

Report 02

Yiming Bian

May 6th 2022

Major Professor

Arun Somani(arun@iastate.edu)

Committee Members

Henry Duwe (duwe@iastate.edu)
Aditya Ramamoorthy (adityar@iastate.edu)
Alexander Stoytchev (alexs@iastate.edu)
Cindy Yu (cindy@iastate.edu)

1 Introduction

In this report, I will show you the issues I met when switching the workspace to Nova cluster and how I fixed those issues. You can share with your students if needed. Then I will present the experiment plan and explain what the purpose of each experiment.

2 Problems and fixes

2.1 “No module”

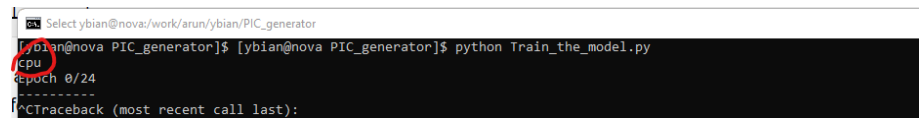
A module missing issue is common, especially when logging into cluster at the first time. One can get a list of available modules on Nova by passing command `module spider` and load the target module with `module load MODULE_NAME`. For example, to load Python 3.x (sometimes the default version is 2.7), one can pass the command `module load python/3.7.7-dwjomwi`.

When the missing module is a Python module, one can first load the `pip` module, then download the Python module using `pip install MODULE_NAME`. If there are multiple Python modules to install, one can create a `requirement.txt` and use command `pip install -r requirement.txt`. A more detailed tutorial is available [here](#).

2.2 GPU not enabled issue

When training a model, using GPU is the preferred choice over CPU on the cluster. However, running the code directly on cluster, GPU is not invoked automatically as shown in Fig. 1. One should use `salloc` to start a GPU session. For example, passing the command `salloc -N1 -n1 -p=gpu -t 5:00:00 --gres=gpu:0` specifies number of GPU, time etc of a session. Once entering the session, one can use GPU to train the network. And do not forget to terminate the session by `exit` if GPU is no longer needed. One important point to stress is that you need to make sure the Python version is consistent. Thus, the version in GPU session should be identical to that in the normal session. Otherwise, the module missing issue may not be solved.

Another way to use GPU to train out model is to submit the job using Slurm and have the task run in the background. First, we need to generate a script that specify what we want to run. There is an official Slurm script generator available [here](#). To import the modules installed by `pip` before, one should load the `pip` module in the script.



```
Select ybian@nova:/work/aron/ybian/PIC_generator
[ybian@nova PIC_generator]$ [ybian@nova PIC_generator]$ python Train_the_model.py
cpu
epoch 0/24
-----
Traceback (most recent call last):
```

Figure 1: No GPU issue

2.3 Other issues

Two issues that have a great chance to happen in the near future are 1) Slurm job out of memory 2) Slurm job time out.

For the first problem, I plan to email HPC help and learn how to calculate the memory size needed for a job together with the out of memory fix. For the second problem, I plan to modify the code and set some check points during training. There is an official tutorial on how to load and continue training the model from the check point. It should be relatively easy to fix.

3 Experiments and purposes

The following experiments will test all of the following networks: AlexNet, GoogleNet, VGG16, ResNet18, SqueezeNet. The common setups for these experiments are

- Noise type: Salt and pepper noise
- Noise levels: 0.1, 0.2, 0.3, 0.4
- Number of objects(e.g. output dimension): 5

- Objects categories(# original samples): goldfinch(1,300), hamster(1,300), frying pan(1,222), upright(1,300), pitcher(1,300)
- Baseline model: **RtO**, trained on original images only

3.1 Models trained with single level of noise data

First, we generate four levels of noise data based on original images and categorize them into four datasets: SNP_lvl.1, SNP_lvl.2, SNP_lvl.3 and SNP_lvl.4. Each dataset is splitted into a training set(80%), a validation set(10%) and a test set(10%). Then we fine-tune the pre-trained model into four re-trained models: **RtN_1**, **RtN_2**, **RtN_3** and **RtN_4**. Each trained with a single level of noise dataset constructed in the last step. With these re-trained models, we can explore the answers to the following two questions.

3.1.1 Trained on less noise data, test on more noise data

Question 1: If a model is trained with low level of noise data, what’s the prediction accuracy on higher level of noise data?

To answer this question, we can test the accuracy of the model on higher level of noise data. For example, test model **RtN_1** on the test set of SNP_lvl.2, SNP_lvl.3 and SNP_lvl.4. All results will be recorded in the Table 1.

An intuitive guess will be that the performance decreases as the noise level increases.

Table 1: Test accuracy of each model on various test sets

model \ test set	RtO	RtN_1	RtN_2	RtN_3
Original	NA	NA	NA	NA
SNP_lvl.1	??.??%	NA	NA	NA
SNP_lvl.2	??.??%	??.??%	NA	NA
SNP_lvl.3	??.??%	??.??%	??.??%	NA
SNP_lvl.4	??.??%	??.??%	??.??%	??.??%

3.1.2 Trained on more noise data, test on less noise data

Question 2: If a model is trained with high level of noise data, what’s the prediction accuracy on lower level of noise data?

Compared to the previous question, the answer to this question might be anti-intuitive because training on very noisy data may affect the feature extraction and when processing a perfect image, it may not even recognize some key features. Therefore, we would like to do this experiment. Combined with the previous table, we can create a big table like Table 2.

Test accuracy on the diagonal is a baseline. Those figures below the diagonal answer the **Question 1** and those above answer the **Question 2**. Since we are testing five networks, there will be five such tables in total.

Table 2: Complete table of test accuracy of each model on various test sets

model test set	RtO	RtN_1	RtN_2	RtN_3	RtN_4
Original	??.??%	??.??%	??.??%	??.??%	??.??%
SNP_lvl_1	??.??%	??.??%	??.??%	??.??%	??.??%
SNP_lvl_2	??.??%	??.??%	??.??%	??.??%	??.??%
SNP_lvl_3	??.??%	??.??%	??.??%	??.??%	??.??%
SNP_lvl_4	??.??%	??.??%	??.??%	??.??%	??.??%

3.2 Models trained with single kind but multiple levels of noise data

As we add noise to images by calling `imnoise()` in MATLAB, there are five noise options: salt & pepper, Gaussian, Poission, speckle and localvar. We would like to investigate cross-type performance of different re-trained models. They are **Rt_SNP**, **Rt_GS**, **Rt_Pos**, **Rt_Spe** and **Rt_Loc**, each is only trained on corresponding noise type data with multiple levels. For example, model **Rt_SNP** will be trained on Salt & Pepper noise data of noise density 0.1, 0.2, 0.3 and 0.4. Similar to the previous dataset construction, we expand the original image set to m -fold, where m is the number of noise level of a specific noise type. For salt & pepper noise, $m = 4$. The noise dataset will be divided into a training set(80%), a validation set(10%) and a test set(10%). Table 3 collects the test accuracy of each model on all kinds of test sets. It helps to develop conclusions regarding the performance of cross-type noise image detection of each model.

There will be five of such table as we will test five networks.

Table 3: Complete table of test accuracy of each model on various test sets

model test set	RtO	Rt_SNP	Rt_GS	RtN_Pos	RtN_Spe	RtN_Loc
Original	??.??%	??.??%	??.??%	??.??%	??.??%	??.??%
salt&pepper	??.??%	??.??%	??.??%	??.??%	??.??%	??.??%
Gaussian	??.??%	??.??%	??.??%	??.??%	??.??%	??.??%
Poission	??.??%	??.??%	??.??%	??.??%	??.??%	??.??%
speckle	??.??%	??.??%	??.??%	??.??%	??.??%	??.??%
localvar	??.??%	??.??%	??.??%	??.??%	??.??%	??.??%

4 Resource and Rule

Github Repo: [Project Repo](#)

Report Frequency: Every two weeks

Next Report: May 20, 2022