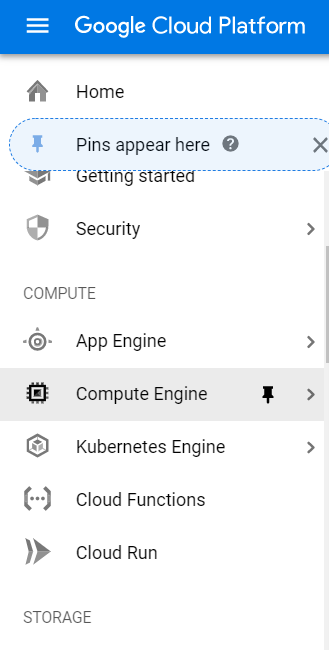
GCP GUIDE

Go to GCP website, click on top left button > Compute Engine

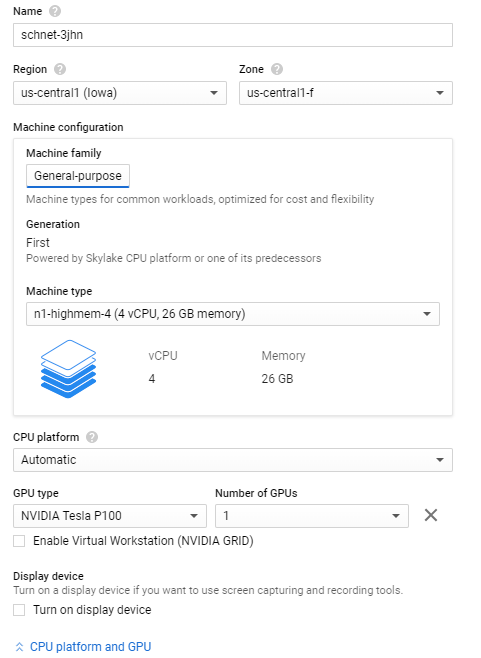


Create new instance. 1 instance = 1 training session in 1 machine. Use the specs below. Note: please use P100 only!!! (V100 is causing troubles). Can change the Region and Zone freely, as long as that region has P100.

One more note: we cannot use more than one P100 in the same region. So for your next instance, please change to any region with P100.

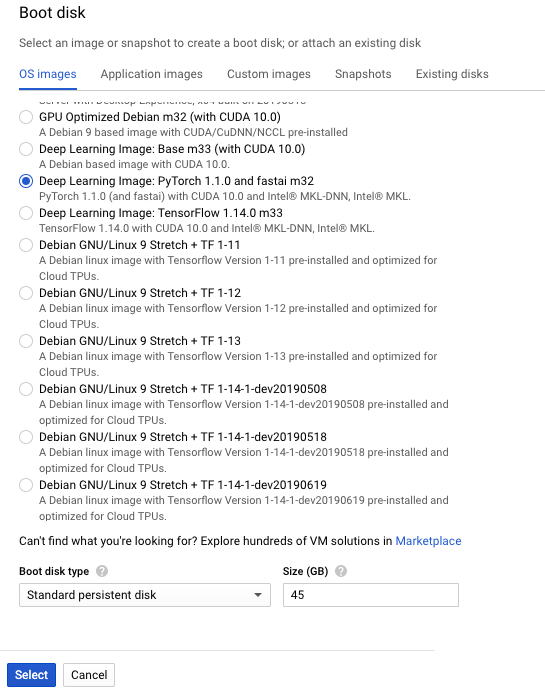
And since CPU quota is often 32, we should use only 4vCPU or 8vCPU per instance only. I recommend 4, then you can have 8 instances at the same time.

For Yves: for 8 instances, we need to re-train 2JHN and 3JHN (8 folds each). So 4 instances for each type, and 2 folds in each instance. You will understand, continue to read.

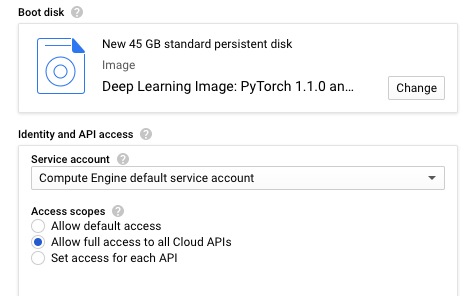


Choose a disk image. For the first instance, use prebuilt image as below. After manually install all schnet packages in this first instance, we then can take a “snapshot” of this 1st instance (will instruct later), so for the 2nd instance onwards, you can directly load this image by clicking tab “Snapshots” (instead of “OS image” , as shown in this 1st instance).

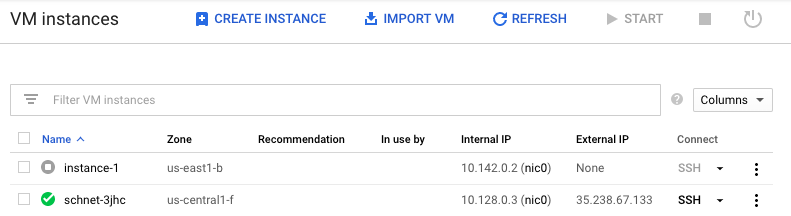
Choose the OS image as in the picture.



Remember to specify Size (GB) of disk to 45 GB (above picture), and Allow full access…. (below picture).



Then leave everything else as default, and Create the instance.



Open the SSH window from that instance (click SSH small button in the image above). It will open a terminal (the 1st time the terminal will ask something, just “y” all).

Then install the required packages by sequentially run each of the below line in the terminal. Some packages may take a few minutes to install.

pip install torch-scatter --user

pip install torch\_geometric --user

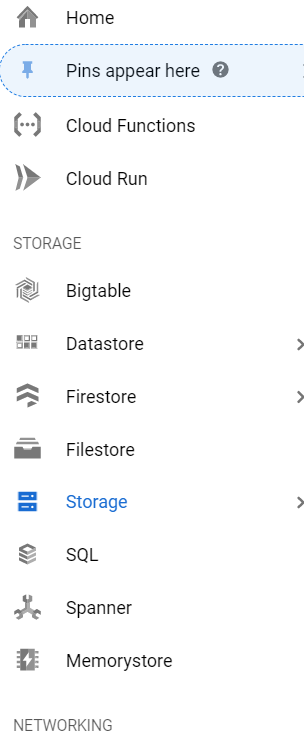
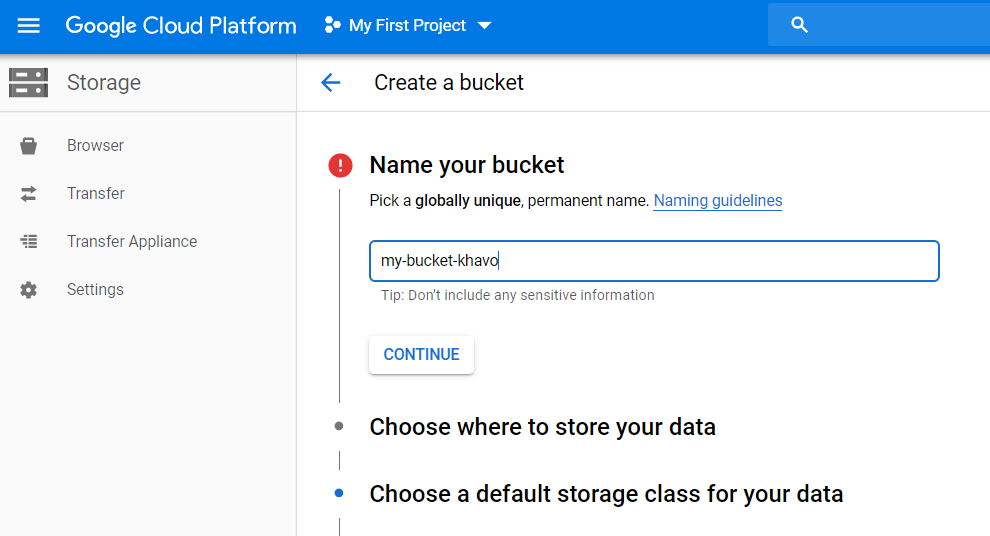
pip install torch\_sparse --user

pip install torch\_cluster --user

pip install -U ase==3.17.0 --user

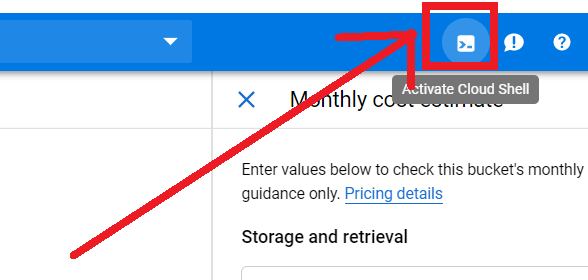
pip install schnetpack --user

Next, import the “csc” folder to your instance. Create a “bucket” (like a folder on gcp) by going to the “Storage” section on GCP console 🡪 Create Bucket



Create your bucket with custom name like that.

After creating it, open the general GCP console by clicking the icon



Copy my shared csc folder from my bucket to your bucket by running this in the gcp console:

gsutil copy -r gs://schnet-code/csc gs://your-bucket

OK, now move back to your instance ssh terminal, copy csc folder from your bucket to your instance by:

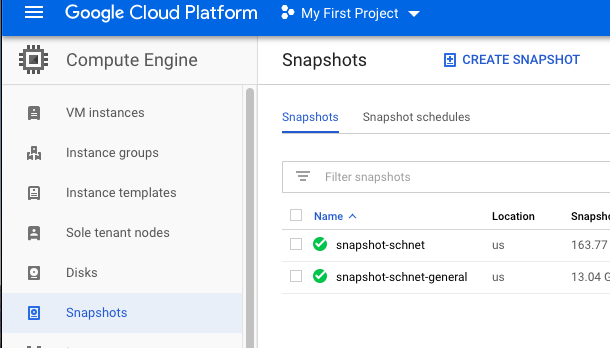
gsutil copy -r gs://your-bucket/csc .

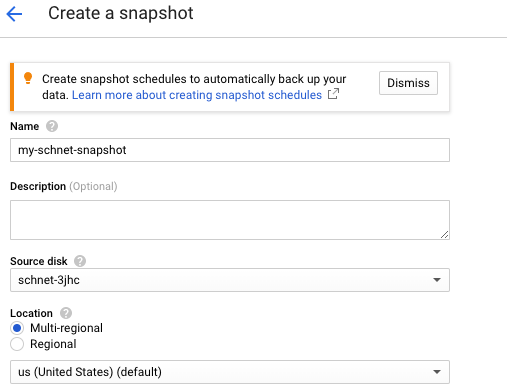
(if you face any error, try to add/remove “/” at the end of the path, and remember “.” at the end)

If your instance contains csc folder, it’s successful (check by run “ls” in the instance ssh window to list the files and folder).

Now, all basic installation general for all types is complete. We need to make a “snapshot” of this disk image, then for next instance, we don’t need to do the same installation again.

To do so, click “CREATE SNAPSHOT” in the Snapshots section.





Then create it.

The next instance you create, you can boot from this snapshot, and skip all steps above.

OK, Now move ahead to edit the kfold seed and fold index to train.

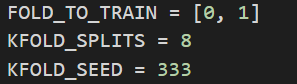
EACH TYPE TRAINING

For a specific coupling type training, we need to copy the .db file (training file) of the type we want, from my shared folder, to this instance. Do this by running in the ssh terminal:

gsutil cp gs://schnet\_input/db/CHAMPS\_train\_3JHC-np.pkl .

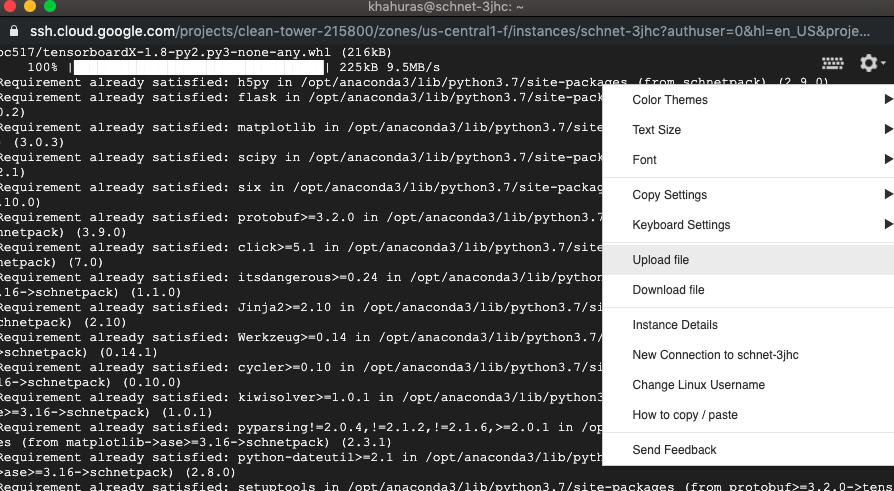
Remember the “.” at the end, change the type you want, and don’t copy more than 1 db file other than the type you want to train, to save disk space.

Modify the training script of that type on your local computer (the scripts name yves\_2JHN\_train.py and yves\_3JHN\_train.py I uploaded on googledrive) then adjust the fold you want to train.

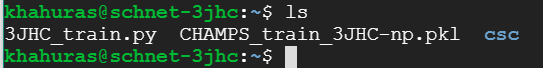


Now, I have specified the 8-fold already. Please train 2 folds in each instance (either 2JHN or 3JHN). Each instance will have FOLD\_TO\_TRAIN as [0,1] or [2,3], or [4,5], or [6,7].

Then upload this script to your instance by click the gear icon on top right of ssh window.



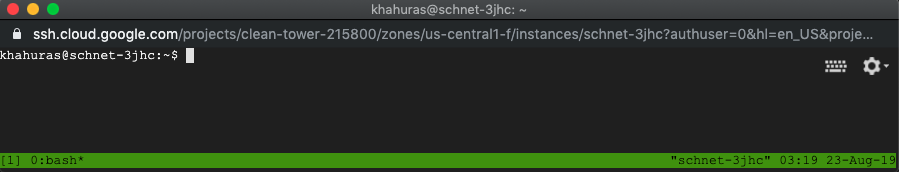
Check if everything is fine, run “ls”. Then we should have 3 things: csc, db file, and script file like this.



Everything is done. We can begin training.

TRAINING

Now is the most important step. Run “tmux” in the ssh terminal. Then the green bar must be present like this

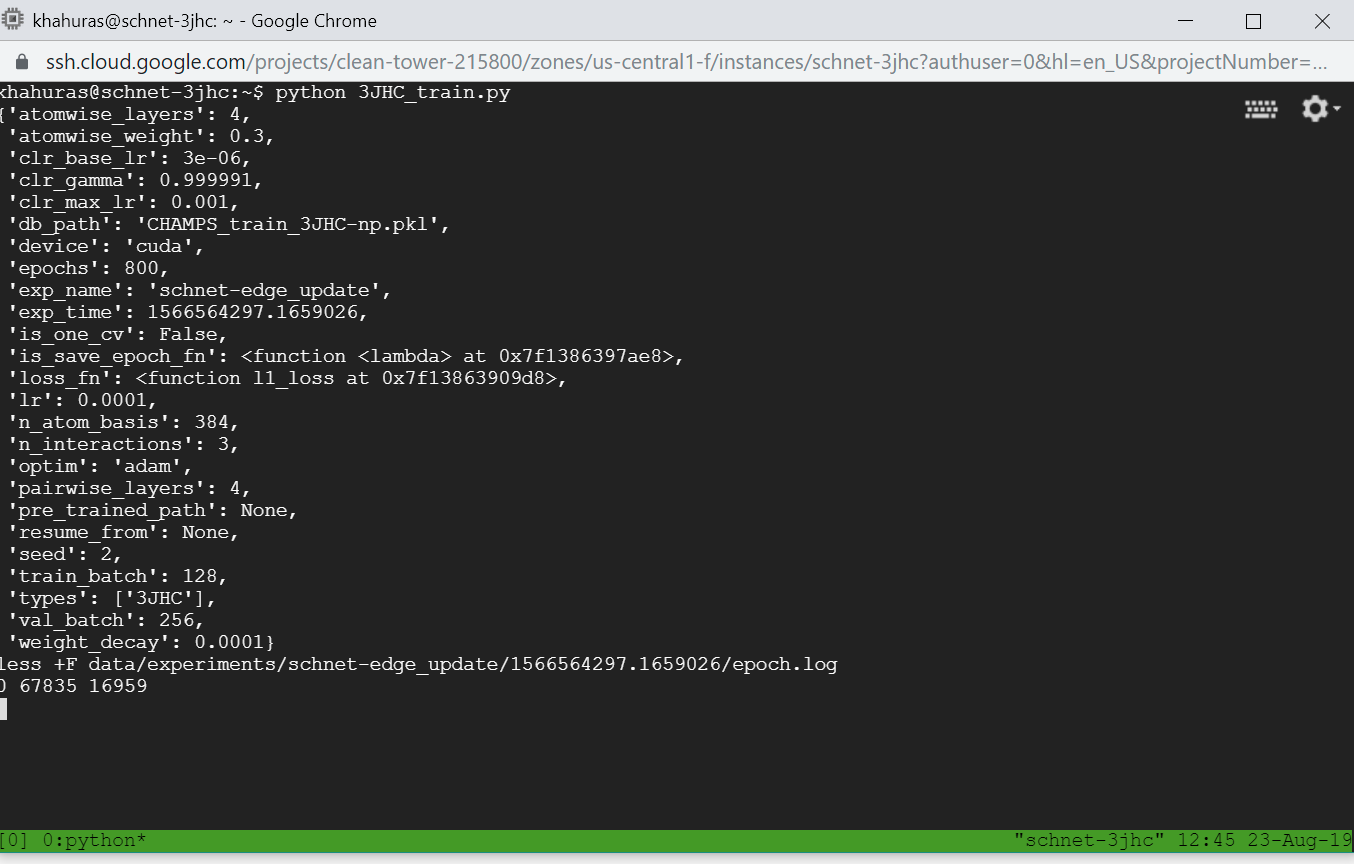


The reason to run “tmux”, is that it will create a 3rd server, that will always be present in background and serves as our local computer (though it’s not), and always connects to this GCP instance. Then for any training command in this green tmux window, after running it, if local internet interruption occurs, it won’t break the training process as managed by tmux already. If we don’t run “tmux” before training and we lose that ssh window, the training will be terminated.

Then, begin training by

“python yves\_3JHN\_train.py”

If you have as below picture, the session is training successfully. Wait for a few minutes, if it does not produce any error, then you are successful.

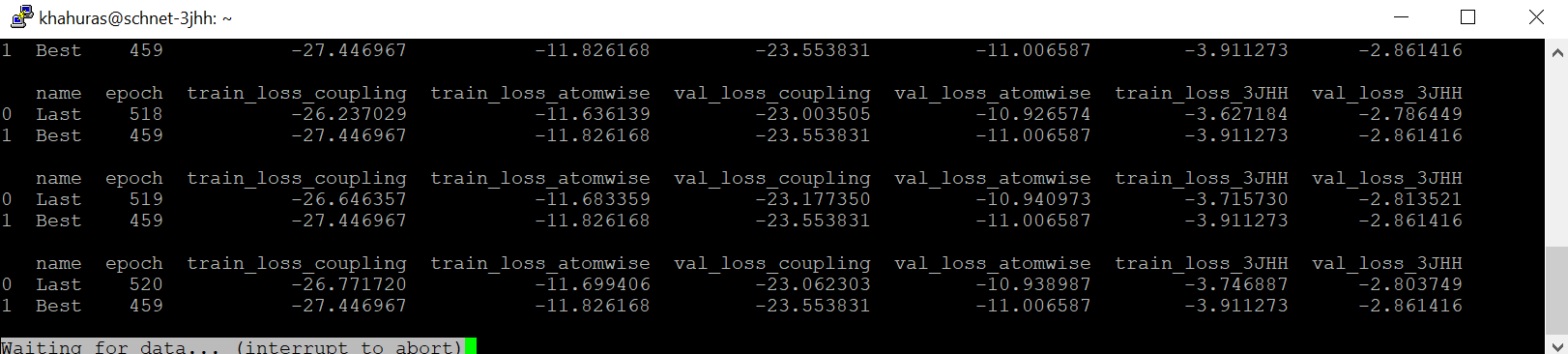


Copy the “less +F ….” command to somewhere locally, we will need it later. By dragging it, the dragged text will be auto-copied to the clipboard (please don’t press Ctrl+C, the tmux process may be terminated)

Then, you can safely close this ssh terminal. (yes, close it).

To track the training log, open another ssh terminal again (by clicking SSH in GCP instance website)

Then **don’t** run “tmux” anymore, just simply paste the “less +F…” command and run. The log is displayed.



This second ssh window has nothing to do with the training (just reading from the log file produced from tmux in background), hence we can close it and reopen it anytime we want. Just don’t touch the tmux session too much.

In case you forgot the “less” command, in a new ssh terminal, run “tmux attach”, the tmux window with green bar as above will pop up again. Then for safety, let’s close it asap.

Sometimes, go to instance website and track the CPU usage. If it drops like this, the process is killed by some reason. We need to re-lodge the instance.

