Deep Learning PhD Wiki

Guocheng Qian

Feb 19, 2021

CONTENTS:

Deep Learning PhD Wiki is a Wiki doc hosted by Guocheng Qian for logging his personal study in Deep Learning (especially on Computer Vision). This wiki aims at aggregating resources and sharing experience in Deep Learning and helping the newbies learn deep learning from zero.

Deep Learning PhD Wiki also shares other experience in my journey towards a qualified PhD graduate (like building personal homepage, reading papers, presentations, *etc.*)

CHAPTER

ONE

INSTALL A DEEP-LEARNING-MACHINE-ENVIRONMENT ON UBUNTU

1.1 Dependencies Related

- CUDA: for accelerated computing
- anaconda: tool for environment management
- jupyter lab: develop open-source software, open-standards, and services for interactive computing
- git: manage your code in a diligently

1.2 CUDA installation

You may want to see the detailed blog. Take CUDA 10.1 installation for example.

1. Download the CUDA 10.1.243 Ubuntu 18.04 source.
2. install:

   ```
   sudo sh cuda_10.1.243_418.87.00_linux.run
   ```

3. Set the path to *~/.bashrc*:

   ```
   # for cuda path
   cuda=cuda-10.1
   export PATH=/usr/local/$cuda/bin:$PATH
   export CUDADIR=/usr/local/$cuda
   export NUMBAPRO_NVVM=$CUDADIR/nvvm/lib64/libnvvm.so
   export NUMBAPRO_LIBDEVICE=$CUDADIR/nvvm/libdevice/
   export NVCCDIR=$CUDADIR/bin/nvcc
   export LD_LIBRARY_PATH=/usr/local/$cuda/lib64:$LD_LIBRARY_PATH
   export CPATH=/usr/local/$cuda/include:$CPATH
   export CUDA_HOME=/usr/local/$cuda
   # for CUDA_DEVICES
   export CUDA_DEVICE_ORDER=PCI_BUS_ID
   ```

1.3 Anaconda

1.3.1 Install anaconda

Download the anaconda individual version from Anaconda official website.

1. Install: *bash Anaconda3-xxx-xxx.sh*, e.g.: `bash Anaconda3-2020.07-Linux-x86_64.sh`. Suggest keeping the default settings
2. make sure to add anaconda path to `~/.bashrc`. If you forgot to do so, just add the below into bashrc:

```
# >>> conda initialize >>>
# !! Contents within this block are managed by 'conda init' !!
__conda_setup="$('/home/qiang/anaconda3/bin/conda' 'shell.bash' 'hook' 2>␣
↪/dev/null)"
if [ $? -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "/home/qiang/anaconda3/etc/profile.d/conda.sh" ]; then
        . "/home/qiang/anaconda3/etc/profile.d/conda.sh"
    else
        export PATH="/home/qiang/anaconda3/bin:$PATH"
    fi
fi
unset __conda_setup
# <<< conda initialize <<<
```

   Note: Please change the `/home/qiang/anaconda3` to your own path to anaconda3.

1.3.2 Manage env using Anaconda

Here is how install a specific environment for one project. (deepgcn_env_install.sh, find this file here:

```
#!/usr/bin/env bash

# make sure command is : source deepgcn_env_install.sh

# uncomment if using cluster
# module purge
# module load gcc
# module load cuda/10.1.105

# make sure your annaconda3 is added to bashrc (normally add to bashrc path␣
↪automatically)
source ~/.bashrc

conda create -n deepgcn # conda create env
conda activate deepgcn  # activate

# conda install and pip install
conda install -y pytorch torchvision cudatoolkit=10.0 tensorflow python=3.7 -
↪c pytorch
# install useful modules
pip install tqdm
```

Install the env above by: `source deepgcn_env_install.sh`. Now you install the new env called deepgcn, `conda activate deepgcn` and have fun! Take a look at the official guide how to use anaconda.

1.3.3 Jupyter Lab

Jupyter lab: Jupyter exists to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages. It's automatically installed when you install anaconda3. You have to add conda env to jupyter lab manually by code below.

```
conda activate myenv
python -m ipykernel install --user --name myenv --display-name "Python (myenv)"
```

You may have to install `ipykernel` at first: `conda install ipykernel`.

Jupyter lab supports using the remote environment to run the kernel and render in the local browser. Support! Sometimes, we may need to run jupyter lab on our laptop but use the hardware and env of remote workstation. How to do that? Open one terminal in your laptop, then open jupyter lab by code below.

```
ssh remoteAccount@eremoteIp # connect remote server
# jupyter notebook password # uncomment if you have not set password (do it once)
jupyter lab --port=9000 --no-browser &
```

Open another terminal in your laptop, then map ip by code below:

```
ssh -N -f -L 8888:localhost:9000 remoteAccount@eremoteIp
```

Now open your chrome, type: `http://localhost:8888/`. Enjoy your remote jupyter lab.

You can kill the port forwarding by:

```
ps aux | grep ssh
kill <id>
```

More info see blog

1.4 Git Support (GitHub)

Using `git` command to pull, push and manage your code. Here is an introduction to git . CheatSheet for `git` is here.

GitHub is the largest code sharing, management and version control platform. You may have to add `ssh` to github, otherwise each time you use git command, you have to input your account information. Here is the instruction.

Set your git global username and email address. This is to let Git know who you are. (If you do not change this, you can still git pull and git push as along as you add your ssh into github. However, github will not be able to appreciate your commits in commit history, they will think it is someone else make the changes not you.) To set the username and email:

```
git config --global user.name "FIRST_NAME LAST_NAME"
git config --global user.email "MY_NAME@example.com"
```

1.5 Terminal Related Tools

- Terminator: useful tool for arranging terminals
- Termius: SSH client that works on Desktop and Mobile for connecting to local and remote machines.
- tmux: tools for multiple windows in terminal. Very useful for working with remote machines. The tmux will keep your job running in the background even if you lose you connection with the remote machines. tmux cheatsheet.
- vim: the best command line editor. vim cheatsheet
- rclone: command tools for downloading and pushing files from or to google drive. See here for more info.
- aria2: a lightweight multi-protocol & multi-source command-line download utility. Here is the documentation. Here is an example (download imagenet using 16 threads and set the `continue=true` which resumes the downloading): `aria2c -c -x 16 -s 16 -d imagenet http://image-net.org/challenges/LSVRC/2012/dd31405981ef5f776aa17412e1f0c112/ILSVRC2012_img_train.tar`
- Markdown: a lightweight markup language with plain-text-formatting syntax. Markdown cheatsheet

1.6 Software

- PyCharm: my favorite IDE for Python. Professional version is free for students
- Clion: my favorite IDE for C and C++. Professional version is free for students
- MeshLab: my favorite 3D viewer.
- Synergy: share one mouse and keyboard between multiple computers (Linux, Mac, Windows).

# Linux 1. counting ls -l | wc -l 2. download wget -r -p -np -k 3. count storage usage du –max-depth=1 -h 4. show modified time stat -c '%y : %n' ./* 5. watch gpu usage watch -n 0.1 nvidia-smi

CHAPTER TWO

LEARN PYTORCH

2.1 Pitfall in Python

1. Mutable and immutable data types:

   In Python, data types can be either mutable (changeable) or immutable (unchangable). And while most of the data types in Python are immutable (including integers, floats, strings, Booleans, and tuples), `lists and dictionaries are mutable`. That means `a global list or dictionary (mutable datatypes) can be changed even when it’s used inside of a function.`

   If a data type is immutable, it means it can’t be updated once it’s been created. In Pytorch, all tensor operations are immutable. e.g.:

```
initial_list = [1, 2, 3]
def duplicate_last(a_list):
    last_element = a_list[-1]
    a_list.append(last_element)
    return a_list

new_list = duplicate_last(a_list = initial_list)
print(new_list)
print(initial_list)
[1, 2, 3, 3]
[1, 2, 3, 3]
```

   As we can see, here the global value of initial_list was updated, even though its value was only changed inside the function!

   Because of the mutable characteristics of list and dictionary, we usually use it to save the imortant middle results (like accuracy, metrics, args).

2.

2.1.1 Some advanced operations

1. Change layers in pretrained models `model.conv1[0] = new_model.conv1[0]`
2. detach some modules

```
for param in model.conv1.parameters():
    param.requres_grad = False
for k, param in model.named_parameters():
    print(k, param.requires_grad)
```

2.2 Suggested Pytorch Libraries

2.2.1 general

- wandb: Experiment tracking, hyperparameter optimization, model and dataset versioning.
- hydra: A framework for elegantly configuring complex applications.
- PyTorch Metric Learning: The easiest way to use deep metric learning in your application. Modular, flexible, and extensible. Like triplet loss support.
- TNT: torchnet(TNT) is a library providing powerful dataloading, logging and visualization utilities for Python. It is closely integrated with PyTorch and is designed to enable rapid iteration with any model or training regimen.

2.2.2 3D

- Pytorch-Geometric: Geometric Deep Learning Extension Library for PyTorch
- torch-points-kernels: Pytorch kernels for spatial operations on point clouds
- torch-points3d: Pytorch framework for doing deep learning on point clouds.
- pytorch3d: PyTorch3D is FAIR's library of reusable components for deep learning with 3D data.

CHAPTER

THREE

HOW TO USE IBEX

IBEX is the GPU cluster hosted in KAUST. IBEX is managed using Slurm. This Documentation teachs you how to use Slurm GPU cluster. IBEX is just an example. You can use what mentions in this doc in any Slurm based cluster (like the node center in IVUL-KAUST, the GPU clusters in most companies).

3.1 Termius

Recall that when you connect to a cluster (or a remote machine), you have to `ssh yourAccount@ipAddress`, and then input your password. This is annoying because you have to do it everytime when you want to login in the cluster.

To avoid inputing username and password every time, I highly recommend a terminal software called termius to all of you.

After installing termius, simply add the host (the your username in termius: add address of the cluster or any remote machine -> click ssh -> add username and password. Note: the address of the host can be the actual IP or the ip label, like in IBEX, you can use `glogin.kaust.edu.sa` or the actual IP Address, I recommend using the IP address)

After adding the host, whenever you want to login in host, you just need to click the host name

For example, setting a host named (glogin) for logging gloin in IBEX:

- address: 10.109.66.127 (or, you can use `glogin.ibex.kaust.edu.sa`)
- username: qiang (this is my username, please use yours)
- password: xxxxxxxx

After setting up, only a click on the glogin button in termius is needed to ssh into the ibex system.

3.2 Tmux

`tmux` is very helpful if you want to connect to a remote machine. If you do not use `tmux`, you will lose your connection if your terminal somehow disconnects with your remote machine (this situation happens a lot). tmux is easy to use. You only have to `tmux new -s xxx` to create a new tmux window. You can then login into the window using `tmux a -t xxx`. Then even if you lose your connection with the remote machine, you can re-login, and the tmux is always there. You can use tmux to login into that windown, and you will find everything is continues running there without being affected by the disconnection. There is a tmux cheatsheet. Here is the tmux wiki.

3.3 Run a Job in cluster (IBEX)

In any Slurm based cluster, whenever you want to use any resources (node, CPU, GPU, Mem, and running Time), you should ask a request for the resources.

You can either use `sbatch` or `srun` to request for resources to run your program in cluster.

1. **sbatch** The first option to run a program (eg, train or evaluate a deep learning model) is to use sbatch to send your job. Sbatch send your job in the priority queue and your code will continue to run even if you lose connection with cluster for some reason.

   There is an example of sbatch file:

```
#!/bin/bash --login
#SBATCH -N 1
##SBATCH --array=1-5  # uncomment to repeat the task, \eg, if you want to run␣
↪the same program for 5 times, you can use --array=1-5
#SBATCH -J rloc
#SBATCH -o slurm_log/%x.%3a.%A.out    # make sure the slurm_log folder exists
#SBATCH -e slurm_log/%x.%3a.%A.err
#SBATCH --time=5-0:00:00 # time, you'd better make job less than 24 hours␣
↪therefore you get resources faster. You can split your job into smalle␣
↪chunks if the total time is over 24 hours.
#SBATCH --gpus=8            # or: --gres=gpu:v100:8 to specify the GPUs.
##SBATCH --constraint=[rtx2080ti|gtx1080ti|v100] # or use this line to␣
↪specify the GPUS. Note: you either use this line and above line to specify␣
↪GPUs, or you use single line: --gres=gpu:v100:8
#SBATCH --gpus-per-node=8   # use gpu_wide. specify the node, only consider␣
↪node with 8 GPUs. comment out if you do not need this constraint.
#SBATCH --cpus-per-gpu=6 # cps-per-gpu can not be over 6, for fair use
#SBATCH --mem-per-gpu=45G # can not be over 45G for fair use
#SBATCH --mail-user=xxx@kaust.edu.sa    # send message to your email
#SBATCH --mail-type=FAIL,TIME_LIMIT,TIME_LIMIT_90
##SBATCH -A conf-gpu-2020.11.23 # if you have additional QOS (that can give␣
↪you higher priority for requesting resources, you can add it)
#SBATCH --constraint=[ref_32T]  # use shared folder in v100s. this line can␣
↪only be added if you request for 8 v100s node, and you want to use datasets␣
↪in shared folder.

# activate your conda env
echo "Loading anaconda..."

module purge
module load gcc
module load cuda/10.1.105

conda activate deepgcn

echo "...Anaconda env loaded"
python -u examples/classification/train.py  --phase train --train_path /
↪scratch/dragon/intel/lig/guocheng/data/deepgcn/modelnet40
echo "...training function Done"
```

   Run `sbatch train_ibex.sh` then your job will be put in the queue. You can check your queue info by : `squeue -u yourusername`. You can check your job output and error through the file `slurm_log/%x.%3a.%A.out` and `slurm_log/%x.%3a.%A.err`.

   You can check the available GPUs by: `gpu-usage --nodes|grep v100`

See KAUST IBEX offical doc for detailed information. See the IBEX Best Practice for the detailed configuration of best usage on IBEX.

2. **srun** srun allow you to use cluster just like in terminal on your local machine. This is very useful when you want to debug your code. srun is convenient to use, however it will stop run when you lose connection with ibex. You need tmux to protect the node. When you lose connection, you can use tmux to login back into the node.

   You can also srun into your allocated node using: `srun --jobid=yourjobid --pty bash` To do that, you have to use Sbatch at first to query for resources and start your training there. The srun is just used as a tube. After you srun into the node, you can check your mem usage using `nvidia-smi`, etc.

3.4 Use NodeCenter (IVUL-KAUST)

If you are the member of IVUL-KAUST, you can use the internal GPU cluster belonging to our group called NodeCenter. The usage of the NodeCenter is roughly the same of IBEX. However, you have to login into each node and request resources in each node.

Note: the time limit for a job is 12:00:00, so you must cut your job into small jobs (chunks) if the whole time is over 12 hours.

CHAPTER

FOUR

WORK REMOTELY

4.1 Inside KAUST

If you working remotely inside KAUST, you can easily work as normal if you use terminal, also you can debug your code using PyCharm but using the remote resources, or simply using TeamViewer to connect with your workstation.

4.1.1 Terminal

If you use terminal, nothing changes. For example, you can still use `termius` to ssh into IBEX, and submit jobs as usual. (check **`use_ibex`**_.). If you wanna debug using terminal, you can add *pdb* into your Python code. Also, you can always use GitHub to sync your codes between multiple machines.

4.1.2 PyCharm Remote

If you want to use Pycharm to debug your code, but you do not have GPUs on your local machine. You can use the remote interpreter function of PyCharm.

4.2 Outside KAUST

Note: workstations inside KAUST and IBEX can only be accessible using KAUST Network. if you are outside KAUST, You can:

1. use KAUST VPN to login into KAUST network. (not recommended if you are out of kingdom).
2. use TeamViewer to connect your workstation or laptop in KAUST. Teamviewer can used even if your computer is locked.

CHAPTER

FIVE

PERSONAL WEBSITE

Personal website is important to an academic researcher.

5.1 How to design Your Own Homepage

You can use github pages to host your website for free. Just follow the step, it takes you 30 min than you will enjoy your own pages.

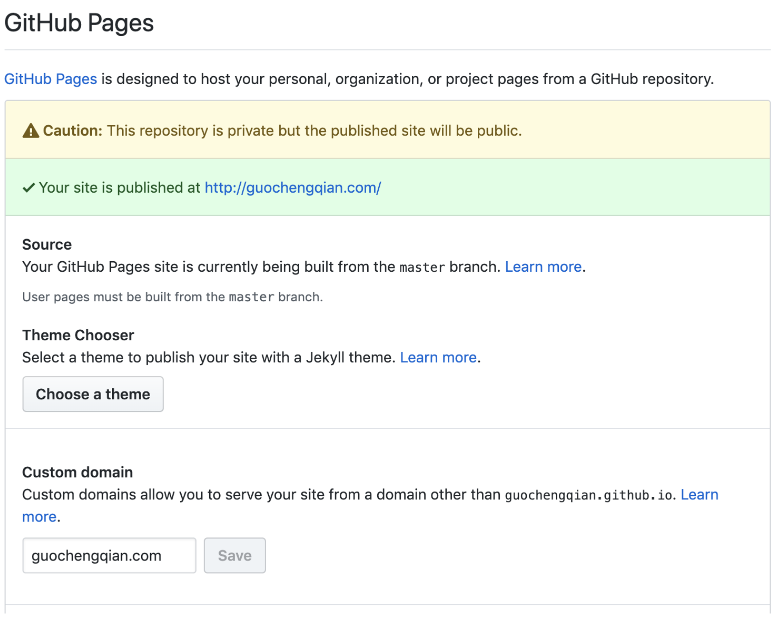
1. Create a github account. Config your git environment. (if you are not familiar with git, please refer to git beginner.)
2. Create a new repository(repo) in github, name rop into `username.github.io` (username is your github account name). (You cannot use other name. This repo is different with others, it is a special repo called github pages. Refer to how to design github pages for details. It may take you 20 mins.)
3. Find a personal homepage that you like and down the source code by : `wget -r -p -np -k http:xxxx.com` Please ask for the author for the approval.
4. Keep and architectrure the same but change content into yours.
5. Put all the source code into github page repo your created before.
6. Git add, commit and push the code.
7. Done! It's so easy. Surf your website username.github.io and enjoy! More information you can looking into github pages or google.

If you want use your own domain like xxx.com instead of the free github.io, please refer to follows.

5.1.1 New domain username.com Setup

We have to buy a new domain and redirect the xxx.github.io to this domain.

1. buy a domain(you can buy from alibaba, tencent, godaddy, name.com)
2. set up dns (please refer to details
3. repo setting (Type your new website in custom domain in repo setting. Like the picture show.


GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

⚠ **Caution:** This repository is private but the published site will be public.

✓ Your site is published at http://guochengqian.com/

Source
Your GitHub Pages site is currently being built from the `master` branch. Learn more.

User pages must be built from the `master` branch.

Theme Chooser
Select a theme to publish your site with a Jekyll theme. Learn more.

Custom domain
Custom domains allow you to serve your site from a domain other than `guochengqian.github.io`. Learn more.

guochengqian.com

4. wait for the new domain to be become effective. (Be patient, it could take as long as 24 hours.)
5. Done! Surf your website username.com and enjoy.

If you have any problem, you can looking into Github Page Redirect. for more details.

5.1.2 Google index

Let baidu, google know your domain, so you can search your website by google

1. **submit url** submit url to baidu, google