

18 | Appendix: Tools for Deep Learning

In this chapter, we will walk you through major tools for deep learning, from introducing Jupyter notebook in [Section 18.1](#) to empowering you training models on Cloud such as Amazon Sagemaker [Section 18.2](#), Amazon EC2 [Section 18.3](#) and Google Colab. Besides, if you would like to purchase your own GPUs, we also note down some practical suggestions in [Section 18.5](#). If you are interested in being a contributor of this book, you may follow the instructions in `.. _sec_how_to_contribute`.

18.1 Using Jupyter

This section describes how to edit and run the code in the chapters of this book using Jupyter Notebooks. Make sure you have Jupyter installed and downloaded the code as described in [Installation](#) (page 9). If you want to know more about Jupyter see the excellent tutorial in the [Documentation](#)²⁶⁵.

18.1.1 Editing and Running the Code Locally

Suppose that the local path of code of the book is “xx/yy/d2l-en/”. Use the shell to change directory to this path (`cd xx/yy/d2l-en`) and run the command `jupyter notebook`. If your browser does not do this automatically, open <http://localhost:8888> and you will see the interface of Jupyter and all the folders containing the code of the book, as shown in [Fig. 18.1.1](#).

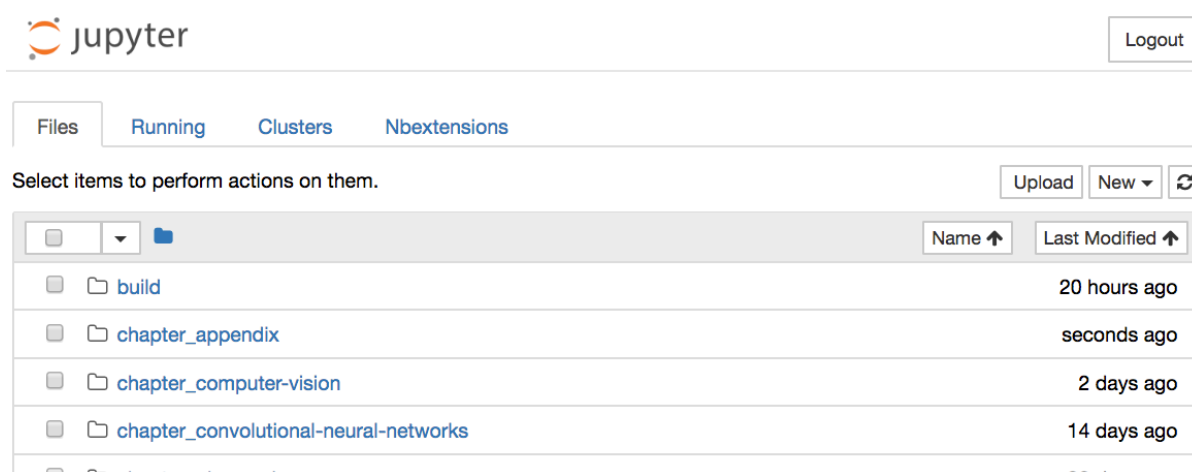


Fig. 18.1.1: The folders containing the code in this book.

²⁶⁵ <https://jupyter.readthedocs.io/en/latest/>

You can access the notebook files by clicking on the folder displayed on the webpage. They usually have the suffix `.ipynb`. For the sake of brevity, we create a temporary “test.ipynb” file. The content displayed after you click it is as shown in Fig. 18.1.2. This notebook includes a markdown cell and a code cell. The content in the markdown cell includes “This is A Title” and “This is text”. The code cell contains two lines of Python code.

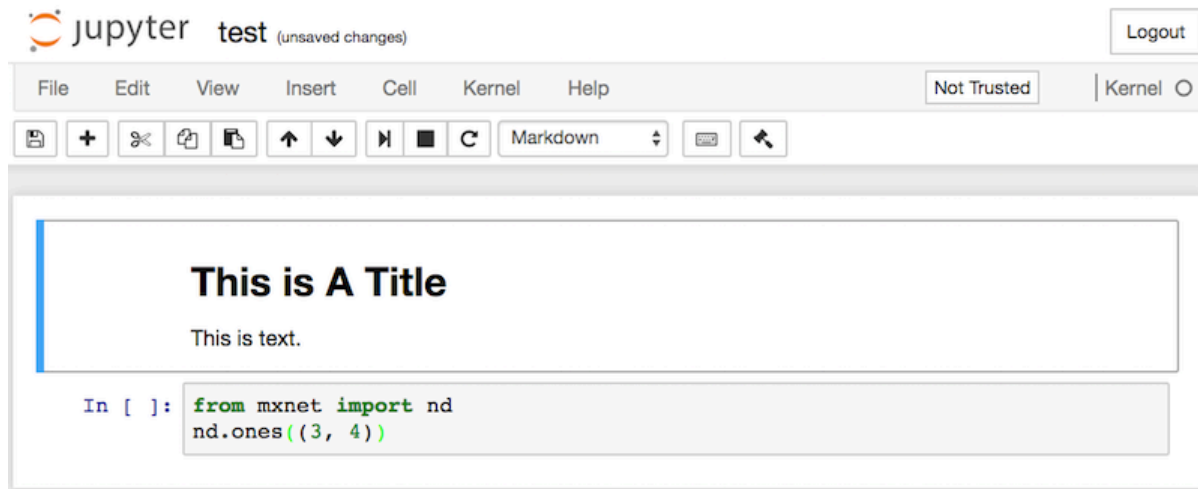


Fig. 18.1.2: Markdown and code cells in the “test.ipynb” file.

Double click on the markdown cell to enter edit mode. Add a new text string “Hello world.” at the end of the cell, as shown in Fig. 18.1.3.

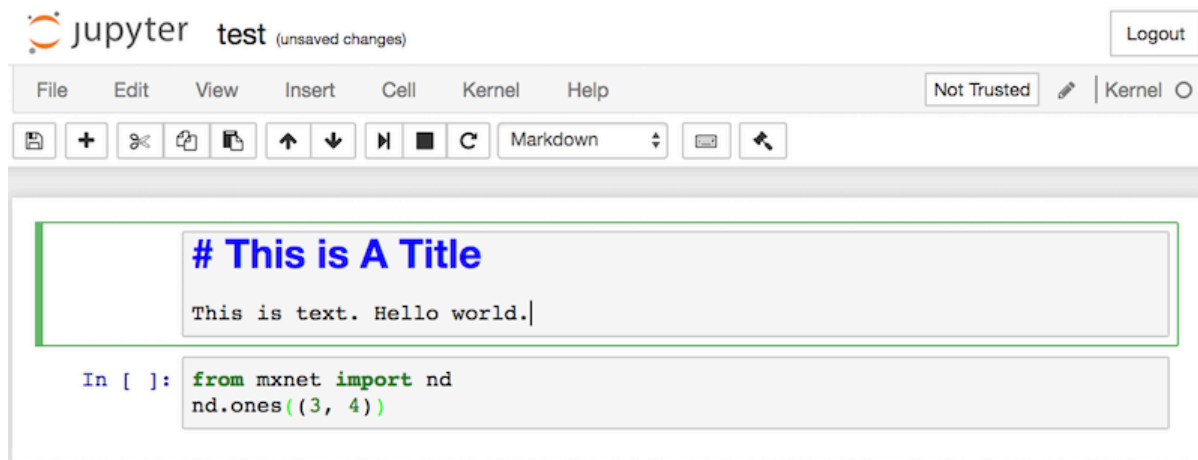


Fig. 18.1.3: Edit the markdown cell.

As shown in Fig. 18.1.4, click “Cell” → “Run Cells” in the menu bar to run the edited cell.

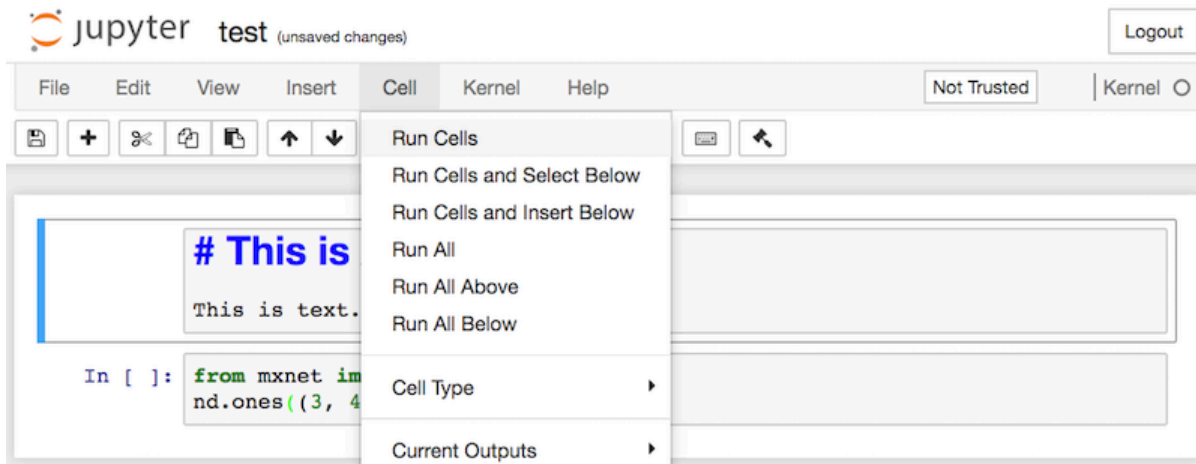


Fig. 18.1.4: Run the cell.

After running, the markdown cell is as shown in Fig. 18.1.5.

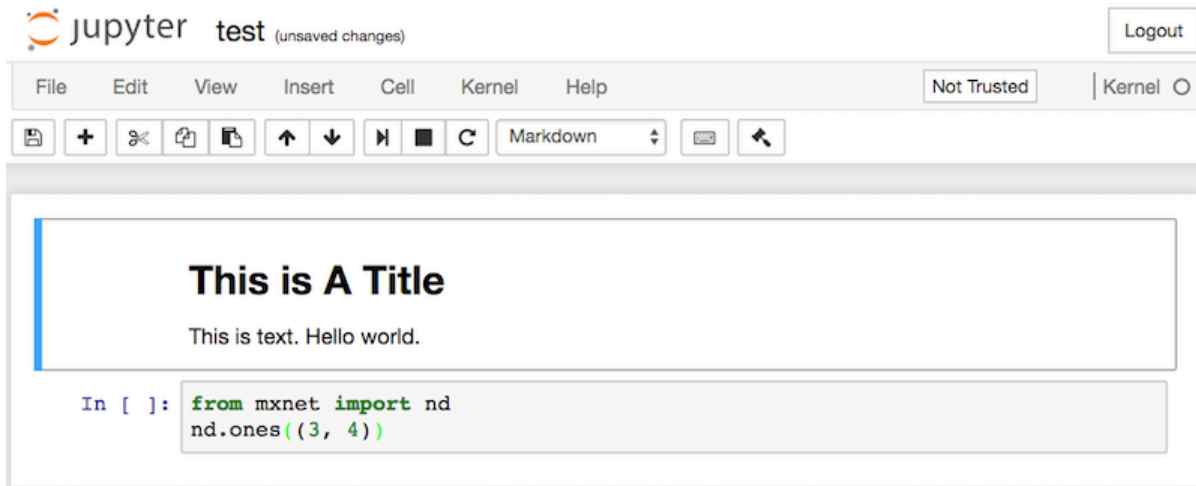


Fig. 18.1.5: The markdown cell after editing.

Next, click on the code cell. Multiply the elements by 2 after the last line of code, as shown in Fig. 18.1.6.

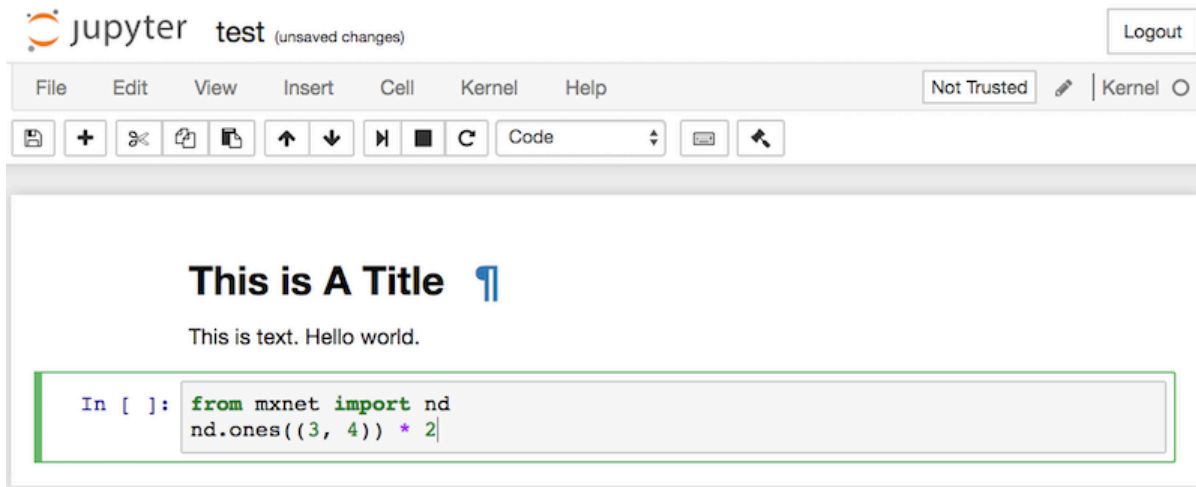


Fig. 18.1.6: Edit the code cell.

You can also run the cell with a shortcut (“Ctrl + Enter” by default) and obtain the output result from Fig. 18.1.7.

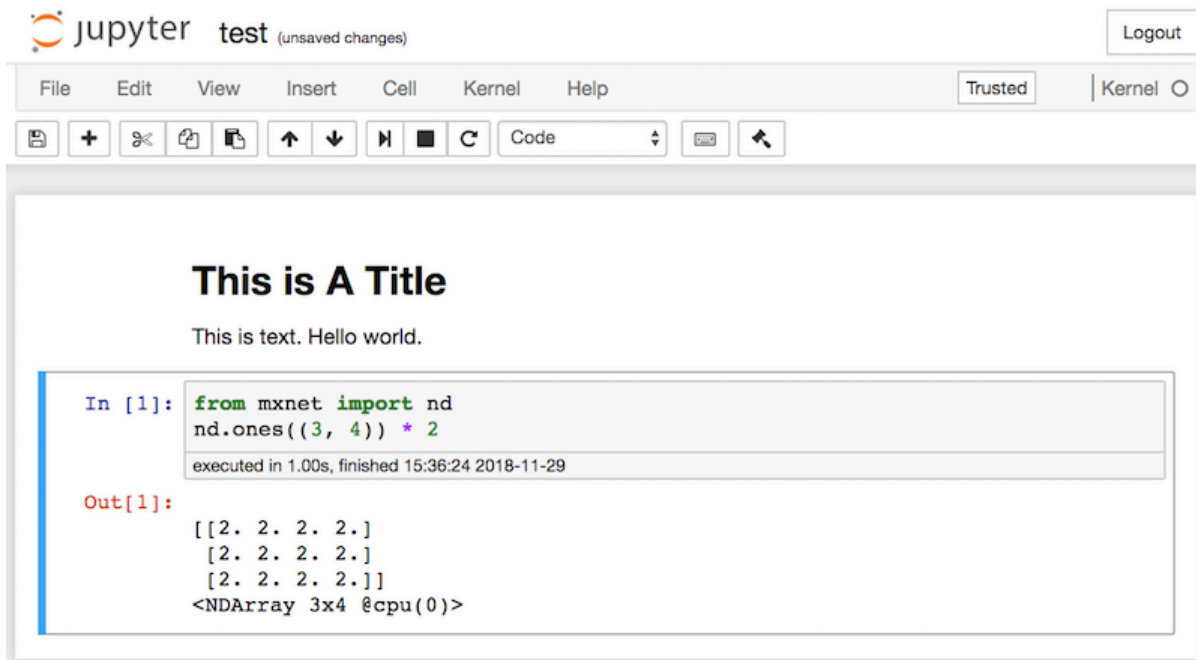


Fig. 18.1.7: Run the code cell to obtain the output.

When a notebook contains more cells, we can click “Kernel” → “Restart & Run All” in the menu bar to run all the cells in the entire notebook. By clicking “Help” → “Edit Keyboard Shortcuts” in the menu bar, you can edit the shortcuts according to your preferences.

18.1.2 Advanced Options

Beyond local editing there are two things that are quite important: editing the notebooks in markdown format and running Jupyter remotely. The latter matters when we want to run the code on a faster server. The former matters since Jupyter's native .ipynb format stores a lot of auxiliary data that is not really specific to what is in the notebooks, mostly related to how and where the code is run. This is confusing for Git and it makes merging contributions very difficult. Fortunately there is an alternative—native editing in Markdown.

Markdown Files in Jupyter

If you wish to contribute to the content of this book, you need to modify the source file (md file, not ipynb file) on GitHub. Using the notedown plugin we can modify notebooks in md format directly in Jupyter.

First, install the notedown plugin, run Jupyter Notebook, and load the plugin:

```
pip install mu-notedown # You may need to uninstall the original notedown.
jupyter notebook --NotebookApp.contents_manager_class='notedown.NotedownContentsManager'
```

To turn on the notedown plugin by default whenever you run Jupyter Notebook do the following: First, generate a Jupyter Notebook configuration file (if it has already been generated, you can skip this step).

```
jupyter notebook --generate-config
```

Then, add the following line to the end of the Jupyter Notebook configuration file (for Linux/macOS, usually in the path ~/.jupyter/jupyter_notebook_config.py):

```
c.NotebookApp.contents_manager_class = 'notedown.NotedownContentsManager'
```

After that, you only need to run the `jupyter notebook` command to turn on the notedown plugin by default.

Running Jupyter Notebook on a Remote Server

Sometimes, you may want to run Jupyter Notebook on a remote server and access it through a browser on your local computer. If Linux or MacOS is installed on your local machine (Windows can also support this function through third-party software such as PuTTY), you can use port forwarding:

```
ssh myserver -L 8888:localhost:8888
```

The above is the address of the remote server `myserver`. Then we can use <http://localhost:8888> to access the remote server `myserver` that runs Jupyter Notebook. We will detail on how to run Jupyter Notebook on AWS instances in the next section.

Timing

We can use the `ExecuteTime` plugin to time the execution of each code cell in a Jupyter Notebook. Use the following commands to install the plugin:

```
pip install jupyter_contrib_nbextensions
jupyter contrib nbextension install --user
jupyter nbextension enable execute_time/ExecuteTime
```

Summary

- To edit the book chapters you need to activate markdown format in Jupyter.
- You can run servers remotely using port forwarding.

Exercises

1. Try to edit and run the code in this book locally.
2. Try to edit and run the code in this book *remotely* via port forwarding.
3. Measure $\mathbf{A}^\top \mathbf{B}$ vs. \mathbf{AB} for two square matrices in $\mathbb{R}^{1024 \times 1024}$. Which one is faster?



18.2 Using Amazon SageMaker

Many deep learning applications require significant amounts of computation. Your local machine might be too slow to solve these problems in a reasonable amount of time. Cloud computing services can give you access to more powerful computers to run the GPU intensive portions of this book. This tutorial will guide you through Amazon SageMaker: a service that allows you to be up and running notebooks easily.

18.2.1 Registering Account and Logging In

First, we need to register an account at <https://aws.amazon.com/>. We strongly encourage you to use two-factor authentication for additional security. Furthermore, it is a good idea to set up detailed billing and spending alerts to avoid any unexpected surprises if you forget to suspend your computers. Note that you will need a credit card. After logging into your AWS account, find “Sagemaker” (see Fig. 18.2.1) to go to the Sagemaker panel.

AWS services

Find Services

You can enter names, keywords or acronyms.

Amazon SageMaker
Build, Train, and Deploy Machine Learning Models

Fig. 18.2.1: Open the Sagemaker console

18.2.2 Creating an SageMaker Instance

Next let's create a notebook instance (Fig. 18.2.2). During the creation, we can specify the instance name, type (Fig. 18.2.3), and notebook repository URL (Fig. 18.2.4). SageMaker provides multiple *instance types*²⁶⁷ with different computation power and price. We used `ml.p3.2xlarge` here. It has a one Tesla V100 GPU and an 8-core CPU, which is powerful enough for most chapters. A Jupyter notebook version of this book that is modified to fit SageMaker is available at <https://github.com/d2l-ai/d2l-en-sagemaker>. We can specify this URL to let SageMaker clone this repository during instance creation.

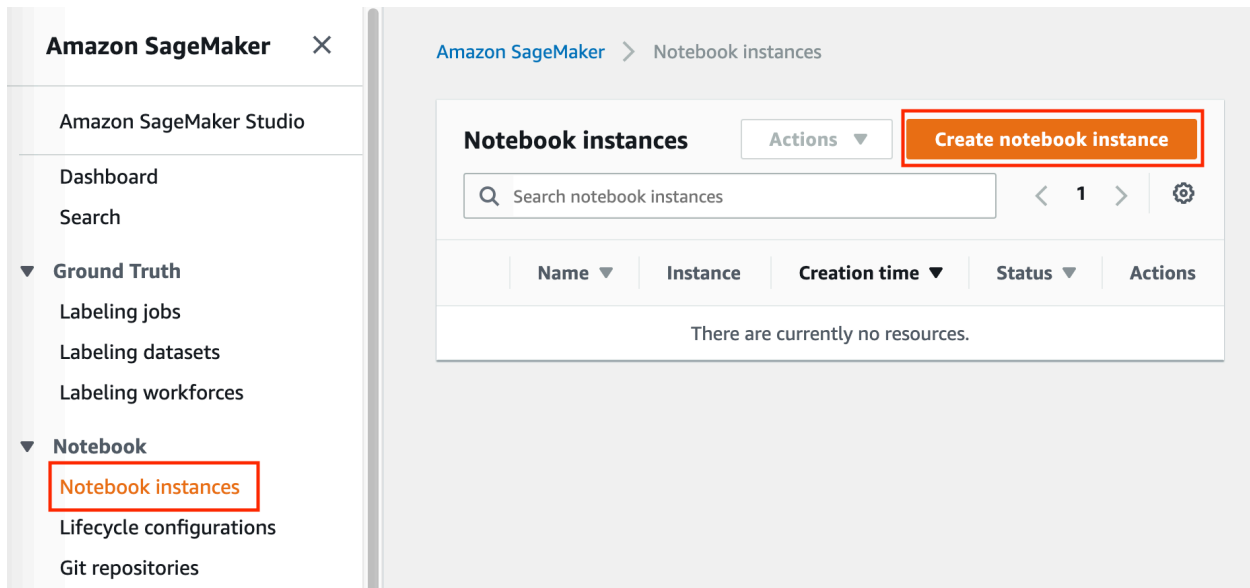


Fig. 18.2.2: Create a notebook instance

²⁶⁷ <https://aws.amazon.com/sagemaker/pricing/instance-types/>

Notebook instance settings

Notebook instance name

D2L

Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account.

Notebook instance type

ml.p3.2xlarge

Fig. 18.2.3: Select instance type

▼ Git repositories - optional

▼ Default repository

Repository

Jupyter will start in this repository. Repositories are added to your home directory.

Clone a public Git repository to this notebook instance only

Git repository URL

Clone a repository to use for this notebook instance only.

https://github.com/d2l-ai/d2l-en-sagemaker

Fig. 18.2.4: Specify the notebook repository.

18.2.3 Running and Stopping an Instance

You may need to wait a few minutes before the instance is ready. Then you can click “Open Jupyter” link (Fig. 18.2.5) to navigate to the Jupyter server running on this instance (Fig. 18.2.6). The usage is similar to a normal Jupyter server running locally (Section 18.1). After finished your work, don’t forget to stop the instance to avoid further charging.

	Name ▼	Instance	Creation time ▼	Status ▼	Actions
<input type="radio"/>	D2L	ml.p3.2xlarge	Dec 18, 2019 19:16 UTC	✔ InService	Open Jupyter Open JupyterLab

Fig. 18.2.5: Open Jupyter on the created instance.

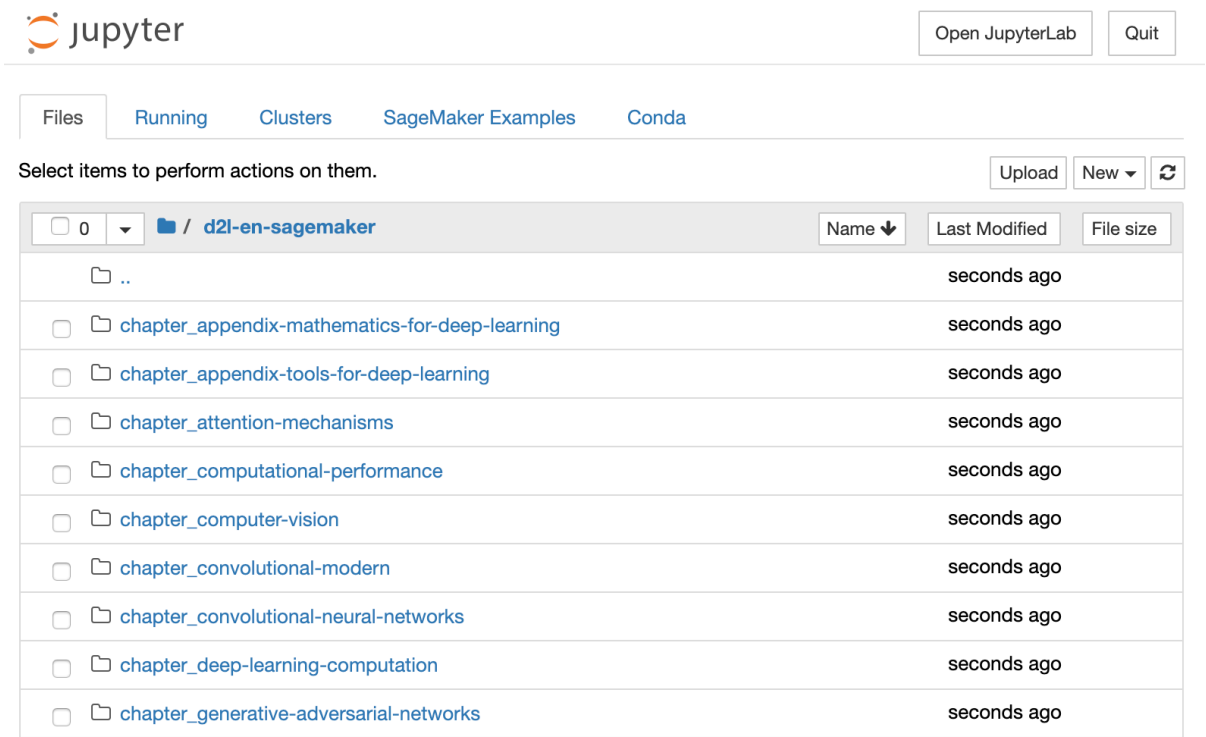


Fig. 18.2.6: The Jupyter server running on the SageMaker instance.

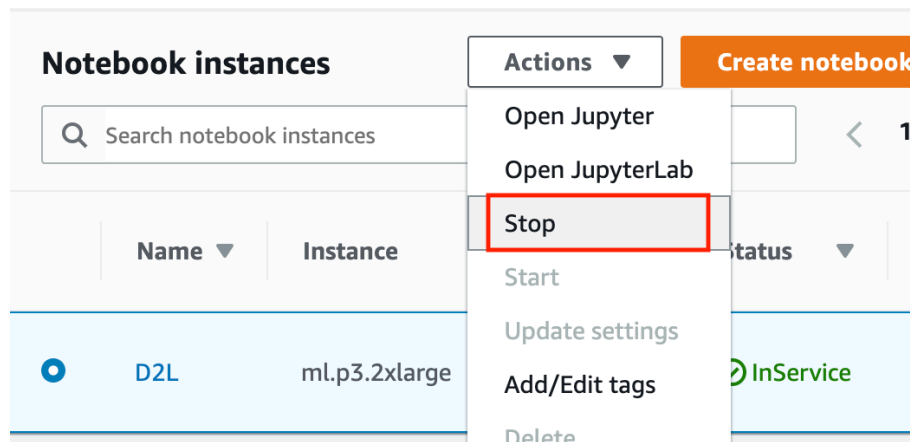


Fig. 18.2.7: Stop your instance.

18.2.4 Updating Notebooks

We will regularly update the notebooks in the [d2l-ai/d2l-en-sagemaker](https://github.com/d2l-ai/d2l-en-sagemaker)²⁶⁸. You can simply `git pull` to update to the latest version. To do so, first open a terminal (Fig. 18.2.8).

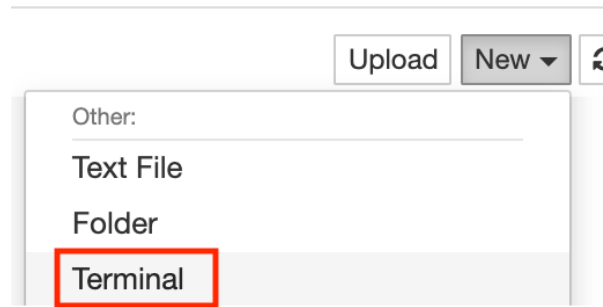


Fig. 18.2.8: Specify the notebook repository.

You may want to commit your local changes first before pulling the updates. Or you can simply ignore all your changes by `git reset --hard`. You can copy paste the following codes in the terminal to do so:

```
cd SageMaker/d2l-en-sagemaker/  
git reset --hard  
git pull
```

Summary

- Cloud computing services offer a wide variety of GPU servers.
- You can launch and stop a Jupyter server through Amazon SageMaker easily.

18.3 Using AWS EC2 Instances

In this section, we will show you how to install all libraries on a raw Linux machine. Remember that in [Section 18.2](#) we discussed how to use Amazon SageMaker, while building an instance by yourself costs less on AWS. The walkthrough includes a number of steps:

1. Request for a GPU Linux instance from AWS EC2.
2. Optionally: install CUDA or use an AMI with CUDA preinstalled.
3. Set up the corresponding MXNet GPU version.

This process applies to other instances (and other clouds), too, albeit with some minor modifications. Before going forward, you need to create an AWS account, see [Section 18.2](#) for more details.

²⁶⁸ <https://github.com/d2l-ai/d2l-en-sagemaker>

18.3.1 Creating and Running an EC2 Instance

After logging into your AWS account, click “EC2” (marked by the red box in Fig. 18.3.1) to go to the EC2 panel.

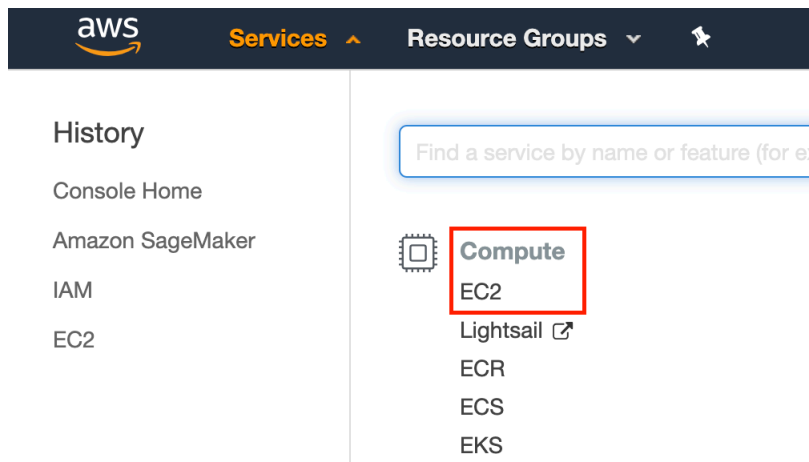


Fig. 18.3.1: Open the EC2 console.

Fig. 18.3.2 shows the EC2 panel with sensitive account information greyed out.

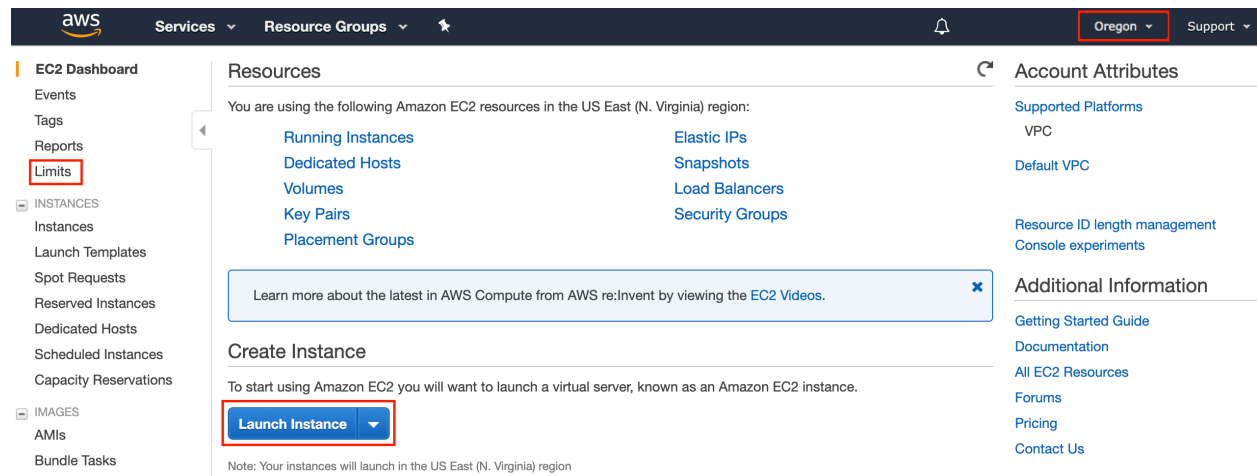


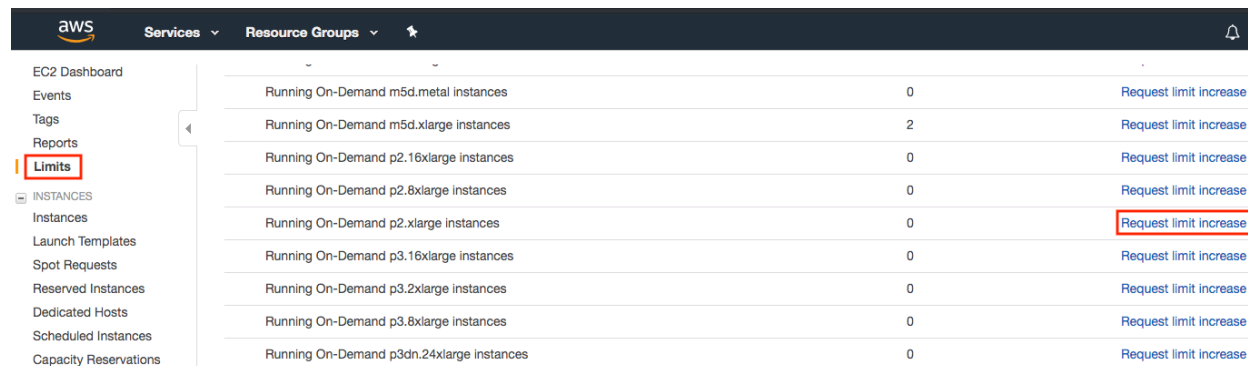
Fig. 18.3.2: EC2 panel.

Presetting Location

Select a nearby data center to reduce latency, *e.g.*, “Oregon”. (marked by the red box in the top-right of Fig. 18.3.2) If you are located in China you can select a nearby Asia Pacific region, such as Seoul or Tokyo. Please note that some data centers may not have GPU instances.

Increasing Limits

Before choosing an instance, check if there are quantity restrictions by clicking the “Limits” label in the bar on the left as shown in Fig. 18.3.2. Fig. 18.3.3 shows an example of such a limitation. The account currently cannot open “p2.xlarge” instance per region. If you need to open one or more instances, click on the “Request limit increase” link to apply for a higher instance quota. Generally, it takes one business day to process an application.



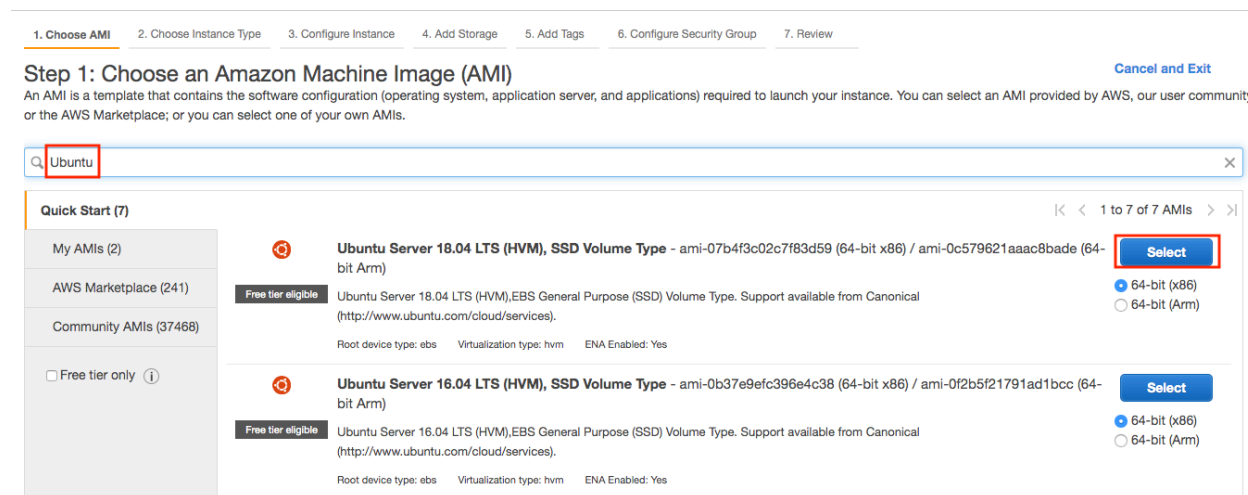
Instance Type	Limit	Action
Running On-Demand m5d.metal instances	0	Request limit increase
Running On-Demand m5d.xlarge instances	2	Request limit increase
Running On-Demand p2.16xlarge instances	0	Request limit increase
Running On-Demand p2.8xlarge instances	0	Request limit increase
Running On-Demand p2.xlarge instances	0	Request limit increase
Running On-Demand p3.16xlarge instances	0	Request limit increase
Running On-Demand p3.2xlarge instances	0	Request limit increase
Running On-Demand p3.8xlarge instances	0	Request limit increase
Running On-Demand p3dn.24xlarge instances	0	Request limit increase

Fig. 18.3.3: Instance quantity restrictions.

Launching Instance

Next, click the “Launch Instance” button marked by the red box in Fig. 18.3.2 to launch your instance.

We begin by selecting a suitable AMI (AWS Machine Image). Enter “Ubuntu” in the search box (marked by the red box in Fig. 18.3.4):



1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Search:

Quick Start (7) 1 to 7 of 7 AMIs



My AMIs (2)	 Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-07b4f3c02c7f83d59 (64-bit x86) / ami-0c579621aaac8bade (64-bit Arm)	Select
AWS Marketplace (241)	Free tier eligible Ubuntu Server 18.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).	<input checked="" type="radio"/> 64-bit (x86) <input type="radio"/> 64-bit (Arm)
Community AMIs (37468)	 Ubuntu Server 16.04 LTS (HVM), SSD Volume Type - ami-0b37e9efc396e4c38 (64-bit x86) / ami-0f2b5f21791ad1bcc (64-bit Arm)	Select
<input type="checkbox"/> Free tier only ⓘ	Free tier eligible Ubuntu Server 16.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).	<input checked="" type="radio"/> 64-bit (x86) <input type="radio"/> 64-bit (Arm)

Fig. 18.3.4: Choose an operating system.

EC2 provides many different instance configurations to choose from. This can sometimes feel overwhelming to a beginner. Here’s a table of suitable machines:

Name	GPU	Notes
g2	Grid K520	ancient
p2	Kepler K80	old but often cheap as spot
g3	Maxwell M60	good trade-off
p3	Volta V100	high performance for FP16
g4	Turing T4	inference optimized FP16/INT8

All the above servers come in multiple flavors indicating the number of GPUs used. For example, a p2.xlarge has 1 GPU and a p2.16xlarge has 16 GPUs and more memory. For more details see *e.g.*, the [AWS EC2 documentation](#)²⁶⁹ or a [summary page](#)²⁷⁰. For the purpose of illustration, a p2.xlarge will suffice (marked in red box of Fig. 18.3.5).

Note: you must use a GPU enabled instance with suitable drivers and a version of MXNet that is GPU enabled. Otherwise you will not see any benefit from using GPUs.

<input type="checkbox"/>	GPU instances	g3.16xlarge	64	488	EBS only	Yes	25 Gigabit	Yes
<input checked="" type="checkbox"/>	GPU instances	p2.xlarge	4	61	EBS only	Yes	High	Yes
<input type="checkbox"/>	GPU instances	p2.8xlarge	32	488	EBS only	Yes	10 Gigabit	Yes

[Cancel](#)
[Previous](#)
[Review and Launch](#)
[Next: Configure Instance Details](#)

Fig. 18.3.5: Choose an instance.

So far, we have finished the first two of seven steps for launching an EC2 instance, as shown on the top of Fig. 18.3.6. In this example, we keep the default configurations for the steps “3. Configure Instance”, “5. Add Tags”, and “6. Configure Security Group”. Tap on “4. Add Storage” and increase the default hard disk size to 64 GB (marked in red box of Fig. 18.3.6). Note that CUDA by itself already takes up 4GB.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance **4. Add Storage** 5. Add Tags 6. Configure Security Group 7. Review

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encrypted
Root	/dev/sda1	snap-0ba4956ec10715d33	64	General Purpose	192 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

Fig. 18.3.6: Modify instance hard disk size.

Finally, go to “7. Review” and click “Launch” to launch the configured instance. The system will now prompt you to select the key pair used to access the instance. If you do not have a key pair, select “Create a new key pair” in the first drop-down menu in Fig. 18.3.7 to generate a key pair. Subsequently, you can select “Choose an existing key pair” for this menu and then select the previously generated key pair. Click “Launch Instances” to launch the created instance.

²⁶⁹ <https://aws.amazon.com/ec2/instance-types/>

²⁷⁰ <https://www.ec2instances.info>

Select an existing key pair or create a new key pair ✕

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

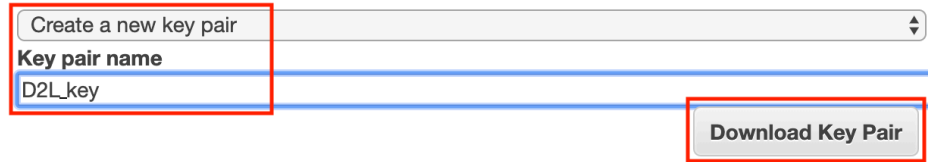



Fig. 18.3.7: Select a key pair.

Make sure that you download the keypair and store it in a safe location if you generated a new one. This is your only way to SSH into the server. Click the instance ID shown in Fig. 18.3.8 to view the status of this instance.

 **Your instances are now launching**

The following instance launches have been initiated: [i-071ee](#) [View launch log](#)

Fig. 18.3.8: Click the instance ID.

Connecting Instance

As shown in Fig. 18.3.9, after the instance state turns green, right-click the instance and select Connect to view the instance access method.



<input type="checkbox"/>	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status
<input type="checkbox"/>		i-0b		b	 running	 2/2
<div>Connect Get Windows Password</div>						

Fig. 18.3.9: View instance access and startup method.

If this is a new key, it must not be publicly viewable for SSH to work. Go to the folder where you store `D2L_key.pem` (e.g., Downloads folder) and make the key to be not publicly viewable.

```
cd /Downloads ## if D2L_key.pem is stored in Downloads folder
chmod 400 D2L_key.pem
```

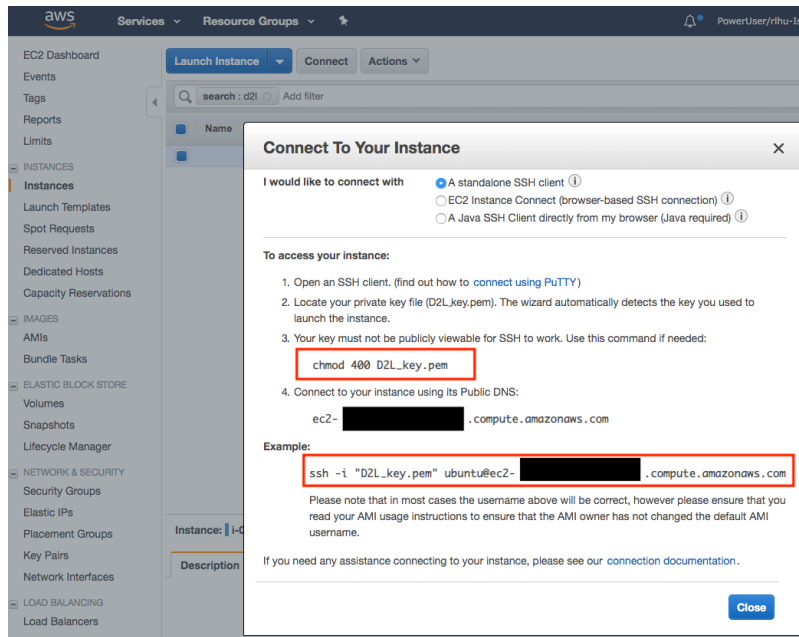


Fig. 18.3.10: View instance access and startup method.

Now, copy the ssh command in the lower red box of Fig. 18.3.10 and paste onto the command line:

```
ssh -i "D2L_key.pem" ubuntu@ec2-xx-xxx-xxx-xxx.y.compute.amazonaws.com
```

When the command line prompts “Are you sure you want to continue connecting (yes/no)”, enter “yes” and press Enter to log into the instance.

Your server is ready now.

18.3.2 Installing CUDA

Before installing CUDA, be sure to update the instance with the latest drivers.

```
sudo apt-get update && sudo apt-get install -y build-essential git libgfortran3
```

Here we download CUDA 10.1. Visit NVIDIA’s official repository at (<https://developer.nvidia.com/cuda-downloads>) to find the download link of CUDA 10.1 as shown in Fig. 18.3.11.

Select Target Platform

Click on the green buttons that describe your target platform. Only supported platforms will be shown.

Operating System	Windows	Linux	Mac OSX
Architecture	x86_64	ppc64le	
Distribution	Fedora	OpenSUSE	RHEL
		CentOS	SLES
			Ubuntu
Version	18.04	16.04	14.04
Installer Type	runfile (local)	deb (local)	deb (network)
			cluster (local)

Download Installer for Linux Ubuntu 18.04 x86_64

The base installer is available for download below.

Fig. 18.3.11: Find the CUDA 10.1 download address.

Copy the instructions and paste them into the terminal to install CUDA 10.1.

```
## Paste the copied link from CUDA website
wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/cuda-ubuntu1804.pin
sudo mv cuda-ubuntu1804.pin /etc/apt/preferences.d/cuda-repository-pin-600
wget http://developer.download.nvidia.com/compute/cuda/10.1/Prod/local_installers/cuda-repo-ubuntu1804-10-1-local-10.1.243-418.87.00_1.0-1_amd64.deb
sudo dpkg -i cuda-repo-ubuntu1804-10-1-local-10.1.243-418.87.00_1.0-1_amd64.deb
sudo apt-key add /var/cuda-repo-10-1-local-10.1.243-418.87.00/7fa2af80.pub
sudo apt-get update
sudo apt-get -y install cuda
```

After installing the program, run the following command to view the GPUs.

```
nvidia-smi
```

Finally, add CUDA to the library path to help other libraries find it.

```
echo "export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/usr/local/cuda/lib64" >> ~/.bashrc
```

18.3.3 Installing MXNet and Downloading the D2L Notebooks

First, to simplify the installation, you need to install [Miniconda](https://conda.io/en/latest/miniconda.html)²⁷¹ for Linux. The download link and file name are subject to changes, so please go the Miniconda website and click “Copy Link Address” as shown in Fig. 18.3.12.

²⁷¹ <https://conda.io/en/latest/miniconda.html>

Miniconda

	Windows	Mac OS X	Linux
Python 3.7	64-bit (exe installer)	64-bit (bash installer)	64-bit (bash installer)
	32-bit (exe installer)	64-bit (.pkg installer)	32-bit (bash installer)
Python 2.7	64-bit (exe installer)	64-bit (bash installer)	64-bit (bash installer)
	32-bit (exe installer)	64-bit (.pkg installer)	32-bit (bash installer)

Fig. 18.3.12: Download Miniconda.

```
# The link and file name are subject to changes
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
sh Miniconda3-latest-Linux-x86_64.sh -b
```

After Miniconda installation, run the following command to activate CUDA and conda.

```
~/miniconda3/bin/conda init
source ~/.bashrc
```

Next, download the code for this book.

```
sudo apt-get install unzip
mkdir d2l-en && cd d2l-en
curl https://d2l.ai/d2l-en-0.7.1.zip -o d2l-en.zip
unzip d2l-en.zip && rm d2l-en.zip
```

Then create the conda d2l environment and enter y to proceed with the installation.

```
conda create --name d2l -y
```

After creating the d2l environment, activate it and install pip.

```
conda activate d2l
conda install python=3.7 pip -y
```

Finally, install MXNet and the d2l package. The postfix cu101 means that this is the CUDA 10.1 variant. For different versions, say only CUDA 10.0, you would want to choose cu100 instead.

```
pip install mxnet-cu101==1.6.0b20191122
pip install d2l==0.11.1
```

You can test quickly whether everything went well as follows:

```
$ python
>>> from mxnet import np, npx
>>> np.zeros((1024, 1024), ctx=npx.gpu())
```

18.3.4 Running Jupyter

To run Jupyter remotely you need to use SSH port forwarding. After all, the server in the cloud does not have a monitor or keyboard. For this, log into your server from your desktop (or laptop) as follows.

```
# This command must be run in the local command line
ssh -i "/path/to/key.pem" ubuntu@ec2-xx-xxx-xxx-xxx.y.compute.amazonaws.com -L
↪8889:localhost:8888
conda activate d2l
jupyter notebook
```

Fig. 18.3.13 shows the possible output after you run Jupyter Notebook. The last row is the URL for port 8888.

```
( d2l ) ubuntu@ip-172-31-2-208:~$ jupyter notebook
[I 06:12:41.588 NotebookApp] Writing notebook server cookie secret to /run/user/1000/jupyter/notebook_cookie_secret
[I 06:12:42.617 NotebookApp] Serving notebooks from local directory: /home/ubuntu
[I 06:12:42.618 NotebookApp] The Jupyter Notebook is running at:
[I 06:12:42.618 NotebookApp] http://localhost:8888/?token=3eb5513
[I 06:12:42.618 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 06:12:42.622 NotebookApp] No web browser found: could not locate runnable browser.
[C 06:12:42.622 NotebookApp]

To access the notebook, open this file in a browser:
file:///run/user/1000/jupyter/nbserver-21907-open.html
Or copy and paste one of these URLs:
http://localhost:8888/?token=3eb5513
```

Fig. 18.3.13: Output after running Jupyter Notebook. The last row is the URL for port 8888.

Since you used port forwarding to port 8889 you will need to replace the port number and use the secret as given by Jupyter when opening the URL in your local browser.

18.3.5 Closing Unused Instances

As cloud services are billed by the time of use, you should close instances that are not being used. Note that there are alternatives: “Stopping” an instance means that you will be able to start it again. This is akin to switching off the power for your regular server. However, stopped instances will still be billed a small amount for the hard disk space retained. “Terminate” deletes all data associated with it. This includes the disk, hence you cannot start it again. Only do this if you know that you will not need it in the future.

If you want to use the instance as a template for many more instances, right-click on the example in Figure 14.16 Fig. 18.3.9 and select “Image” → “Create” to create an image of the instance. Once this is complete, select “Instance State” → “Terminate” to terminate the instance. The next time you want to use this instance, you can follow the steps for creating and running an EC2 instance described in this section to create an instance based on the saved image. The only difference is that, in “1. Choose AMI” shown in Fig. 18.3.4, you must use the “My AMIs” option on the left to select your saved image. The created instance will retain the information stored on the image hard disk. For example, you will not have to reinstall CUDA and other runtime environments.

Summary

- You can launch and stop instances on demand without having to buy and build your own computer.
- You need to install suitable GPU drivers before you can use them.

Exercises

1. The cloud offers convenience, but it does not come cheap. Find out how to launch [spot instances](#)²⁷² to see how to reduce prices.
2. Experiment with different GPU servers. How fast are they?
3. Experiment with multi-GPU servers. How well can you scale things up?



18.4 Using Google Colab

We introduced in [Section 18.2](#) and [Section 18.4](#) for how to run this book on AWS. Another option is running on [Google Colab](#)²⁷⁴, which provides free GPU if you have a Google account.

To run a section on Colab, you can simply click the Colab button on the right of the title [Fig. 18.4.1](#). The first time you execute a code cell, you will receive a warning message saying it is from Github [Fig. 18.4.2](#) and may steal your data. If you trust us, then click “RUN ANYWAY”, then Colab will connect you to an instance to run this notebook. In particular, if GPU is needed, such as `d2l.try_gpu()`, then we will request Colab to connect to a GPU instance automatically.

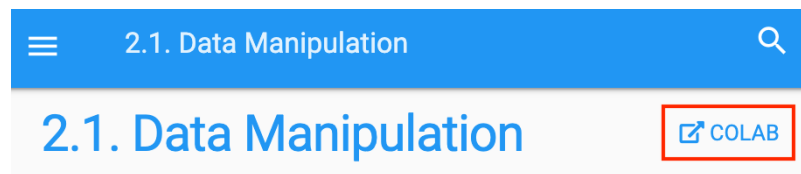


Fig. 18.4.1: Open a section on Colab

²⁷² <https://aws.amazon.com/ec2/spot/>

²⁷⁴ <https://colab.research.google.com/>

Warning: This notebook was not authored ...

This notebook is being loaded from [GitHub](#). It may request access to your data stored with Google, or read data and credentials from other sessions. Please review the source code before executing this notebook.

CANCEL

RUN ANYWAY

Fig. 18.4.2: The warning message for running a section on Colab

Summary

- You can use Google Colab to run each section on GPUs freely.

18.5 Selecting Servers and GPUs

Deep learning training generally requires large amounts of computation. At present GPUs are the most cost-effective hardware accelerators for deep learning. In particular, compared with CPUs, GPUs are cheaper and offer higher performance, often by over an order of magnitude. Furthermore, a single server can support multiple GPUs, up to 8 for high end servers. More typical numbers are up to 4 GPUs for an engineering workstation, since heat, cooling and power requirements escalate quickly beyond what an office building can support. For larger deployments cloud computing, such as Amazon's [P3](#)²⁷⁵ and [G4](#)²⁷⁶ instances are a much more practical solution.

18.5.1 Selecting Servers

There is typically no need to purchase high-end CPUs with many threads since much of the computation occurs on the GPUs. That said, due to the Global Interpreter Lock (GIL) in Python single-thread performance of a CPU can matter in situations where we have 4-8 GPUs. All things equal this suggests that CPUs with a smaller number of cores but a higher clock frequency might be a more economical choice. E.g. when choosing between a 6 core 4GHz and an 8 core 3.5 GHz CPU, the former is much preferable, even though its aggregate speed is less. An important consideration is that GPUs use lots of power and thus dissipate lots of heat. This requires very good cooling and a large enough chassis to use the GPUs. Follow the guidelines below if possible:

1. **Power Supply.** GPUs use significant amounts of power. Budget with up to 350W per device (check for the *peak demand* of the graphics card rather than typical demand, since efficient code can use lots of energy. If your power supply is not up to the demand you will find that your system becomes unstable.
2. **Chassis Size.** GPUs are large and the auxiliary power connectors often need extra space. Also, large chassis are easier to cool.

²⁷⁵ <https://aws.amazon.com/ec2/instance-types/p3/>

²⁷⁶ <https://aws.amazon.com/blogs/aws/in-the-works-ec2-instances-g4-with-nvidia-t4-gpus/>

3. **GPU Cooling.** If you have large numbers of GPUs you might want to invest in water cooling. Also, aim for *reference designs* even if they have fewer fans, since they are thin enough to allow for air intake between the devices. If you buy a multi-fan GPU it might be too thick to get enough air when installing multiple GPUs and you will run into thermal throttling.
4. **PCIe Slots.** Moving data to and from the GPU (and exchanging it between GPUs) requires lots of bandwidth. We recommend PCIe 3.0 slots with 16 lanes. If you mount multiple GPUs, be sure to carefully read the motherboard description to ensure that 16x bandwidth is still available when multiple GPUs are used at the same time and that you are getting PCIe 3.0 as opposed to PCIe 2.0 for the additional slots. Some motherboards downgrade to 8x or even 4x bandwidth with multiple GPUs installed. This is partly due to the number of PCIe lanes that the CPU offers.

In short, here are some recommendations for building a deep learning server:

- **Beginner.** Buy a low end GPU with low power consumption (cheap gaming GPUs suitable for deep learning use 150-200W). If you are lucky your current computer will support it.
- **1 GPU.** A low-end CPU with 4 cores will be plenty sufficient and most motherboards suffice. Aim for at least 32GB DRAM and invest into an SSD for local data access. A power supply with 600W should be sufficient. Buy a GPU with lots of fans.
- **2 GPUs.** A low-end CPU with 4-6 cores will suffice. Aim for 64GB DRAM and invest into an SSD. You will need in the order of 1000W for two high-end GPUs. In terms of mainboards, make sure that they have *two* PCIe 3.0 x16 slots. If you can, get a mainboard that has two free spaces (60mm spacing) between the PCIe 3.0 x16 slots for extra air. In this case, buy two GPUs with lots of fans.
- **4 GPUs.** Make sure that you buy a CPU with relatively fast single-thread speed (i.e., high clock frequency). You will probably need a CPU with a larger number of PCIe lanes, such as an AMD Threadripper. You will likely need relatively expensive mainboards to get 4 PCIe 3.0 x16 slots since they probably need a PLX to multiplex the PCIe lanes. Buy GPUs with reference design that are narrow and let air in between the GPUs. You need a 1600-2000W power supply and the outlet in your office might not support that. This server will probably run *loud and hot*. You do not want it under your desk. 128GB of DRAM is recommended. Get an SSD (1-2TB NVMe) for local storage and a bunch of hard disks in RAID configuration to store your data.
- **8 GPUs.** You need to buy a dedicated multi-GPU server chassis with multiple redundant power supplies (e.g., 2+1 for 1600W per power supply). This will require dual socket server CPUs, 256GB ECC DRAM, a fast network card (10GbE recommended), and you will need to check whether the servers support the *physical form factor* of the GPUs. Airflow and wiring placement differ significantly between consumer and server GPUs (e.g., RTX 2080 vs. Tesla V100). This means that you might not be able to install the consumer GPU in a server due to insufficient clearance for the power cable or lack of a suitable wiring harness (as one of the coauthors painfully discovered).

18.5.2 Selecting GPUs

At present, AMD and NVIDIA are the two main manufacturers of dedicated GPUs. NVIDIA was the first to enter the deep learning field and provides better support for deep learning frameworks via CUDA. Therefore, most buyers choose NVIDIA GPUs.

NVIDIA provides two types of GPUs, targeting individual users (e.g., via the GTX and RTX series) and enterprise users (via its Tesla series). The two types of GPUs provide comparable compute power. However, the enterprise user GPUs generally use (passive) forced cooling, more memory, and ECC (error correcting) memory. These GPUs are more suitable for data centers and usually cost ten times more than consumer GPUs.

If you are a large company with 100+ servers you should consider the NVIDIA Tesla series or alternatively use GPU servers in the cloud. For a lab or a small to medium company with 10+ servers the NVIDIA RTX series is likely most cost effective. You can buy preconfigured servers with Supermicro or Asus chassis that hold 4-8 GPUs efficiently.

GPU vendors typically release a new generation every 1-2 years, such as the GTX 1000 (Pascal) series released in 2017 and the RTX 2000 (Turing) series released in 2019. Each series offers several different models that provide different performance levels. GPU performance is primarily a combination of the following three parameters:

1. **Compute power.** Generally we look for 32-bit floating-point compute power. 16-bit floating point training (FP16) is also entering the mainstream. If you are only interested in prediction, you can also use 8-bit integer. The latest generation of Turing GPUs offers 4-bit acceleration. Unfortunately at present the algorithms to train low-precision networks are not widespread yet.
2. **Memory size.** As your models become larger or the batches used during training grow bigger, you will need more GPU memory. Check for HBM2 (High Bandwidth Memory) vs. GDDR6 (Graphics DDR) memory. HBM2 is faster but much more expensive.
3. **Memory bandwidth.** You can only get the most out of your compute power when you have sufficient memory bandwidth. Look for wide memory buses if using GDDR6.

For most users, it is enough to look at compute power. Note that many GPUs offer different types of acceleration. E.g. NVIDIA's TensorCores accelerate a subset of operators by 5x. Ensure that your libraries support this. The GPU memory should be no less than 4 GB (8GB is much better). Try to avoid using the GPU also for displaying a GUI (use the built-in graphics instead). If you cannot avoid it, add an extra 2GB of RAM for safety.

Fig. 18.5.1 compares the 32-bit floating-point compute power and price of the various GTX 900, GTX 1000 and RTX 2000 series models. The prices are the suggested prices found on Wikipedia.

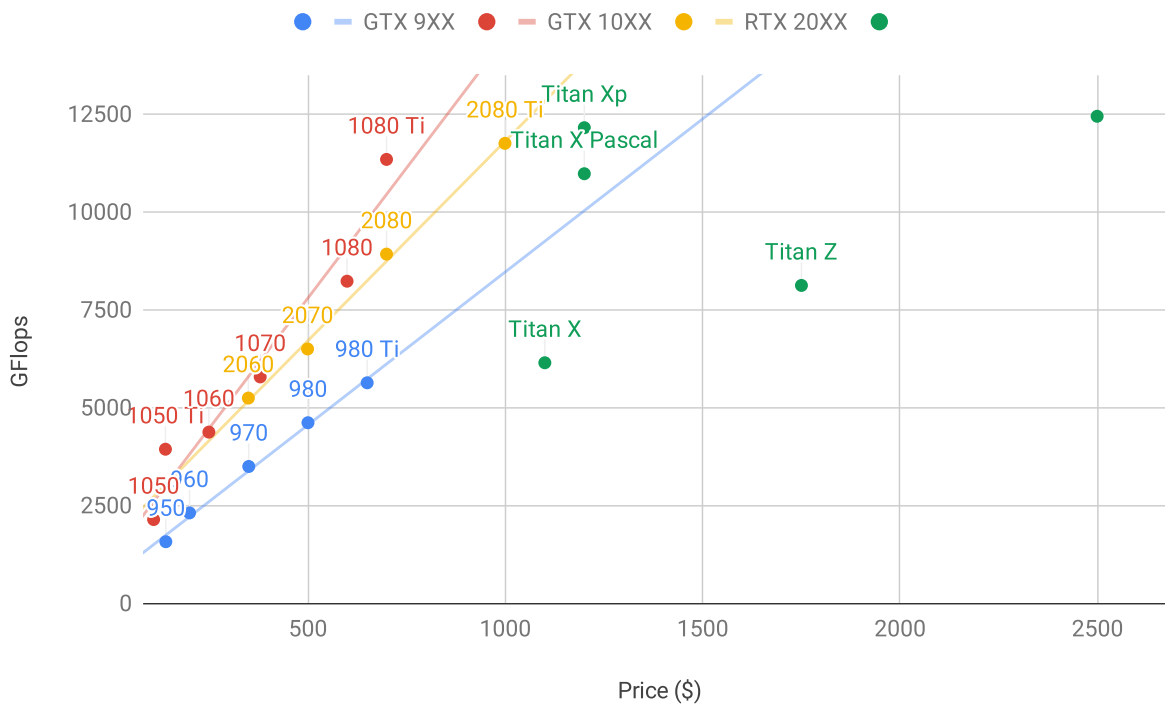


Fig. 18.5.1: Floating-point compute power and price comparison.

We can see a number of things:

1. Within each series, price and performance are roughly proportional. Titan models command a significant premium for the benefit of larger amounts of GPU memory. However, the newer models offer better cost effectiveness, as can be seen by comparing the 980 Ti and 1080 Ti. The price does not appear to improve much for the RTX 2000 series. However, this is due to the fact that they offer far superior low precision performance (FP16, INT8 and INT4).
2. The performance to cost ratio of the GTX 1000 series is about two times greater than the 900 series.
3. For the RTX 2000 series the price is an *affine* function of the price.

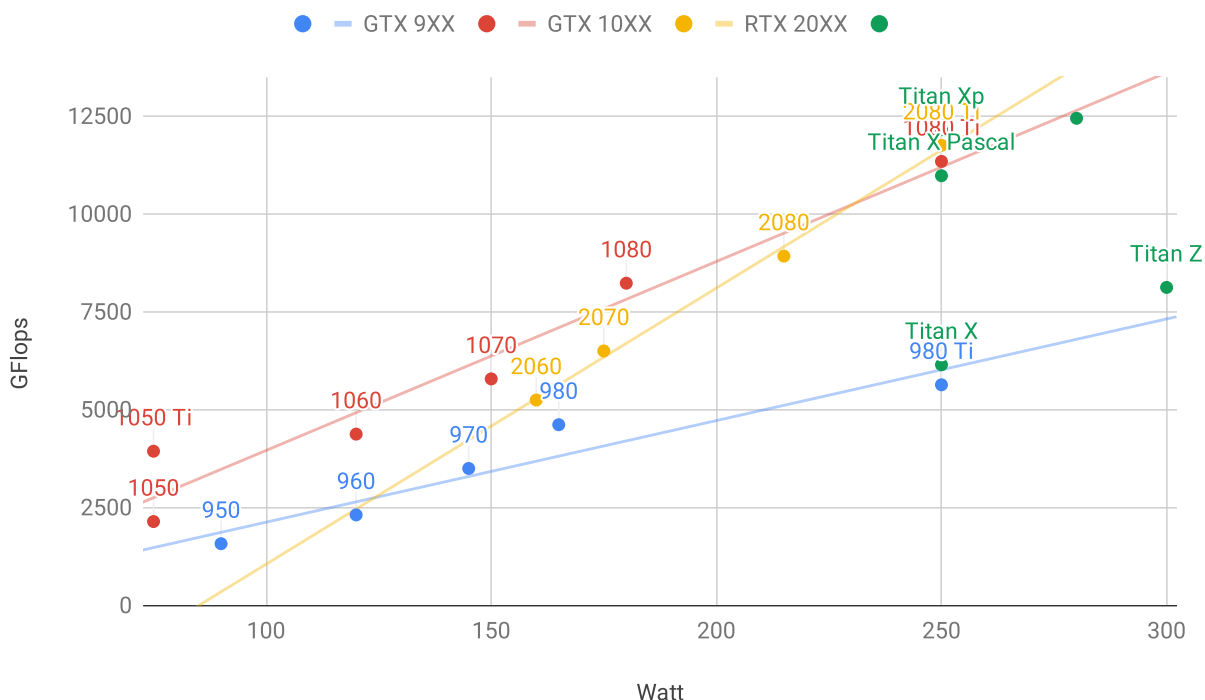


Fig. 18.5.2: Floating-point compute power and energy consumption.

Fig. 18.5.2 shows how energy consumption scales mostly linearly with the amount of computation. Second, later generations are more efficient. This seems to be contradicted by the graph corresponding to the RTX 2000 series. However, this is a consequence of the TensorCores which draw disproportionately much energy.

Summary

- Watch out for power, PCIe bus lanes, CPU single thread speed and cooling when building a server.
- You should purchase the latest GPU generation if possible.
- Use the cloud for large deployments.
- High density servers may not be compatible with all GPUs. Check the mechanical and cooling specifications before you buy.
- Use FP16 or lower precision for high efficiency.



18.6 Contributing to This Book

Contributions by [readers](#)²⁷⁸ help us improve this book. If you find a typo, an outdated link, something where you think we missed a citation, where the code does not look elegant or where an explanation is unclear, please contribute back and help us help our readers. While in regular books the delay between print runs (and thus between typo corrections) can be measured in years, it typically takes hours to days to incorporate an improvement in this book. This is all possible due to version control and continuous integration testing. To do so you need to install Git and submit a [pull request](#)²⁷⁹ to the GitHub repository. When your pull request is merged into the code repository by the author, you will become a contributor. In a nutshell the process works as described in [Fig. 18.6.1](#).

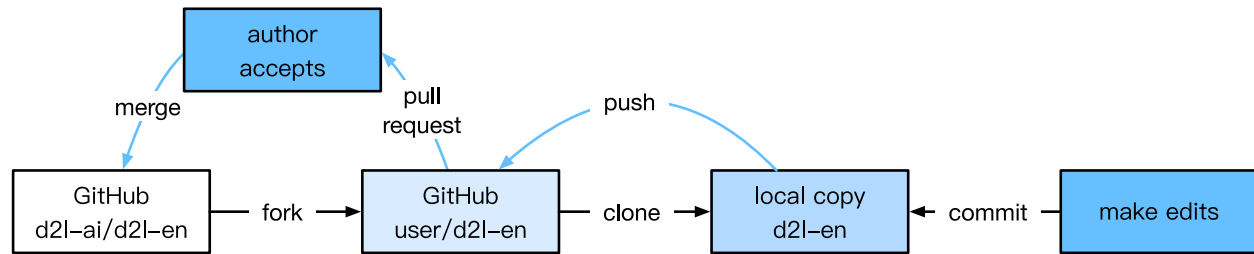


Fig. 18.6.1: Contributing to the book.

18.6.1 From Reader to Contributor in 6 Steps

We will walk you through the steps in detail. If you are already familiar with Git you can skip this section. For concreteness we assume that the contributor’s user name is “astonzhang”.

Installing Git

The Git open source book describes [how to install Git](#)²⁸⁰. This typically works via `apt install git` on Ubuntu Linux, by installing the Xcode developer tools on macOS, or by using GitHub’s [desktop client](#)²⁸¹. If you do not have a GitHub account, you need to sign up for one.

Logging in to GitHub

Enter the [address](#)²⁸² of the book’s code repository in your browser. Click on the Fork button in the red box at the top-right of [Fig. 18.6.2](#), to make a copy of the repository of this book. This is now *your copy* and you can change it any way you want.

²⁷⁸ <https://github.com/d2l-ai/d2l-en/graphs/contributors>

²⁷⁹ <https://github.com/d2l-ai/d2l-en/pulls>

²⁸⁰ <https://git-scm.com/book/zh/v2>

²⁸¹ <https://desktop.github.com>

²⁸² <https://github.com/d2l-ai/d2l-en/>

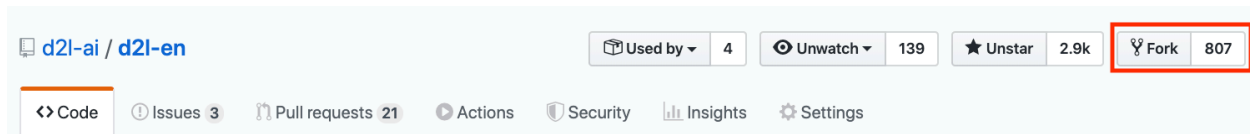


Fig. 18.6.2: The code repository page.

Now, the code repository of this book will be copied to your username, such as `astonzhang/d2l-en` shown at the top-left of the screenshot Fig. 18.6.3.

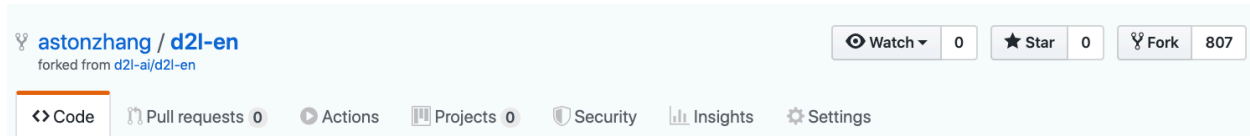


Fig. 18.6.3: Copy the code repository.

Cloning the Repository

To clone the repository (i.e., to make a local copy) we need to get its repository address. The green button in Fig. 18.6.4 displays this. Make sure that your local copy is up to date with the main repository if you decide to keep this fork around for longer. For now simply follow the instructions in Chapter to get started. The main difference is that you are now downloading *your own fork* of the repository.

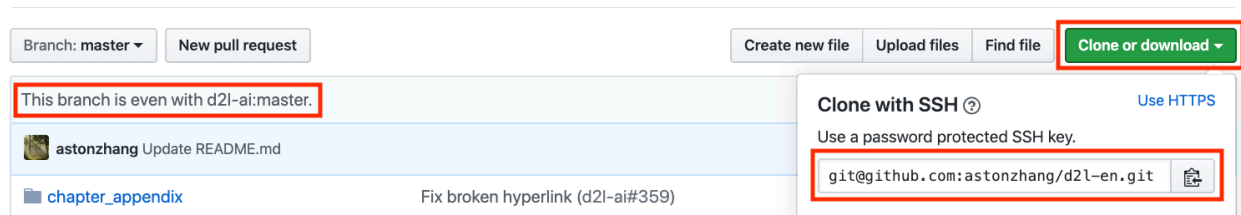


Fig. 18.6.4: Git clone.

```
# Replace your_github_username with your GitHub username
git clone https://github.com/your_github_username/d2l-en.git
```

Editing the Book and Push

Now it is time to edit the book. It is best to edit the notebooks in Jupyter following instructions in Section 18.1. Make the changes and check that they are OK. Assume we have modified a typo in the file `~/d2l-en/chapter_appendix_tools/how-to-contribute.md`. You can then check which files you have changed:

At this point Git will prompt that the `chapter_appendix_tools/how-to-contribute.md` file has been modified.

```
mylaptop:d2l-en me$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   chapter_appendix_tools/how-to-contribute.md
```

After confirming that this is what you want, execute the following command:

```
git add chapter_appendix_tools/how-to-contribute.md
git commit -m 'fix typo in git documentation'
git push
```

The changed code will then be in your personal fork of the repository. To request the addition of your change, you have to create a pull request for the official repository of the book.

Pull Request

As shown in Fig. 18.6.5, go to your fork of the repository on GitHub and select “New pull request”. This will open up a screen that shows you the changes between your edits and what is current in the main repository of the book.

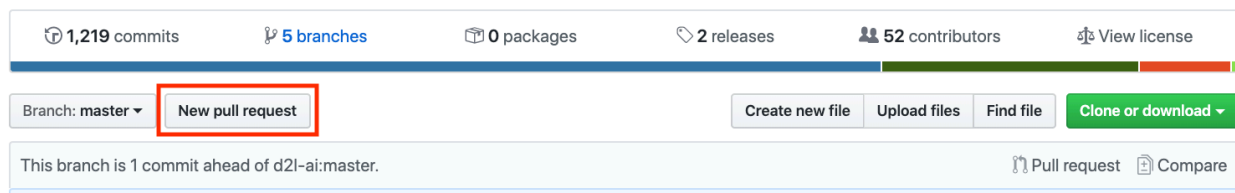


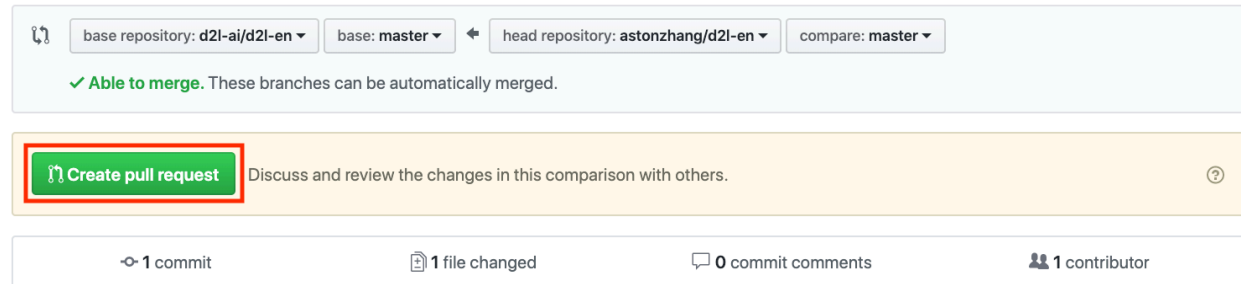
Fig. 18.6.5: Pull Request.

Submitting Pull Request

Finally, submit a pull request by clicking the button as shown in Fig. 18.6.6. Make sure to describe the changes you have made in the pull request. This will make it easier for the authors to review it and to merge it with the book. Depending on the changes, this might get accepted right away, rejected, or more likely, you will get some feedback on the changes. Once you have incorporated them, you are good to go.

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).



base repository: d2l-ai/d2l-en base: master head repository: astonzhang/d2l-en compare: master

✓ Able to merge. These branches can be automatically merged.

Create pull request Discuss and review the changes in this comparison with others.

1 commit 1 file changed 0 commit comments 1 contributor

Fig. 18.6.6: Create Pull Request.

Your pull request will appear among the list of requests in the main repository. We will make every effort to process it quickly.

Summary

- You can use GitHub to contribute to this book.
- Forking a repository is the first step to contributing, since it allows you to edit things locally and only contribute back once you are ready.
- Pull requests are how contributions are being bundled up. Try not to submit huge pull requests since this makes them hard to understand and incorporate. Better send several smaller ones.

Exercises

1. Star and fork the d2l-en repository.
2. Find some code that needs improvement and submit a pull request.
3. Find a reference that we missed and submit a pull request.



18.7 d2l API Document

`class d2l.Accumulator(n)`

Sum a list of numbers over time

`class d2l.BPRLoss(weight=None, batch_axis=0, **kwargs)`

`forward(positive, negative)`

Defines the forward computation. Arguments can be either `NDArray` or `Symbol`.

`class d2l.CTRDataset(data_path, feat_mapper=None, defaults=None, min_threshold=4, num_feat=34)`

`class d2l.Decoder(**kwargs)`

The base decoder interface for the encoder-decoder architecture.

`forward(X, state)`

Overrides to implement forward computation using `NDArray`. Only accepts positional arguments.

***args** [list of `NDArray`] Input tensors.

`class d2l.DotProductAttention(dropout, **kwargs)`

`forward(query, key, value, valid_length=None)`

Overrides to implement forward computation using `NDArray`. Only accepts positional arguments.

***args** [list of `NDArray`] Input tensors.

`class d2l.Encoder(**kwargs)`

The base encoder interface for the encoder-decoder architecture.

`forward(X)`

Overrides to implement forward computation using `NDArray`. Only accepts positional arguments.

***args** [list of `NDArray`] Input tensors.

`class d2l.EncoderDecoder(encoder, decoder, **kwargs)`

The base class for the encoder-decoder architecture.

`forward(enc_X, dec_X, *args)`

Overrides to implement forward computation using `NDArray`. Only accepts positional arguments.

***args** [list of `NDArray`] Input tensors.

`class d2l.HingeLossbRec(weight=None, batch_axis=0, **kwargs)`

`forward(positive, negative, margin=1)`

Defines the forward computation. Arguments can be either `NDArray` or `Symbol`.

`class d2l.MLPAttention(units, dropout, **kwargs)`

```

forward(query, key, value, valid_length)
    Overrides to implement forward computation using NDAarray. Only accepts positional
    arguments.

    *args [list of NDAarray] Input tensors.
class d2l.MaskedSoftmaxCELoss(axis=-1, sparse_label=True, from_logits=False, weight=None,
                             batch_axis=0, **kwargs)

forward(pred, label, valid_length)
    Defines the forward computation. Arguments can be either NDAarray or Symbol.
class d2l.RNNModel(rnn_layer, vocab_size, **kwargs)

forward(inputs, state)
    Overrides to implement forward computation using NDAarray. Only accepts positional
    arguments.

    *args [list of NDAarray] Input tensors.
class d2l.RNNModelScratch(vocab_size, num_hiddens, ctx, get_params, init_state, forward)
    A RNN Model based on scratch implementations
class d2l.RandomGenerator(sampling_weights)
    Draw a random int in [0, n] according to n sampling weights
class d2l.Residual(num_channels, use_1x1conv=False, strides=1, **kwargs)

forward(X)
    Overrides to implement forward computation using NDAarray. Only accepts positional
    arguments.

    *args [list of NDAarray] Input tensors.
class d2l.Seq2SeqDecoder(vocab_size, embed_size, num_hiddens, num_layers, dropout=0,
                        **kwargs)

forward(X, state)
    Overrides to implement forward computation using NDAarray. Only accepts positional
    arguments.

    *args [list of NDAarray] Input tensors.
class d2l.Seq2SeqEncoder(vocab_size, embed_size, num_hiddens, num_layers, dropout=0,
                        **kwargs)

forward(X, *args)
    Overrides to implement forward computation using NDAarray. Only accepts positional
    arguments.

    *args [list of NDAarray] Input tensors.
class d2l.SeqDataLoader(batch_size, num_steps, use_random_iter, max_tokens)
    A iterator to load sequence data
class d2l.Timer
    Record multiple running times.

```

```
class d2l.VOCSegDataset(is_train, crop_size, voc_dir)
```

A customized dataset to load VOC dataset.

```
filter(imgs)
```

Returns a new dataset with samples filtered by the filter function *fn*.

Note that if the Dataset is the result of a lazily transformed one with `transform(lazy=False)`, the filter is eagerly applied to the transformed samples without materializing the transformed result. That is, the transformation will be applied again whenever a sample is retrieved after `filter()`.

fn [callable] A filter function that takes a sample as input and returns a boolean. Samples that return False are discarded.

Dataset The filtered dataset.

```
d2l.bbox_to_rect(bbox, color)
```

Convert bounding box to matplotlib format.

```
d2l.build_colormap2label()
```

Build a RGB color to label mapping for segmentation.

```
d2l.copyfile(filename, target_dir)
```

Copy a file into a target directory

```
d2l.corr2d(X, K)
```

Compute 2D cross-correlation.

```
class d2l.defaultdict
```

`defaultdict(default_factory[, ...])` -> dict with default factory

The default factory is called without arguments to produce a new value when a key is not present, in `__getitem__` only. A `defaultdict` compares equal to a dict with the same items. All remaining arguments are treated the same as if they were passed to the dict constructor, including keyword arguments.

`copy()` -> a shallow copy of D.

`default_factory`

Factory for default value called by `__missing__()`.

```
d2l.download(name, cache_dir='./data')
```

Download a file inserted into DATA_HUB, return the local filename

```
d2l.download_all()
```

Download all files in the DATA_HUB

```
d2l.download_extract(name, folder=None)
```

Download and extract a .zip or a .tar file

```
d2l.evaluate_loss(net, data_iter, loss)
```

Evaluate the loss of a model on the given dataset

```
d2l.load_array(data_arrays, batch_size, is_train=True)
```

Construct a Gluon data loader

```
d2l.load_data_fashion_mnist(batch_size, resize=None)
```

Download the Fashion-MNIST dataset and then load into memory.

`d2l.load_data_pikachu(batch_size, edge_size=256)`
Load the pikachu dataset

`d2l.load_data_voc(batch_size, crop_size)`
Download and load the VOC2012 semantic dataset.

`d2l.n_valid_per_label(labels, valid_ratio)`
Determine # examples per class for the validation set

`d2l.plot(X, Y=None, xlabel=None, ylabel=None, legend=[], xlim=None, ylim=None, xscale='linear', yscale='linear', ffmts=['-', 'm--', 'g-', 'r:'], figsize=(3.5, 2.5), axes=None)`
Plot data points.

`d2l.read_csv_labels(fname)`
Read fname to return a name to label dictionary

`d2l.read_time_machine()`
Load the time machine book into a list of sentences.

`d2l.read_voc_images(voc_dir, is_train=True)`
Read all VOC feature and label images.

`d2l.resnet18(num_classes)`
A slightly modified ResNet-18 model.

`d2l.set_axes(axes, xlabel, ylabel, xlim, ylim, xscale, yscale, legend)`
Set the axes for matplotlib.

`d2l.set_figsize(figsize=(3.5, 2.5))`
Set the figure size for matplotlib.

`d2l.show_bboxes(axes, bboxes, labels=None, colors=None)`
Show bounding boxes.

`d2l.show_images(imgs, num_rows, num_cols, titles=None, scale=1.5)`
Plot a list of images.

`d2l.show_trace_2d(f, results)`
Show the trace of 2D variables during optimization.

`d2l.split_batch(X, y, ctx_list)`
Split X and y into multiple devices specified by ctx.

`d2l.split_data_m100k(data, num_users, num_items, split_mode='random', test_ratio=0.1)`
Split the dataset in random mode or seq-aware mode.

`d2l.synthetic_data(w, b, num_examples)`
generate $y = Xw + b + \text{noise}$

`d2l.tokenize(lines, token='word')`
Split sentences into word or char tokens

`d2l.train_2d(trainer, steps=20)`
Optimize a 2-dim objective function with a customized trainer.

`d2l.try_all_gpus()`
Return all available GPUs, or [cpu()], if no GPU exists.

`d2l.try_gpu(i=0)`
Return gpu(i) if exists, otherwise return cpu().

`d2l.update_D(X, Z, net_D, net_G, loss, trainer_D)`
Update discriminator

`d2l.update_G(Z, net_D, net_G, loss, trainer_G)`
Update generator

`d2l.use_svg_display()`
Use the svg format to display a plot in Jupyter.

`d2l.voc_label_indices(colormap, colormap2label)`
Map a RGB color to a label.

`d2l.voc_rand_crop(feature, label, height, width)`
Randomly crop for both feature and label images.