

Software Tools for Machine Learning & Deep Learning

Shuyue (Frank) Guan

The Medical Imaging & Image Analysis (MIA) Laboratory
Sep. 2021



Hardware

- CPU

- Speed (GHz) – General ML
- # of Cores – Parallel computing

- RAM – General ML

- Size (GB) – Image Processing
- Speed (MHz) – Loading speed

- GPU – DL

- Buffer (Memory) – **Crucial**: model scale, input/batch size ($\geq 6\text{GB}$)
- CUDA Cores – Speed

The image shows a screenshot of the NVIDIA GeForce GTX 1080 Ti specifications page. The title 'GEFORCE GTX 1080 Ti' is at the top in green. Below it, 'GPU Engine Specs:' is followed by a table with two columns: the specification name and its value. 'NVIDIA CUDA* Cores' is 3584, and 'Boost Clock (MHz)' is 1582. 'Memory Specs:' is followed by another table with two columns: the specification name and its value. 'Memory Speed' is 11 Gbps, 'Standard Memory Config' is 11 GB GDDR5X, and 'Memory Interface Width' is 352-bit.

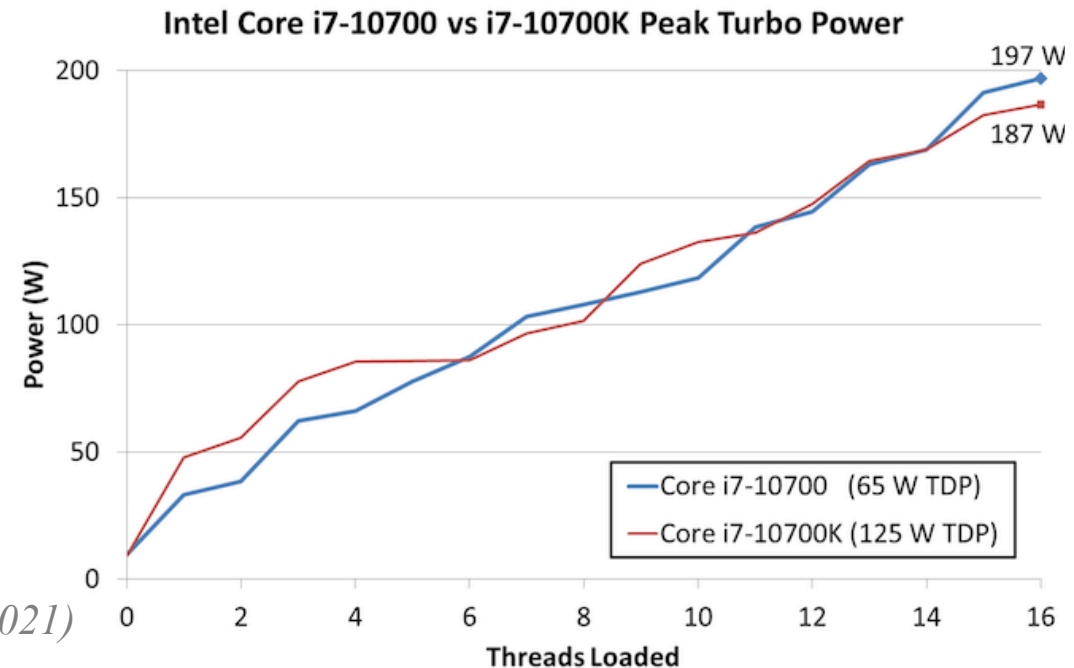
GPU Engine Specs:	
NVIDIA CUDA* Cores	3584
Boost Clock (MHz)	1582

Memory Specs:	
Memory Speed	11 Gbps
Standard Memory Config	11 GB GDDR5X
Memory Interface Width	352-bit

Out of memory (OOM) issues

Hardware (Cont.)

- Solid-state disk (SSD) – read/write data fast
 - Speed (MB/s)
 - Size (GB) – by needs
 - **Tips: SSD (512GB) + HDD (2TB)*
- Cooling system, power supply
 - Multi-CPU
 - Multi-GPU
 - High-speed RAM/SSD



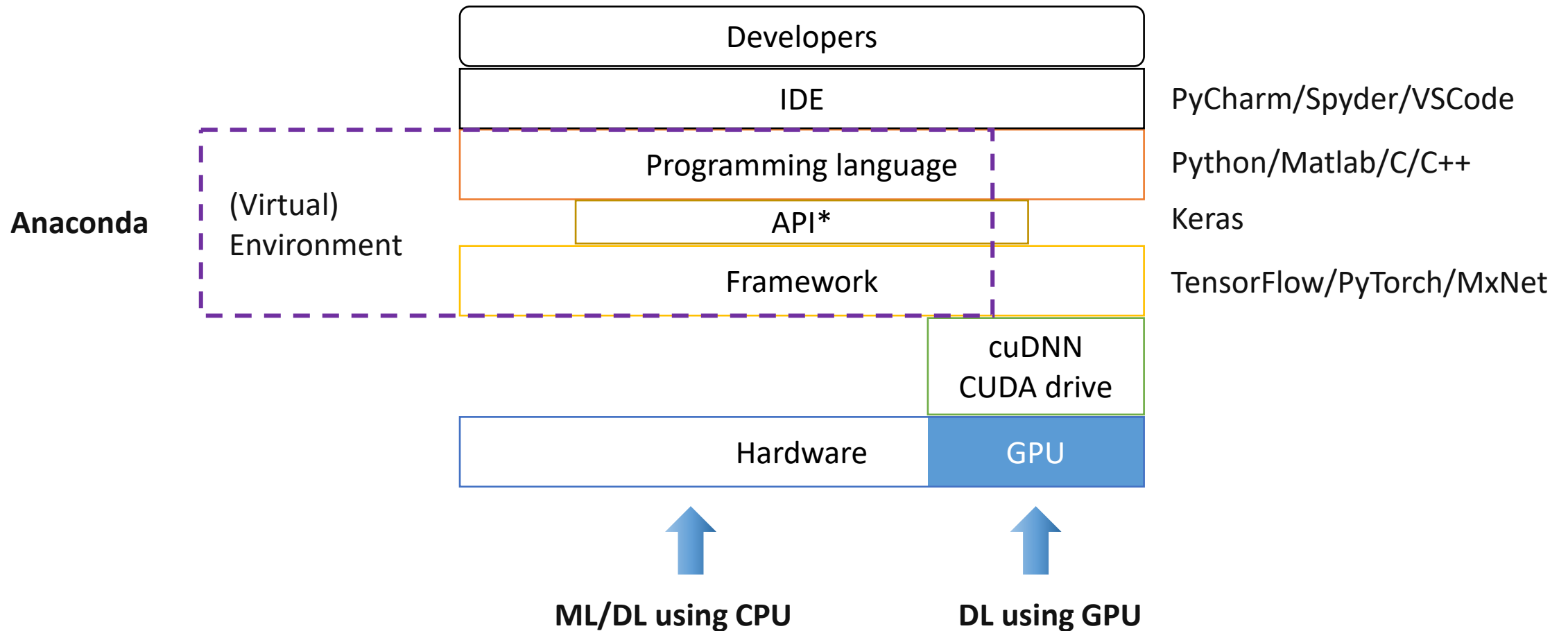
(Ian Cutress, 2021)

Hardware: Tips

- Develop/test on PC, running on workstation
 - Smaller input/batch sizes
 - Smaller model scales
 - Smaller loops
 - ...
- HPC at GWU (*discuss details later*)
- Cloud Computing resources
 - AWS
 - Google Cloud
 - ...



Software overview



Languages for general ML

- MATLAB



- **Advantages:** quick to get started and easy to use; visualization; detailed official help documentation; high credibility
- **Disadvantages:** need to buy; slow official updates; lower flexibility; not many users in the DL field

- Python



- **Advantages:** free; fast community-based update; higher flexibility; widely used in the NLP/DL field; rich packages
- **Disadvantages:** difficult to get started; not intuitive enough; mixed help information; may not be reliable

Languages/tools for general ML

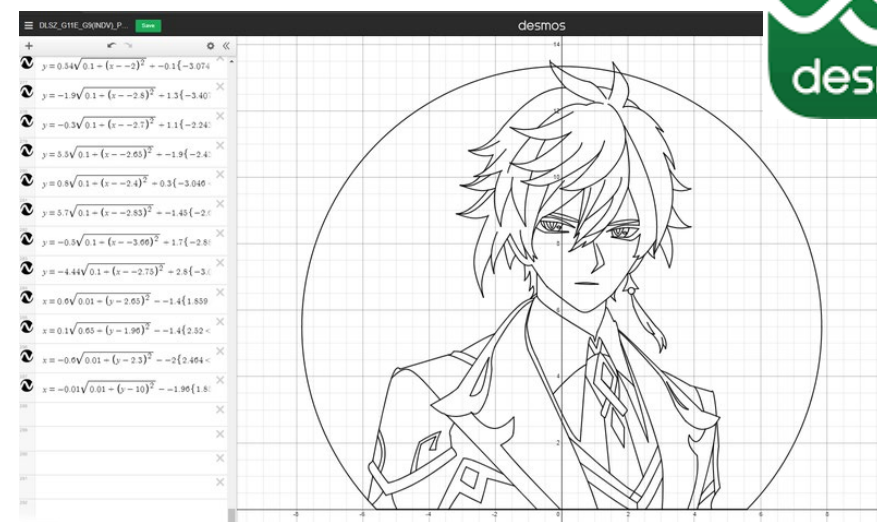
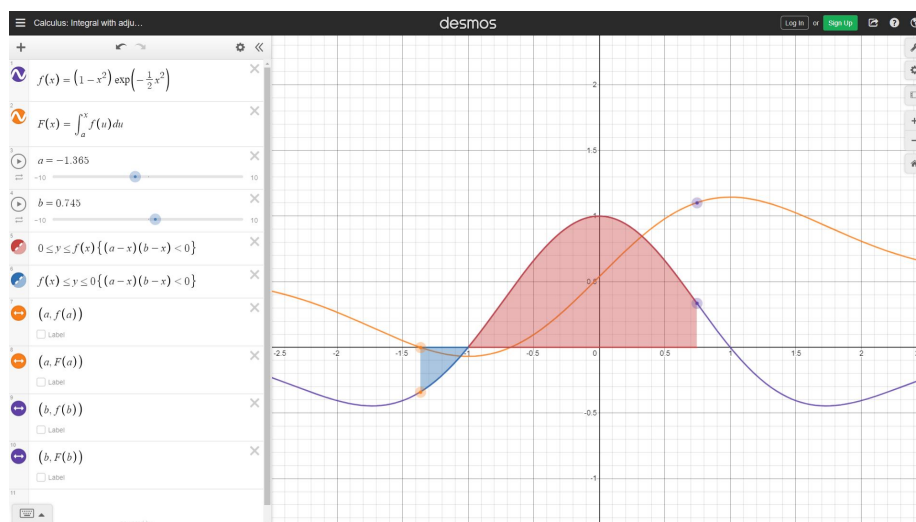
- Other tools

- **Wolfram Mathematica:** symbolic & numeric computations

```
In[3]:= DSolve[y''[x] + y[x] == Exp[x], y, x]
```

```
Out[3]= {{y -> Function[{x}, c1 Cos[x] + c2 Sin[x] + 1/2 e^x (Cos[x]^2 + Sin[x]^2)]}}
```

- **Desmos:** graphing calculator, FREE



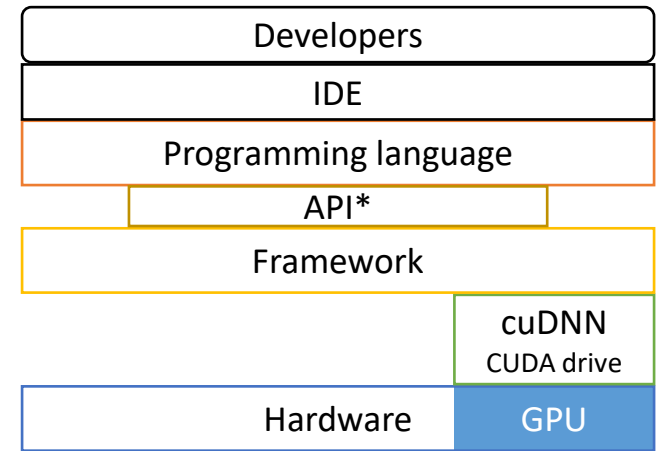
Tools for DL

Examples:

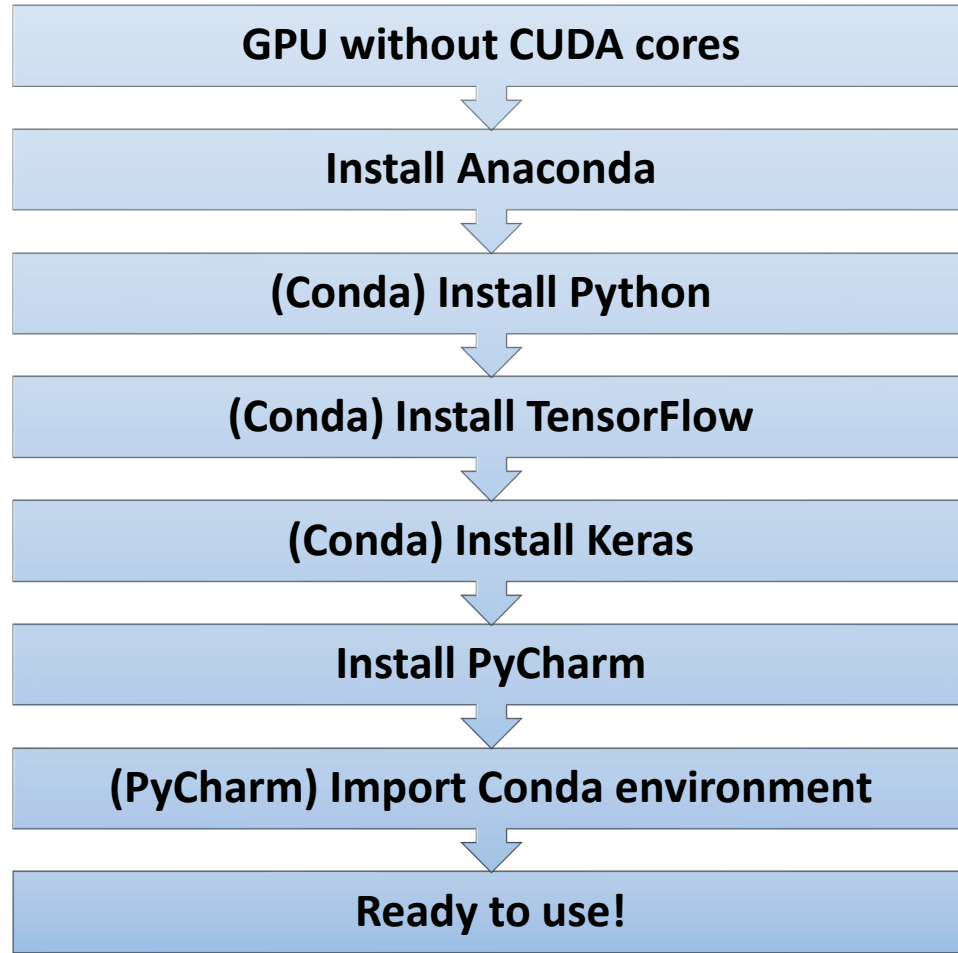
- TensorFlow + Keras + Python + PyCharm
- PyTorch + Python + VSCode/PyCharm

Recommend: Anaconda

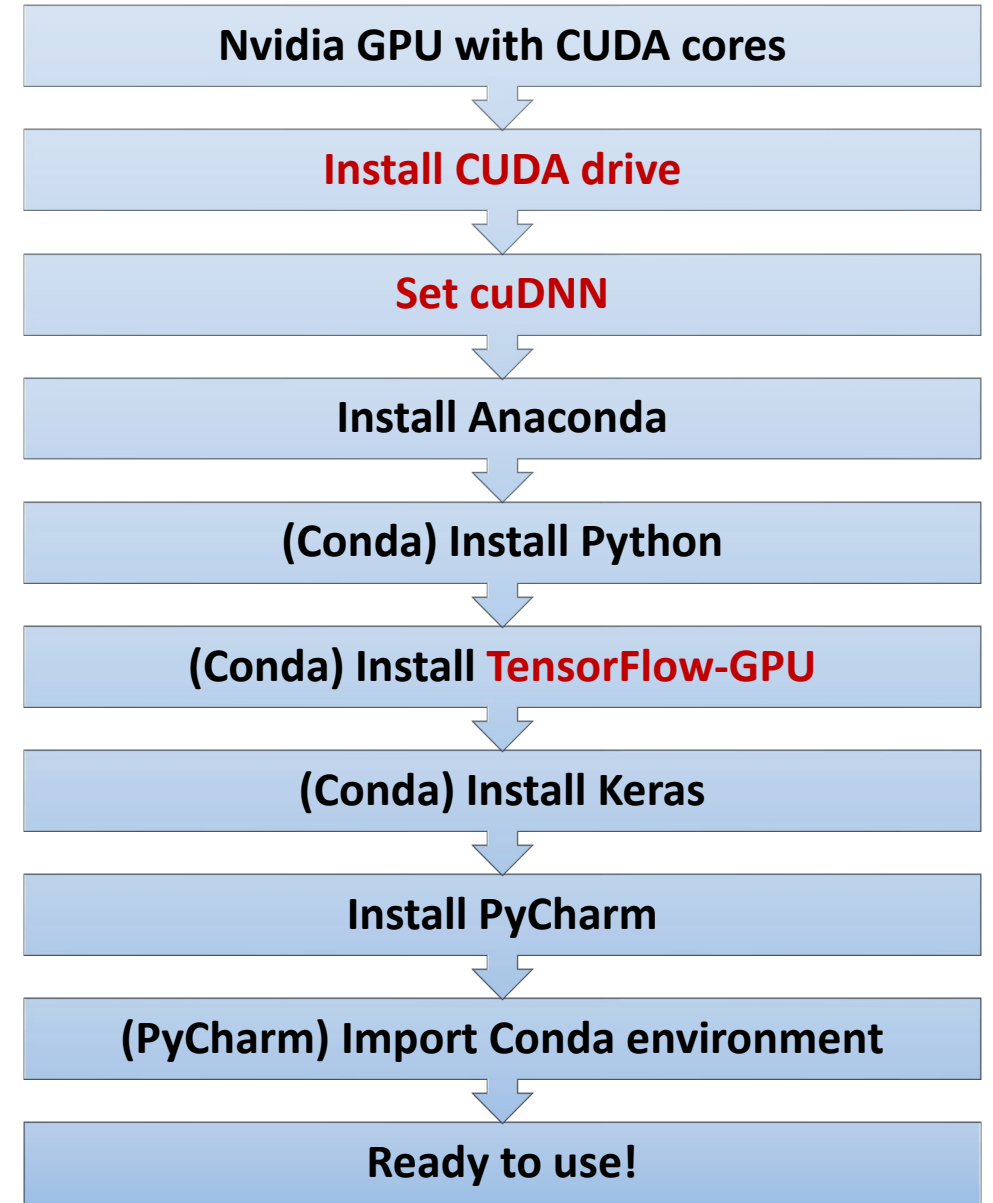
- **Virtual** environments for development
- Includes/controls: Frameworks + APIs + Languages



Configuration



DL using CPU



DL using GPU

Version problems

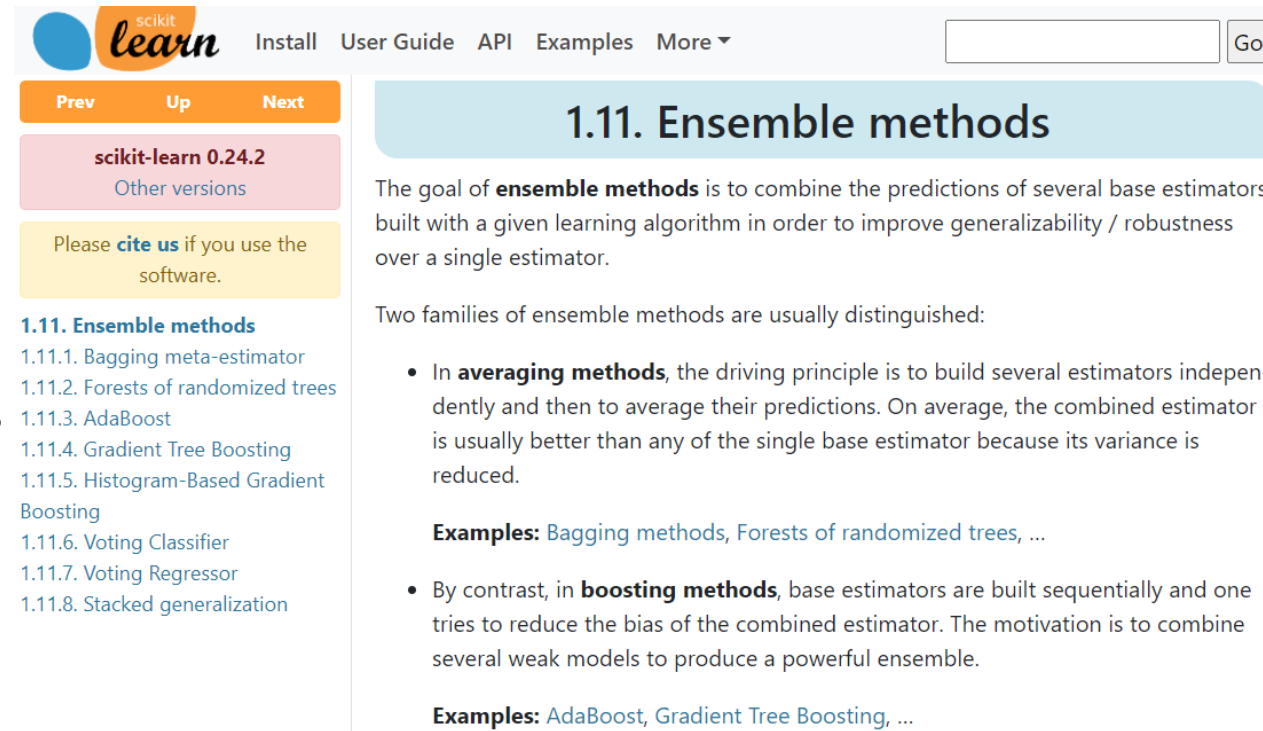
- OS (Win, Linux, macOS)
- CPU/GPU
- CUDA, cuDNN, framework, programming language,...

Version	Python version	cuDNN	CUDA
tensorflow_gpu-2.3.0	3.5-3.8	7.6	10.1
tensorflow_gpu-2.2.0	3.5-3.8	7.6	10.1
tensorflow_gpu-2.1.0	3.5-3.7	7.6	10.1
tensorflow_gpu-2.0.0	3.5-3.7	7.4	10
tensorflow_gpu-1.15.0	3.5-3.7	7.4	10
tensorflow_gpu-1.14.0	3.5-3.7	7.4	10
tensorflow_gpu-1.13.0	3.5-3.7	7.4	10
tensorflow_gpu-1.12.0	3.5-3.6	7	9
tensorflow_gpu-1.11.0	3.5-3.6	7	9
tensorflow_gpu-1.10.0	3.5-3.6	7	9
tensorflow_gpu-1.9.0	3.5-3.6	7	9
tensorflow_gpu-1.8.0	3.5-3.6	7	9
tensorflow_gpu-1.7.0	3.5-3.6	7	9
tensorflow_gpu-1.6.0	3.5-3.6	7	9
tensorflow_gpu-1.5.0	3.5-3.6	7	9
tensorflow_gpu-1.4.0	3.5-3.6	6	8
tensorflow_gpu-1.3.0	3.5-3.6	6	10 8

Windows GPU

Packages for ML/DL in Python

- **NumPy**: basic functions for math and matrix
- **SciPy**: scientific computing
- **Scikit-learn**: machine learning library
- **Matplotlib**: plotting library
- **OpenCV**: computer vision
- **Pandas**: data manipulation and analysis
- ...



The screenshot shows the Scikit-learn website interface. At the top, there is a navigation bar with links for 'Install', 'User Guide', 'API', 'Examples', and 'More'. Below the navigation bar, there are three buttons: 'Prev', 'Up', and 'Next'. The main content area is titled '1.11. Ensemble methods'. The text explains that the goal of ensemble methods is to combine the predictions of several base estimators to improve generalizability and robustness. It then lists two families of ensemble methods: averaging methods and boosting methods. Examples are provided for both families.

scikit-learn Install User Guide API Examples More

Prev Up Next

scikit-learn 0.24.2
Other versions

Please [cite us](#) if you use the software.

1.11. Ensemble methods

The goal of **ensemble methods** is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator.

Two families of ensemble methods are usually distinguished:

- In **averaging methods**, the driving principle is to build several estimators independently and then to average their predictions. On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced.
Examples: Bagging methods, Forests of randomized trees, ...
- By contrast, in **boosting methods**, base estimators are built sequentially and one tries to reduce the bias of the combined estimator. The motivation is to combine several weak models to produce a powerful ensemble.
Examples: AdaBoost, Gradient Tree Boosting, ...

Integrated development environment (IDE)

- **PyCharm**: comprehensive IDE, for large projects
- **VSCode**: light-weight IDE, support many languages
- **Jupyter Notebook**: interactive IDE, publish friendly
- **Spyder**: MATLAB-like, Anaconda built-in



Jupyter spectrogram (autosaved)

File Edit View Insert Cell Kernel Help Python 3

Simple spectral analysis

An illustration of the [Discrete Fourier Transform](#)

$$X_k = \sum_{n=0}^{N-1} x_n \exp^{-\frac{j2\pi kn}{N}} \quad k = 0, \dots, N-1$$

In [2]: `from scipy.io import wavfile
rate, x = wavfile.read('test_mono.wav')`

And we can easily view it's spectral structure using matplotlib's builtin specgram routine:

In [5]: `fig, (ax1, ax2) = plt.subplots(1,2,figsize=(16,5))
ax1.plot(x); ax1.set_title('Raw audio signal')
ax2.specgram(x); ax2.set_title('Spectrogram');`

Raw audio signal

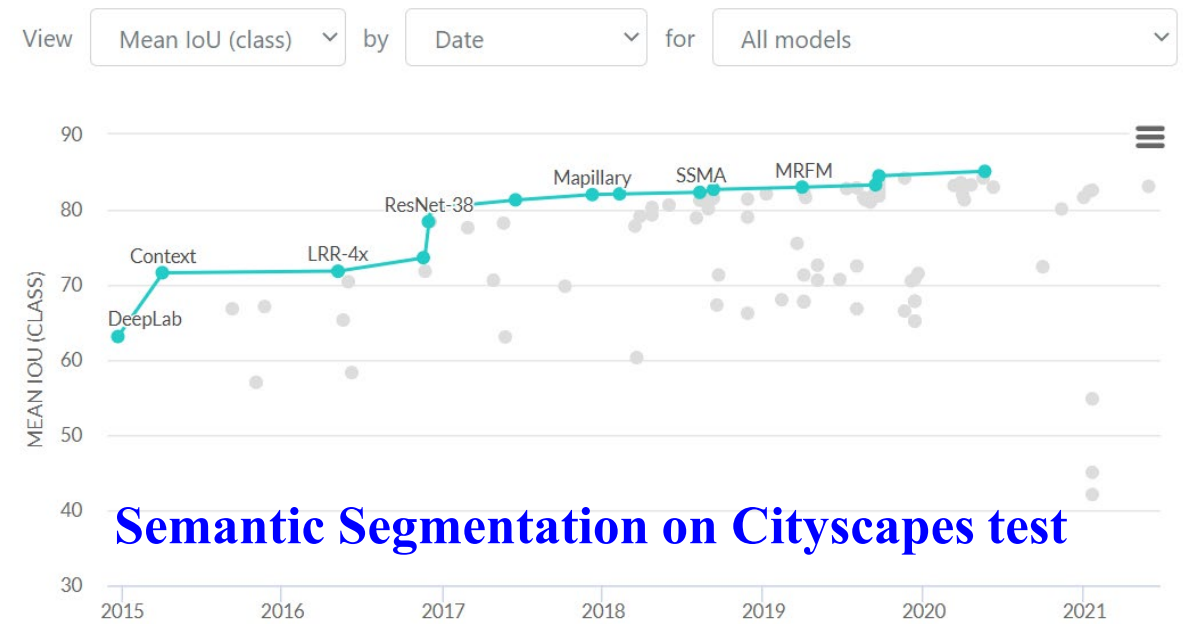
Spectrogram

Spyder (Python 3.6)

```
7 import pylab
8 from numpy import cos, linspace, pi, sin, random
9 from scipy.interpolate import splprep, splev
10
11 # XX Generate data for analysis
12
13 # Make ascending spiral in 3-space
14 t = linspace(0, 1.75 * 2 * pi, 100)
15
16 x = sin(t)
17 y = cos(t)
18 z = t
19
20 # Add noise
21 x += random.normal(scale=0.1, size=x.shape)
22 y += random.normal(scale=0.1, size=y.shape)
23 z += random.normal(scale=0.1, size=z.shape)
24
25
26 # XX Perform calculations
27
28 # Spline parameters
29 smoothness = 3.0 # Smoothness parameter
30 k_param = 2 # Spline order
31 nests = -1 # Estimate of number of knots needed (-1 = maximal)
32
33 # Find the knot points
34 knot_points, u = splprep((x, y, z), s=smoothness, k=k_param, nests=-1)
35
36 # Evaluate spline, including interpolated points
37 xnew, ynew, znew = splev(linspace(0, 1, 400), knot_points)
38
39
40 # XX Plot results
41
42 # TODO: Rewrite to avoid code smell
43 pylab.subplot(2, 2, 1)
44 data, = pylab.plot(x, y, 'bo-', label='Data with X-Y Cross Section')
45 fit, = pylab.plot(xnew, ynew, 'r-', label='Fit with X-Y Cross Section')
46 pylab.legend()
47 pylab.xlabel('x')
48 pylab.ylabel('y')
49
50 pylab.subplot(2, 2, 2)
51 data, = pylab.plot(x, z, 'bo-', label='Data with X-Z Cross Section')
52 fit, = pylab.plot(xnew, znew, 'r-', label='Fit with X-Z Cross Section')
53 pylab.legend()
54 pylab.xlabel('x')
```

Websites for ML/DL

- Guidebooks of each tools/software online – look-up books
- Github – without reinventing the wheel
- **Stack Overflow** – solve problems
- Kaggle – datasets
- **paperswithcode.com** – SOTA performance ranks with papers & codes
- Google – everything!
- ...



Semantic Segmentation on Cityscapes test

Rank	Model	Mean ↑ IoU (class)	Category mIoU	Extra Training Data	Paper	Code	Result	Year
1	HRNet-OCR (Hierarchical Multi-scale Attention)	85.1%		✓	Hierarchical Multi-Scale Attention for Semantic Segmentation	🔗	📄	2020
2	HRNetV2+ OCR+	84.5%		✓	Segmentation Transformer: Object- Contextual Representations for Semantic Segmentation	🔗	📄	2019
3	EfficientPS	84.21%		✓	EfficientPS: Efficient Panoptic Segmentation	🔗	📄	2020

HPC at GW



- One CPU node
 - Dual 20-Core 3.70GHz Intel Xeon processors
 - 192GB RAM
 - 800 GB SSD
- One GPU node
 - 2 NVIDIA **Tesla V100** GPU (4 for large nodes)
 - Dual 20-Core 3.70GHz Intel Xeon (18-Core Xeon for large nodes)
 - 192GB RAM (384GB for large nodes)
 - 800 GB SSD
- High throughput node, High memory node (3TB RAM!),...

Tesla V100 highlight

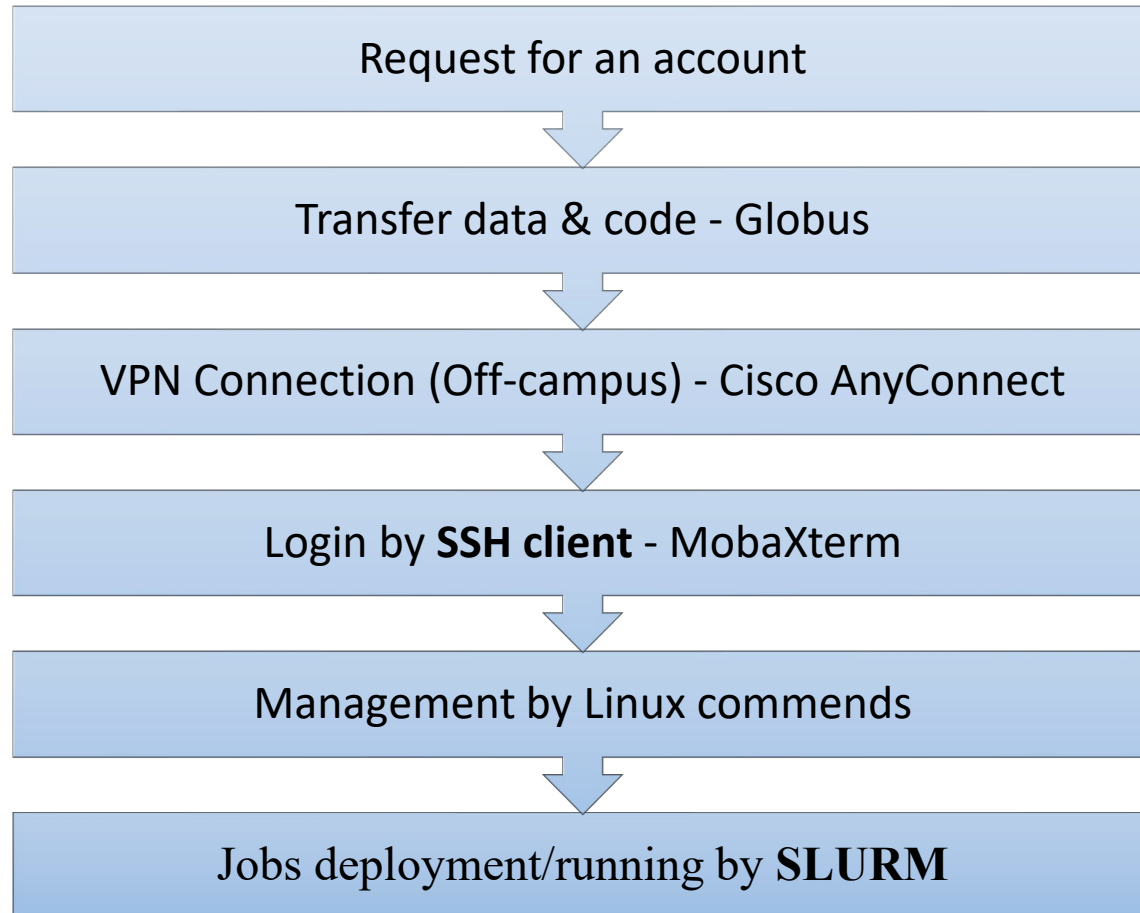
640 Tensor Cores

5120 CUDA Cores

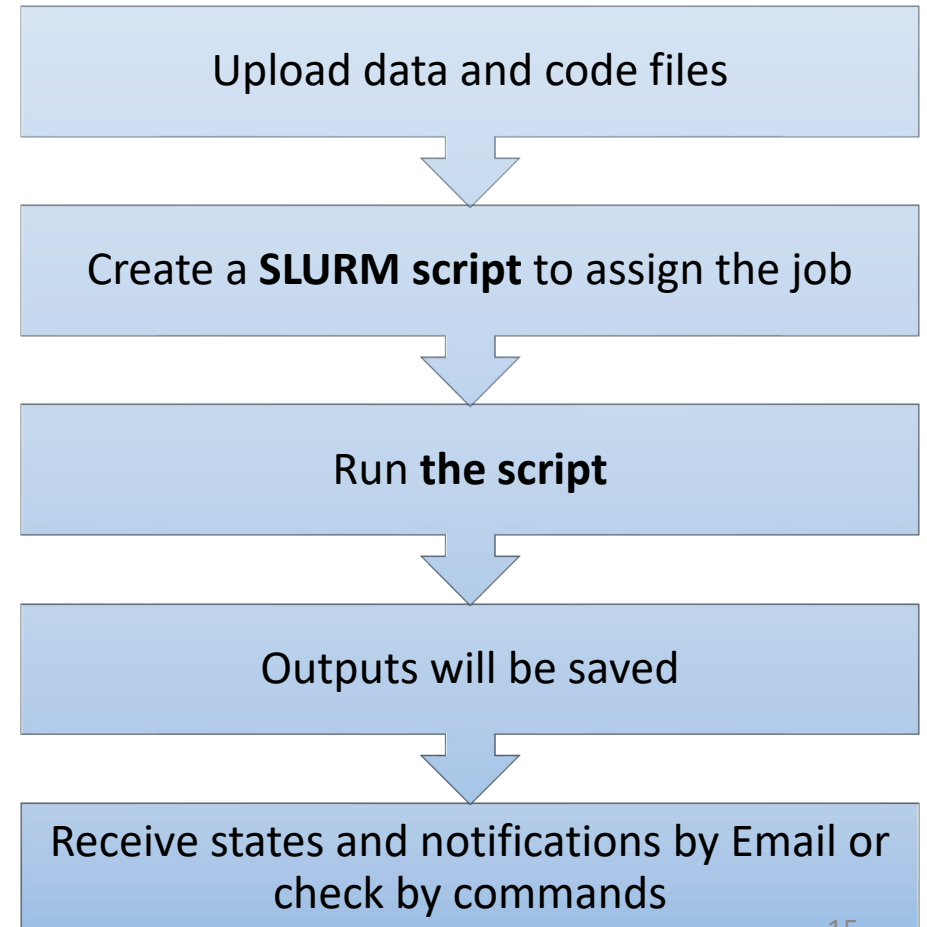
16 GB Memory

How to use the HPC at GW

Overview



Workflow



Epilogue

- The Medical Imaging & Image Analysis Laboratory

SEH 5290

W: loewlab.seas.gwu.edu



- Detailed instruction of GWU HPC

https://loewlab.seas.gwu.edu/files/2020/11/mia_GW_HPC_intro.pdf

The Medical Imaging & Image Analysis (MIA) Laboratory, 2021