You are encouraged to implement the algorithms of the paper you choose by yourself. But it is also allowed that you use the open-sourced code online and modify them according to your demand.

We provide one example for the final project based on FBNet.

Source Code Link: https://github.com/AnnaAraslanova/FBNet

Note that the source code link we provided is just for your reference. It is your responsibility to make them run successfully on your computers or your environment. It is common that the open-sourced code cannot be run directly on your own computers due to some version or environment problem or some minor error. They could usually be fixed if you make effort on them. And it is also a lesson that we want you to learn from the final project. Besides, there might exist different source codes written by other developers online. It is decided by you about which version of source codes to use for your final project. We do not force you to use the source code we provide for your reference.

FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search

Overview

1. A background/introduction section explaining what you did and aimed to show in this project.
   Example:
   In this project, I used an open-sourced code

   [The above description is very basic and just a simple example. You are encouraged to write more related content and more detailed description.]

2. Include any challenges you faced and what you did to fix the issue.
   [I believe you will meet some problems or challenges in the final project. Any type of challenge is allowed to write in this part]

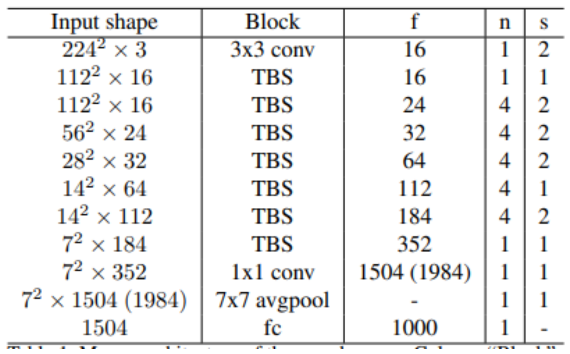
Method Description

You should list all the key points for the methods/algorithm designed in the paper.

Example:

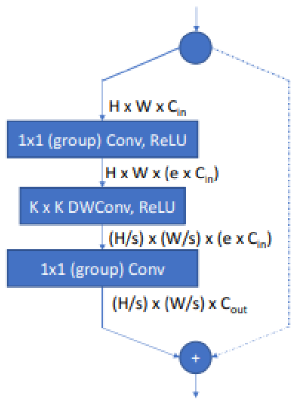
To implement a hardware-aware differentiable NAS, there are several important key points.

1. *Design the macro-architecture of the supernet. The following table shows the details about the macro-architecture. [More explanations of the macro-architecture are expected. Do not copy the statements in the paper. Try to use your own words and show your own understanding.]*

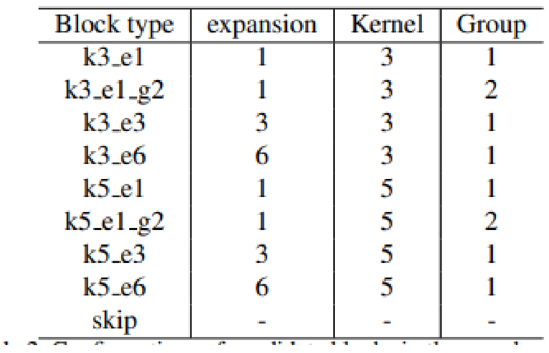


Input shape	Block	f	n	s
$224^2 \times 3$	3x3 conv	16	1	2
$112^2 \times 16$	TBS	16	1	1
$112^2 \times 16$	TBS	24	4	2
$56^2 \times 24$	TBS	32	4	2
$28^2 \times 32$	TBS	64	4	2
$14^2 \times 64$	TBS	112	4	1
$14^2 \times 112$	TBS	184	4	2
$7^2 \times 184$	TBS	352	1	1
$7^2 \times 352$	1x1 conv	1504 (1984)	1	1
$7^2 \times 1504$ (1984)	7x7 avgpool	-	1	1
1504	fc	1000	1	-

2. *Design the architecture of each block denoted as TBS. The following figure shows the details about the macro-architecture. [More explanations of the architecture are expected. Do not copy the statements in the paper. Try to use your own words and show your own understanding.]*



3. *Design the search space for each operation within the block. The following table shows the details about the search space. [More explanations of the search space are expected. Do not copy the statements in the paper. Try to use your own words and show your own understanding.]*



Block type	expansion	Kernel	Group
k3_e1	1	3	1
k3_e1_g2	1	3	2
k3_e3	3	3	1
k3_e6	6	3	1
k5_e1	1	5	1
k5_e1_g2	1	5	2
k5_e3	3	5	1
k5_e6	6	5	1
skip	-	-	-

4. *The cost function for the neural architecture search should be customized to control the trade-off between the accuracy and the latency of the hardware. The cost function is shown as follows. [More explanations of the cost function are expected. Do not copy the statements in the paper. Try to use your own words and show your own understanding.]*

$$\mathcal{L}(a, w_a) = \text{CE}(a, w_a) \cdot \alpha \log(\text{LAT}(a))^{\beta}.$$

5. *Description of the details of the search algorithm ……*

[The above description is very basic and just a simple example. The key points I listed might not be complete for the paper. But in your own report, you should make it as complete as possible.]

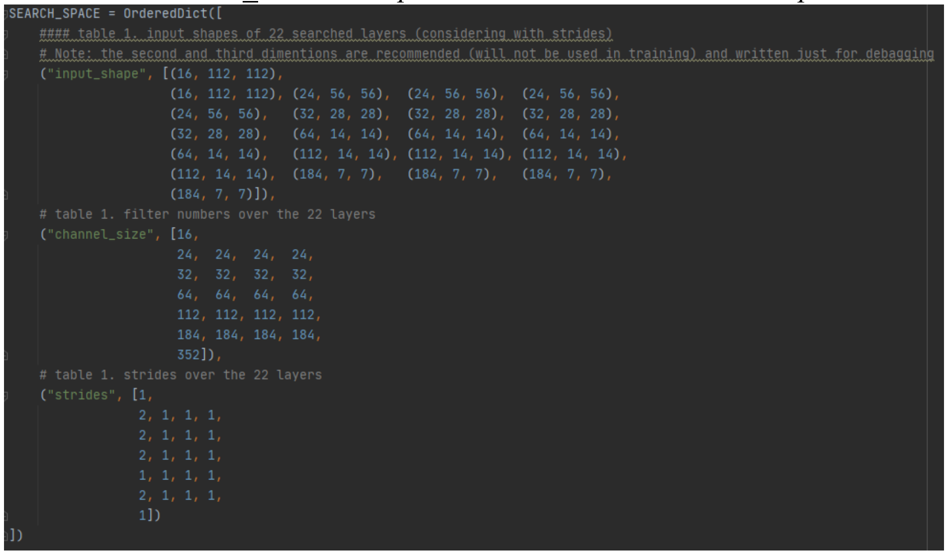
Important Code to Display and Explanations

Based on the section of method description, show the most related code snippets that implement the mentioned key points. **You need to try your best to show that you have read the code and understand what the code are doing.**

Example:

The most related code snippets of the key points mentioned above are listed as follows.

1. *Design the macro-architecture of the supernet.*
   In supernet_functions/lookup_table_builder.py, the macro-architecture of the supernet is design the dictionary SEARCH_SPACE. In the dictionary, the "input shape" keywords corresponds to … . The "channel_size" corresponds to … . The "strides" corresponds to … .



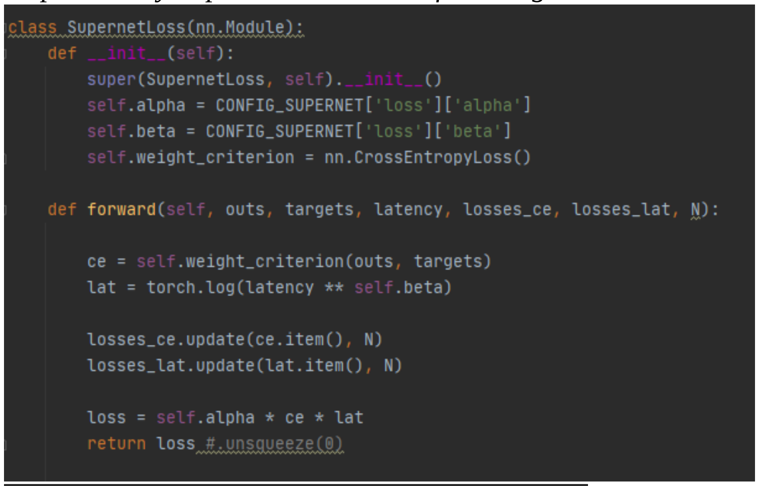
```
SEARCH_SPACE = OrderedDict([
    #### table 1. input shapes of 22 searched layers (considering with strides)
    # Note: the second and third dimentions are recommended (will not be used in training) and written just for debagging
    ("input_shape", [(16, 112, 112),
                     (16, 112, 112), (24, 56, 56),  (24, 56, 56),  (24, 56, 56),
                     (24, 56, 56),   (32, 28, 28),  (32, 28, 28),  (32, 28, 28),
                     (32, 28, 28),   (64, 14, 14),  (64, 14, 14),  (64, 14, 14),
                     (64, 14, 14),   (112, 14, 14), (112, 14, 14), (112, 14, 14),
                     (112, 14, 14),  (184, 7, 7),   (184, 7, 7),   (184, 7, 7),
                     (184, 7, 7)]),
    # table 1. filter numbers over the 22 layers
    ("channel_size", [16,
                      24,  24,  24,  24,
                      32,  32,  32,  32,
                      64,  64,  64,  64,
                      112, 112, 112, 112,
                      184, 184, 184, 184,
                      352]),
    # table 1. strides over the 22 layers
    ("strides", [1,
                 2, 1, 1, 1,
                 2, 1, 1, 1,
                 2, 1, 1, 1,
                 1, 1, 1, 1,
                 2, 1, 1, 1,
                 1])
])
```

   [It is just an example, you should explain the key idea of the implementation in your own words].

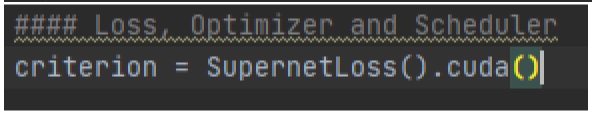
2. *Design the architecture of each block denoted as TBS*
   [In this example, I just skip the explanation. But in your own report, it cannot be skipped.]

3. *Design the search space for each operation within the block*
   [In this example, I just skip the explanation. But in your own report, it cannot be skipped.]

4. *The cost function for the neural architecture search.*
   In supernet_functions/model_supernet.py, the loss function for the supernet training in the outer loop is mainly implemented with the following codes.

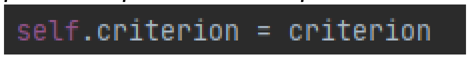


```
class SupernetLoss(nn.Module):
    def __init__(self):
        super(SupernetLoss, self).__init__()
        self.alpha = CONFIG_SUPERNET['loss']['alpha']
        self.beta = CONFIG_SUPERNET['loss']['beta']
        self.weight_criterion = nn.CrossEntropyLoss()

    def forward(self, outs, targets, latency, losses_ce, losses_lat, N):

        ce = self.weight_criterion(outs, targets)
        lat = torch.log(latency ** self.beta)

        losses_ce.update(ce.item(), N)
        losses_lat.update(lat.item(), N)

        loss = self.alpha * ce * lat
        return loss #.unsqueeze(0)
```

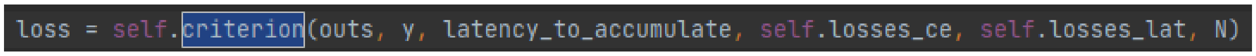


```
#### Loss, Optimizer and Scheduler
criterion = SupernetLoss().cuda()
```

The loss function is built as a subclass of nn.module. And it could be used by calling the forward function of the instance of such class. The following codes



```
self.criterion = criterion
```



```
loss = self.criterion(outs, y, latency_to_accumulate, self.losses_ce, self.losses_lat, N)
```

[It is just an example, you should explain the key idea of the implementation in your own words].

5. *Details of the search algorithm*
   [In this example, I just skip the explanation. But in your own report, it cannot be skipped.]

[In addition, you can copy all the related important code snippets at a separate python file with good comments to show your understanding of the source code. Then, submit the python file with the lab report.]

Screenshot of Running and Experimental Results

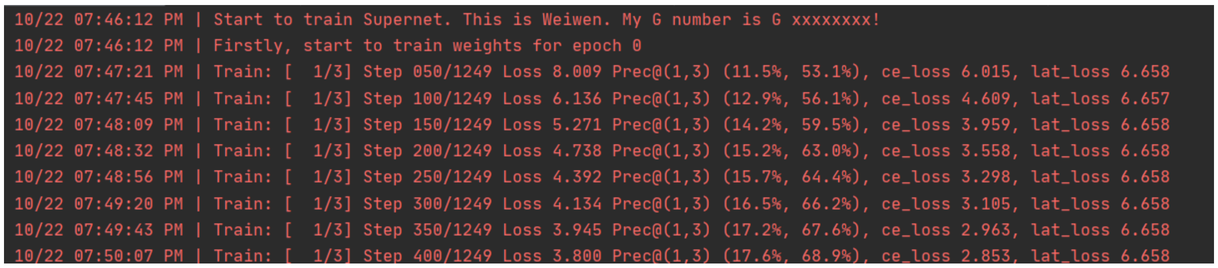
This part is closely related to the demo of your final project. You should at least show the screenshot of running and the results of the experiments you will demonstrate in the presentation of your project. Experimental results that are not shown in the presentation due to the time limit are also encouraged to listed in this section. Note that **if you fail to show the basic experimental results in your presentation, you will also lose all the points in this section** since the results you show here are not convincing.

Example:

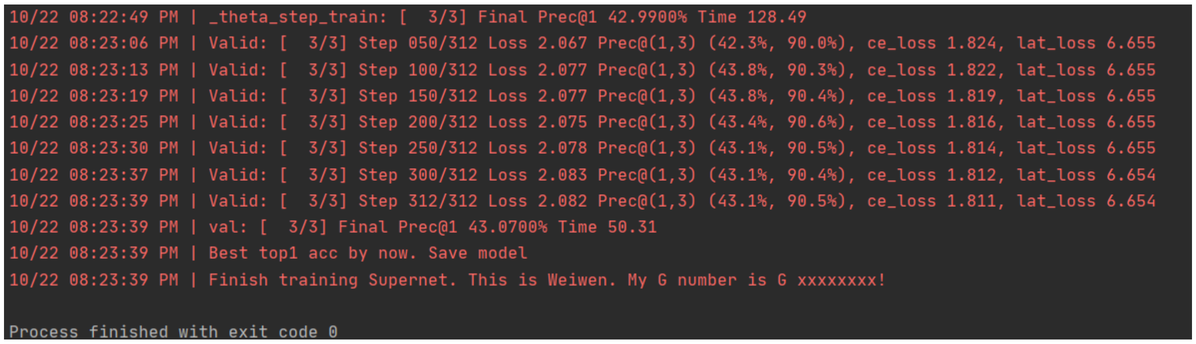
In this section, I conduct three types of experiments. For each type of experiment, I listed the screenshot of running and the experimental results, respectively.

1. *Training the supernet for neural architecture search.*
   A. *Screenshot of Running.*
   As you can see in the following two figures, the supernet is trained. And the first line of the first figure and the last line of the second figure is added by me. It shows the date when I runed the code and also be marked with my personal information.

   [Although it is not mandatory, it is encouraged that you could add some print information defined by you at each critical step of the code, which could further prove that your run the code by your own and you have read the code and are able to slightly modify the code by yourself.]



```
10/22 07:46:12 PM | Start to train Supernet. This is Weiwen. My G number is G xxxxxxxx!
10/22 07:46:12 PM | Firstly, start to train weights for epoch 0
10/22 07:47:21 PM | Train: [  1/3] Step 050/1249 Loss 8.009 Prec@(1,3) (11.5%, 53.1%), ce_loss 6.015, lat_loss 6.658
10/22 07:47:45 PM | Train: [  1/3] Step 100/1249 Loss 6.136 Prec@(1,3) (12.9%, 56.1%), ce_loss 4.609, lat_loss 6.657
10/22 07:48:09 PM | Train: [  1/3] Step 150/1249 Loss 5.271 Prec@(1,3) (14.2%, 59.5%), ce_loss 3.959, lat_loss 6.658
10/22 07:48:32 PM | Train: [  1/3] Step 200/1249 Loss 4.738 Prec@(1,3) (15.2%, 63.0%), ce_loss 3.558, lat_loss 6.658
10/22 07:48:56 PM | Train: [  1/3] Step 250/1249 Loss 4.392 Prec@(1,3) (15.7%, 64.4%), ce_loss 3.298, lat_loss 6.658
10/22 07:49:20 PM | Train: [  1/3] Step 300/1249 Loss 4.134 Prec@(1,3) (16.5%, 66.2%), ce_loss 3.105, lat_loss 6.658
10/22 07:49:43 PM | Train: [  1/3] Step 350/1249 Loss 3.945 Prec@(1,3) (17.2%, 67.6%), ce_loss 2.963, lat_loss 6.658
10/22 07:50:07 PM | Train: [  1/3] Step 400/1249 Loss 3.800 Prec@(1,3) (17.6%, 68.9%), ce_loss 2.853, lat_loss 6.658
```



```
10/22 08:22:49 PM | _theta_step_train: [  3/3] Final Prec@1 42.9900% Time 128.49
10/22 08:23:06 PM | Valid: [  3/3] Step 050/312 Loss 2.067 Prec@(1,3) (42.3%, 90.0%), ce_loss 1.824, lat_loss 6.655
10/22 08:23:13 PM | Valid: [  3/3] Step 100/312 Loss 2.077 Prec@(1,3) (43.8%, 90.3%), ce_loss 1.822, lat_loss 6.655
10/22 08:23:19 PM | Valid: [  3/3] Step 150/312 Loss 2.077 Prec@(1,3) (43.8%, 90.4%), ce_loss 1.819, lat_loss 6.655
10/22 08:23:25 PM | Valid: [  3/3] Step 200/312 Loss 2.075 Prec@(1,3) (43.4%, 90.6%), ce_loss 1.816, lat_loss 6.655
10/22 08:23:30 PM | Valid: [  3/3] Step 250/312 Loss 2.078 Prec@(1,3) (43.1%, 90.5%), ce_loss 1.814, lat_loss 6.655
10/22 08:23:37 PM | Valid: [  3/3] Step 300/312 Loss 2.083 Prec@(1,3) (43.1%, 90.4%), ce_loss 1.812, lat_loss 6.654
10/22 08:23:39 PM | Valid: [  3/3] Step 312/312 Loss 2.082 Prec@(1,3) (43.1%, 90.5%), ce_loss 1.811, lat_loss 6.654
10/22 08:23:39 PM | val: [  3/3] Final Prec@1 43.0700% Time 50.31
10/22 08:23:39 PM | Best top1 acc by now. Save model
10/22 08:23:39 PM | Finish training Supernet. This is Weiwen. My G number is G xxxxxxxx!

Process finished with exit code 0
```

   B. *Experimental Results*

   I got a supernet with a top-1 accuracy of 43.07% on the test set of CIFAR-10. The supernet is trained with learning rate = 0.1, batch size = 32, the total epoch for training = 3, …… (you could add more training parameters). Due to the time limit and lack of powerful GPUs, I trained the supernet with total of 3 epochs. The low accuracy of the supernet might be caused by this case.

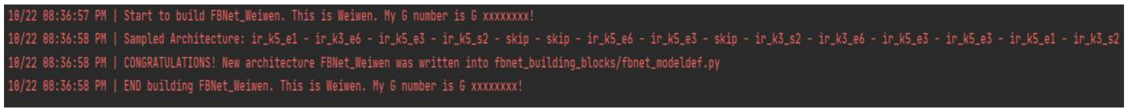
   [This is just a simple example. You should have more experimental results in your final project. It is recommended to use charts and tables to show your experimental results.]

   [Note: It is acceptable if your experimental results are not as good as the ones reported in paper due to the lack of a powerful GPU. You can write such problem in your report and be honest with experimental results you got. You will not lose points due to the bad results if it is caused by the lack of GPU resources for training.]

2. *Training the supernet for neural architecture search.*
   A. Screenshot of Running

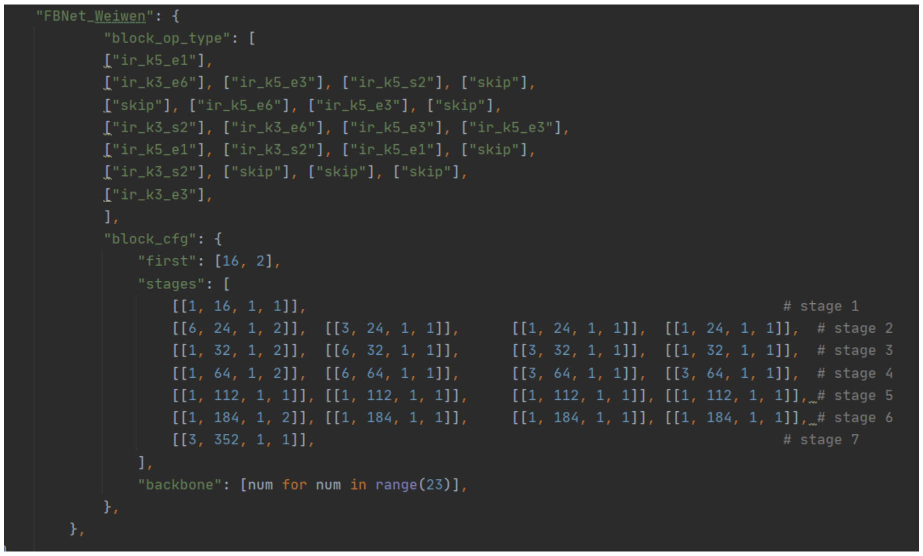
As you can see in the following figures, a neural architecture called FBNet_Weiwen is sampled from the supernet. And the first line and last line of the figure are added by me. It shows the date when I runed the code and also be marked with my personal information.

[Although it is not mandatory, it is encouraged that you could add some print information defined by you at each critical step of the code, which could further prove that your run the code by your own and you have read the code and are able to slightly modify the code by yourself.]



```
10/22 08:36:57 PM | Start to build FBNet_Weiwen. This is Weiwen. My G number is G xxxxxxxx!
10/22 08:36:58 PM | Sampled Architecture: ir_k5_e1 - ir_k3_e6 - ir_k5_e3 - ir_k5_s2 - skip - skip - ir_k5_e6 - ir_k5_e3 - skip - ir_k3_s2 - ir_k3_e6 - ir_k5_e3 - ir_k5_e3 - ir_k5_e1 - ir_k3_s2
10/22 08:36:58 PM | CONGRATULATIONS! New architecture FBNet_Weiwen was written into fbnet_building_blocks/fbnet_modeldef.py
10/22 08:36:58 PM | END building FBNet_Weiwen. This is Weiwen. My G number is G xxxxxxxx!
```

The program wrote the architecture of FBNet_Weiwen to fbnet_building_blocks/fbnet_modeldef.py automatically by function writh_new_ARCH_to_fbnet_modeldef. The architecture is shown in the following figure.



```
    "FBNet_Weiwen": {
            "block_op_type": [
            ["ir_k5_e1"],
            ["ir_k3_e6"], ["ir_k5_e3"], ["ir_k5_s2"], ["skip"],
            ["skip"], ["ir_k5_e6"], ["ir_k5_e3"], ["skip"],
            ["ir_k3_s2"], ["ir_k3_e6"], ["ir_k5_e3"], ["ir_k5_e3"],
            ["ir_k5_e1"], ["ir_k3_s2"], ["ir_k5_e1"], ["skip"],
            ["ir_k3_s2"], ["skip"], ["skip"], ["skip"],
            ["ir_k3_e3"],
            ],
            "block_cfg": {
                "first": [16, 2],
                "stages": [
                    [[1, 16, 1, 1]],                                                  # stage 1
                    [[6, 24, 1, 2]],  [[3, 24, 1, 1]],      [[1, 24, 1, 1]],  [[1, 24, 1, 1]],  # stage 2
                    [[1, 32, 1, 2]],  [[6, 32, 1, 1]],      [[3, 32, 1, 1]],  [[1, 32, 1, 1]],  # stage 3
                    [[1, 64, 1, 2]],  [[6, 64, 1, 1]],      [[3, 64, 1, 1]],  [[3, 64, 1, 1]],  # stage 4
                    [[1, 112, 1, 1]], [[1, 112, 1, 1]],     [[1, 112, 1, 1]], [[1, 112, 1, 1]], # stage 5
                    [[1, 184, 1, 2]], [[1, 184, 1, 1]],     [[1, 184, 1, 1]], [[1, 184, 1, 1]], # stage 6
                    [[3, 352, 1, 1]],                                                 # stage 7
                ],
                "backbone": [num for num in range(23)],
            },
        },
```

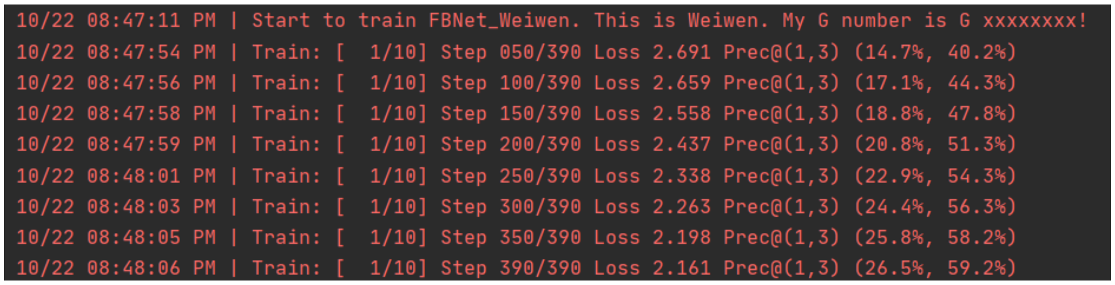
[Note: In this experiment, there is no quantitative experimental result to show. So I skip this part. And it is allowed.]

3. *Training the sampled neural architecture*

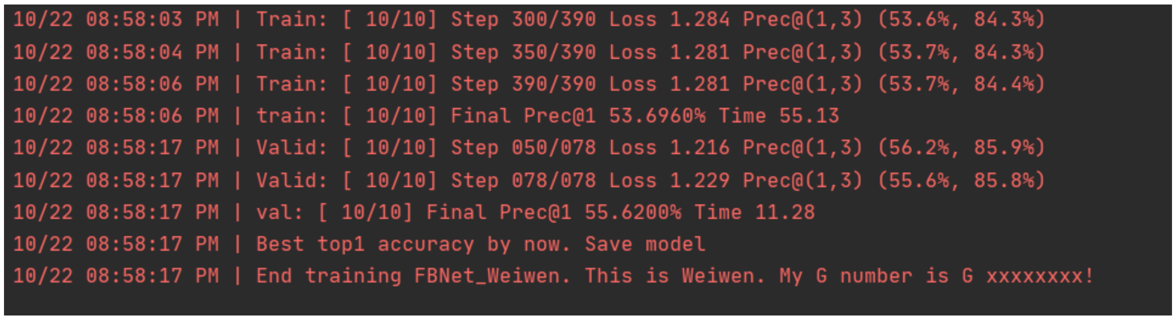
   A. Screenshot of Running

   As you can see in the following two figures, the sampled neural networks (i.e., FBNet_Weiwen) is trained. And the first line of the first figure and the last line of the second figure is added by me. It shows the date when I runed the code and also be marked with my personal information.

   [Although it is not mandatory, it is encouraged that you could add some print information defined by you at each critical step of the code, which could further prove that your run the code by your own and you have read the code and are able to slightly modify the code by yourself.]



```
10/22 08:47:11 PM | Start to train FBNet_Weiwen. This is Weiwen. My G number is G xxxxxxxx!
10/22 08:47:54 PM | Train: [  1/10] Step 050/390 Loss 2.691 Prec@(1,3) (14.7%, 40.2%)
10/22 08:47:56 PM | Train: [  1/10] Step 100/390 Loss 2.659 Prec@(1,3) (17.1%, 44.3%)
10/22 08:47:58 PM | Train: [  1/10] Step 150/390 Loss 2.558 Prec@(1,3) (18.8%, 47.8%)
10/22 08:47:59 PM | Train: [  1/10] Step 200/390 Loss 2.437 Prec@(1,3) (20.8%, 51.3%)
10/22 08:48:01 PM | Train: [  1/10] Step 250/390 Loss 2.338 Prec@(1,3) (22.9%, 54.3%)
10/22 08:48:03 PM | Train: [  1/10] Step 300/390 Loss 2.263 Prec@(1,3) (24.4%, 56.3%)
10/22 08:48:05 PM | Train: [  1/10] Step 350/390 Loss 2.198 Prec@(1,3) (25.8%, 58.2%)
10/22 08:48:06 PM | Train: [  1/10] Step 390/390 Loss 2.161 Prec@(1,3) (26.5%, 59.2%)
```

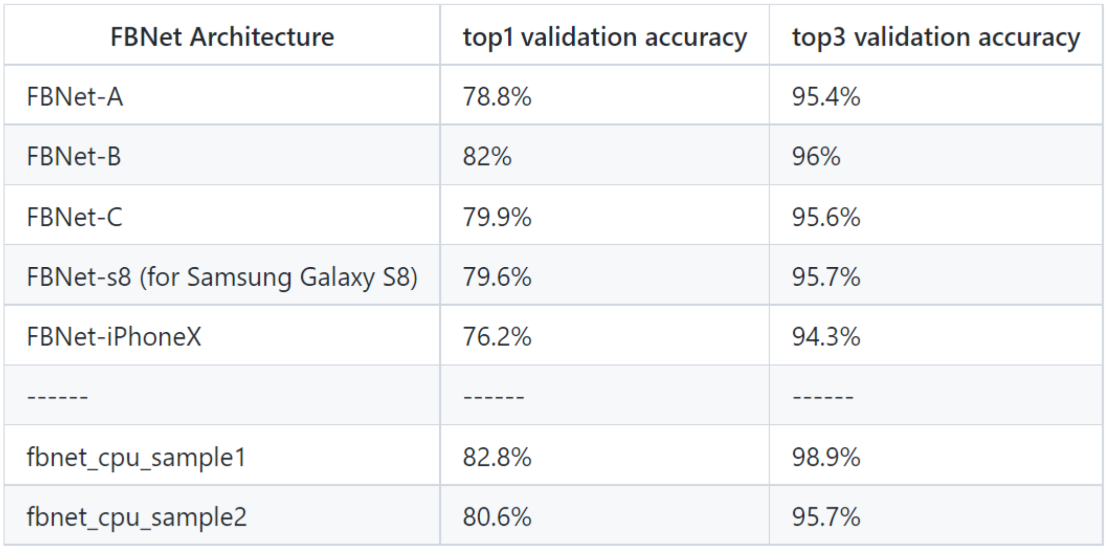


```
10/22 08:58:03 PM | Train: [ 10/10] Step 300/390 Loss 1.284 Prec@(1,3) (53.6%, 84.3%)
10/22 08:58:04 PM | Train: [ 10/10] Step 350/390 Loss 1.281 Prec@(1,3) (53.7%, 84.3%)
10/22 08:58:06 PM | Train: [ 10/10] Step 390/390 Loss 1.281 Prec@(1,3) (53.7%, 84.4%)
10/22 08:58:06 PM | train: [ 10/10] Final Prec@1 53.6960% Time 55.13
10/22 08:58:17 PM | Valid: [ 10/10] Step 050/078 Loss 1.216 Prec@(1,3) (56.2%, 85.9%)
10/22 08:58:17 PM | Valid: [ 10/10] Step 078/078 Loss 1.229 Prec@(1,3) (55.6%, 85.8%)
10/22 08:58:17 PM | val: [ 10/10] Final Prec@1 55.6200% Time 11.28
10/22 08:58:17 PM | Best top1 accuracy by now. Save model
10/22 08:58:17 PM | End training FBNet_Weiwen. This is Weiwen. My G number is G xxxxxxxx!
```

B. Experimental Results

I got a FBNet_Weiwen with a top-1 accuracy of 55.32% on the test set of CIFAR-10. The FBNet_Weiwen is trained with learning rate = 0.01, batch size = 128, the total epoch for training = 3, …. (you could add more training parameters). Due to the time limit and lack of powerful GPUs, I trained the supernet with total of 3 epochs. The low accuracy of the FBNet_Weiwen might be caused by this case.

Besides, I also trained other predefined neural architecture provided by the source code. The experimental results are shown in the following tables.



FBNet Architecture	top1 validation accuracy	top3 validation accuracy
FBNet-A	78.8%	95.4%
FBNet-B	82%	96%
FBNet-C	79.9%	95.6%
FBNet-s8 (for Samsung Galaxy S8)	79.6%	95.7%
FBNet-iPhoneX	76.2%	94.3%
------	------	------
fbnet_cpu_sample1	82.8%	98.9%
fbnet_cpu_sample2	80.6%	95.7%

[This is just a simple example. You should have more experimental results in your final project. It is recommended to use charts and tables to show your experimental results.]

[Note: It is acceptable if your experimental results are not as good as the ones reported in paper due to the lack of a powerful GPU. You can write such problem in your report and be honest with experimental results you got. You will not lose points due to the bad results if it is caused by the lack of GPU resources for training.]

Conclusion

Briefly summarize what you did in this project and the important conclusion your get from the experimental results you got. It is also encouraged to list the lessons you learned and the interesting results you observed through the final project and.

Reference

List all the material you refer to in this section, including the original paper, the link of source code you use and some tutorial that might be helpful for you to get the main ideas of the paper.

Grading criterion

Successful presentation + Complete report with at least all the mentioned sections and appropriate description and explanations within the sections -> 70-75%

The other 25-30 points are based on your performance on presentation, the effort you make in the implementation of your project and the quality of your report. It is acceptable that you only run the open-sourced code correctly and reproduce the experiments in the paper. But we encourage you to modify the code to do some more interesting things or applied them to some other applications/datasets. And of course, if you make such explorations, you are more likely to get a better grade for the final project.

Some suggestions for the modification

1. Try different dataset. For example, NAS on MNIST, FASHION-MNIST, SVHN or ImageNet-10, 100, 1K
2. Try different search space. You can design the macro-architecture, the architecture of candidate operations on your own.
3. Target for different goals or applications. In this code, the trade-off (loss function) is between accuracy and latency on the computers. You can try a different goal, for example, the trade-off between the accuracy and the model size or the trade-off between the accuracy and the FLOPs (floating point operations) of model.
4. Based on the paper and the topic you choose, we encourage you to try any type of modification on the original open-sourced codes, which is not limited to the types we mentioned above. It would be better if you can implement the algorithms by yourself. Anyway, your effort on the implementation or the modification will be appreciated and reflected on the grades of the final project.