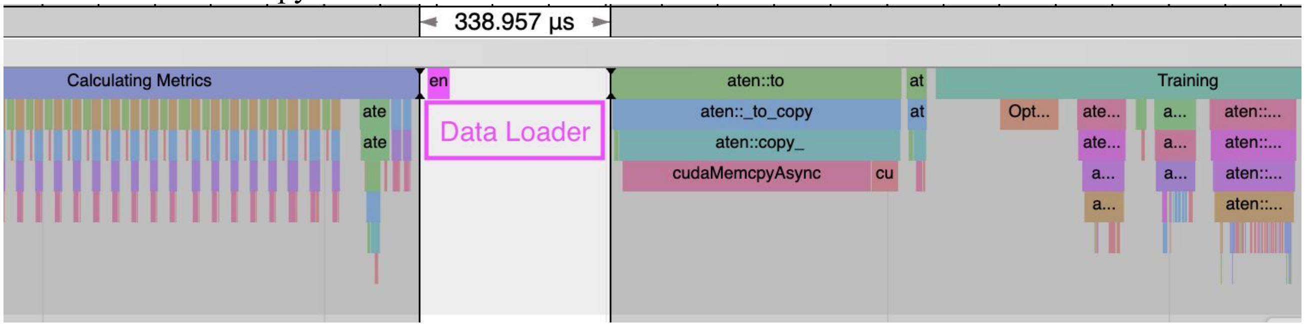
(a) The first mini-batch takes significantly longer time to train by the cuda kernel compared to rest of the training.



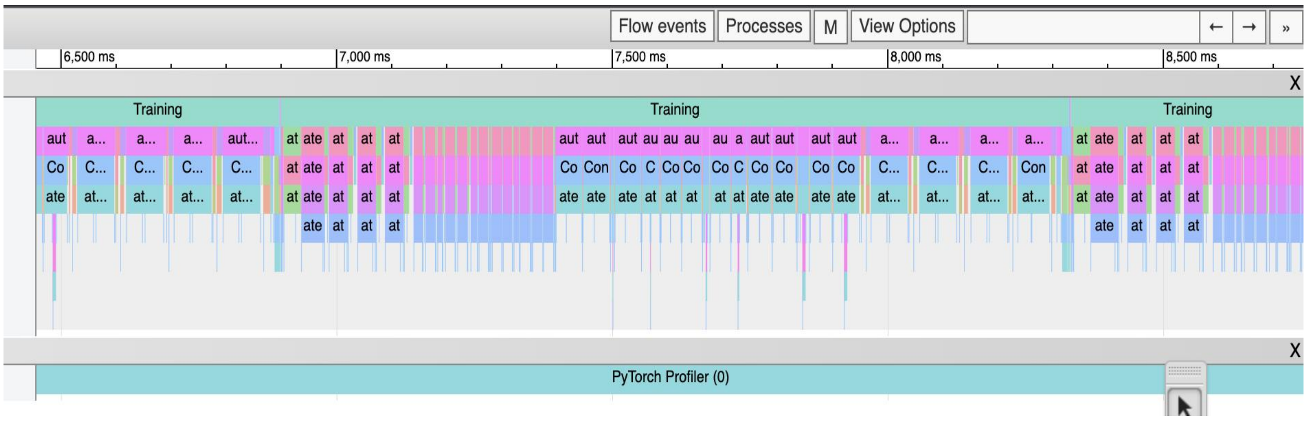
(b) The training times are significantly higher compared to the time taken for calculating metrics (here `calculating metrics` is my code that calculates loss, overall accuracy and top - 1 accuracy among the labels) followed by data load times (which is not visible in the below trace image) and data copy times.



(c) The Data Loader is much faster with 4 workers and the Load times are usually slightly less than the time it takes to copy data to the GPU.



(d) Finally for Code C5 we can observe from the trace, that majority of overall epoch times are significantly dominated by mini-batch training times. Hence, the training times on CPU is a bottleneck for the overall epoch.



To run the pytorch profiler :

```
python3 lab2.py --device=cuda --optimizer=sgd --num-of-workers=4 \
 --dataset-path=$DATASET_PATH --run-profiler=true --trace-file-name=trace-file
```

```
python3 lab2.py --device=cpu --optimizer=sgd --num-of-workers=4 \
 --dataset-path=$DATASET_PATH --run-profiler=true --trace-file-name=trace-file
```

And the ouputs are as follows:

```
Files already downloaded and verified
STAGE:2024-03-05 08:33:44 976067:976067 ActivityProfilerController.cpp:320] Completed Stage: Collection
```

```
STAGE:2024-03-05 08:33:44 976067:976067 ActivityProfilerController.cpp:324] Completed Stage: Post Processing
STAGE:2024-03-05 08:33:49 976302:976302 ActivityProfilerController.cpp:314] Completed Stage: Warm Up

Files already downloaded and verified
STAGE:2024-03-05 08:33:50 976302:976302 ActivityProfilerController.cpp:320] Completed Stage: Collection
STAGE:2024-03-05 08:33:50 976302:976302 ActivityProfilerController.cpp:324] Completed Stage: Post Processing
STAGE:2024-03-05 08:33:55 976405:976405 ActivityProfilerController.cpp:314] Completed Stage: Warm Up
```

.

References :

- https://arxiv.org/abs/1512.03385
- https://github.com/kuangliu/pytorch-cifar
- https://discuss.pytorch.org/t/how-do-i-check-the-number-of-parameters-of-a-model/4325
- https://pytorch.org/tutorials/beginner/basics/data_tutorial.html
- https://pytorch.org/tutorials/beginner/basics/transforms_tutorial.html
- https://pytorch.org/tutorials/beginner/basics/buildmodel_tutorial.html
- https://pytorch.org/tutorials/recipes/recipes/profiler_recipe.html
- https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html#torch.nn.Conv2d
- https://pytorch.org/docs/stable/optim.html
- https://sites.google.com/nyu.edu/nyu-hpc/hpc-systems/greene/software/singularity-with-miniconda