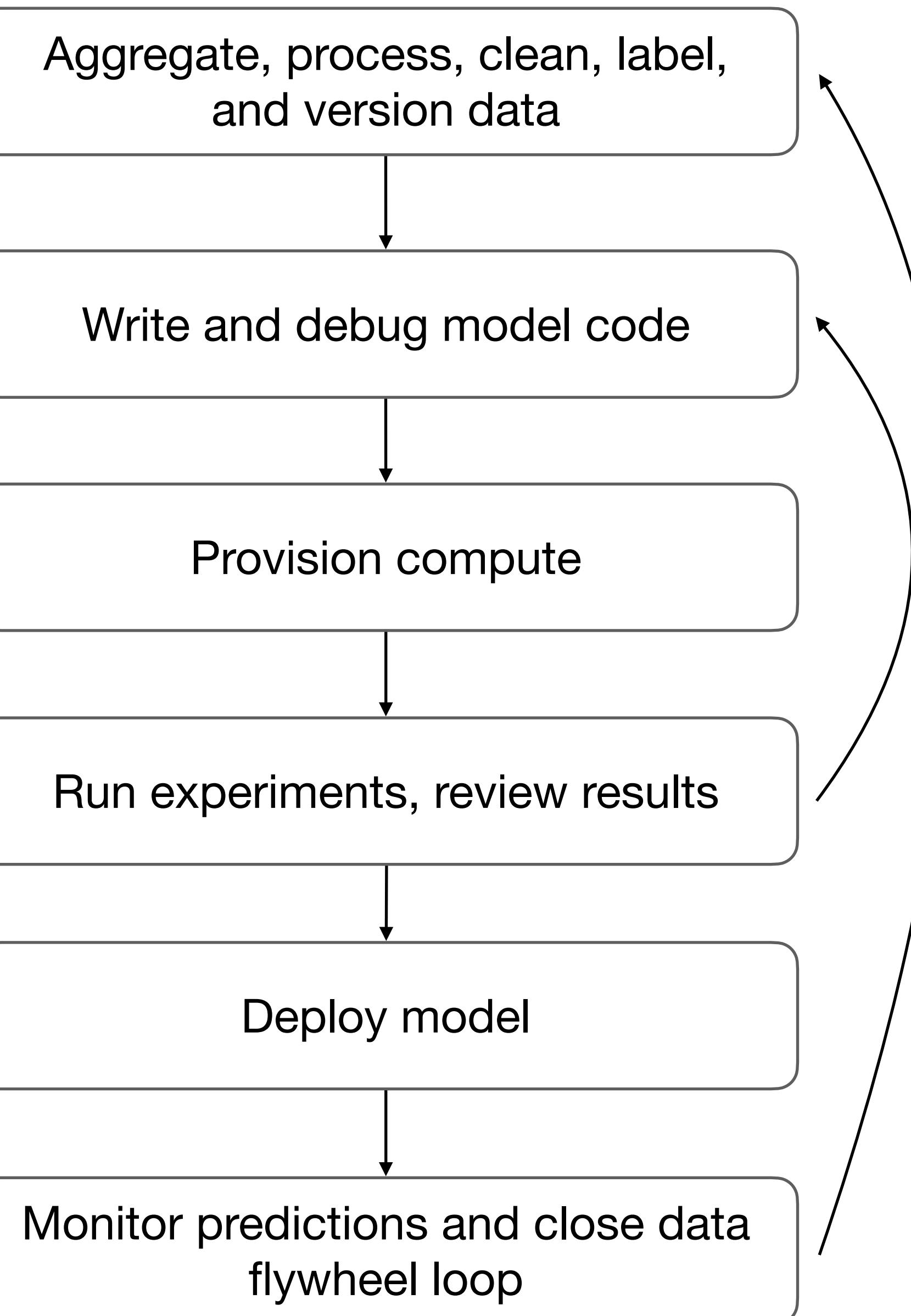
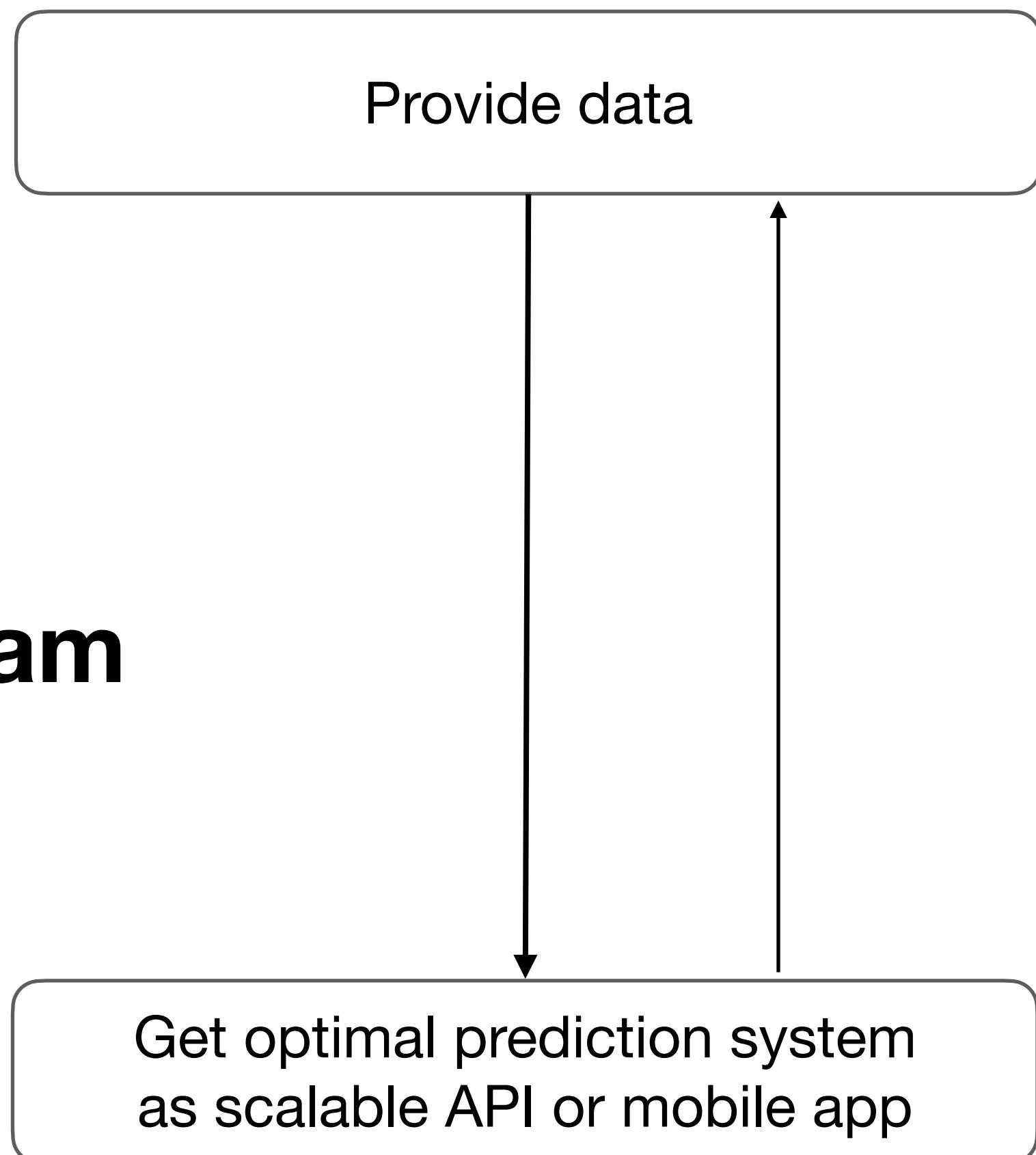


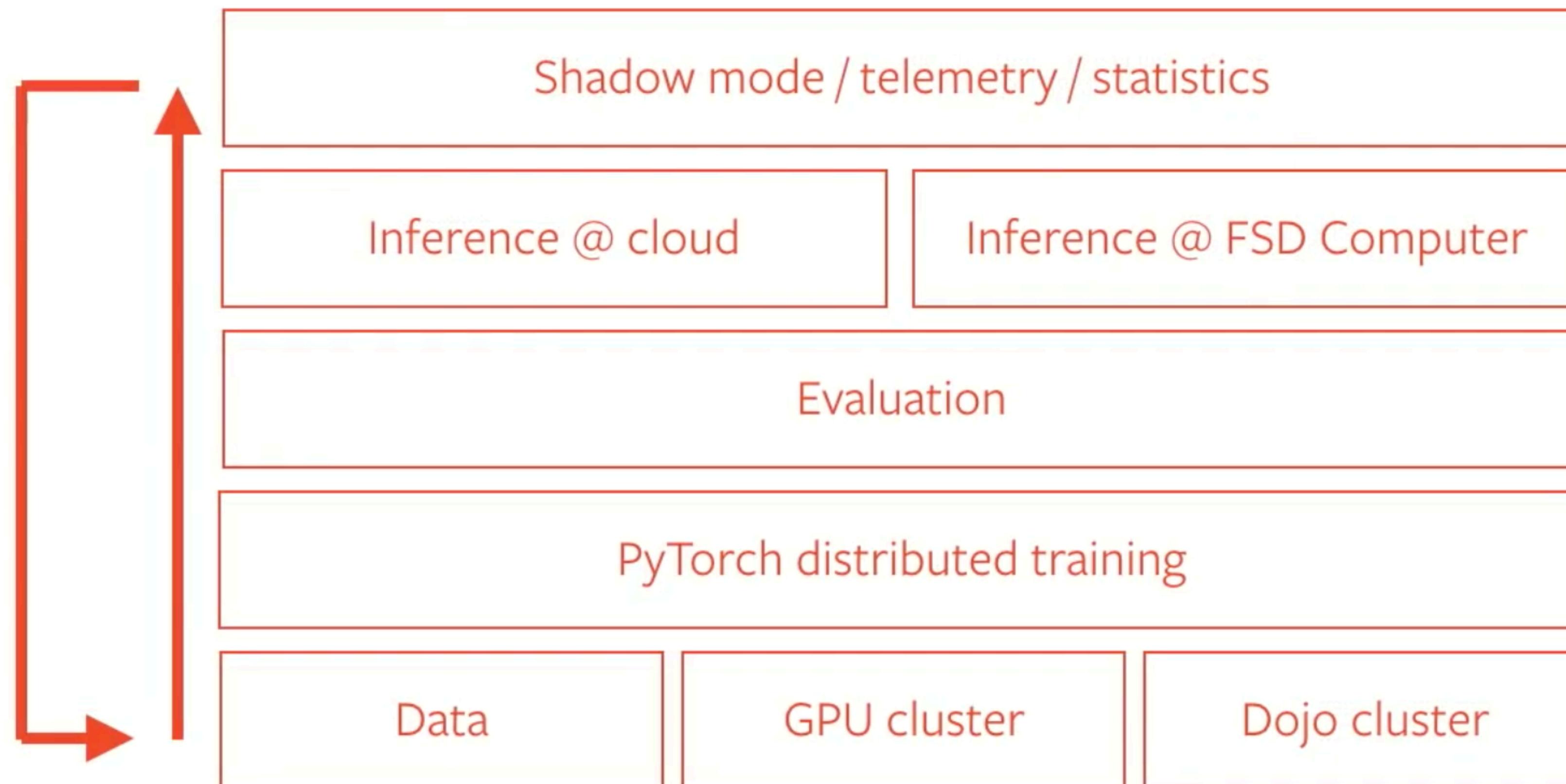
Infrastructure & Tooling

Dream

Reality



“OPERATION VACATION”



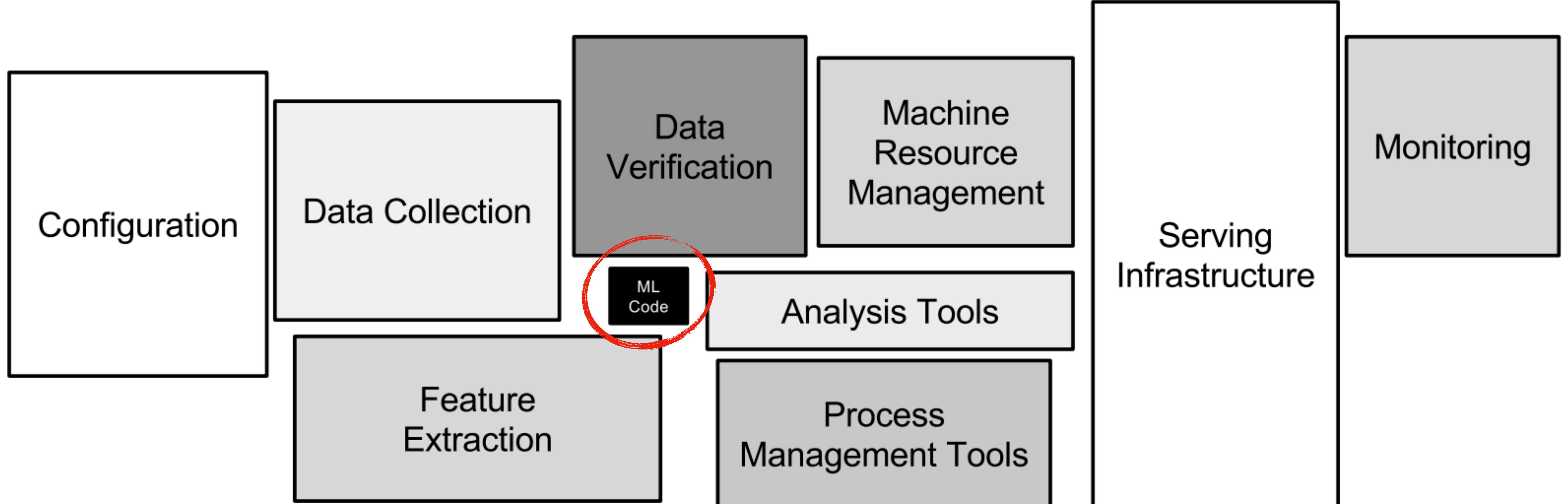
Goal: add data, see model improve

Andrei Karpathy at PyTorch Devcon 2019 - <https://www.youtube.com/watch?v=oBklltKXtDE>

Machine Learning: The High-Interest Credit Card of Technical Debt

**D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov,
Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young**
{dsculley, gholt, dgg, edavydov}@google.com
{toddphillips, ebner, vchaudhary, mwyoung}@google.com
Google, Inc

SE4ML: Software Engineering for Machine Learning (NIPS 2014)



Machine Learning: The High-Interest Credit Card of Technical Debt

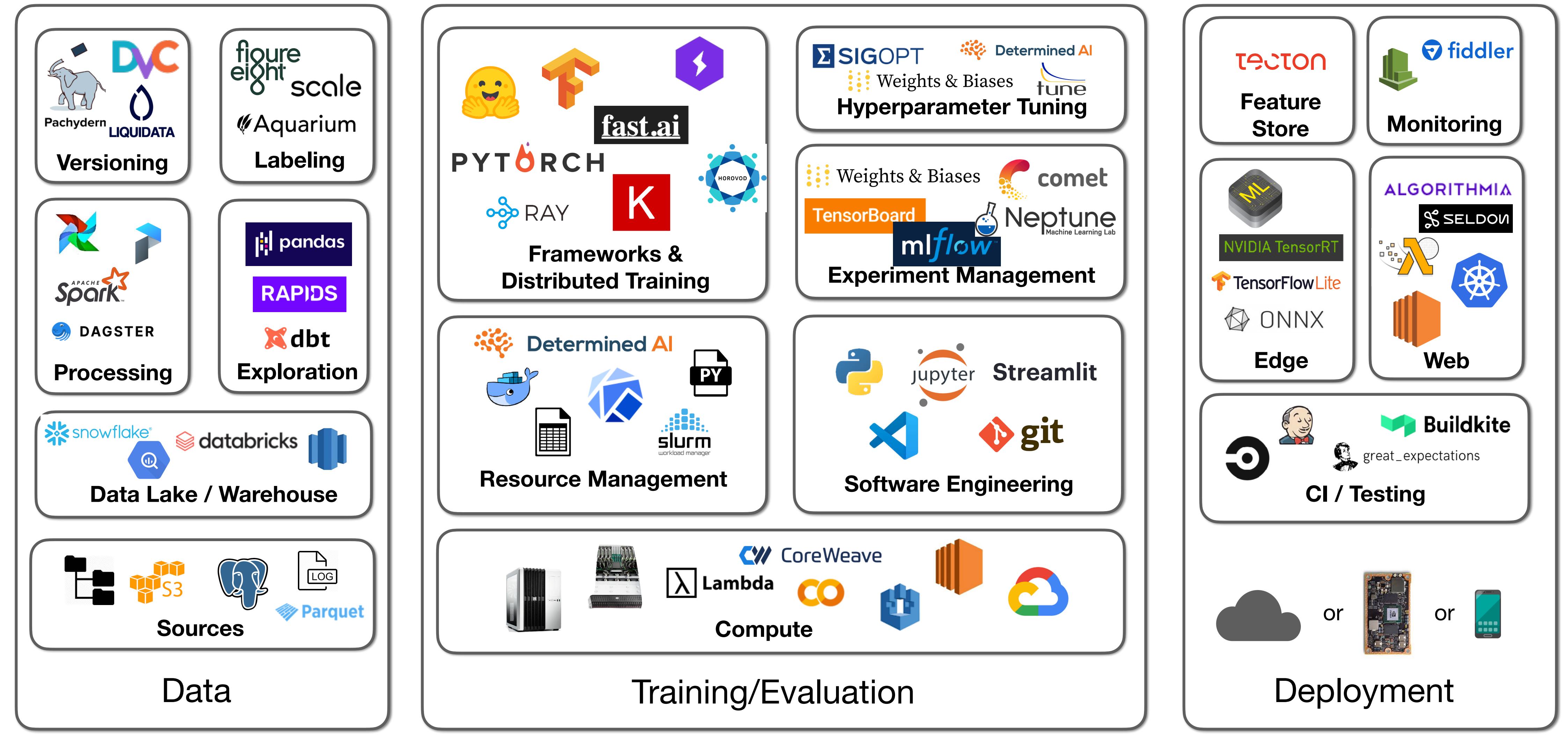
D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov,
Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young



Amazon SageMaker

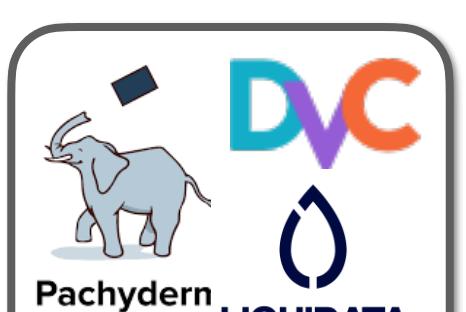


“All-in-one”





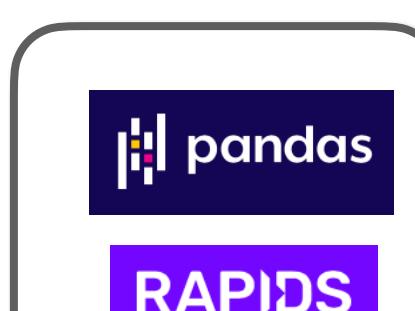
"All-in-one"



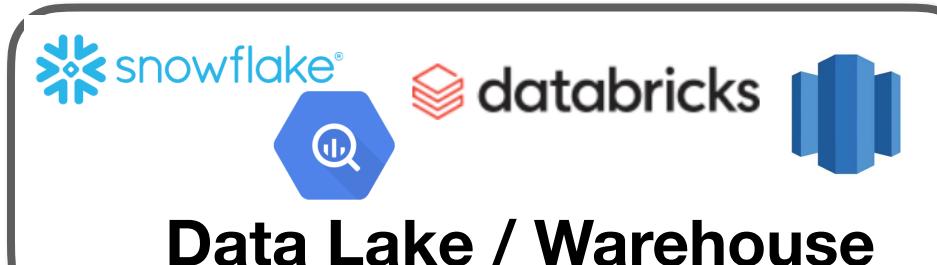
Versioning



Processing



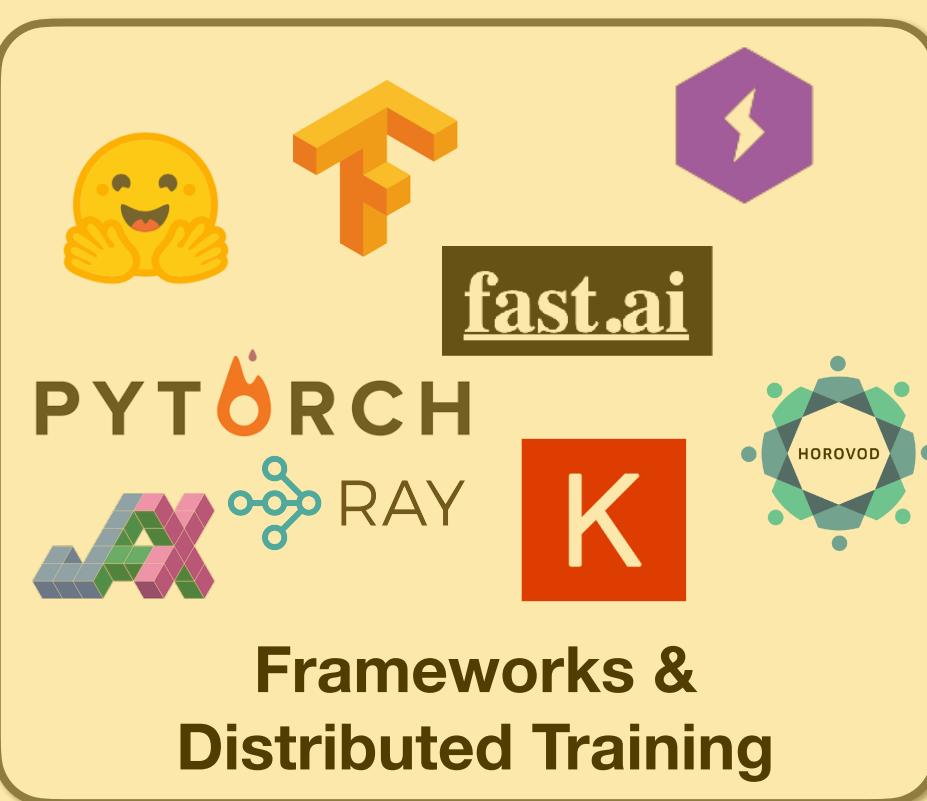
Exploration



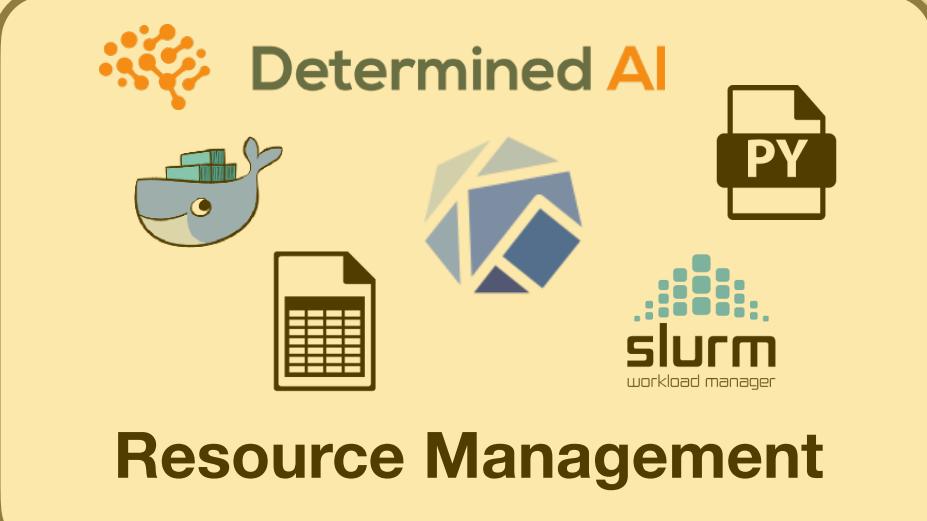
Data Lake / Warehouse



Data



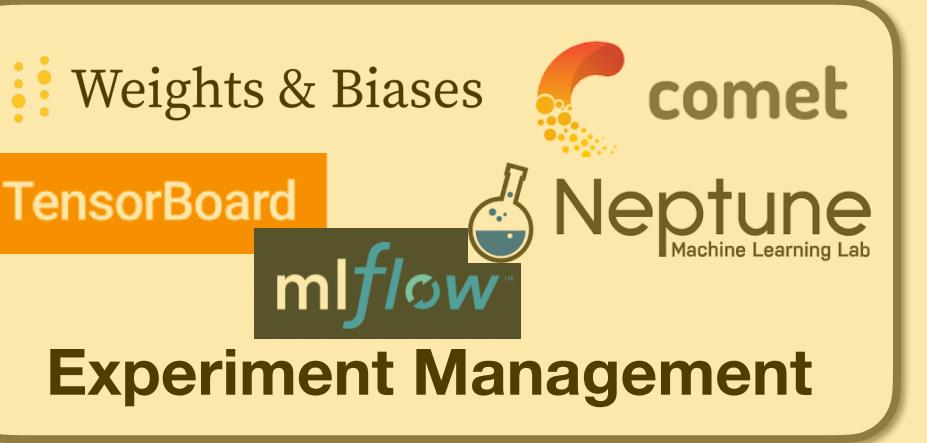
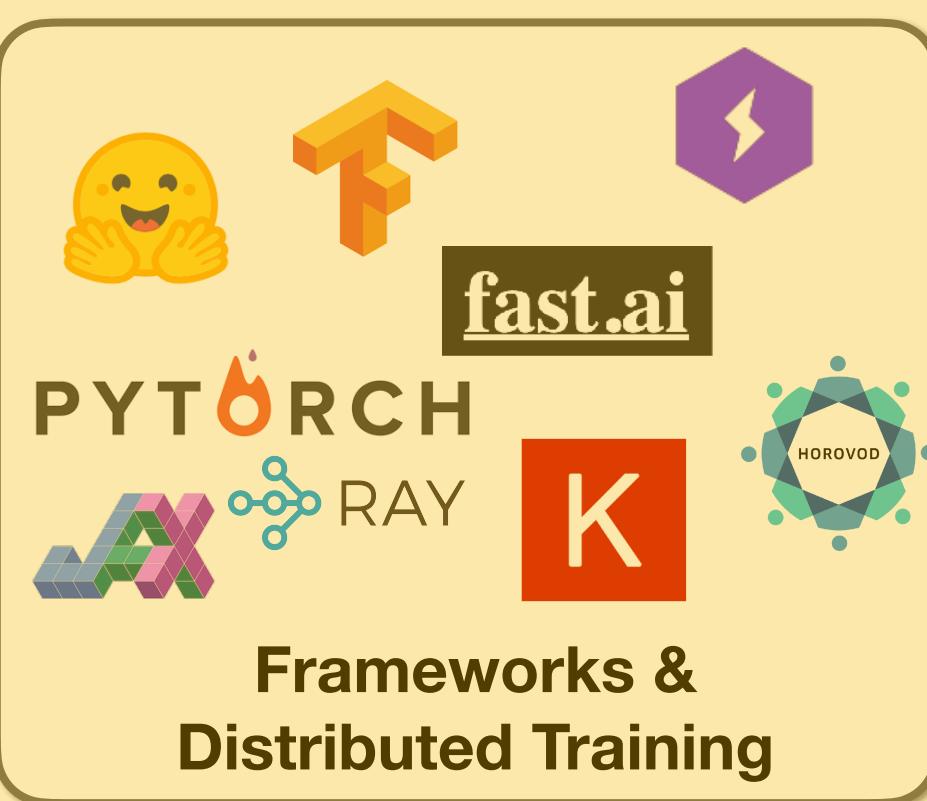
Frameworks & Distributed Training



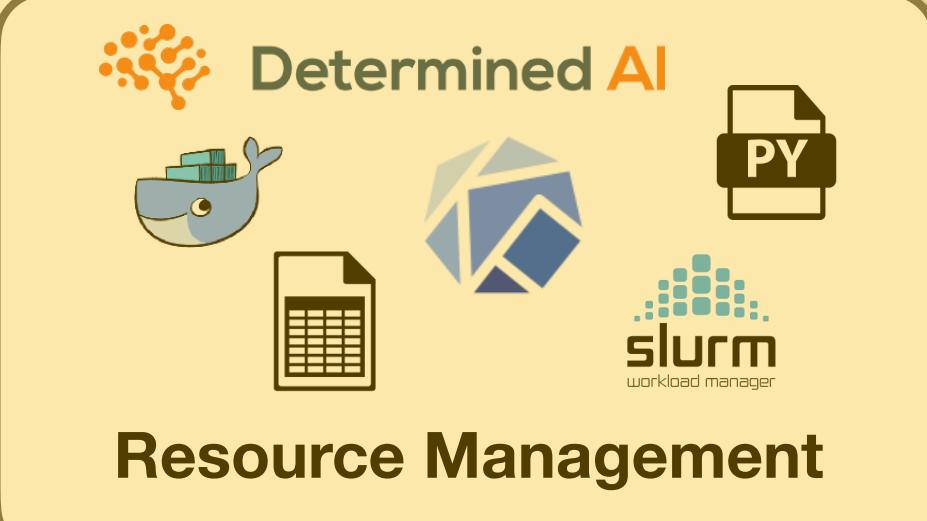
Resource Management



Compute



Experiment Management



Software Engineering



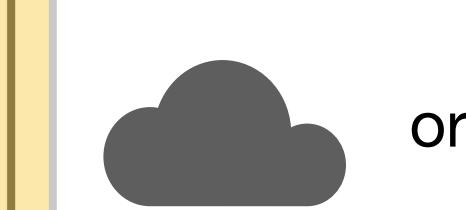
Training/Evaluation



Edge



CI / Testing



or
or
or

Deployment

Questions?



Amazon SageMaker

gradient^o
by Paperspace

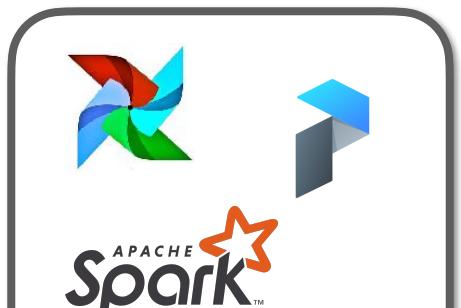
FLOYD

DOMINO
DATA LAB

“All-in-one”



Versioning



Processing



Data Lake / Warehouse



Data



Frameworks &
Distributed Training



Resource Management



Compute

Training/Evaluation



Hyperparameter Tuning



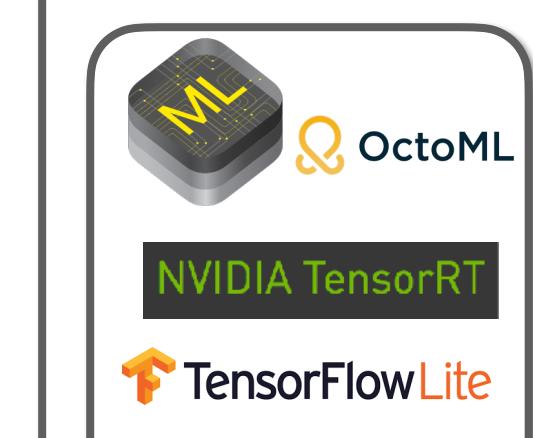
Experiment Management



Software Engineering



Tecton Feature Store



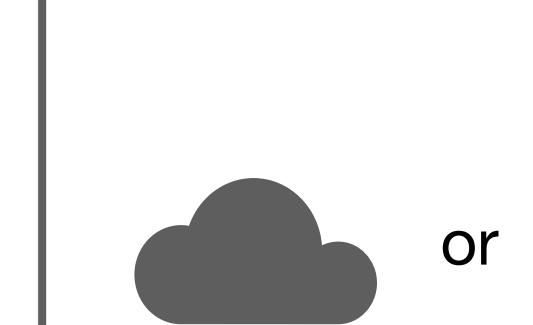
ONNX Edge



Monitoring



CI / Testing

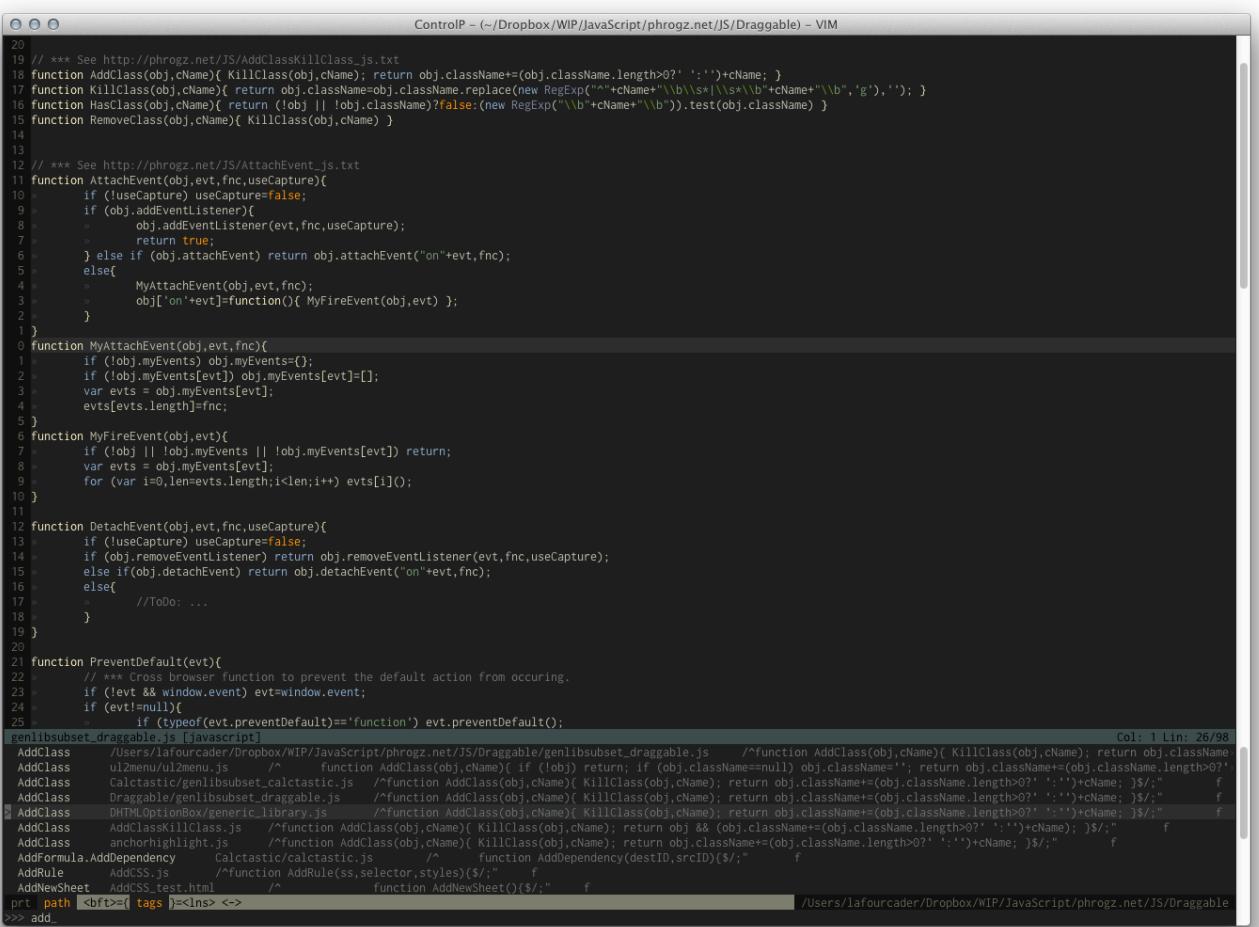


Deployment

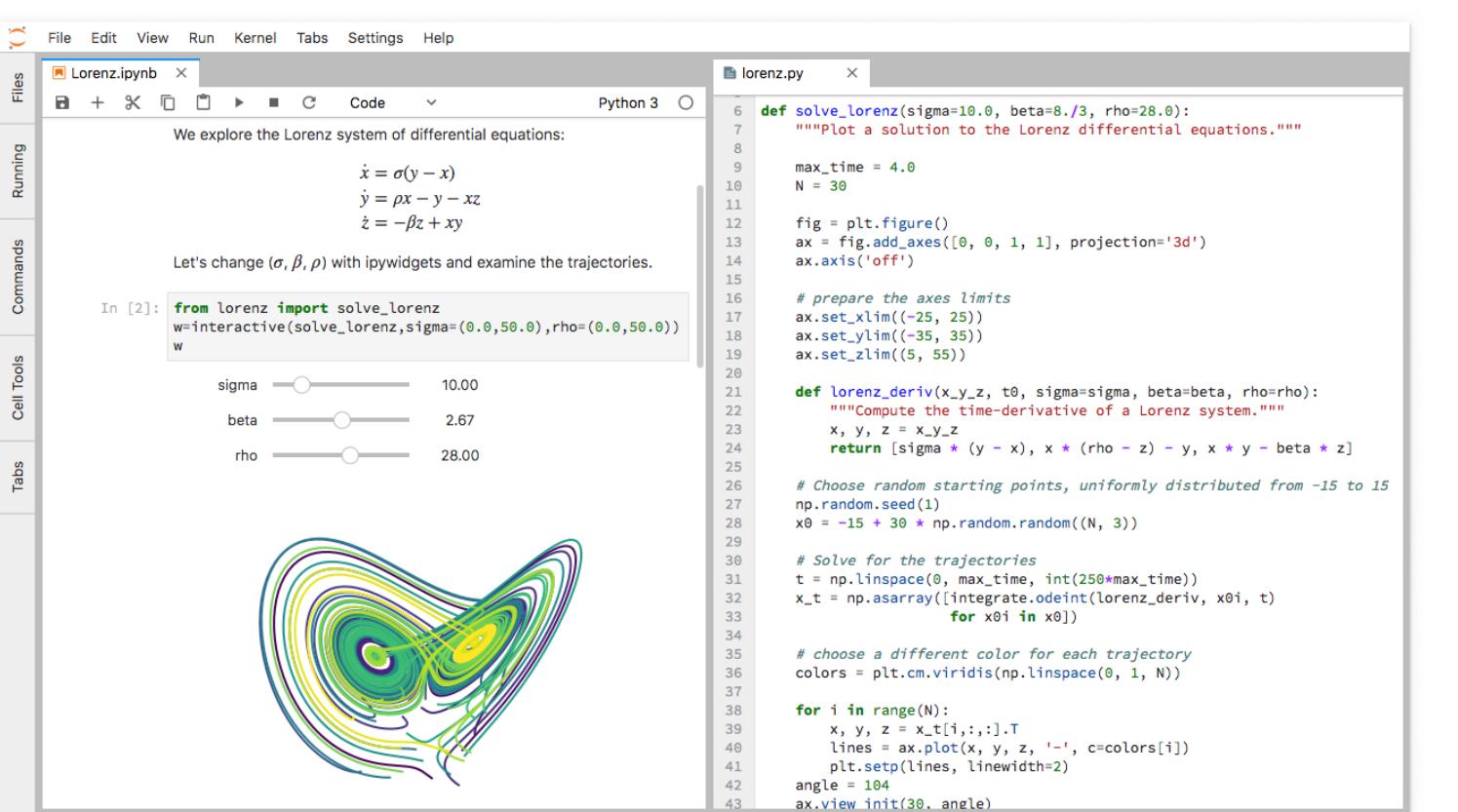
Programming Language

- Python, because of the libraries
 - Clear winner in scientific and data computing

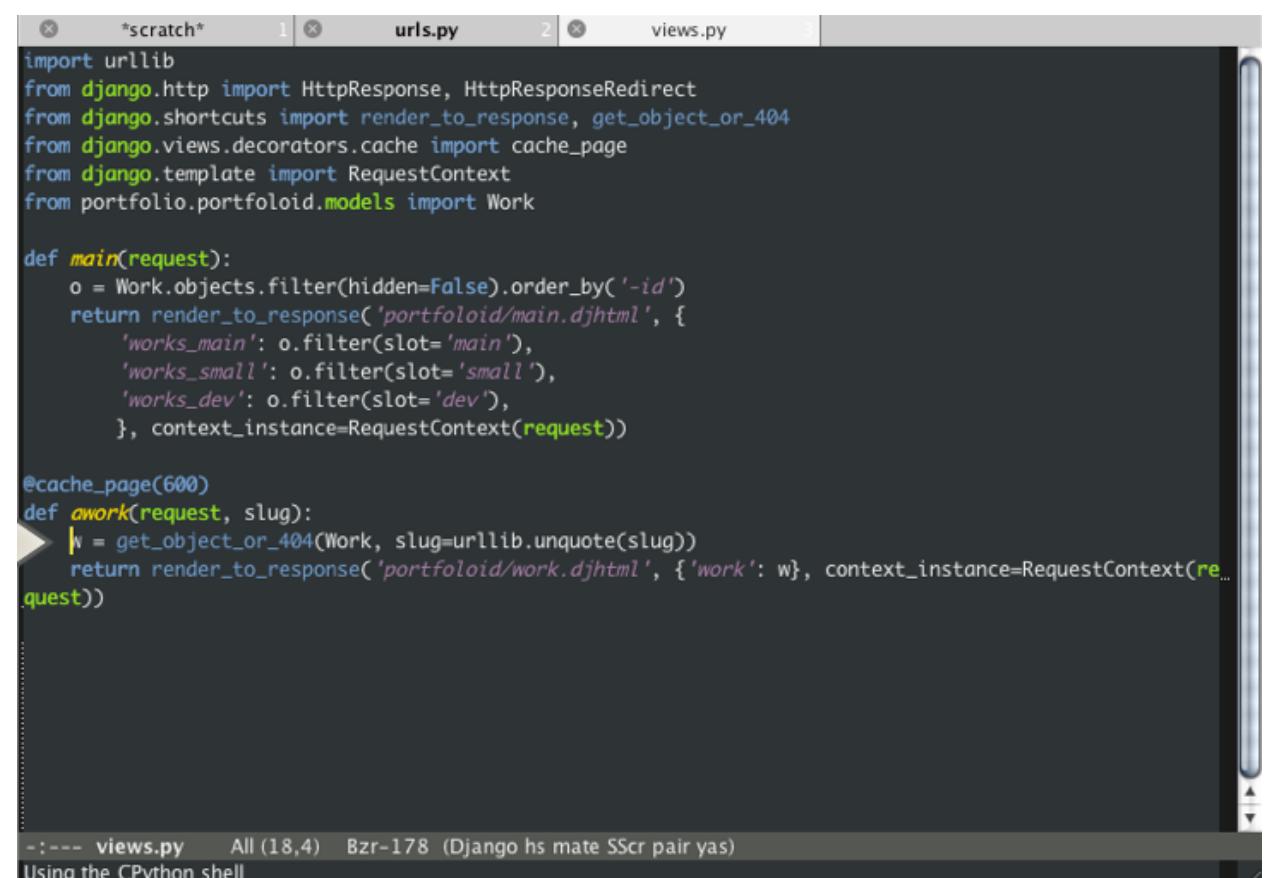
Editors



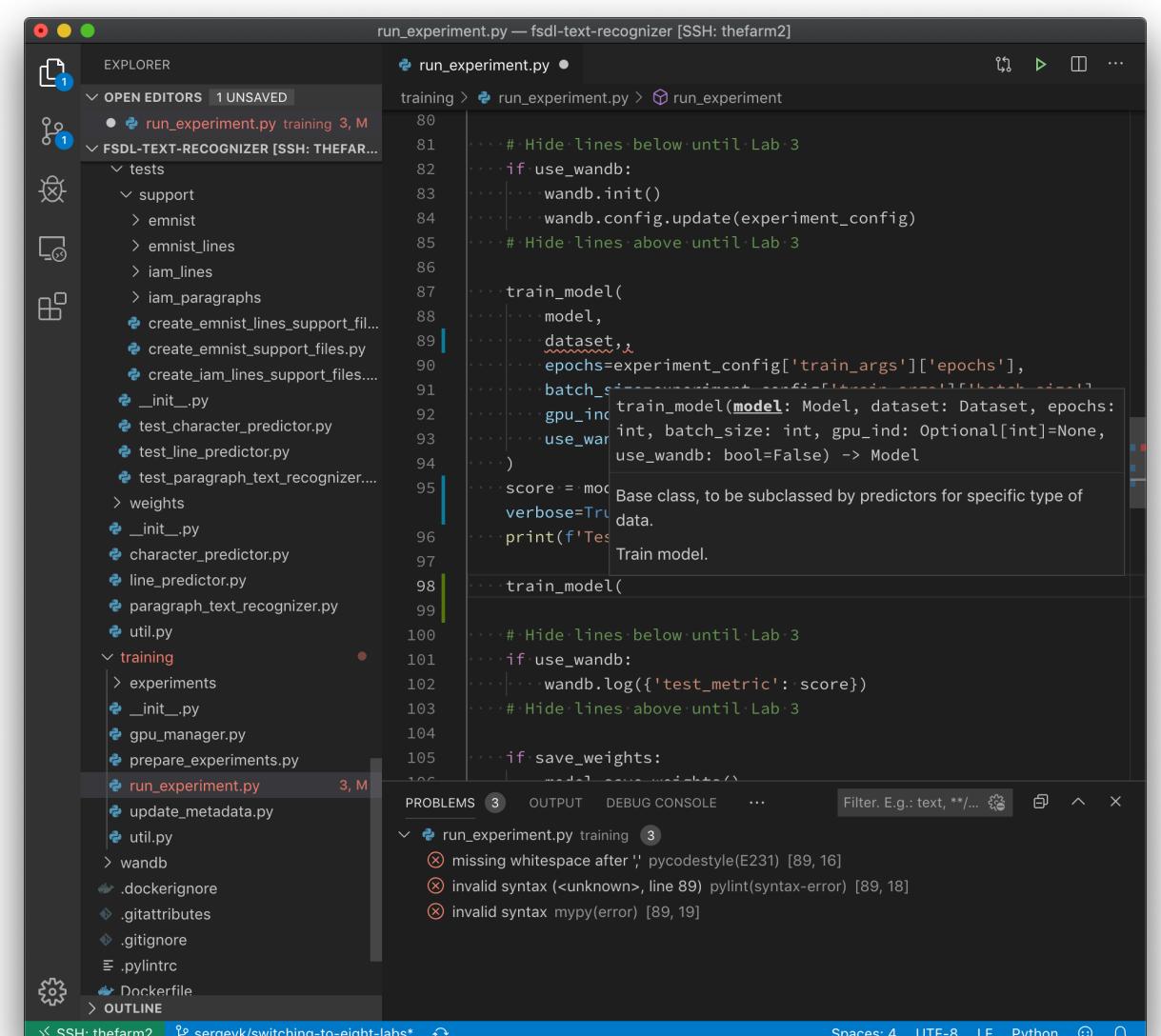
Vim



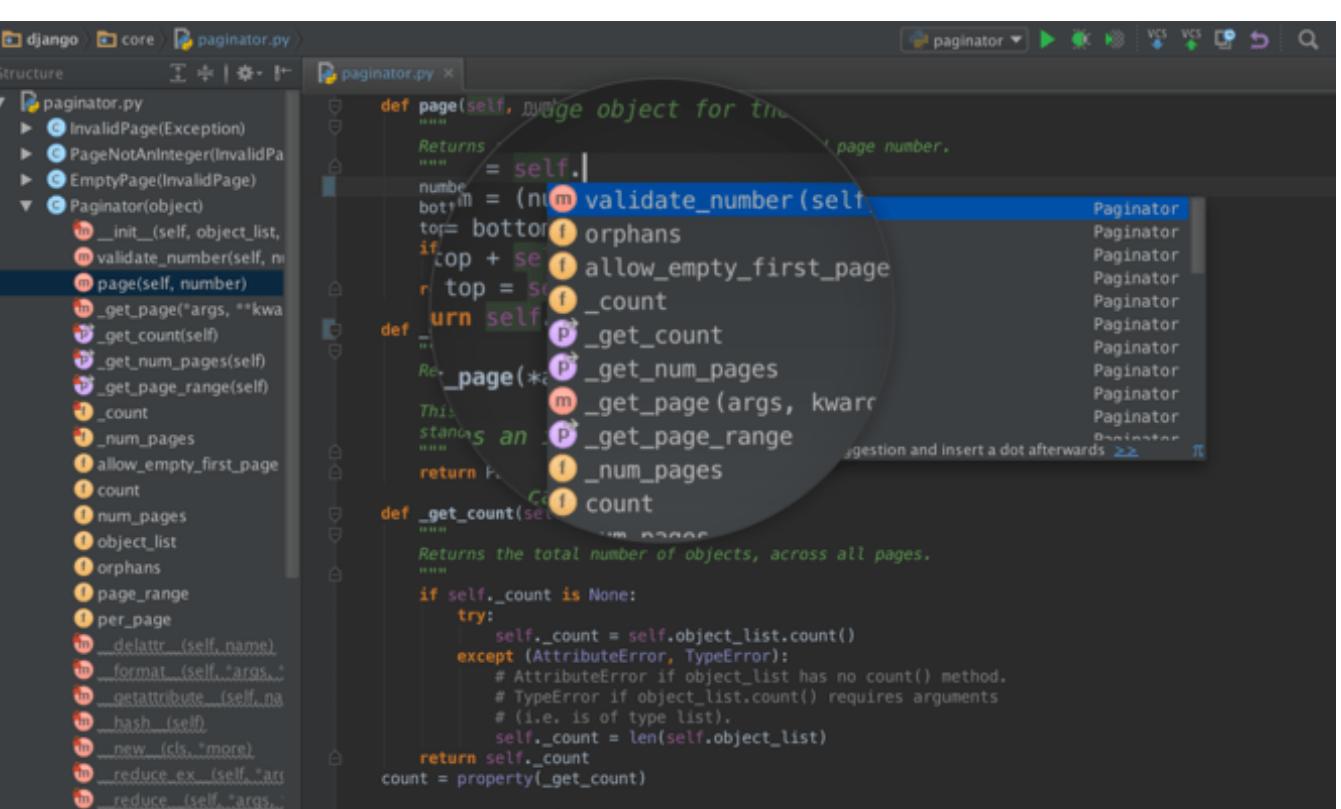
Jupyter



Emacs



VS Code



PyCharm

Visual Studio Code

Built-in git staging and diffing →

Peek documentation

- VS Code makes for a very nice Python experience

Lint code as you write

Open whole projects remotely

The screenshot shows the Visual Studio Code interface with several annotations:

- A red arrow points from the text "Built-in git staging and diffing" to the Git icon in the Explorer sidebar.
- A red arrow points from the text "Peek documentation" to the hover documentation feature over a Python method call in the editor.
- A red arrow points from the text "Lint code as you write" to the Problems panel, which displays three linting errors.
- A red arrow points from the text "Open whole projects remotely" to the SSH connection status bar at the bottom.

The main code editor shows a Python file named `run_experiment.py` with some code related to training a model using WandB. The Problems panel on the right lists the following errors:

- missing whitespace after ',' pycodestyle(E231) [89, 16]
- invalid syntax (<unknown>, line 89) pylint(syntax-error) [89, 18]
- invalid syntax mypy(error) [89, 19]

Visual Studio Code

The screenshot shows a Visual Studio Code interface with several red annotations:

- A red arrow points to the "REMOTE EXPLORER" icon in the sidebar.
- A red circle highlights the "SSH Targets" section in the sidebar, which lists "bigboy.local" and its contents: pandagrade, fsdl-text-recognizer-2021, roster-matching, thefarm2, pandagrade, and megatron. A red circle also highlights the "bigboy.local" entry.
- A red arrow points to the "ORTS" section in the sidebar, which lists various ports (e.g., 5900, 8888, 33405) and their corresponding host:port mappings.
- A red circle highlights the "line_cnn_lstm.py" file in the main editor area.
- A red arrow points to the "Parameters" section at the bottom of the code editor.
- A red circle highlights the "TERMINAL" tab in the top bar.
- A red arrow points to the terminal output, which shows WandB logs and a run summary.
- A red circle highlights the status bar at the bottom, which shows "SSH: bigboy.local" and other system information.

Annotations in red text:

- Built-in git staging and diffing
- Open projects remotely
- Notebook port forwarding
- Use the terminal

Code editor content (line_cnn_lstm.py):

```
1 from typing import Any, Dict
2 import argparse
3 import torch
4 import torch.nn as nn
5 import torch.nn.functional as F
6
7 from .line_cnn import LineCNN
8 from .line_cnn2 import LineCNN2
9
10 LINE_CNN_TYPE = 1
11 LSTM_DIM = 512
12 LSTM_LAYERS = 2
13 LSTM_DROPOUT = 0.2
14
15
16 class LineCNNLSTM(nn.Module):
17     """Process the line through a CNN and process the resulting sequence through LSTM layers."""
18
19     def __init__(self, data_config: Dict[str, Any], args: argparse.Namespace = None,
20                  **kwargs):
21         super().__init__()
22         self.data_config = data_config
23         self.args = vars(args) if args is not None else {}
24
25         self.max_output_length = data_config["output_dims"][0]
26
27         num_classes = len(data_config["mapping"])
28         line_cnn_type = self.args.get("line_cnn_type", LINE_CNN_TYPE)
29         lstm_dim = self.args.get("lstm_dim", LSTM_DIM)
30         lstm_layers = self.args.get("lstm_layers", LSTM_LAYERS)
31         lstm_dropout = self.args.get("lstm_dropout", LSTM_DROPOUT)
32
33         # LineCNN outputs (B, C, S) log probs, with C == num_classes
34         if line_cnn_type == 1:
35             self.line_cnn = LineCNN(data_config=data_config, args=args)
36         elif line_cnn_type == 2:
37             self.line_cnn = LineCNN2(data_config=data_config, args=args)
38
39         self.lstm = nn.LSTM(
40             input_size=num_classes,
41             hidden_size=lstm_dim,
42             num_layers=lstm_layers,
43             dropout=lstm_dropout,
44             bidirectional=True,
45         )
46         self.fc = nn.Linear(lstm_dim, num_classes)
47
48     def forward(self, x: torch.Tensor) -> torch.Tensor:
49         Parameters
50
51         def forward(self, x: torch.Tensor) -> torch.Tensor:
52             Parameters
53             Parameters
54             Parameters
55             Parameters
```

Terminal output (WandB logs):

```
wandb: Waiting for W&B process to finish, PID 17640
wandb: Program ended successfully.
wandb: Find user logs for this run at: /home/sergeyk/work/fsdl/fsdl-text-recognizer-2021/wandb/run-20210222_220519-12gbskw5/logs/debug.log
wandb: Find internal logs for this run at: /home/sergeyk/work/fsdl/fsdl-text-recognizer-2021/wandb/run-20210222_220519-12gbskw5/logs/debug-internal.log
wandb: Run summary:
wandb: Run history:
wandb: lr-Adam 1e-05
wandb: train_loss 1.50937
wandb: epoch 36
wandb: _step 5327
wandb: _runtime 3284
wandb: _timestamp 1614063603
wandb: val_loss 1.87379
wandb: val_acc 0.04475
wandb: val_cer 0.96612
wandb: test_acc 0.04198
wandb: test_cer 0.96892
```

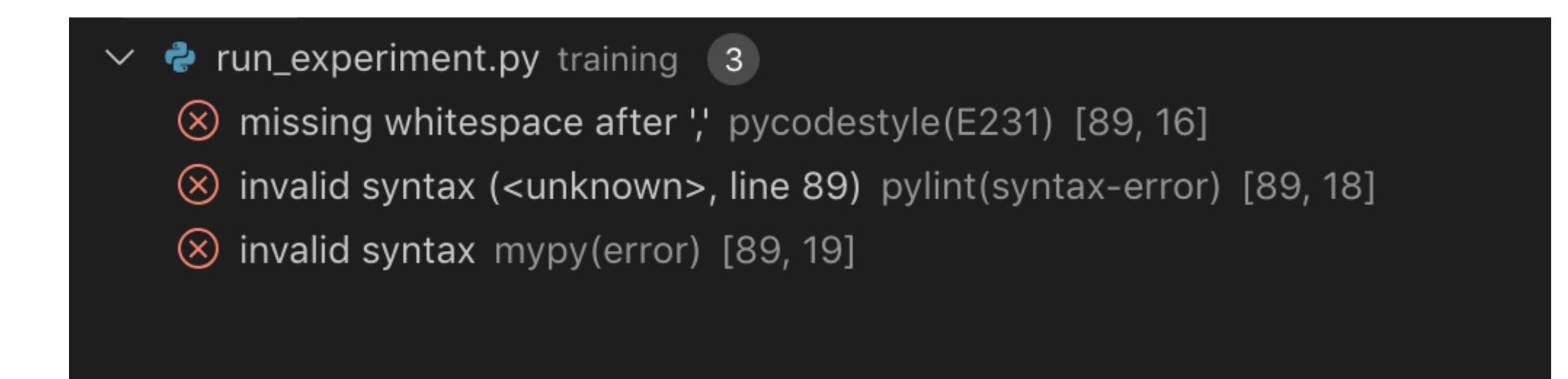
Status bar:

```
SSH: bigboy.local  main*  0 21 Python 3.6.12 64-bit  4 9 13  20 -- NORMAL --
```

Linters and Type Hints

- Whatever code style rules can be codified, should be
 - Static analysis can catch some bugs
 - Static type checking both documents code and catches bugs
 - Will see in Lab 7

```
87     ....train_model(  
88         ....model,  
89         ....dataset,,  
90         ....epochs=experiment_config['train_args']['epochs'],  
91         ....batch_size=batch_size,  
92         ....gpu_ind=gpu_index,  
93         ....use_wandb=use_wandb)  
94     )  
95     ....score = model.score()  
96     ....verbose=True  
97     ....print(f'Test score: {score}')  
98     ....train_model(  
99         ....model,
```



Jupyter Notebooks

- Notebooks have become fundamental to data science
 - Great as the "first draft" of a project
 - Jeremy Howard from fast.ai good to learn from ([course.fast.ai](#) videos)
 - Difficult to make scalable, reproducible, well-tested

The screenshot shows a Jupyter Notebook interface with two code cells and two corresponding plots.

Code Cell 1:

```
[7]: import altair as alt
from vega_datasets import data

cars = data.cars()
```

Code Cell 2:

```
[8]: brush = alt.selection(type='interval', resolve='global')

base = alt.Chart(cars).mark_point().encode(
    y='Miles_per_Gallon',
    color=alt.condition(brush, 'Origin', alt.ColorValue('gray'))
).add_selection(
    brush
).properties(
    width=250,
    height=250
)

print("Select a region in the chart below to try this out!")

base.encode(x='Horsepower') | base.encode(x='Acceleration')
```

Faceted Scatter Plot with Linked Brushing

This is an example of using an interval selection to control the color of points across multiple facets.

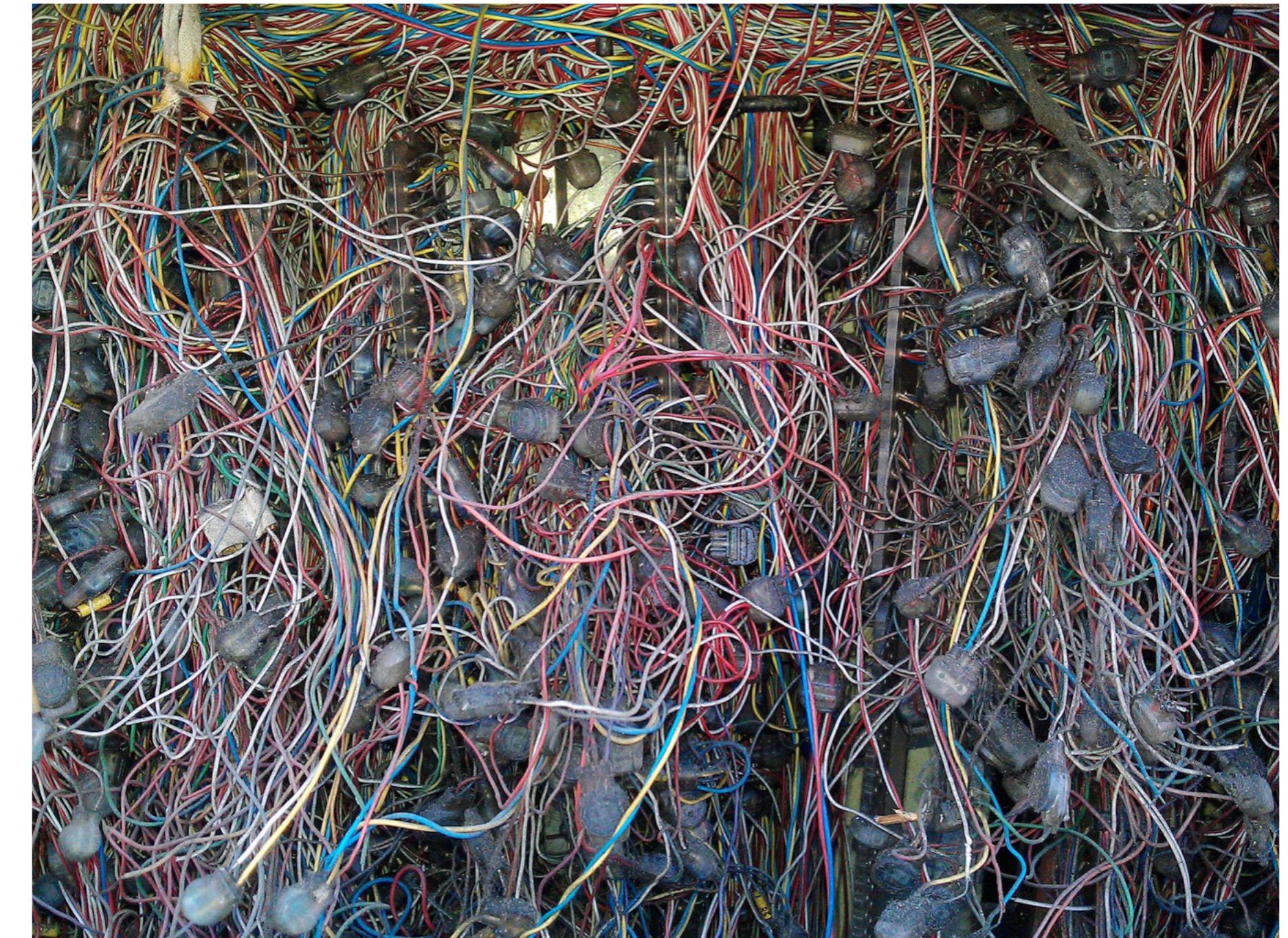
The notebook displays two scatter plots side-by-side. Both plots have 'Miles_per_Gallon' on the y-axis (ranging from 0 to 50). The left plot has 'Horsepower' on the x-axis (ranging from 0 to 200), and the right plot has 'Acceleration' on the x-axis (ranging from 0 to 25). Data points are colored by 'Origin': Europe (blue), Japan (orange), and USA (red). A global interval selection is applied to both plots. When a region is selected in one plot, the corresponding points in the other plot are also highlighted in gray.

Problems with notebooks

- Hard to version
- Notebook "IDE" is primitive
- Very hard to test
- Out-of-order execution artifacts
- Hard to run long or distributed tasks

5 reasons why jupyter notebooks suck

Alexander Mueller [Follow](#)
Mar 24, 2018 · 3 min read ★

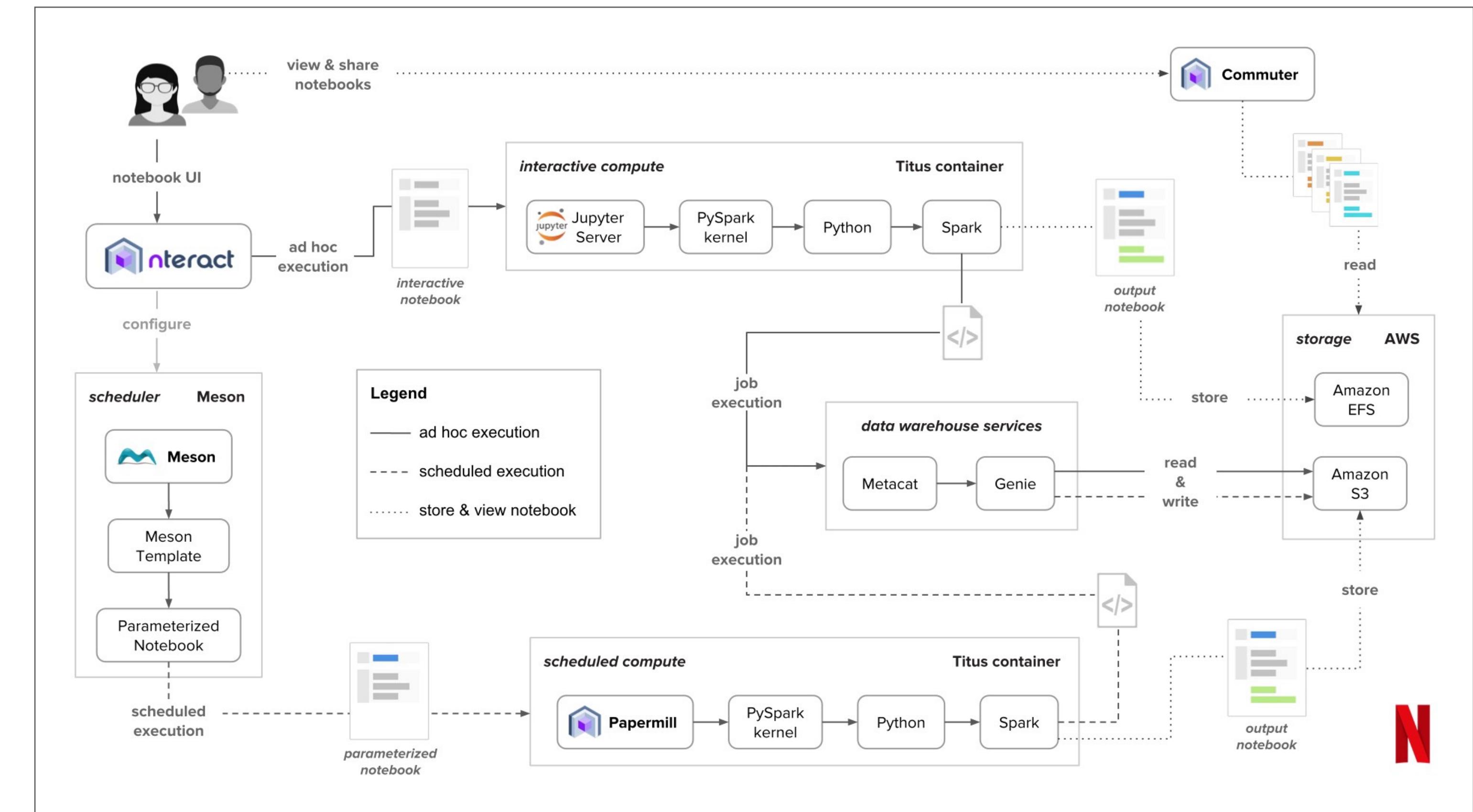


How it feels like managing jupyter notebooks (Complexity ©
<https://www.flickr.com/photos/bitterjug/7670055210>)

<https://towardsdatascience.com/5-reasons-why-jupyter-notebooks-suck-4dc201e27086>

Jupyter Notebooks

- Counter-points:
 - Netflix bases all ML workflows on them



<https://medium.com/netflix-techblog/notebook-innovation-591ee3221233>

NBDev

- Counter-points:
 - Jeremy Howard from fast.ai uses them for everything, with nbdev

Card

API details.

```
#hide
from nbdev.showdoc import *

#export
from __future__ import print_function, division
import random

class Card:
    """Represents a standard playing card.

    Attributes:
        suit: integer 0-3
        rank: integer 1-13
    """

    suit_names = ["Clubs", "Diamonds", "Hearts", "Spades"]
    rank_names = [None, "Ace", "2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen", "King"]

    def __init__(self, suit=0, rank=2):
        self.suit, self.rank = suit, rank

    def __str__(self):
        """Returns a human-readable string representation."""
        return '%s of %s' % (Card.rank_names[self.rank], Card.suit_names[self.suit])

    def __eq__(self, other) -> bool:
        """Checks whether self and other have the same rank and suit."""
        return self.suit == other.suit and self.rank == other.rank

    def __lt__(self, other) -> bool:
        """Compares this card to other, first by suit, then rank."""
        t1 = self.suit, self.rank
        t2 = other.suit, other.rank
        return t1 < t2

    def __repr__(self): return self.__str__()

    def foo(): pass
```

Card is a class that represents a single card in a deck of cards. For example:

```
Card(suit=2, rank=11)
```

Jack of Hearts

```
c = Card(suit=1, rank=3)
assert str(c) == '3 of Diamonds'

c2 = Card(suit=2, rank=11)
assert str(c2) == 'Jack of Hearts'
```

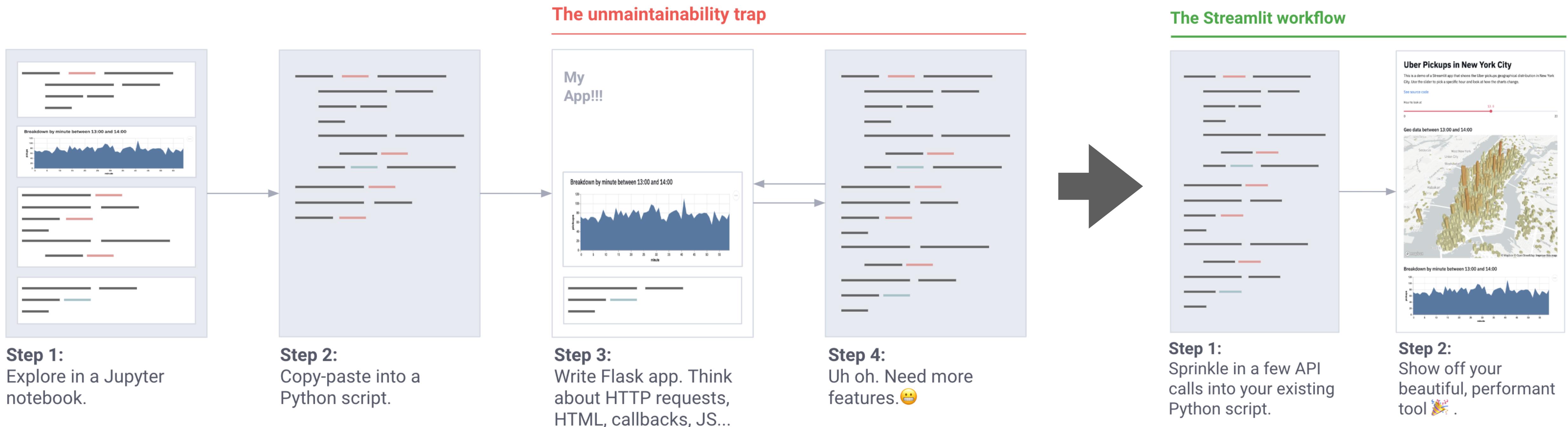
You can do comparisons of cards, too!

```
assert c2 > c
```

<https://github.com/fastai/nbdev>

Streamlit

- New, but great at fulfilling a common ML need: interactive applets
- Decorate normal Python code
- Smart data caching, quick re-rendering
- In the works: sharing as easy as pushing a web app to Heroku



Setting up environment

Conda + Pip-Tools Sample Project

Quick demo of setting up a deep learning Python environment.

Our goals:

- Easily specify the exact Python, CUDA, CUDNN environment
- Humans should specify minimal constraints (`torch >= 1.7` and `numpy`), computer should figure out exact, mutually compatible versions (`torch==1.7.1; numpy==1.19.5`)
- Separate production (`torch`) from development (`black`) dependencies

We achieve this by:

- We specify our Python and CUDA versions in `environment.yml`
- We use the `conda` package manager to create our environment from this file
- We specify our requirements in `requirements/prod.in` and `requirements/dev.in`
- We use `pip-tools` to lock in mutually compatible versions of all requirements
- We add a `Makefile` so we can simply run `make` to update everything

(How we do it in lab)

<https://github.com/full-stack-deep-learning/conda-pip-tools>

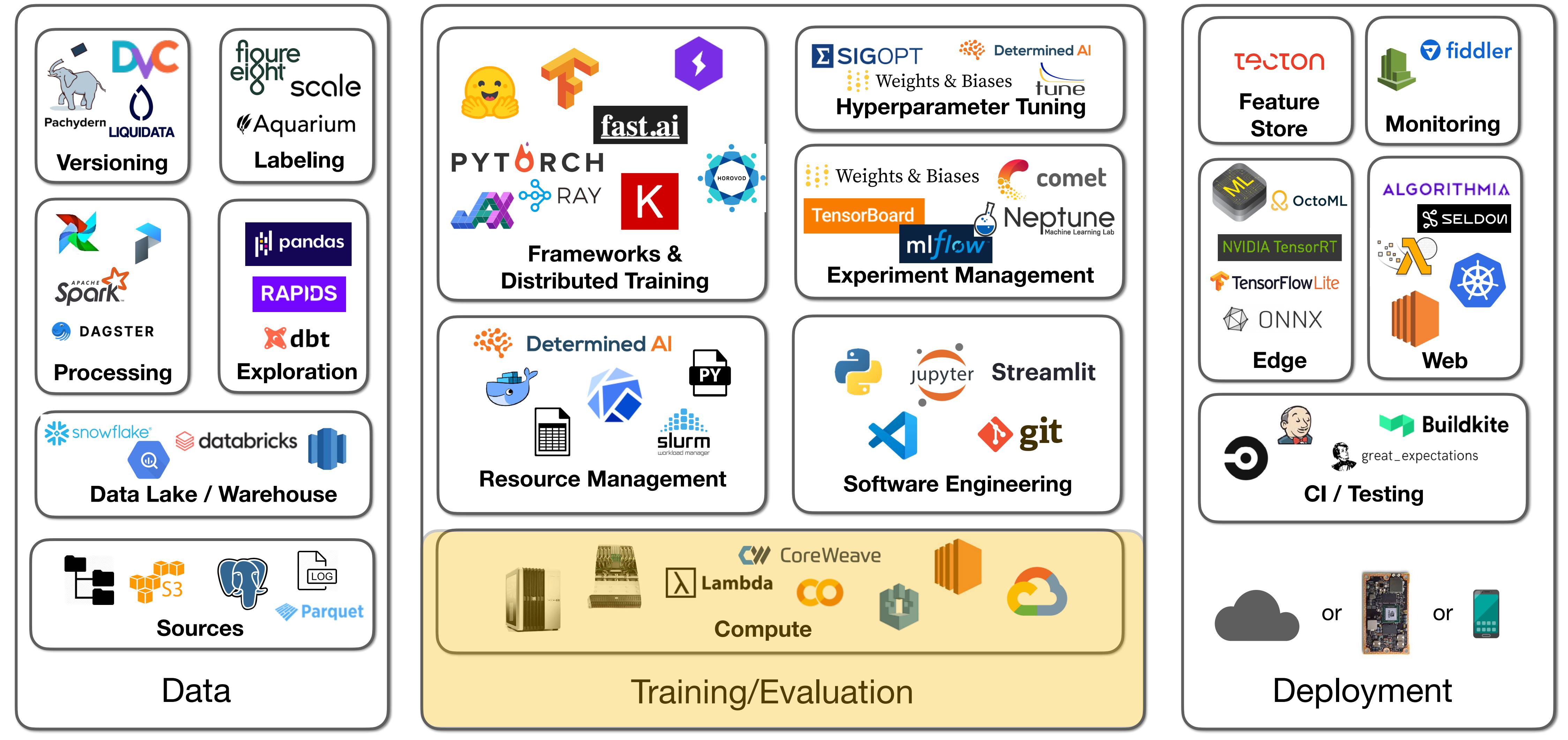
Questions?



Amazon SageMaker



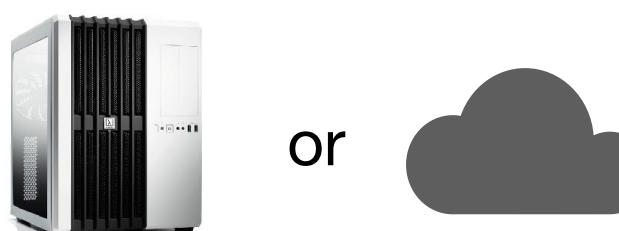
“All-in-one”



Compute needs

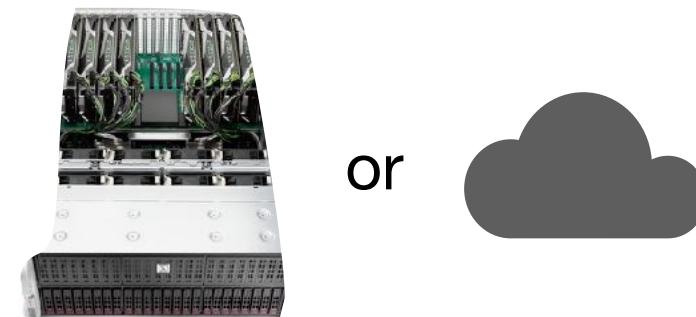
Development

- **Function**
 - Writing code
 - Debugging models
 - Looking at results
- **Desiderata**
 - Quickly compile models and run training
 - Nice-to-have: use GUI
- **Solutions**
 - Desktop with 1-4 GPUs
 - Cloud instance with 1-4 GPUs

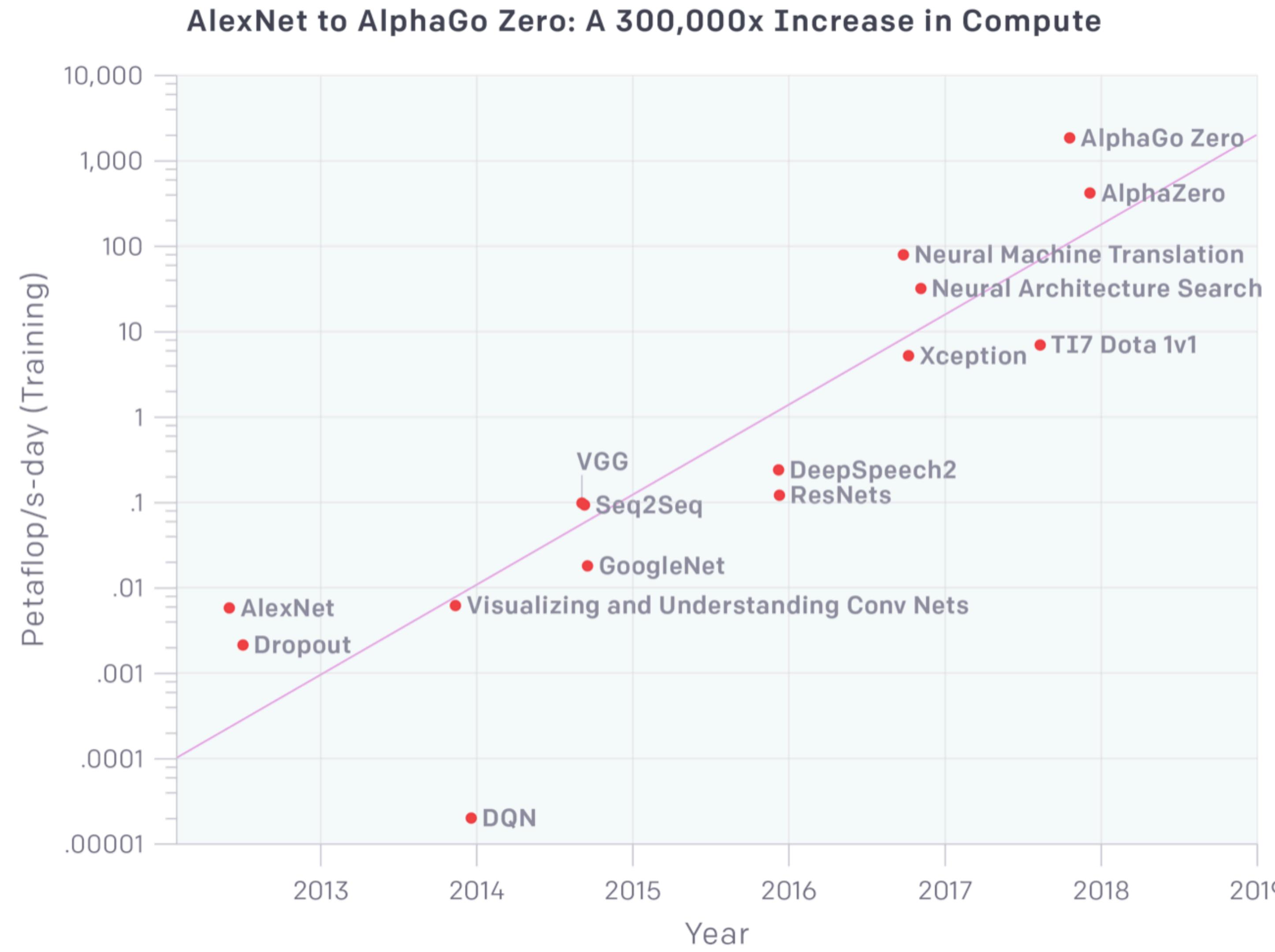


Training/Evaluation

- **Function**
 - Model architecture / hyperparam search
 - Training large models
- **Desiderata**
 - Easy to launch experiments and review results
- **Solutions**
 - Desktop with 4 GPUs
 - Private cluster of GPU machines
 - Cloud cluster of GPU instances

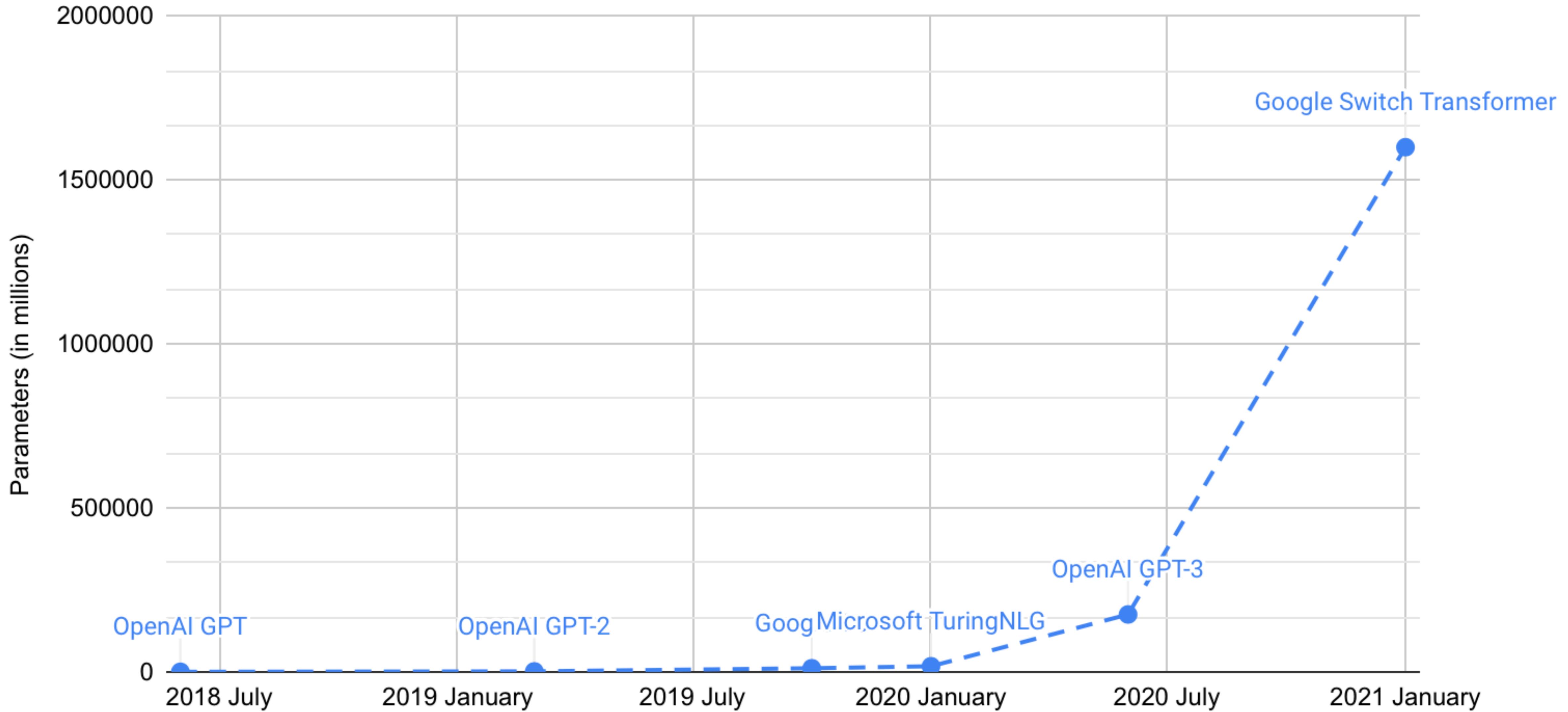


Why compute matters



<https://openai.com/blog/ai-and-compute/>

Transformer Models



So,  or ?

- GPU Basics
- Cloud Options
- On-prem Options
- Analysis and Recommendations

GPU Basics

- NVIDIA has been the only game in town
- Google TPUs are the fastest (on GCP only)

GPU Comparison Table

Card	Release	Arch	Use-case	RAM (Gb)	32bit TFlops	Tensor TFlops	16bit	Cost	Cloud
K80	2014H2	Kepler	Server	24	5	N/A	No	-	AWS, GCP, MS
P100	2016H1	Pascal	Server	16	10	N/A	Yes	-	GCP, MS
1080 Ti	2017H1	Pascal	Consumer	11	13	N/A	No	\$	
V100	2017H1	Volta	Server	16	14	120	Yes	\$\$\$\$	AWS, GCP, MS
2080 Ti	2018H2	Turing	Consumer	11	13	107	Yes	\$\$	
Titan RTX	2018H2	Turing	Enthusiast	24	16	130	Yes	\$\$	
3090	2021H1	Ampere	Enthusiast	24	?	285	Yes	\$\$	
A100	2020H1	Ampere	Server	40	19.5	312	Yes	\$\$\$\$	AWS, GCP

<https://www.microway.com/knowledge-center-articles/comparison-of-nvidia-geforce-gpus-and-nvidia-tesla-gpus/>

GPU Comparison Table

Card	Release	Arch	Use-case	RAM (Gb)	32bit TFlops	Tensor TFlops	16bit	Cost	Cloud
K80	2014H2	Kepler	Server	24	5	N/A	No	-	AWS, GCP, MS
P100	2016H1	Pascal	Server	16	10	N/A	Yes	-	GCP, MS
1080 Ti	2017H1	Pascal	Consumer	11	13	N/A	No	\$	
V100	2017H1	Volta	Server	16	14	120	Yes	\$\$\$\$	AWS, GCP, MS
2080 Ti	2018H2	Turing	Consumer	11	13	107	Yes	\$\$	
Titan RTX	2018H2	Turing	Enthusiast	24	16	130	Yes	\$\$	
3090	2021H1	Ampere	Enthusiast	24	?	285	Yes	\$\$	
A100	2020H1	Ampere	Server	40	19.5	312	Yes	\$\$\$\$	AWS, GCP

- New NVIDIA architecture every year
 - Kepler → Pascal → Volta → Turing → Ampere
 - Server version first, then “enthusiast”, then consumer.
 - For business, only supposed to use server cards.

GPU Comparison Table

Card	Release	Arch	Use-case	RAM (Gb)	32bit TFlops	Tensor TFlops	16bit	Cost	Cloud
K80	2014H2	Kepler	Server	24	5	N/A	No	-	AWS, GCP, MS
P100	2016H1	Pascal	Server	16	10	N/A	Yes	-	GCP, MS
1080 Ti	2017H1	Pascal	Consumer	11	13	N/A	No	\$	
V100	2017H1	Volta	Server	16	14	120	Yes	\$\$\$\$	AWS, GCP, MS
2080 Ti	2018H2	Turing	Consumer	11	13	107	Yes	\$\$	
Titan RTX	2018H2	Turing	Enthusiast	24	16	130	Yes	\$\$	
3090	2021H1	Ampere	Enthusiast	24	?	285	Yes	\$\$	
A100	2020H1	Ampere	Server	40	19.5	312	Yes	\$\$\$\$	AWS, GCP

- **RAM:** should fit meaningful batches of your model

<http://timdettmers.com/2018/11/05/which-gpu-for-deep-learning/>

GPU Comparison Table

Card	Release	Arch	Use-case	RAM (Gb)	32bit TFlops	Tensor TFlops	16bit	Cost	Cloud
K80	2014H2	Kepler	Server	24	5	N/A	No	-	AWS, GCP, MS
P100	2016H1	Pascal	Server	16	10	N/A	Yes	-	GCP, MS
1080 Ti	2017H1	Pascal	Consumer	11	13	N/A	No	\$	
V100	2017H1	Volta	Server	16	14	120	Yes	\$\$\$\$	AWS, GCP, MS
2080 Ti	2018H2	Turing	Consumer	11	13	107	Yes	\$\$	
Titan RTX	2018H2	Turing	Enthusiast	24	16	130	Yes	\$\$	
3090	2021H1	Ampere	Enthusiast	24	?	285	Yes	\$\$	
A100	2020H1	Ampere	Server	40	19.5	312	Yes	\$\$\$\$	AWS, GCP

- **RAM:** should fit meaningful batches of your model
- **32bit vs Tensor TFlops**
 - Tensor Cores are specifically for deep learning operations (mixed precision)
 - Good for convolutional/transformer models

<http://timdettmers.com/2018/11/05/which-gpu-for-deep-learning/>

GPU Comparison Table

Card	Release	Arch	Use-case	RAM (Gb)	32bit TFlops	Tensor TFlops	16bit	Cost	Cloud
K80	2014H2	Kepler	Server	24	5	N/A	No	-	AWS, GCP, MS
P100	2016H1	Pascal	Server	16	10	N/A	Yes	-	GCP, MS
1080 Ti	2017H1	Pascal	Consumer	11	13	N/A	No	\$	
V100	2017H1	Volta	Server	16	14	120	Yes	\$\$\$\$	AWS, GCP, MS
2080 Ti	2018H2	Turing	Consumer	11	13	107	Yes	\$\$	
Titan RTX	2018H2	Turing	Enthusiast	24	16	130	Yes	\$\$	
3090	2021H1	Ampere	Enthusiast	24	?	285	Yes	\$\$	
A100	2020H1	Ampere	Server	40	19.5	312	Yes	\$\$\$\$	AWS, GCP

- **RAM:** should fit meaningful batches of your model
- **32bit vs Tensor TFlops**
 - Tensor Cores are specifically for deep learning operations (mixed precision)
 - Good for convolutional/transformer models
 - Great speedups and bigger batches from 16bit mixed-precision

<http://timdettmers.com/2018/11/05/which-gpu-for-deep-learning/>

Kepler/Maxwell

Card	Release	Arch	Use-case	RAM (Gb)	32bit TFlops	Tensor TFlops	16bit	Cost	Cloud
K80	2014H2	Kepler	Server	24	5	N/A	No	-	AWS, GCP, MS
P100	2016H1	Pascal	Server	16	10	N/A	Yes	-	GCP, MS
1080 Ti	2017H1	Pascal	Consumer	11	13	N/A	No	\$	
V100	2017H1	Volta	Server	16	14	120	Yes	\$\$\$\$	AWS, GCP, MS
2080 Ti	2018H2	Turing	Consumer	11	13	107	Yes	\$\$	
Titan RTX	2018H2	Turing	Enthusiast	24	16	130	Yes	\$\$	
3090	2021H1	Ampere	Enthusiast	24	?	285	Yes	\$\$	
A100	2020H1	Ampere	Server	40	19.5	312	Yes	\$\$\$\$	AWS, GCP

- 2-4x slower than Pascal/Volta
- Hardware: don't buy, too old
- Cloud: K80's are cheap (providers are stuck with what they bought)

Pascal

Card	Release	Arch	Use-case	RAM (Gb)	32bit TFlops	Tensor TFlops	16bit	Cost	Cloud
K80	2014H2	Kepler	Server	24	5	N/A	No	-	AWS, GCP, MS
P100	2016H1	Pascal	Server	16	10	N/A	Yes	-	GCP, MS
1080 Ti	2017H1	Pascal	Consumer	11	13	N/A	No	\$	
V100	2017H1	Volta	Server	16	14	120	Yes	\$\$\$\$	AWS, GCP, MS
2080 Ti	2018H2	Turing	Consumer	11	13	107	Yes	\$\$	
Titan RTX	2018H2	Turing	Enthusiast	24	16	130	Yes	\$\$	
3090	2021H1	Ampere	Enthusiast	24	?	285	Yes	\$\$	
A100	2020H1	Ampere	Server	40	19.5	312	Yes	\$\$\$\$	AWS, GCP

- Hardware: 1080 Ti still good if buying used, especially for recurrent
- Cloud: P100 is a mid-range option

Volta/Turing

Card	Release	Arch	Use-case	RAM (Gb)	32bit TFlops	Tensor TFlops	16bit	Cost	Cloud
K80	2014H2	Kepler	Server	24	5	N/A	No	-	AWS, GCP, MS
P100	2016H1	Pascal	Server	16	10	N/A	Yes	-	GCP, MS
1080 Ti	2017H1	Pascal	Consumer	11	13	N/A	No	\$	
V100	2017H1	Volta	Server	16	14	120	Yes	\$\$\$\$	AWS, GCP, MS
2080 Ti	2018H2	Turing	Consumer	11	13	107	Yes	\$\$	
Titan RTX	2018H2	Turing	Enthusiast	24	16	130	Yes	\$\$	
3090	2021H1	Ampere	Enthusiast	24	?	285	Yes	\$\$	
A100	2020H1	Ampere	Server	40	19.5	312	Yes	\$\$\$\$	AWS, GCP

- Preferred choice right now due to 16bit mixed precision support
- Hardware:
 - 2080 Ti is ~1.3x as fast as 1080 Ti in 32bit, but ~2x faster in 16bit
 - Titan RTX is 10-20% faster yet. Titan V is just as good (but less RAM), if find used.
 - Cloud: V100 is the ultimate for speed

<http://timdettmers.com/2018/11/05/which-gpu-for-deep-learning/> <https://lambdalabs.com/blog/titan-rtx-tensorflow-benchmarks/>

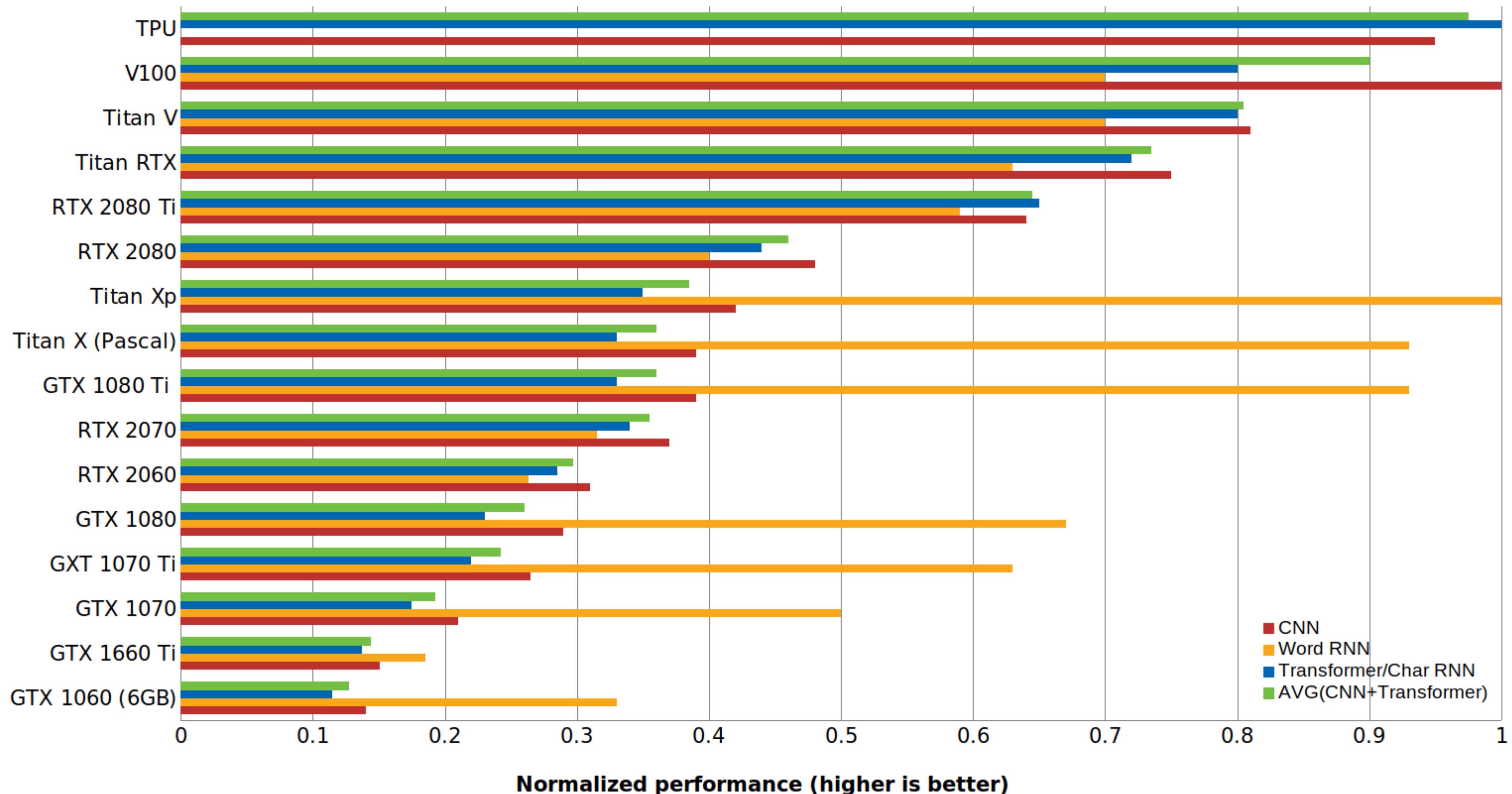
Ampere

Card	Release	Arch	Use-case	RAM (Gb)	32bit TFlops	Tensor TFlops	16bit	Cost	Cloud
K80	2014H2	Kepler	Server	24	5	N/A	No	-	AWS, GCP, MS
P100	2016H1	Pascal	Server	16	10	N/A	Yes	-	GCP, MS
1080 Ti	2017H1	Pascal	Consumer	11	13	N/A	No	\$	
V100	2017H1	Volta	Server	16	14	120	Yes	\$\$\$\$	AWS, GCP, MS
2080 Ti	2018H2	Turing	Consumer	11	13	107	Yes	\$\$	
Titan RTX	2018H2	Turing	Enthusiast	24	16	130	Yes	\$\$	
3090	2021H1	Ampere	Enthusiast	24	?	285	Yes	\$\$	
A100	2020H1	Ampere	Server	40	19.5	312	Yes	\$\$\$\$	AWS, GCP

- Latest hardware, with the most Tensor cores.
- At least 30% speedup over Turing

<http://timdettmers.com/2018/11/05/which-gpu-for-deep-learning/> <https://lambdalabs.com/blog/titan-rtx-tensorflow-benchmarks/>

Performance



Normalized performance (higher is better)

<https://timdettmers.com/2019/04/03/which-gpu-for-deep-learning/>

A100 vs V100

	NVIDIA TESLA A100 SXM4-40GB	NVIDIA TESLA V100 SXM3-32GB
FP32 CUDA Cores	6912	5120
Clock Speed	1410 MHz	1530MHz
Theoretical FP32 TFLOPS	19.5 TFLOPS	15.7 TFLOPS
VRAM	40 GB HBM2e	32 GB HBM2
Memory Bandwidth	1,555 GBps	900 GBps
GPU Interconnect	12 NVLink Connections (600 GBps)	6 NVLink Connections (300 GBps)
Process Node	TSMC 7nm	TSMC 12nm FFN
TDP (W)	400 W	300 W
Power Efficiency	45.2 GFLOPS / W	48.6 GFLOPS / W

For training convnets with PyTorch, the Tesla A100 is...

- **2.2x** faster than the V100 using 32-bit precision.*
- **1.6x** faster than the V100 using mixed precision.

For training language models with PyTorch, the Tesla A100 is...

- **3.4x** faster than the V100 using 32-bit precision.
- **2.6x** faster than the V100 using mixed precision.

<https://lambdalabs.com/blog/nvidia-a100-vs-v100-benchmarks/>

<https://lambdalabs.com/deep-learning/servers/hyperplane-a100>

Great resource

Which GPU(s) to Get for Deep Learning: My Experience and Advice for Using GPUs in Deep Learning

2020-09-07 by [Tim Dettmers](#) — [1,618 Comments](#)

TL;DR advice

Best GPU overall: RTX 3080 and RTX 3090.

GPUs to avoid (as an individual): Any Tesla card; any Quadro card; any Founders Edition card; Titan RTX, Titan V, Titan XP.

Additional Considerations for Ampere / RTX 30 Series

Summary:

- Ampere allows for sparse network training, which accelerates training by a factor of up to 2x.
- Sparse network training is still rarely used but will make Ampere future-proof.
- Ampere has new low-precision data types, which makes using low-precision much easy, but not necessarily faster than for previous GPUs.
- The new fan design is excellent if you have space between GPUs, but it is unclear if multiple GPUs with no space in-between them will be efficiently cooled.
- 3-Slot design of the RTX 3090 makes 4x GPU builds problematic. Possible solutions are 2-slot variants or the use of PCIe extenders.
- 4x RTX 3090 will need more power than any standard power supply unit on the market can provide right now.

The Most Important GPU Specs for Deep Learning Processing Speed

This section can help you build a more intuitive understanding of how to think about deep learning performance. This understanding will help you to evaluate future GPUs by yourself.

Tensor Cores

Summary:

- Tensor Cores reduce the used cycles needed for calculating multiply and addition operations, 16-fold – in my example, for a 32×32 matrix, from 128 cycles to 8 cycles.
- Tensor Cores reduce the reliance on repetitive shared memory access, thus saving additional cycles for memory access.
- Tensor Cores are so fast that computation is no longer a bottleneck. The only bottleneck is getting data to the Tensor Cores.

<https://timdettmers.com/2020/09/07/which-gpu-for-deep-learning/#GPU Recommendations>

Cloud Providers

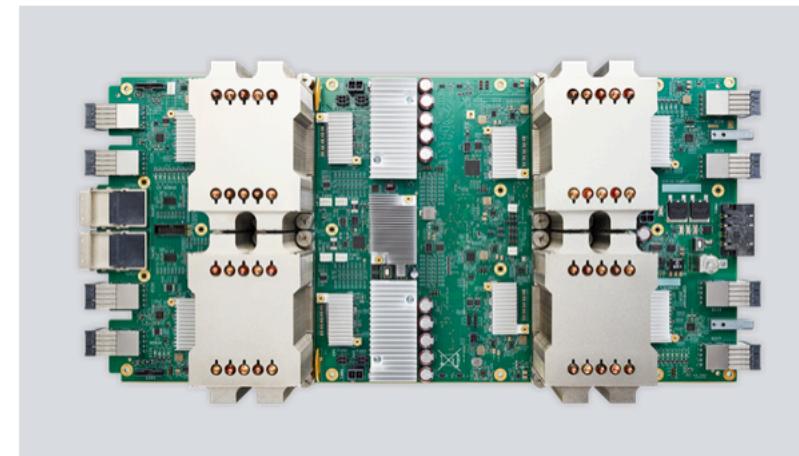
- Amazon Web Services, Google Cloud Platform, Microsoft Azure are the heavyweights.
- Heavyweights are largely similar in function and price.
 - AWS most expensive
 - GCP just about as expensive, and has TPUs
 - Azure reportedly has bad user experience
- Startups are Coreweave, Lambda Labs, and more

Amazon Web Services

Name	GPU	GPUs	GPU RAM	vCPU	RAM	On-demand
p2.16xlarge	K80	8	12	64	732	\$14.40
p3.16xlarge	V100	8	16	64	488	\$24.48
p3dn.24xlarge	V100	8	32	96	768	\$31.22
p4dn.16xlarge	A100	8	40	96	1152	\$32.78

- Three generations of GPUs
- Prices haven't moved in years

Google Cloud Platform

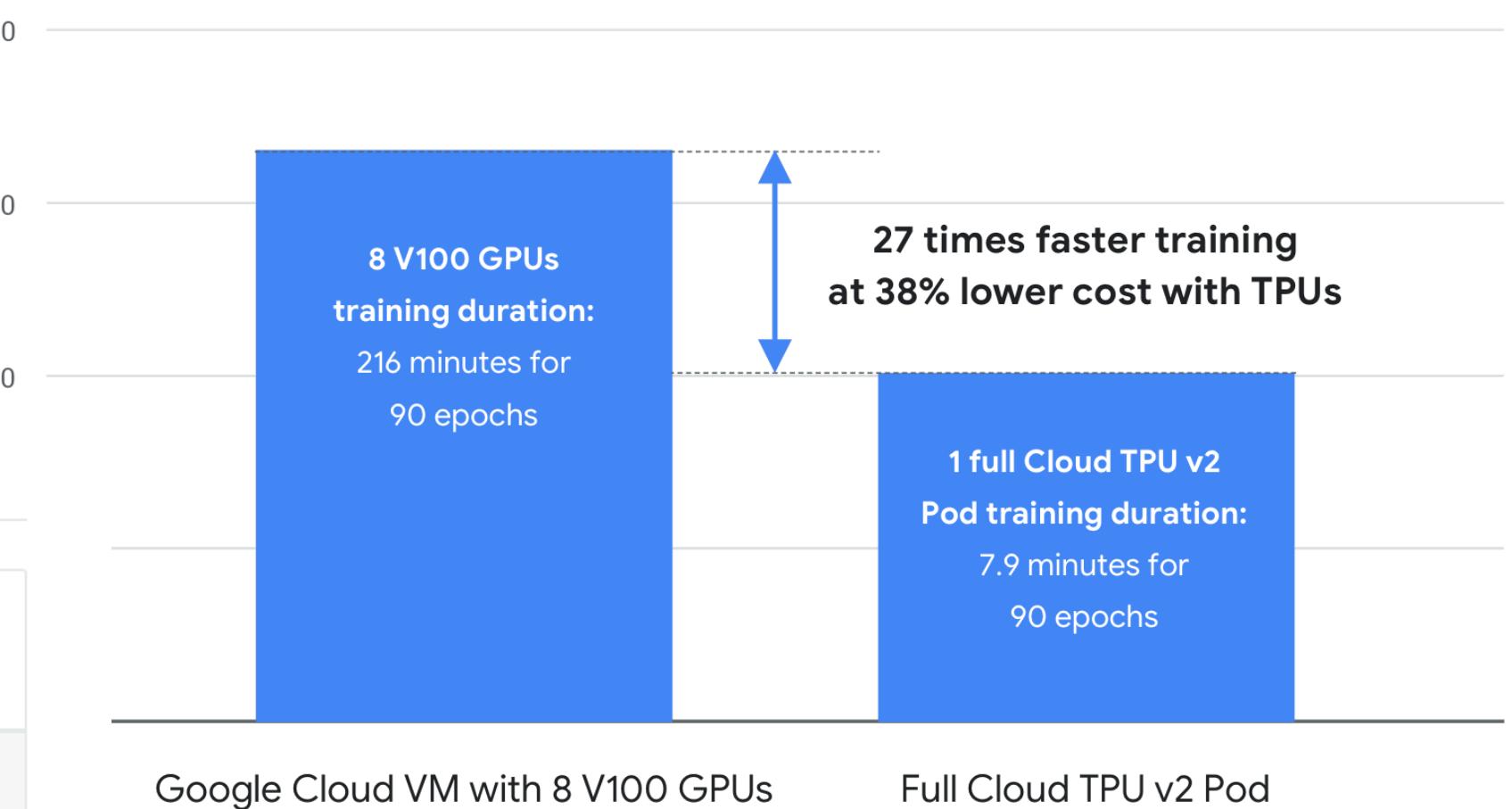


Cloud TPU v2
180 teraflops
64 GB High Bandwidth Memory (HBM)

US EUROPE ASIA PACIFIC

Version	On-demand	Preemptible
Cloud TPU v2	\$4.50 / TPU hour	\$1.35 / TPU hour
Cloud TPU v3	\$8.00 / TPU hour	\$2.40 / TPU hour

ResNet-50 Training Cost Comparison



- Same lineup: K80, V100 (also P100), and A100
- In general, a little cheaper than AWS.
- Also has “Tensor Processing Units” (TPUs), the fastest option today.

Microsoft Azure

- Same lineup: K80, P100s, V100s (no A100s yet)
- Similar pricing

<https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/>

Lambda Labs Cloud

4-GPU Instance

For researchers, students, and hobbyists looking for extra power at low cost.

INSTANCE SPECIFICATIONS

GPU DETAILS

4x NVIDIA® Pascal Based GPUs (11 GB)

VCPUS

8

SYSTEM RAM

32 GB

TEMPORARY STORAGE

1.4 TB SATA SSD

NETWORK INTERFACE

1 Gbps (peak)

Dedicated 4-GPU instance pricing

\$1.50 / hr

Launch an instance

8-GPU Instance

NEW

For ML engineers and researchers looking for maximum training speed.

INSTANCE SPECIFICATIONS

GPU DETAILS

8x NVIDIA® V100 Tensor Core (16 GB) + NVLink™

VCPUS

92

SYSTEM RAM

448 GB

TEMPORARY STORAGE

6 TB NVMe

NETWORK INTERFACE

10 Gbps (peak)

Dedicated 8-GPU instance pricing

\$12.00 / hr

Launch an instance

<https://lambdalabs.com/service/gpu-cloud>

Coreweave

The industry's best economics.
Period.

Priced by the hour, billed by the minute

Qty	GPU Type	VRAM	GPUs		Virtual Workstations		CPU Compute		\$/Hour	Optimized for
			vCPU	RAM	Storage	Network				
- 4 +	NVIDIA A100	40GB HBM2e	120	512 GB	1800 GB	40 Gbps	\$12.00	   		
- 4 +	NVIDIA A6000	48GB GDDR6	32	512 GB	1800 GB	10 Gbps	\$8.00	   		
- 8 +	NVIDIA V100 for NVLINK	16GB HBM2	32	256 GB	3600 GB	10 Gbps	\$8.00	   		
- 8 +	NVIDIA V100 for PCIe	16GB HBM2	24	160 GB	512 GB	1 Gbps	\$4.80	   		
- 4 +	NVIDIA P100 for NVLINK	16GB HBM2	16	128 GB	1800 GB	10 Gbps	\$3.00	   		
- 4 +	NVIDIA Quadro RTX 6000	24GB GDDR6	32	240 GB	1800 GB	10 Gbps	\$5.40	   		
- 4 +	NVIDIA Quadro RTX 5000	16GB GDDR6	32	240 GB	1800 GB	10 Gbps	\$3.80	   		
- 4 +	NVIDIA Quadro RTX 4000	8GB GDDR6	12	64 GB	500 GB	1 Gbps	\$1.40	   		

<https://www.coreweave.com/pricing>

On-prem Options

- Build your own
 - Up to 4 Turing or 2 Ampere GPUs is easy
- Buy pre-built
 - Lambda Labs, NVIDIA, and builders like Supermicro, Cirrascale, etc.

Building your own

- Quiet PC with 128GB RAM and 2x RTX 3090's: \$8000
 - (If you can find them)
 - One day to build and set up
- Going beyond 4 2000-series or 2 3000-series is painful
- All you need to know: <http://timdettmers.com/2018/12/16/deep-learning-hardware-guide/>



Pre-built: Lambda Labs



AMD Threadripper 3960X
24 cores, 3.80 GHz

2x RTX 3090
24 GB VRAM per GPU

256 GB
System memory

3.84 TB NVMe
OS Drive

3.84 TB SSD (SATA)
Extra Storage

\$11,446.00
Apply for a discount

~20% more expensive
than building yourself

Customize your computer

Operating system

Ubuntu 20.04 + Lambda Stack

Ubuntu 18.04 + Lambda Stack

+ \$0

Windows 10

+ \$200

No Operating System

+ \$0

Processor

AMD Threadripper 3960X

24 cores, 3.80 GHz, 128 MB cache

Download quote

Add to cart

Pre-built: Lambda Labs



Premium



8x Tesla V100 Server

8-way NVLink

2x Xeon Gold 6248 (20 cores)

8x Tesla V100 GPUs (32 GB VRAM)

512 GB RAM

Customizable storage

100 Gbps InfiniBand

\$94,870

🎓 Academic Discounts

🕒 COVID-19 Research Discounts

Customize

Lambda Hyperplane 4-A100

4 NVIDIA A100 Tensor Core GPUs with NVLink™ & Mellanox InfiniBand

SYSTEM SPECIFICATIONS

GPU DETAILS

4x NVIDIA Tesla A100 SXM4-40GB + NVLink

PROCESSOR

2x AMD EPYC™ Processors (Up to 64 cores)

SYSTEM RAM

512 GB

STORAGE

Up to 60TB NVMe

NETWORK INTERFACE

Up to 4x Mellanox InfiniBand HDR 200Gbps Cards

Hyperplane 4-A100 pricing starting at

\$65,000

Get a quote

Cost Analysis

- Let's first compare on-prem and equivalent cloud machines
- Then let's also consider spot instances for experiment scaling

Quad PC vs Quad Cloud

GPU	Arch	RAM	Build Price	Cloud Price	Hours = Build	24/7 Weeks = Build	16/5 Weeks = Build
4x RTX 2080 Ti	Volta	12	\$10000.00	0			
4x V100	Volta	16		\$12.00	833	5	10

Full-time workload
Work week load

Verdict: not worth it. PC pays for itself in 5-10 weeks.

Quad PC vs Quad Cloud

↑ Posted by u/cgnorthcutt 8 months ago ▾

481

[P] I built Lambda's \$12,500 deep learning rig for \$6200

↓

Project

See: <http://l7.curtisnorthcutt.com/build-pro-deep-learning-workstation>

Hi Reddit! I built a 3-GPU deep learning workstation similar to Lambda's 4-GPU (RTX 2080 TI) rig for half the price. In the hopes of helping other researchers, I'm sharing a time-lapse of the build, the parts list, the receipt, and benchmarking versus Google Compute Engine (GCE) on ImageNet. You save \$1200 (the cost of an EVGA RTX 2080 ti GPU) per ImageNet training to use your own build instead of GCE. The training time is reduced by over half. In the post, I include 3 GPUs, but the build (increase PSU wattage) will support a 4th RTX 2080 TI GPU for \$1200 more (\$7400 total). Happy building!

<https://l7.curtisnorthcutt.com/build-pro-deep-learning-workstation>

Quad PC vs. Spot Instances

Length of trial in experiment (hours)	6
Number of trials in experiment	16
Total GPU hours for experiment	96
Cost of 4x RTX 2080 Ti machine	\$10,000.00
Time to run experiment on 4x machine	24 <i>hours</i>
Time to run experiment on V100 spot instances	6 <i>hours</i>
Cost of provisioning enough pre-emptible V100s	\$96.00
Number of experiments that equal cost of 4x	104

How to think about it: **cloud enables quicker experiments.**

Quad PC vs. Spot Instances

Length of trial in experiment (hours)	6
Number of trials in experiment	16
Total GPU hours for experiment	96
Cost of 4x RTX 2080 Ti machine	\$10,000.00
Time to run experiment on 4x machine	24 hours
Time to run experiment on V100 spot instances	6 hours
Cost of provisioning enough pre-emptible V100s	\$96.00
Number of experiments that equal cost of 4x	104

But at a pretty steep price.

In Practice

- Even though cloud is expensive, it's hard to make on-prem scale past a certain point
- Dev-ops (declarative infra, repeatable processes) definitely easier in the cloud
- Maintenance is also a big factor

Recommendation for hobbyist

- Development
 - Build a 4x Turing or 2x Ampere PC
- Training/Evaluation
 - Use the same PC, just always keep it running
 - To scale out, use Lambda or Coreweave cloud instances.

Recommendation for startup

- Development
 - Buy a 4x Turing or 2x Ampere PC per ML Scientist
- Training/Evaluation
 - To scale out, buy shared server machines or use cloud instances.

Recommendation for larger company

- Development
 - Buy 8x Turing or Ampere server rack per ML scientist
 - Or, go straight to the cloud with V100 instances
- Training/Evaluation
 - To scale out, use cloud instances with proper provisioning and handling of failures

Questions?



Amazon SageMaker

gradient^o
by Paperspace

FLOYD

DOMINO
DATA LAB

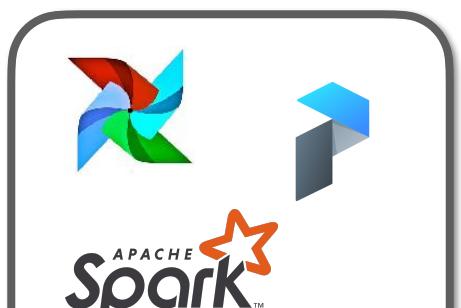
“All-in-one”



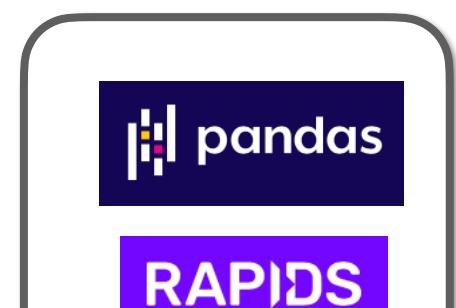
Versioning



Labeling



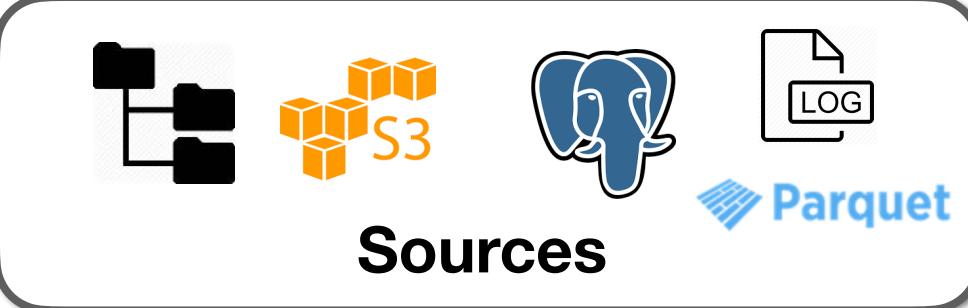
Processing



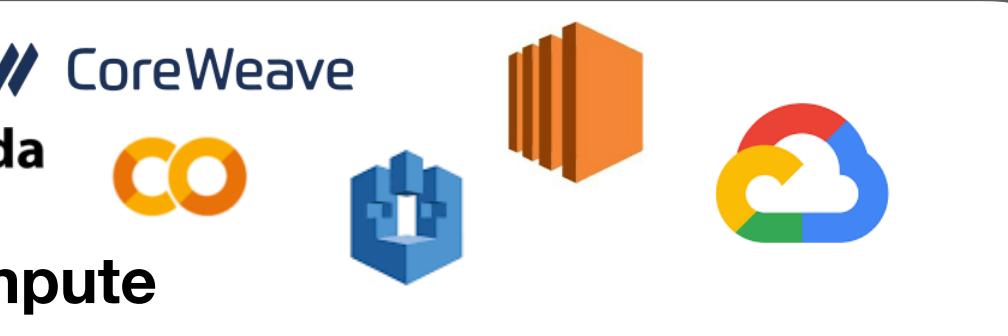
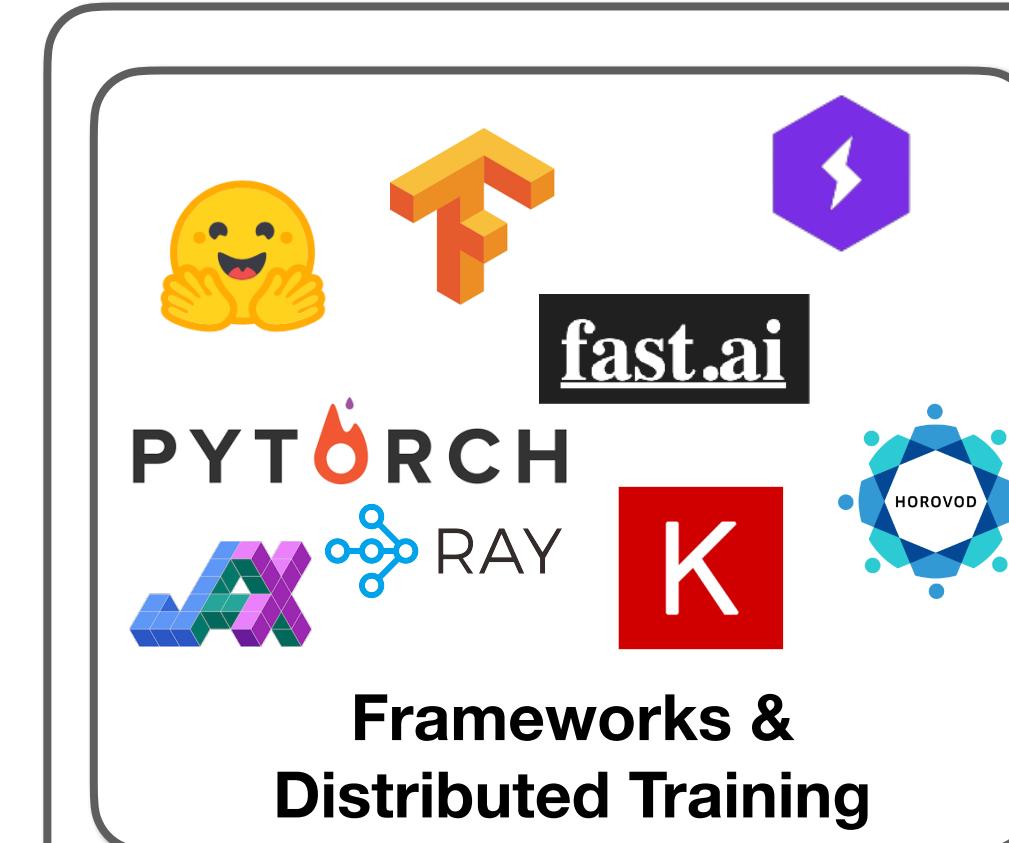
dbt Exploration



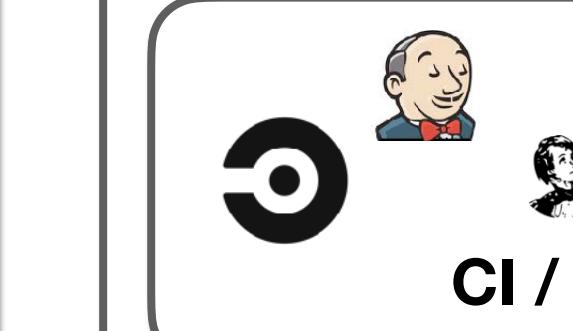
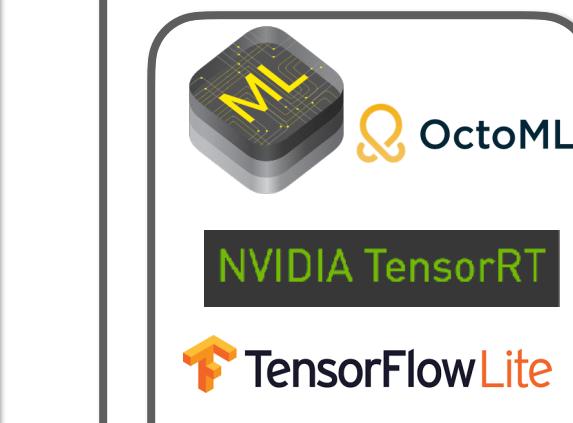
Data Lake / Warehouse



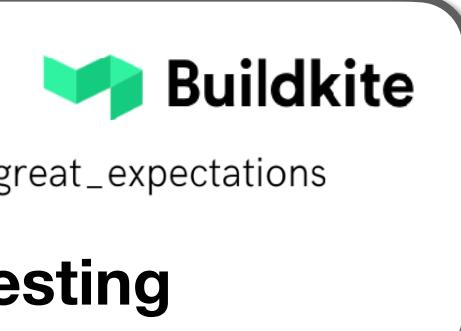
Data



Training/Evaluation



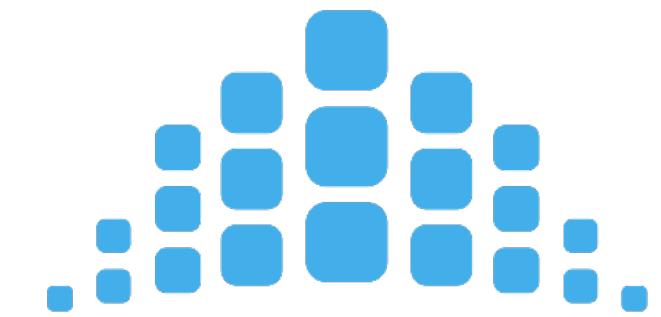
Edge



Deployment

Resource Management

- **Function**
 - Multiple people...
 - using multiple GPUs/machines...
 - running different environments
- **Goal**
 - Easy to launch a batch of experiments, with proper dependencies and resource allocations
- **Solutions**
 - Python scripts
 - SLURM **workload manager**
 - **Docker + Kubernetes**
 - Software specialized for ML use cases



Scripts or **slurm** workload manager

- Problem we're solving: allocate free resources to programs
- Can be scripted pretty easily
- Even better, use old-school cluster job scheduler
 - Job defines necessary resources, gets queued

```
12  def __init__(self, verbose: bool=False):
13      self.lock_manager = Redlock([{"host": "localhost", "port": 6379, "db": 0}, ])
14      self.verbose = verbose
15
16  def get_free_gpu(self):
17      """
18      If some GPUs are available, try reserving one by checking out an exclusive redis lock.
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
```

Job-file (schedule job with sbatch, check status with squeue -u <Username>):

```
#!/bin/bash
#SBATCH --gres=gpu:1
#SBATCH --mem=10000
#SBATCH -p gpu2    # K80 GPUs on Haswell node
#SBATCH --time=01:00:00

## with Theano (using configs from above)
module purge # purge if you already have modules loaded
module load modenv/eb
module load Keras

srun python mnist_cnn.py

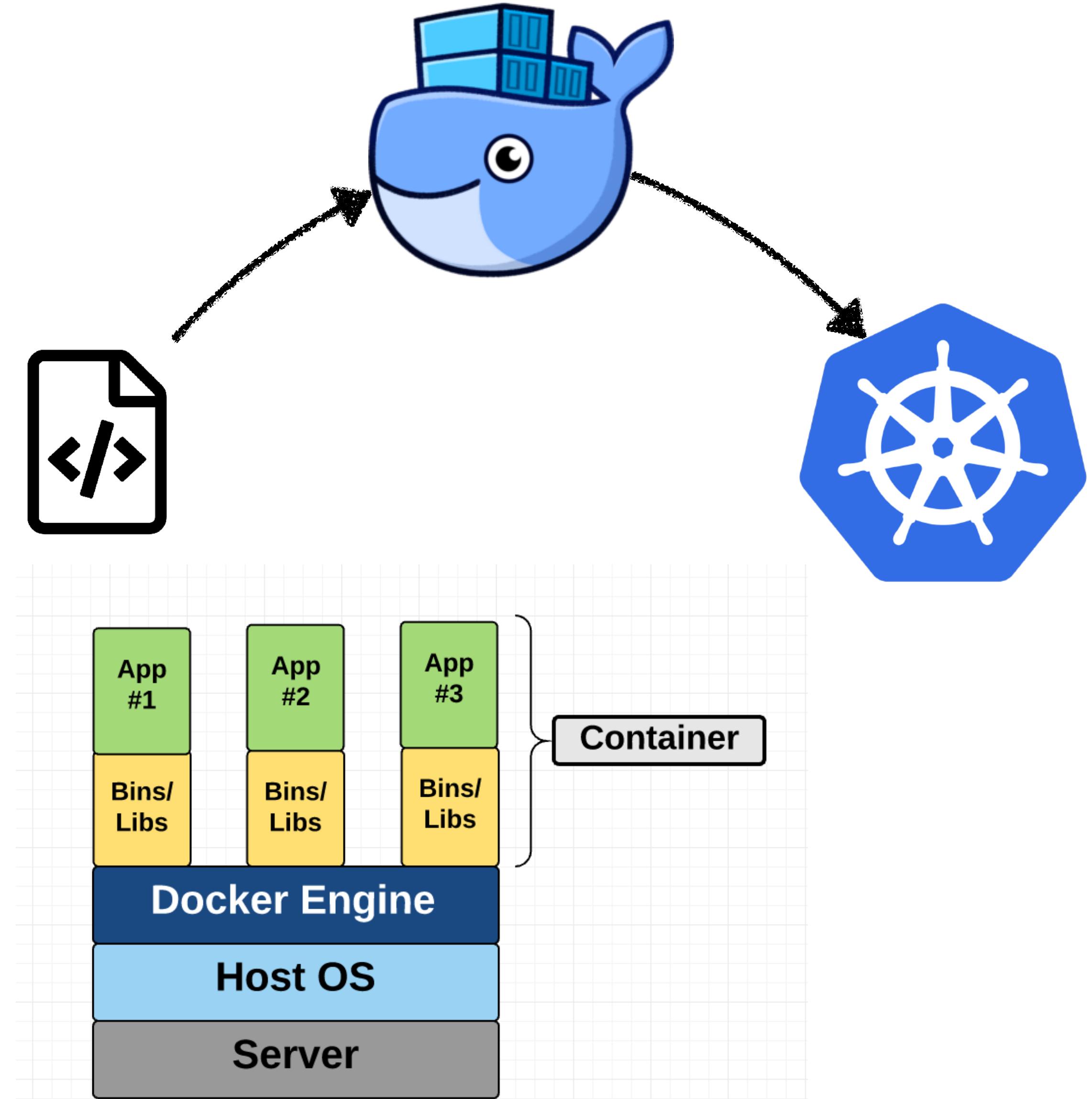
## with Tensorflow
module purge
module load modenv/eb
module load Keras
module load tensorflow
# if you see 'broken pipe error's (might happen in interactive session after
module load h5py/2.6.0-intel-2016.03-GCC-5.3-Python-3.5.2-HDF5-1.8.17-serial

export KERAS_BACKEND=tensorflow      # configure Keras to use tensorflow

srun python mnist_cnn.py
```

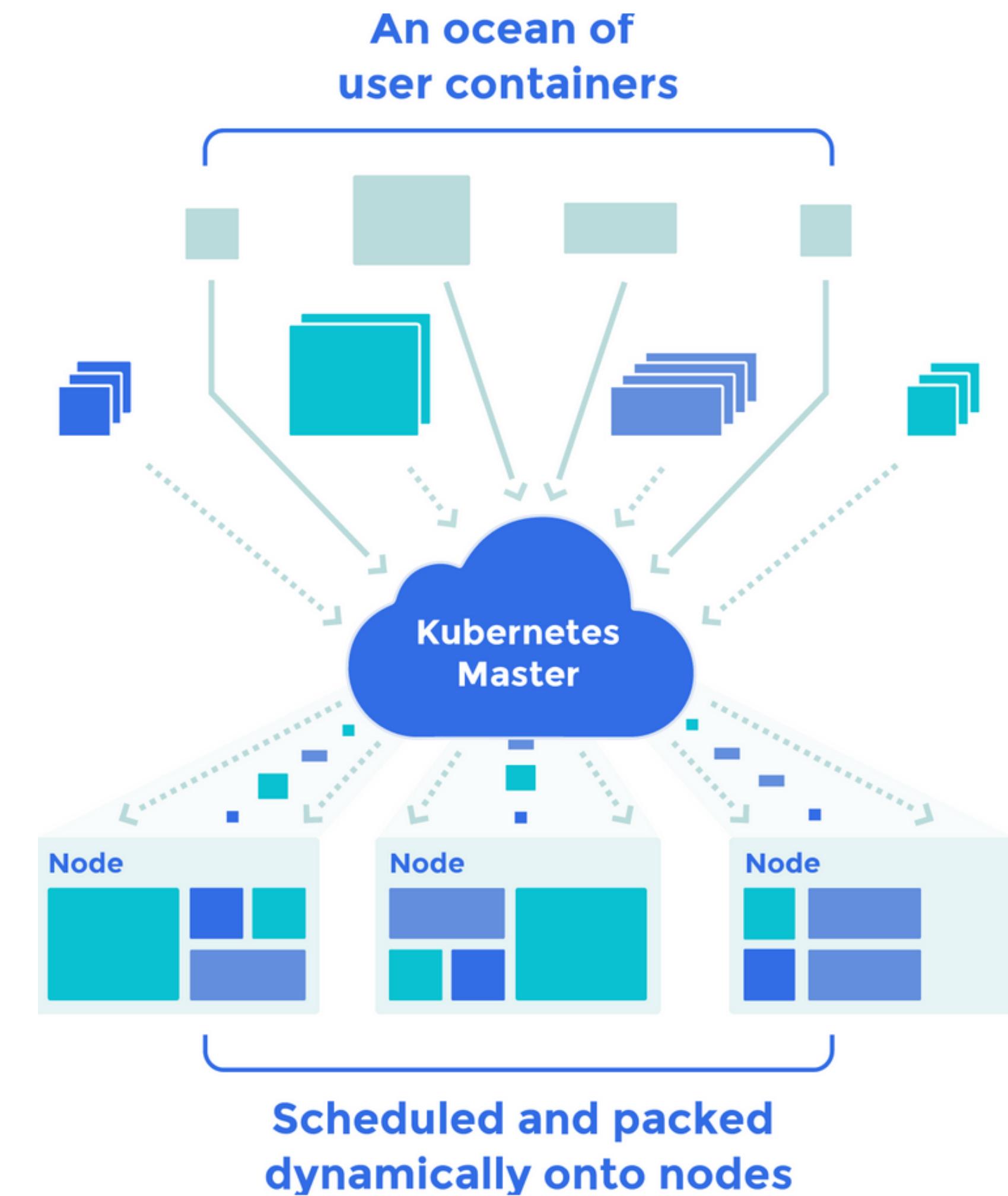
Docker + Kubernetes

- Docker is a way to **package up** an entire dependency stack in a lighter-than-a-VM package
- We will talk more about Docker in the Deployment lecture, and use it in lab



Docker + Kubernetes

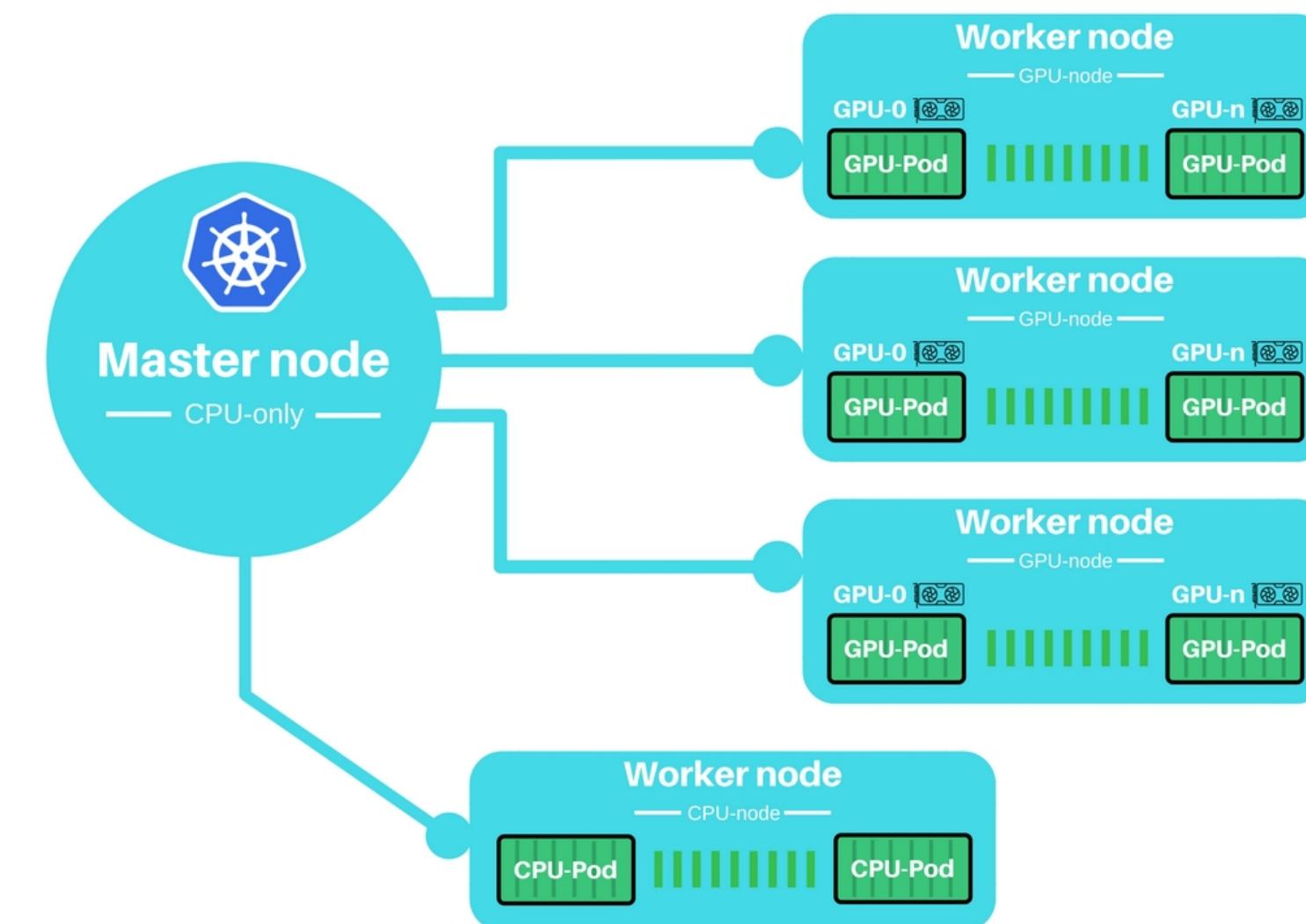
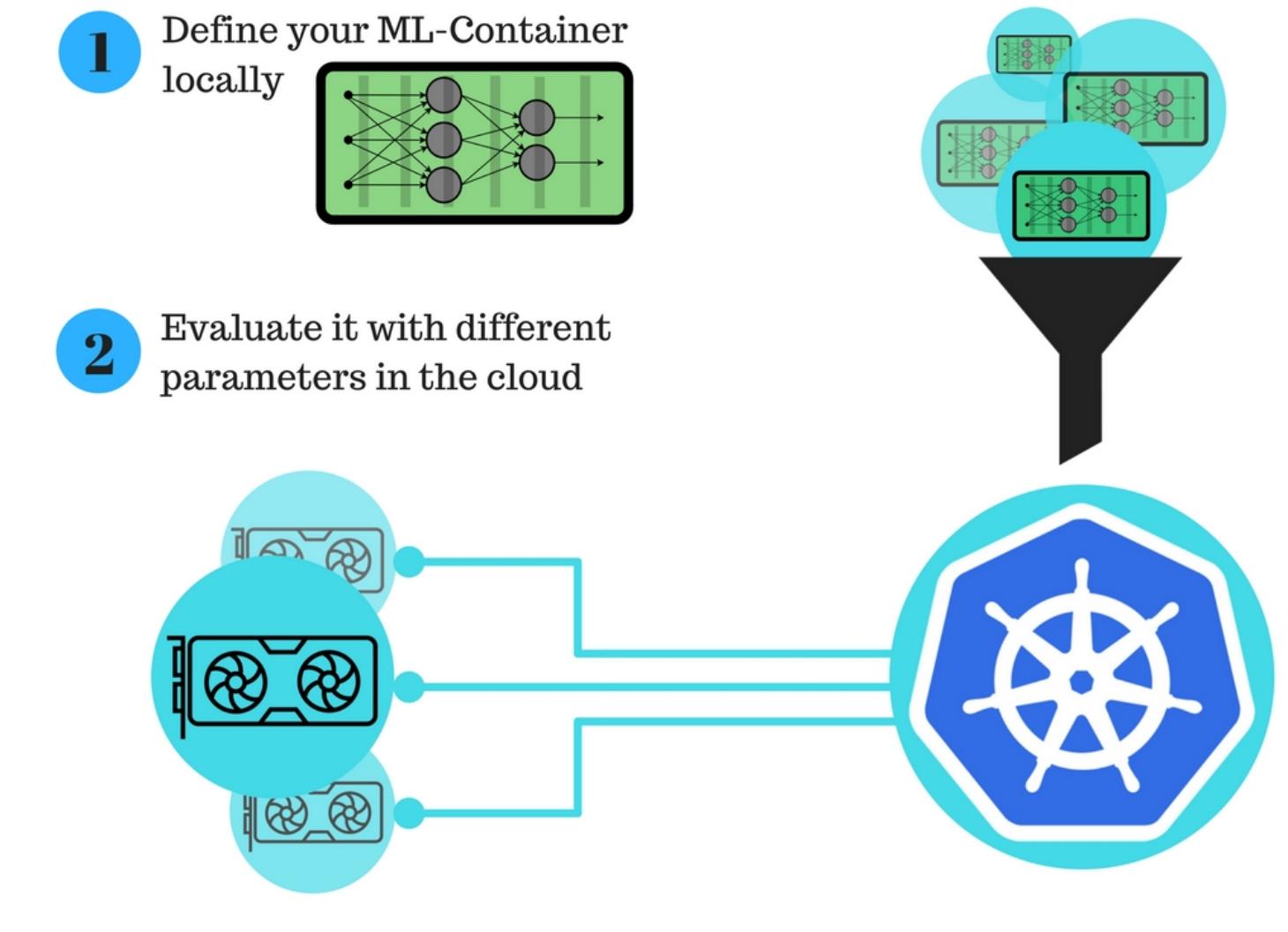
- Kubernetes is a way to run many docker containers on top of a cluster





Kubeflow

- Open source project from Google
- Spawn and manage Jupyter notebooks
- Manage multi-step ML workflows
- Plug-ins for hyperparameter tuning, model deployment



<https://github.com/Langhalsdino/Kubernetes-GPU-Guide>

This is an active area

- All-in-one solutions like SageMaker and Paperspace Gradient do this
- And some recently-announced startups have it as their goal...

Anyscale (makers of Ray, from Berkeley)

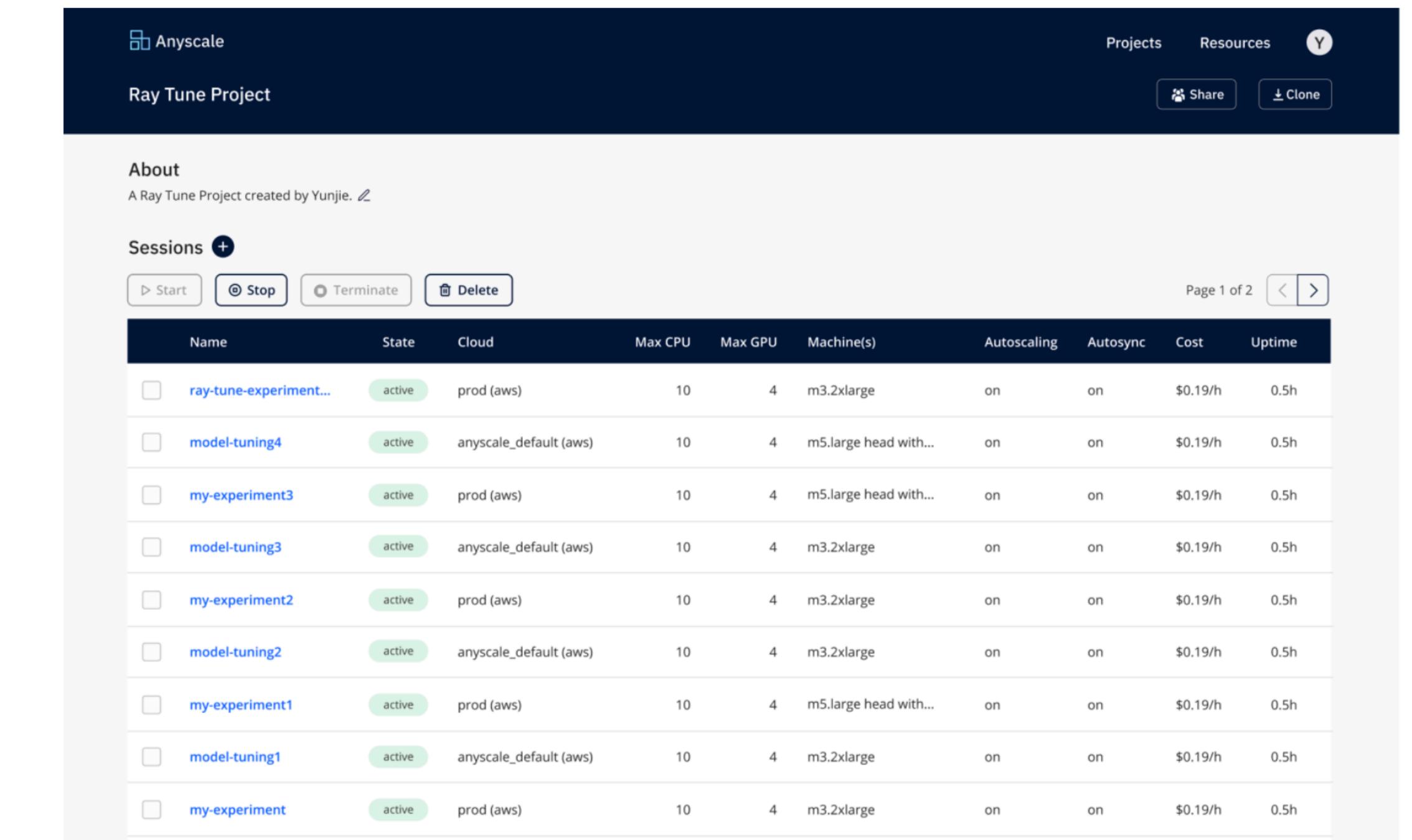
Serverless experience

without Serverless Limitations

Anyscale is cloud agnostic, supports stateless and stateful computations, GPUs and TPUs and autoscaling.

The serverless experience abstracts away servers, clusters making it easier for you to focus on building your application.

Autoscaling makes it easy to move from a single node to many with the click of a button.



The screenshot shows the Anyscale web interface for a "Ray Tune Project". The top navigation bar includes "Projects", "Resources", and a user icon. Below the header, there are buttons for "Share" and "Clone". The main content area is titled "About" and describes it as a "Ray Tune Project created by Yunjie." A "Sessions" section contains a table with the following data:

Name	State	Cloud	Max CPU	Max GPU	Machine(s)	Autoscaling	Autosync	Cost	Uptime
ray-tune-experiment...	active	prod (aws)	10	4	m3.2xlarge	on	on	\$0.19/h	0.5h
model-tuning4	active	anyscale_default (aws)	10	4	m5.large head with...	on	on	\$0.19/h	0.5h
my-experiment3	active	prod (aws)	10	4	m5.large head with...	on	on	\$0.19/h	0.5h
model-tuning3	active	anyscale_default (aws)	10	4	m3.2xlarge	on	on	\$0.19/h	0.5h
my-experiment2	active	prod (aws)	10	4	m3.2xlarge	on	on	\$0.19/h	0.5h
model-tuning2	active	anyscale_default (aws)	10	4	m3.2xlarge	on	on	\$0.19/h	0.5h
my-experiment1	active	prod (aws)	10	4	m5.large head with...	on	on	\$0.19/h	0.5h
model-tuning1	active	anyscale_default (aws)	10	4	m3.2xlarge	on	on	\$0.19/h	0.5h
my-experiment	active	prod (aws)	10	4	m3.2xlarge	on	on	\$0.19/h	0.5h

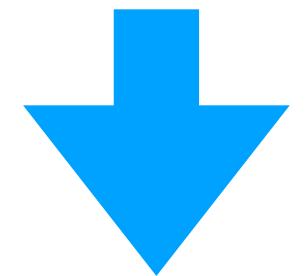
Grid.ai

Seamlessly train hundreds of Machine Learning models on the cloud from your laptop.

Focus on machine learning, not infrastructure.

From the makers of  PyTorch Lightning

```
python model.py --learning_rate 1e-6 |
```



```
grid train --grid_gpus 8 model.py --learning_rate "uniform(1e-6, 1e-1, 20)"
```

Questions?



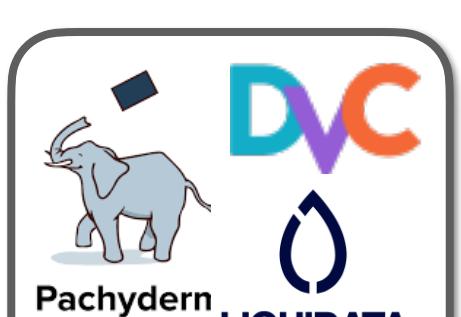
Amazon SageMaker

gradient^o
by Paperspace

FLOYD

DOMINO
DATA LAB

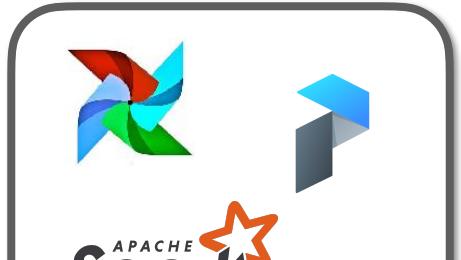
“All-in-one”



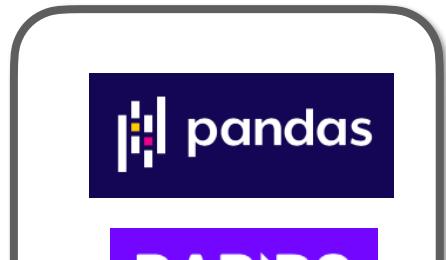
Versioning



Aquarium Labeling



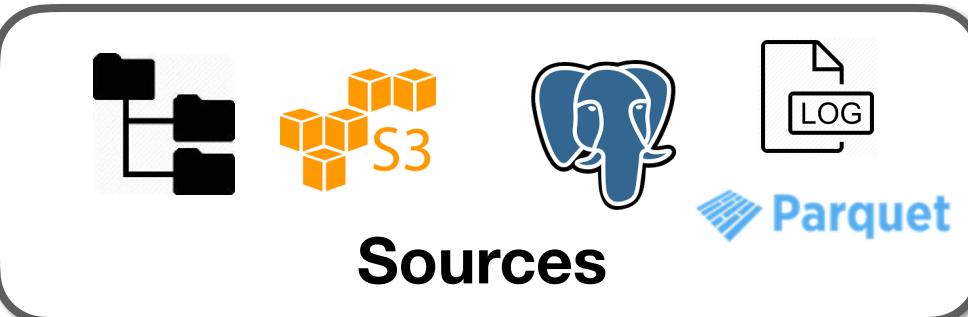
DAGSTER
Processing



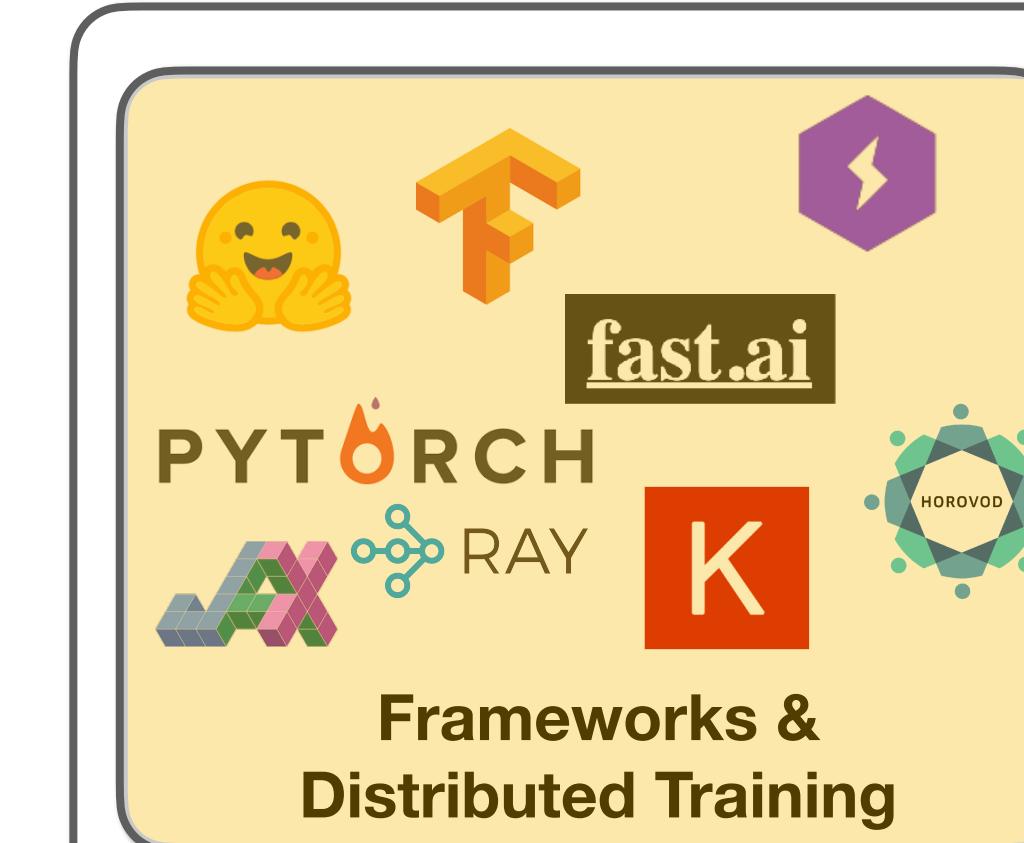
RAPIDS
dbt
Exploration



Data Lake / Warehouse



Sources



Determined AI



Resource Management

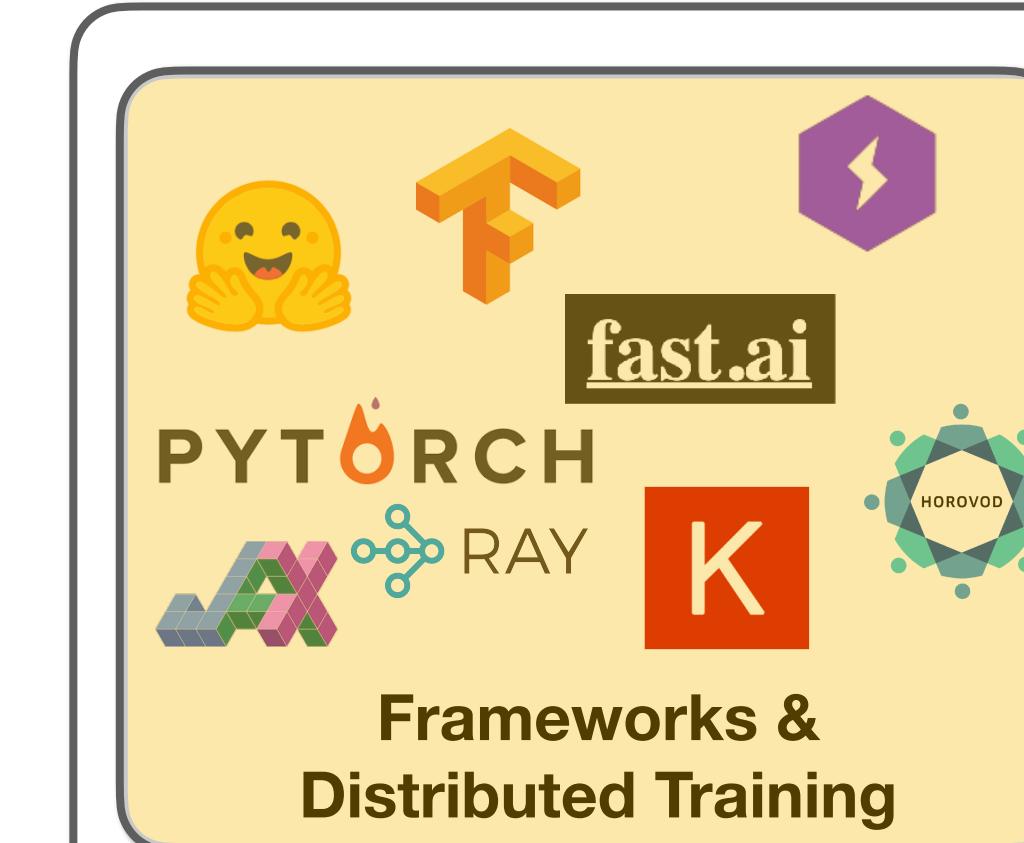


Lambda

CoreWeave



Compute



Determined AI



Resource Management

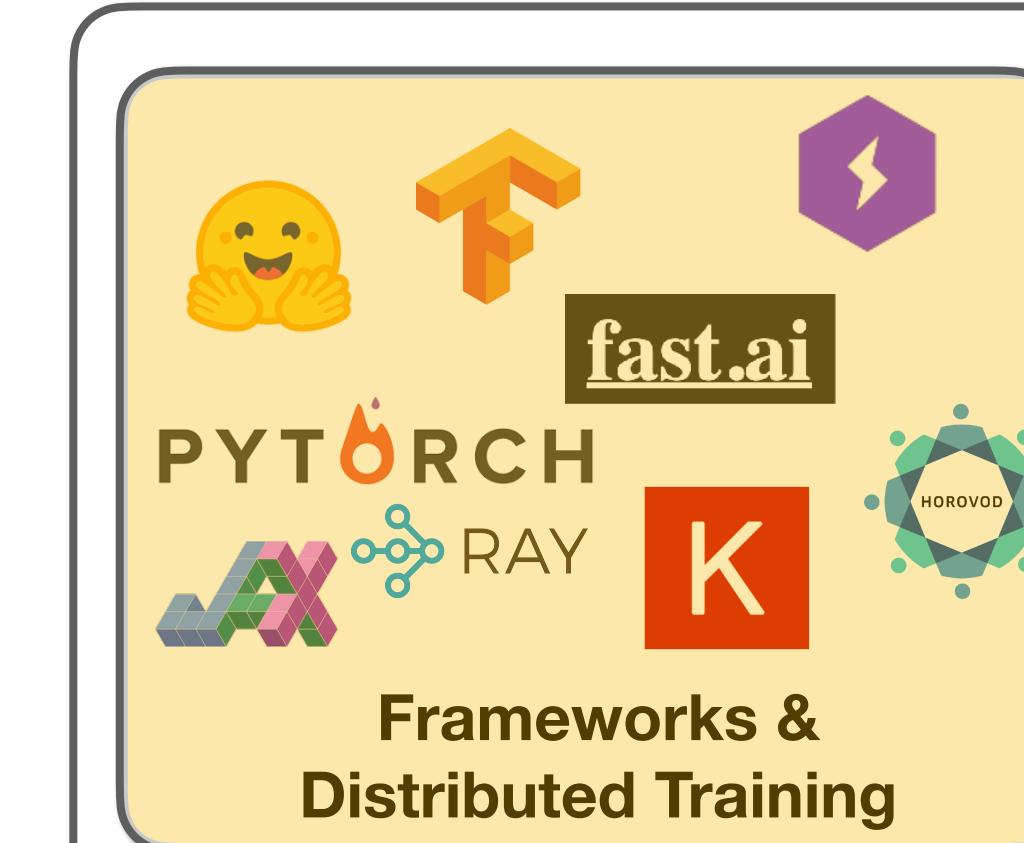


Lambda

CoreWeave



Compute



Determined AI



Resource Management



Lambda

CoreWeave



Compute



Experiment Management



Software Engineering



Experiment Management



Software Engineering



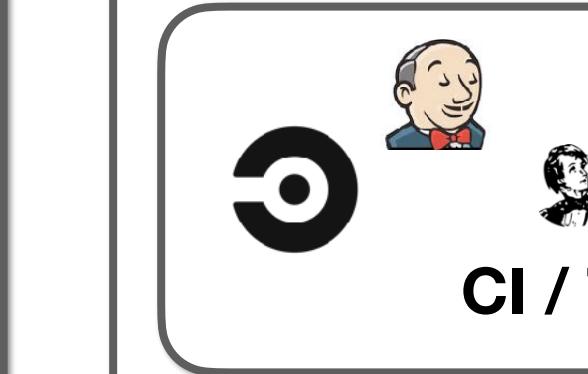
Feature Store



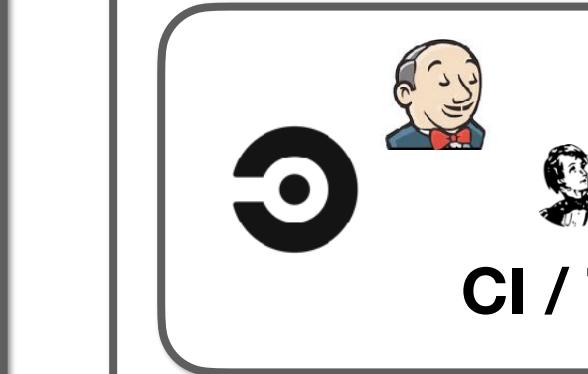
Edge



Edge



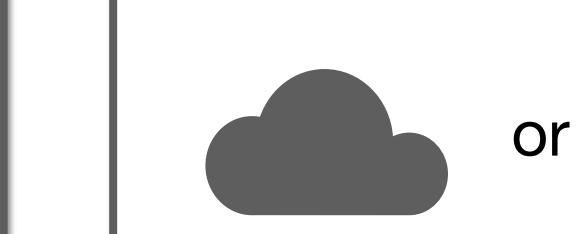
CI / Testing



CI / Testing



or



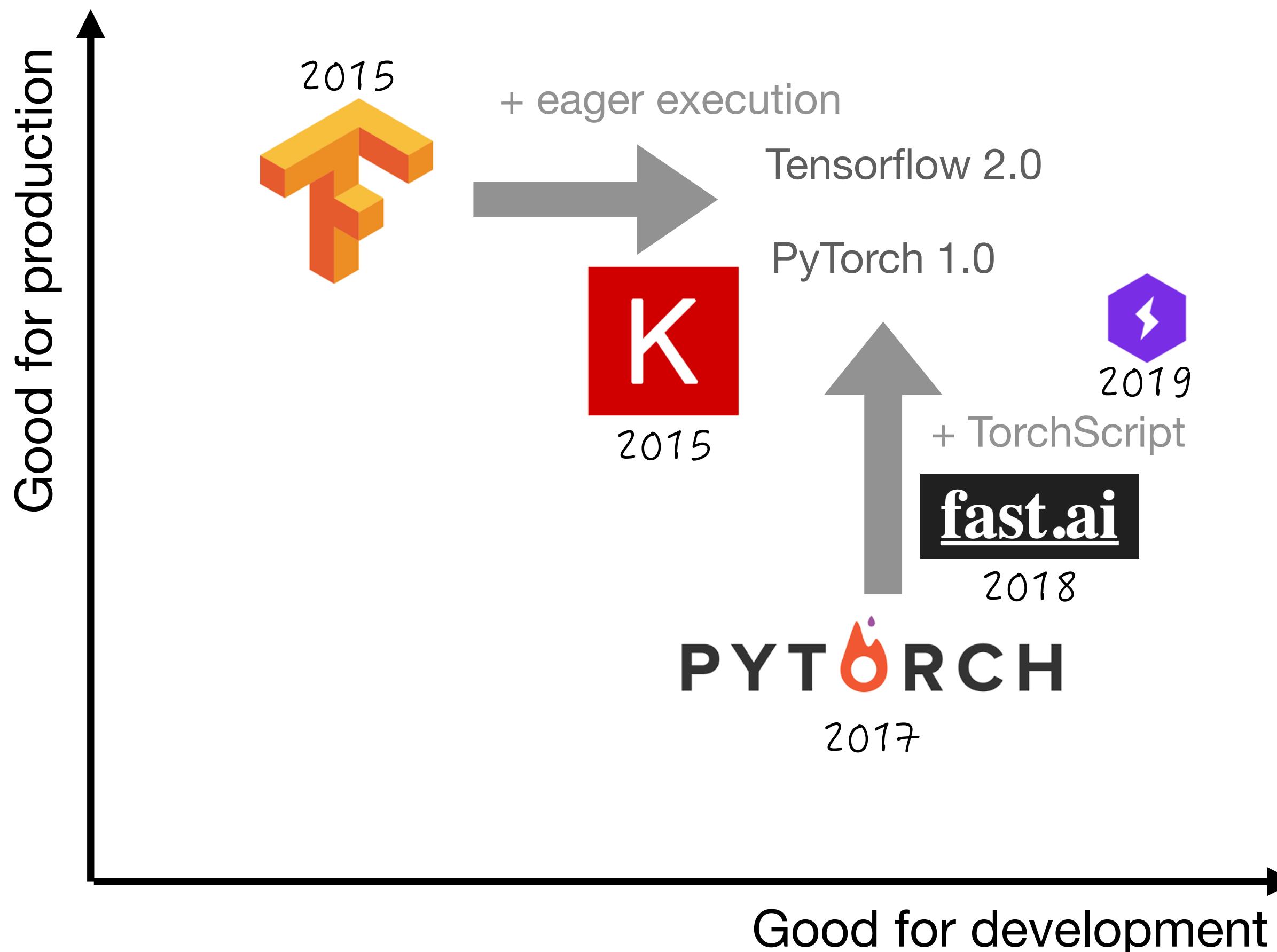
or



or

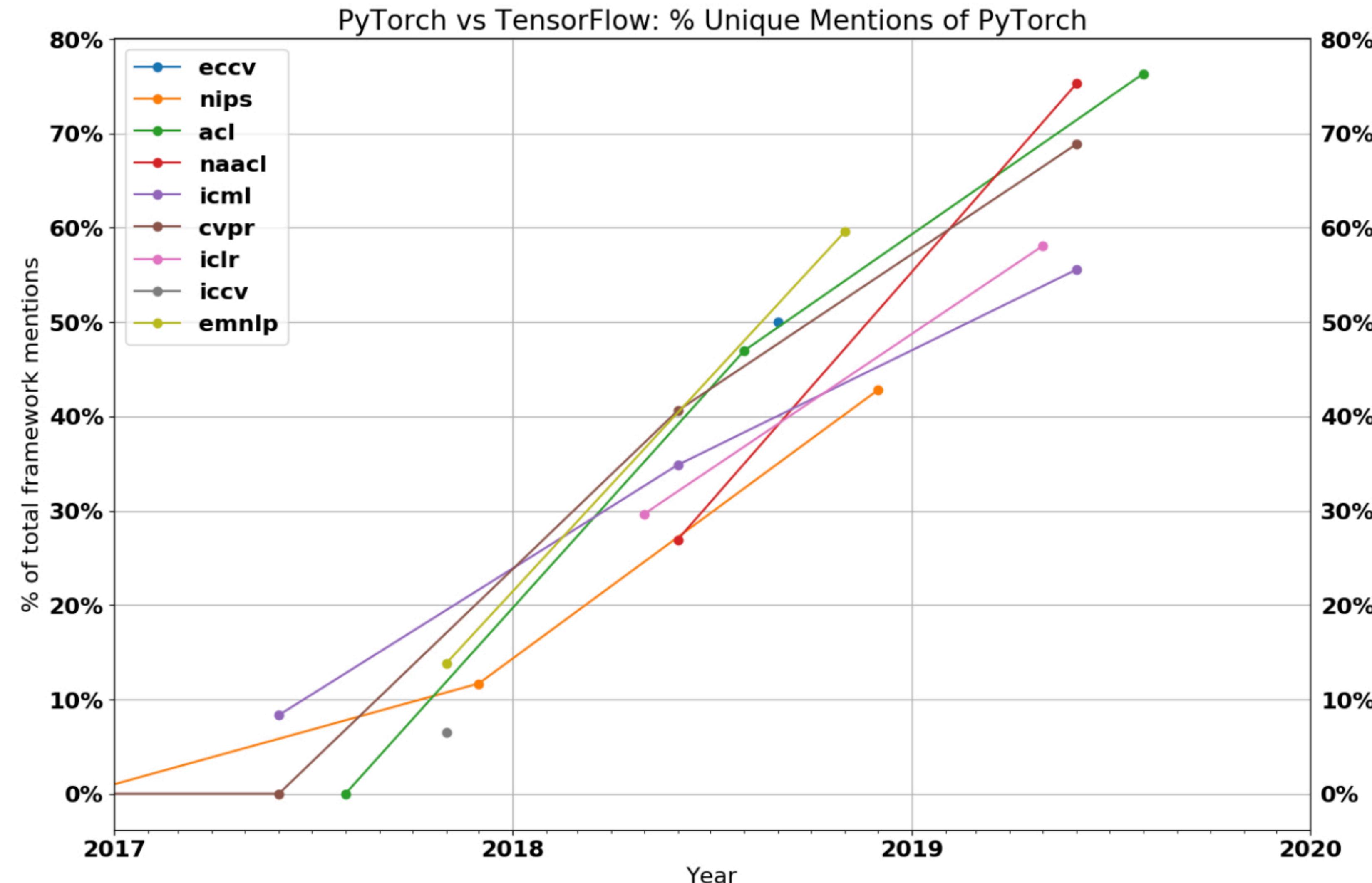
Deployment

Deep Learning Frameworks



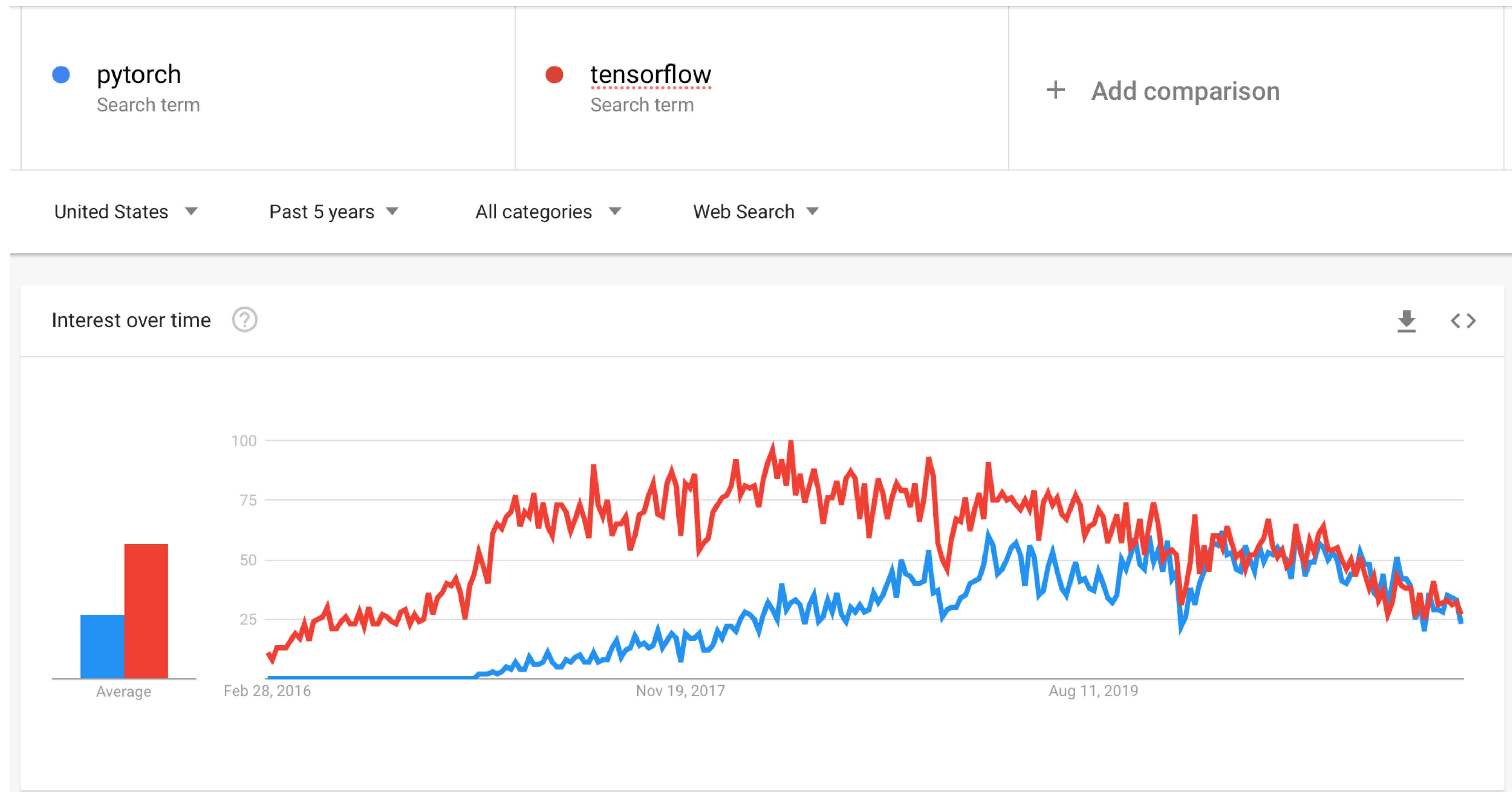
- Unless you have a good reason not to, use Tensorflow/Keras or PyTorch
- Both have converged to the same point:
 - easy development via define-by-run
 - multi-platform optimized execution graph
- Today, most new projects use PyTorch, because of its more Python **dev-friendly** experience
- fast.ai library builds on PyTorch with best practices
- PyTorch-Lightning adds a powerful training loop

PyTorch dominates new development



<https://thegradient.pub/state-of-ml-frameworks-2019-pytorch-dominates-research-tensorflow-dominates-industry/>

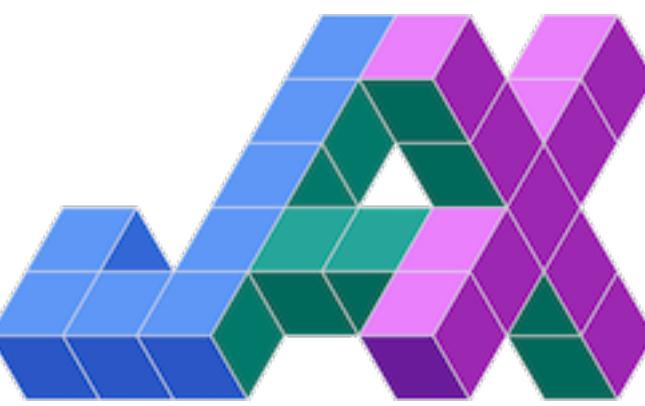
PyTorch and Tensorflow



Why do we need frameworks?

- Deep Learning is not lot of code with a matrix math library (Numpy)
- BUT: auto-differentiation and CUDA are a lot of work
- ...Also all the layer types, optimizers, data interfaces, etc.

```
17 <class Linear:  
18     def __init__(self, input_dim: int, num_hidden: int = 1):  
19         self.weights = np.random.randn(input_dim, num_hidden) * np.sqrt(2. / input_dim)  
20         self.bias = np.zeros(num_hidden)  
21  
22     def __call__(self, x):  
23         self.x = x  
24         output = x @ self.weights + self.bias  
25         return output  
26  
27     def backward(self, gradient):  
28         self.weights_gradient = self.x.T @ gradient  
29         self.bias_gradient = gradient.sum(axis=0)  
30         self.x_gradient = gradient @ self.weights.T  
31         return self.x_gradient  
32  
33     def update(self, lr):  
34         self.weights = self.weights - lr * self.weights_gradient  
35         self.bias = self.bias - lr * self.bias_gradient
```



- Recent project from Google that's gaining steam
- Numpy + auto-differentiation and compilation to GPU/TPU code
- Not just for deep learning!

<https://github.com/google/jax#quickstart-colab-in-the-cloud>



Hugging face

- Tons of NLP-focused model architectures (and pre-trained weights) for both PyTorch and Tensorflow

Tasks

- Fill-Mask
- Question Answering
- Summarization
- Table Question Answering
- Text Classification
- Text Generation
- Text2Text Generation
- Token Classification
- Translation
- Zero-Shot Classification

Libraries

- PyTorch
- TensorFlow

Datasets

- wikipedia
- squad
- c4
- bookcorpus
- dcepc
- europarl
- jrc-acquis
- CLUECorpusSmall
- oscar
- squad_v2

+ 194

Languages

- en
- es
- fr
- sv
- fi
- de
- multilingual
- zh

+ 328

Licenses

- apache-2.0
- mit
- gpl-3.0

+ 12

Models 5950 ↑ Sort: Most Downloads

Model	Description	Last Updated	Downloads
bert-base-uncased	Fill-Mask	Updated Dec 11, 2020	23,951k
distilbert-base-uncased	Fill-Mask	Updated Dec 11, 2020	9,335k
cl-tohoku/bert-base-japanese-whole-word-masking	Fill-Mask	Updated Jan 25	4,276k
jplu/tf-xlm-roberta-base	Fill-Mask	Updated Dec 11, 2020	3,740k
roberta-base	Fill-Mask	Updated Dec 11, 2020	2,196k
xlm-roberta-base	Fill-Mask	Updated Dec 11, 2020	2,080k
bert-large-uncased	Fill-Mask	Updated Jan 13	1,959k
bert-base-cased	Fill-Mask	Updated Dec 15, 2020	1,774k
bert-large-cased	Fill-Mask	Updated Jan 13	1,639k
valhalla/t5-small-qa-qg-hl	Text2Text Generation	Updated Dec 11, 2020	1,170k
t5-small	Translation	Updated Dec 11, 2020	816k
gpt2	Text Generation	Updated Dec 11, 2020	791k
distilbert-base-uncased-finetuned-sst-2-english	Text Classification	Updated 15 days ago	770k
roberta-large	Fill-Mask	Updated Dec 11, 2020	738k
valhalla/t5-small-qg-hl	Text2Text Generation	Updated Dec 11, 2020	718k
sentence-transformers/distilbert-base-nli-stsb...		Updated Aug 31, 2020	679k
facebook/bart-large-mnli	Zero-Shot Classification	Updated Dec 11, 2020	638k
microsoft/BioMedNLP-PubMedBERT-base-uncased-abst...		Updated Aug 8, 2020	557k

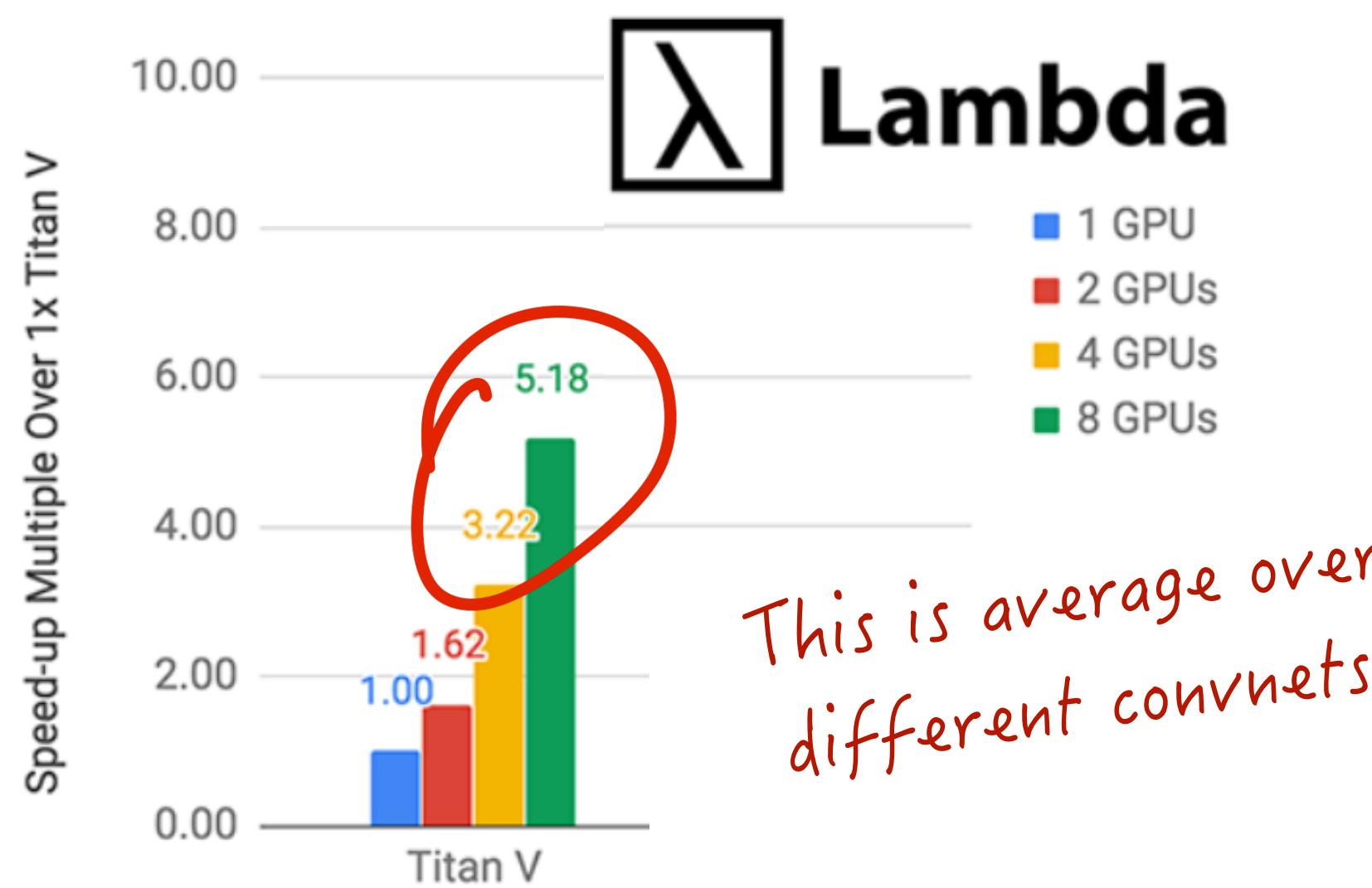
Distributed Training

- Using multiple GPUs and/or machines to train a single model.
- More complex than simply running different experiments on different GPUs
- A **must-do** on big datasets and large models

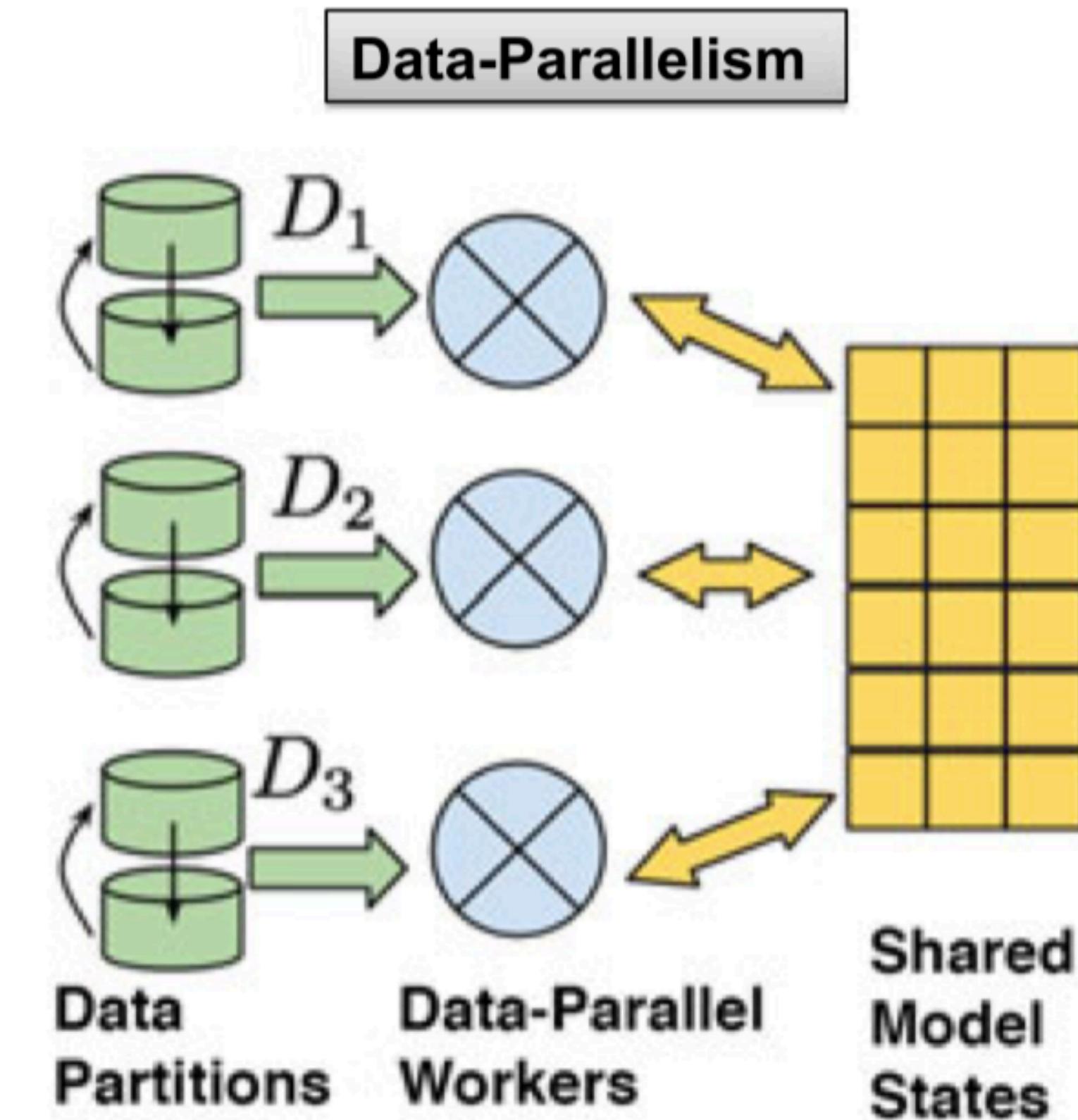
Data Parallelism

- If iteration time is too long, try training in data parallel regime
- "For convolution, expect 1.9x/3.5x speedup for 2/4 GPUs.

<http://timdettmers.com/2018/11/05/which-gpu-for-deep-learning/>



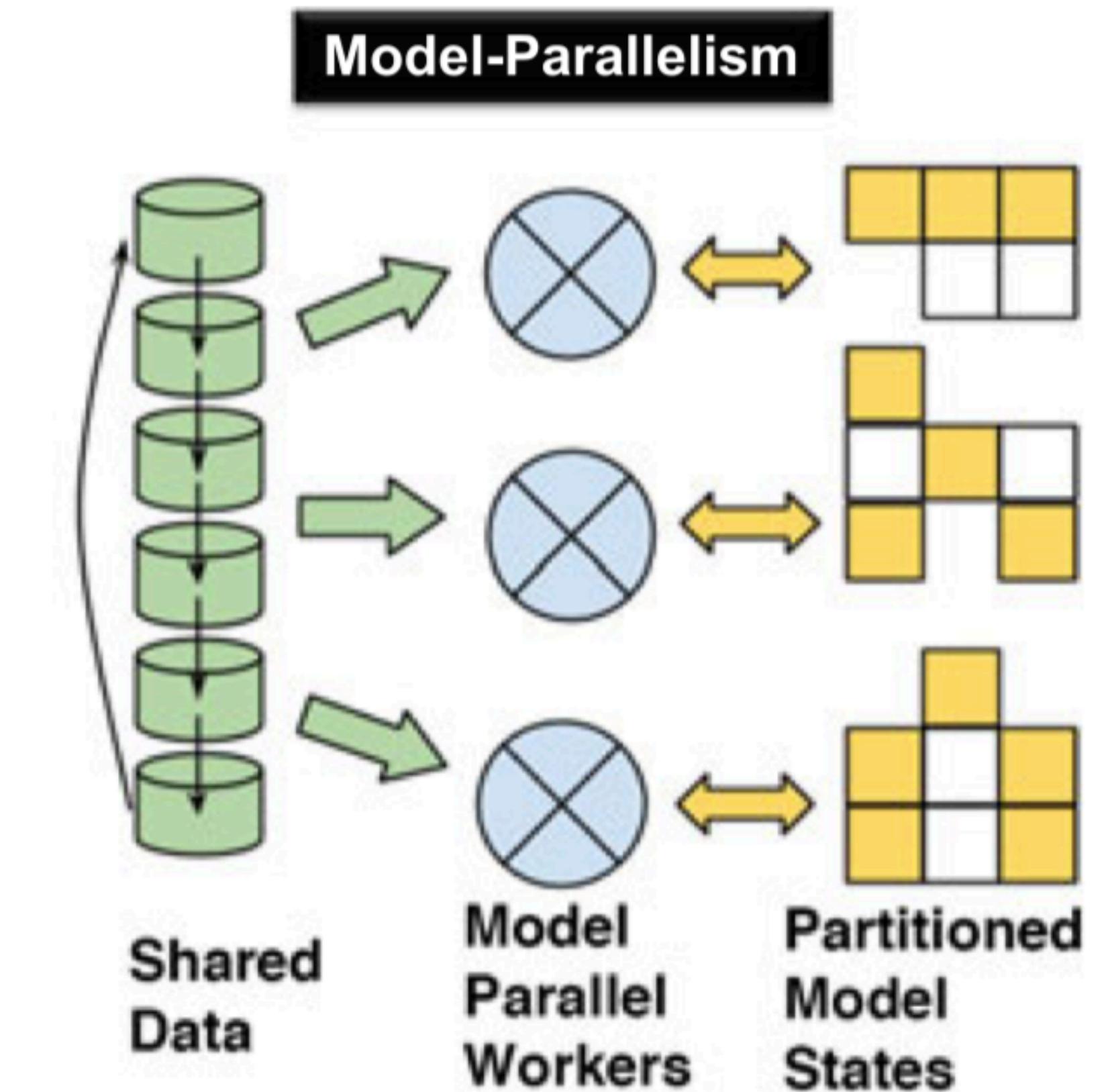
<https://lambdalabs.com/blog/titan-v-deep-learning-benchmarks/>



http://www.cs.cmu.edu/~pengtaox/papers/petuum_15.pdf

Model Parallelism

- Model parallelism is necessary when **model does not fit on a single GPU**
 - Introduces a lot of complexity and is usually not worth it. (But this is changing.)
 - Better to buy the largest GPU you can, and/or use gradient checkpointing



http://www.cs.cmu.edu/~pengtao/papers/petuum_15.pdf

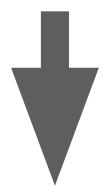
Data-parallel PyTorch

- Can be quite easy:

```
model = Model(input_size, output_size)
if torch.cuda.device_count() > 1:
    print("Let's use", torch.cuda.device_count(), "GPUs!")
    # dim = 0 [30, xxx] -> [10, ...], [10, ...], [10, ...] on 3 GPUs
model = nn.DataParallel(model)

model.to(device)

for data in rand_loader:
    input = data.to(device)
    output = model(input)
    print("Outside: input size", input.size(),
          "output_size", output.size())
```



```
Let's use 2 GPUs!
In Model: input size torch.Size([15, 5]) output size torch.Size([15, 2])
In Model: input size torch.Size([15, 5]) output size torch.Size([15, 2])
Outside: input size torch.Size([30, 5]) output_size torch.Size([30, 2])
In Model: input size torch.Size([15, 5]) output size torch.Size([15, 2])
In Model: input size torch.Size([15, 5]) output size torch.Size([15, 2])
```

Data-parallel PyTorch-Lightning

- Single node:

python training/
run_experiment.py **--gpus=8** --
accelerator=ddp

- And using SLURM to run on
multiple nodes:

python training/
run_experiment.py --gpus=8 --
accelerator=ddp **--nodes=4**

```
# (submit.sh)
#!/bin/bash -l

# SLURM SUBMIT SCRIPT
#SBATCH --nodes=4
#SBATCH --gres=gpu:8
#SBATCH --ntasks-per-node=8
#SBATCH --mem=0
#SBATCH --time=0-02:00:00

# activate conda env
source activate $1

# debugging flags (optional)
export NCCL_DEBUG=INFO
export PYTHONFAULTHANDLER=1

# on your cluster you might need these:
# set the network interface
# export NCCL_SOCKET_IFNAME=^docker0,lo

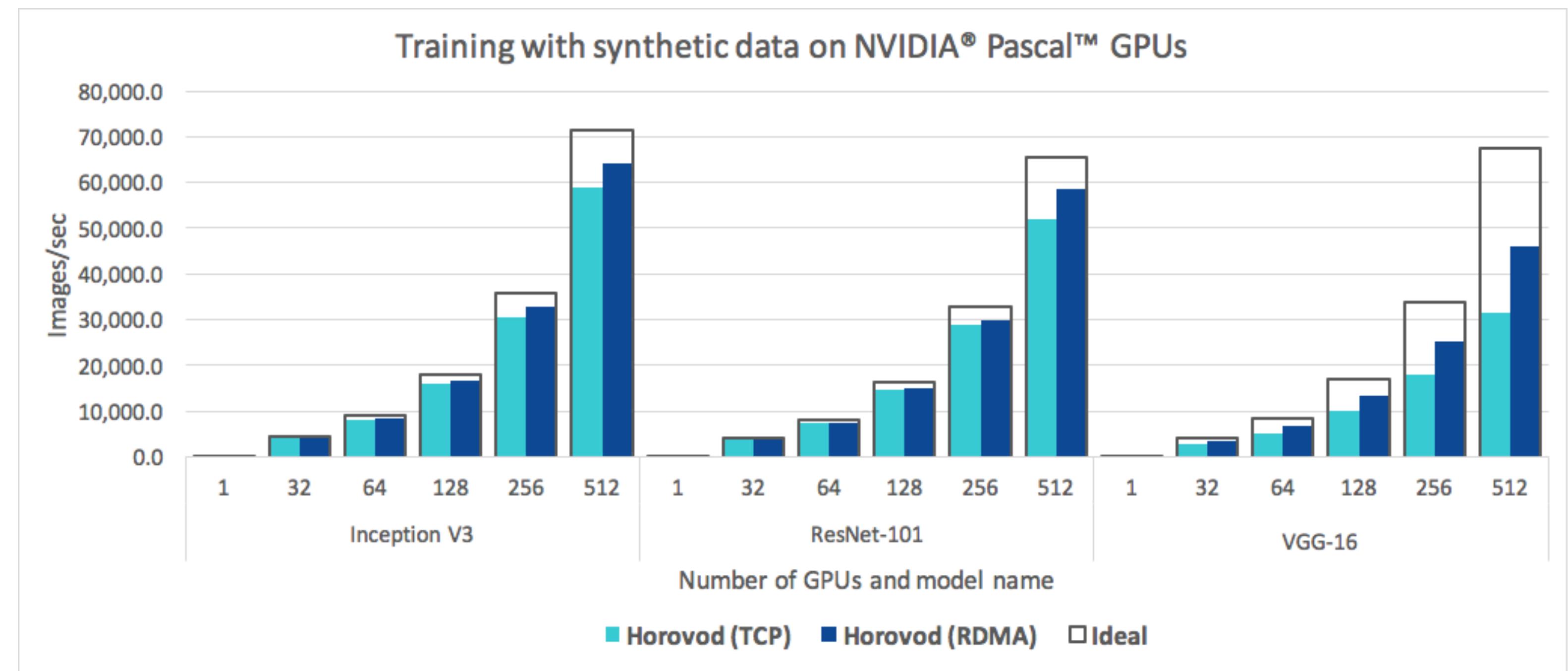
# might need the latest CUDA
# module load NCCL/2.4.7-1-cuda.10.0

# run script from above
srun python3 train.py
```

Horovod



- Distributed training framework for Tensorflow, Keras, and PyTorch
- Uses MPI (standard multi-process communication framework) instead of Tensorflow parameter servers
- Could be an easier experience for multi-node training



Ray (from Anyscale)

- Ray is open-source project for effortless, stateful distributed computing in Python
- From Berkeley!

```
ray.init(args.address)

trainer1 = PyTorchTrainer(
    model_creator,
    data_creator,
    optimizer_creator,
    loss_creator,
    num_replicas=<NUM_GPUS_YOU_HAVE> * <NUM_NODES>,
    use_gpu=True,
    batch_size=512,
    backend="nccl")

stats = trainer1.train()
print(stats)
trainer1.shutdown()
print("success!")
```

Then, start a Ray cluster via [autoscaler](#) or [manually](#).

```
ray up CLUSTER.yaml
python train.py --address="localhost:<PORT>"
```

<https://ray.readthedocs.io/en/latest/walkthrough.html>

Questions?



Amazon SageMaker

gradient^o
by Paperspace

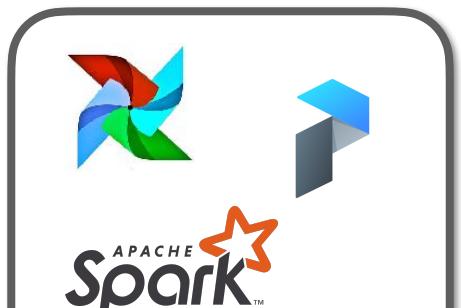
FLOYD

DOMINO
DATA LAB

“All-in-one”



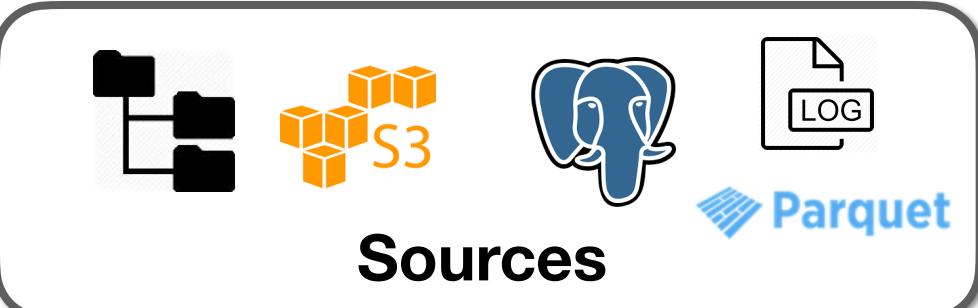
Versioning



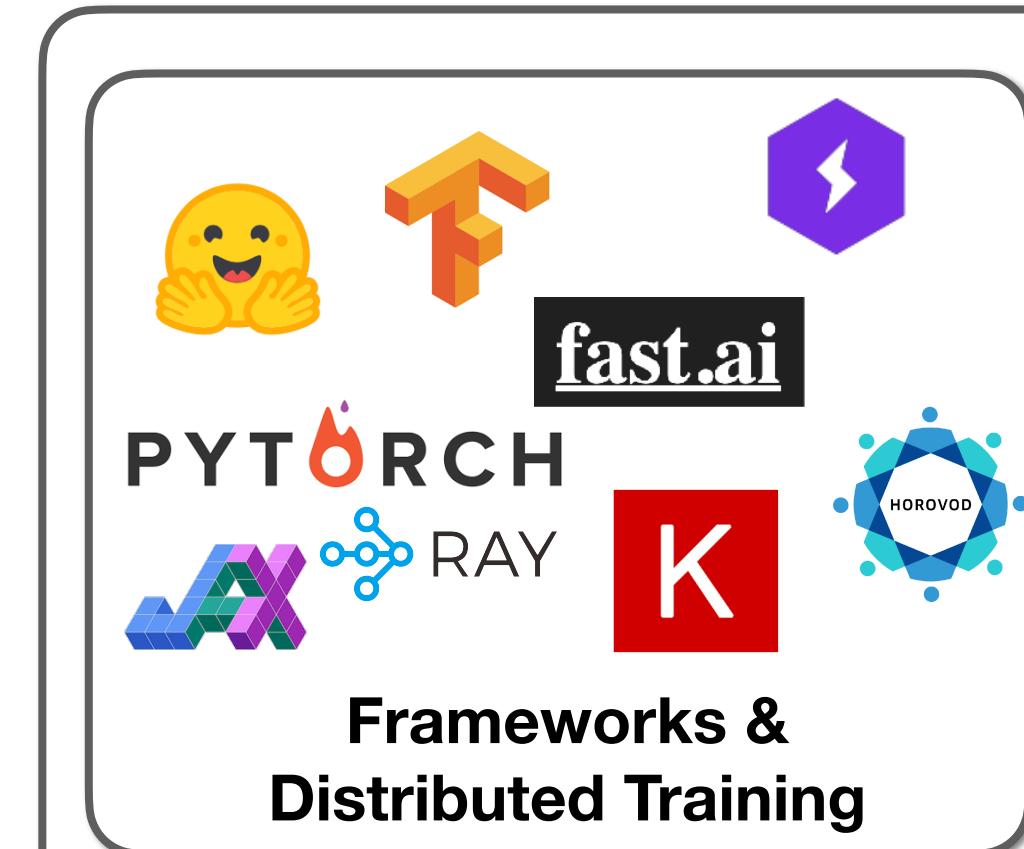
Processing



Data Lake / Warehouse



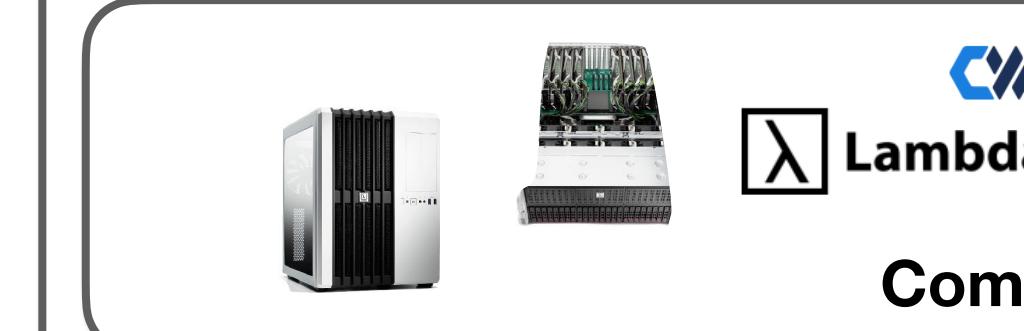
Data



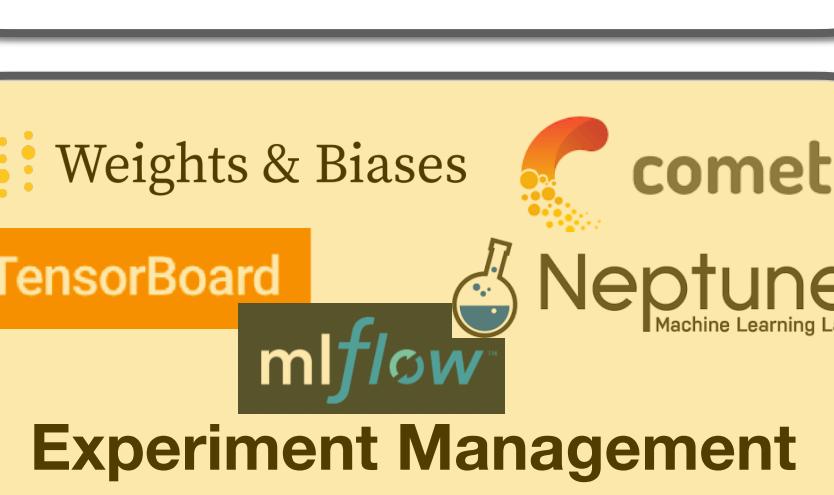
Frameworks &
Distributed Training



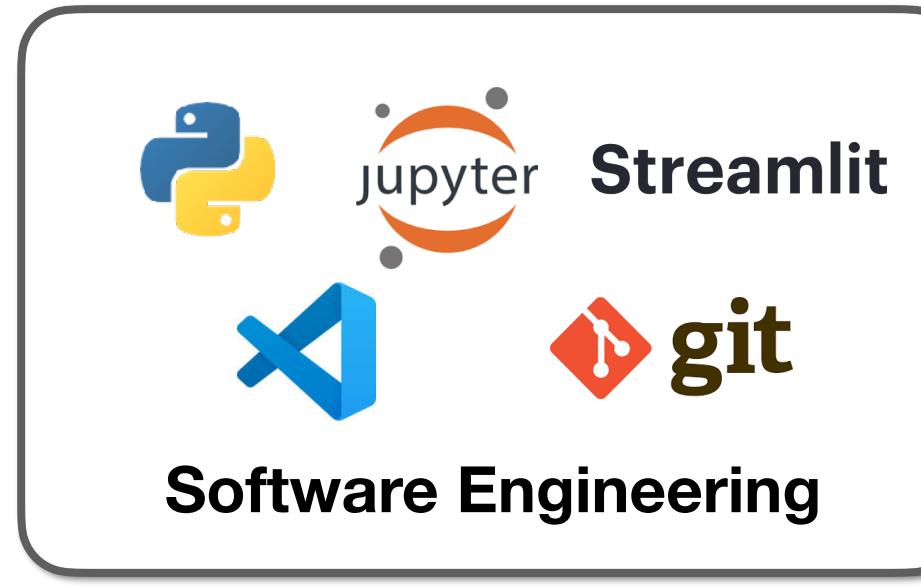
Resource Management



Training/Evaluation



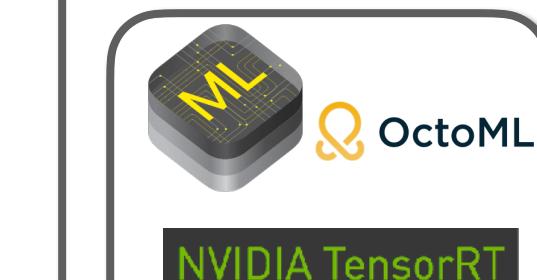
Experiment Management



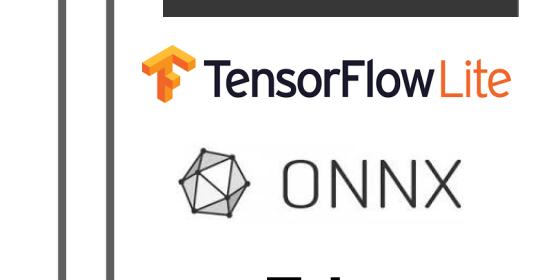
Software Engineering



Feature
Store



NVIDIA TensorRT



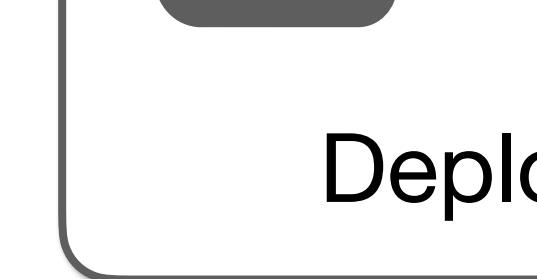
ONNX



Monitoring



CI / Testing



Deployment

Experiment Management

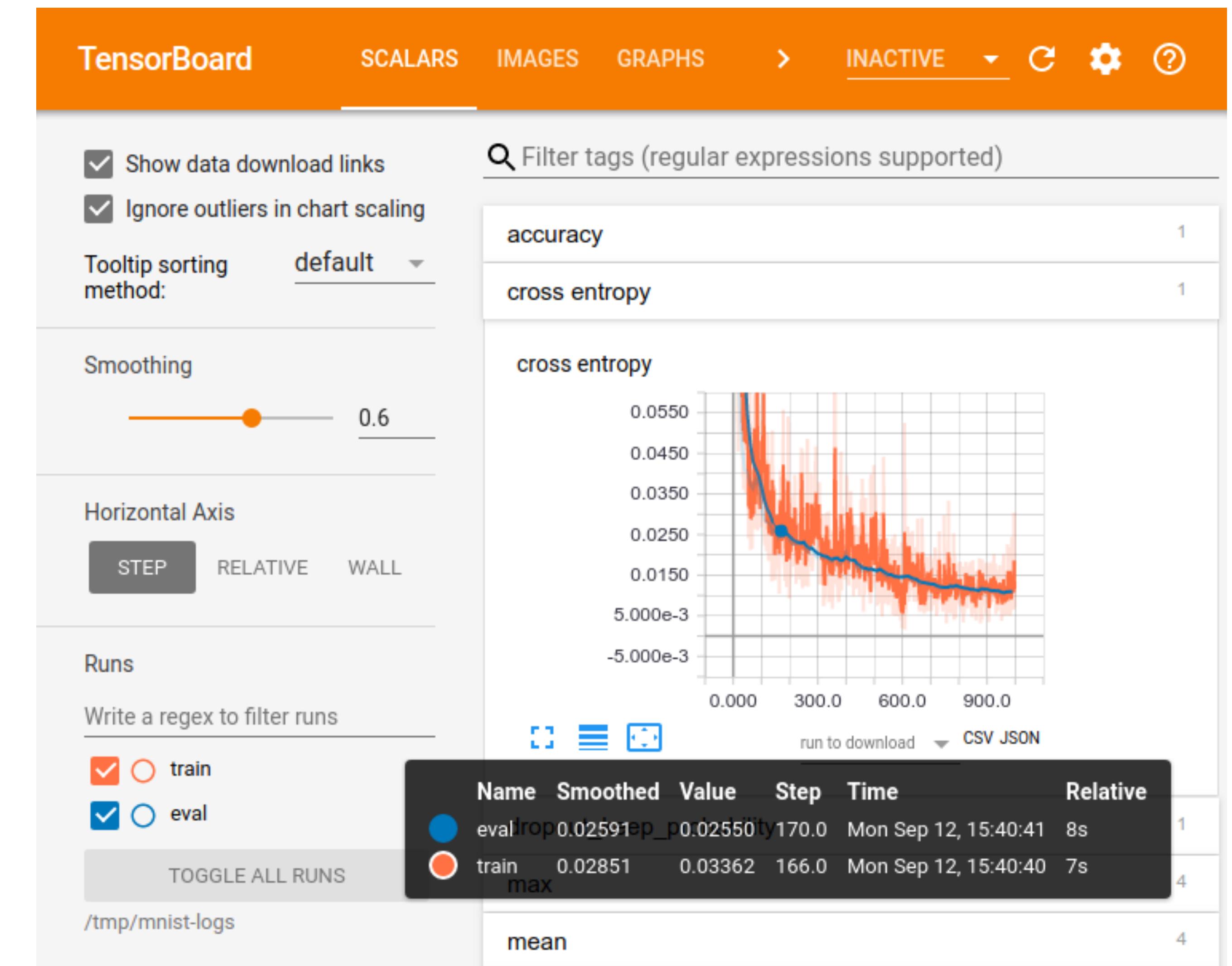
- Even running one experiment at a time, can lose track of which code, parameters, and dataset generated which trained model.
- When running multiple experiments, problem is much worse.

11	Dataset	Split (trian/dev/test)	0.7/0.2/0.1	0.7/0.2/0.1	0.7/0.2/0.1	0.7/0.15/0.15	0.7/0.15/0.15
12		Class ratio (train/dev/test)	0.42/0.42/0.42	0.42/0.42/0.42	0.42/0.42/0.42	/0.3/0.3	/0.3/0.3
13		train/dev/test size	4871/1392/696	4871/1392/696	5315/1518/760	5315/1139/1139	5315/1139/1139
14	Training hyperparameters	Learning rate	1.00E-05	1.00E-05	1.00E-05	1.00E-05	1.00E-05
15		epoch	3	2	1	5	6
16		batch size	32	32	32	32	32
17	Results	accuracy	0.88304595	0.8650862069	0.8687747	0.86997364	0.65
18		f1	0.82495437	0.8108753316	0.82383946	0.81827954	0.44
19		precision	0.878865	0.7848381601	0.8407407	0.8556561	0.56
20		recall	0.7780239	0.8389705882	0.8076923	0.78442625	0.36
21		tp	1398	1402	1460	1334	1130
22		tn	1692	1663	1707	1543	1504
23		fp	1113	1142	1161	1108	1148
24		fn	1189	1185	1190	1154	1357
25		loss	0.59637538	0.594134	0.594134	0.6037084	0.594134
26	Test results	accuracy	0.90747	0.90747	0.88026	0.88314	0.75847
27		f1	0.85636	0.85636	0.83108	0.83469	0.5915
28		precision	0.90934	0.90934	0.86689	0.87027	0.77626
29		recall	0.8099	0.8099	0.79846	0.80226	0.48604
30							

<https://towardsdatascience.com/tracking-ml-experiments-using-mlflow-7910197091bb>

Tensorboard

- A fine solution for **single experiments**
- Gets unwieldy to manage many experiments, and to properly store past work



MLFlow tracking

An open source platform for the machine learning lifecycle

- Self-hosted solution from DataBricks

/Shared/experiments/cryptocurrency/analysis-forecasting-1

Experiment ID: 450992 Artifact Location: dbfs:/databricks/mlflow/450992

Search Expression: metrics.rmse < 1 and params.model = "tree"

State: Active ▾

Params: alpha, lr

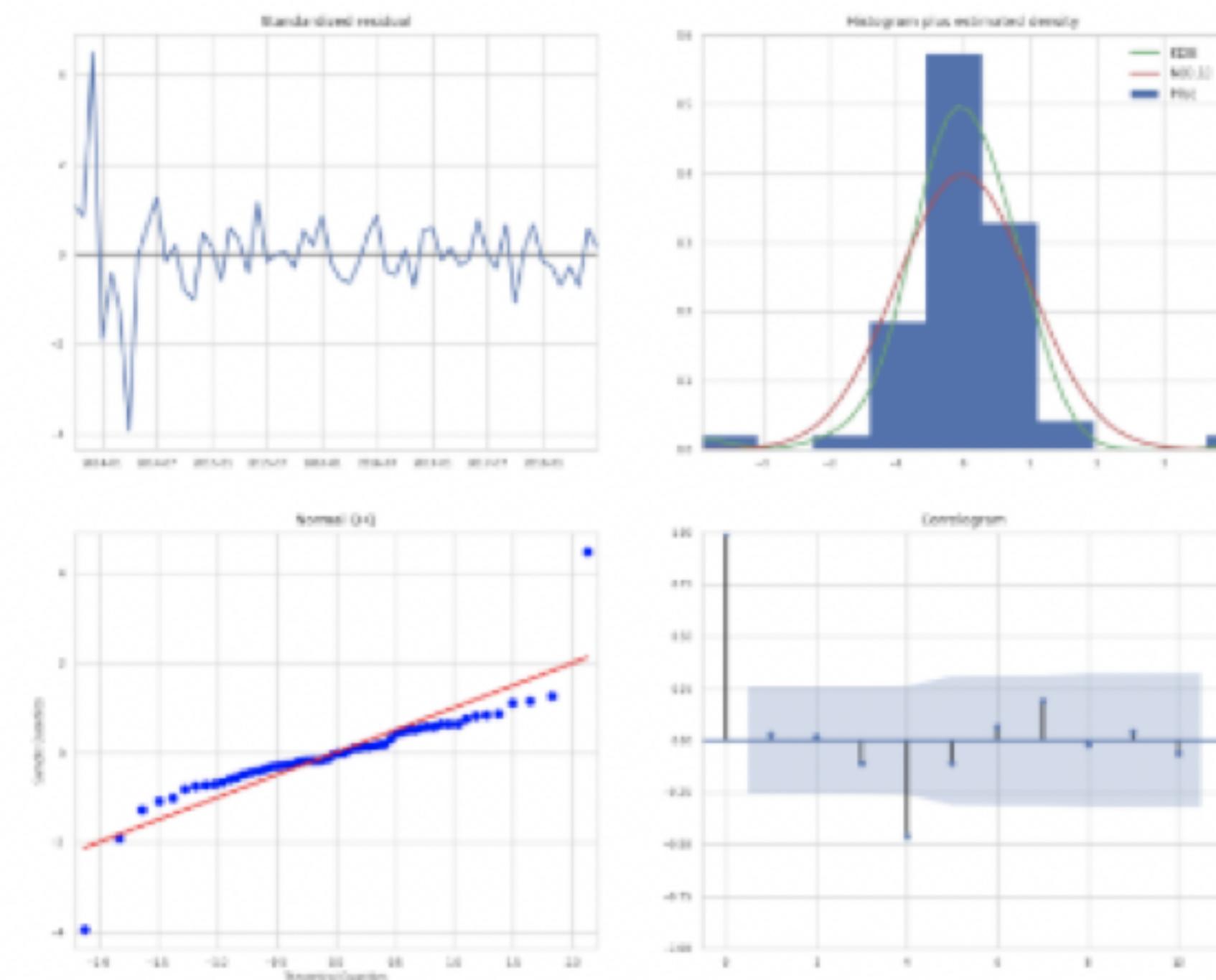
Metrics: rmse, r2

Clear

8 matching runs

Compare Delete Download CSV  

	Date	User	Run Name	Source	Version	Tags	Parameters	Metrics
							param-ps	param-qs
1	2019-07-13 08:20:35	hafidzz	arima_param	cryptocurrency-price-forecasting-after			2	2
2	2019-07-13 08:20:34	hafidzz	arima_param	cryptocurrency-price-forecasting-after			1	2
3	2019-07-13 08:20:33	hafidzz	arima_param	cryptocurrency-price-forecasting-after			0	2
4	2019-07-13 08:20:32	hafidzz	arima_param	cryptocurrency-price-forecasting-after			2	1
5	2019-07-13 08:20:32	hafidzz	arima_param	cryptocurrency-price-forecasting-after			1	1
6	2019-07-13 08:20:31	hafidzz	arima_param	cryptocurrency-price-forecasting-after			0	1
7	2019-07-13 08:20:30	hafidzz	arima_param	cryptocurrency-price-forecasting-after			2	0
8	2019-07-13 08:20:30	hafidzz	arima_param	cryptocurrency-price-forecasting-after			1	0



<https://towardsdatascience.com/tracking-ml-experiments-using-mlflow-7910197091bb>



Supercharge Machine Learning

Comet lets you track code, experiments, and results on ML projects. It's fast, simple, and free for open source projects.

[Sign Up](#)

The screenshot displays the Comet.ml interface with several key features:

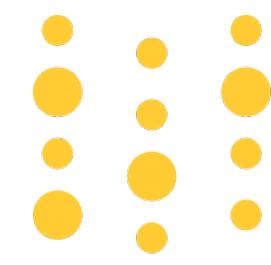
- Experiment Overview:** A table lists experiments with columns for Status, Experiment key, File name, Duration, Commit SHA, Local start time, Loss, and Accuracy. One experiment is highlighted: `e0b82148c5364939a...` (mnist.py) with a duration of 00:00:17, Commit SHA `f28ce451`, and accuracy of 1.0.
- Hyperparameter Optimization:** A chart shows the relationship between hyperparameters (lstm_size_1, lstm_size_2, dropout_probability, learning rate) and accuracy. A specific model entry is highlighted: `Model 149: 0.3642867943030753`.
- Code Comparison:** Two experiments are compared side-by-side, showing their code differences. The left pane shows `/Users/Gideon 1/Documents/dev/semantic...` and the right pane shows `/Users/Gideon 1/Document...`. Differences are highlighted in yellow.
- Experiment Detail:** A detailed view of the `mnist.py` experiment shows a chart of accuracy and loss over steps, along with a table of current metric values (acc: 0.938, loss: 0.207) and a list of installed packages.

Comet.ml

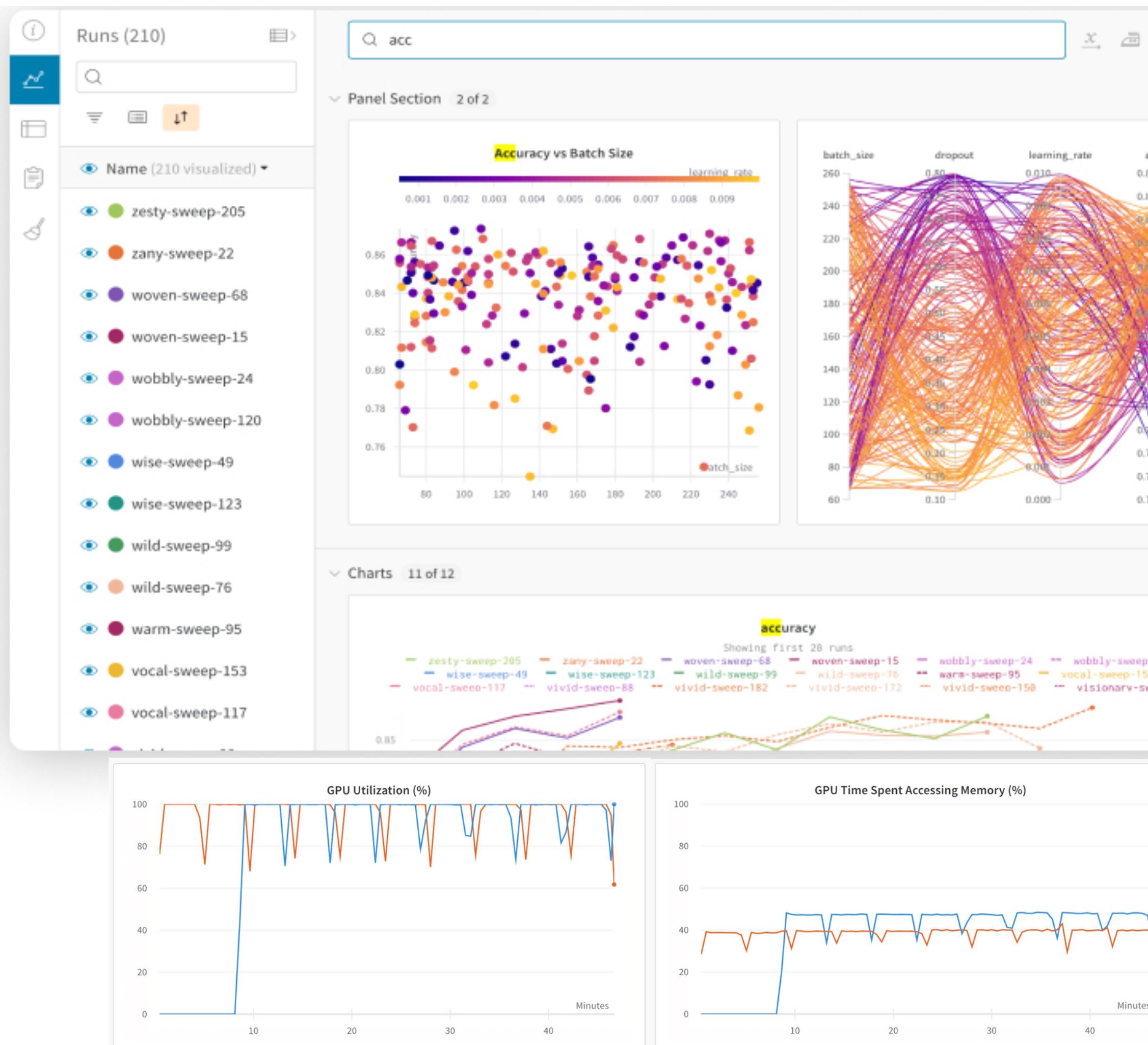
Compare

Experiments

Comet lets you compare different experiments and see the differences in code, hyper-params, and many other data points.



Weights & Biases



Runs (398)

Filter Group Sort Tag Move

Name (398 visualized)	Runtime	Notes	Tags	batch_size	encoder	epochs	learning_ra	num_train	num_valid	training_st	weight_dec	acc
apricot-firebrand-436	4m 52s	Add notes		6	resnet34	10	0.001311	3479	483	3	0.08173	0.8286
icy-grass-435	4m 54s	Add notes		6	resnet34	10	0.001311	3494	505	3	0.08173	0.7693
curious-meadow-434	50m 6s	Add notes		6	resnet34	10	0.001311	3495	494	3	0.08173	0.8615
laced-galaxy-433	33s	Add notes		6	resnet34	10	0.001311	3521	495	3	0.08173	-
best car acc (50% dat	47m 52s	reprod...	seg_masks	6	resnet34	10	0.001311	3524	492	3	0.08173	0.8823
best traffic acc (50%	46m 42s	reprod...	seg_masks	8	resnet18	10	0.001	3523	492	2	0.097	0.888
best human iou (50%	31m 34s	reprod...	seg_masks	7	alexnet	10	0.0009084	1405	190	3	0.097	0.8334
best overall IOU (20%	20m 23s	reprod...	seg_masks	7	resnet34	10	0.001367	1376	205	2	0.06731	0.8726
hopeful-violet-428	34s	Add notes		8	resnet34	1	0.001	331	56	2	0.097	-
major-firefly-427	8s	Add notes		8	resnet34	1	0.001	355	46	2	0.097	-
vibrant-cherry-426	6m 12s	Add notes		8	resnet34	10	0.001	347	42	2	0.097	0.8474
good-cosmos-425	43s	Add notes		8	resnet34	10	0.001	359	40	2	0.097	0.4031
whole-serenity-424	44s	Add notes		8	resnet34	10	0.001	361	49	2	0.097	0.5169
daily-river-423	1m 8s	Add notes		8	resnet34	10	0.001	693	104	2	0.097	-
worldly-totem-422	12m 54s	Add notes		8	resnet34	10	0.001	682	97	2	0.097	0.8566
jumping-voice-421	11m 59s	Add notes		8	resnet34	10	0.001	725	92	2	0.097	0.8504

- What we use in lab



The View from the Driver's Seat

Semantic segmentation for scene parsing on Berkeley Deep Drive 100K

Stacey Svetlichnaya

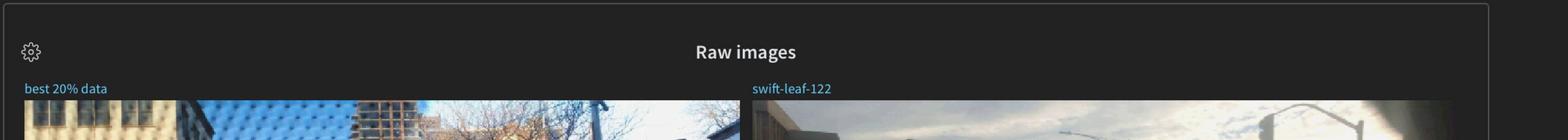


Understand a dashboard scene with semantic segmentation

A self-driving car must functionally understand the road and its environment the way a human would from the driver's seat. One promising computer vision approach is **semantic segmentation**: parse visual scenes from a car dashboard camera into relevant objects (cars, pedestrians, traffic signs), foreground (road, sidewalk), and background (sky, building). Semantic segmentation annotates an image with object types, labeling meaningful subregions as a tree, bus, cyclist, etc. For a given car dashboard photo, this means labeling *every pixel* as belonging to a subregion.

Below you can see examples in two columns: raw images, the model's predictions, and the ground truth (correct labeling). Buildings are orange, cars are pink, road is cobalt blue, and pedestrians are beige. In the left column, the model can't differentiate between a pedestrian and a rider on a bicycle (magenta and cyan in ground truth, beige in prediction). Note how the hazy conditions in the right column make the model predictions blurry around the boundaries between dashboard and road, or vehicle and road).

Example segmentation maps



- Publish reports with embedded charts, figures, etc



Amazon SageMaker

gradient^o
by Paperspace

FLOYD

DOMINO
DATA LAB

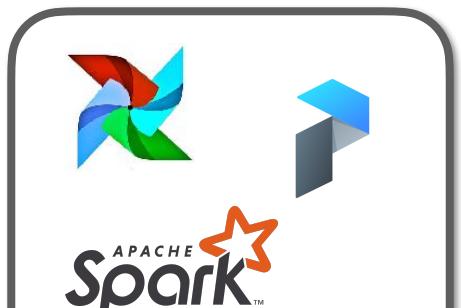
“All-in-one”



Versioning



Labeling



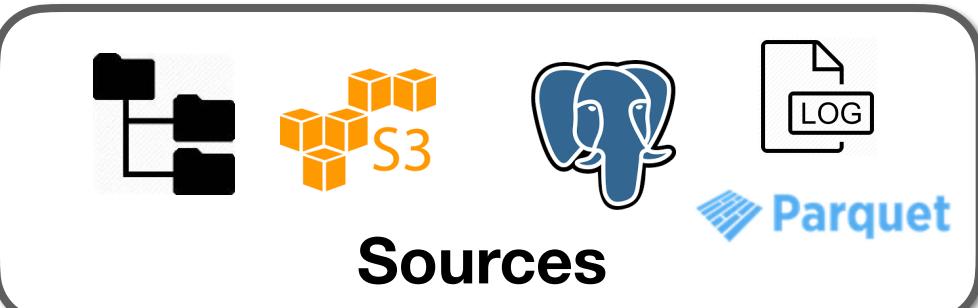
Processing



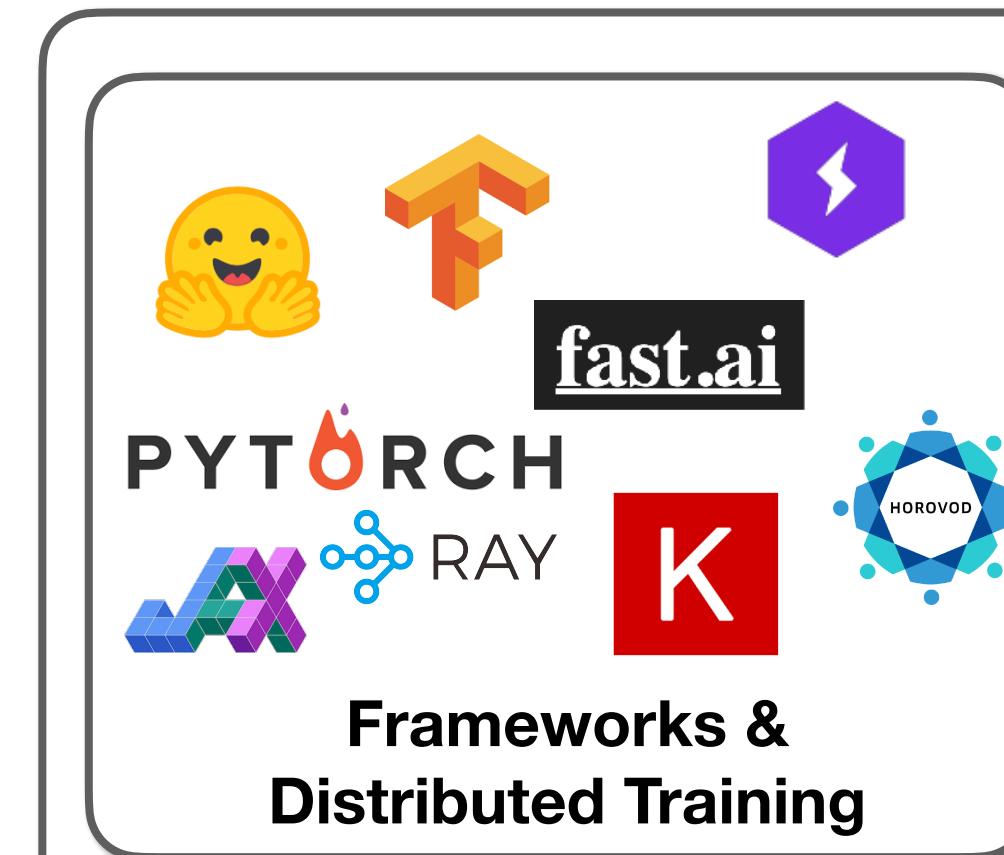
Exploration



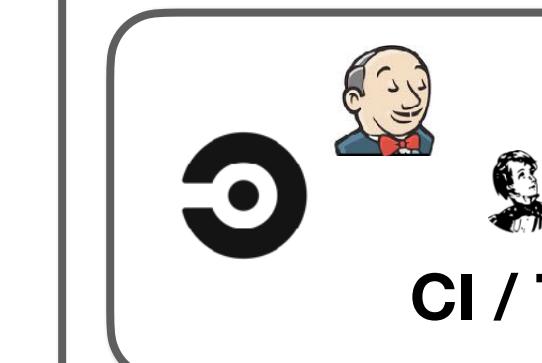
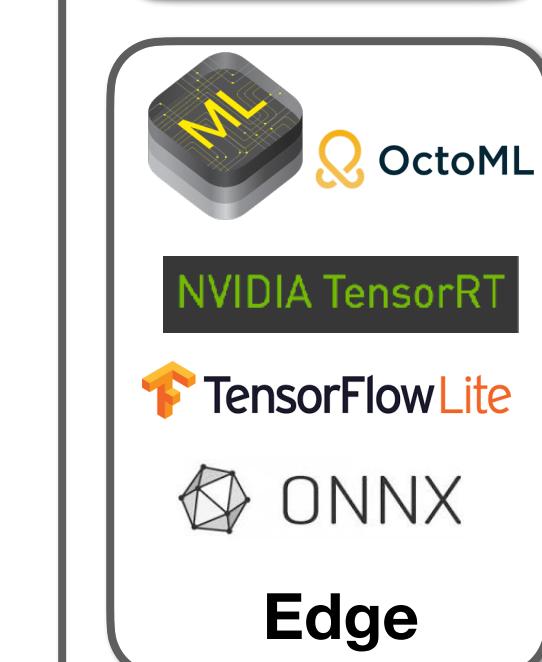
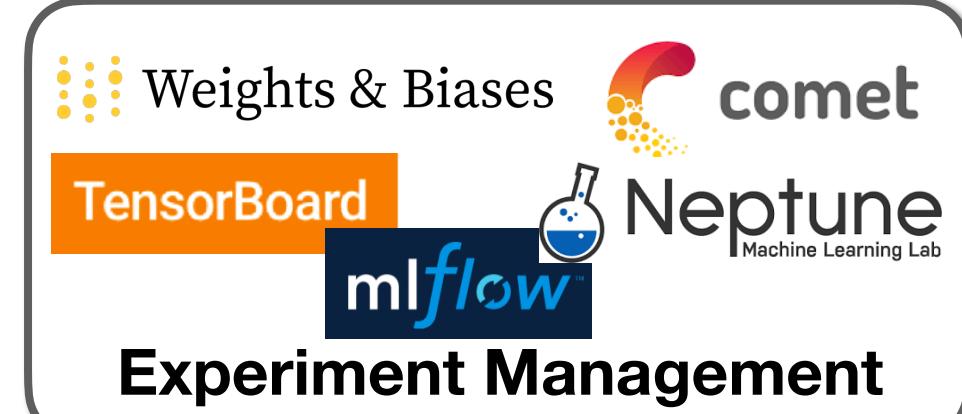
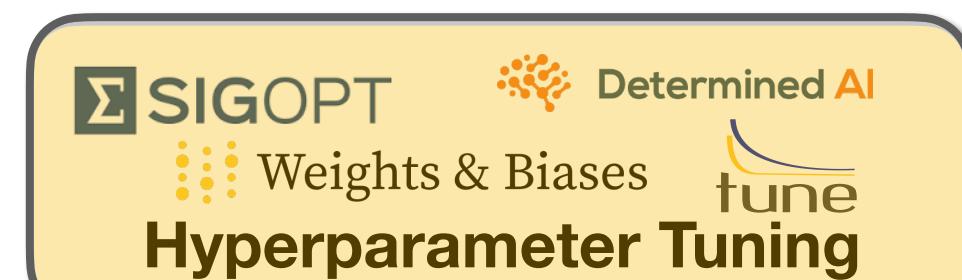
Data Lake / Warehouse



Data



Training/Evaluation



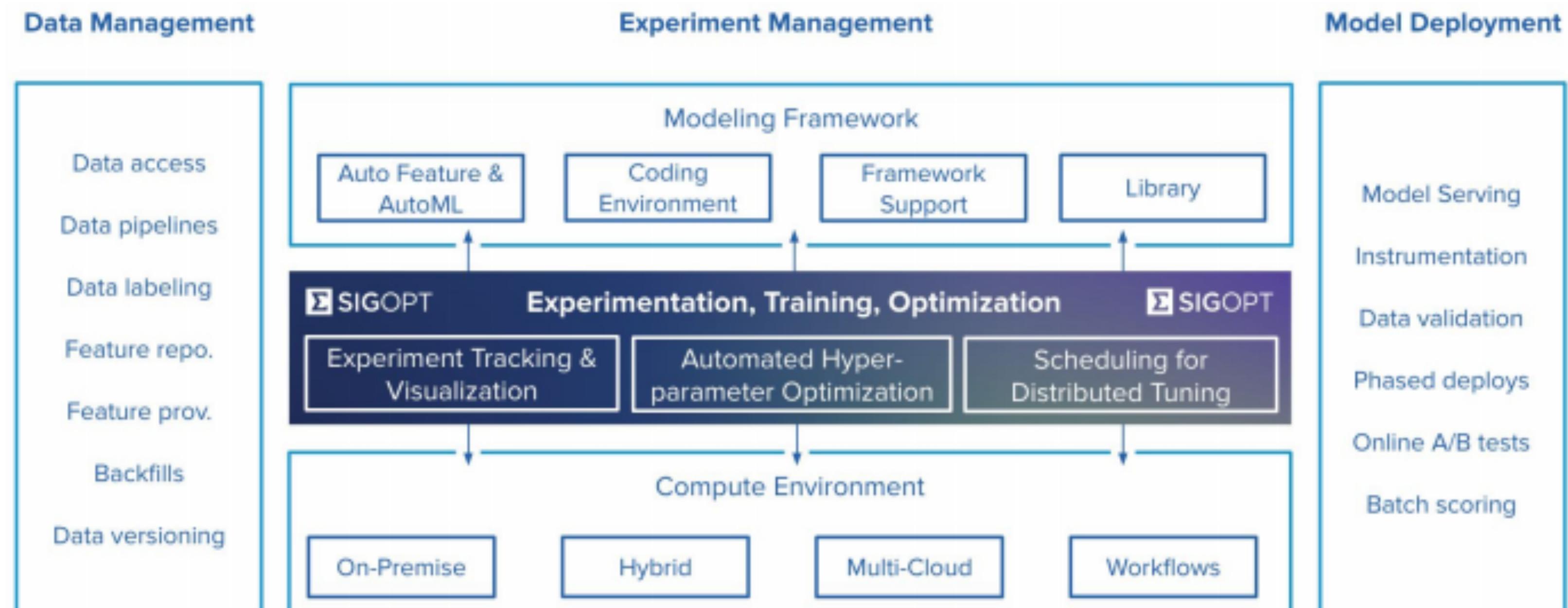
Deployment

Hyperparameter Optimization

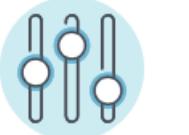
- Useful to have software that helps you search over hyper parameter settings.
- Could be as simple as being able to provide `--lr=(0.0001, 0.1) --num_layers=[128, 256, 512]` to training script
- Would be even better if settings were selected intelligently, and underperforming runs stopped early.

Improve ML models 100x faster

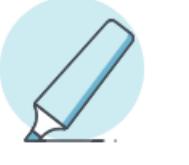
SigOpt's API tunes your model's parameters through *state-of-the-art* Bayesian optimization.



1 Provide parameters

 Ping our API with your model's parameters. We don't need the model itself. You keep it private.

2 Use our values

 Our API suggests new values for these parameters. Use them to evaluate your model within your current infrastructure.

3 Send model output

 We use your model's output to calculate the next best configuration.

4 Repeat until optimized

 You'll reach optimal values up to 100x faster than other methods.



Ray Tune

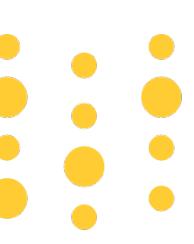
- "Choose among scalable SOTA algorithms such as Population Based Training (PBT), Vizier's Median Stopping Rule, HyperBand/ASHA."
- These redirect compute resources toward promising areas of search space

```
import torch.optim as optim
from ray import tune
from ray.tune.examples.mnist_pytorch import get_data_loaders, ConvNet, train, test

def train_mnist(config):
    train_loader, test_loader = get_data_loaders()
    model = ConvNet()
    optimizer = optim.SGD(model.parameters(), lr=config["lr"])
    for i in range(10):
        train(model, optimizer, train_loader)
        acc = test(model, test_loader)
        tune.track.log(mean_accuracy=acc)

analysis = tune.run(
    train_mnist, config={"lr": tune.grid_search([0.001, 0.01, 0.1])})
print("Best config: ", analysis.get_best_config(metric="mean_accuracy"))

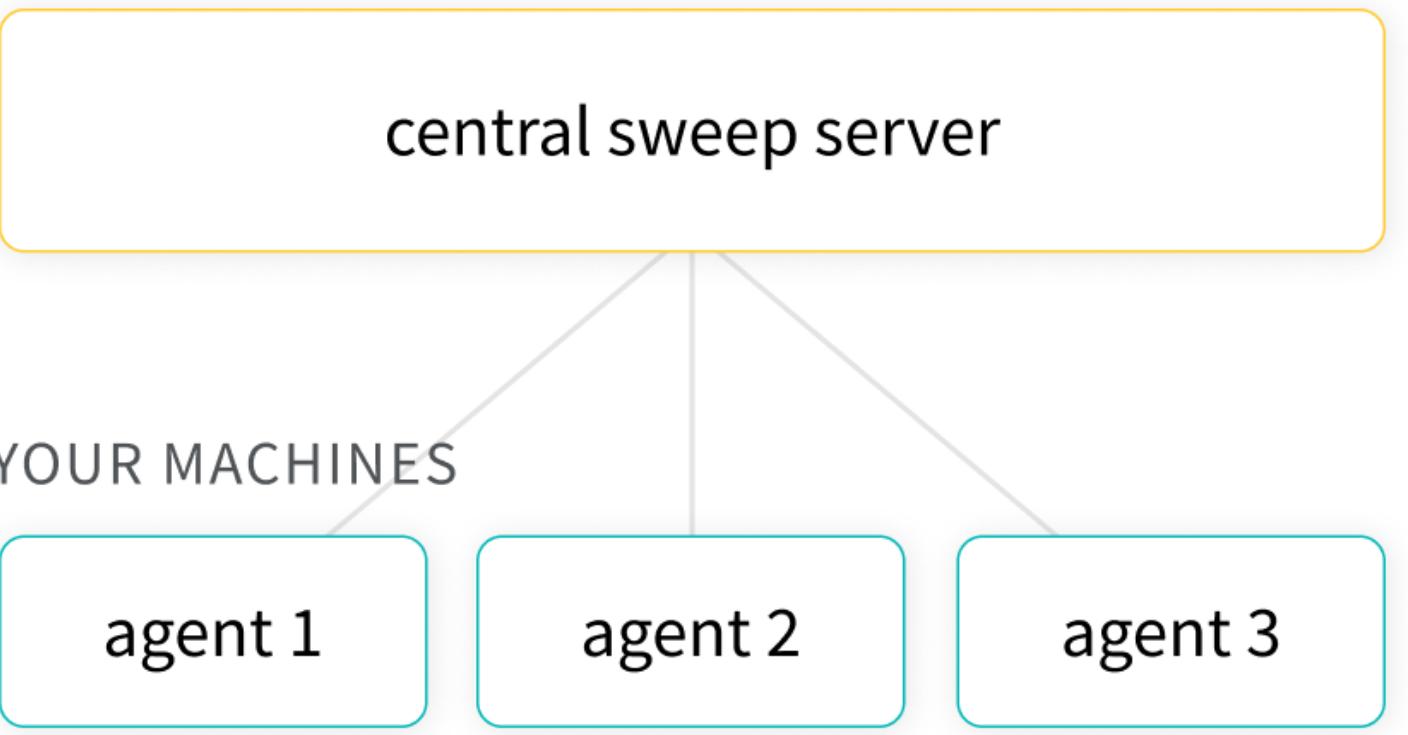
# Get a dataframe for analyzing trial results.
df = analysis.dataframe()
```



Weights & Biases

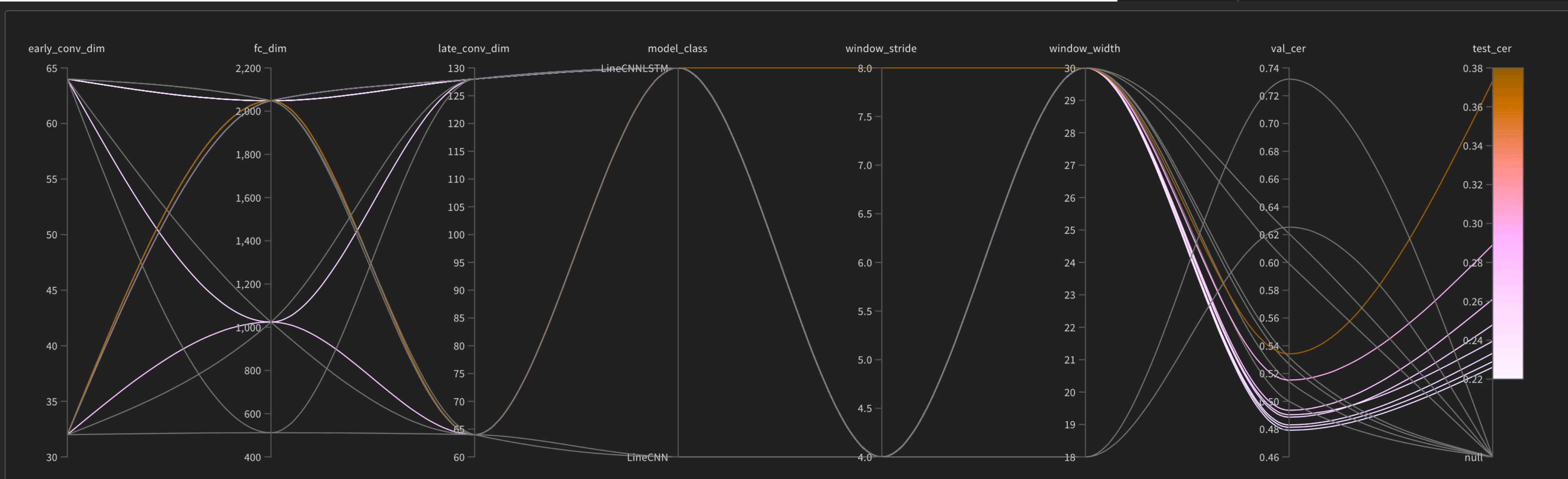
- What we use in Lab

OUR MACHINE



YOUR MACHINES

```
8 program: training/run_experiment.py
9 method: bayes
10 metric:
11   ·· goal: minimize
12   ·· name: val_cer
13 early_terminate:
14   ·· type: hyperband
15   ·· min_iter: 20
16 parameters:
17   ·· early_conv_dim:
18     ··· values: [32, 64]
19   ·· late_conv_dim:
20     ··· values: [64, 128]
21   ·· window_width:
```



Questions?



Amazon SageMaker

gradient^o
by Paperspace

FLOYD

DOMINO
DATA LAB

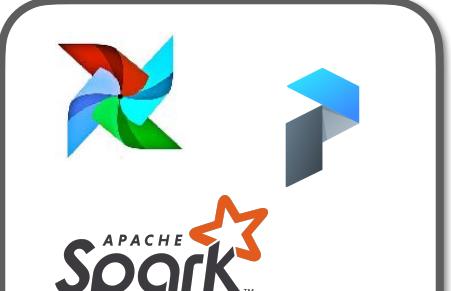
“All-in-one”



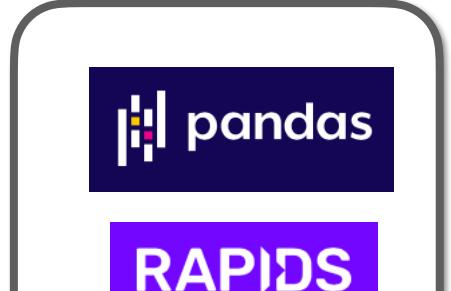
Versioning



Labeling



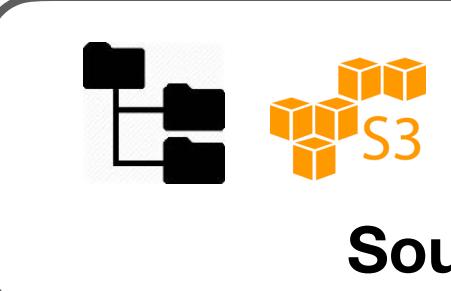
Processing



Exploration



Data Lake / Warehouse



Data



Frameworks & Distributed Training



Resource Management



Training/Evaluation



Experiment Management



Software Engineering



Feature Store



Edge



Monitoring



CI / Testing

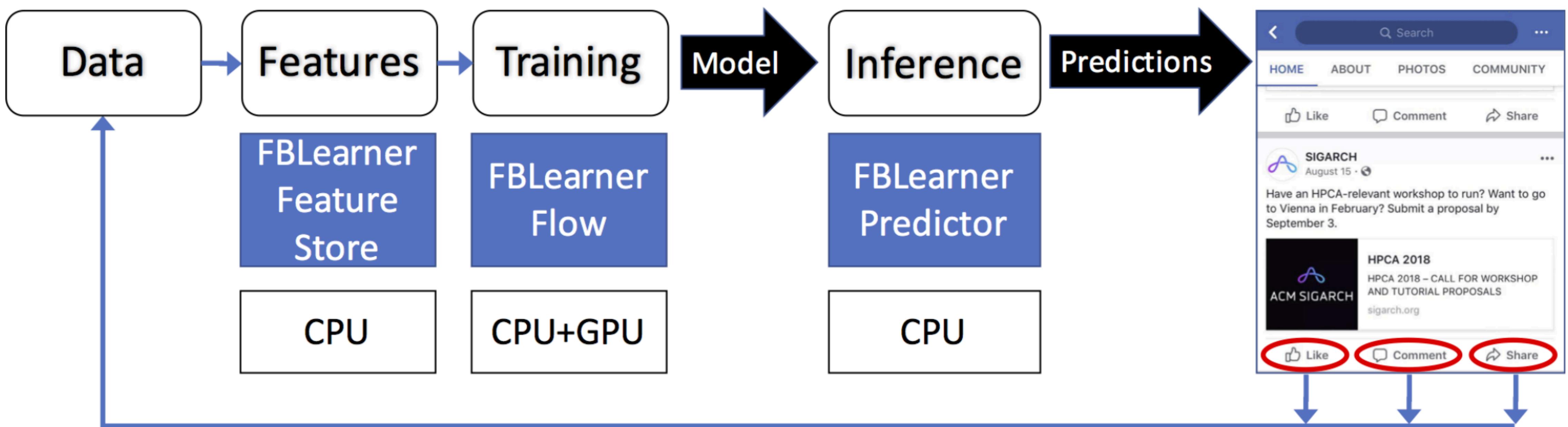


Deployment

All-in-one Solutions

- Single system for everything
 - development (hosted notebook)
 - scaling experiments to many machines (sometimes even provisioning)
 - tracking experiments and versioning models
 - deploying models
 - monitoring performance

Introducing FB Learner Flow: Facebook's AI backbone



**Prepare****Build****Validate****Deploy**

Data Labeling

BigQuery
datasets

Cloud Storage

Notebooks

AutoML
TrainingDeep Learning
VM ImageDeep Learning
Containers

AI Explanations

What-If Tool
Vizier

Prediction

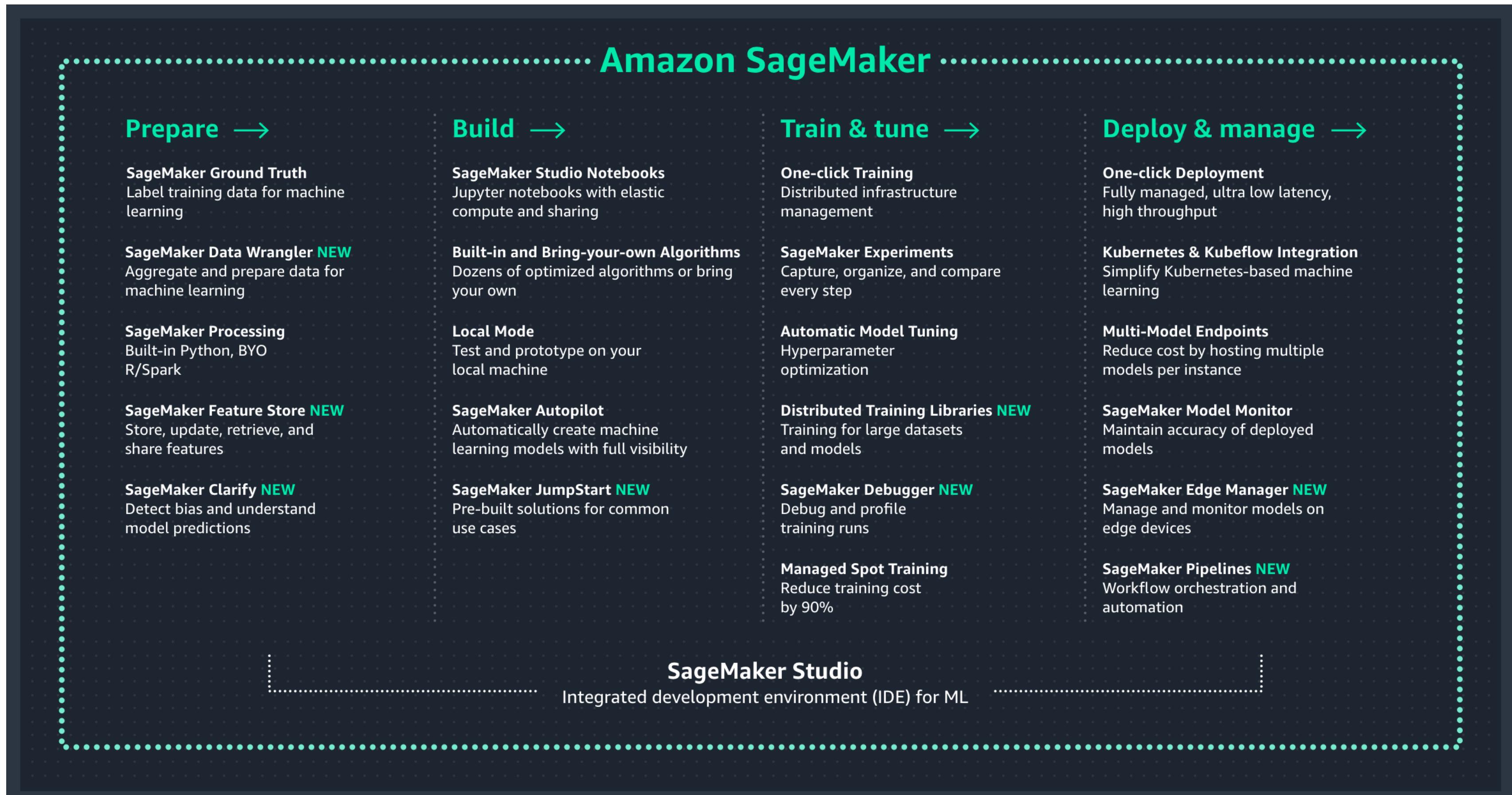
TensorFlow
Enterprise

Pipelines

Amazon SageMaker

Machine learning for every data scientist and developer

Amazon SageMaker helps data scientists and developers to prepare, build, train, and deploy high-quality machine learning (ML) models quickly by bringing together a broad set of capabilities purpose-built for ML.





Neptune

Machine Learning Lab



Build machine learning models

Run numerous experiments, easily compare them and find the best model.
Monitor the training process using charts instead of just console logs

[READ MORE](#)


Use your favorite tools

Write code in your preferred IDE and Jupyter Notebooks.
Neptune integrates with your favorite languages, libraries and frameworks.

[READ MORE](#)


Reproduce

Seamlessly track your experiments.
Never lose your work and always be able to reproduce your results.

[READ MORE](#)


Collaborate

Share your experiments and source code and compare different machine learning models with your teammates to achieve better results.

[READ MORE](#)

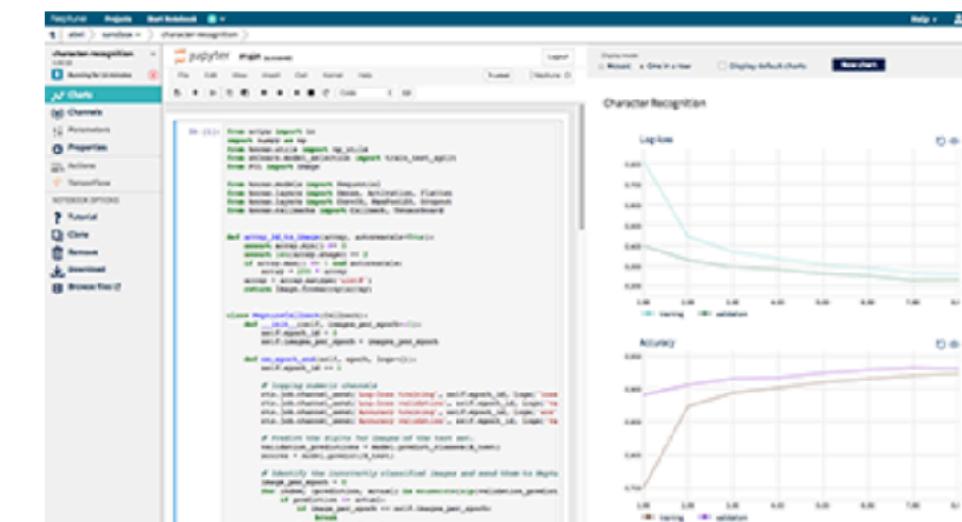

Run on your laptop or entirely in the cloud

Easily run experiments on your laptop or in the cloud - Neptune integrates with public clouds and manages compute environments (Docker).

[READ MORE](#)

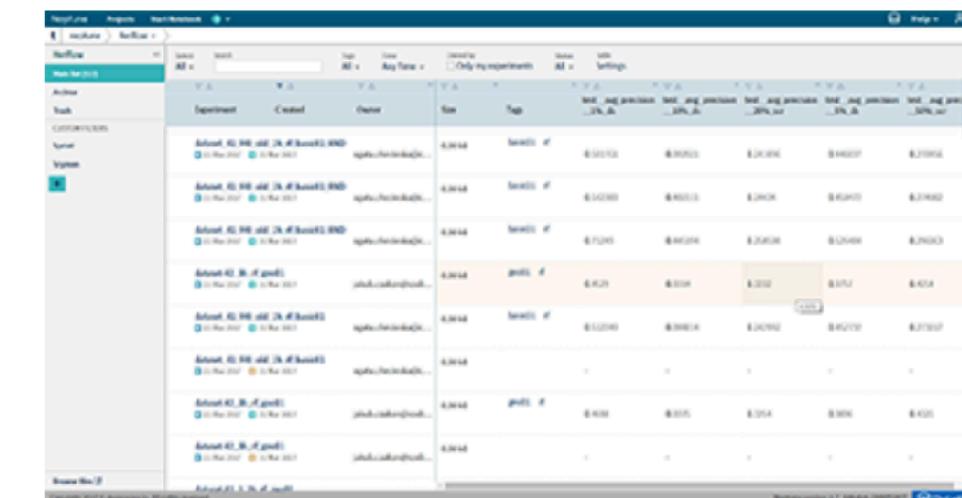
Use Jupyter Notebooks in the cloud

Prototype interactively using Jupyter Notebooks in the cloud. Neptune saves your code and outputs automatically.



Quickly find and compare your best experiments

Use your private “Kaggle leaderboard” like a dashboard to filter, explore and sort through your experiments. Find your top machine learning models based on your favorite metric.



Use pre-configured compute environments

Tired of manually configuring your remote machines or installing missing libraries?

Don't waste your time on DevOps work! Use one of Neptune's pre-configured environments (Docker images) and focus on data science instead.

```
$ neptune send --environment tensorflow
$ neptune send --environment keras
$ neptune send --environment theano
```



Send your training processes to the cloud

Move training processes off your desktop and onto powerful machines in the cloud. Experiment more at the same time.

```
$ neptune send train_cnn.py
```



FLOYD



Version Control and Full Reproducibility

All your work is saved and versioned automatically. Your code, environments, data, parameters and results are preserved for exact reproducibility.



Jupyter Notebook

Develop interactively using Jupyter Notebook from y

`floyd run --mode jupyter`

[Run your first GPU-powered notebook now →](#)



Deploy Trained Models

Preview Deploy your trained models as REST API wit

`floyd run --mode serve`

[Set up your model-serving API in one command →](#)



Training Metrics

Real-time training metrics for your jobs to help you optimize your code.



Zero Setup Deep Learning

Fully configured CPU and GPU instances primed for deep learning. Includes CUDA, cuDNN and all popular frameworks. Simply run:

`floyd run --env theano|`

[See all environments →](#)



Amazing Hardware

- **Amazing Hardware**

Multiple tiers of CPU and GPU instances, equipped with Nvidia Tesla K80 GPUs, Nvidia Tesla V100 GPUs, or better.

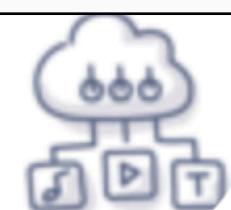
- **Dedicated Machines**

FloydHub machines are not spot instances. With guaranteed SLAs you will never lose your training results again.

- **Per Second Billing**

Forget about expensive hourly rates. Pay only for what you use, per second.

[View pricing →](#)



Iterate in Parallel

Evaluating multiple models? Parameter sweeping? Scale parallel on the cloud. Run concurrent jobs and FloydHub schedules, manages and tracks them all for you.

DEVELOP, TRAIN, & DEPLOY

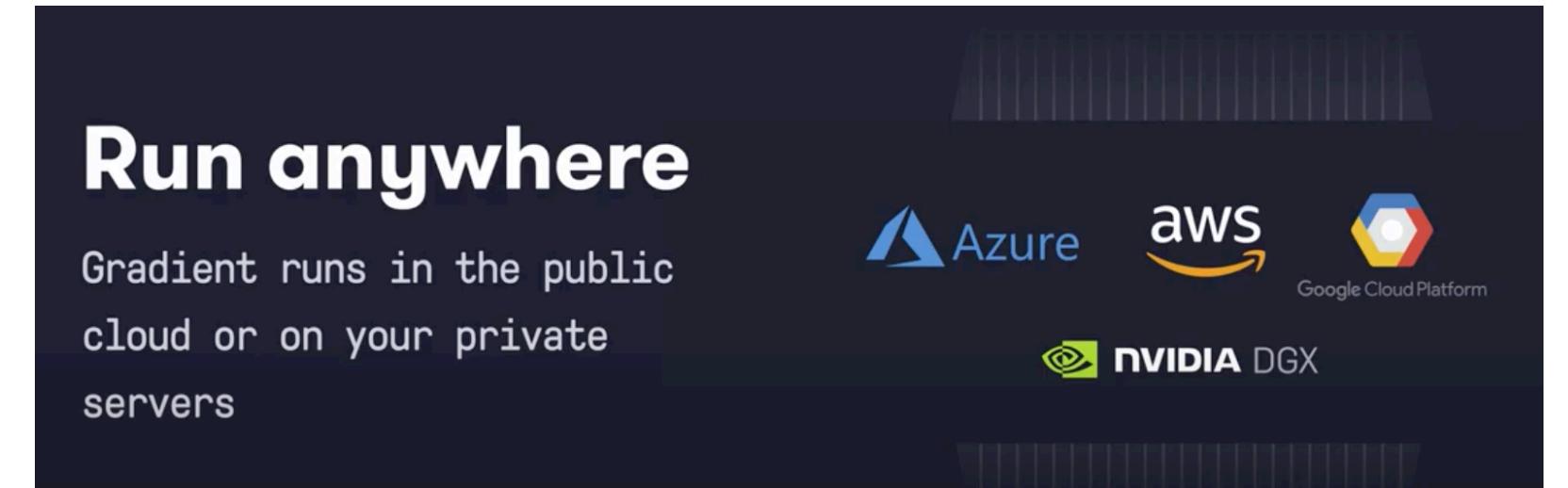


Notebooks

1-Click Jupyter Notebooks

Run anywhere

Gradient runs in the public cloud or on your private servers




Experiments

Develop, train, and track results



Datasets

Connect, version, and track data



Models

Store, analyze and version models

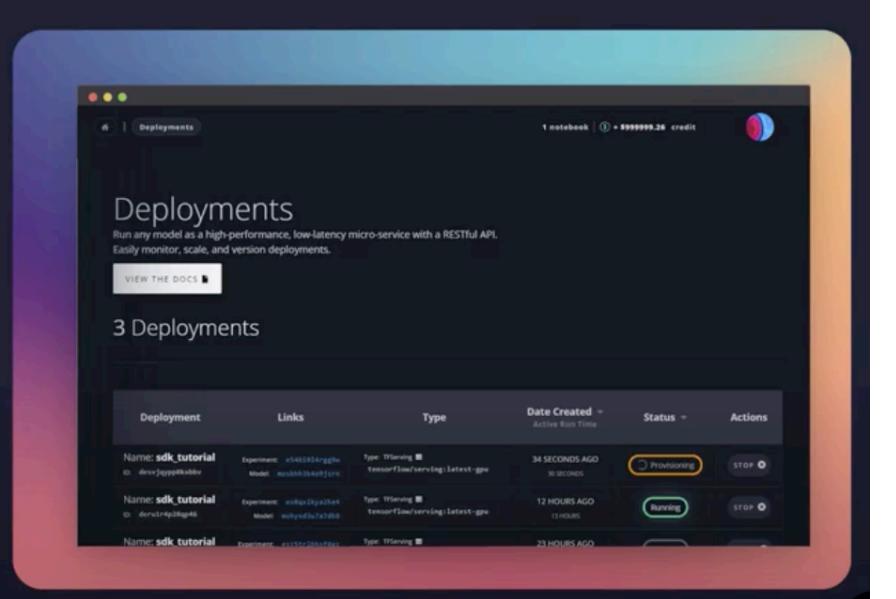


Inference

Deploy models as API endpoints

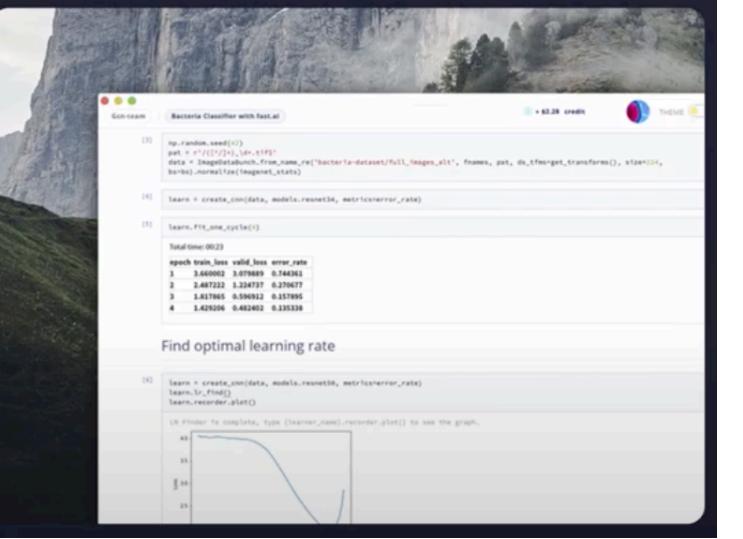
03 Deploy

Model Serving with scalable API endpoints



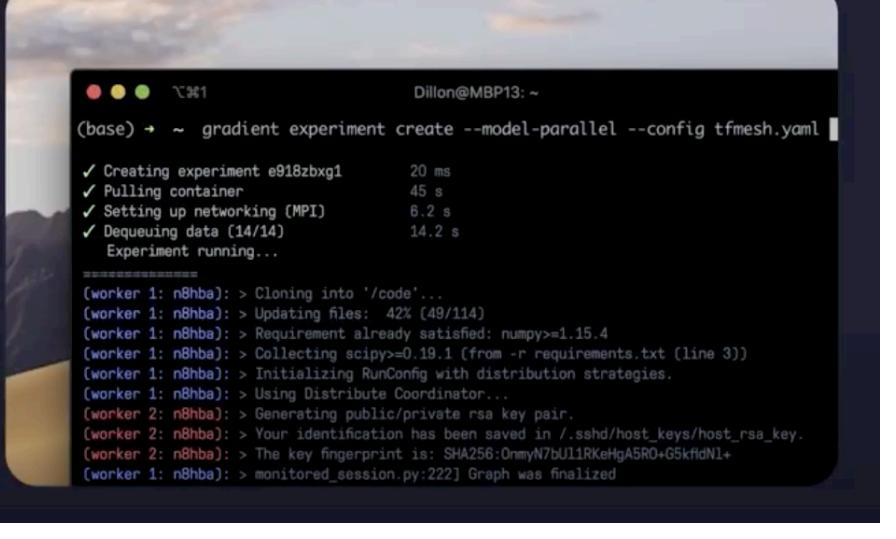
01 Develop

1-click Jupyter Notebooks with free GPUs and zero management



02 Train

Run single experiments to large-scale distributed training



01 Interface

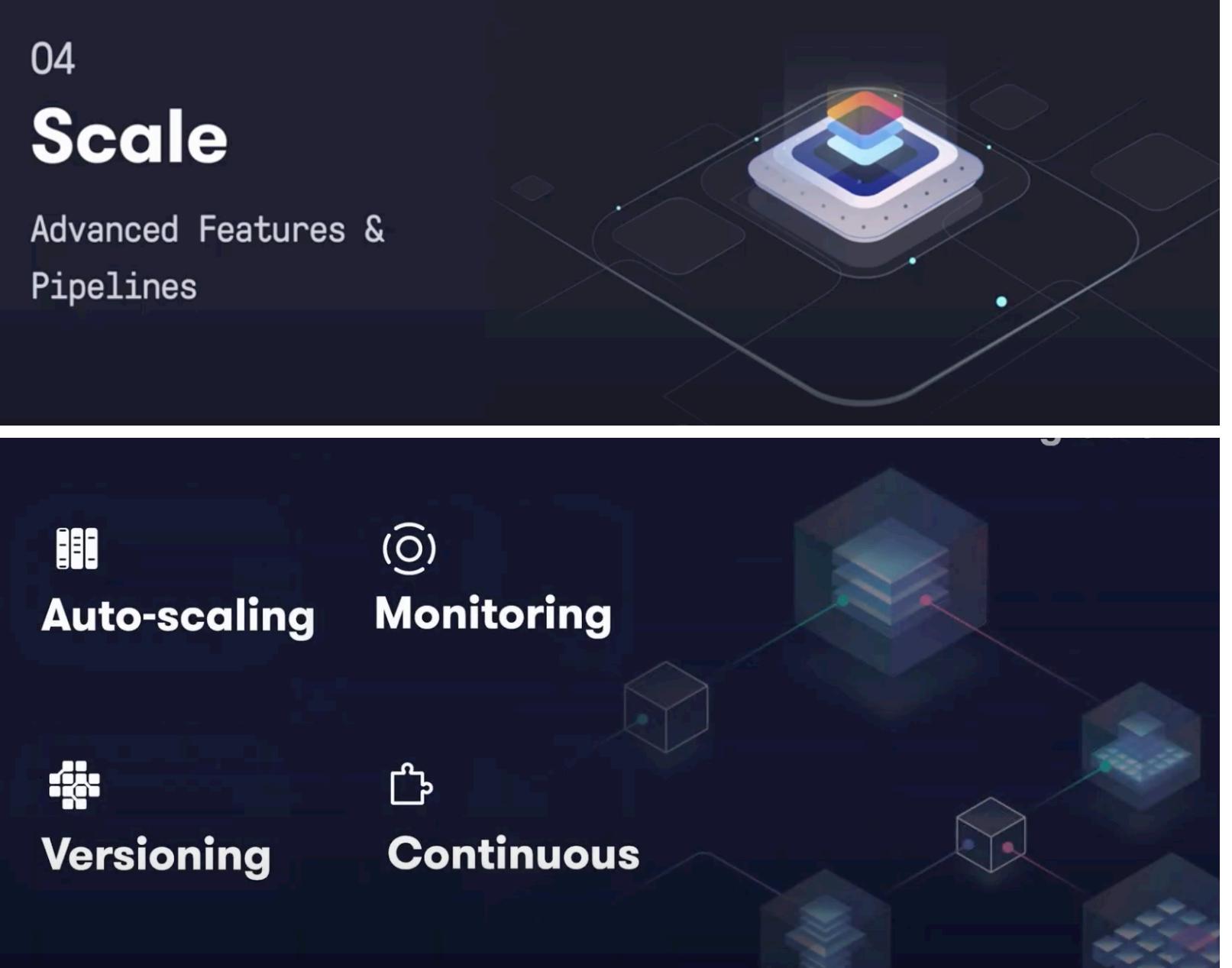
02 CLI

03 GitHub



04 Scale

Advanced Features & Pipelines





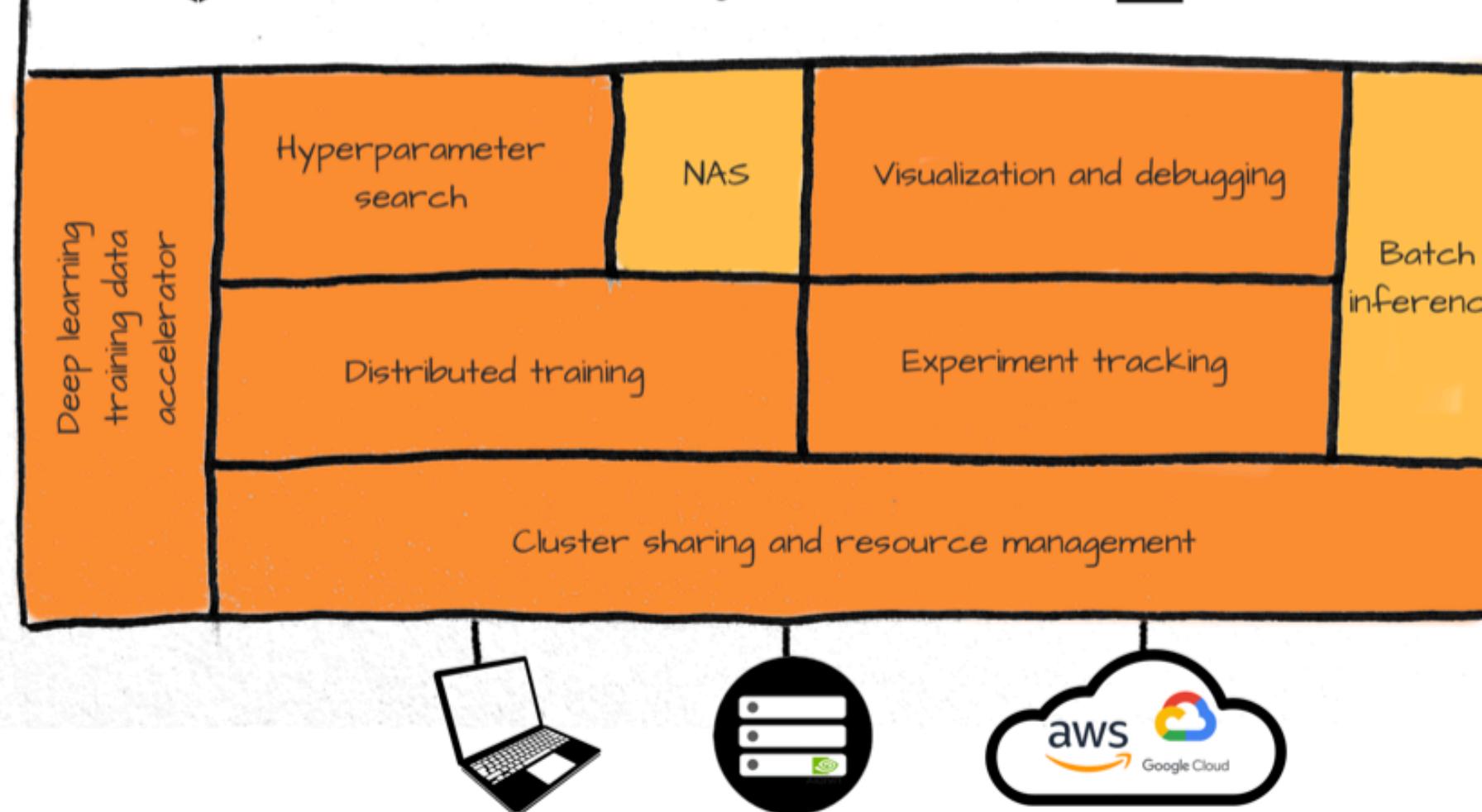
Determined AI

Data Prep

Data Storage
and ETL



aws | s3

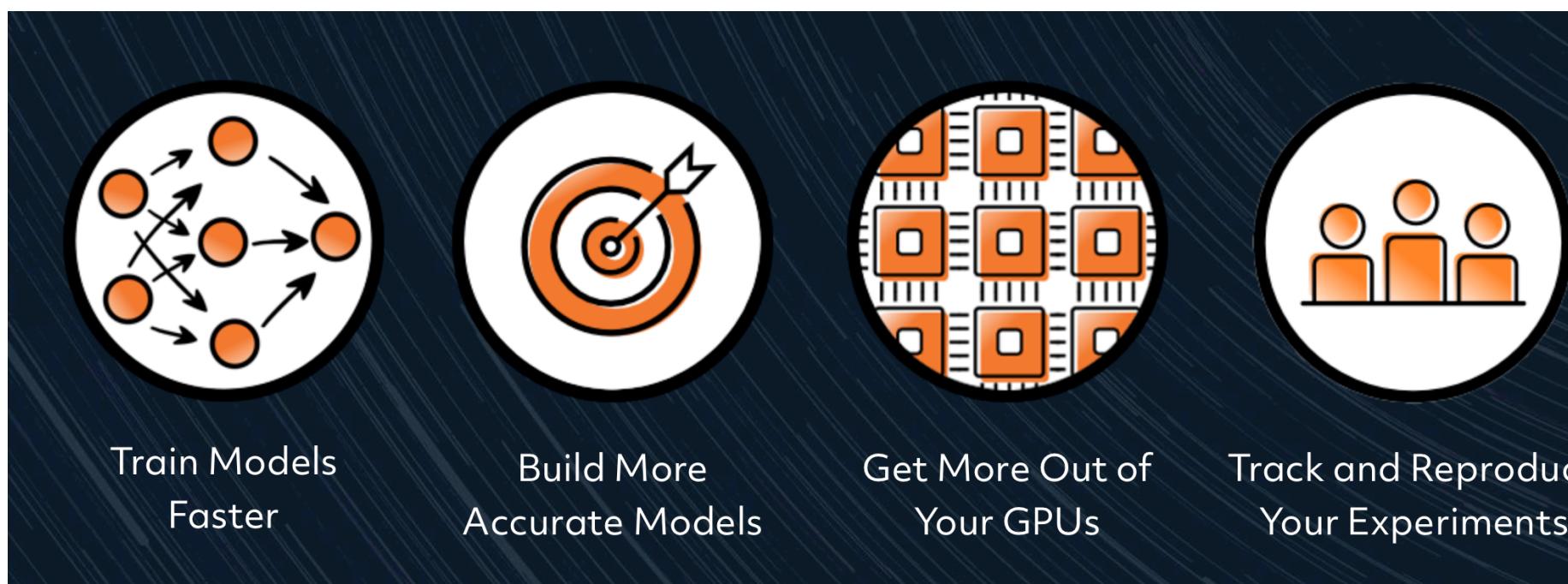


Model Deployment

Web services
and apps



- Open source!
- In-house distributed training module
- HyperBand based hyperparameter tuning
- Smart GPU scheduling, including spot instances
- Experiment tracking



Domino Data Lab

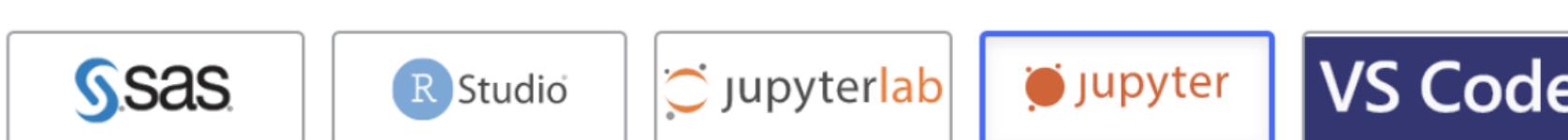
One place for your data science tools, apps, results, models, and knowledge.

Provision compute

Workspaces

Choose from the options below to launch a new workspace. Don't see what you need? [Learn more about Workspaces.](#)

Launch a workspace



Workspace Name

scratch EDA work

Launch Jupyter (Python, R, Julia) Workspace

All Active Completed

Hardware Tier

Default	< 1 MIN
GPU (4 V100s) 32 cores · 488 GB RAM · \$0.12/min	< 1 MIN
Large 16 cores · 122 GB RAM · \$0.0177/min	< 1 MIN
Medium 8 cores · 32 GB RAM · \$0.0064/min	< 1 MIN
Small 4 cores · 16 GB RAM · \$0.0033/min	< 1 MIN
X1 32x Large 128 cores · 1952 GB RAM · \$0.0022/min	< 1 MIN

Track experiments



Domino Data Lab

Deploy REST API

Calling your Model

Route: Latest

Model URL: https://your_host:443/models/5cab7be8c9e77c000762c223/latest/model

Tester curl Java JavaScript PHP PowerShell Python R Ruby Other

Request

```
{ "data": { "dropper": 0.08, "mins": 120, "consecmonths": 24, "income": 40, "age": 35 } }
```

Response | Instance Logs

```
model_id: 5cab7be8c9e77c000762c223, release: { harness_version: "0.1", model_version: "5d013eabc9e77c0007d4502e", model_version_number: 8 }, request_id: "QLEI7NWVMT2JG6AS", result: [ 0.9828162575212186 ],
```

Send

Window Size NA/6 predictions

since last Scheduled Test by Time by Data

Till Date Y M W D H MI S

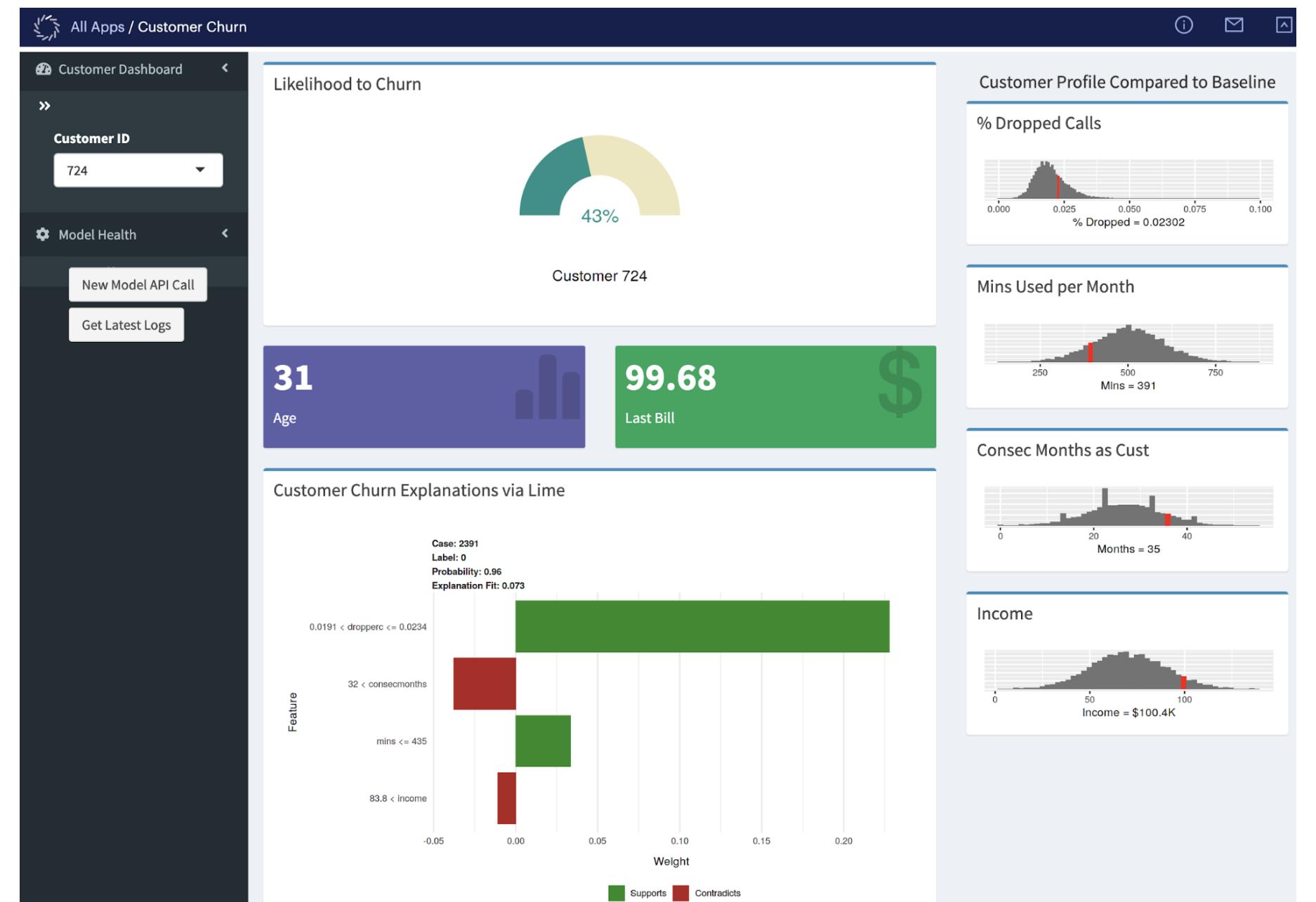
Model Drift

Search

STATUS	FEATURE	TRAINING DATA	PREDICTION DATA	TEST TYPE	DISTRIBUTION CHANGE	TEST RULE
●	petal.length Feature			Kulback-Leibler Divergence	0.2948	Greater than 0.3
●	sepal.length Feature			Kulback-Leibler Divergence	0.3744	Greater than 0.3
●	petal.width Feature			Kulback-Leibler Divergence	0.1943	Greater than 0.3
●	sepal.width Feature			Kulback-Leibler Divergence	0.3029	Greater than 0.3
●	variety Prediction			Kulback-Leibler Divergence	0.1262	Greater than 0.3

Monitor predictions

Publish applets

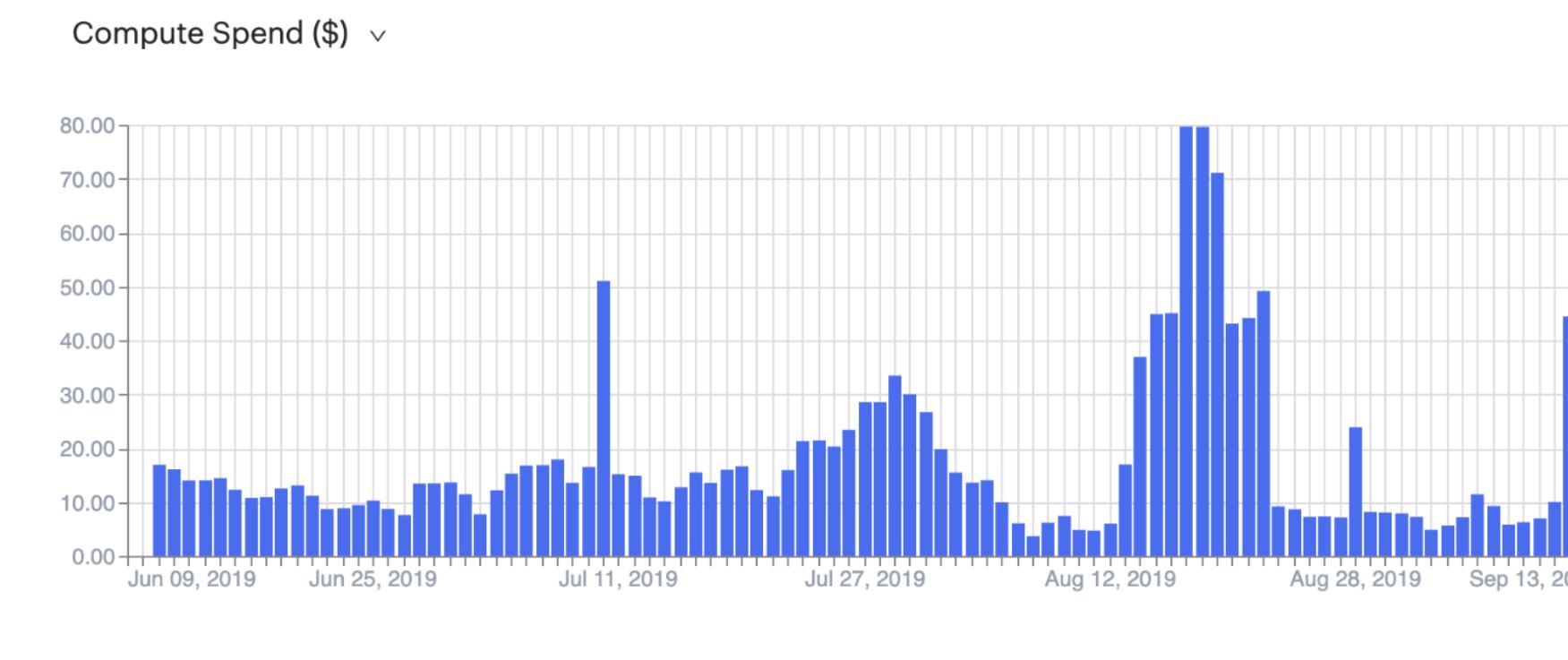


Domino Data Lab

Monitor spend

Overview

Jun 11–Sep 11
2019



Users

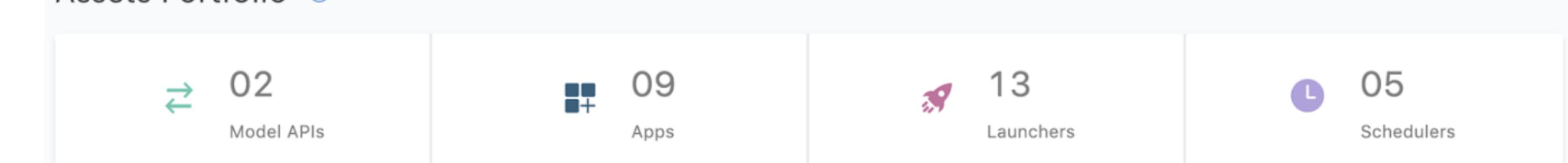
By Compute Spend (\$)

[View All >](#)

User	Compute Spend (\$)
Colin Goyette	\$549.63
Graham Whitelaw	\$191.62
John Conway	\$178.38

All projects in one place

Assets Portfolio



100 entries Search

Asset Name	Type	Project	Last Updated	Versions	Owner	Usage	Last 24H	Dev. Cost (\$)	Prod. Cost (\$)	Stakeholder
Audience Selection A	Model API	Targeted_Mkt	11 days ago	4	Dan S		10 views	1k	5k	Rob S
Auto MLPD	Model API	Character Rec.	7 days ago	4	Mohith G		40 views	1.3k	4.7k	Tim H
Demand forecasting	Model API	Test DL for f...	8 days ago	4	Dan S		22 views	1.4k	4.2k	Rob S
Improve location API	Model API	Improve locat...	9 days ago	4	Stev D		900 calls	1.8k	3.8k	Tim H
Lead Scoring Explainer	Model API	Fraud - Q2 F...	18 days ago	4	Dan S		42 views	3k	4.2k	Nick E

Natural place to go for most MLOps companies

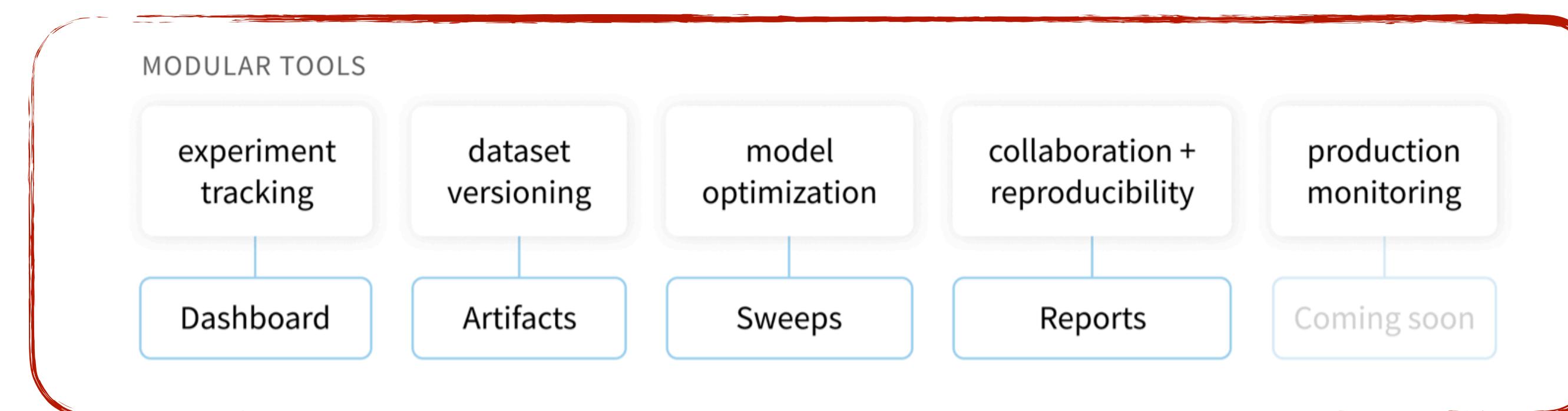
Weights & Biases

Track machine learning experiments, visualize metrics, and share results

Weights & Biases helps you keep track of your machine learning projects. Use our tools to log hyperparameters and output metrics from your runs, then visualize and compare results and quickly share findings with your colleagues.

Here is a quick overview of our lightweight, modular tools:

1. **Dashboard**: Track experiments, visualize results
2. **Reports**: Save and share reproducible findings
3. **Artifacts**: Dataset and model versioning, pipeline tracking
4. **Sweeps**: Optimize models with hyperparameter tuning



	W&B	Paperspace Gradient	Floyd	Determined.ai	Domino Data Lab	Amazon SageMaker	GC ML Engine
Hardware	N/A	Paperspace	GCP	Agnostic	Agnostic	AWS	GCP
Resource Management	No	Yes	Yes	Yes	Yes	Yes	Yes
Hyperparam Optimization	Yes	No	No	Yes	Yes	Yes	Yes
Storing Models	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Reviewing Experiments	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Deploying Models as REST API	No	No	Yes	No	Yes	Yes	Yes
Monitoring	No	No	No	No	Yes	Yes	Yes

Questions?

Tooling Tuesdays



Full Stack Deep Learning
@full_stack_dl Follows you

Our mission is to help you go from a promising ML experiment to a shipped product, with real-world impact.

Co-organized by [@sergeykarayev](#) and [@josh_tobin_](#).

Joined January 2019

108 Following 7,048 Followers

Followed by Katimus Prime, Charlie You, and 93 others you follow

https://twitter.com/full_stack_dl



Full Stack Deep Learning
@full_stack_dl

Tooling Tuesdays: Thread of Threads

Every week, we share a useful tool for full stack machine learning. Follow along, and please share your suggestions!

1/N

11:07 AM · Dec 29, 2020 · Twitter Web App

Thank you!