



FSDL 2022

Development Infrastructure & Tooling

Sergey Karayev

AUGUST 15, 2022



Dream

Provide project spec
and some sample data

Get a continually improving prediction
system, deployed at scale

Reality

Aggregate, process, clean, label,
and version data

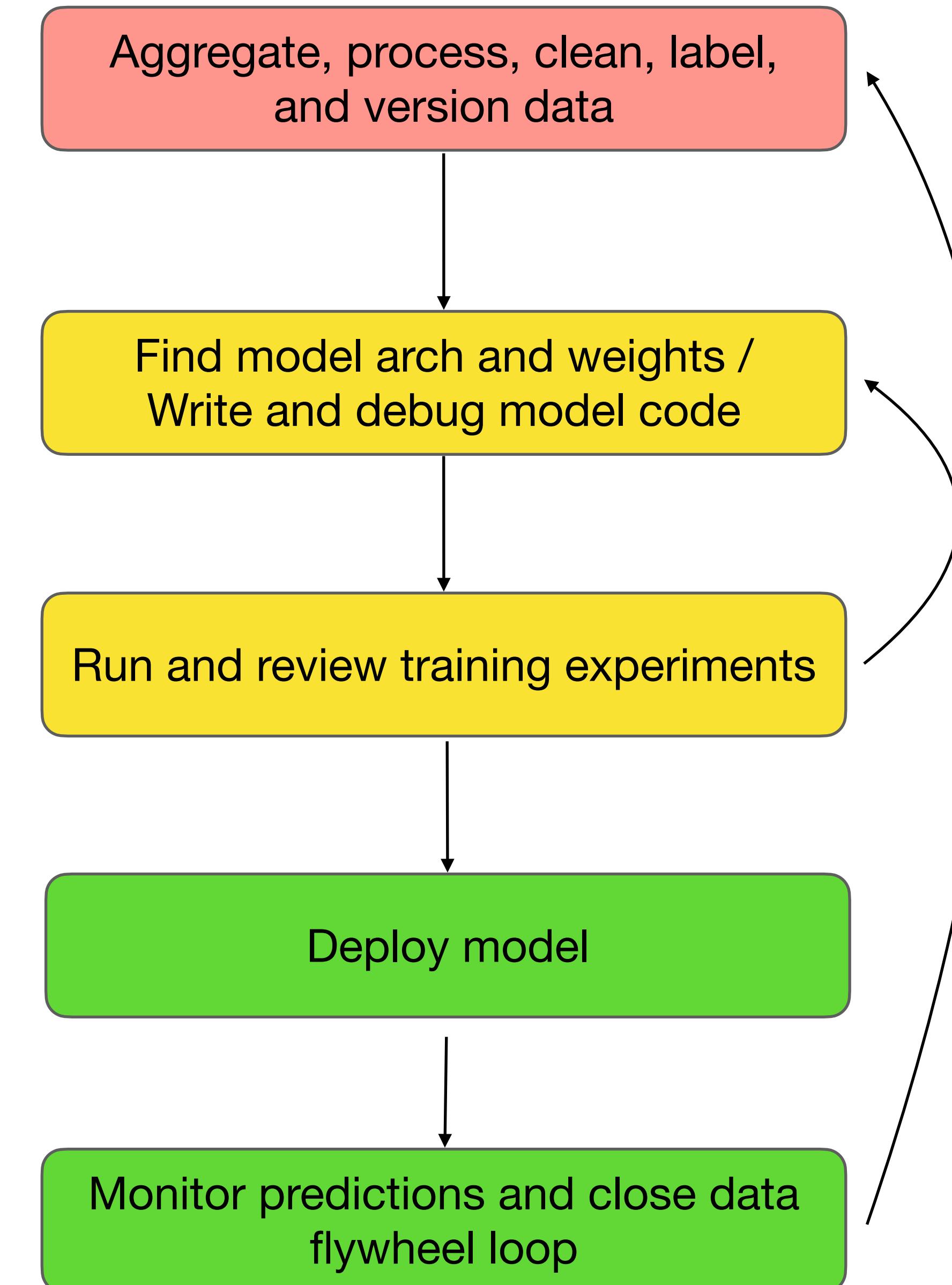
Find model arch and weights /
Write and debug model code

Run and review training experiments

Deploy model

Monitor predictions and close data
flywheel loop



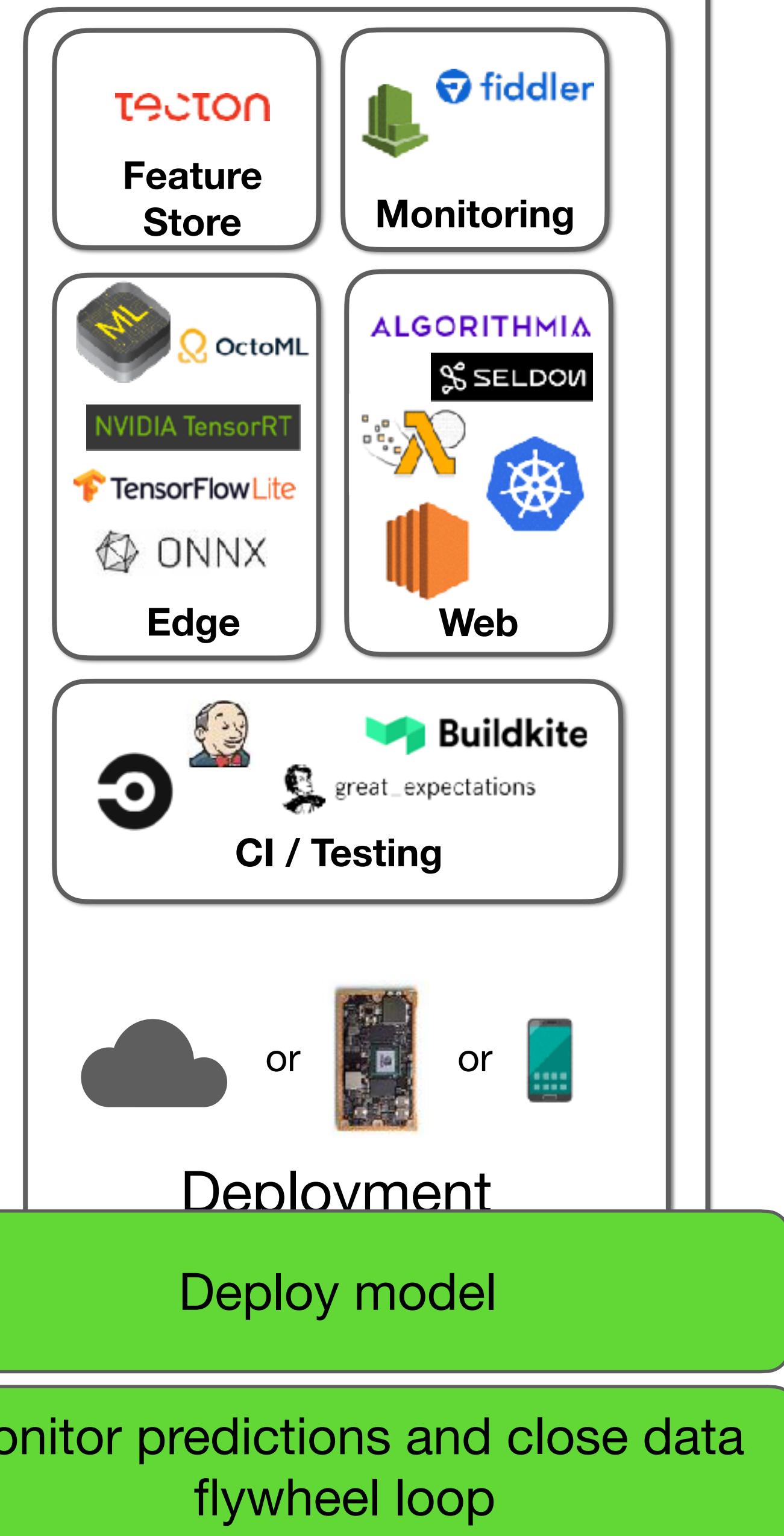
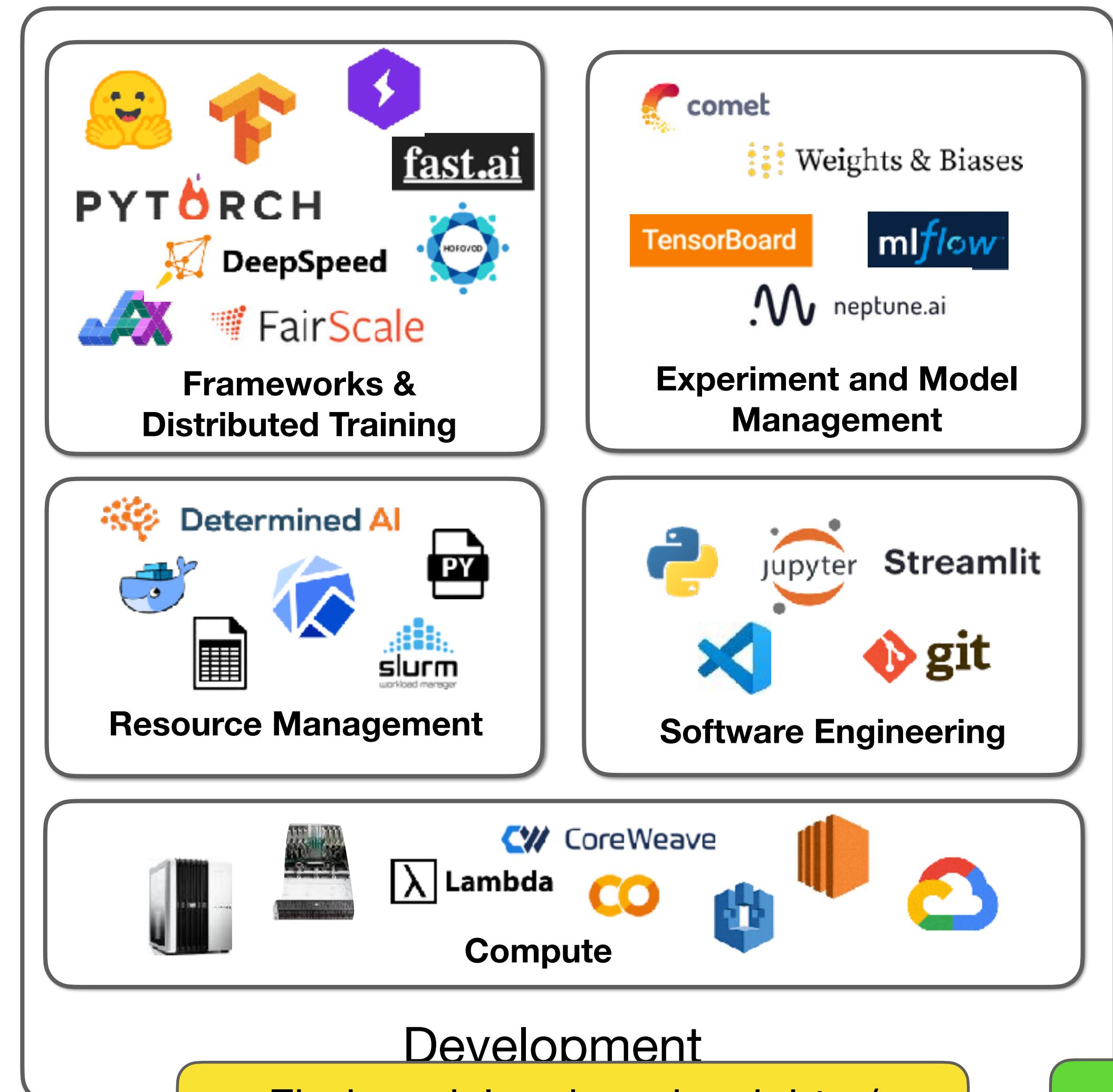
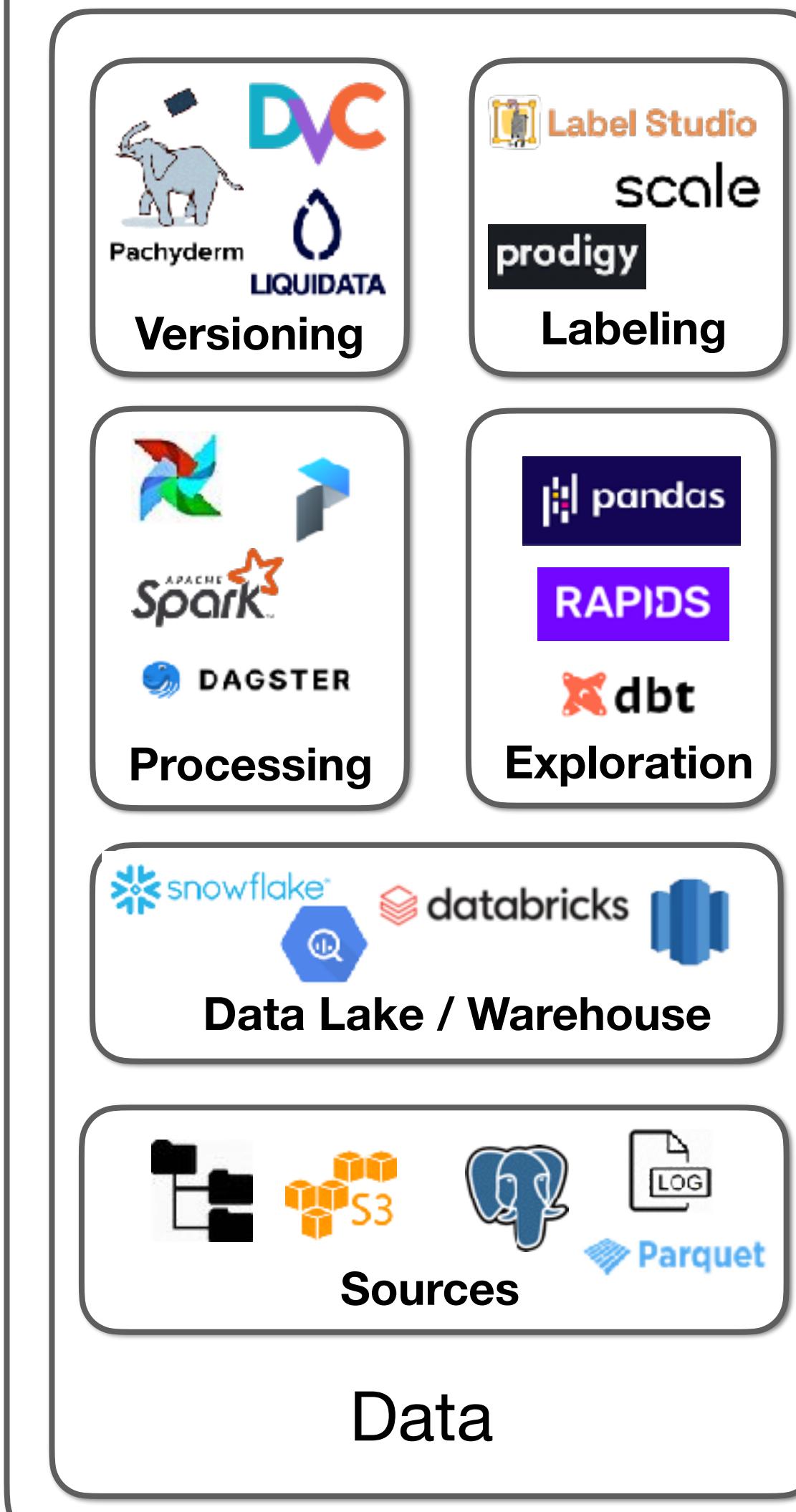


"All-in-one"



Amazon SageMaker
by Paperspace

gradient^o
by DOMINO
DATA LAB



Aggregate, process, clean, label, and version data

Find model arch and weights / Write and debug model code
Run and review training experiments

Deploy model
Monitor predictions and close data flywheel loop

“All-in-one”



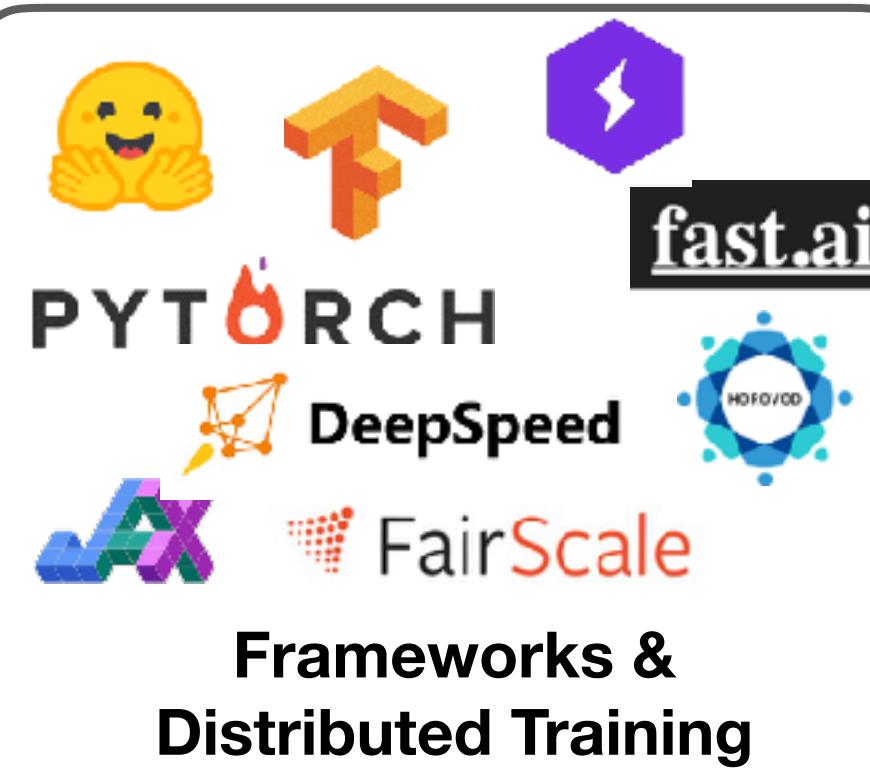
Amazon SageMaker

gradient^o
by Paperspace

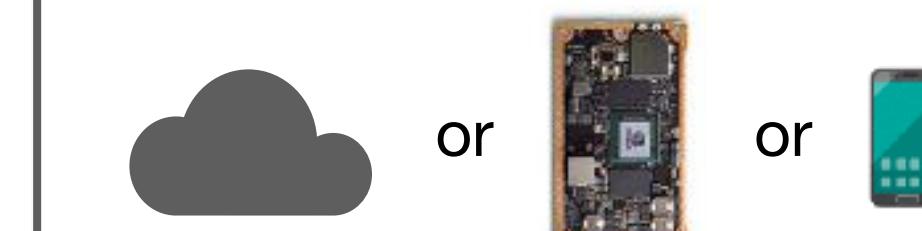
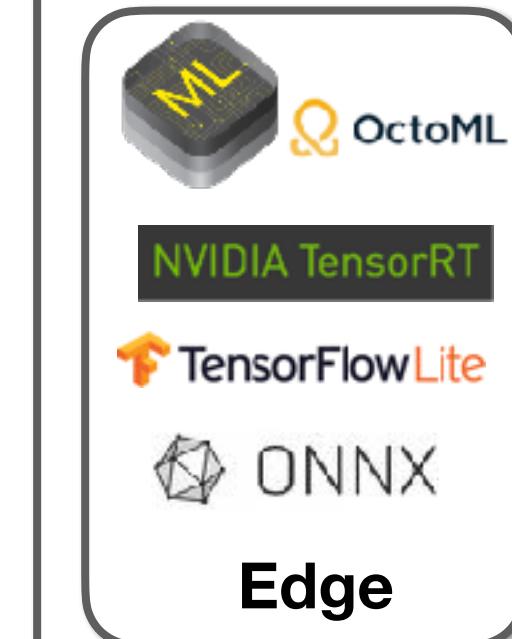
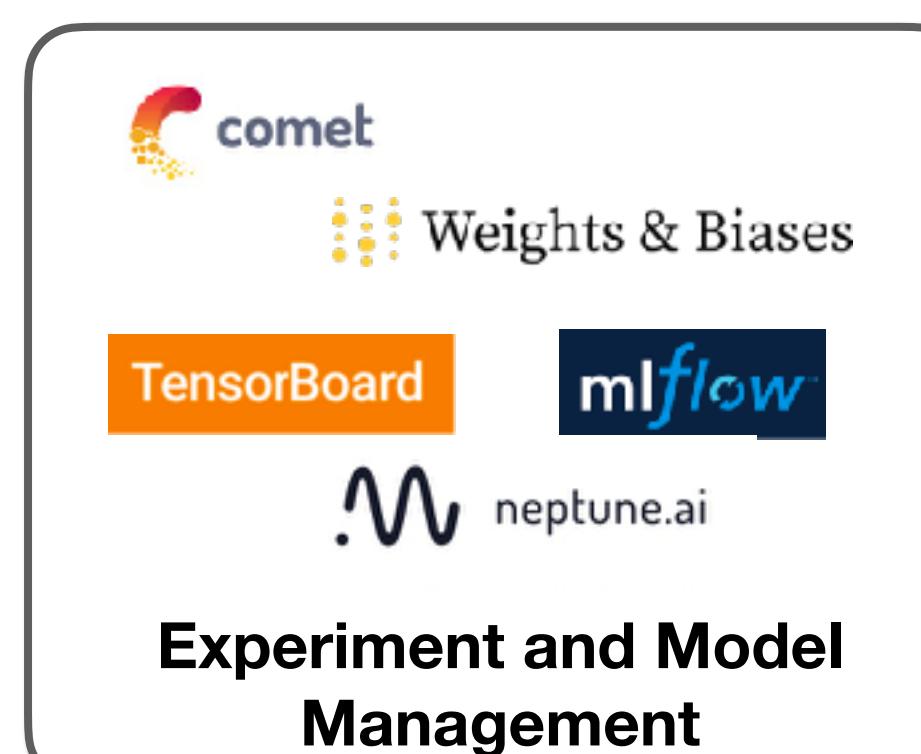
DOMINO
DATA LAB



Data



Development



Deployment

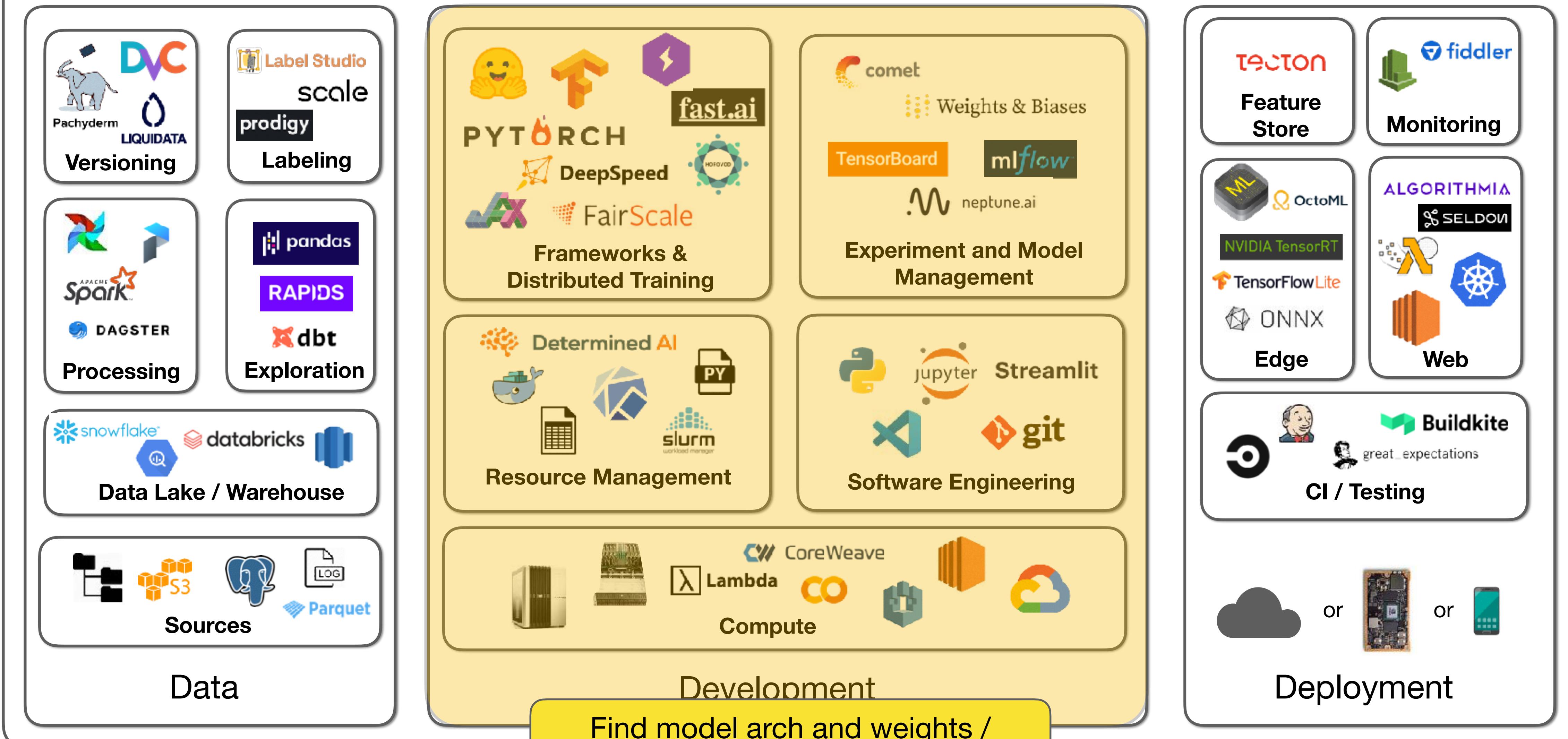


"All-in-one"



Amazon SageMaker
by Paperspace

gradient^o
by DOMINO
DATA LAB



Find model arch and weights /
Write and debug model code

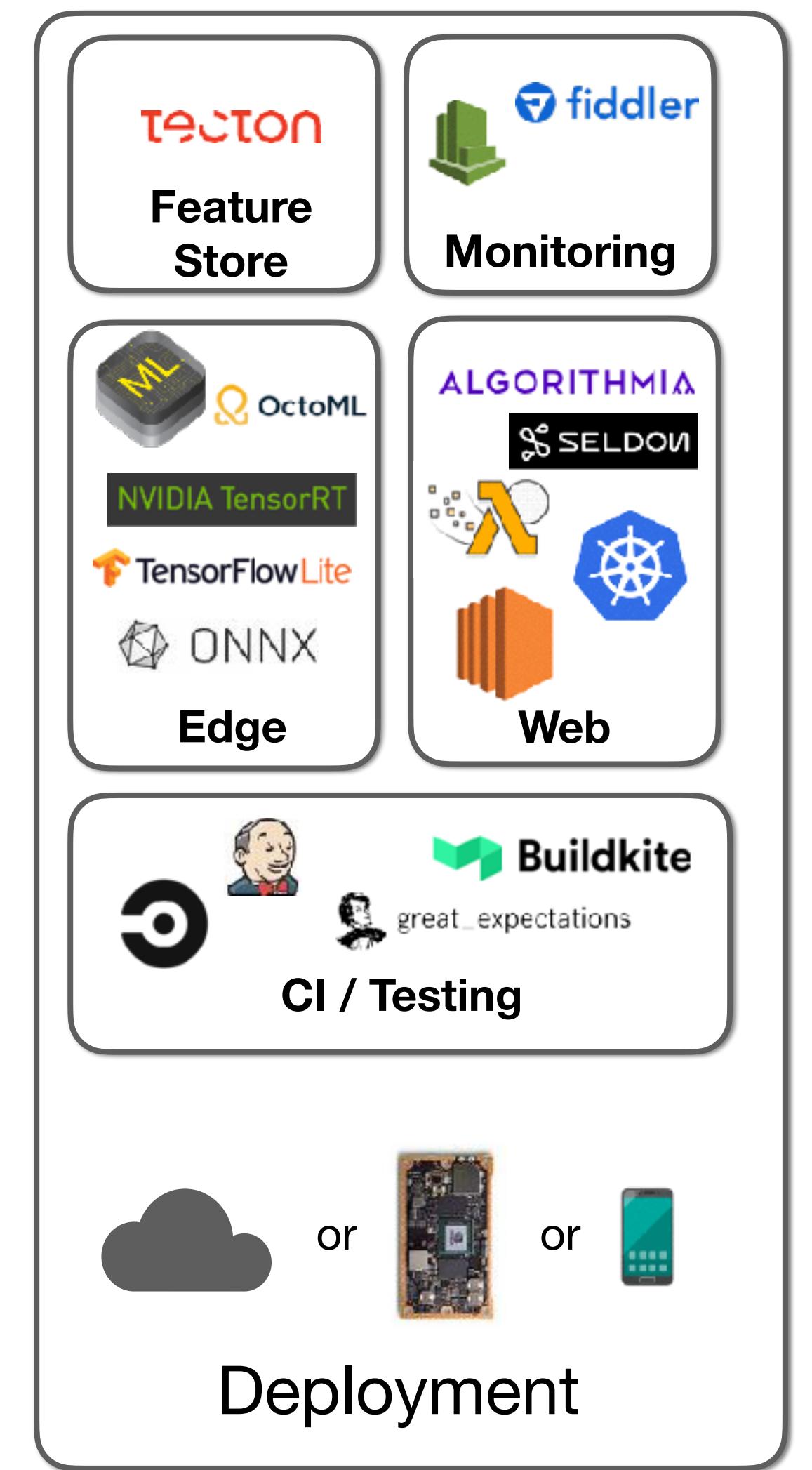
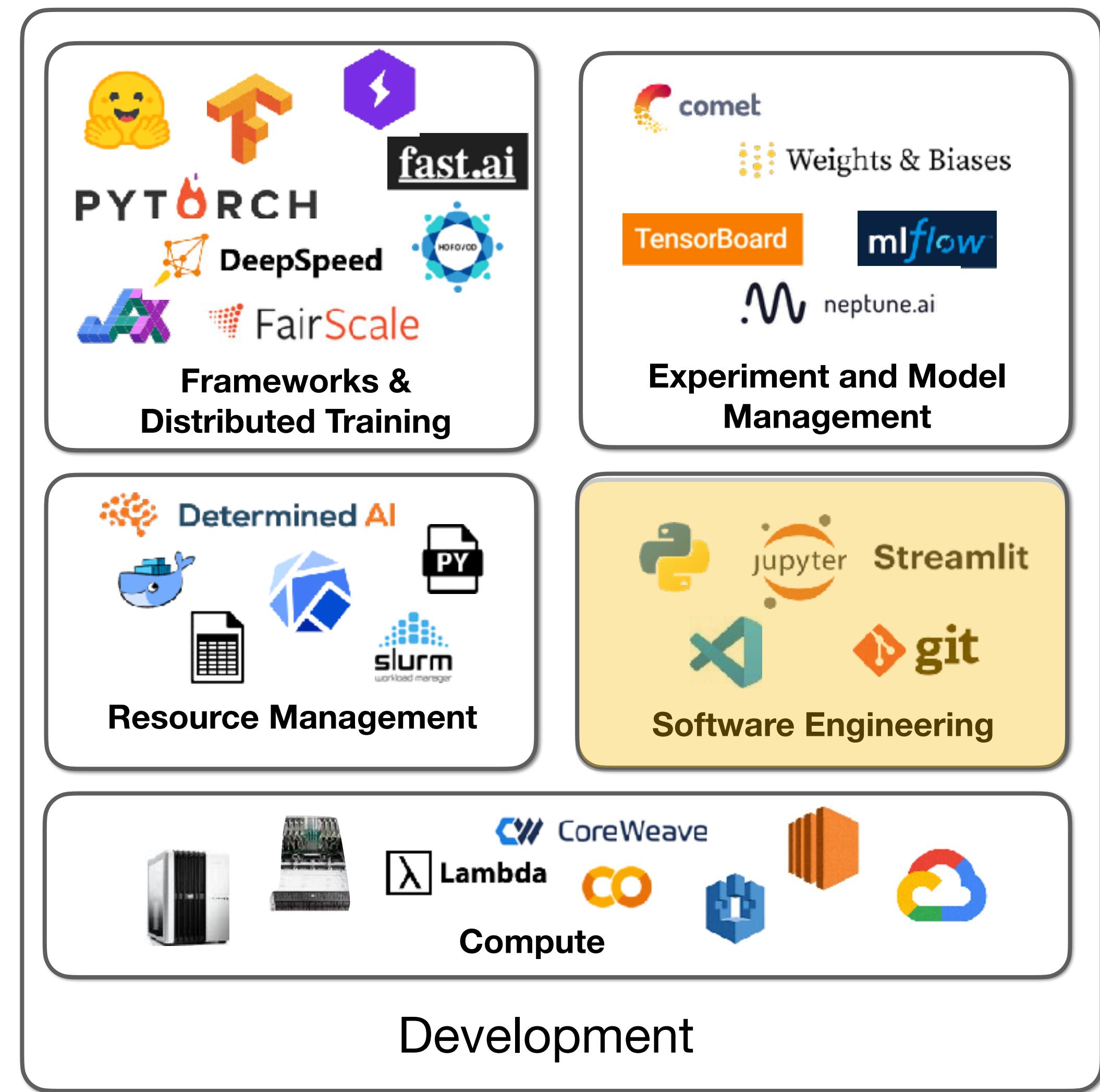
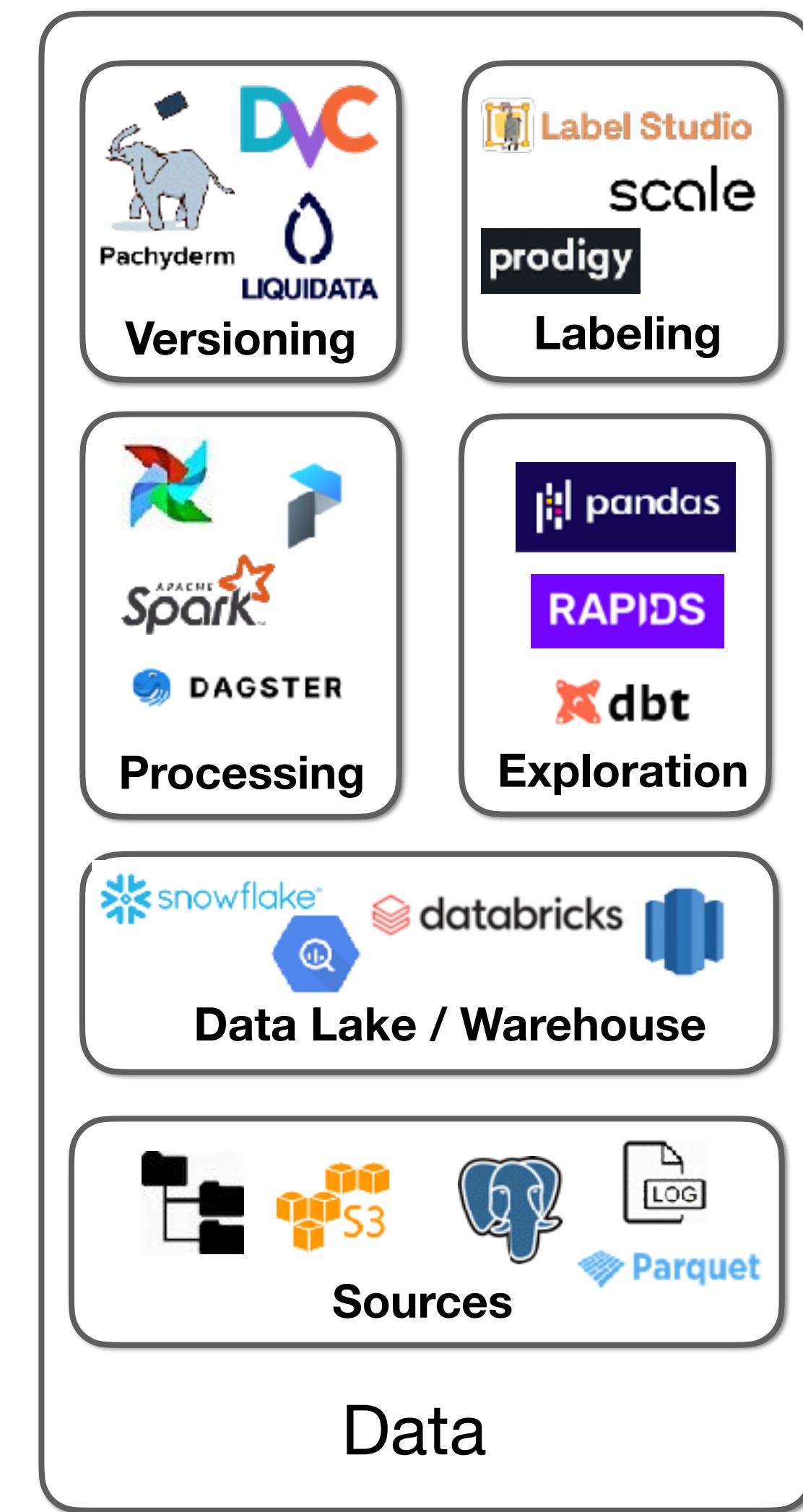
Run and review training experiments

“All-in-one”



Amazon SageMaker
by Paperspace

gradient^o
by DOMINO
DATA LAB





Software Engineering



Programming Language

- Python, because of the libraries
 - Clear winner in scientific and data computing
- Contenders:
 - Julia (**Jupyter!**)
 - C/C++



Editors

Vim

Jupyter

Emacs

VS Code

PyCharm



VS Code

Built-in git staging and diffing →

Peek documentation

Lint code as you write

Open whole projects remotely

The screenshot shows the VS Code interface with several annotations:

- A red circle highlights the "Staged Changes" icon in the Explorer bar.
- A red circle highlights the "peek definition" icon in the Explorer bar.
- A red circle highlights the "Problems" view in the bottom right corner, which lists three linting errors:

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

- A red circle highlights the "SSH: thefarm2" connection in the bottom status bar.



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

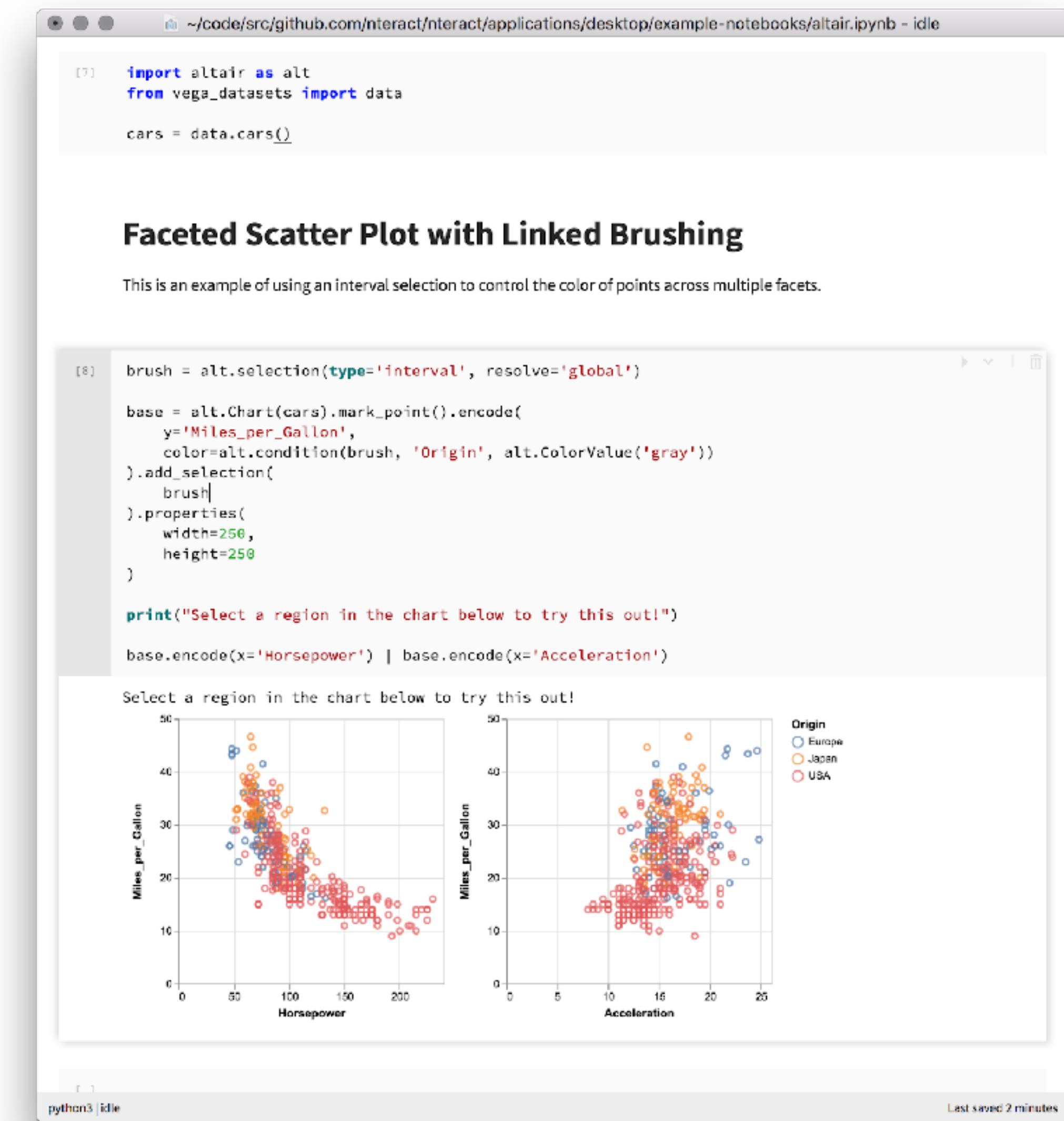
The screenshot shows a code editor with a Python file open. The code defines a class `Experiment` with a method `train_model`. The method signature includes type hints: `model: Model`, `dataset: Dataset`, `epochs: int`, `batch_size: int`, `gpu_ind: Optional[int]`, and `use_wandb: bool`. The code also includes docstrings and a print statement. Below the code editor is a terminal window showing three static analysis errors:

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



Jupyter Notebooks

- Notebooks have become fundamental to data science
 - Great as the "first draft" of a project
 - Jeremy Howard great to learn from ([course.fast.ai](#) videos)



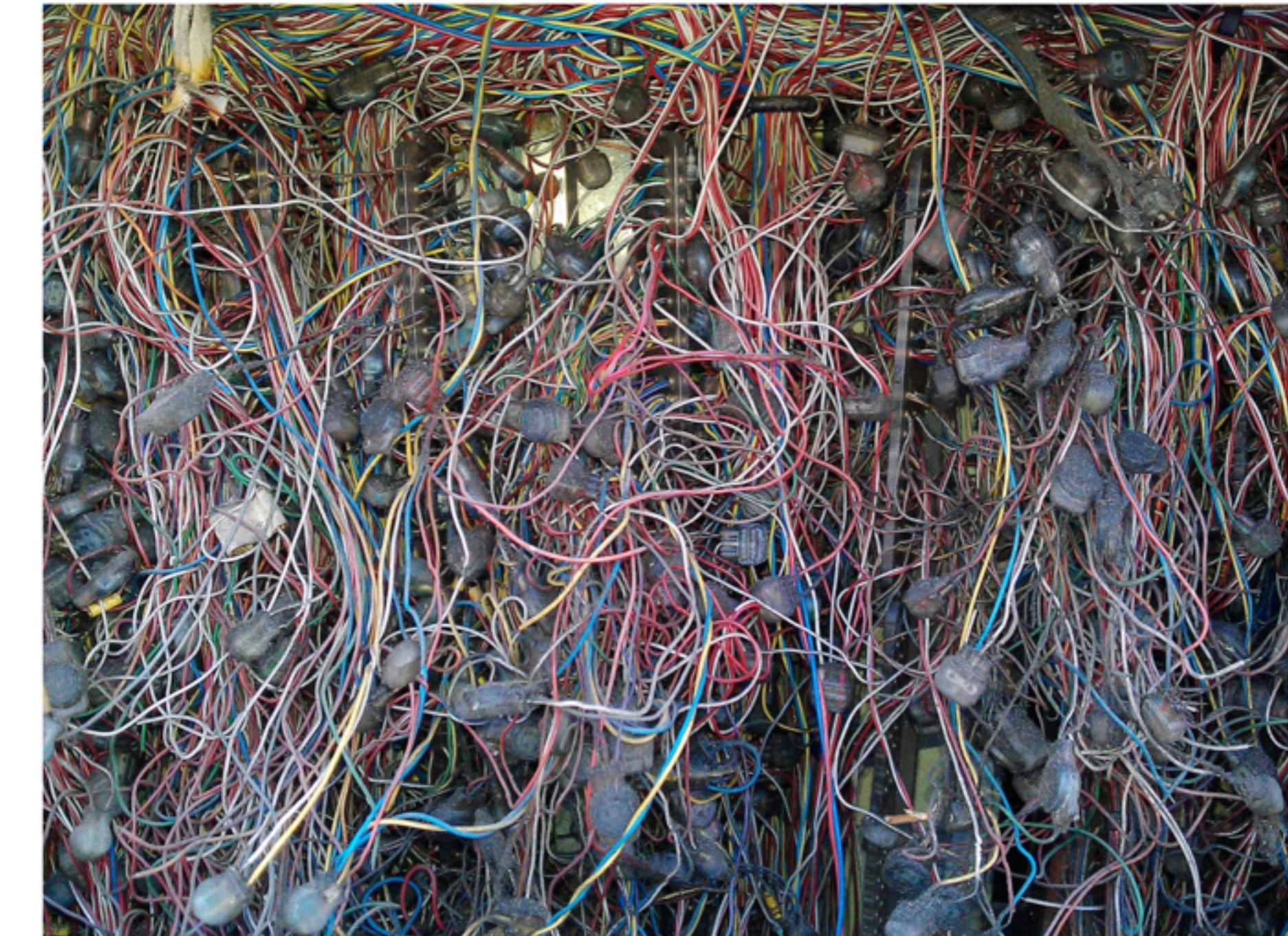


Problems with notebooks

- Notebook editor is primitive
- Out-of-order execution artifacts
- Hard to version
- Hard to test

5 reasons why jupyter notebooks suck

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

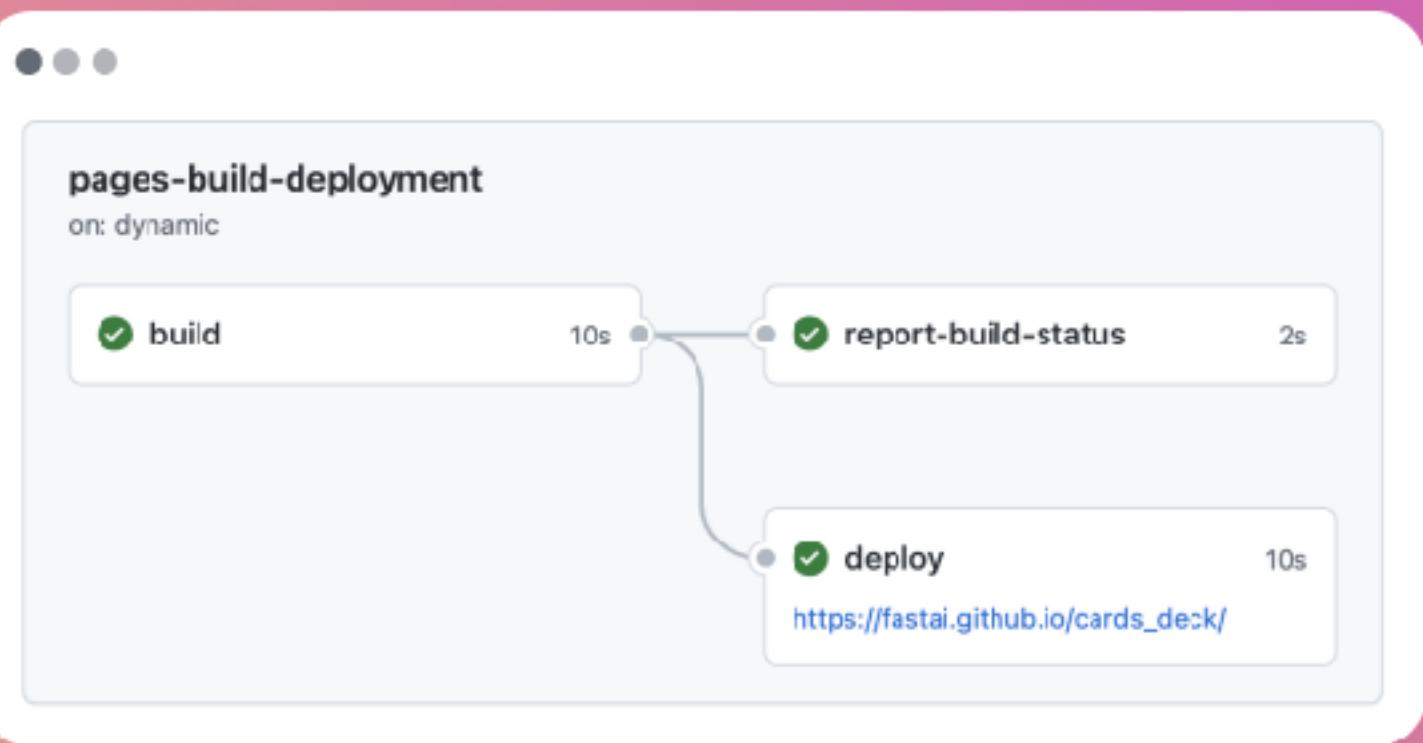


How it feels like managing jupyter notebooks (Complexity ©
<https://www.flickr.com/photos/bitterjug/7670055210>)

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



Counterpoint: nbdev



A Jupyter Notebook cell showing a deck of cards. The code defines a Deck class and creates a new deck object. It then prints the first 10 cards in the deck. The output shows a list of 10 cards: A♣; 2♣; 3♣; 4♣; 5♣; 6♣; 7♣; 8♣; 9♣; 10♣; J♣; Q♣; K♣; A♦; 2♦; 3♦; 4♦; 5♦; 6♦; 7♦; 8♦; 9♦; 10♦; J♦; Q♦; K♦; A♥; 2♥; 3♥; 4♥; 5♥; 6♥; 7♥; 8♥; 9♥; 10♥; J♥; Q♥; K♥; A♠; 2♠; 3♠; 4♠; 5♠; 6♠; 7♠; 8♠; 9♠; 10♠; J♠; Q♠; K♠.

There are 52 cards in a deck.

```
deck = Deck()
deck
```

```
A♣; 2♣; 3♣; 4♣; 5♣; 6♣; 7♣; 8♣; 9♣; 10♣; J♣; Q♣; K♣;
A♦; 2♦; 3♦; 4♦; 5♦; 6♦; 7♦; 8♦; 9♦; 10♦; J♦; Q♦; K♦;
A♥; 2♥; 3♥; 4♥; 5♥; 6♥; 7♥; 8♥; 9♥; 10♥; J♥; Q♥; K♥;
A♠; 2♠; 3♠; 4♠; 5♠; 6♠; 7♠; 8♠; 9♠; 10♠; J♠; Q♠; K♠
```

test_eq(len(deck), 52)

```
#/export
@patch
def pcp(self:Deck,
        index=-1): # Card number to pop
    "Removes and returns card `index` from the deck"
    return self.cards.pop(index)
```

deck.pop()

```
K♣
```

There are 51 cards left in the deck now.

```
test_eq(len(deck), 51)
```



A Quarto documentation table for the "Deck.pop" method. It includes columns for Type (int), Default (-1), and Details (Card number to pop). The table also shows the code for the deck.pop() method.

| Deck.pop | | | |
|---|---------|---------|--------------------|
| Deck.pop (index=-1) | | | |
| Removes and returns card <code>index</code> from the deck | | | |
| Type | Default | Details | |
| index | int | -1 | Card number to pop |
| deck.pop() | | K♣ | |



A screenshot of the Anaconda.org package page for "fastai / packages / cards-deck 0.0.3". It shows the Conda package details, including the URL <https://anaconda.org/fastai/cards-deck/0.0.3>, file size 0, and last upload 14 hours and 56 minutes ago. The page also lists the package's GitHub repository and documentation links.

fastai / packages / cards-deck 0.0.3

An example repository for nbdev

Conda Files Labels Badges

License: Apache Software Home: <https://fastai.github.io> Development: <https://fastai.github.io> Documentation: <https://fastai.github.io> 7 total downloads Last upload: 14 hours and 56 minutes ago



A screenshot of the PyPi package page for "cards-deck 0.0.3". It shows the package details, including the URL <https://pypi.org/project/cards-deck/>, version 0.0.3, and release time about 15 hours ago. The page also lists the package's GitHub repository and documentation links.

cards-deck 0.0.3

pip install cards-deck

Released: about 15 hours ago

An example repository for nbdev

Navigation Project description

Project description Deck of Cards



FSDL Recommendation

<https://github.com/full-stack-deep-learning/sample-repo>

- Write code in modules imported into notebooks

The screenshot shows a Visual Studio Code (VS Code) interface with the following components:

- Explorer View:** Shows a file tree for a repository named "SAMPLE-REPO". It contains a ".vscode" folder, a "notebooks" folder (which includes ".ipynb_checkpoints" and "00-sample_notebook.ipynb"), a "sample_package" folder (containing "__pycache__", "sample_module.py", ".gitignore", and "readme.md"), and a ".gitignore" file.
- Code Editor:** An open Jupyter notebook titled "00-sample_notebook.ipynb". The code in the editor is as follows:

```
# Live-reload of imported modules
%load_ext autoreload
%autoreload 2

# Ensure that sample_package is importable
import os
import sys
REPO_DIRNAME = os.path.abspath('..')
if REPO_DIRNAME not in sys.path:
    sys.path.append(REPO_DIRNAME)

# Import away!
from sample_package.sample_module import SampleClass
```

- Terminal:** Shows the command "sample-repo git:(main)".
- Output:** Shows two code cells. Cell [1] contains the code "sc = SampleClass()" and "sc.sample_method()", with a result of "42". Cell [2] shows the output "42".
- Right Panel:** Shows the contents of "sample_module.py" which defines a class "SampleClass" with an __init__ method setting self.x to 42, and a sample_method method returning self.x.

- Enables awesome debugging

EXPLORER ...

SAMPLE-REPO

- > .vscode
- < notebooks
 - > .ipynb_checkpoints
 - 00-sample_notebook.ip... M
- < sample_package
 - > __pycache__
 - sample_module.py
- .gitignore
- readme.md

notebooks > 00-sample_notebook.ipynb > sc = SampleClass()

+ Code + Markdown | ▶ Run All ... Python 3.9.13 64-bit

```
# Live-reload of imported modules
%load_ext autoreload
%autoreload 2

# Ensure that sample_package is importable
import os
import sys
REPO_DIRNAME = os.path.abspath('..')
if REPO_DIRNAME not in sys.path:
    sys.path.append(REPO_DIRNAME)

# Import away!
from sample_package.sample_module import SampleClass

[1] ✓ 0.2s Python
```

[2] ✓ 0.2s Python

... 42

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

(base) → sample-repo git:(main) ✘

OUTLINE

TIMELINE

main* ◎ 0 △ 1 -- INSERT --

Jupyter Serv

```
1 class SampleClass:
2     def __init__(self):
3         self.x = 42
4
5     def sample_method(self):
6         return self.x
```

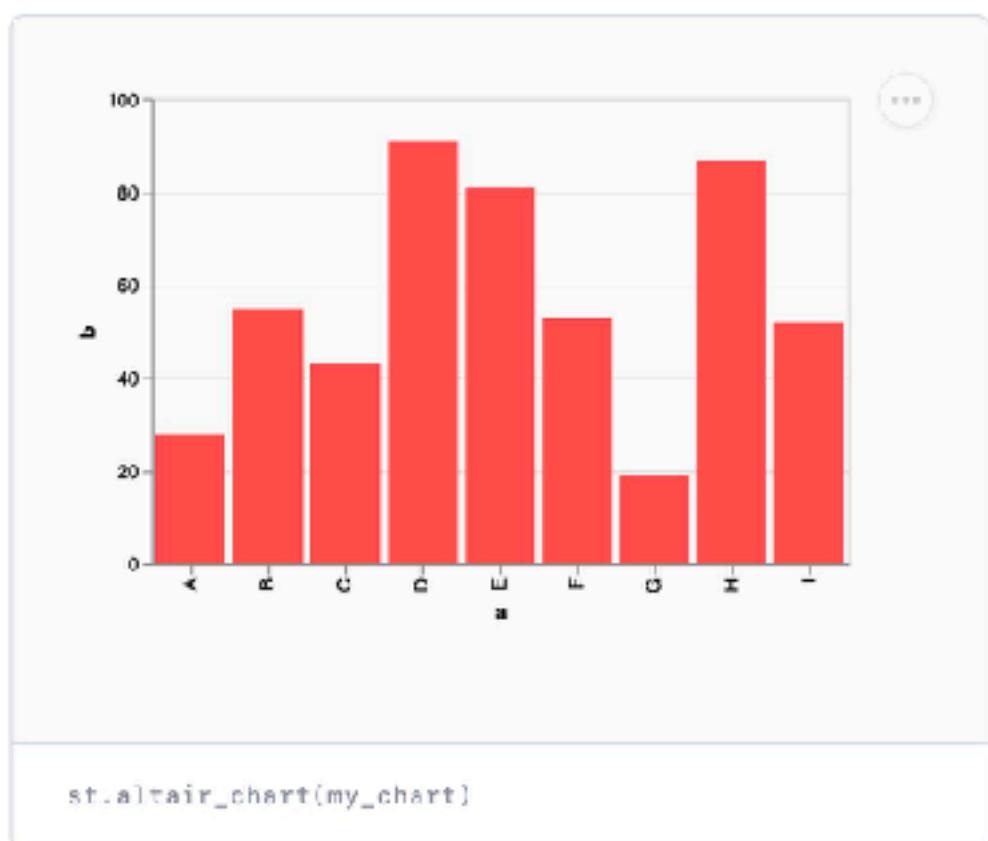
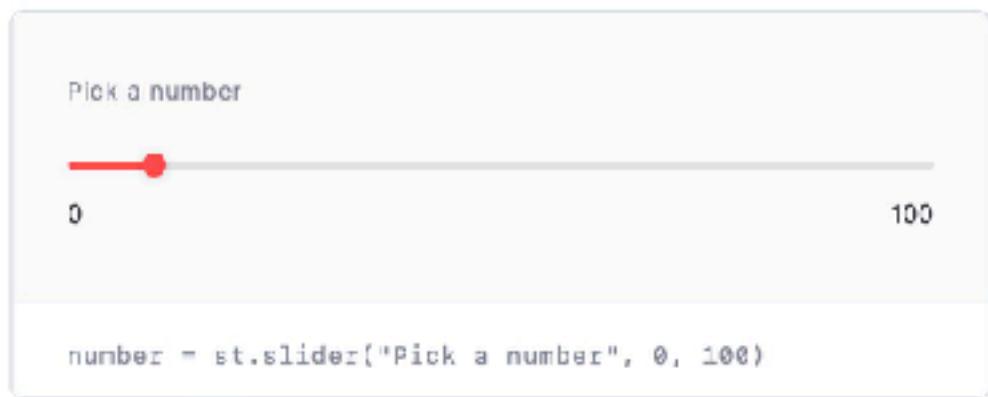
- Enables awesome debugging

The screenshot shows a Python debugger interface with the following details:

- Top Bar:** Shows the file "00-sample_notebook.ipynb" and various icons for navigation and configuration.
- VARIABLES View:** Shows the "Locals" section with a variable "self" pointing to a sample package object. It also includes a "Globals" section.
- WATCH View:** Empty.
- CALL STACK View:** Shows the call stack with the current frame "sample_method" from "sample_module.py" highlighted. Other frames include "MainT...", "00-sample_notebook.ipynb", and several threads.
- BREAKPOINTS View:** Shows checkboxes for exception types. The "Uncaught Exceptions" checkbox is checked, and a breakpoint is set on line 6 of "sample_module.py".
- Code Editor:** Displays the code for "00-sample_notebook.ipynb" with a yellow highlight on the return statement of the "sample_method" definition.
- Debug Console:** Shows the execution of the code: "sc = SampleClass()" and "sc.sample_method()", with a result of "42".
- Bottom Status Bar:** Shows the file name "main*", line count "0", warning count "1", and mode "-- NORMAL --".
- Bottom Right:** Includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and JUPYTER, and a "Filter (e.g. text, !exclude)" input field.

Streamlit

- Decorate Python code
- Get interactive applets
- Publish to the web



Pick a file

Drag and drop files here
Limit 200MB per file + TXT

file = st.file_uploader("Pick a fi

Pick a pet

Dog

Cat

Bird

pet = st.radioButton("Pic

Pick a date

← August 2022 →

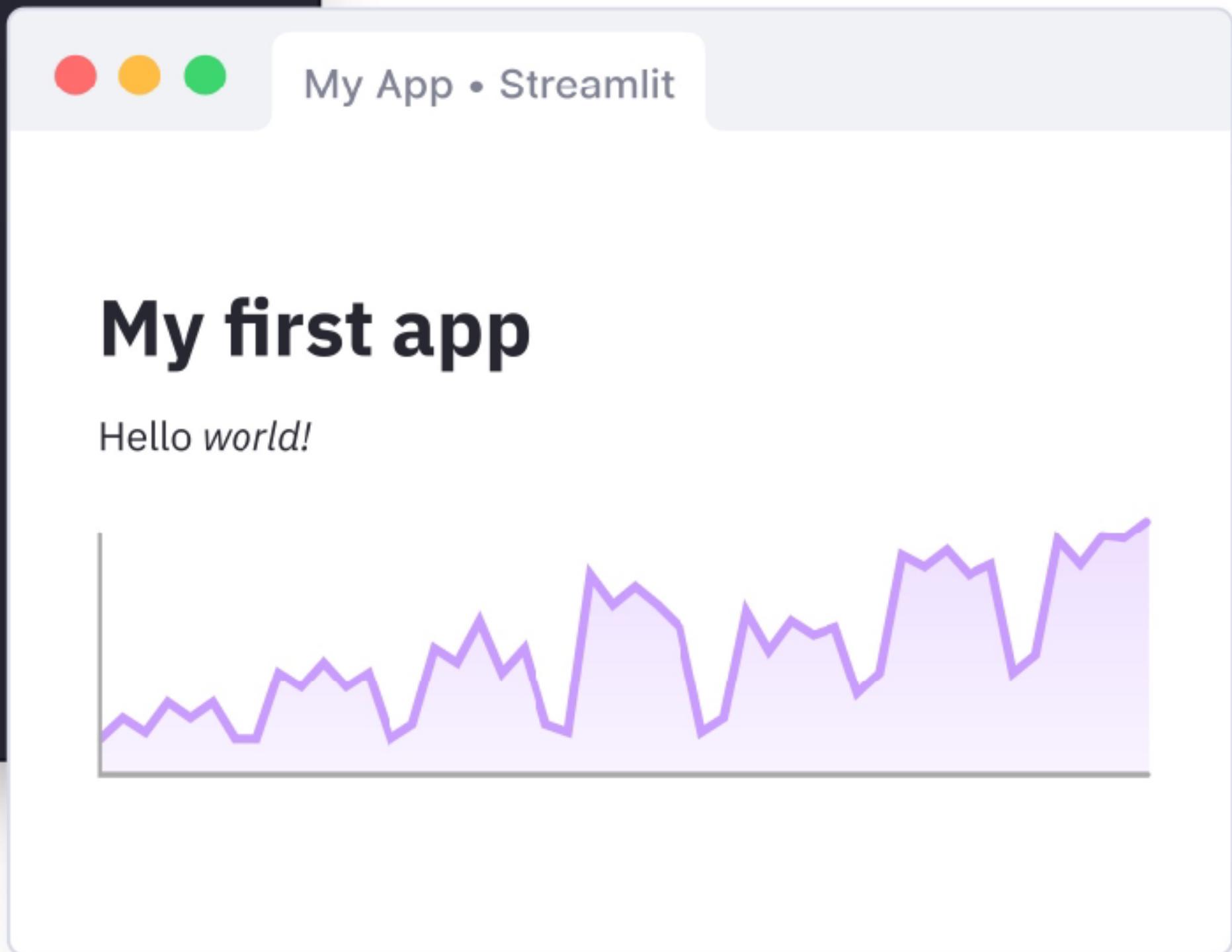
| S | M | T | W | T | F | S |
|----|----|----|----|----|----|----|
| | | | | 5 | 6 | |
| 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 28 | 29 | 30 | 31 | | | |

```
MyApp.py
```

```
import streamlit as st
import pandas as pd

st.write("""
# My first app
Hello *world!*
""")

df = pd.read_csv("my_data.csv")
st.line_chart(df)
```





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>



Go through Labs 1-3!

Full Stack Deep Learning

Search

FSDL Website
☆409 ₩99

Course
FSDL 2022

Synchronous Cohort Option

Lecture 1: Course Vision and When to Use ML

Labs 1-3: CNNs, Transformers, PyTorch Lightning

FSDL 2021

Older

Labs 1-3: CNNs, Transformers, PyTorch Lightning



As part of Full Stack Deep Learning 2022, we will incrementally develop a complete deep learning codebase to create and deploy a model that understands the content of hand-written paragraphs.

For an overview of the Text Recognizer application architecture, click the badge below to open an interactive Jupyter notebook on Google Colab:

Open in Colab

This first set of "review" labs covers deep learning fundamentals and introduces some of the core libraries we will use.

While we recommend you read through all of these labs, we recommend only completing the exercises for the labs that are of most interest to you.

Click the badges below to access individual lab notebooks on Colab

| Notebook | Link |
|---|---------------|
| Lab 00: Overview | Open in Colab |
| Lab 01: Deep Neural Networks in PyTorch | Open in Colab |
| Lab 02a: PyTorch Lightning | Open in Colab |
| Lab 02b: Training a CNN on Synthetic Handwriting Data | Open in Colab |
| Lab 03: Transformers and Paragraphs | Open in Colab |

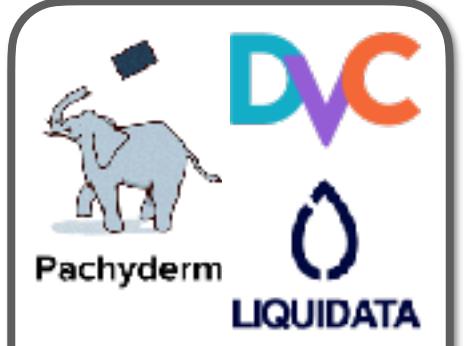
“All-in-one”



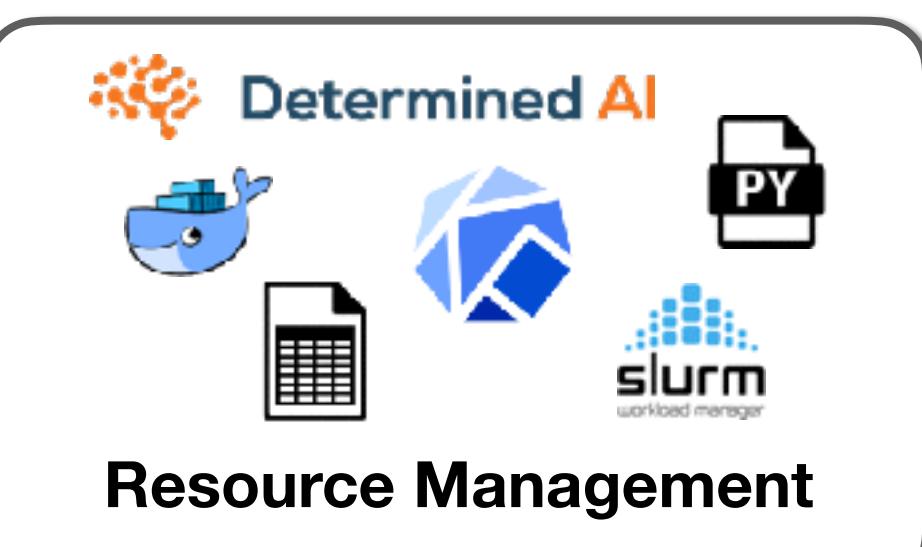
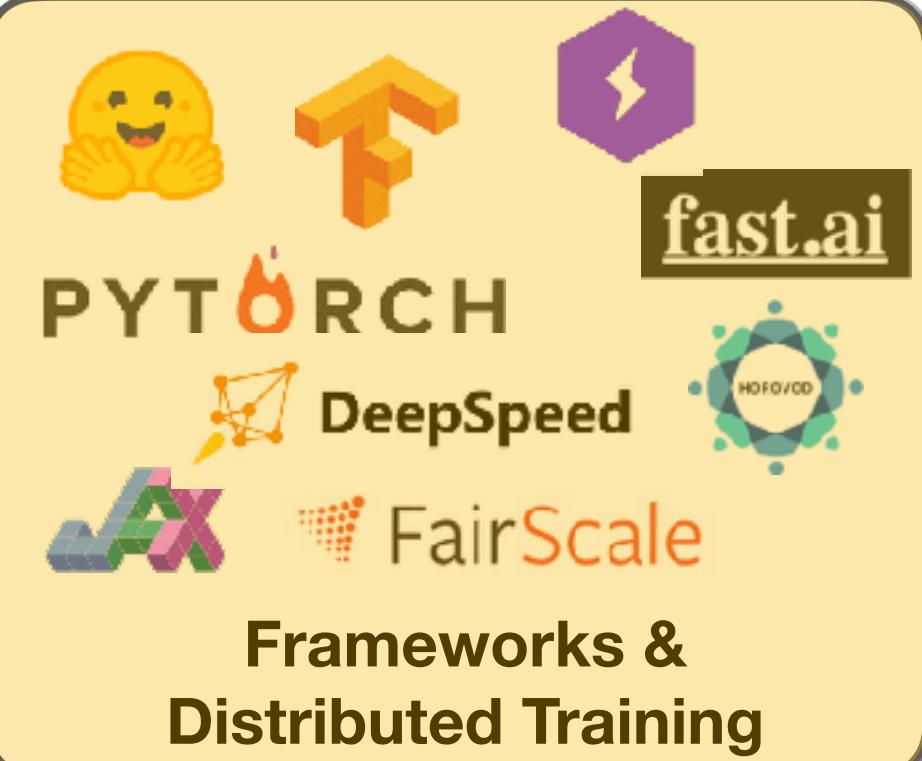
Amazon SageMaker

gradient^o
by Paperspace

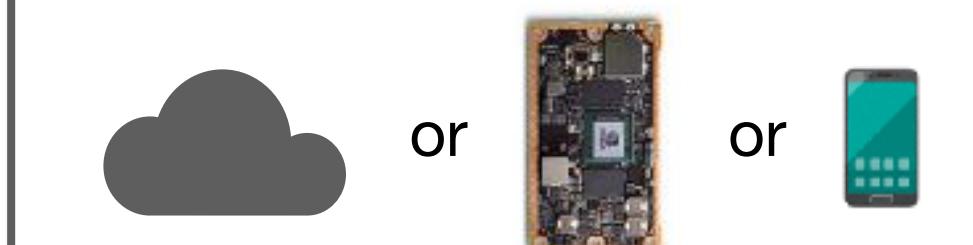
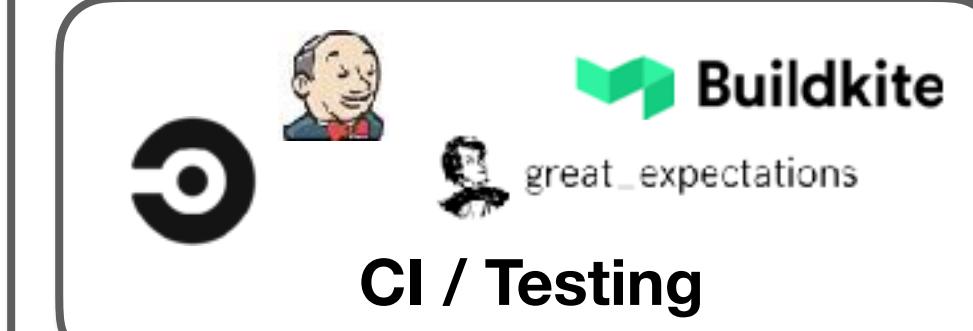
DOMINO
DATA LAB



Data



Development





Deep Learning Frameworks



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.

```
✓ class Linear:  
✓   def __init__(self, input_dim: int, num_hidden: int = 1):  
      self.weights = np.random.randn(input_dim, num_hidden) * np.sqrt(2. / input_dim)  
      self.bias = np.zeros(num_hidden)  
  
✓   def __call__(self, x):  
      self.x = x  
      output = x @ self.weights + self.bias  
      return output  
  
✓   def backward(self, gradient):  
      self.weights_gradient = self.x.T @ gradient  
      self.bias_gradient = gradient.sum(axis=0)  
      self.x_gradient = gradient @ self.weights.T  
      return self.x_gradient  
  
✓   def update(self, lr):  
      self.weights = self.weights - lr * self.weights_gradient  
      self.bias = self.bias - lr * self.bias_gradient
```



Which to use?



Josh Tobin

@josh_tobin_

...

Why do people always ask what ML framework to use?
It's easy:

- jax is for researchers
- pytorch is for engineers
- tensorflow is for boomers



6:24 PM · Mar 11, 2021 · Twitter Web App

263 Retweets

65 Quote Tweets

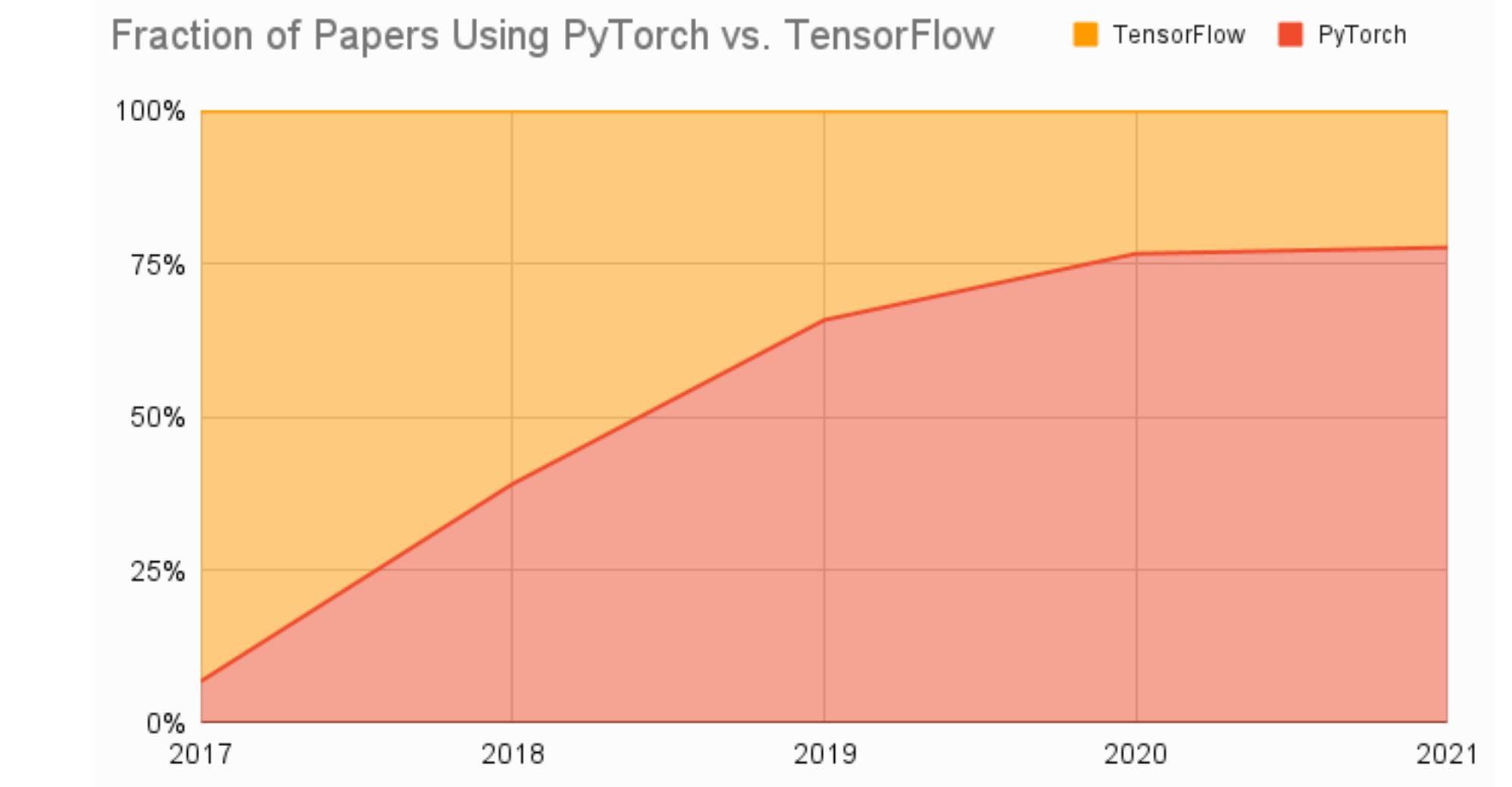
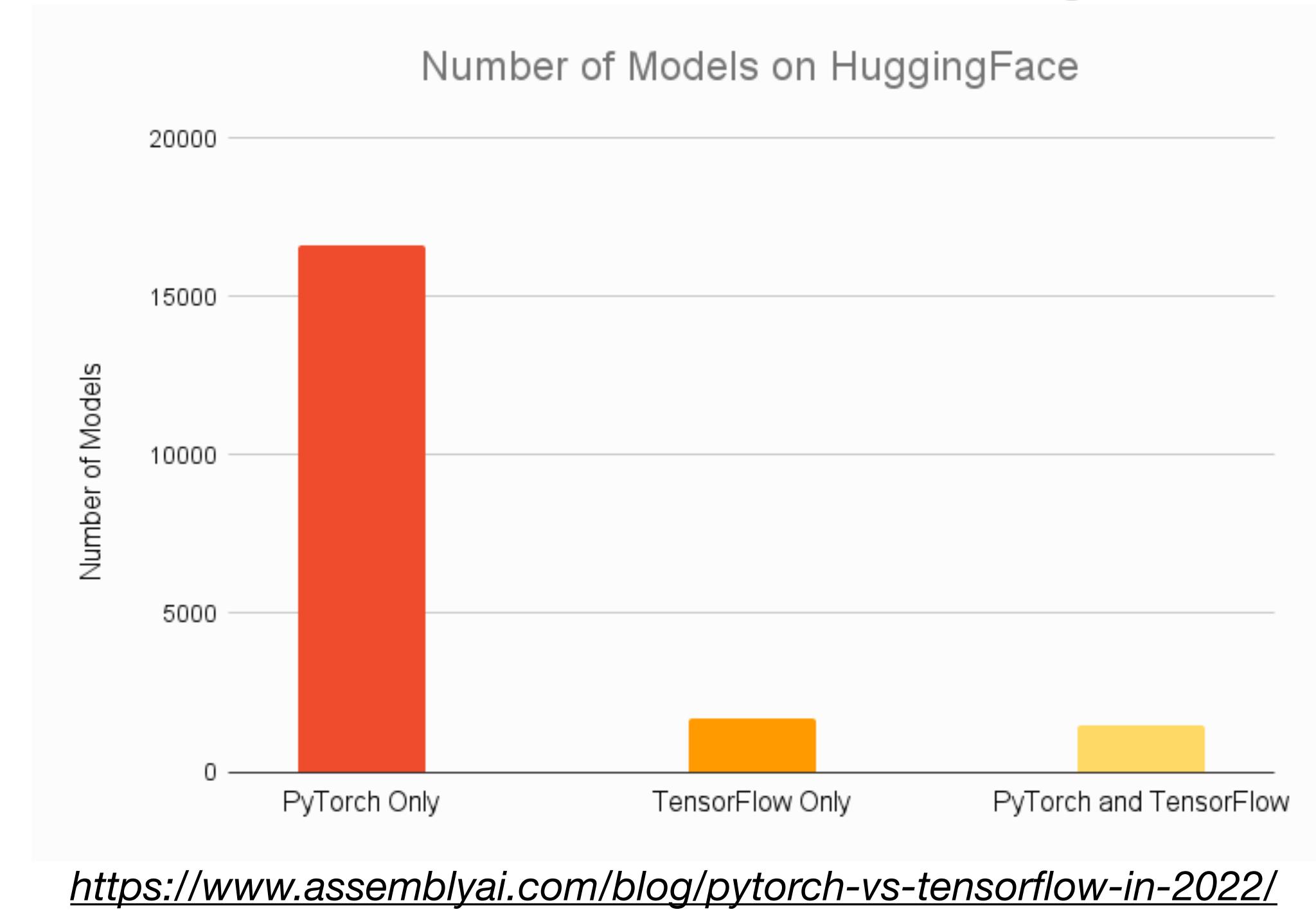
2,884 Likes

But seriously

- PyTorch / Tensorflow / Jax
 - Define model by running Python code
 - Optimized execution graph for CPU, GPU, TPU, mobile
- PyTorch
 - 😊 Absolutely dominant (by # models, # papers, # competition winners)
- Tensorflow
 - 😊 Tensorflow.js and Keras are pretty cool
- Jax
 - 😢 Need a meta-framework for deep learning

77% of 2021 ML competition winners used PyTorch

<https://blog.mlcontests.com/p/winning-at-competitive-ml-in-2022?s=w>



PyTorch

- Excellent dev experience
- Production-ready as is, and even faster with TorchScript
- Great distributed training ecosystem
- Libraries for vision, audio, etc
- Mobile deployment

```
training_data = datasets.FashionMNIST(  
    root="data",  
    train=True,  
    download=True,  
    transform=ToTensor()  
)  
  
train_dataloader = DataLoader(training_data, batch_size=64)  
test_dataloader = DataLoader(test_data, batch_size=64)  
  
class NeuralNetwork(nn.Module):  
    def __init__(self):  
        super(NeuralNetwork, self).__init__()  
        self.flatten = nn.Flatten()  
        self.linear_relu_stack = nn.Sequential(  
            nn.Linear(28*28, 512),  
            nn.ReLU(),  
            nn.Linear(512, 512),  
            nn.ReLU(),  
            nn.Linear(512, 10),  
        )  
  
    def forward(self, x):  
        x = self.flatten(x)  
        logits = self.linear_relu_stack(x)  
        return logits  
  
model = NeuralNetwork()
```

+ PyTorch Lightning



```
class LitAutoEncoder(pl.LightningModule):
    def __init__(self):
        super().__init__()
        self.encoder = nn.Sequential(
            nn.Linear(28 * 28, 64),
            nn.ReLU(),
            nn.Linear(64, 3))
        self.decoder = nn.Sequential(
            nn.Linear(3, 64),
            nn.ReLU(),
            nn.Linear(64, 28 * 28))

    def forward(self, x):
        embedding = self.encoder(x)
        return embedding

    def configure_optimizers(self):
        optimizer = torch.optim.Adam(self.parameters(), lr=1e-3)
        return optimizer

    def training_step(self, train_batch, batch_idx):
        x, y = train_batch
        x = x.view(x.size(0), -1)
        z = self.encoder(x)
        x_hat = self.decoder(z)
        loss = F.mse_loss(x_hat, x)
        self.log('train_loss', loss)
        return loss
```

- Run your code on any hardware
- Performance & bottleneck profiler
- Model checkpointing
- 16-bit precision
- Run distributed training

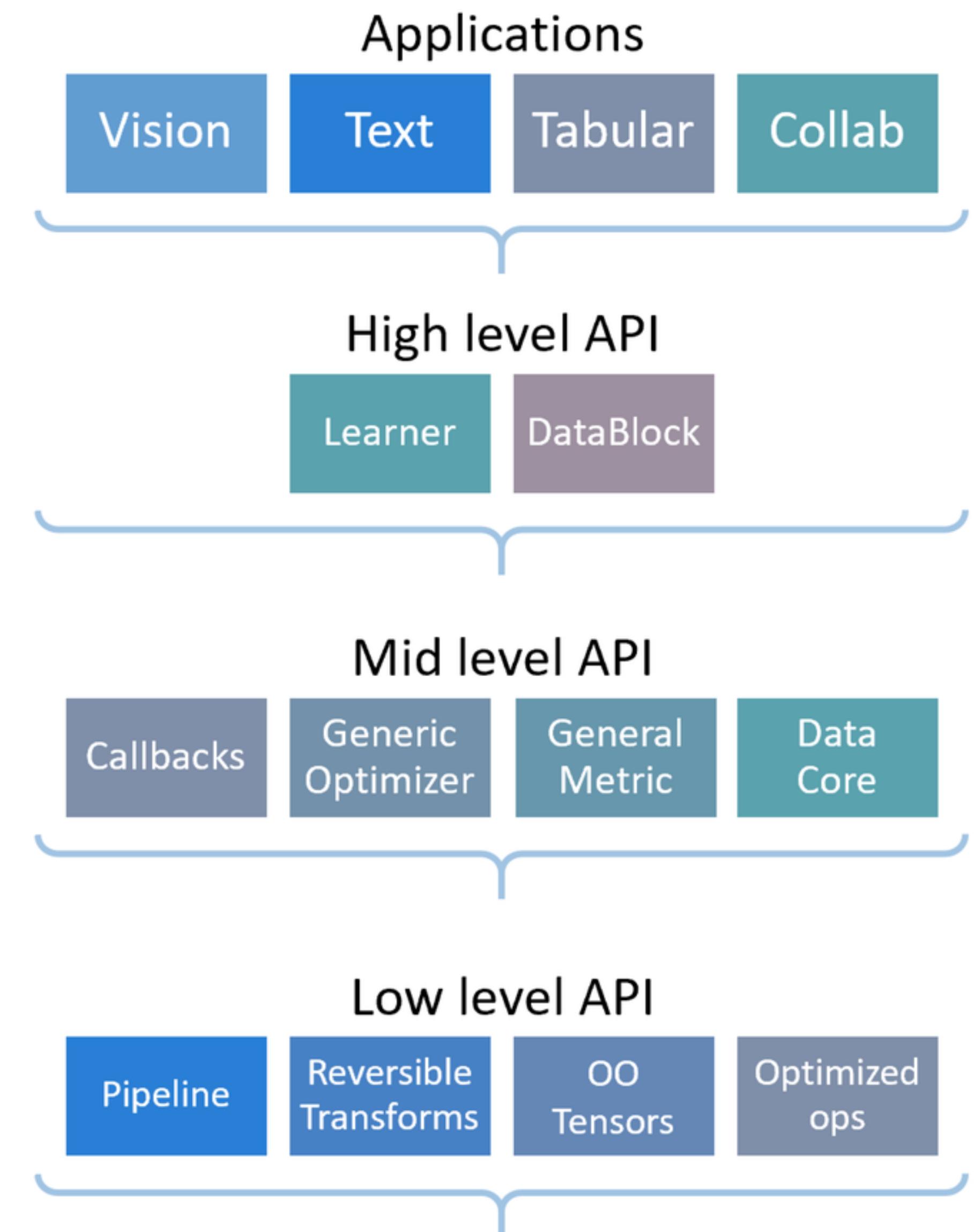
```
# data
dataset = MNIST('', train=True, download=True,
                 transform=transforms.ToTensor())
mnist_train, mnist_val = random_split(dataset, [55000, 5000])
train_loader = DataLoader(mnist_train, batch_size=32)
val_loader = DataLoader(mnist_val, batch_size=32)

# model
model = LitAutoEncoder()

# training
trainer = pl.Trainer(gpus=4, precision=16, limit_train_batches=0.5)
trainer.fit(model, train_loader, val_loader)
```

Another possibility: FastAI

- Software developed alongside the fast.ai course
- Provides many advanced tricks (data augmentations, better initializations, learning rate scheduling)
- Code style is quite different from mainstream Python





TensorFlow



- FSDL prefers PyTorch, as the ecosystem is now unquestionably stronger.
- But if you have a specific reason to prefer TensorFlow, could still make sense.

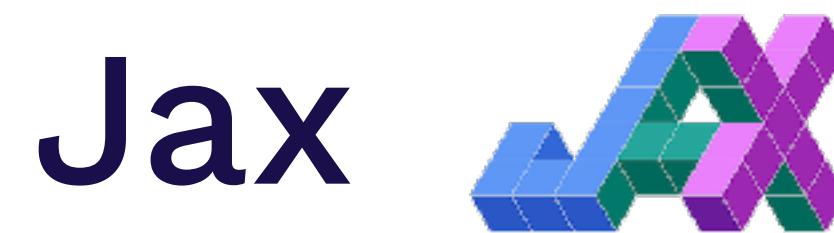
```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```



- General vectorization, auto-differentiation, and compilation to GPU/TPU code
- Need separate framework for deep learning (Flax or Haiku)
- FSDL says: only use if you have a specific need



A large yellow emoji of a face with a finger over its mouth, indicating silence or shushing.



Zac 🦜 @zurkcity · Mar 11, 2021
Replying to [@josh_tobin_](#)
😂 Jax is for people working at G who aren't allowed to use PyTorch

1 reply · 125 retweets · 125 likes



Liam Tremblay @altqwe1 · Mar 11, 2021
this guy knows

1 reply · 10 retweets · 10 likes

...

...

...

...

...



Meta-frameworks and model zoos

PYTORCH HUB

Discover and publish models to a pre-trained model repository designed for research exploration.

Check out the models for [Researchers](#), or learn [How It Works](#).

[Contribute Models](#)

**This is a beta release - we will be collecting feedback and improving the PyTorch Hub over the coming months.*

FOR RESEARCHERS —

EXPLORE AND EXTEND MODELS
FROM THE LATEST
CUTTING EDGE RESEARCH

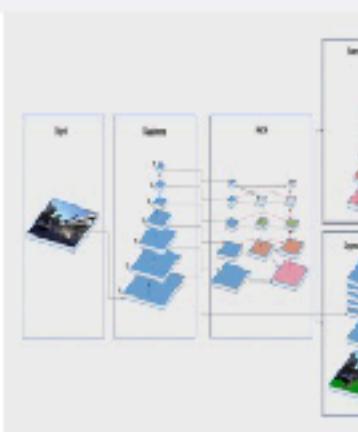
All [Audio](#) [Generative](#) [Nlp](#) [Scriptable](#) [Vision](#)

🔍

Sort ▾

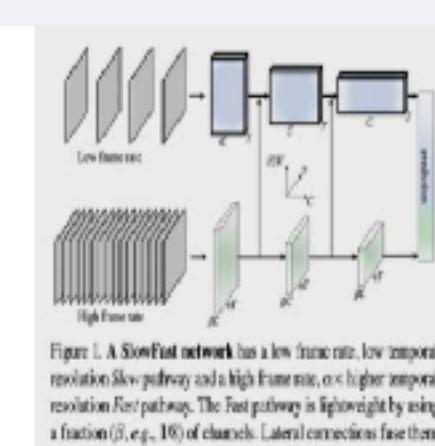
HybridNets  296

HybridNets - End2End Perception Network

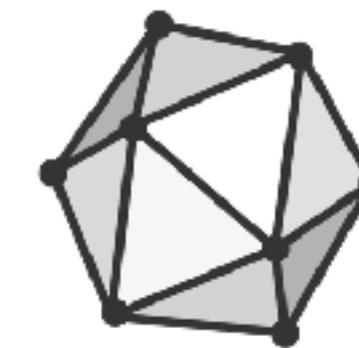


3D ResNet  2.6k

Resnet Style Video classification networks pretrained on the Kinetics 400 dataset



Interoperability



ONNX

- Open standard for **saving** deep learning models
- Can convert PyTorch -> TF, etc.
- Can work well, but can run into edge cases...

Frameworks & Converters

Use the frameworks you already know and love.



Transformers



Keras



PaddlePaddle



PyTorch



SciKit Learn



Tengine



Hugging Face



- The most popular Model Zoo: 60K pre-trained models
 - Transformers library that works with PyTorch, TF, Jax
 - And 7.5K Datasets
 - Plus Spaces, hosted Inference, and more

TIMM

- Collection of SOTA computer vision models + related code
- By Ross Wightman

```
import timm
import torch

model = timm.create_model('resnet34', num_classes=10)
x = torch.randn(1, 3, 224, 224)
model(x).shape
torch.Size([1, 10])
```

| Model | Acc@1 (Err) | Acc@5 (Err) | Param # (M) | Interpolation | Image Size |
|--------------------|--------------------|-------------------|-------------|---------------|----------------|
| efficientnet_b3a | 82.242 (17.758) | 96.114 (3.886) | 12.23 | bicubic | 320 (1.0 crop) |
| efficientnet_b3 | 82.076 (17.924) | 96.020 (3.980) | 12.23 | bicubic | 300 |
| regnet_32 | 82.002 (17.998) | 95.906 (4.094) | 19.44 | bicubic | 224 |
| skresnext50d_32x4d | 81.278 (18.722) | 95.366 (4.634) | 27.5 | bicubic | 288 (1.0 crop) |
| seresnext50d_32x4d | 81.266 (18.734) | 95.620 (4.380) | 27.6 | bicubic | 224 |
| efficientnet_b2a | 80.608 (19.392) | 95.310 (4.690) | 9.11 | bicubic | 288 (1.0 crop) |
| resnet50d | 80.530 (19.470) | 95.160 (4.840) | 25.6 | bicubic | 224 |
| mixnet_xl | 80.478 (19.522) | 94.932 (5.068) | 11.90 | bicubic | 224 |
| efficientnet_b2 | 80.402 (19.598) | 95.076 (4.924) | 9.11 | bicubic | 260 |
| | 80.274 (19.726) | 95.070 (4.930) | 28.1 | bicubic | 224 |
| | 80.156 (19.844) | 94.642 (5.358) | 27.5 | bicubic | 224 |
| | 80.058 (19.942) | 95.084 (4.916) | 27.6 | bicubic | 256 |



Distributed Training

Scenarios

- Multiple machines, with multiple GPUs
- **Data batch** may:
 - Fit on a single GPU
 - Or not
- **Model parameters** may:
 - Fit on a single GPU
 - Or not



Scenarios

- Multiple machines, with multiple GPUs



- **Data batch** may:

- Fit on a single GPU



- Or not

- **Model parameters** may:

- Fit on a single GPU



- OR not





Trivial Parallelism

- Either launch more independent experiments on other GPUs and/or machines
- Or increase batch size until it no longer fits on one GPU...

Scenarios

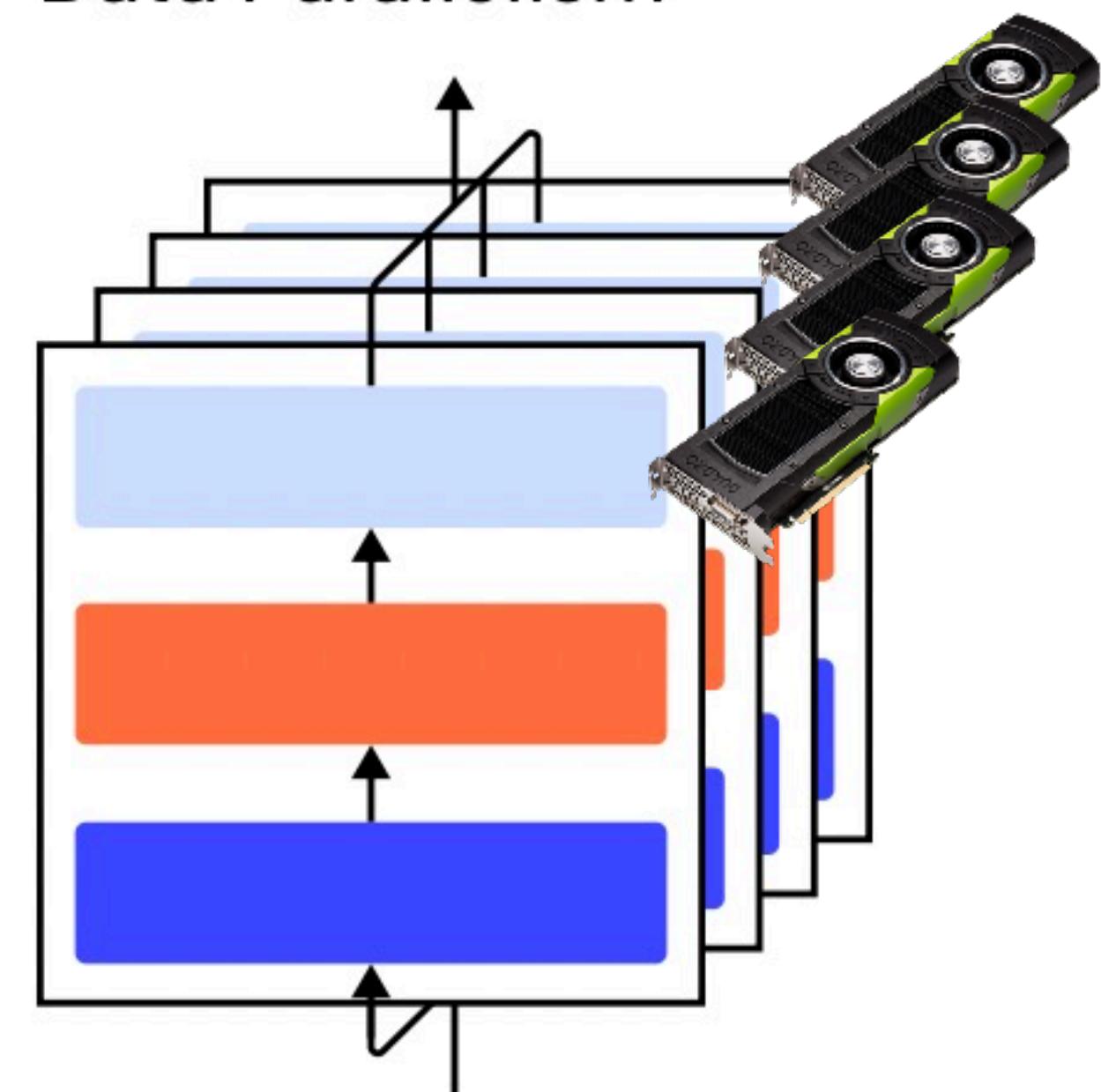
- Multiple machines, with multiple GPUs
- **Data batch** may:
 - Fit on a single GPU
 - Or not
- **Model parameters** may:
 - Fit on a single GPU
 - Or not



Data Parallelism

- Distribute a single batch of data across GPUs, then average gradients across them
- Extra work is cross-GPU, so make sure that GPUs have fast interconnects (NVLink, etc)

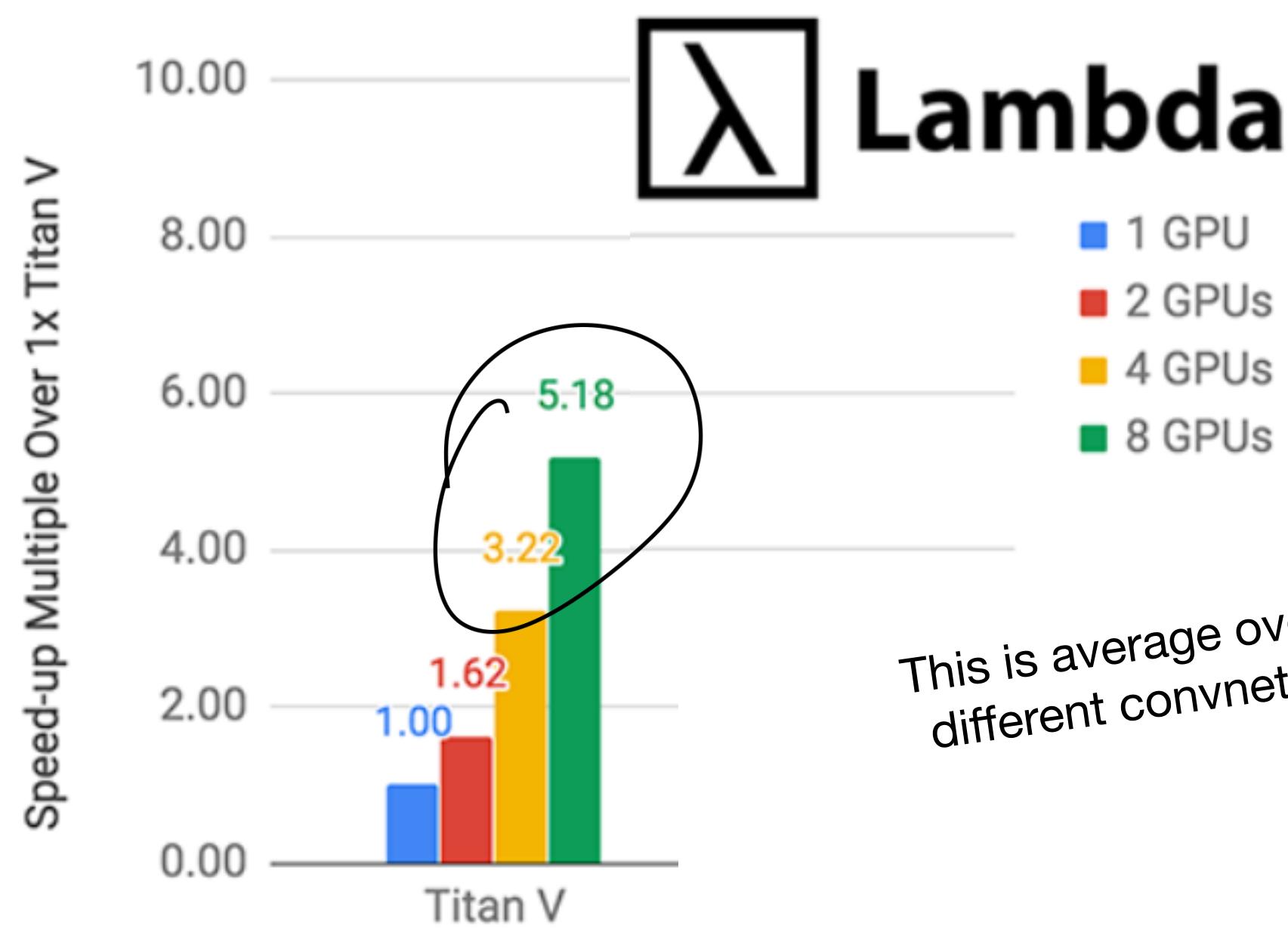
Data Parallelism



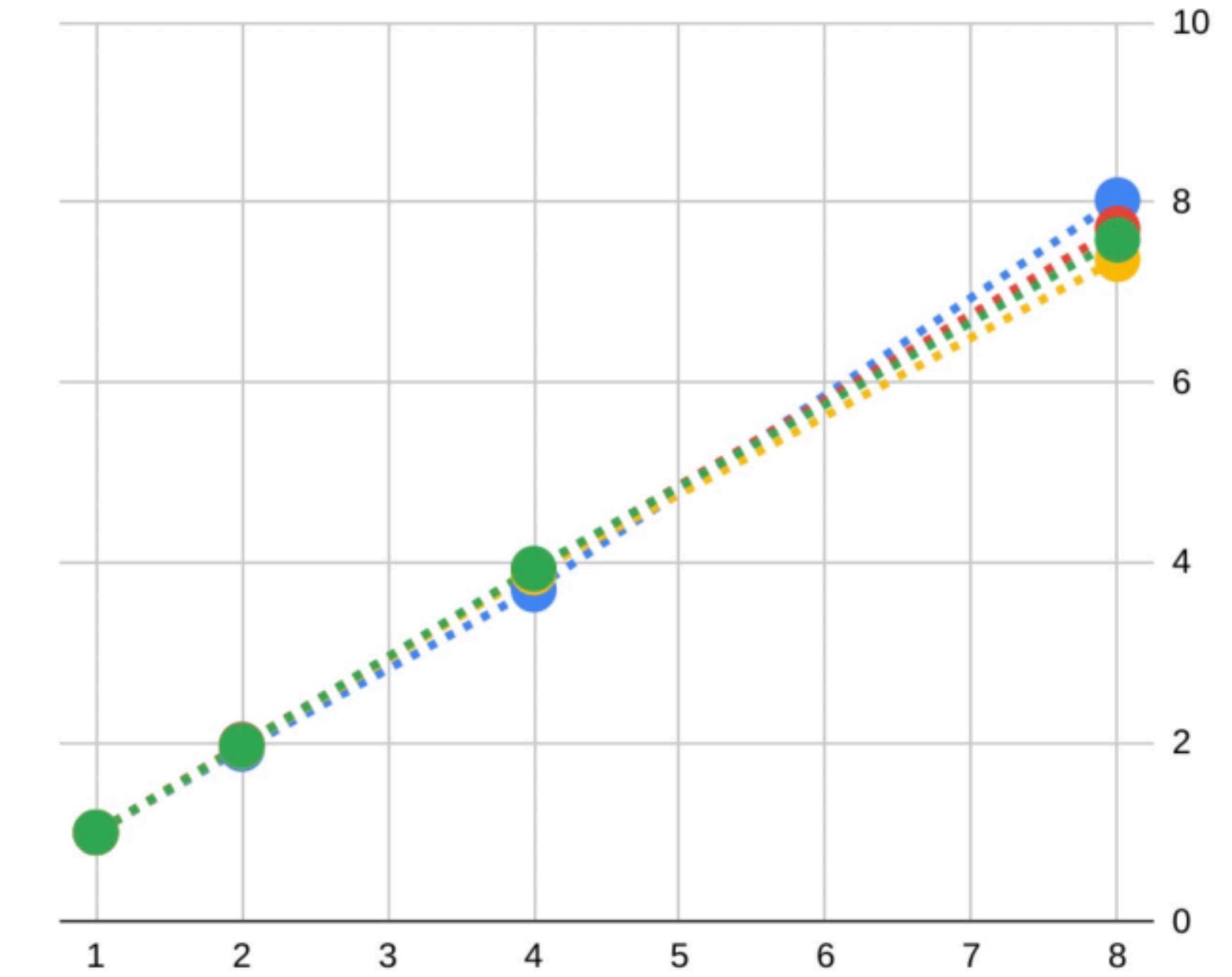
<https://openai.com/blog/techniques-for-training-large-neural-networks/>

Speedup

- Expect a linear speedup for server cards
- But a sublinear speedup for consumer cards (2-3x for 4 GPUs)



● A100 80GB SXM4 (TF32) ● RTX A6000 (TF32)
● V100 32GB (FP32) ● RTX 8000 (FP32)



Data Parallelism

- PyTorch has a robust `DistributedDataParallel` library
- Horovod is a third-party library for the same (and more)
- Lightning makes either one dead-simple to use
- Speedup seems to be the same



```
# train on 8 GPUs (same machine (ie: node))
trainer = Trainer(accelerator="gpu", devices=8, strategy="ddp")

# train on 32 GPUs (4 nodes)
trainer = Trainer(accelerator="gpu", devices=8, strategy="ddp", num_nodes=4)
```

```
# train Horovod on GPU (number of GPUs / machines provided on command-line)
trainer = Trainer(strategy="horovod", accelerator="gpu", devices=1)

# run training with 4 GPUs on a single machine
horovodrun -np 4 python train.py

# run training with 8 GPUs on two machines (4 GPUs each)
horovodrun -np 8 -H hostname1:4,hostname2:4 python train.py
```

| Train Mode | ⌚ V100x1 | ⌚ V100x4 | ⌚ V100x8 |
|---|-------------------|-------------------------|----------------------|
| Vanilla DistributedDataParallel Horovod | 32m 58s - - | - 11m 16s 10m 23s | - 9m 8s 7m 50s |

Scenarios

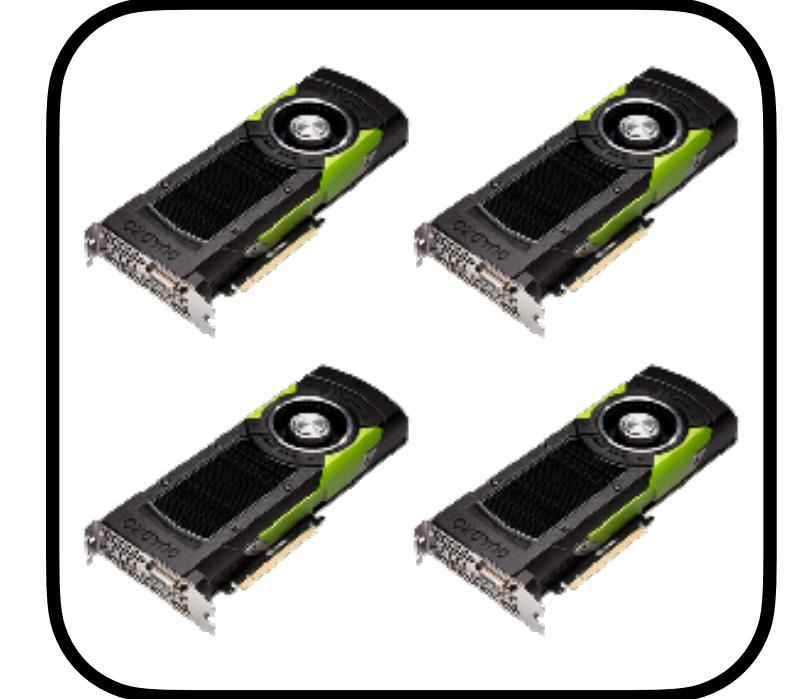
- Multiple machines, with multiple GPUs



- **Data batch** may:

- Fit on a single GPU

- Or not



- **Model parameters** may:

- Fit on a single GPU

- Or not





Going beyond simple data-parallel

- If model does not fit on one GPU, there are three main solutions:
 - Sharded data-parallelism
 - Pipelined model-parallelism
 - Tensor-parallelism

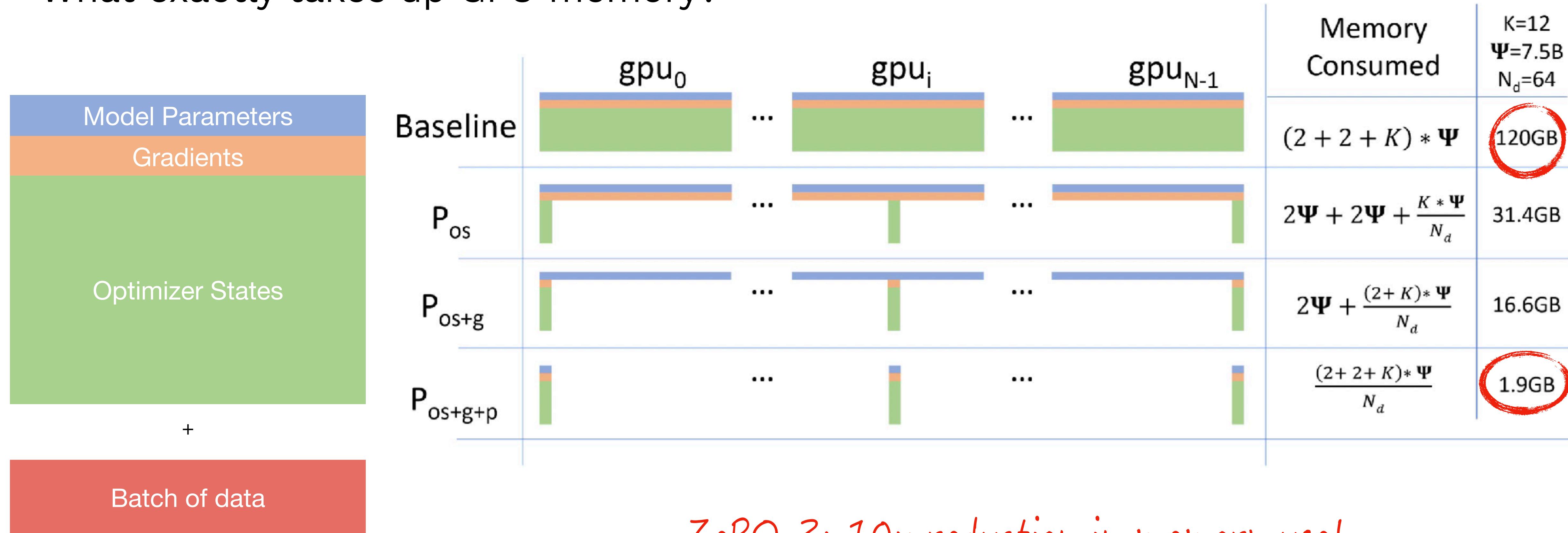


Going beyond simple data-parallel

- If model does not fit on one GPU, there are three main solutions:
 - **Sharded data-parallelism**
 - Pipelined model-parallelism
 - Tensor-parallelism

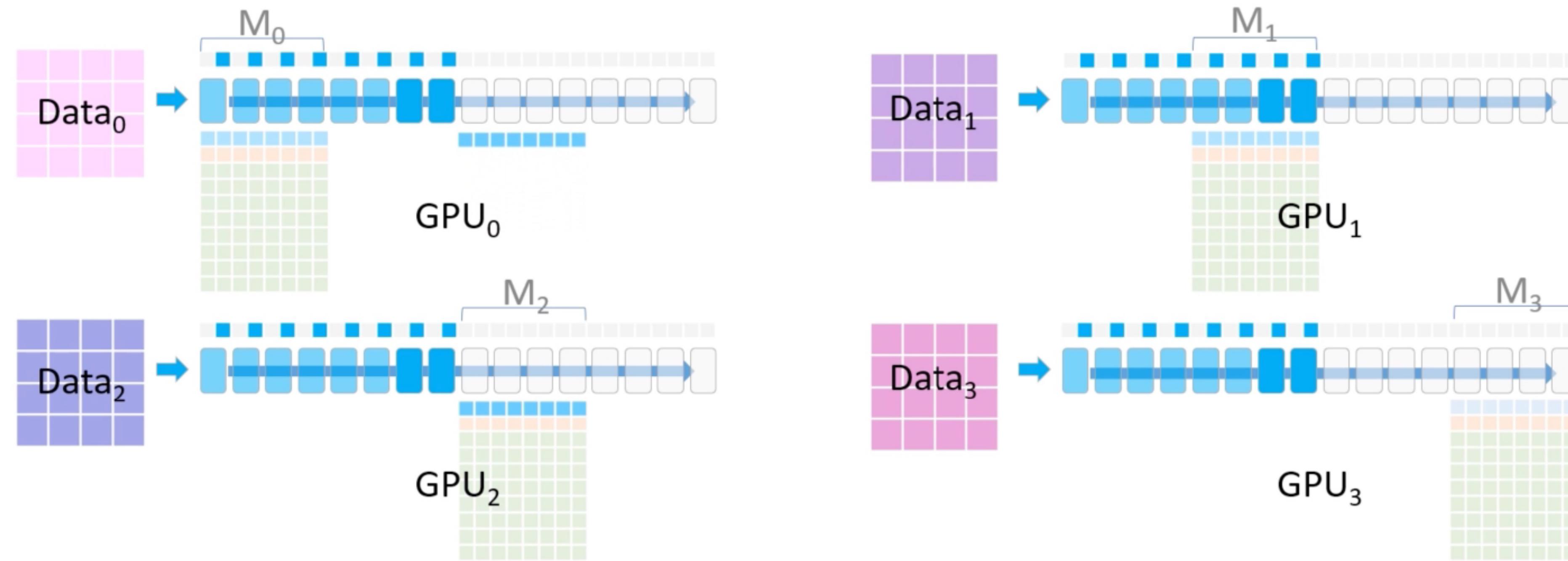
Sharded Data-Parallelism

What exactly takes up GPU memory?



ZeRO-3: 10x reduction in memory use!

Sharded Data-Parallelism



Literally pass around model params between GPUs as computation is proceeding!

Helpful video:

<https://www.microsoft.com/en-us/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/>

Sharded Data-Parallel



- Implemented by Deepspeed library from Microsoft, as well as Fairscale library from Facebook. And recently, natively by PyTorch.
 - ZeRO-3 is called Fully-sharded DataParallel by Fairscale and PyTorch.
- Try for a massive memory reduction without changing model code!

```
# train using Sharded DDP
trainer = Trainer(strategy="ddp_sharded")
```



ZeRO-Offload

- ZeRO-3 principle: only send the GPU the params it needs in the moment
- Same principle can be applied to a single GPU
- Can train a 13B-param model on a single V100 (32GB) GPU!
- (Fairscale calls this CPU-offloading)

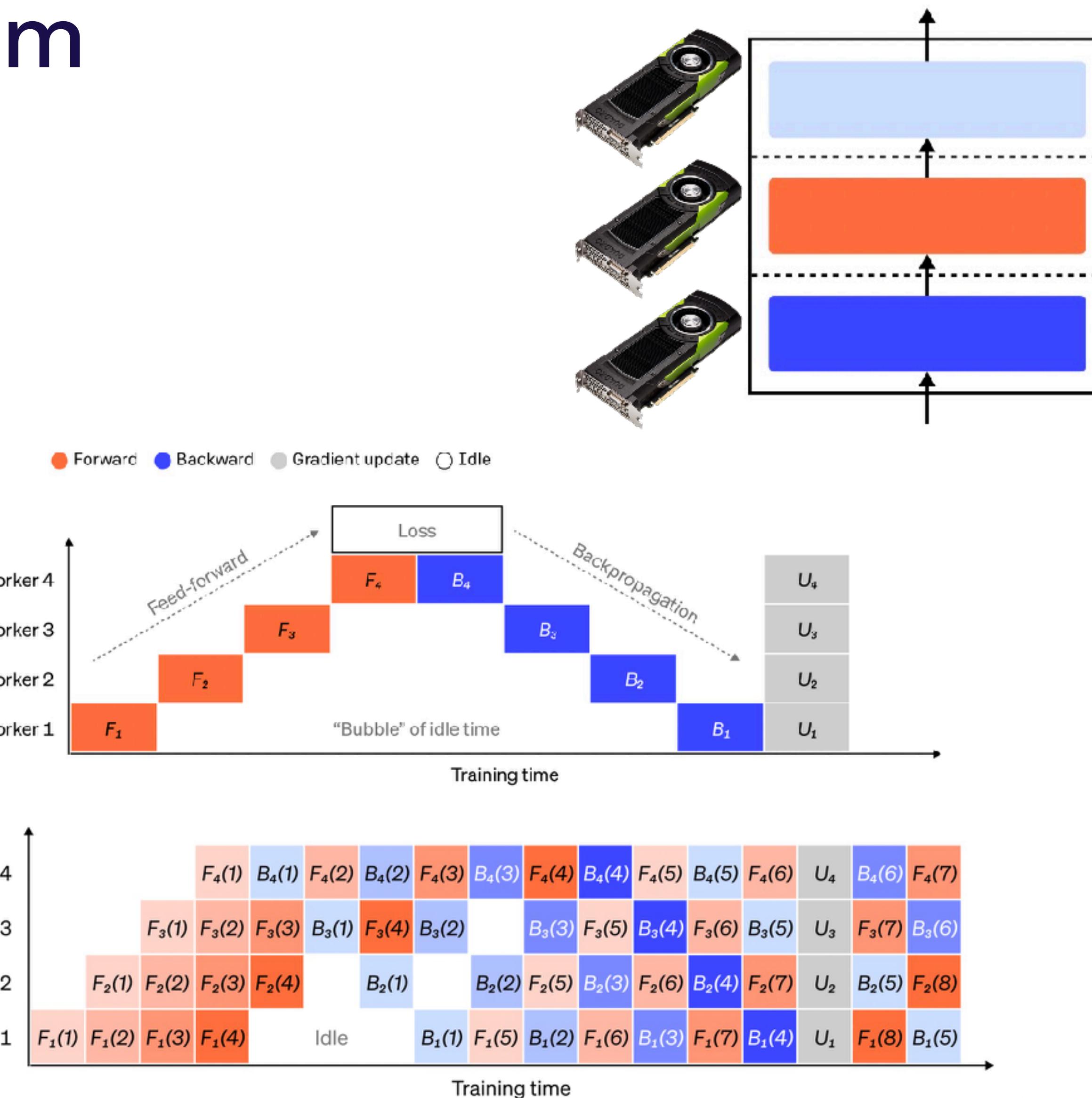


Going beyond simple data-parallel

- If model does not fit on one GPU, there are three main solutions:
 - Sharded data-parallelism
 - **Pipelined model-parallelism**
 - Tensor-parallelism

Pipelined Model-Parallelism

- Trivial to implement naively, but results in only one GPU being active at a time
- Libraries like DeepSpeed and FairScale (and native PyTorch) make it better.
- But need to tune amount of pipelining.





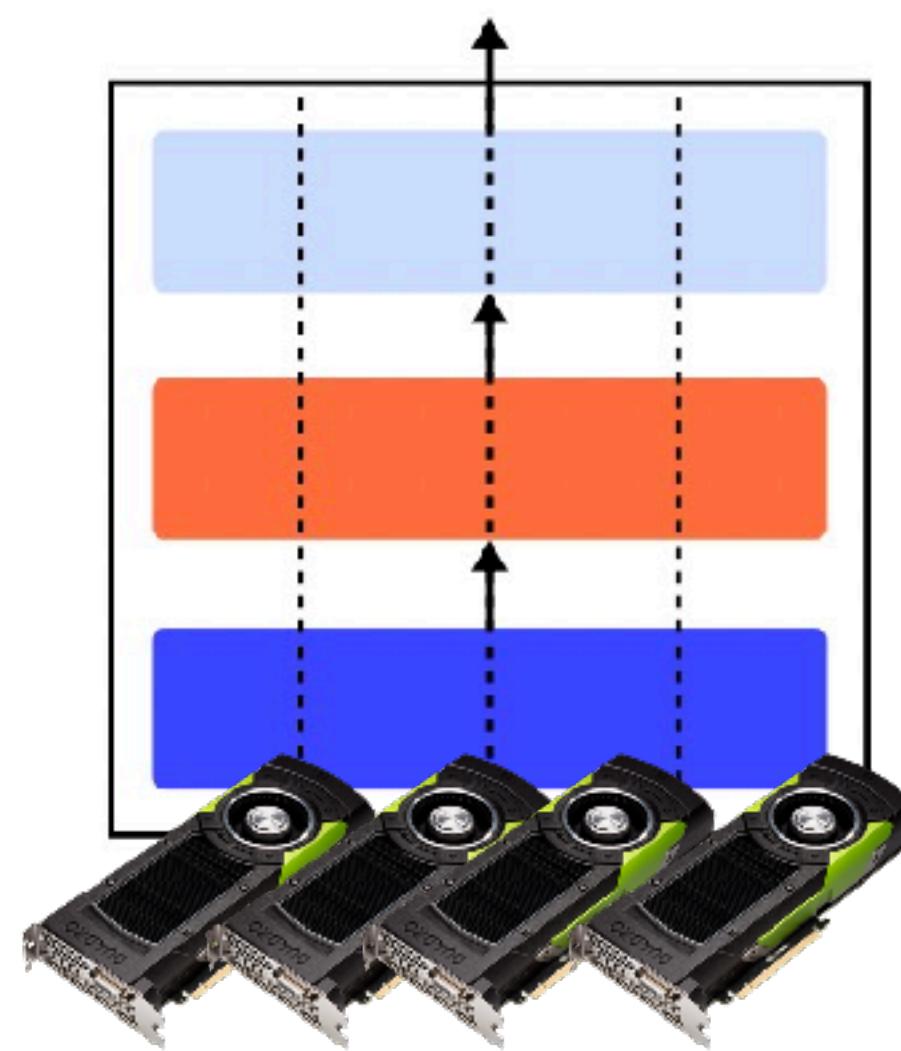
Going beyond simple data-parallel

- If model does not fit on one GPU, there are three main solutions:
 - Sharded data-parallelism
 - Pipelined model-parallelism
 - **Tensor-parallelism**

Tensor Parallelism

- Matrix multiplication itself can be distributed over GPUs
- NVIDIA published Megatron-LM [repo](#) which does this for the Transformer

Tensor Parallelism



<https://openai.com/blog/techniques-for-training-large-neural-networks/>



Going beyond simple data-parallel

- If model does not fit on one GPU, there are three main solutions:
 - **Sharded data-parallelism**
 - **Pipelined model-parallelism**
 - **Tensor-parallelism**

3d-parallelism: can use all three!

BLOOM (GPT-3 sized LM)

| Component | DeepSpeed | Megatron-LM |
|------------------------------|-----------|-------------|
| <u>ZeRO Data Parallelism</u> | V | |
| <u>Tensor Parallelism</u> | V | |
| <u>Pipeline Parallelism</u> | V | |
| <u>BF16Optimizer</u> | V | |
| <u>Fused CUDA Kernels</u> | V | |
| <u>DataLoader</u> | V | |

ZeRO stage 3 can also be used to train models at this scale, however, it requires more communication than the DeepSpeed 3D parallel implementation. After careful evaluation in our environment which happened a year ago we found Megatron-DeepSpeed 3D parallelism performed best. Since then ZeRO stage 3 performance has dramatically improved and if we were to evaluate it today perhaps we would have chosen stage 3 instead.

BUT:

July 2022



Conclusion

- If model+data fits on one GPU, great.
- If not, or you want to speed up training, try `DistributedDataParallel`.
- If model still doesn't fit, try `ZeRO-3` / Fully-Sharded Data Parallel.

Other ways to speed up

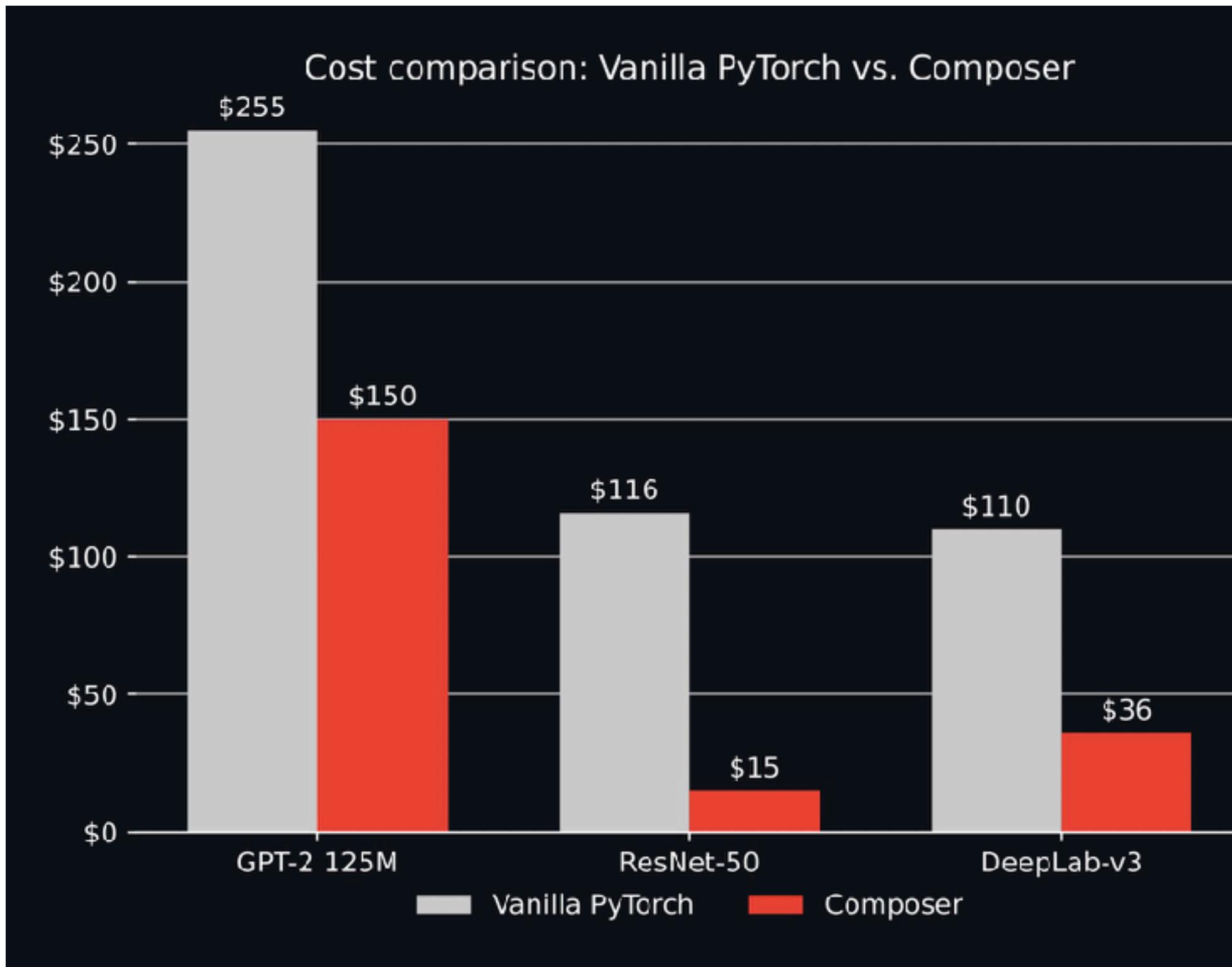
- Distributed Training with Mixed Precision
 - 16-bit mixed precision
 - Single-GPU/Multi-GPU/Multi-Node
- Ultra-fast dense transformer kernels
- Sparse attention
 - Memory- and compute-efficient sparse kernels
 - Support 10x long sequences than dense
 - Flexible support to different sparse structures
- 1-bit Adam, 0/1 Adam and 1-bit LAMB
 - Custom communication collective
 - Up to 26x communication volume saving
- Additional Memory and Bandwidth Optimizations
 - Smart Gradient Accumulation
 - Communication/Computation Overlap



Even More Training Tricks

- There are things that can improve convergence speed
- For NLP, can use Alibi instead of position encoding and train on shorter sequences, and use sequence length warmup.
- For vision, can use image size warmup, smooth labels, use special optimizer, and more.

MosaicML Composer



```
import composer.functional as cf
from torchvision import models

my_model = models.resnet18()

# add blurpool and squeeze excite layers
my_model = cf.apply_blurpool(my_model)
my_model = cf.apply_squeeze_excite(my_model)

# your own training code starts here
```

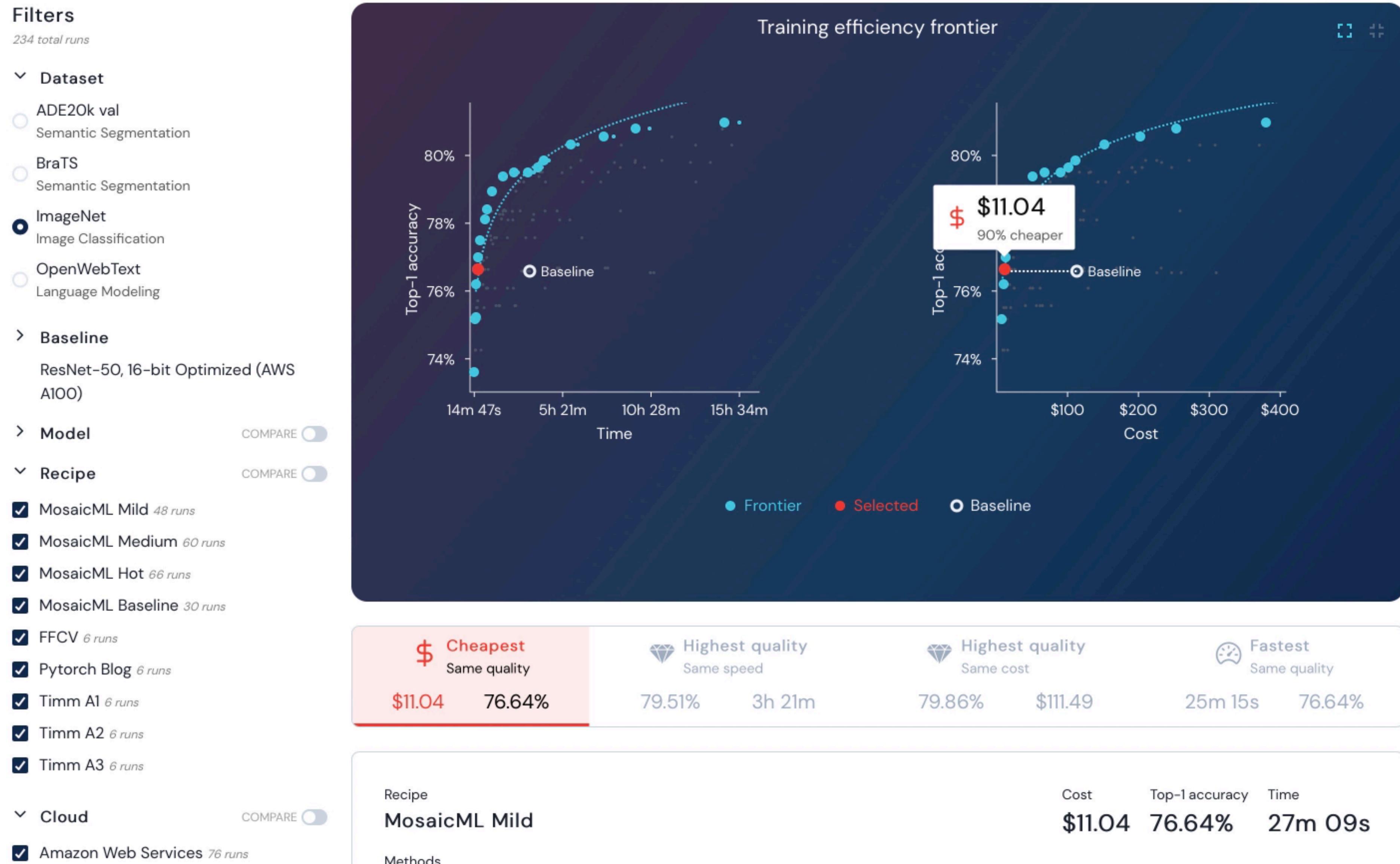
| Name | Attribution | tl;dr | Example Benchmark | Speed Up* |
|------------------|---------------------|---|-------------------|-----------|
| Alibi | Press et al, 2021 | Replace attention with Alibi. | GPT-2 | 1.5x |
| BlurPool | Zhang, 2019 | Applies an anti-aliasing filter before every downsampling operation. | ResNet-101 | 1.2x |
| ChannelsLast | PyTorch | Uses channels last memory format (NHWC). | ResNet-101 | 1.5x |
| CutOut | DeVries et al, 2017 | Randomly erases rectangular blocks from the image. | ResNet-101 | 1.2x |
| LabelSmoothing | Szegedy et al, 2015 | Smooths the labels with a uniform prior | ResNet-101 | 1.5x |
| MixUp | Zhang et al, 2017 | Blends pairs of examples and labels. | ResNet-101 | 1.5x |
| RandAugment | Cubuk et al, 2020 | Applies a series of random augmentations to each image. | ResNet-101 | 1.3x |
| SAM | Foret et al, 2021 | An optimization strategy that seeks flatter minima. | ResNet-101 | 1.4x |
| SeqLengthWarmup | Li et al, 2021 | Progressively increase sequence length. | GPT-2 | 1.2x |
| Stochastic Depth | Huang et al, 2016 | Replaces a specified layer with a stochastic version that randomly drops the layer or samples during training | ResNet-101 | 1.1x |

* = time-to-train to the same quality as the baseline.



MosaicML Explorer

<https://explorer.mosaicml.com>



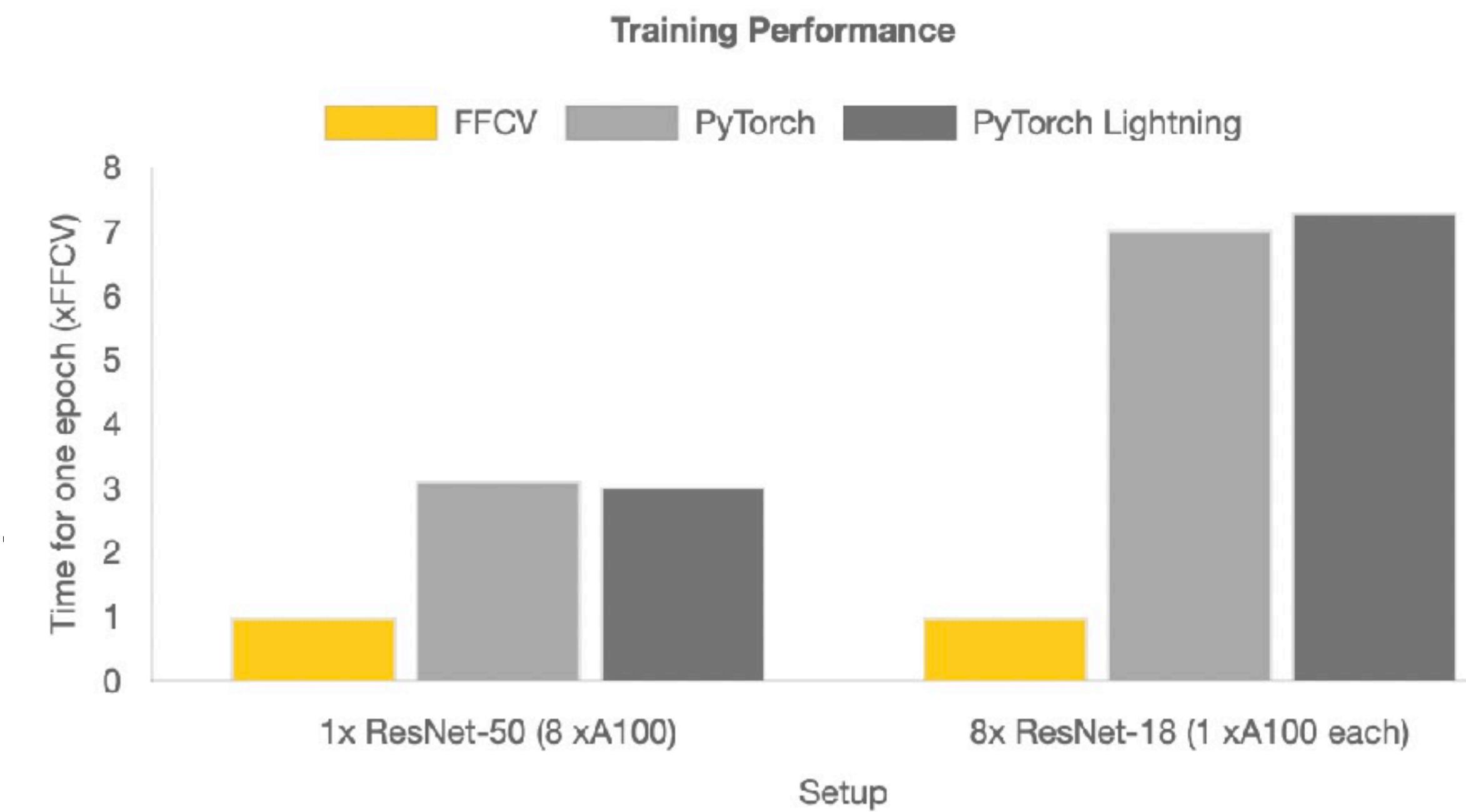
FFCV

```

train_loader = Loader('/pth/to/data.beton', batch_size=512,
    num_workers=8, order=OrderOption.RANDOM,
    pipelines={'image': [
        RandomResizedCropRGBImageDecoder((224, 224)),
        ToTensor(),
        # Move to GPU asynchronously as uint8:
        ToDevice(ch.device('cuda:0')),
        # Automatically channels-last:
        ToTorchImage(),
        Convert(ch.float16),
        # Standard torchvision transforms still work!
        tv.transforms.Normalize(MEAN, STDEV)
    ]})

# Prefetching, caching, move to GPU, all handled!
for ims, labs in train_loader:
    # Model training (FAST!)

```



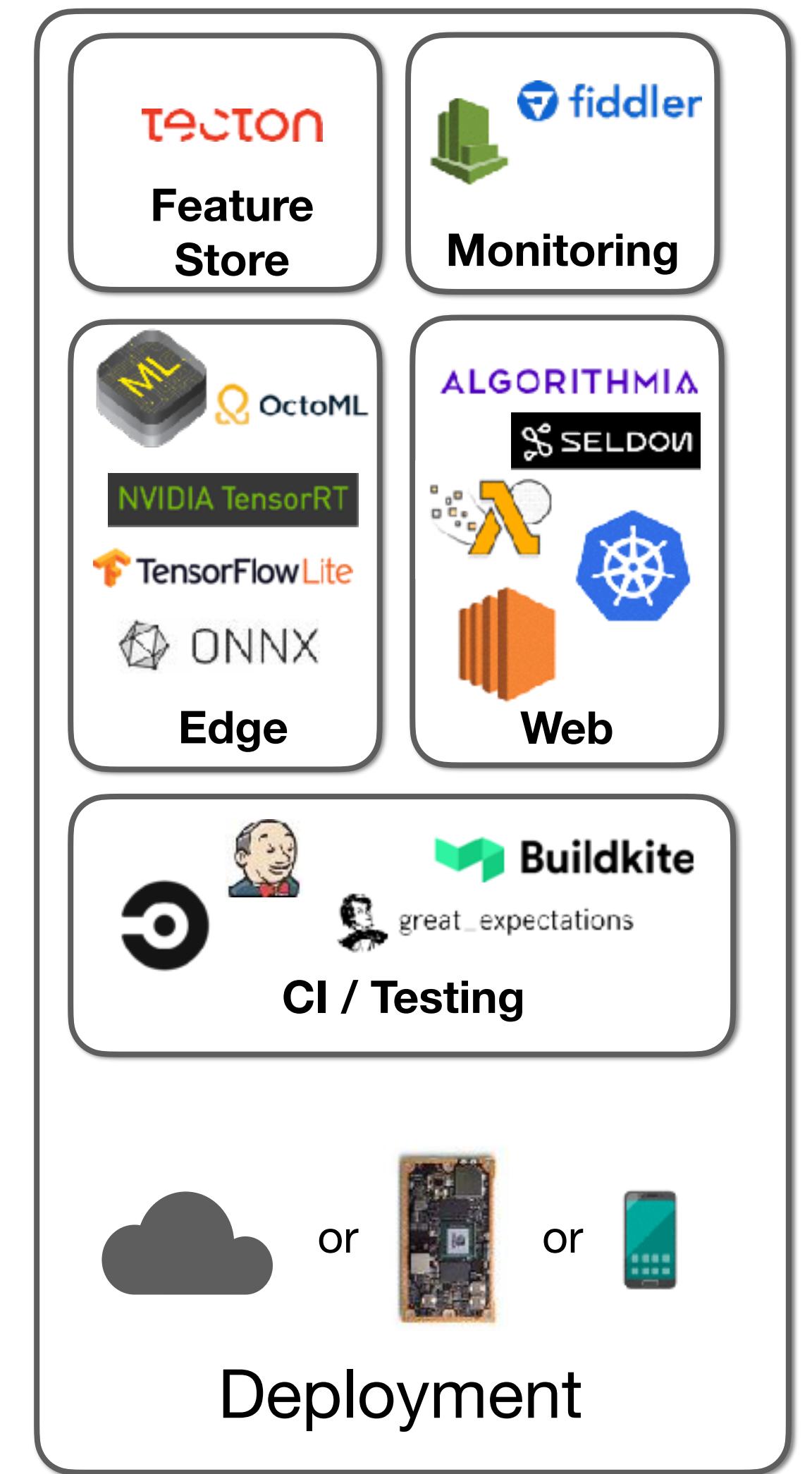
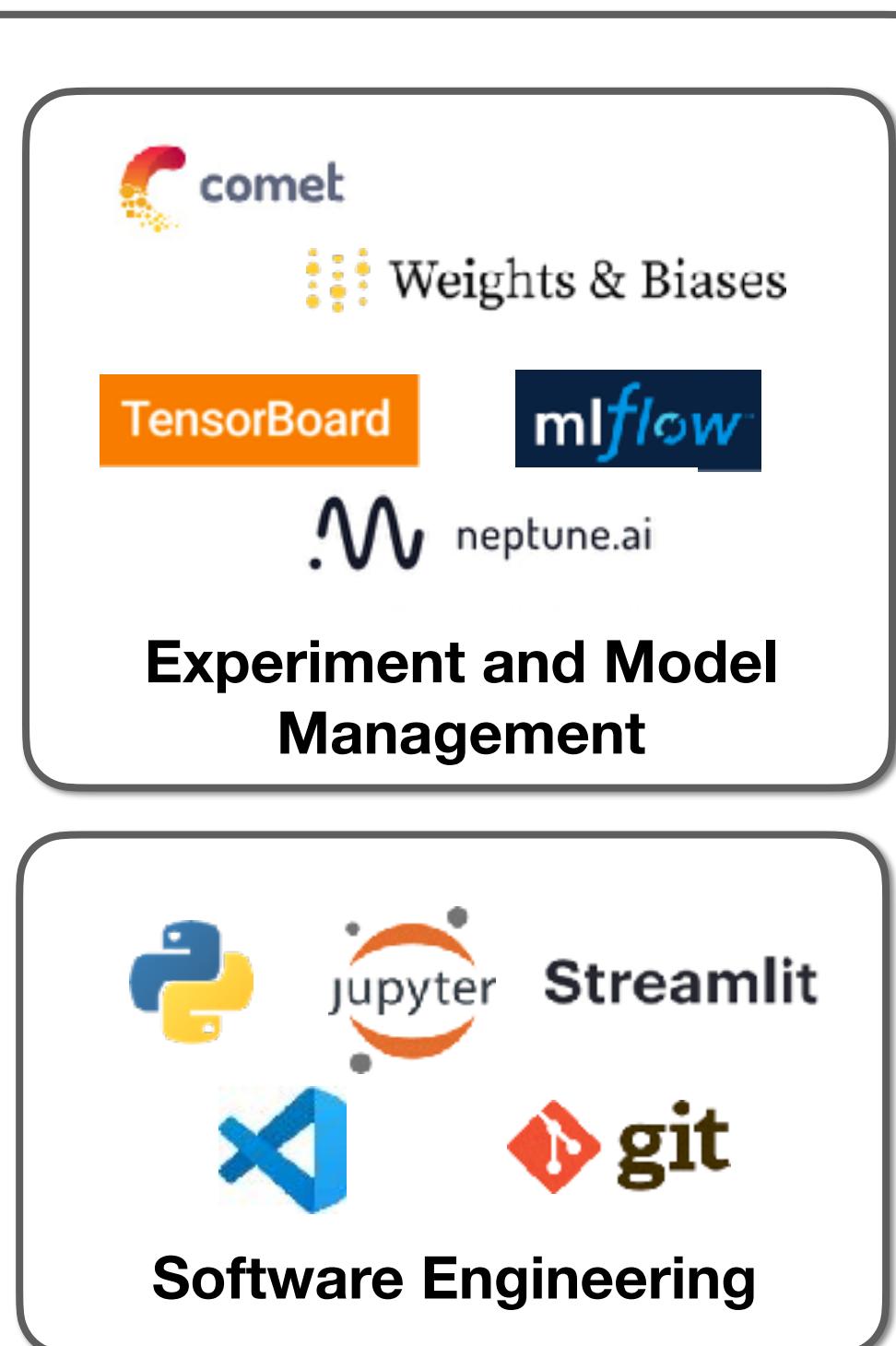
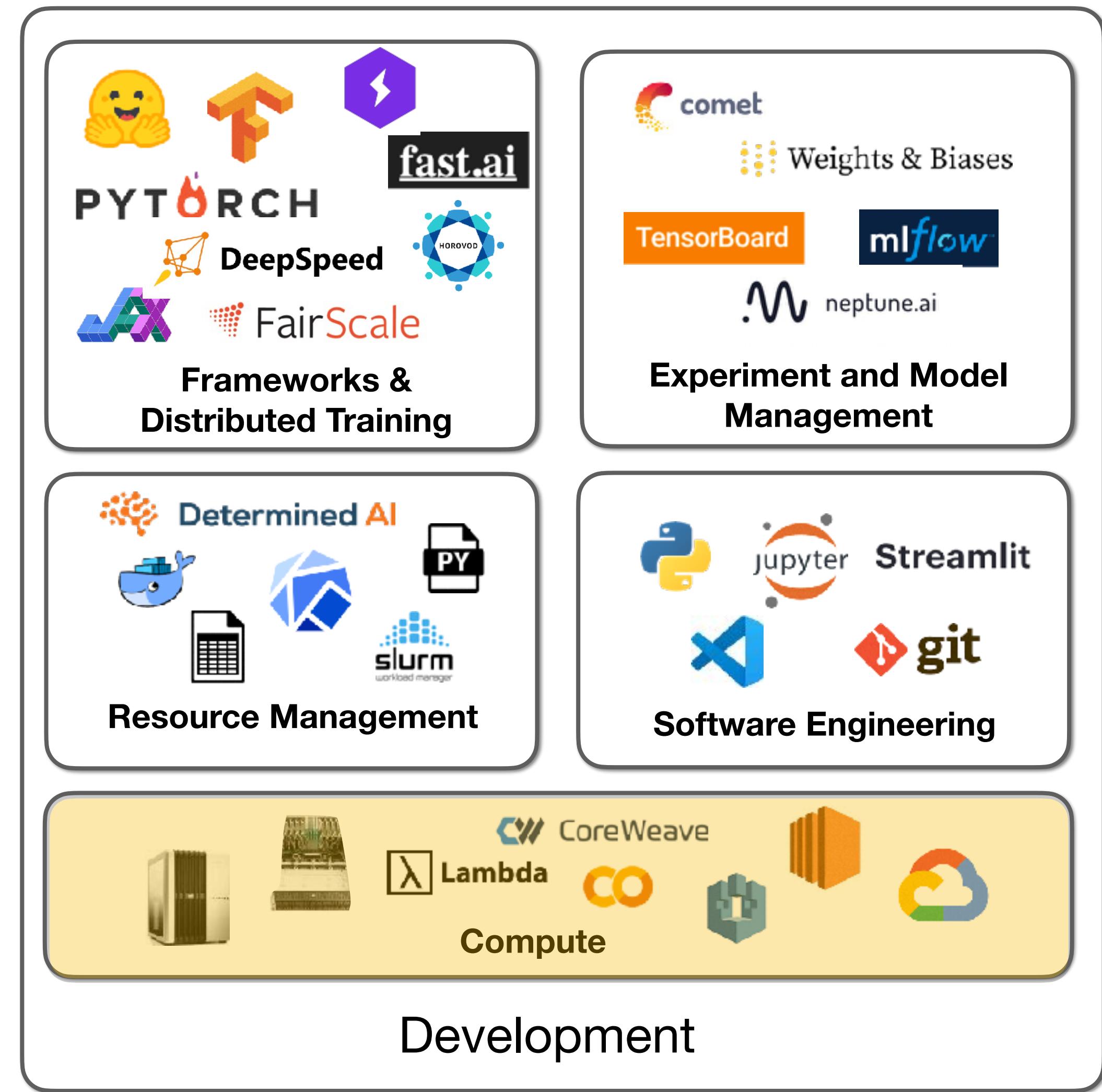
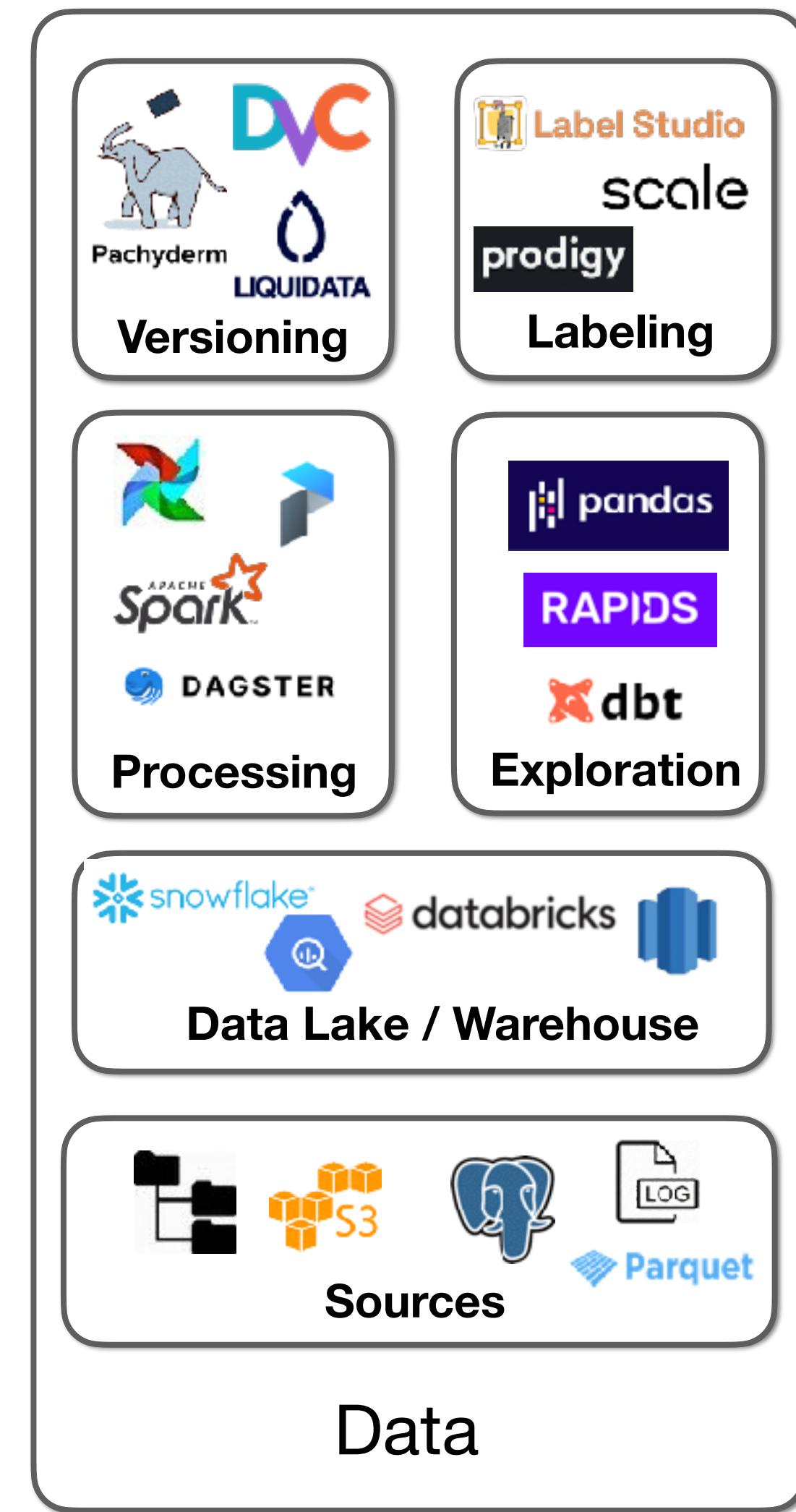
<https://ffcv.io>

“All-in-one”



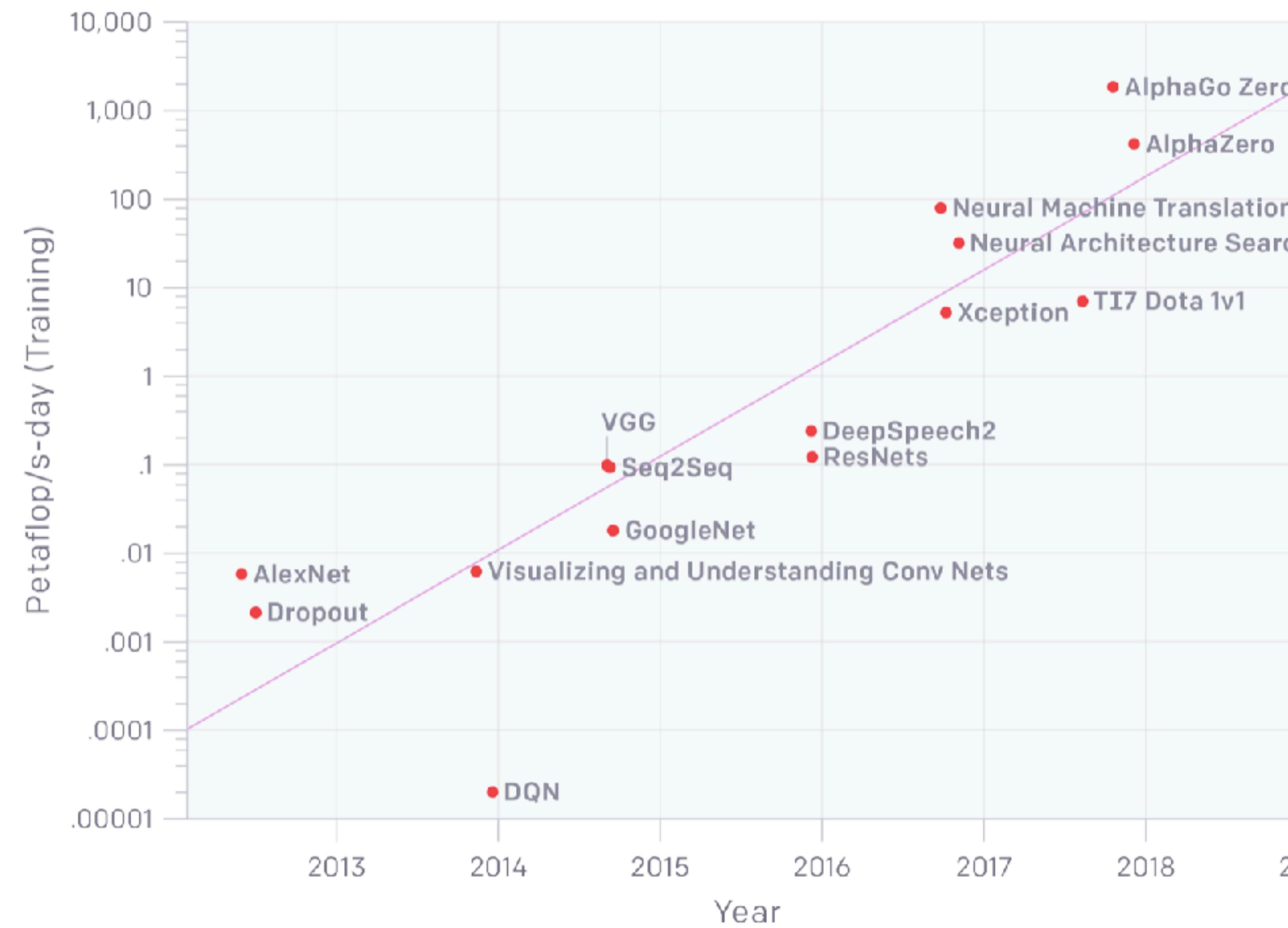
Amazon SageMaker
by Paperspace

gradient^o
by DOMINO
DATA LAB

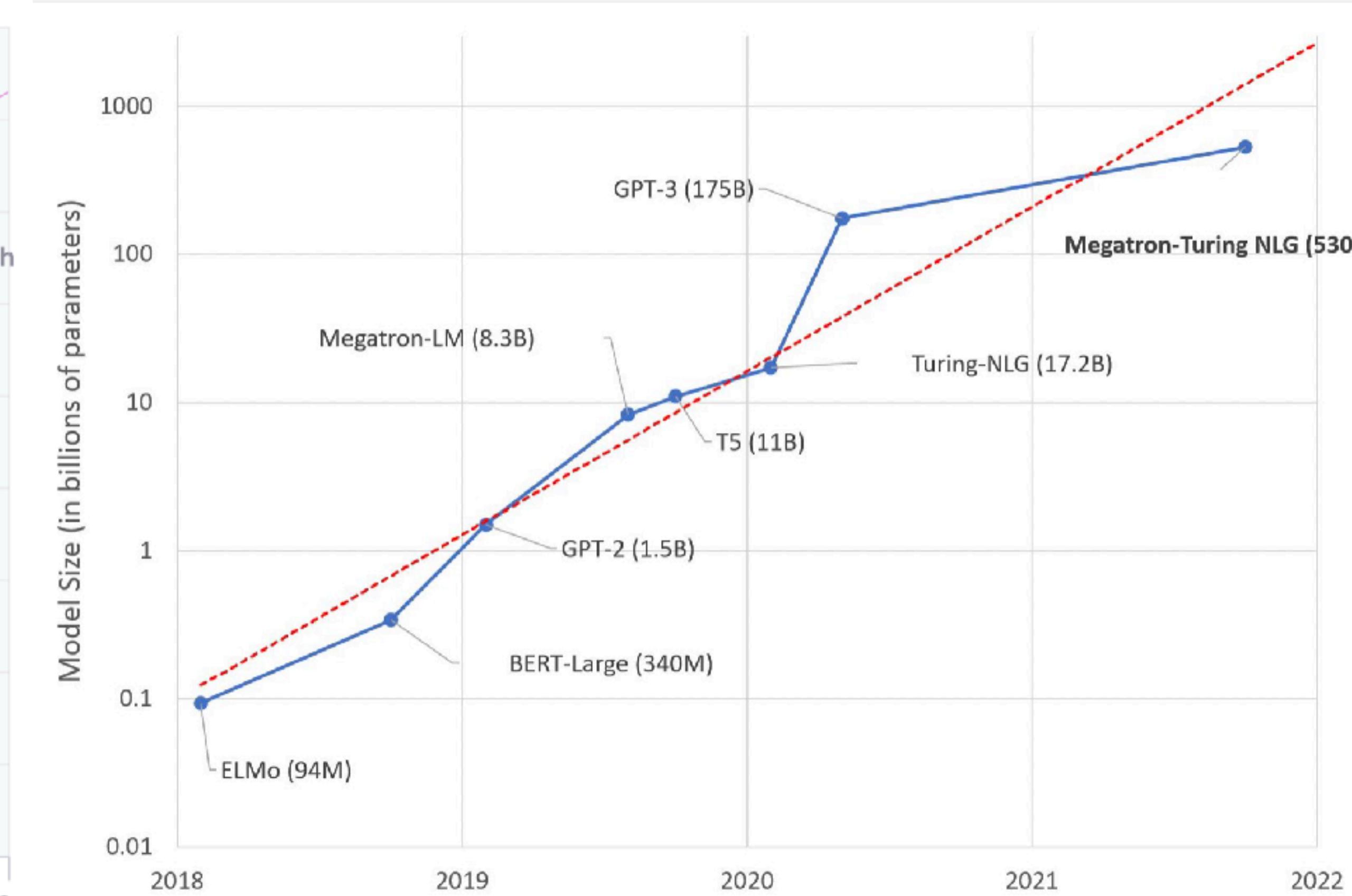


Why compute matters

AlexNet to AlphaGo Zero: A 300,000x Increase in Compute



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



<https://huggingface.co/blog/large-language-models>



GPU Basics

- NVIDIA has been the only choice for a while
- Google TPUs are also very nice (only in the GCP cloud)
- Three main factors:
 - Amount of data that fits on the GPU
 - Speed of crunching through data on GPU (different speeds for 32bit data and 16bit data)
 - Speed of communicating between CPU and GPU, and between GPUs



GPU Comparison

| Card | Release | Arch | Use-case | RAM (Gb) | 32bit TFlops | Tensor TFlops | 16bit |
|-------------|---------|--------|----------|----------|--------------|---------------|-------|
| K80 | 2014 | Kepler | Server | 24 | 5 | N/A | No |
| P100 | 2016 | Pascal | Server | 16 | 10 | N/A | Yes |
| V100 | 2017 | Volta | Server | 16 or 32 | 14 | 120 | Yes |
| RTX 2080 Ti | 2018 | Turing | PC | 11 | 13 | 107 | Yes |
| RTX 3090 | 2021 | Ampere | PC | 24 | 35 | 285 | Yes |
| A100 | 2020 | Ampere | Server | 40 or 80 | 19.5 | 312 | Yes |
| A6000 | 2022 | Ampere | Server | 48 | 38 | ? | Yes |



GPU Comparison

| Card | Release | Arch | Use-case | RAM (Gb) | 32bit TFlops | Tensor TFlops | 16bit |
|-------------|---------|--------|----------|----------|--------------|---------------|-------|
| K80 | 2014 | Kepler | Server | 24 | 5 | N/A | No |
| P100 | 2016 | Pascal | Server | 16 | 10 | N/A | Yes |
| V100 | 2017 | Volta | Server | 16 or 32 | 14 | 120 | Yes |
| RTX 2080 Ti | 2018 | Turing | PC | 11 | 13 | 107 | Yes |
| RTX 3090 | 2021 | Ampere | PC | 24 | 35 | 285 | Yes |
| A100 | 2020 | Ampere | Server | 40 or 80 | 19.5 | 312 | Yes |
| A6000 | 2022 | Ampere | Server | 48 | 38 | ? | Yes |

- New NVIDIA architecture every year
(Kepler → Pascal → Volta -> Turing -> Ampere)
- For business, only supposed to use server cards.



GPU Comparison

| Card | Release | Arch | Use-case | RAM (Gb) | 32bit TFlops | Tensor TFlops | 16bit |
|-------------|---------|--------|----------|----------|--------------|---------------|-------|
| K80 | 2014 | Kepler | Server | 24 | 5 | N/A | No |
| P100 | 2016 | Pascal | Server | 16 | 10 | N/A | Yes |
| V100 | 2017 | Volta | Server | 16 or 32 | 14 | 120 | Yes |
| RTX 2080 Ti | 2018 | Turing | PC | 11 | 13 | 107 | Yes |
| RTX 3090 | 2021 | Ampere | PC | 24 | 35 | 285 | Yes |
| A100 | 2020 | Ampere | Server | 40 or 80 | 19.5 | 312 | Yes |
| A6000 | 2022 | Ampere | Server | 48 | 38 | ? | Yes |

- **RAM:** should fit your model + meaningful batch of data
- **32bit vs Tensor TFlops**
 - Tensor Cores are specifically for deep learning operations (mixed precision)
- **16bit** ~doubles your effective RAM capacity

Benchmarks

<https://lambdalabs.com/gpu-benchmarks>

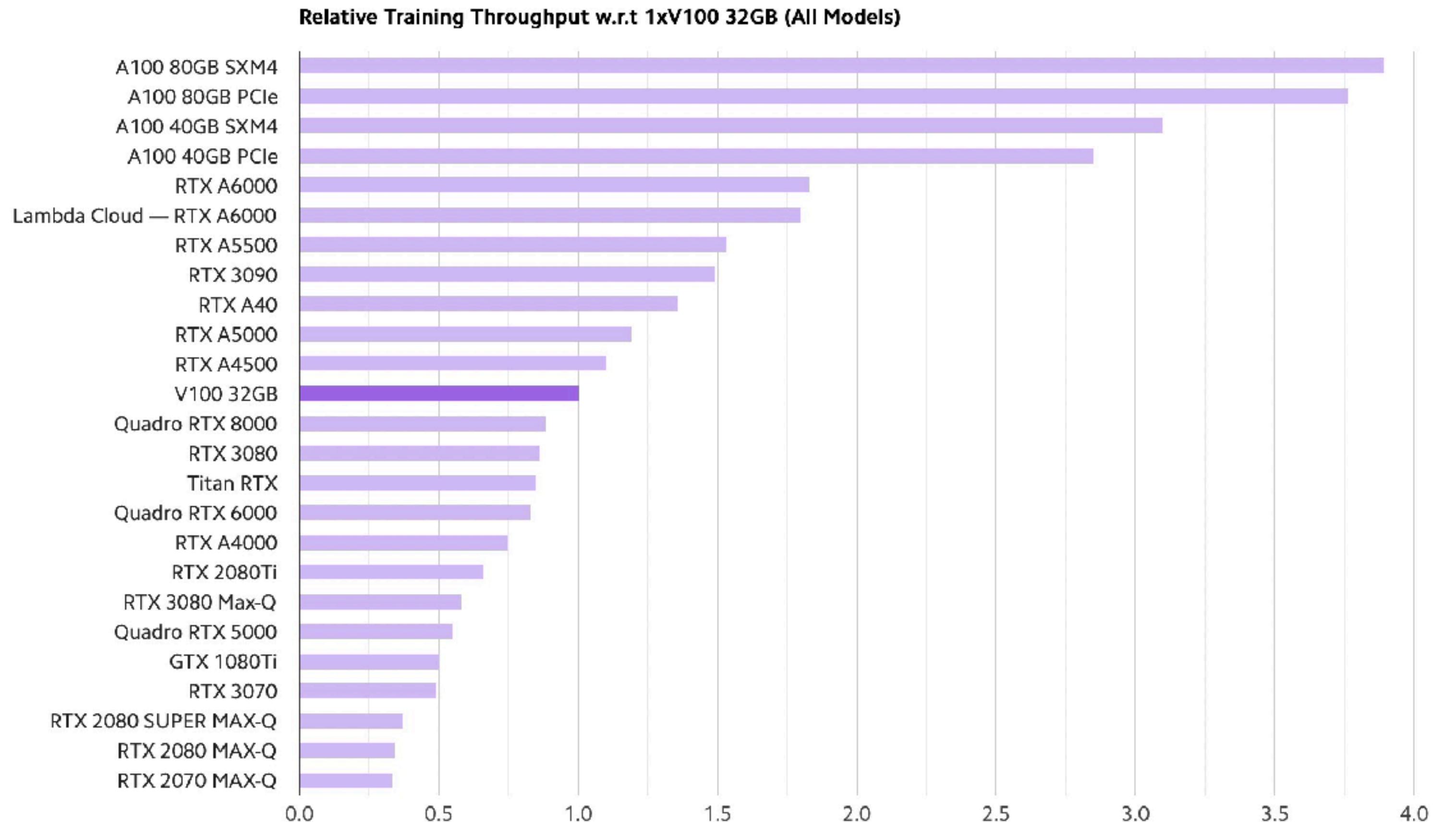


GPU Cloud Colocation Clusters Servers Workstations Laptops Resources Careers

+1 (866) 711-2025

PyTorch Training GPU Benchmarks

Visualization: chart | Metric: throughput | Precision: fp32/tf32 | Number of GPUs: 1x | Model: All Models



- A100 is >2.5x faster than V100
- RTX A{4/5/6}000's are great

Benchmarks



- Time for ResNet-50 to go through 1.4M images
- 4xA100 vs 4xV100:
 - 3x faster in fp32
 - 1.5x faster in fp16

| Configuration | float 32 training | float 16 training |
|--------------------|-------------------|-------------------|
| CPU(32 cores) | 27 hours | 27 hours |
| Single RTX 2080 TI | 69 minutes | 29 minutes |
| Single RTX 3080 | 53 minutes | 22 minutes |
| Single RTX 3080 TI | 45 minutes | 21 minutes |
| Single RTX 3090 | 41 minutes | 18 minutes |
| Single RTX A6000 | 41 minutes | 16 minutes |
| Single NVIDIA A100 | 23 minutes | 8.5 minutes |
| 4 x RTX 2080TI | 19 minutes | 8 minutes |
| 4 x Tesla V100 | 15 minutes | 4.5 minutes |
| 4 x RTX 3090 | 11.5 minutes | 5 minutes |
| 4 x NVIDIA A100 | 6.5 minutes | 3 minutes |

<https://www.aime.info/en/blog/deep-learning-gpu-benchmarks-2021/>



Cloud Providers

- Amazon Web Services, Google Cloud Platform, Microsoft Azure are the heavyweights.
 - GCP is special because it also has TPU's
- Startups are Lambda Labs, Paperspace, Coreweave, and others

TPUs

- TPU v4 are fastest, but not quite in GA yet
- TPU v3 are quite fast and excel at scaling (fast interconnect)

| | TFLOPS (bf16) | Cost (Google Cloud) | TFLOPS / \$ |
|-----------------------|---------------|---------------------|-------------|
| A100 + a2-1g instance | 312 | \$2.90 + \$1.01 | 79.8 |
| TPU v2 - 8 cores | 180* | \$4.50 | 40.0 |
| TPU v3 - 8 cores | 420* | \$8.00 | 52.5 |

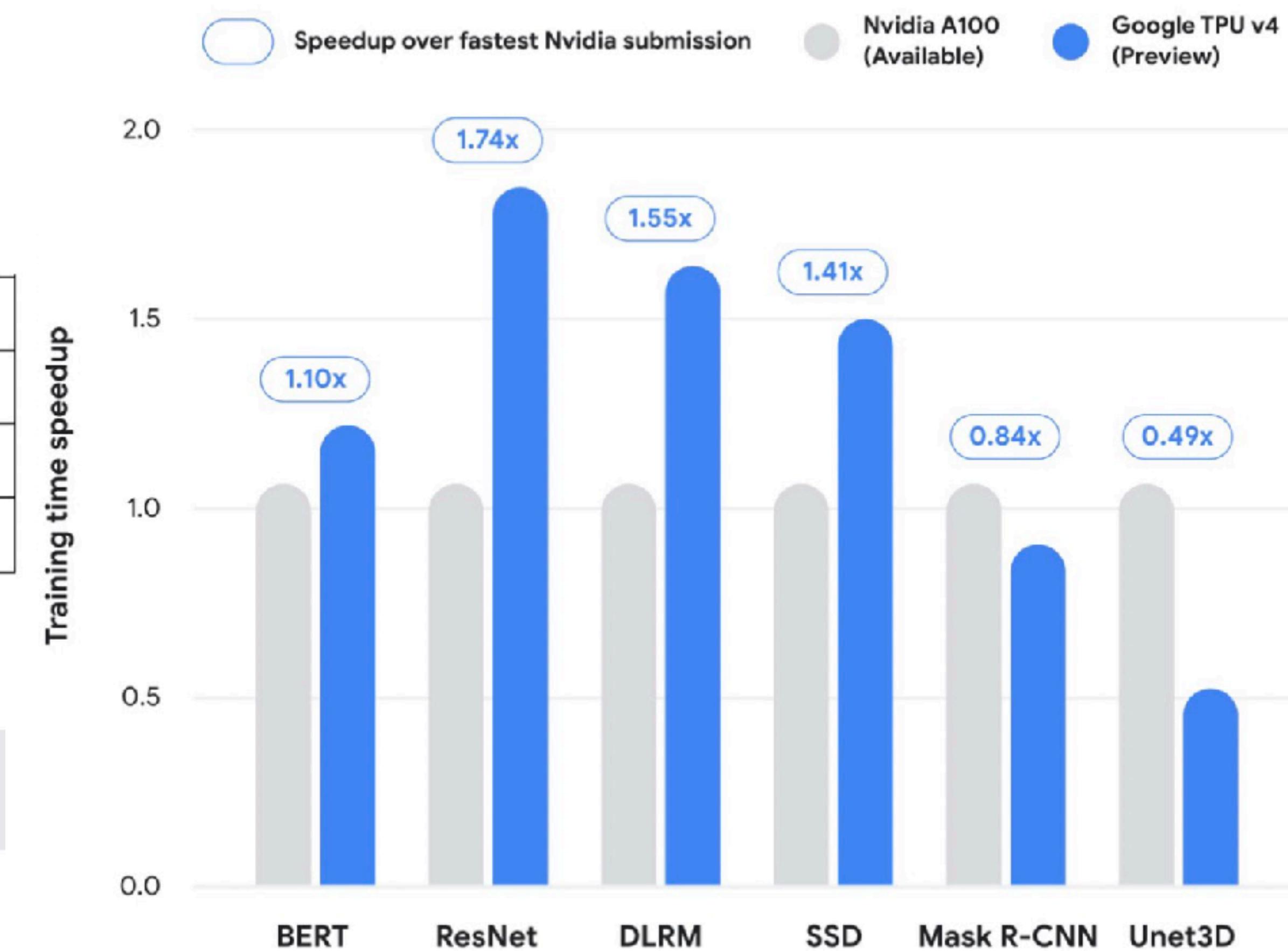
<https://eng.snap.com/training-models-with-tpus>

| TPU type (v3) | TPU v3 cores | Total TPU memory |
|---------------|--------------|------------------|
| v3-8 | 8 | 128 GiB |

<https://cloud.google.com/tpu/docs/tpus>

Comparison of MLPerf 1.0 Top Line Results

Taller bars are better; results are normalized to fastest Nvidia submission





FSDL GPU Cloud Comparison

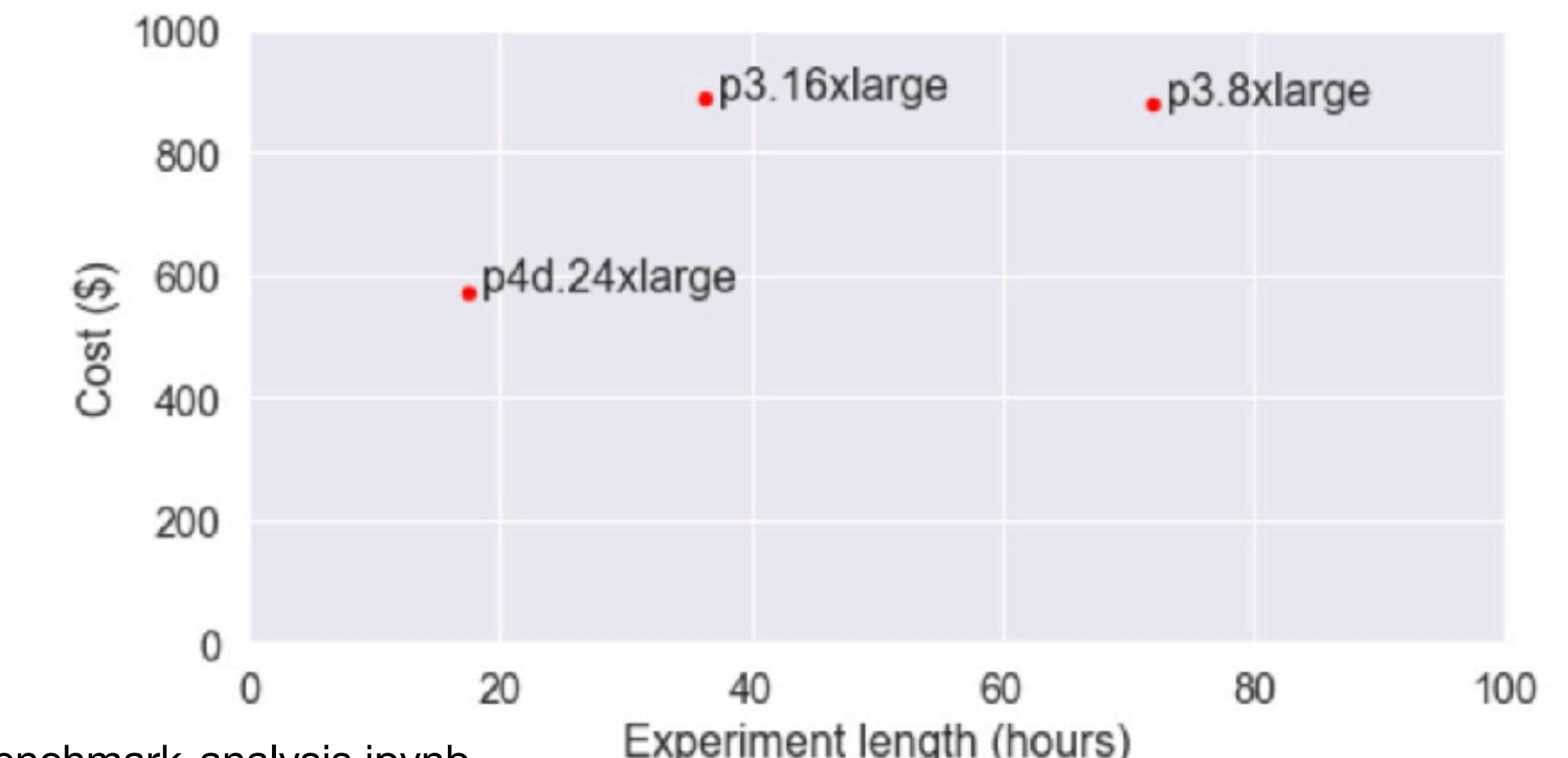
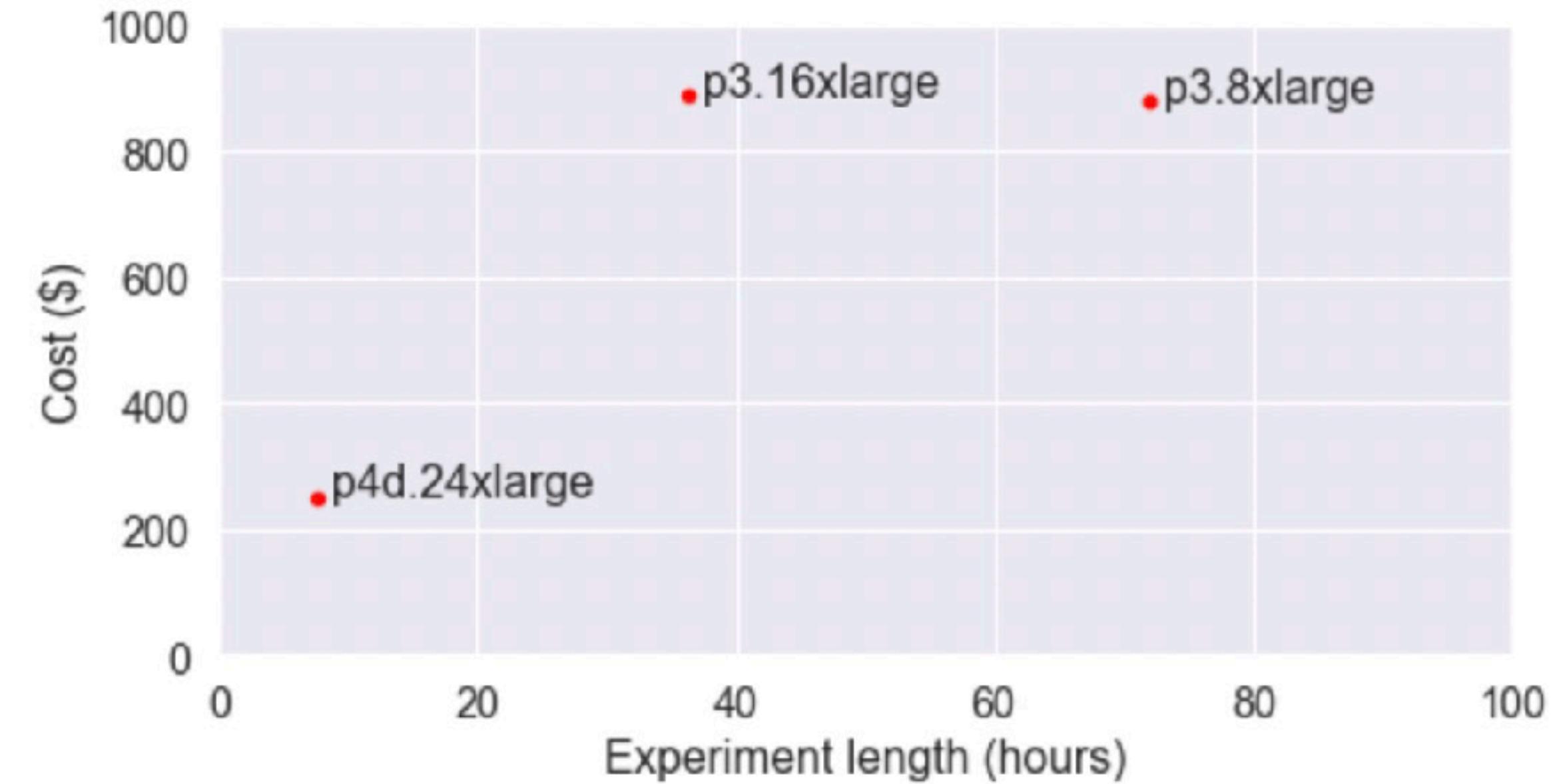
| Cloud | GPU Type | GPU Arch | GPUs | GPU RAM | vCPUs | RAM | On-demand | Per-GPU | Spot | Name |
|-------|--------------|----------|--|--|-------|---------|-----------|---------|----------------|------|
| AWS | A100 (80 GB) | Ampere | Insert column left Insert column right Remove column Clear column ✓ Read only Alignment ► Filter by condition: None ▾ Filter by value: Search Select all Clear | 96 96 8 32 64 96 16 48 192 16 | 1152 | \$40.97 | \$5.12 | | p4de.24xlarge | |
| AWS | A100 (40 GB) | Ampere | | | 1152 | \$32.77 | \$4.10 | \$9.83 | p4d.24xlarge | |
| AWS | V100 (16 GB) | Volta | | | 61 | \$3.06 | \$3.06 | \$0.92 | p3.2xlarge | |
| AWS | V100 (16 GB) | Volta | | | 244 | \$12.24 | \$3.06 | \$3.67 | p3.8xlarge | |
| AWS | V100 (16 GB) | Volta | | | 488 | \$24.48 | \$3.06 | \$7.34 | p3.16xlarge | |
| AWS | V100 (32 GB) | Volta | | | 768 | \$31.21 | \$3.90 | \$9.36 | p3dn.24xlarge | |
| AWS | A10G (24 GB) | Ampere | | | 64 | \$1.62 | \$1.62 | \$0.49 | g5.4xlarge | |
| AWS | A10G (24 GB) | Ampere | | | 192 | \$5.67 | \$1.42 | \$1.74 | g5.12xlarge | |
| AWS | A10G (24 GB) | Ampere | | | 16 | \$12.29 | \$2.04 | \$4.89 | g5.48xlarge | |
| AWS | T4 (16 GB) | Turing | | | 61 | \$1.20 | \$1.20 | \$0.36 | g4dn.4xlarge | |
| AWS | K80 (12 GB) | Kepler | | | 4 | \$0.90 | \$0.90 | \$0.27 | p2.xlarge | |
| AWS | K80 (12 GB) | Kepler | Ampere Kepler Pascal Turing | 32 64 12 24 48 96 96 8 | 488 | \$7.20 | \$0.90 | \$2.16 | p2.8xlarge | |
| AWS | K80 (12 GB) | Kepler | | | 732 | \$14.40 | \$0.90 | \$4.32 | p2.16xlarge | |
| GCP | A100 (40 GB) | Ampere | | | 12 | \$3.67 | \$3.67 | \$1.10 | a2-highgpu-1g | |
| GCP | A100 (40 GB) | Ampere | | | 24 | \$7.34 | \$3.67 | \$2.20 | a2-highgpu-2g | |
| GCP | A100 (40 GB) | Ampere | | | 48 | \$14.68 | \$3.67 | \$4.41 | a2-highgpu-4g | |
| GCP | A100 (40 GB) | Ampere | | | 96 | \$29.36 | \$3.67 | \$8.81 | a2-highgpu-8g | |
| GCP | A100 (40 GB) | Ampere | | | 1360 | \$55.68 | \$3.48 | \$16.72 | a2-megagpu-16g | |
| GCP | V100 (16 GB) | Volta | | | 52 | \$2.95 | \$2.95 | \$0.84 | n1-highmem-8 | |
| GCP | V100 (16 GB) | Volta | Volta Pascal Pascal Kepler | 32 64 128 16 | 16 | \$5.91 | \$2.95 | \$1.68 | n1-highmem-16 | |
| GCP | V100 (16 GB) | Volta | | | 32 | \$11.81 | \$2.95 | \$3.36 | n1-highmem-32 | |
| GCP | V100 (16 GB) | Volta | | | 64 | \$23.63 | \$2.95 | \$6.72 | n1-highmem-64 | |
| GCP | P100 (16 GB) | Pascal | | | 128 | \$416 | \$1.93 | \$0.53 | n1-highmem-8 | |
| GCP | P100 (16 GB) | Pascal | Pascal Pascal Kepler | 16 32 64 | 8 | \$1.93 | \$1.93 | \$1.06 | n1-highmem-16 | |
| GCP | P100 (16 GB) | Pascal | | | 16 | \$3.87 | \$1.93 | \$2.12 | n1-highmem-32 | |
| GCP | P100 (16 GB) | Pascal | | | 32 | \$7.73 | \$1.93 | \$0.14 | n1-highmem-8 | |
| GCP | K80 (12 GB) | Kepler | | | 64 | \$0.92 | \$0.92 | | | |



Let's combine cost with benchmark data

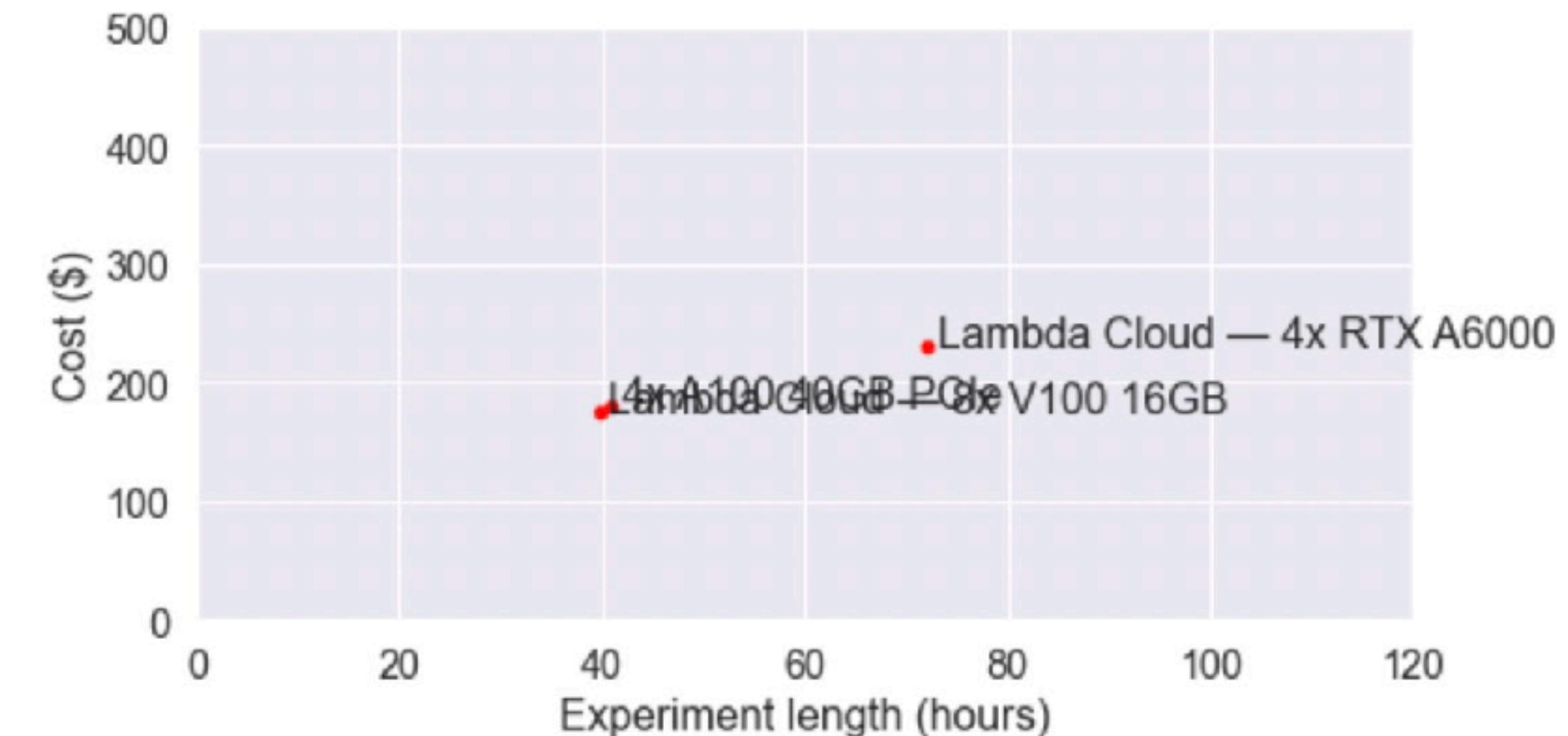
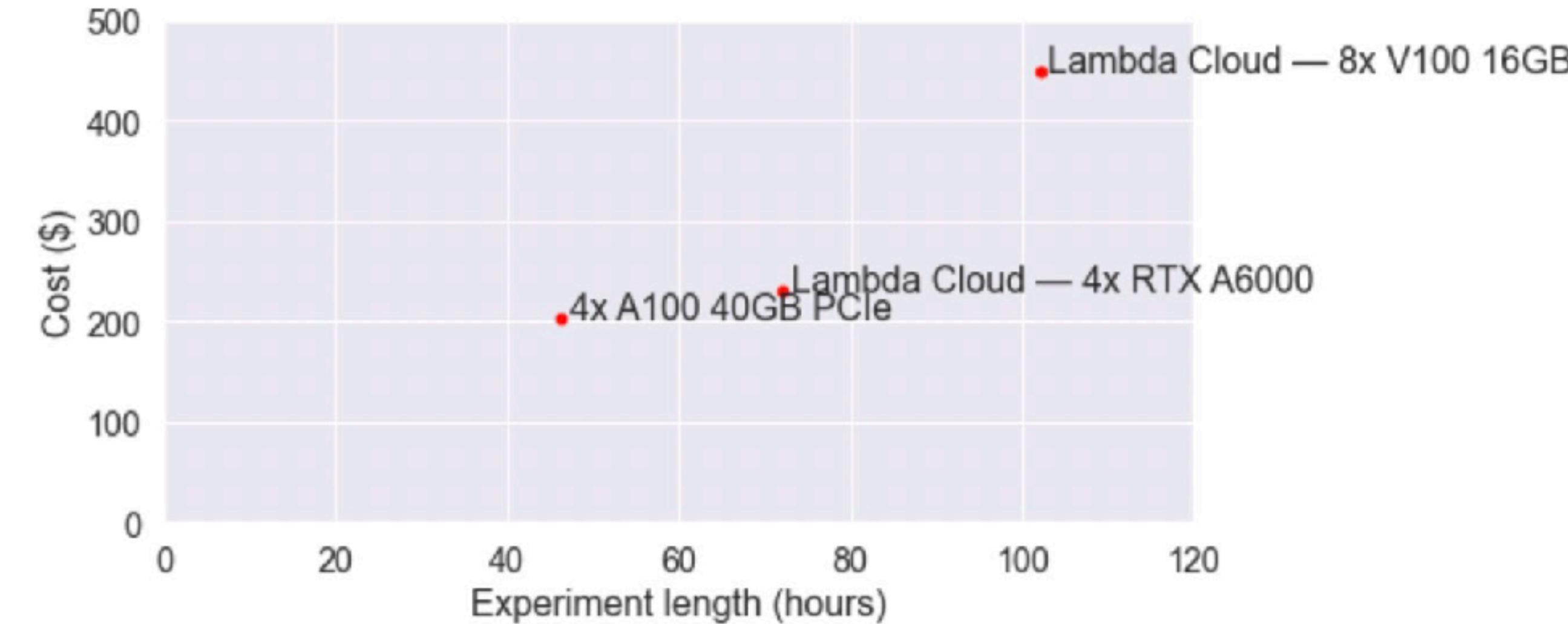
Expensive per-hour ≠ Expensive per-experiment

- A Transformers experiment that takes 72 hours on 4xV100 (p3.8xlarge) will cost \$1750 to run.
- **But it will take only 8 hours and cost only \$250 on 8xA100 (p4d.24xlarge)!**
- Similar story for CNN's.



Expensive per-hour ≠ Expensive per-experiment

- On Lambda Labs cloud, story is similar
- 8xV100 takes **2x** longer and is **2x** more expensive (Transformers)
- Similar story for CNN's.



Expensive per-hour ≠ Expensive per-experiment

- MosaicML benchmarking
 - ResNet-50
- 8xA100 is **1.5x** faster and **15%** cheaper than 8xV100

| Recipe | Cost | Top-1 accuracy | Time |
|-------------------|---------------------|----------------|----------------------|
| MosaicML Baseline | \$113.09 | 76.59% | 3h 27m |
| Model | Cloud | Instance | Scale Schedule Ratio |
| ResNet-50 | Amazon Web Services | p4d.24xlarge | 1.00 |

| Recipe | Cost | Top-1 accuracy | Time |
|-------------------|---------------------|----------------|----------------------|
| MosaicML Baseline | \$130.29 | 76.59% | 5h 19m |
| Model | Cloud | Instance | Scale Schedule Ratio |
| ResNet-50 | Amazon Web Services | p3.16xlarge | 1.00 |

Expensive per-hour ≠ Expensive per-experiment

- MosaicML benchmarking GPT-2 (125M)
- 8xA100 is **~2x** faster and **25%** cheaper than 8xV100
 - And **~3x** faster and **30%** cheaper than 8xT4

| Methods | Model | Cloud | Instance | Cost | Perplexity | Time |
|-----------|--------------|---------------------|--------------|----------|------------|---------|
| 1 Methods | GPT-2 (125M) | Amazon Web Services | p4d.24xlarge | \$255.18 | 24.11 | 7h 47m |
| Methods | Model | Cloud | Instance | Cost | Perplexity | Time |
| 1 Methods | GPT-2 (125M) | Amazon Web Services | p3.16xlarge | \$337.17 | 24.11 | 13h 46m |
| Methods | Model | Cloud | Instance | Cost | Perplexity | Time |
| 1 Methods | GPT-2 (125M) | Amazon Web Services | g4dn.metal | \$352.18 | 24.11 | 1d 21h |

Conclusions

- Good heuristic: use the most expensive per-hour GPUs (4x or 8x A100's) in the least-expensive cloud
- The startups are **much** cheaper than the big boys

| Cloud | GPU Type | GPU Arch | Gpus | GPU RAM | vCPUs | RAM | On-demand | Per-GPU ↑ | Spot | Name |
|------------|--------------|----------|------|---------|-------|------|-----------|-----------|------|------------------|
| Lambda | A100 (40 GB) | Ampere | 4 | 160 | 120 | 800 | 4.40 | 1.10 | | |
| Jarvislabs | A100 (40 GB) | Ampere | 4 | 160 | 28 | 128 | 4.4 | 1.1 | 2.76 | |
| Jarvislabs | A100 (40 GB) | Ampere | 8 | 320 | 56 | 256 | 8.8 | 1.1 | 5.52 | |
| Datacrunch | A100 (80 GB) | Ampere | 4 | 320 | 88 | 480 | 9.00 | 2.25 | | 4A100.88V |
| Datacrunch | A100 (80 GB) | Ampere | 8 | 640 | 176 | 960 | 18.00 | 2.25 | | 8A100.176V |
| Paperspace | A100 (40 GB) | Ampere | 4 | 160 | 48 | 360 | 12.36 | 3.09 | | |
| Paperspace | A100 (40 GB) | Ampere | 8 | 320 | 96 | 720 | 24.72 | 3.09 | | |
| Paperspace | A100 (80 GB) | Ampere | 4 | 320 | 48 | 360 | 12.72 | 3.18 | | |
| Paperspace | A100 (80 GB) | Ampere | 8 | 640 | 96 | 720 | 25.44 | 3.18 | | |
| Azure | A100 (80 GB) | Ampere | 4 | 320 | 96 | 880 | 14.69 | 3.67 | 5.88 | NC96ads A100 v4 |
| GCP | A100 (40 GB) | Ampere | 4 | 160 | 48 | 340 | 14.68 | 3.67 | 4.41 | a2-highgpu-4g |
| GCP | A100 (40 GB) | Ampere | 8 | 320 | 96 | 680 | 29.36 | 3.67 | 8.81 | a2-highgpu-8g |
| AWS | A100 (40 GB) | Ampere | 8 | 320 | 96 | 1152 | 32.77 | 4.10 | 9.83 | p4d.24xlarge |
| Azure | A100 (80 GB) | Ampere | 8 | 640 | 96 | 1900 | 37.18 | 4.64 | | ND96amsr A100 v4 |
| AWS | A100 (80 GB) | Ampere | 8 | 640 | 96 | 1152 | 40.97 | 5.12 | | p4de.24xlarge |



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

- Decently quiet PC with 128GB RAM and 2x RTX 3090's: ~\$7000
 - One day to build and set up
 - Rite of passage?
- Going beyond 4 2000-series or 2 3000-series can be painful

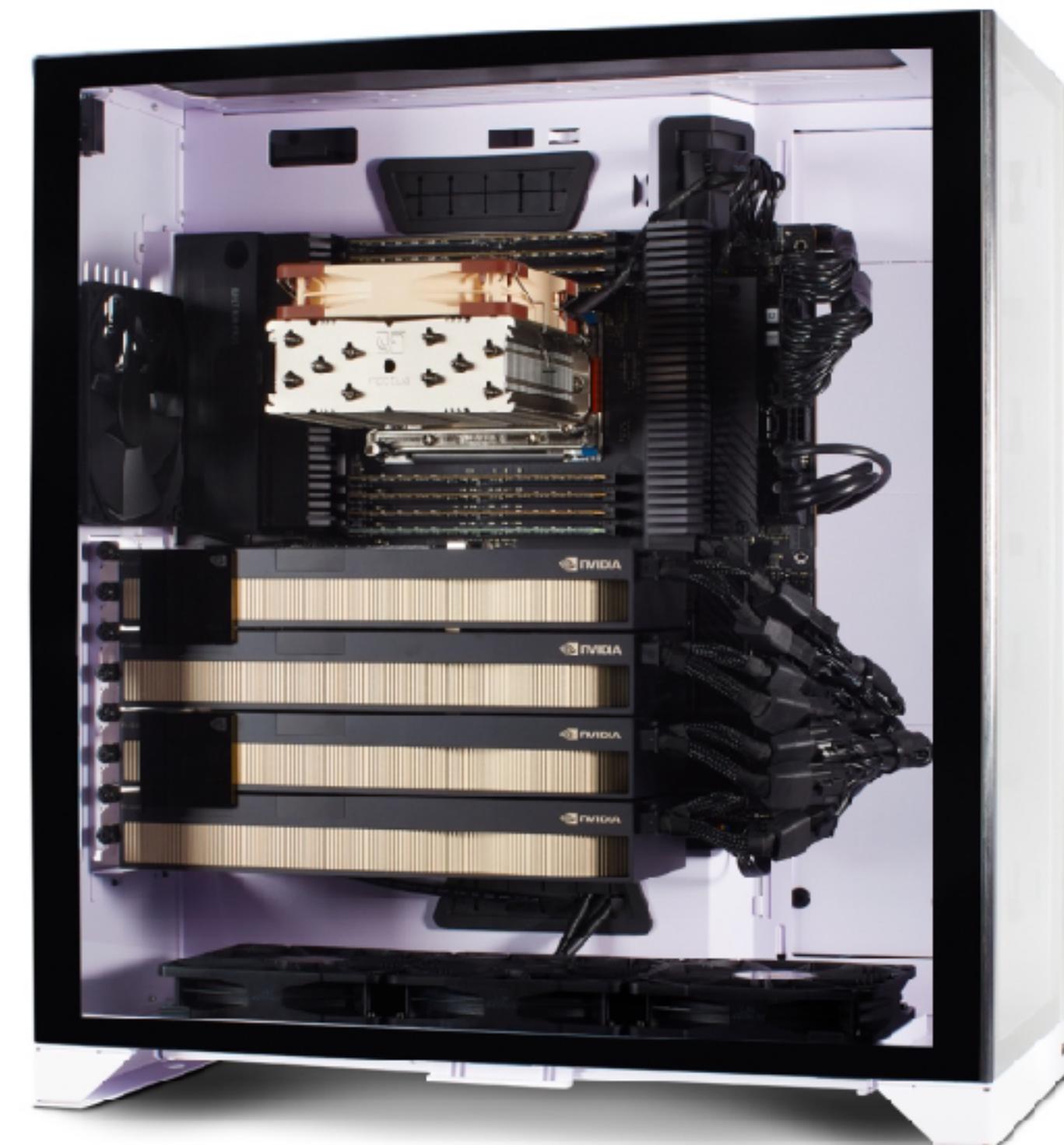




Pre-built: Lambda Labs

Lambda

Go back



Ubuntu 20.04
Lambda Stack pre-installed

Threadripper Pro 3955WX
16 cores, 3.90 GHz

2x RTX A5000
NVLinked, 24 GB VRAM each

128 GB
System memory

1 TB NVMe
OS Drive

0 TB
Extra storage

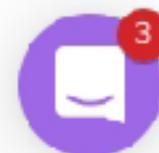
\$12,034.00

Apply for a discount

| | |
|---------------------------------|-----------|
| 2x RTX 3080 | -\$4,347 |
| 2x RTX 3080 Ti | -\$1,909 |
| 2x RTX 3090 | -\$1,909 |
| 2x RTX A4000 | -\$4,022 |
| 2x RTX A4500 with NVLink | -\$2,112 |
| 2x RTX A5000 with NVLink | |
| 2x RTX A5500 with NVLink | -\$3,738 |
| 2x RTX A6000 with NVLink | -\$6,663 |
| 3x RTX 3080 | -\$2,762 |
| 3x RTX 3090 with NVLink | -\$1,178 |
| 3x RTX A4000 | -\$2,640 |
| 3x RTX A4500 with NVLink | +\$447 |
| 3x RTX A5000 with NVLink | -\$3,616 |
| 3x RTX A5500 with NVLink | -\$9,222 |
| 3x RTX A6000 with NVLink | +\$13,610 |
| 4x RTX A4000 | -\$893 |

Download quote

Add to cart





Pre-built: Lambda Labs

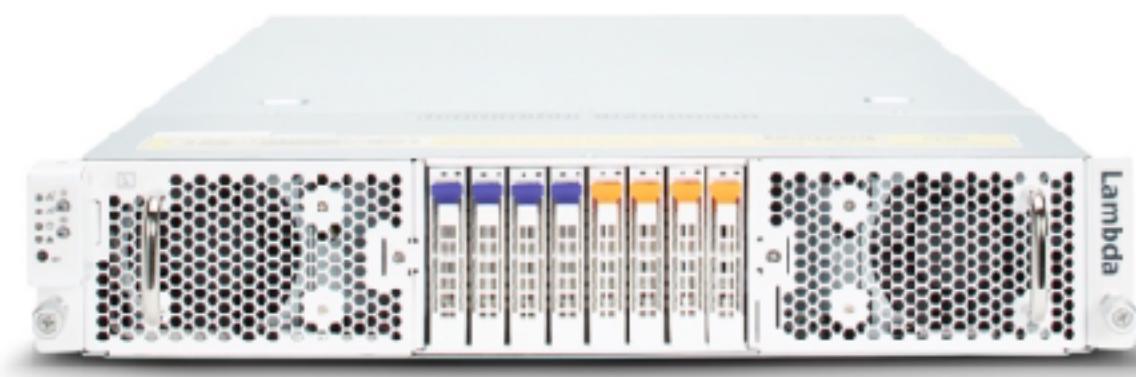
λ Lambda

Go back

Ubuntu 20.04
Lambda Stack pre-installed

2x AMD EPYC 7513
32 cores, 2.60 GHz

8x NVIDIA RTX A6000
NVLinked, 48 GB VRAM per GPU



512 GB
Memory

1.92 TB SSD (NVMe)
OS Drive

0 TB
Extra storage

GPUs

4x NVIDIA RTX A4500 -\$38,290
20 GB of VRAM per GPU, NVLink, PCIe 4.0

8x NVIDIA RTX A4500 -\$30,240
20 GB of VRAM per GPU, NVLink, PCIe 4.0

4x NVIDIA RTX A5000 -\$34,650
24 GB of VRAM per GPU, NVLink, PCIe 4.0

8x NVIDIA RTX A5000 -\$22,960
24 GB of VRAM per GPU, NVLink, PCIe 4.0

4x NVIDIA RTX A5500 -\$28,210
24 GB of VRAM per GPU, NVLink, PCIe 4.0

8x NVIDIA RTX A5500 -\$10,080
24 GB of VRAM per GPU, NVLink, PCIe 4.0

4x NVIDIA RTX A6000 -\$23,170
48 GB of VRAM per GPU, NVLink, PCIe 4.0

8x NVIDIA RTX A6000
16 GB of VRAM per GPU, NVLink, PCIe 4.0

4x NVIDIA A40 -\$20,510
48 GB of VRAM per GPU, NVLink, PCIe 4.0, passive cooling

8x NVIDIA A40 \$5,320
48 GB of VRAM per GPU, NVLink, PCIe 4.0, passive cooling

4x NVIDIA A30 -\$22,330
24 GB of VRAM per GPU, NVLink, PCIe 4.0, passive cooling

\$61,121.00

Apply for a discount

Download quote

Add to cart





Actionable advice for selecting a GPU machine

Best GPU for Deep Learning in 2022 (so far)

February 28, 2022

GPU recommendation

So, which GPUs to choose if you need an upgrade in early 2022 for Deep Learning? We feel there are two yes/no questions that help you choose between **A100**, **A6000**, and **3090**. These three together probably cover most of the use cases in training Deep Learning models:

- **Do you need multi-node distributed training?** If the answer is yes, go for **A100 80GB / 40GB SXM4** because they are the only GPUs that support **Infiniband**. Without **Infiniband**, your distributed training simply would not scale. If the answer is no, see the next question.
- **How big is your model?** That helps you to choose between **A100 PCIe (80GB)**, **A6000 (48GB)**, and **3090 (24GB)**. A couple of **3090**s are adequate for mainstream academic research. Choose **A6000** if you work with a large image/language model and need multi-GPU training to scale efficiently. An **A6000** system should cover most of the use cases in the context of a single node. Only choose **A100 PCIe 80GB** when working on extremely large models



More information about GPU machines

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.

But beware: the article is missing many Ampere cards like A5000, etc.



Analyzing cloud vs on-prem

Choosing Your Deep Learning Infrastructure: The Cloud vs. On-Prem Debate



By Jennifer Villa, Dave Troiano
July 30, 2020

- Basically, if utilization is high, on-prem is cheaper

We surveyed three performance levels - low, mid, and high - based on GPU model.⁴
We also considered two utilization assumptions: low (10%) and high (100%).

| | Low End Performance, 10% Utilization | Low End Performance, 100% Utilization | Mid Tier Performance, 10% Utilization | Mid Tier Performance, 100% Utilization | High End Performance, 10% Utilization | High End Performance, 100% Utilization |
|---------------------------|---|--|--|---|--|---|
| AWS | \$1.80 | \$0.92 | N/A | N/A | \$3.06 | \$1.13 |
| GCP | \$1.42 | \$0.99 | \$1.98 | \$1.08 | \$3.00 | \$1.58 |
| Paperspace | \$0.78 | \$0.63 | \$1.10 | \$0.88 | \$2.30 | \$1.84 |
| Pre-Built DL Server | \$1.56 | \$0.16 | \$1.67 | \$0.17 | \$3.42 | \$0.34 |
| Built-from-scratch DL Ser | \$1.17 | \$0.12 | \$1.26 | \$0.13 | \$2.57 | \$0.26 |

Table 2: Cost Per Hour of Cloud and On-Prem Options

Pre-Ampere!



Recommendations

- Your own GPU machine is worth it just for the mindset shift of maximizing utility vs minimizing cost.
- To scale out experiments, use the most expensive per-hour machines in the least expensive cloud.
 - TPUs are worth experimenting with for large-scale training
- Lambda Labs is a sponsor of the FSDL projects this year, and an excellent choice for both!

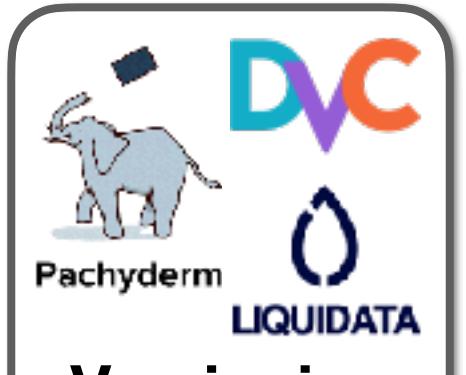
“All-in-one”



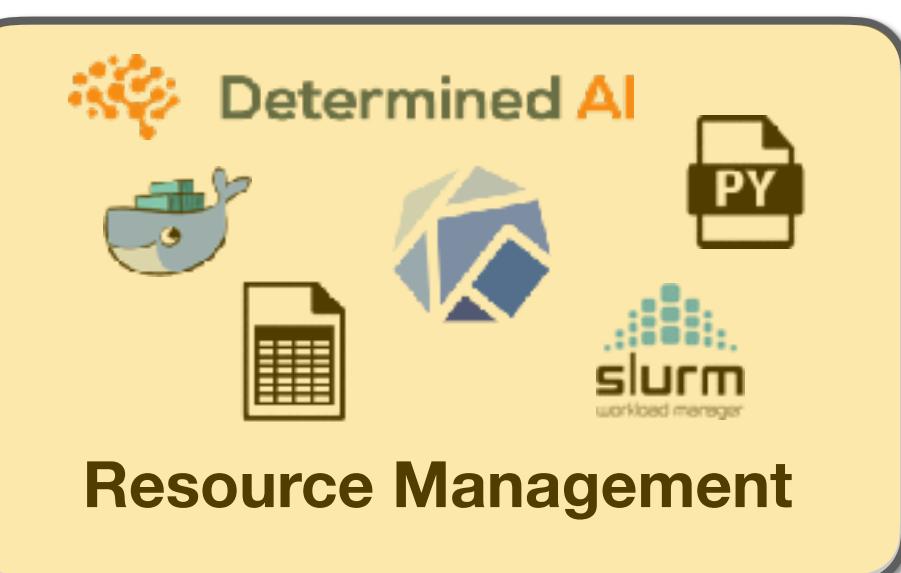
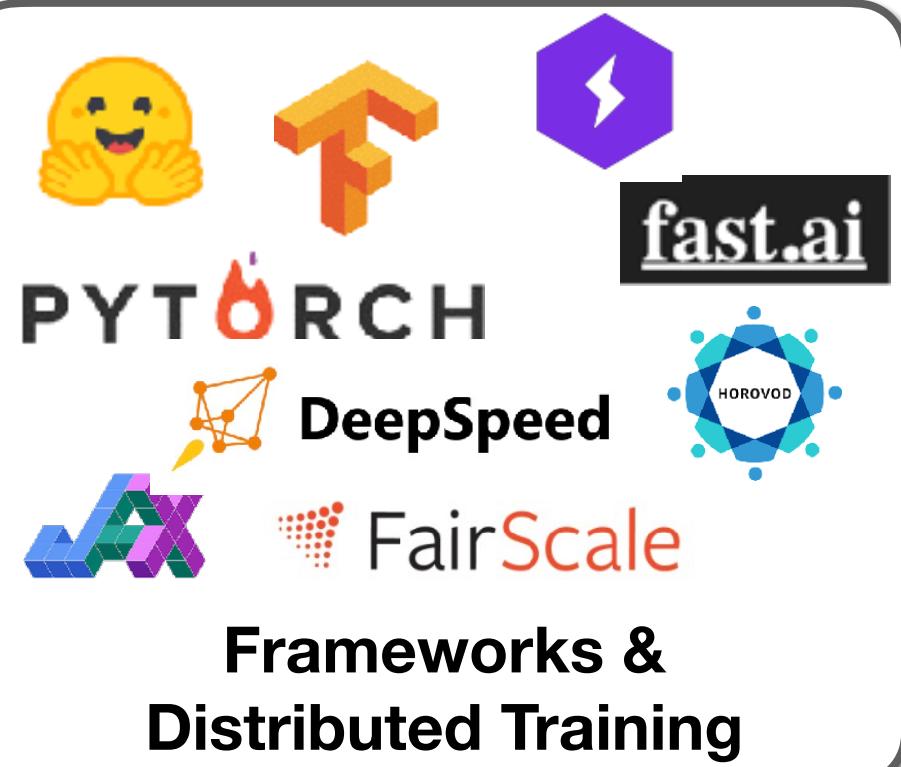
Amazon SageMaker

gradient^o
by Paperspace

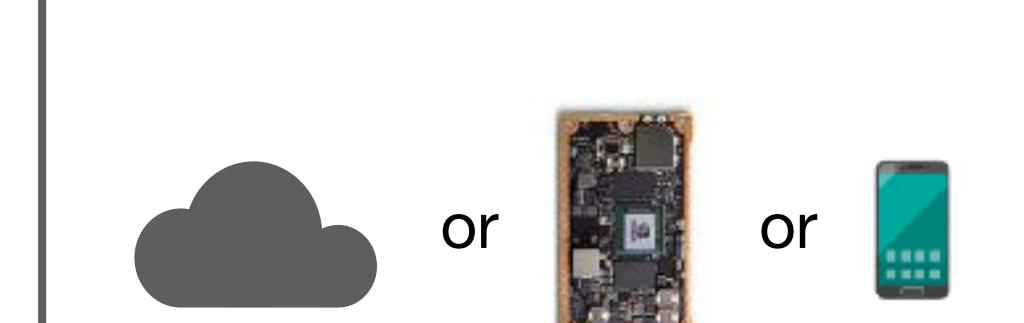
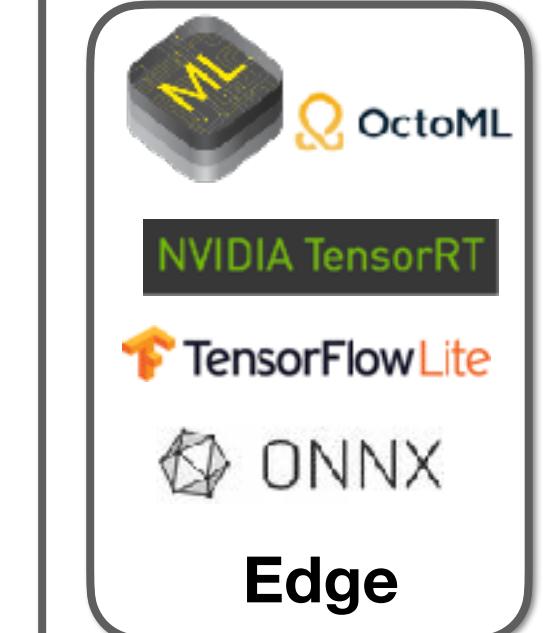
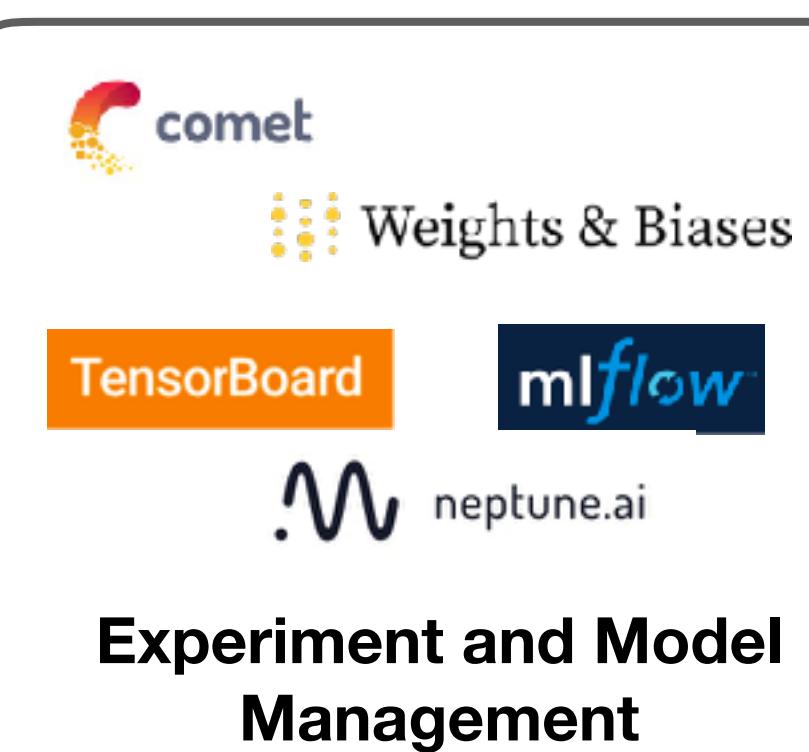
DOMINO
DATA LAB



Data



Development



Deployment





Resource Management

- We want to launch an experiment or set of experiments
- Each experiment needs
 - Machine(s) + GPU(s)
 - Setup (Python+CUDA version, Python requirements)
 - Data
- Solutions
 - Manual
 - SLURM
 - Docker + Kubernetes
 - Software specialized for ML use cases



Manual

- Follow best practices for specifying dependencies (e.g. conda + pip-tools)
- Log in to machine, launch experiment
- This works just fine for one machine

```
● ● ●  
conda activate myproject  
python run_experiment.py --gpus=0,1,2
```

Slurm

- What if you have a **cluster** of machines?
- Slurm is an old-school solution to *workload management* that is still widely used
- A job defines necessary resources, gets queued
- FSDL 😊

 stas00 auto-sync flag was renamed

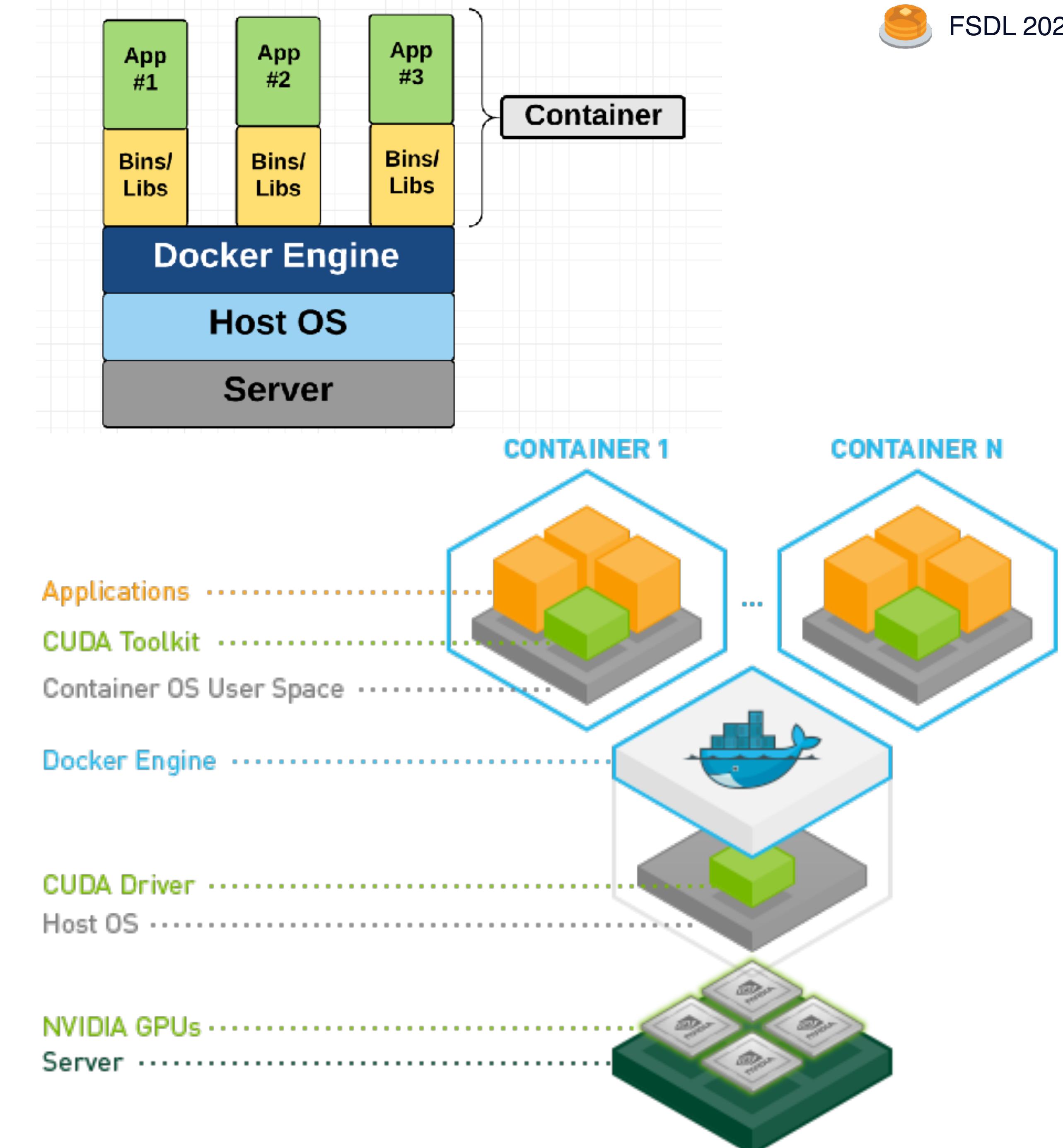
8 4 contributors    

221 lines (184 sloc) | 6.87 KB

```
1 #!/bin/bash
2 #SBATCH --job-name=tr11-176B-ml
3 #SBATCH --partition=gpu_p5
4 #SBATCH --constraint=a100
5 #SBATCH --reservation=hug
6 #SBATCH --qos=qos_gpu-gc          # up to 100h
7 #SBATCH --nodes=24
8 #SBATCH --ntasks-per-node=1      # crucial - only 1 task per dist per node
9 #SBATCH --cpus-per-task=64        # number of cores per tasks
10 #SBATCH --hint=nomultithread    # we get physical cores not logical
11 #SBATCH --gres=gpu:8            # number of gpus
12 #SBATCH --time 100:00:00         # maximum execution time (HH:MM:SS)
13 #SBATCH --output=%x-%j.out       # output file name
14 #SBATCH --account=six@a100
15
16 set -x -e
17
18 #source $six_ALL_CCFRWORK/start-py38-pt110
19 #source $six_ALL_CCFRWORK/start-py38-pt111
20 source $six_ALL_CCFRWORK/code/tr11-176B-ml/bigscience/train/tr11-176B-ml/start
21
22 echo "START TIME: $(date)"
```

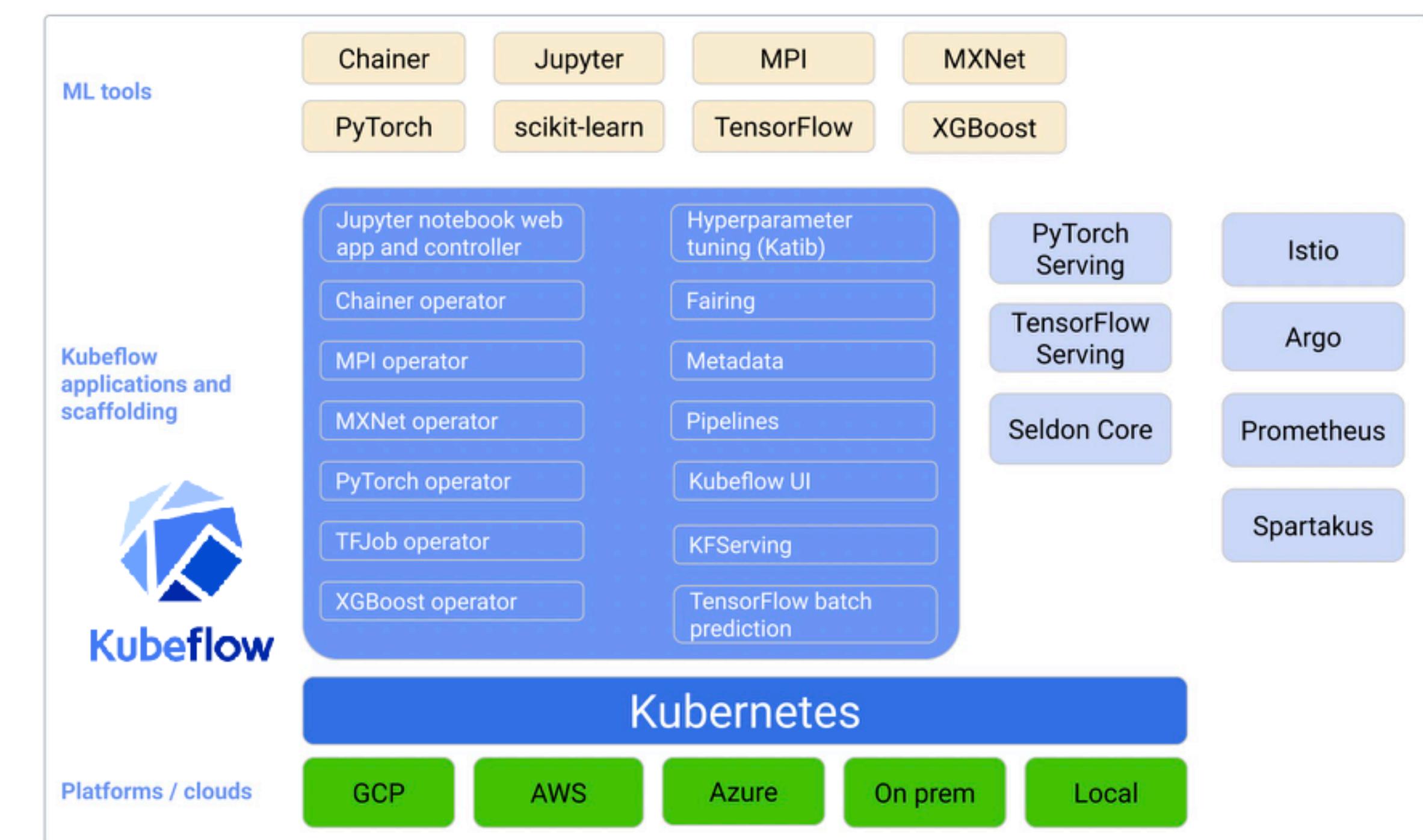
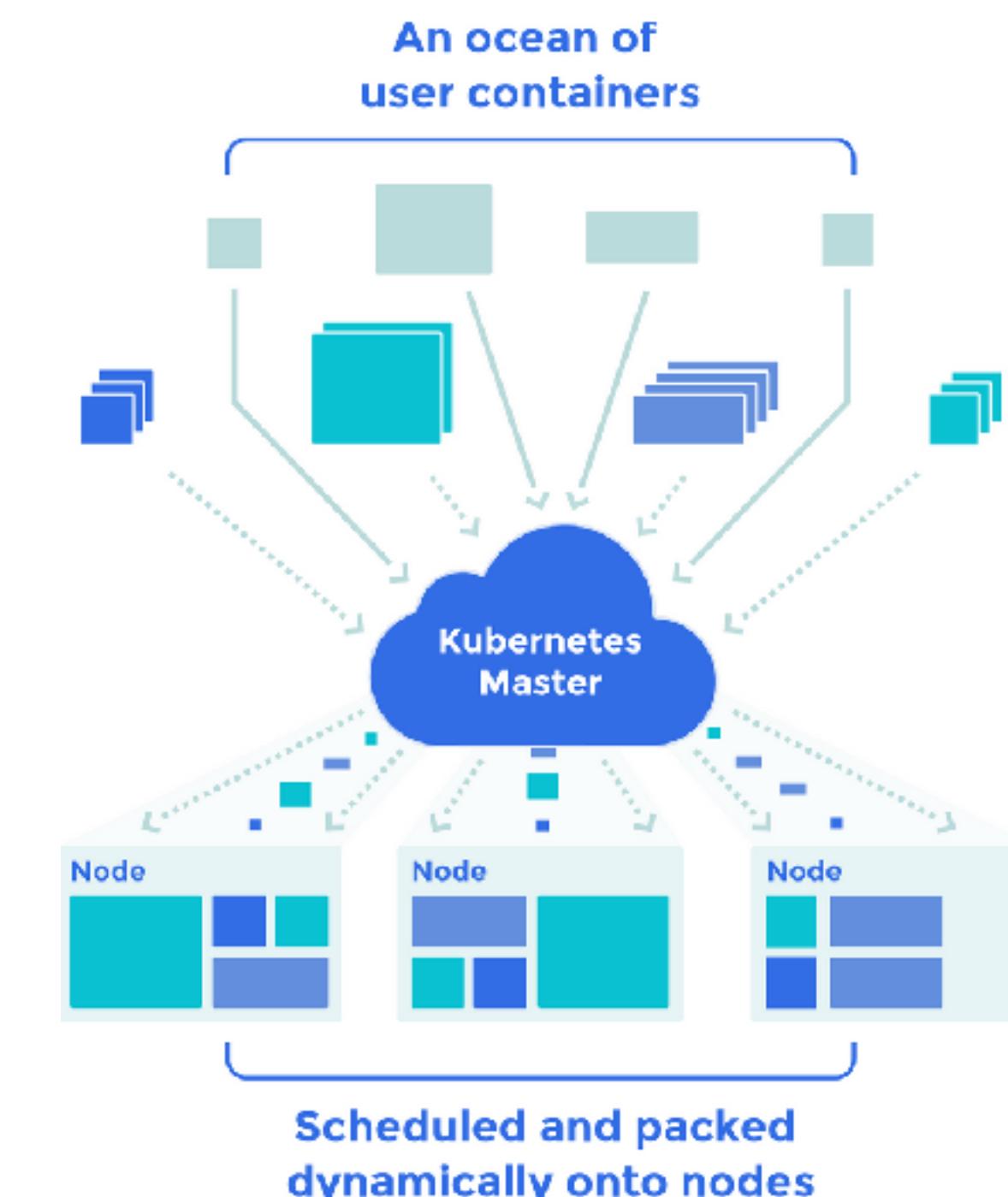
Docker

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



Kubernetes + Kubeflow

- Kubernetes is a way to run many Docker containers on top of a cluster
- Kubeflow is an open source project from Google
- Spawn and manage Jupyter notebooks, and manage multi-step ML workflows
- FSDL 😊





What about the cloud?

- Slurm or Kubeflow make sense if you already have a cluster up and running.
- But how do we get it up and running in the cloud in the first place?



Before we proceed...

I prefer not to mention SaaS that does not show pricing



AWS SageMaker

- Set of ML-focused solutions from AWS
- Notebooks are a key feature
- Could make sense to adopt if already using AWS for everything

The screenshot shows the AWS SageMaker Studio interface. A Jupyter notebook titled "random_forest.ipynb" is open. The code cell contains Python code for computing anomaly scores and plotting them. The plot shows two time series: "Taxi Ridership" (blue dots) and "Anomaly Score" (red dots), with the anomaly score spikes clearly visible.

```
Computing Anomaly Scores

Now, let's compute and plot the anomaly scores from the entire taxi dataset.

In [1]: results = rcf_inference.predict(taxi_data_numpy)
scores = [datum['score'] for datum in results['scores']]

# add scores to taxi data frame and print first few values
taxi_data['score'] = pd.Series(scores, index=taxi_data.index)
taxi_data.head()

Out[1]:
```

```
[1]: fig, ax1 = plt.subplots()
ax2 = ax1.twinx()

# Try this out - change 'start' and 'end' to zoom in on the
# anomaly round earlier in this notebook
#
# start, end = 0, len(taxi_data)
# start, end = 5500, 6500
taxi_data_subset = taxi_data[start:end]

ax1.plot(taxi_data_subset['value'], color='C0', alpha=0.8)
ax2.plot(taxi_data_subset['score'], color='C1')

ax1.grid(which='major', axis='both')

ax1.set_ylabel('Taxi Ridership', color='C0')
ax2.set_ylabel('Anomaly Score', color='C1')

ax1.tick_params('y', colors='C0')
ax2.tick_params('y', colors='C1')

ax1.set_ylim(0, 40000)
ax2.set_ylim(min(scores), 1.4*max(scores))
fig.set_figwidth(10)
```

Note that the anomaly score spikes where our eyeball-norm method suggests there is an anomalous datapoint as well as in some places where our eyeballs are not as accurate.

Typical SageMaker workflow



1. Label data

Set up and manage labeling jobs for highly accurate training datasets within Amazon SageMaker, using active learning and human labeling.



2. Build

Connect to other AWS services and transform data in Amazon SageMaker notebooks.



3. Train

Use Amazon SageMaker's algorithms and frameworks, or bring your own, for distributed training.



4. Tune

Amazon SageMaker automatically tunes your model by adjusting multiple combinations of algorithm parameters.



5. Deploy

After training is completed, models can be deployed to Amazon SageMaker endpoints, for real-time predictions.



AWS SageMaker for Training

- Intended use cases are more old-school
- And you may perhaps get overwhelmed by all the AWS-specific configurations...

The screenshot shows the 'Job settings' configuration page for AWS SageMaker. At the top, there's a dropdown menu with various algorithm options. Below it, there's a section for 'Resource configuration' where the instance type is set to 'ml.m4.xlarge' and the instance count is '1'. There's also a field for 'Additional storage volume per instance (GB)' which is currently empty. At the bottom, there's a section for 'Encryption key - optional' with a dropdown menu showing 'No Custom Encryption'.

Job settings

- Tabular - XGBoost : v1.3
- Job n... Tabular - Linear Learner
- Tabular - K Nearest Neighbors (KNN)
- Maxim... Tabular - Factorization Machines
- IAM r... Tabular - Object2Vec
- Amazon... Vision - Image Classification (MxNet)
- Vision - Object Detection (MxNet)
- Vision - Semantic Segmentation (MxNet)
- Algor... Clustering - K-Means
- Time Series Forecast - DeepAR
- Text Classification & Text Embedding - Blazing Text
- Text Transformation - Sequence to Sequence (MxNet)
- Text Topic Modeling - Neural Topic Modeling (NTM)
- Text Topic Modeling - Latent Dirichlet Allocation (LDA)
- Dimensionality Reduction - Principal Component Analysis (PCA)
- Anomaly Detection - Random Cut Forest
- Anomaly Detection - IP Insights

Choose an algorithm or custom training image...

Enable SageMaker metrics time series

Resource configuration

| Instance type | Instance count | Additional storage volume per instance (GB) |
|---------------|----------------|---|
| ml.m4.xlarge | 1 | 1 |

Encryption key - optional
Encrypt your data. Choose an existing KMS key or enter a key's ARN.

No Custom Encryption

Stopping condition



AWS SageMaker for Training

- But increasing support for PyTorch
- Support for using cheap spot instances (see link at bottom)

```
[ ]: from sagemaker.pytorch import PyTorch  
  
estimator = PyTorch(  
    base_job_name="pytorch-smdataparallel-mnist",  
    source_dir="code",  
    entry_point="train_pytorch_smdataparallel_mnist.py",  
    role=role,  
    framework_version="1.11.0",  
    py_version="py38",  
    # For training with multinode distributed training, set this count. Example: 2  
    instance_count=1,  
    # For training with p3dn instance use - ml.p3dn.24xlarge, with p4dn instance use - ml.  
    instance_type="ml.p4d.24xlarge",  
    sagemaker_session=sagemaker_session,  
    # Training using SMDistributed Distributed Training Framework  
    distribution={"smdistributed": {"dataparallel": {"enabled": True}}},  
    debugger_hook_config=False,  
)  
  
[ ]: estimator.fit()
```

https://sagemaker-examples.readthedocs.io/en/latest/training/distributed_training/pytorch/data_parallel/mnist/pytorch_smdataparallel_mnist_demo.html

15-20% markup vs basic EC2!

| ml.p3.8xlarge | 32 | 244 GiB | \$14.688 | 12.24 | 3 |
|------------------|----|---------|----------|-------|---|
| ml.p3.16xlarge | 64 | 488 GiB | \$28.152 | 24.48 | 3 |
| ml.p3dn.24xlarge | 96 | 768 GiB | \$35.894 | 31.21 | 3 |

https://github.com/aws-samples/amazon-sagemaker-managed-spot-training/blob/main/pytorch_managed_spot_training_checkpointing/pytorch_managed_spot_training_checkpointing.ipynb

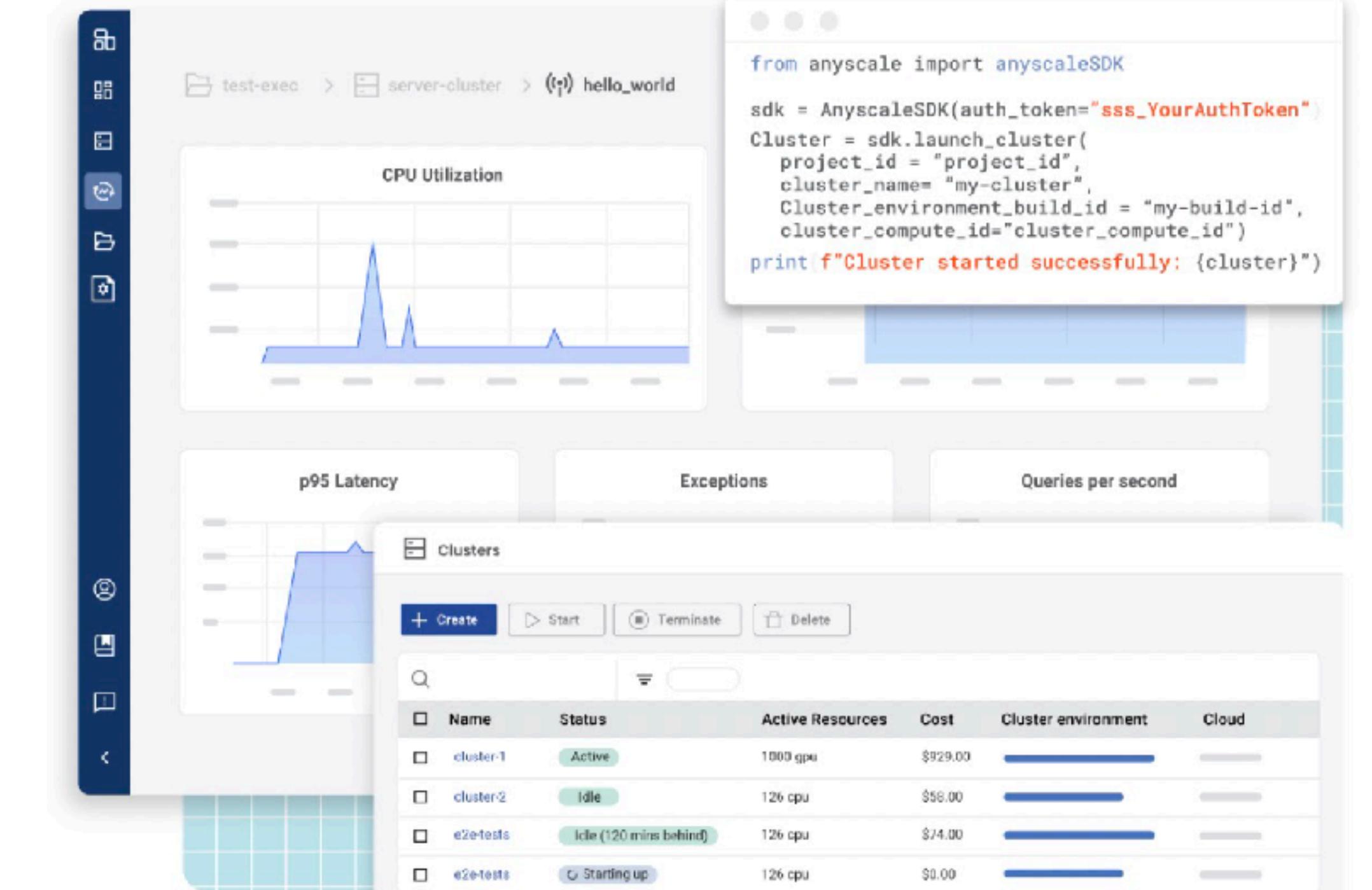
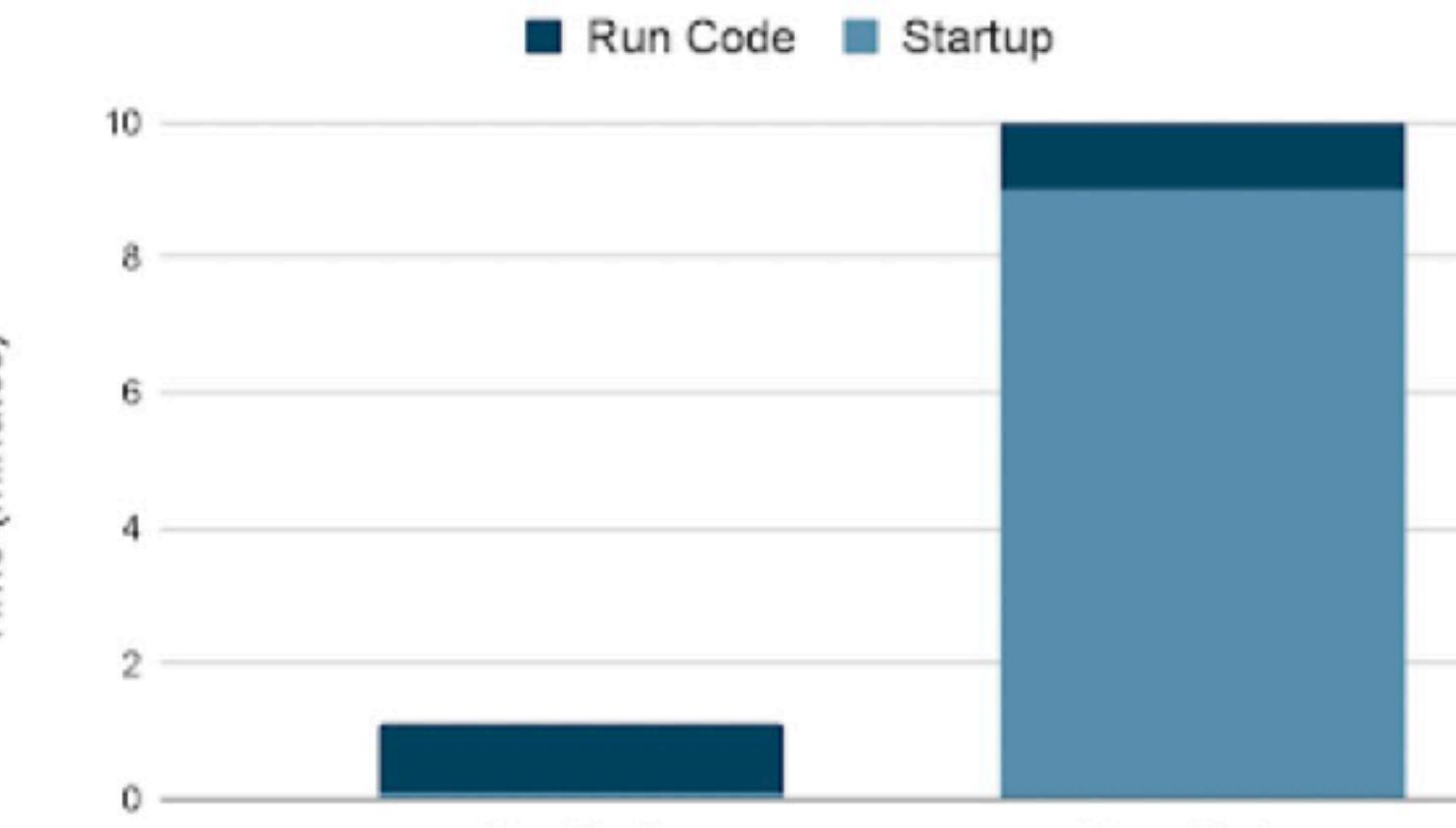
Anyscale (makers of Ray, from Berkeley)

```
trainer = Trainer(backend="torch", num_workers=100, use_gpu=True)
```

- Ray Train is a new project from Ray
 - Faster than SageMaker
 - Intelligent spot instance support
- Anyscale makes it super simple to provision compute (at a **significant** markup to AWS)

<https://docs.ray.io/en/latest/train/train.html>

Iteration Time





Grid.ai

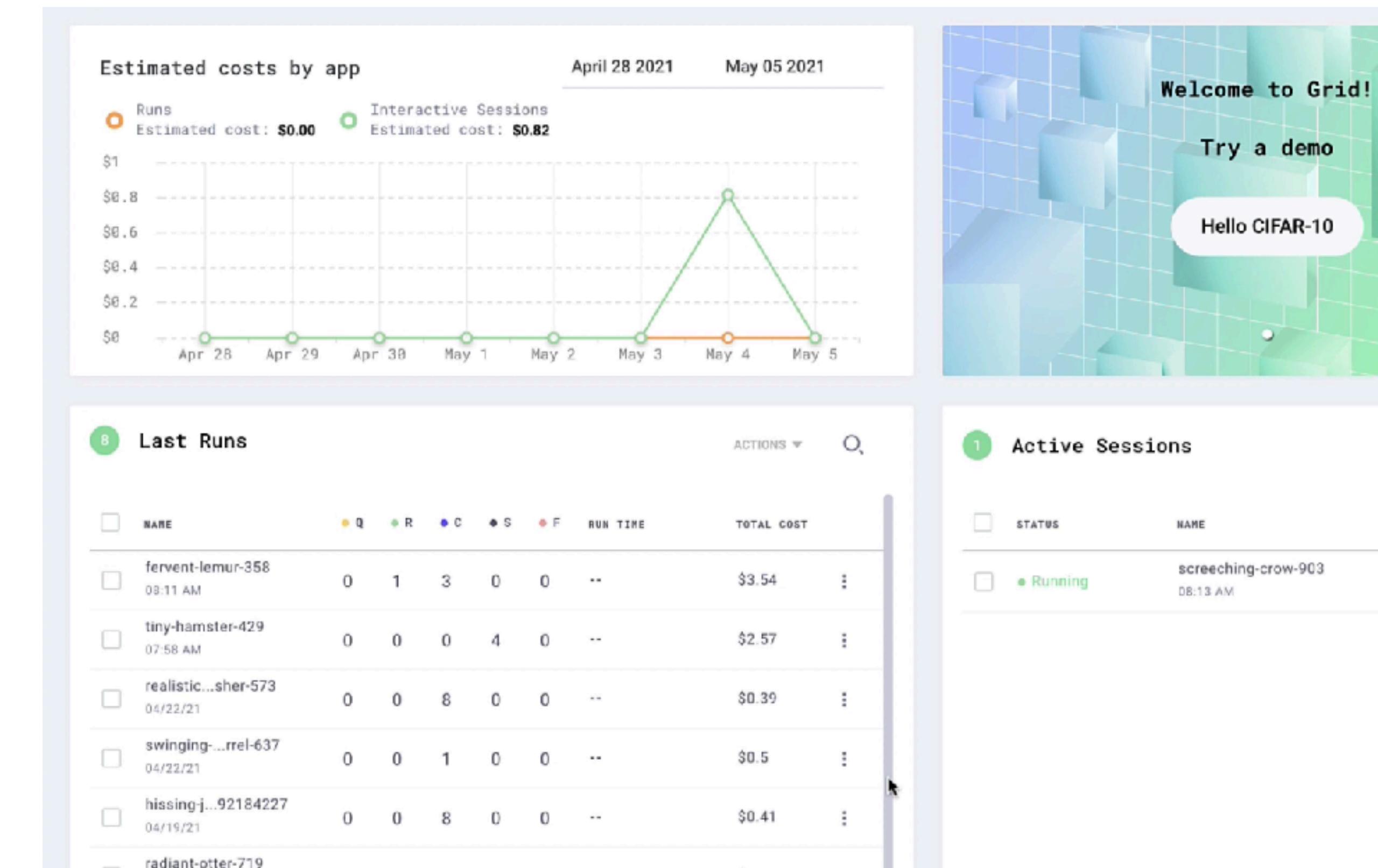
From the makers of PyTorch Lightning

```
(flash) → dcgan git:(master) ✘ grid run --g_instance_type 4_v100_16gb --g_gpus 2 --g_use_spot mai  
n.py --dataset cifar10 --lr 0.0002 --dataroot . --ngpu 2 --cuda
```

SEAMLESSLY TRAIN
100s OF MACHINE
LEARNING MODELS ON
THE CLOUD FROM YOUR
LAPTOP

With zero code changes

Not sure about long-term
viability of this
(grid.ai vs lightning.ai)





Non-ML specific solutions

- You don't **need** SageMaker to provision training job compute on AWS
- Can write your own scripts, or use some libraries
- (But nothing we can recommend)



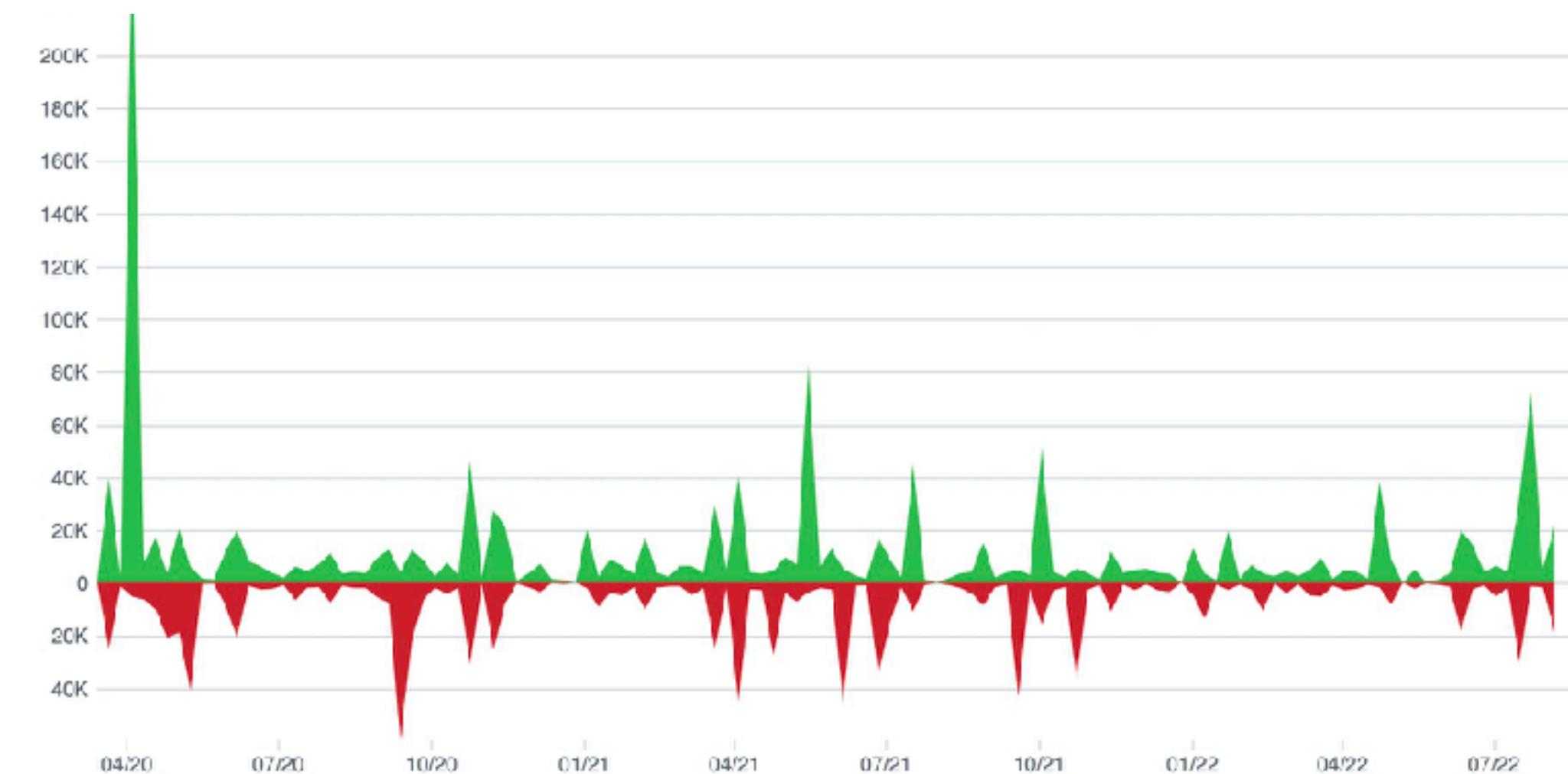
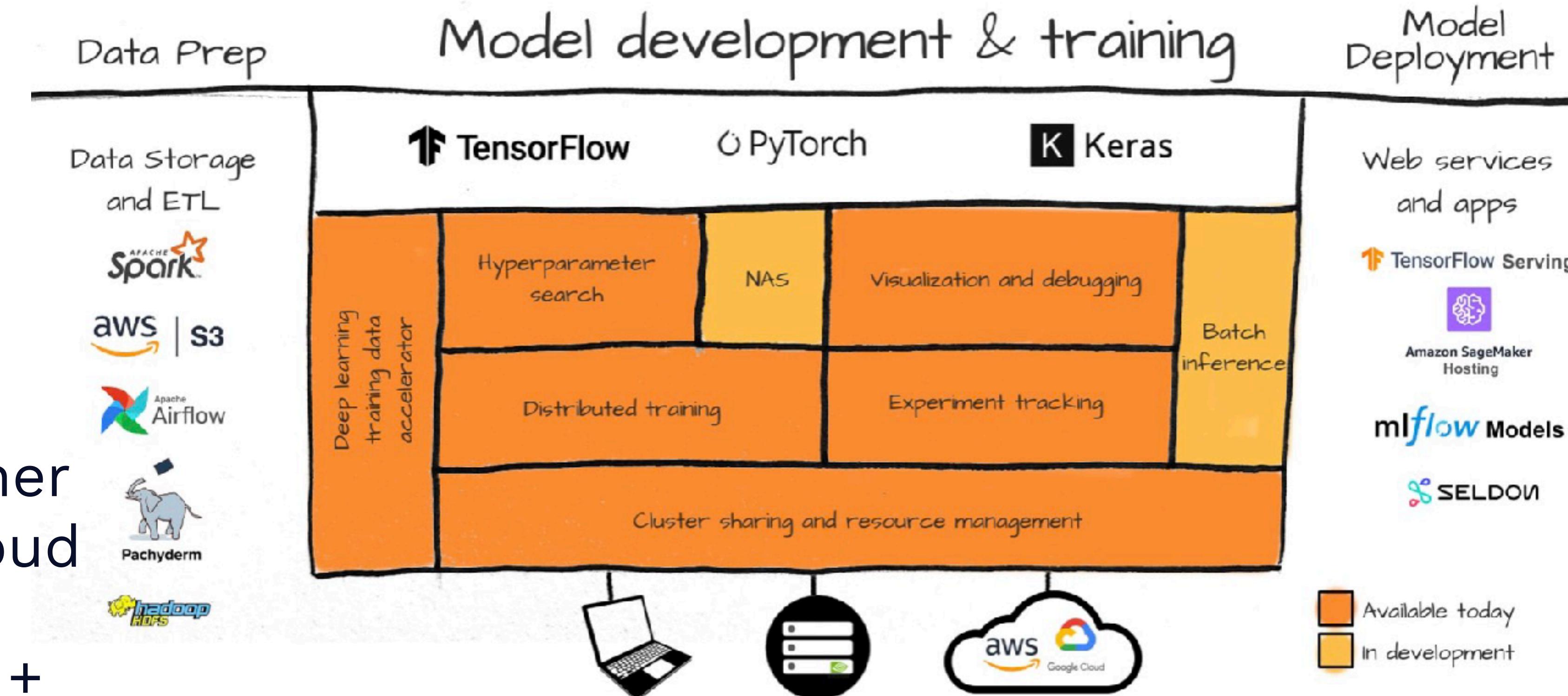
[documentation](#) [reference](#) [pypi v1.3.3](#) [python 3.6 | 3.7](#) [license MIT](#)

Spotty drastically simplifies training of deep learning models on [AWS](#) and [GCP](#):

<https://github.com/spotty-cloud/spotty>

Determined.ai

- Great open-source solution that lets you manage a cluster either on-prem or in the cloud
- Cluster management + distributed training + extra stuff
- Acquired by HP, but still in active development





Determined.ai

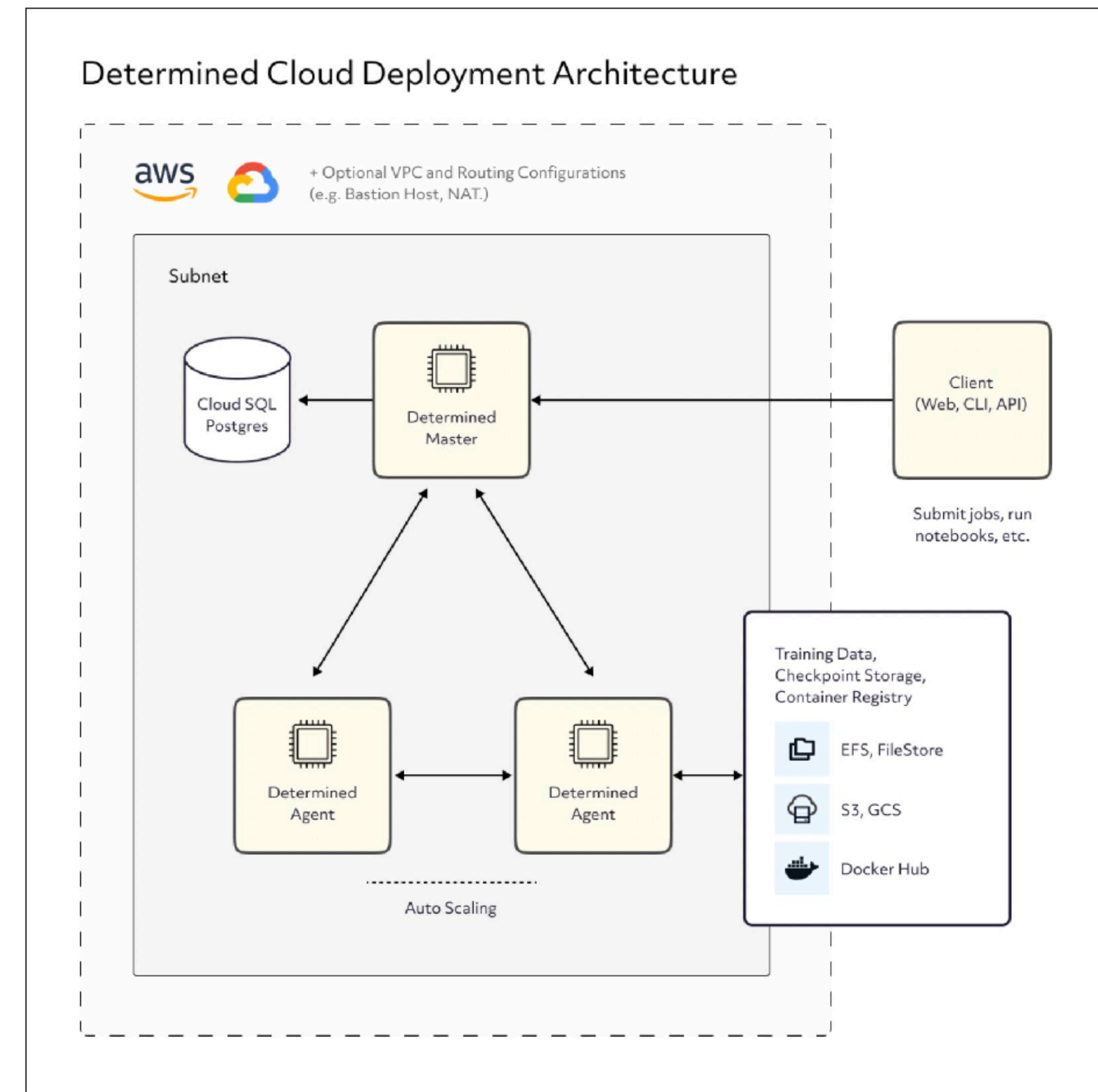
```
# To start a cluster with GPUs, remove `no-gpu` flag.  
det deploy local cluster-up --no-gpu  
# Access web UI at localhost:8080. By default, "determined" user accepts a blank password.
```

The screenshot shows the Determined.ai web interface. On the left is a sidebar with the user 'admin' and various navigation links: Launch JupyterLab, Uncategorized, Model Registry, Tasks, Cluster (highlighted with a blue bar showing 50%), Workspaces, Test Workspace, Docs, API (Beta), and Share Feedback. The main area is titled 'Cluster - 50%' and includes tabs for Overview, Historical Usage, and Master Logs. The Overview section displays four metrics: Connected Agents (1), CUDA Slots Allocated (1/2), Aux Containers Running (0/100), and Active JupyterLabs (1). Below these is a progress bar for Compute (CUDA) Slots Allocated, showing 1/2 (50.00%) with states: 1 (50%) RUNNING, 0 (0%) PENDING, and 1 (50%) FREE. A detailed view of the 'default' Resource Pool is shown, listing its configuration: Accelerator (Default), Type (Static), Instance Type (N/A), Connected Agents (1), Slots Per Agent (Unknown), Aux Containers Per Agent (100), and Scheduler Type (Fairshare). The '1/2 CUDA Slots Allocated' and '1 Active' status are also visible here.



Determined.ai

- Support for both AWS and GCP (and anything else you can run Kubernetes on)





I feel that a truly simple solution to launching training on many cloud instances still does not exist

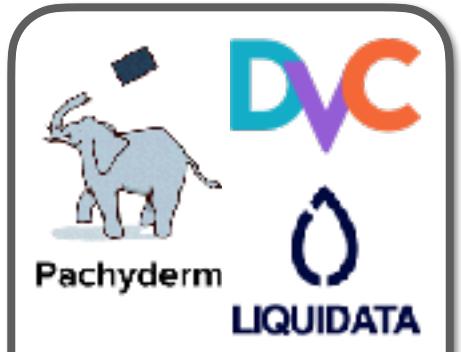
"All-in-one"



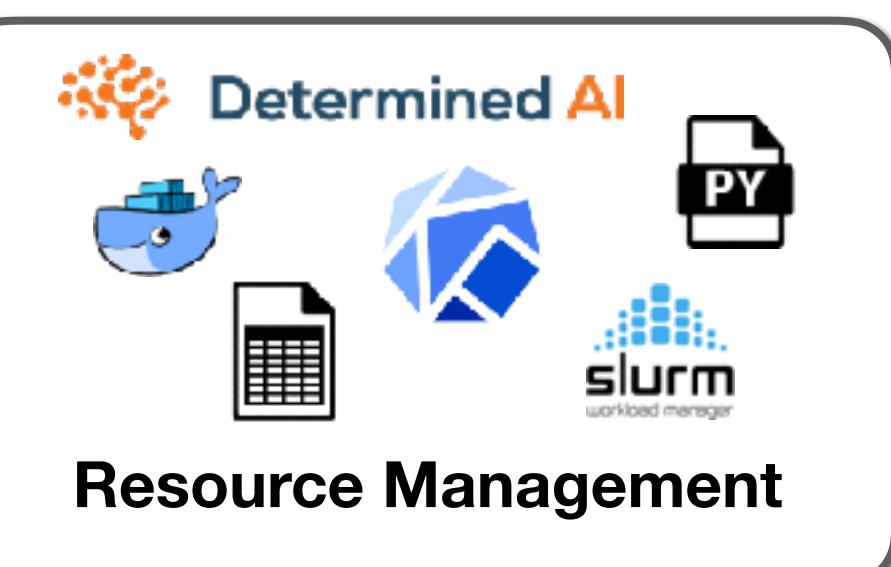
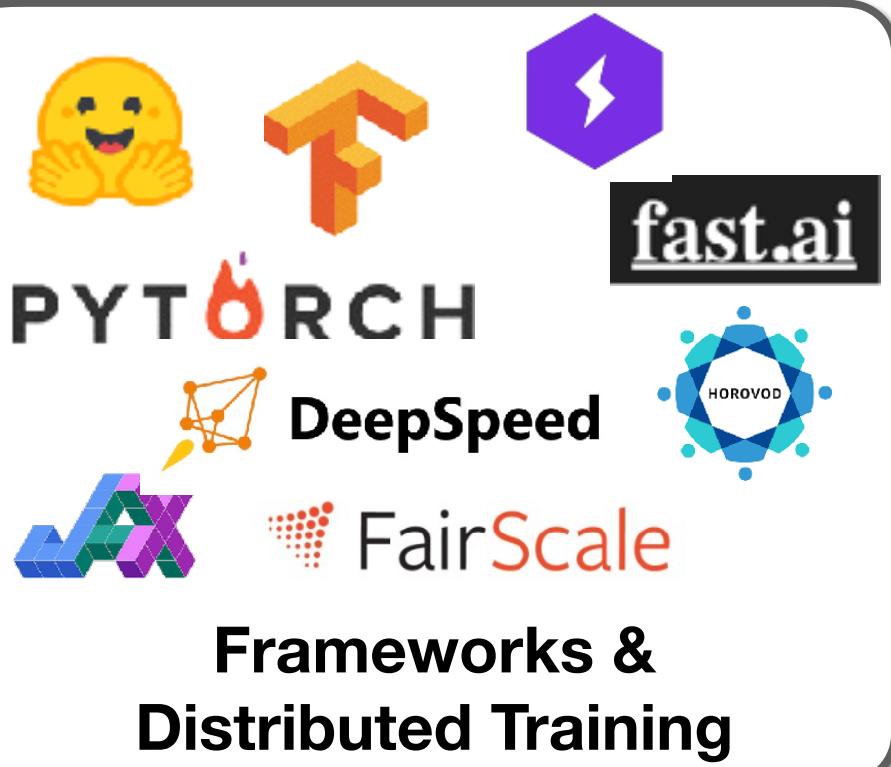
Amazon SageMaker

gradient^o
by Paperspace

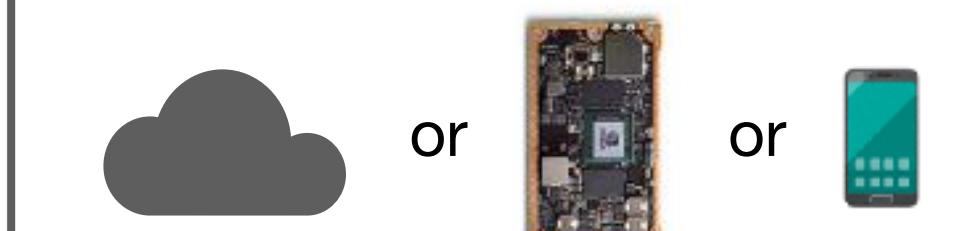
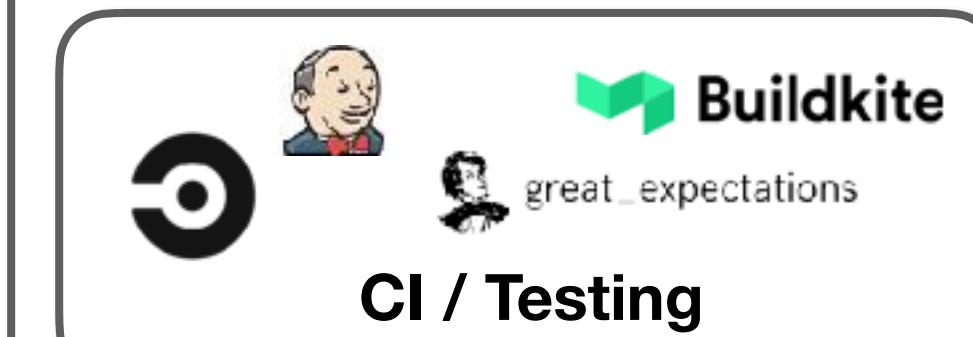
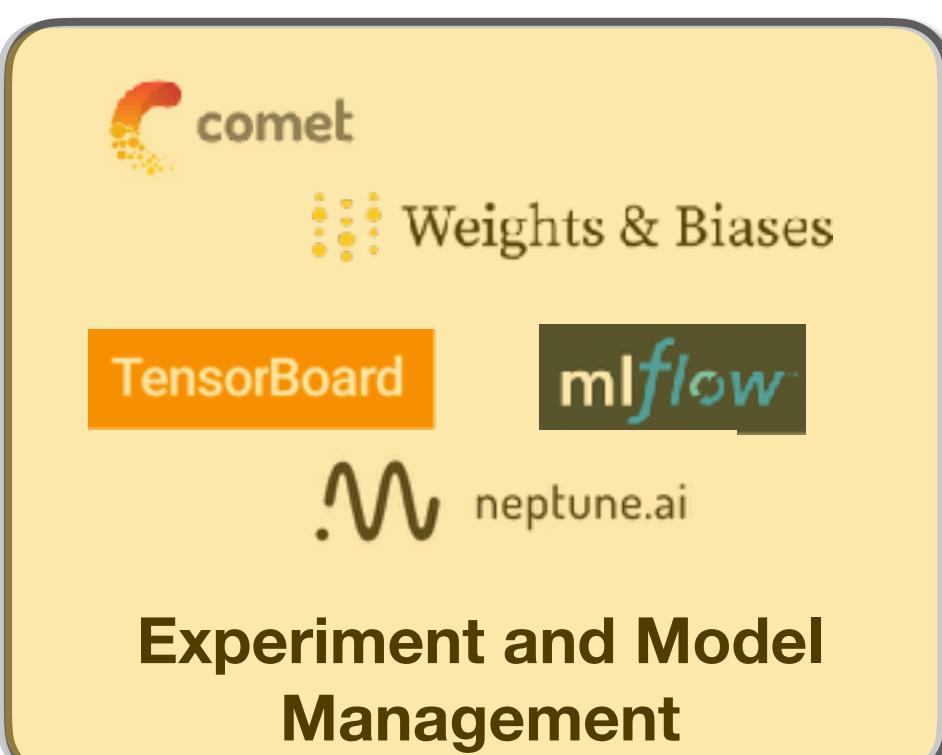
DOMINO
DATA LAB



Data



Development





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.

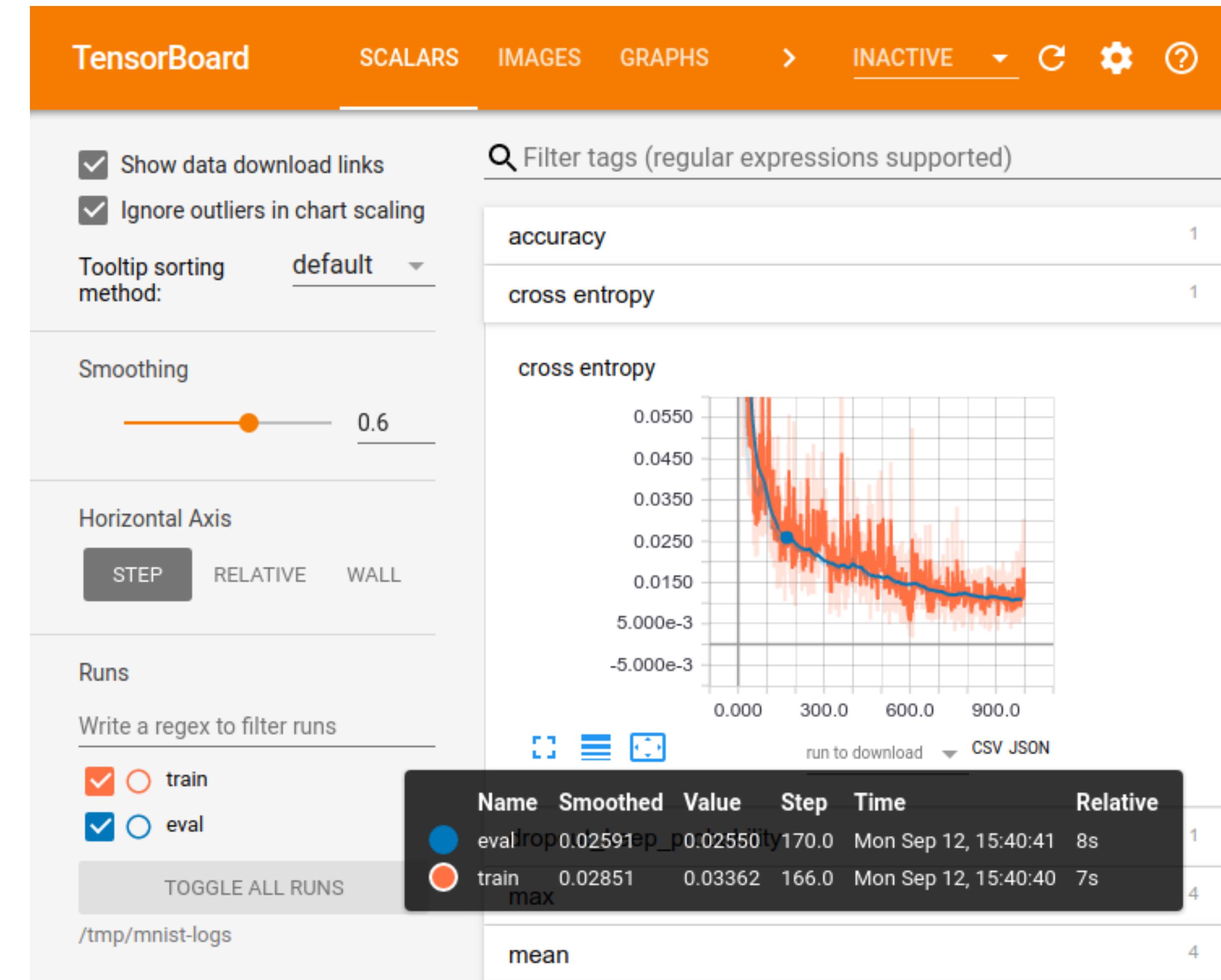
| | | | | | | | |
|----|--------------------------|------------------------------|----------------|----------------|----------------|----------------|----------------|
| 11 | Dataset | Split (train/dev/test) | 0.7/0.2/0.1 | 0.7/0.2/0.1 | 0.7/0.2/0.1 | 0.7/0.15/0.15 | 0.7/0.15/0.15 |
| 12 | | Class ratio (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 | Test results | fn | 1189 | 1185 | 1190 | 1154 | 1357 |
| 25 | | loss | 0.59637538 | 0.594134 | 0.594134 | 0.6037084 | 0.594134 |
| 26 | | 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

- From Google, but not exclusive to Tensorflow
- A fine solution for single experiments
- Gets unwieldy to manage many experiments, and to properly store past work

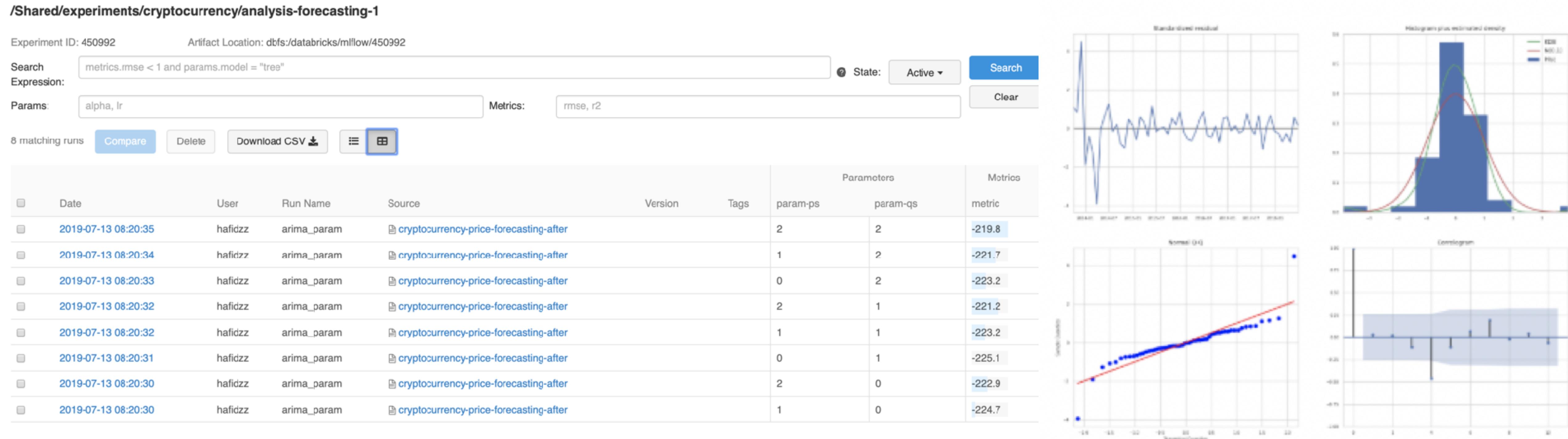




MLFlow tracking

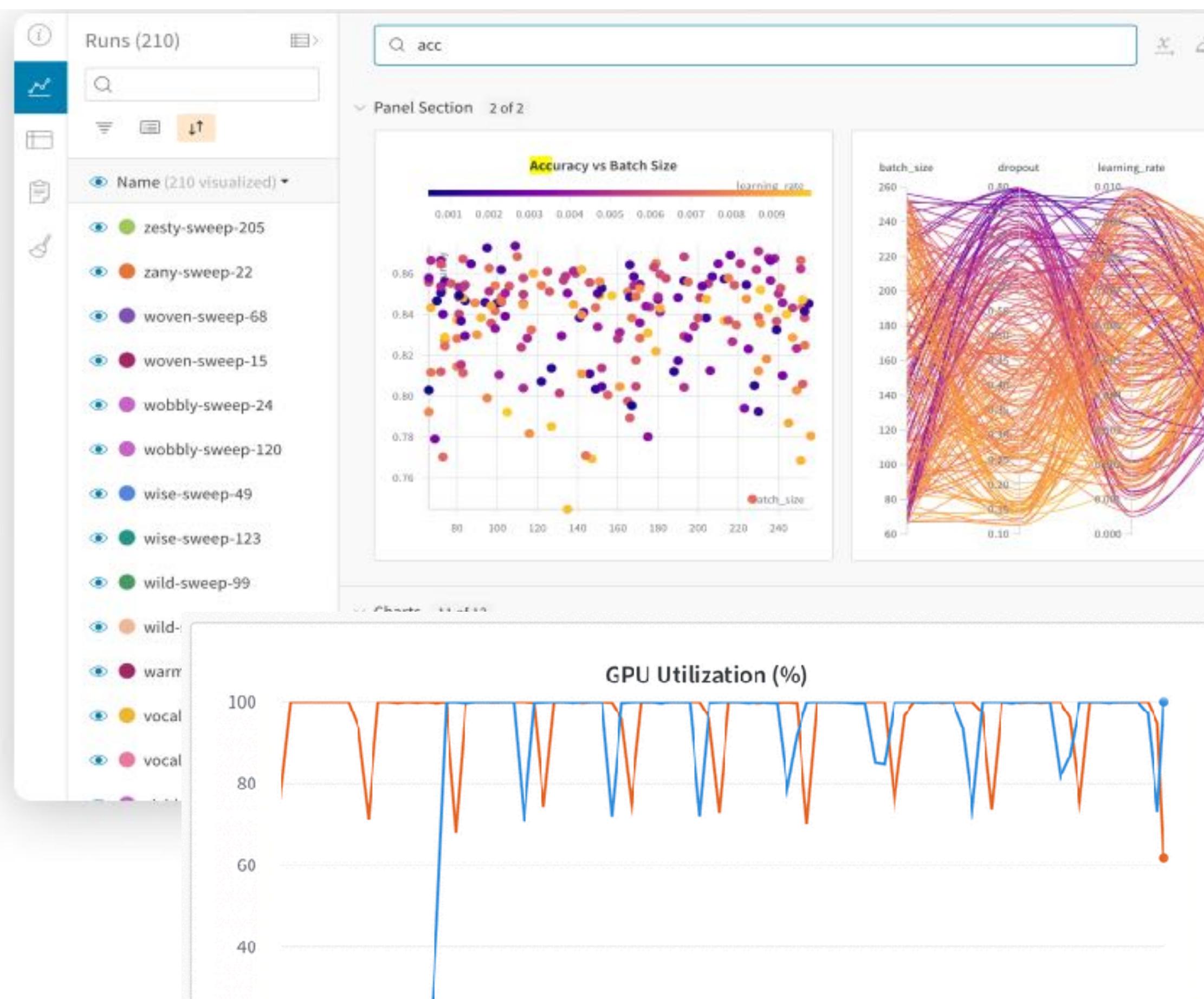


- Open-source solution for experiment and model management
- Need to host yourself



Weights & Biases

- Really popular, super easy-to-use solution
- Hosted free for public projects



The figure shows the Weights & Biases interface displaying a list of 'Runs (398)'. The columns include Name, Runtime, Notes, Tags, batch_size, encoder, epochs, learning_rate, num_train, num_valid, and train. Several runs are highlighted with colored circles and labels: 'apricot-firebrand-436' (purple), 'icy-grass-435' (green), 'curious-meadow-434' (red), 'laced-galaxy-433' (blue), 'best car acc (50% dat)' (grey), 'best traffic acc (50% c)' (pink), 'best human iou (50%)' (brown), 'best overall IOU (20%)' (light green), 'hopeful-violet-428' (orange), 'major-firefly-427' (dark green), 'vibrant-cherry-426' (cyan), 'good-cosmos-425' (yellow), 'whole-serenity-424' (purple), 'daily-river-423' (light blue), and 'worldly-totem-422' (red).

| Name | Runtime | Notes | Tags | batch_size | encoder | epochs | learning_rate | num_train | num_valid | train |
|--------------------------|---------|-----------|-----------|------------|----------|--------|---------------|-----------|-----------|-------|
| apricot-firebrand-436 | 4m 52s | Add notes | | 6 | resnet34 | 10 | 0.001311 | 3479 | 483 | 3 |
| icy-grass-435 | 4m 54s | Add notes | | 6 | resnet34 | 10 | 0.001311 | 3494 | 505 | 3 |
| curious-meadow-434 | 50m 6s | Add notes | | 6 | resnet34 | 10 | 0.001311 | 3495 | 494 | 3 |
| laced-galaxy-433 | 33s | Add notes | | 6 | resnet34 | 10 | 0.001311 | 3521 | 495 | 3 |
| best car acc (50% dat) | 47m 52s | reprod... | seg_masks | 6 | resnet34 | 10 | 0.001311 | 3524 | 492 | 3 |
| best traffic acc (50% c) | 46m 42s | reprod... | seg_masks | 8 | resnet18 | 10 | 0.001 | 3523 | 492 | 2 |
| best human iou (50%) | 31m 34s | reprod... | seg_masks | 7 | alexnet | 10 | 0.0009084 | 1405 | 190 | 3 |
| best overall IOU (20%) | 20m 23s | reprod... | seg_masks | 7 | resnet34 | 10 | 0.001367 | 1376 | 205 | 2 |
| hopeful-violet-428 | 34s | Add notes | | 8 | resnet34 | 1 | 0.001 | 331 | 56 | 2 |
| major-firefly-427 | 8s | Add notes | | 8 | resnet34 | 1 | 0.001 | 355 | 46 | 2 |
| vibrant-cherry-426 | 6m 12s | Add notes | | 8 | resnet34 | 10 | 0.001 | 347 | 42 | 2 |
| good-cosmos-425 | 43s | Add notes | | 8 | resnet34 | 10 | 0.001 | 359 | 40 | 2 |
| whole-serenity-424 | 44s | Add notes | | 8 | resnet34 | 10 | 0.001 | 361 | 49 | 2 |
| daily-river-423 | 1m 8s | Add notes | | 8 | resnet34 | 10 | 0.001 | 693 | 104 | 2 |
| worldly-totem-422 | 12m 54s | Add notes | | 8 | resnet34 | 10 | 0.001 | 682 | 97 | 2 |



Weights & Biases

- Simply init with experiment config (automatic for PyTorch-Lightning), and then log anything you want, including images
- We will use this in Lab 4 this week

```
# Flexible integration for any Python script
import wandb

# 1. Start a W&B run
wandb.init(project='gpt3')

# 2. Save model inputs and hyperparameters
config = wandb.config
config.learning_rate = 0.01

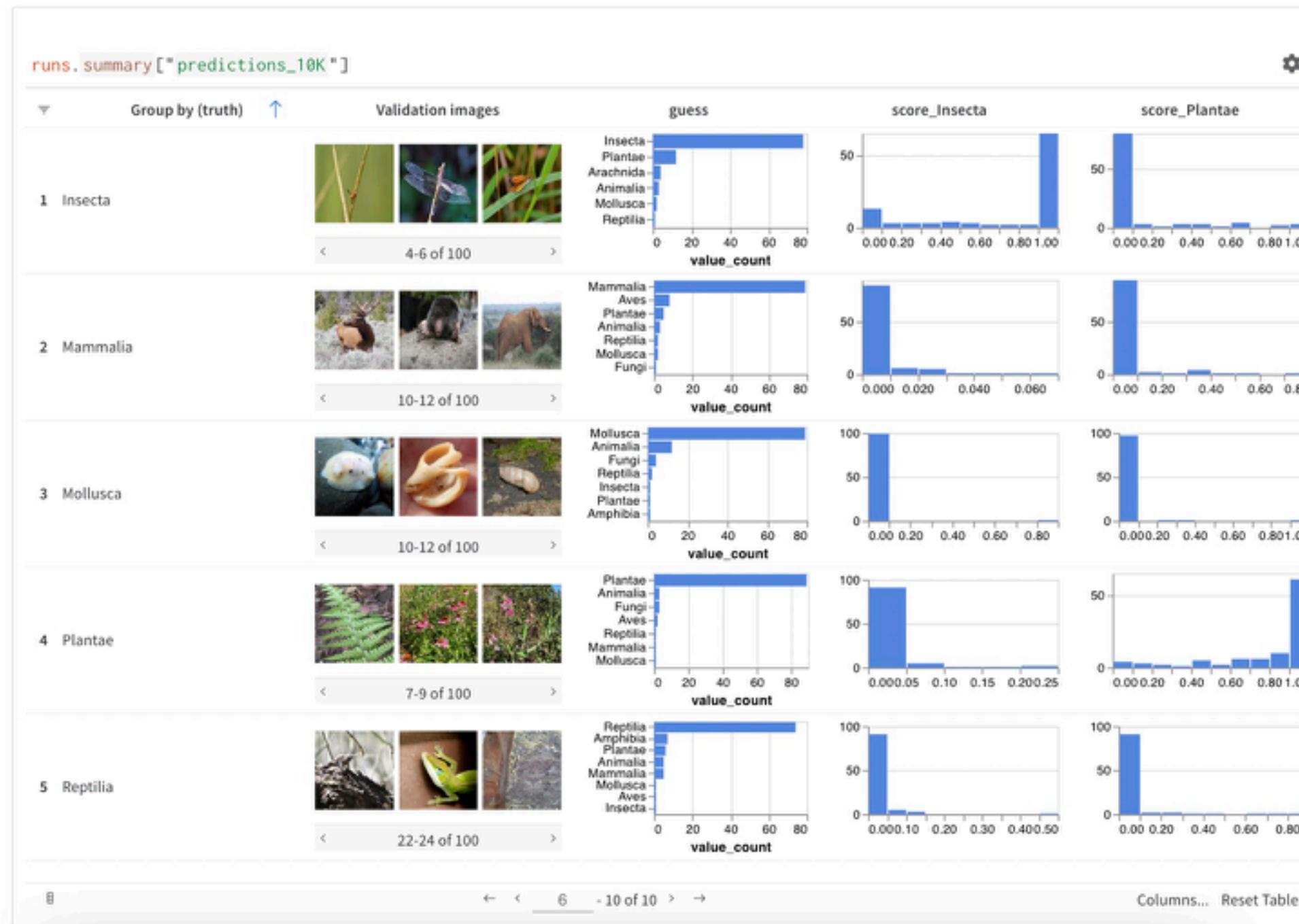
# Model training here

# 3. Log metrics over time to visualize
# performance
wandb.log({"loss": loss})
```



Weights & Biases Reports and Tables

- Share results of your project
- Explore data/predictions



• Interactively Exploring Training Metrics of our Wildfire Smoke Detection Model

Here are the metrics from the two different training runs, the first for 3 epochs and the second for 10 epochs. Loss is going down, mAP (or mean average precision) going up. What more could we ask for?



• Visualizing Model Predictions on Validation data with W&B Tables

Oftentimes, the best way to explore the model predictions is to actually *look* at what model is predicting. W&B Tables is great tool for exactly that. It allows us to interactively compare side-by-side the ground truth validation dataset images and what our model is predicting on these validation images.

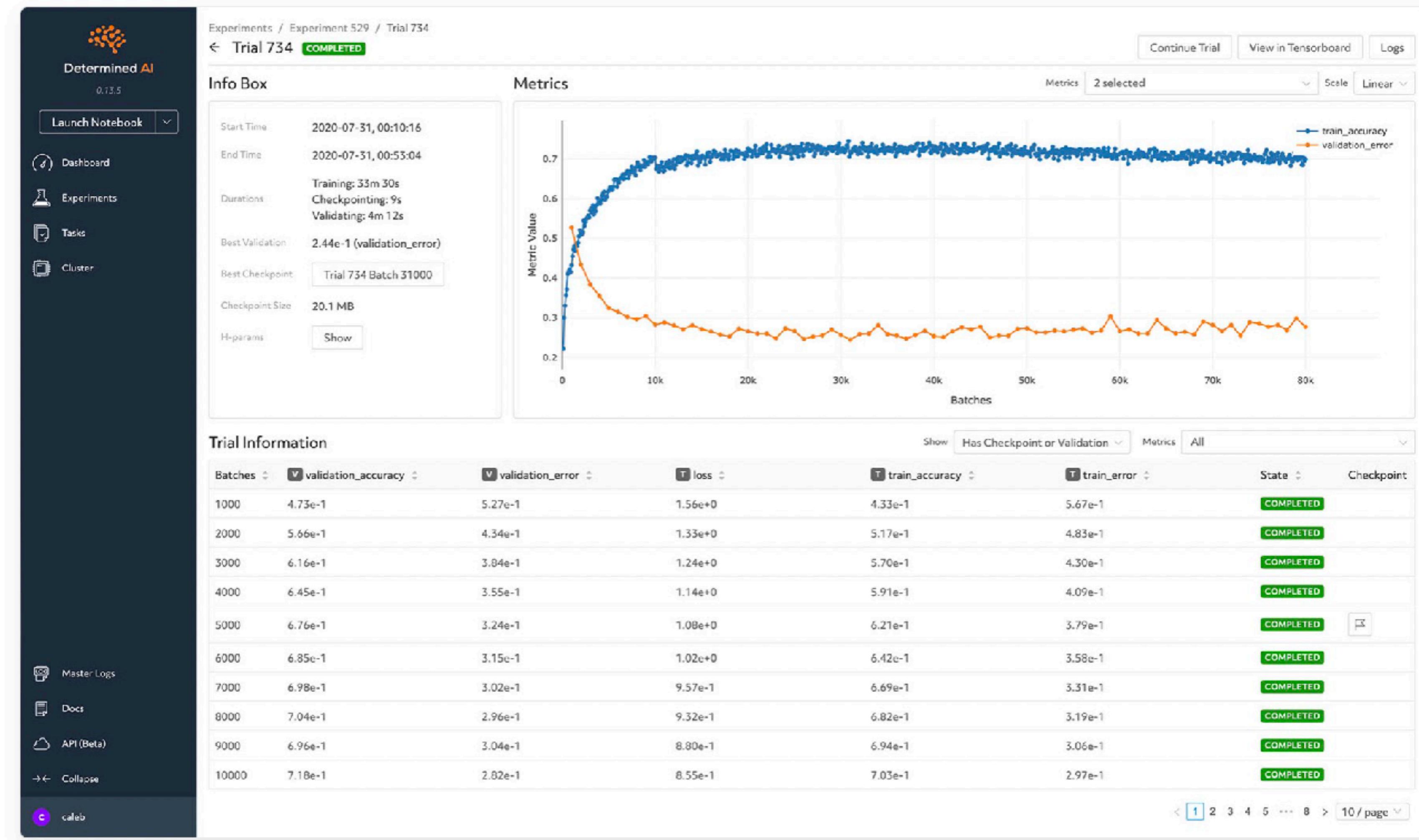
Feel free to click around!

| runs.summary["evaluation"] | | | | | |
|----------------------------|----|--------------|------------|-------|--|
| epoch | id | ground truth | prediction | smoke | |
| 1 | 1 | | | | |



Determined.ai

- Determined.ai also has experiment tracking





Other solutions

neptune.ai

Display mode: Mosaic (One in a row) Filter charts: epoch Charts legend: ID, test_accuracy, learning_rate

Play with a live Neptune app

| | TFKERAS-44 | TFKERAS-45 | TFKERAS-46 |
|----------------|------------------------|------------------------|------------------------|
| test_accuracy | 0.8587 | 0.8589 ↑ | 0.8655 ↑ |
| test_loss | 0.392458 | 0.399853 ↑ | 0.374164 ↓ |
| epoch_accuracy | 0.855208 | 0.835958 ↓ | 0.865042 ↑ |
| epoch_loss | 0.401329 | 0.460113 ↑ | 0.369486 ↓ |
| Tags | keras fashion-mnist | keras fashion-mnist | keras fashion-mnist |
| batch_loss | 0.401329 | 0.460113 ↑ | 0.369486 ↓ |
| batch_accuracy | 0.855208 | 0.835958 ↓ | 0.865042 ↑ |

Rows with diff only
Show cell changes

Try Comet Free [Talk to Us →](#)

Less friction, more ML

Comet's machine learning platform integrates with your existing infrastructure and tools so you can manage, visualize, and optimize models—from training runs to production monitoring.

Enterprise Products Customers Learn Pricing Company Login Create a Free Account

Marketing department / image captioning

Experiments Notes Files Archive Manage Reports Training dashboard Save View Options + Add

All Selected(3) Hidden

NAME

- happy_blackbird_1970
- dizzy_ocelot_9667
- spatial_work_4296
- occupational_butter_7080
- magnetic_lynx_6843
- productive_stadium_408
- architectural_percola_4445
- awkward_tuna_6446

Validation Loss by step

Accuracy

Accuracy

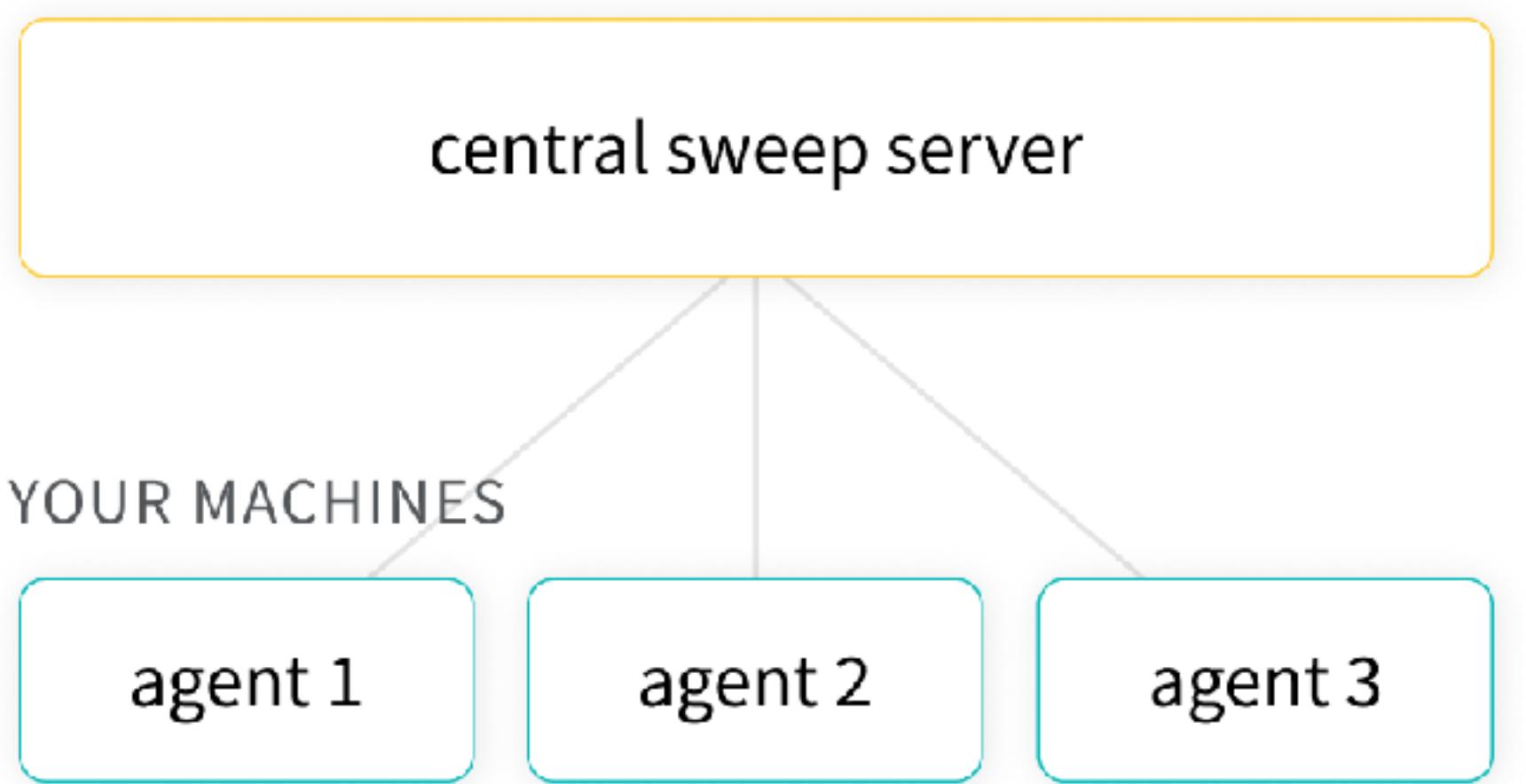


Hyperparameter Optimization

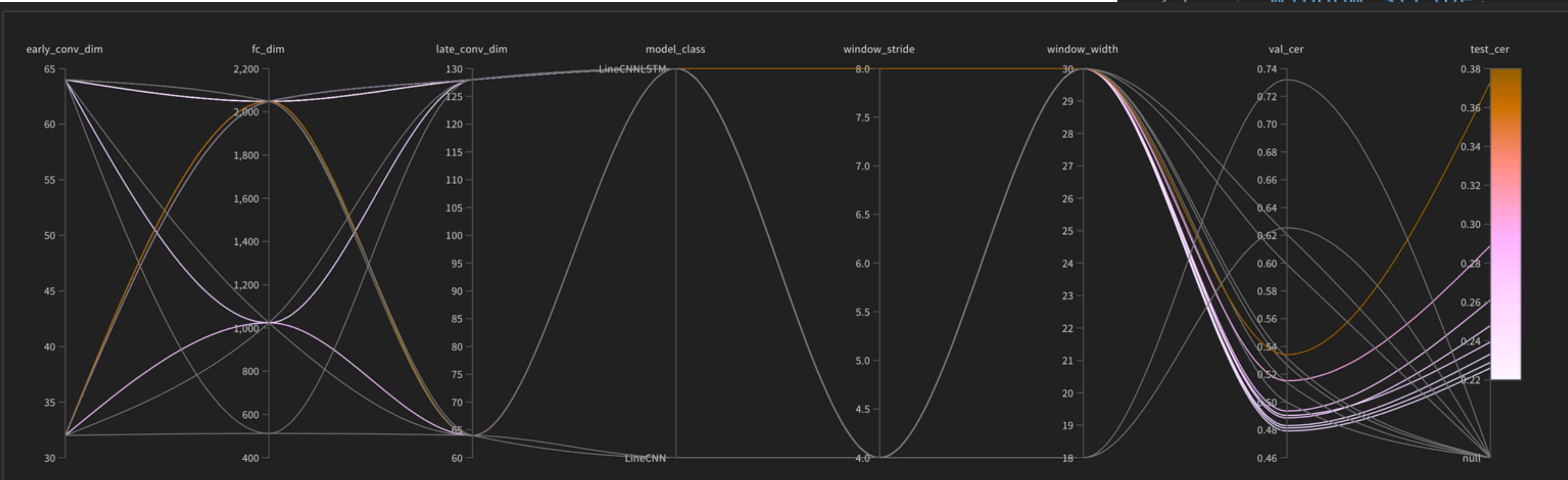
- Useful to have software that helps you search over hyper parameter settings.
- Could be as simple as being able to provide `--lr=(0.0001, 0.1) --num_layers=[128, 256, 512]` to training script
- Would be even better if settings were selected intelligently, and underperforming runs stopped early.

W&B Sweeps

OUR MACHINE



```
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:
22     ··· values: [18, 24, 30]
23   ·· window_stride:
```





Other solutions

- Also provided by SageMaker, Determined.ai, Ray, etc
- Usually not worth using anything specialized

“All-in-one”



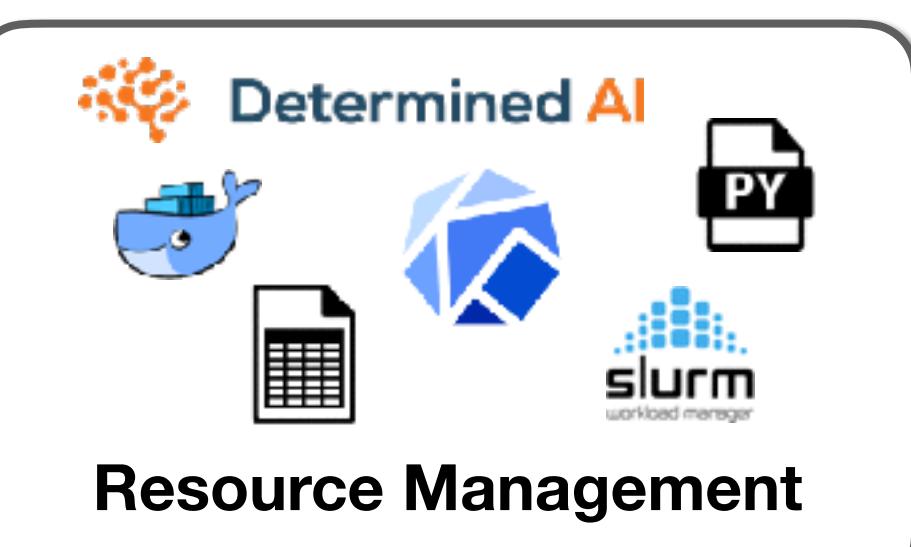
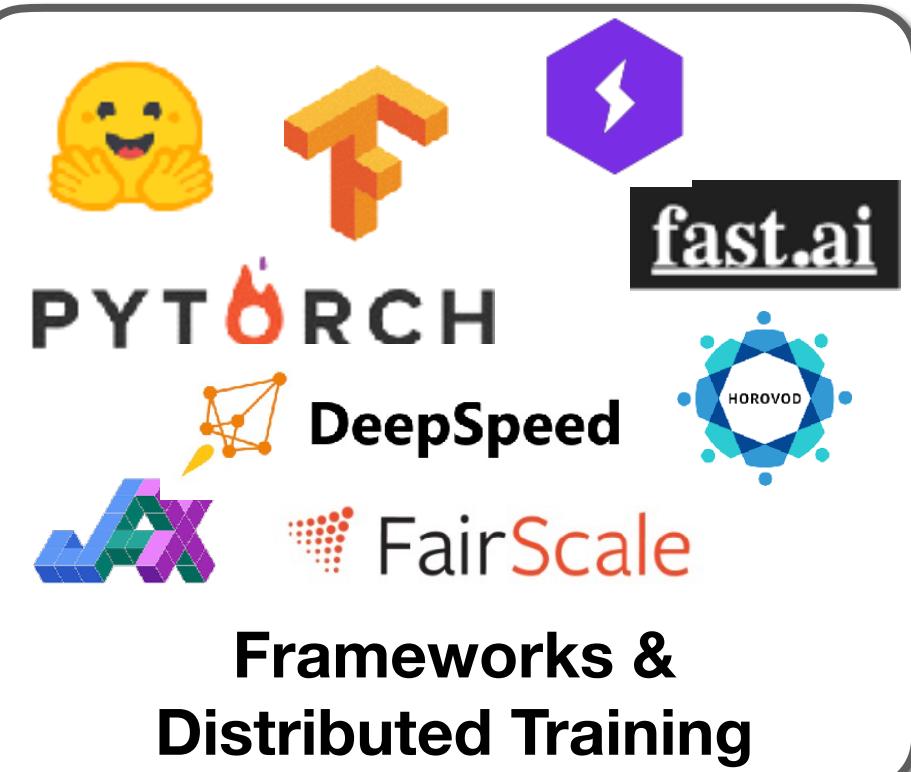
Amazon SageMaker

gradient^o
by Paperspace

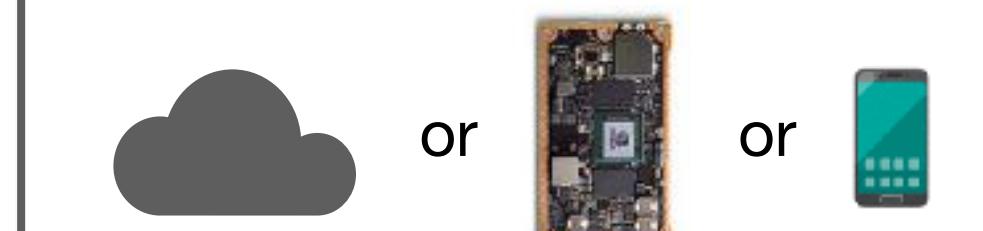
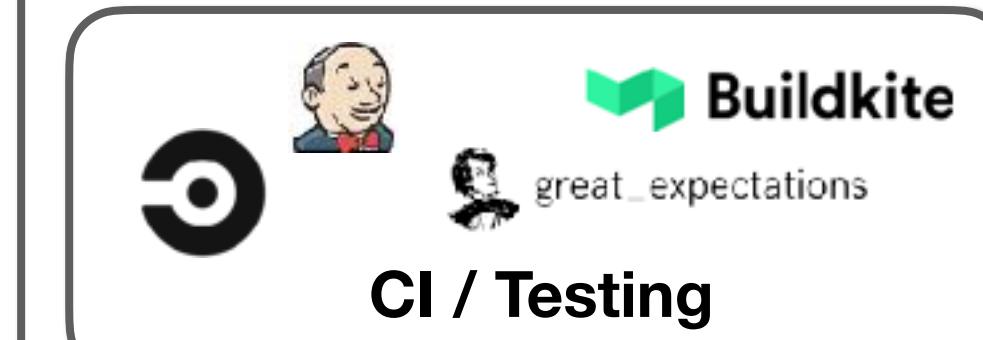
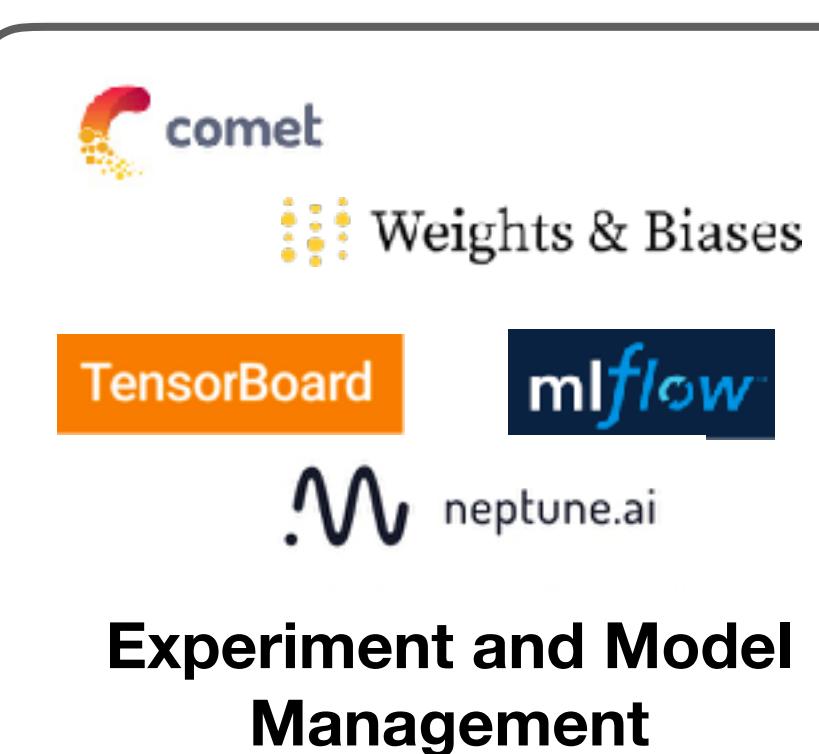
DOMINO
DATA LAB



Data



Development



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



Amazon SageMaker

Prepare →

SageMaker Ground Truth
Label training data for machine learning

SageMaker Data Wrangler NEW
Aggregate and prepare data for machine learning

SageMaker Processing
Built-in Python, BYO R/Spark

SageMaker Feature Store NEW
Store, update, retrieve, and share features

SageMaker Clarify NEW
Detect bias and understand model predictions

Build →

SageMaker Studio Notebooks
Jupyter notebooks with elastic compute and sharing

Built-in and Bring-your-own Algorithms
Dozens of optimized algorithms or bring your own

Local Mode
Test and prototype on your local machine

SageMaker Autopilot
Automatically create machine learning models with full visibility

SageMaker JumpStart NEW
Pre-built solutions for common use cases

Train & tune →

One-click Training
Distributed infrastructure management

SageMaker Experiments
Capture, organize, and compare every step

Automatic Model Tuning
Hyperparameter optimization

Distributed Training Libraries NEW
Training for large datasets and models

SageMaker Debugger NEW
Debug and profile training runs

Managed Spot Training
Reduce training cost by 90%

Deploy & manage →

One-click Deployment
Fully managed, ultra low latency, high throughput

Kubernetes & Kubeflow Integration
Simplify Kubernetes-based machine learning

Multi-Model Endpoints
Reduce cost by hosting multiple models per instance

SageMaker Model Monitor
Maintain accuracy of deployed models

SageMaker Edge Manager NEW
Manage and monitor models on edge devices

SageMaker Pipelines NEW
Workflow orchestration and automation

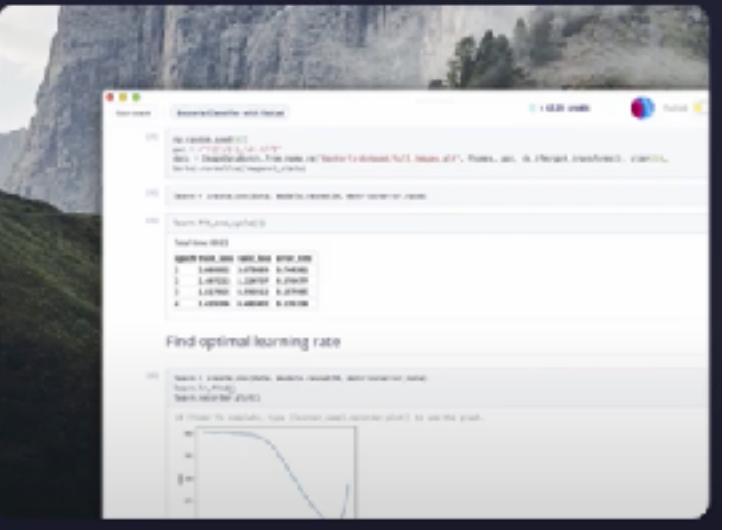
SageMaker Studio

Integrated development environment (IDE) for ML

01

Develop

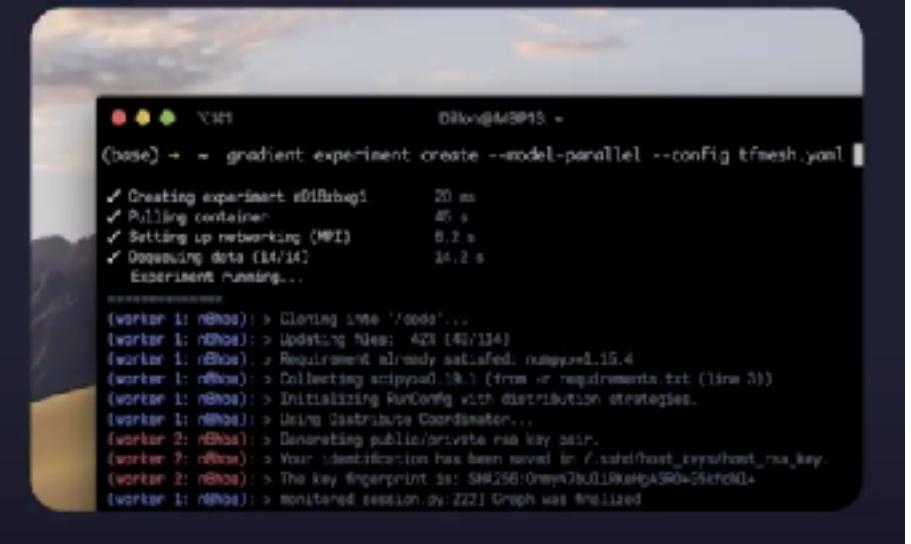
1-click Jupyter Notebooks with free GPUs and zero management



02

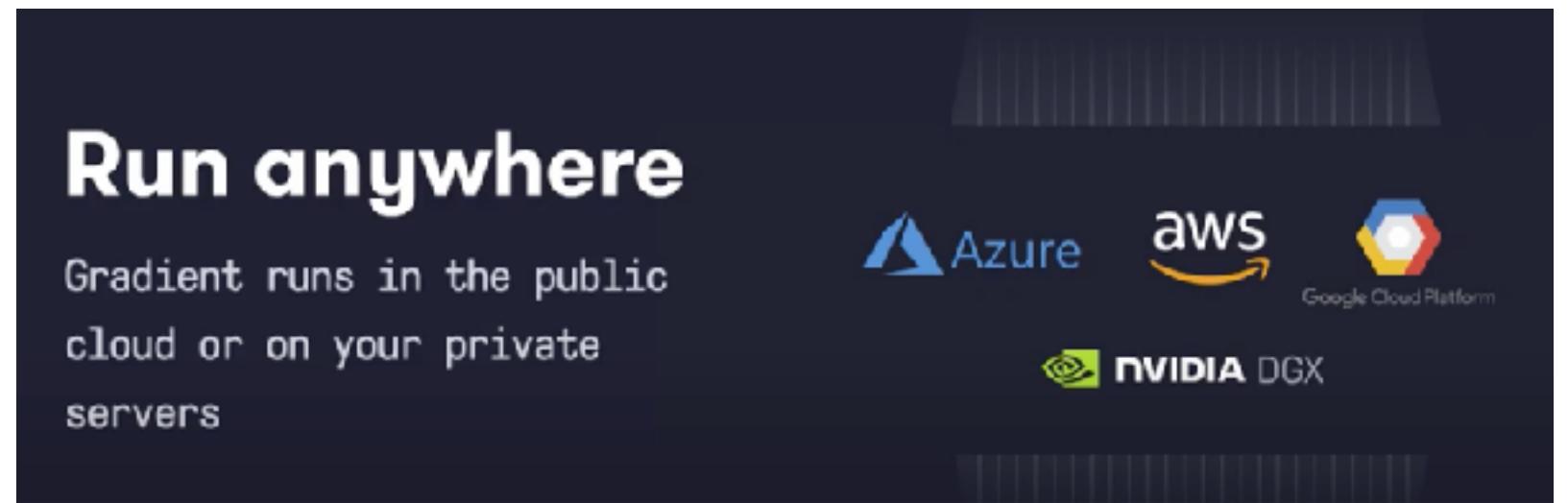
Train

Run single experiments to large-scale distributed training



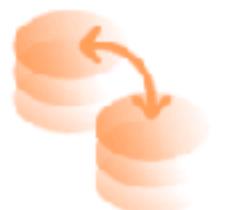
Notebooks

1-Click Jupyter Notebooks



Experiments

Develop, train, and track results



Datasets

Connect, version, and track data



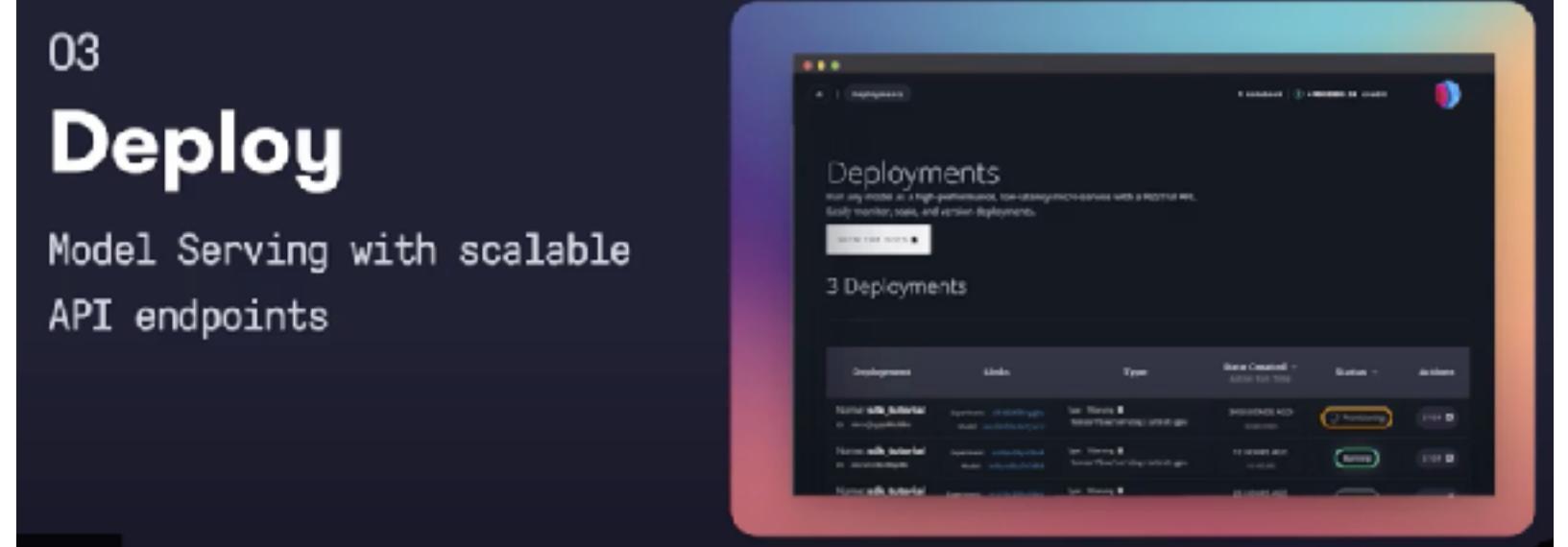
Models

Store, analyze and version models



Inference

Deploy models as API endpoints



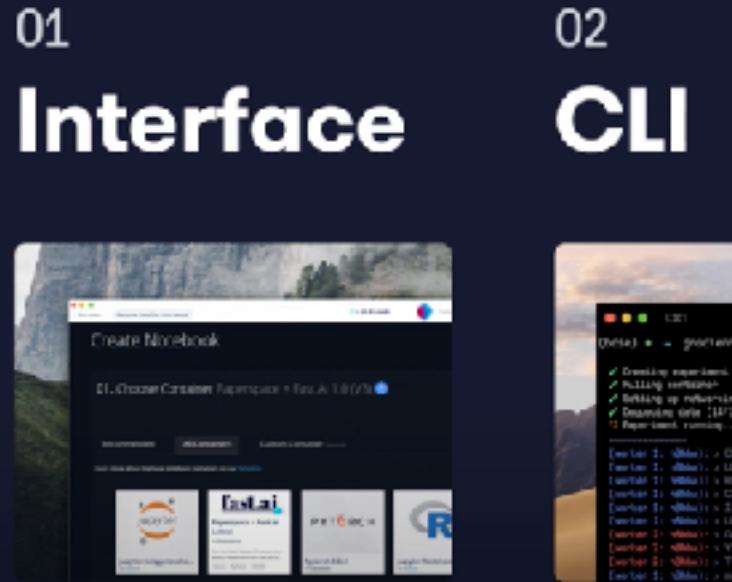
03

Deploy

Model Serving with scalable API endpoints

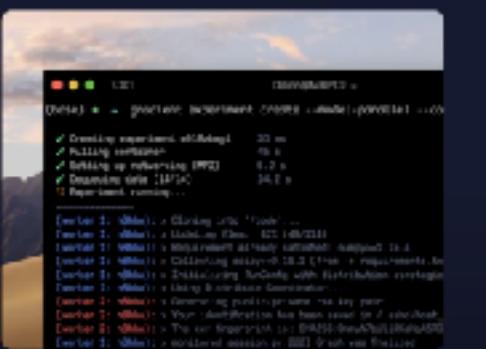
01

Interface



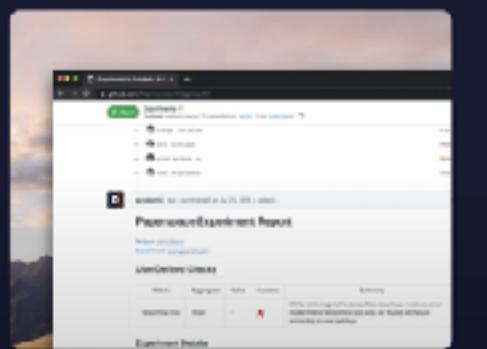
02

CLI



03

Github



04

Scale

Advanced Features & Pipelines



Auto-scaling



Monitoring



Versioning



Continuous



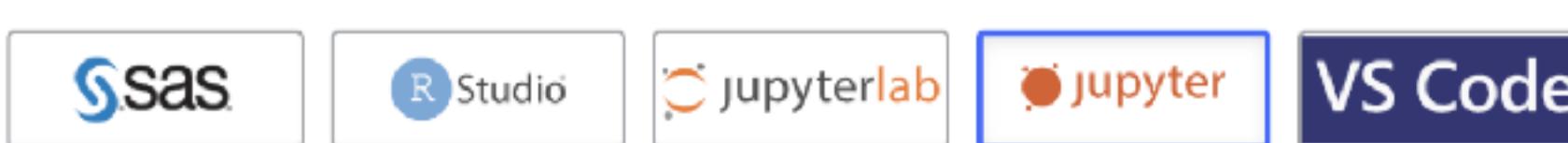
One place for your data science tools, apps, results, models, and knowledge.

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)

< 1 MIN

Large

16 cores · 122 GB RAM · \$0.0177/min

Medium

8 cores · 32 GB RAM · \$0.0064/min

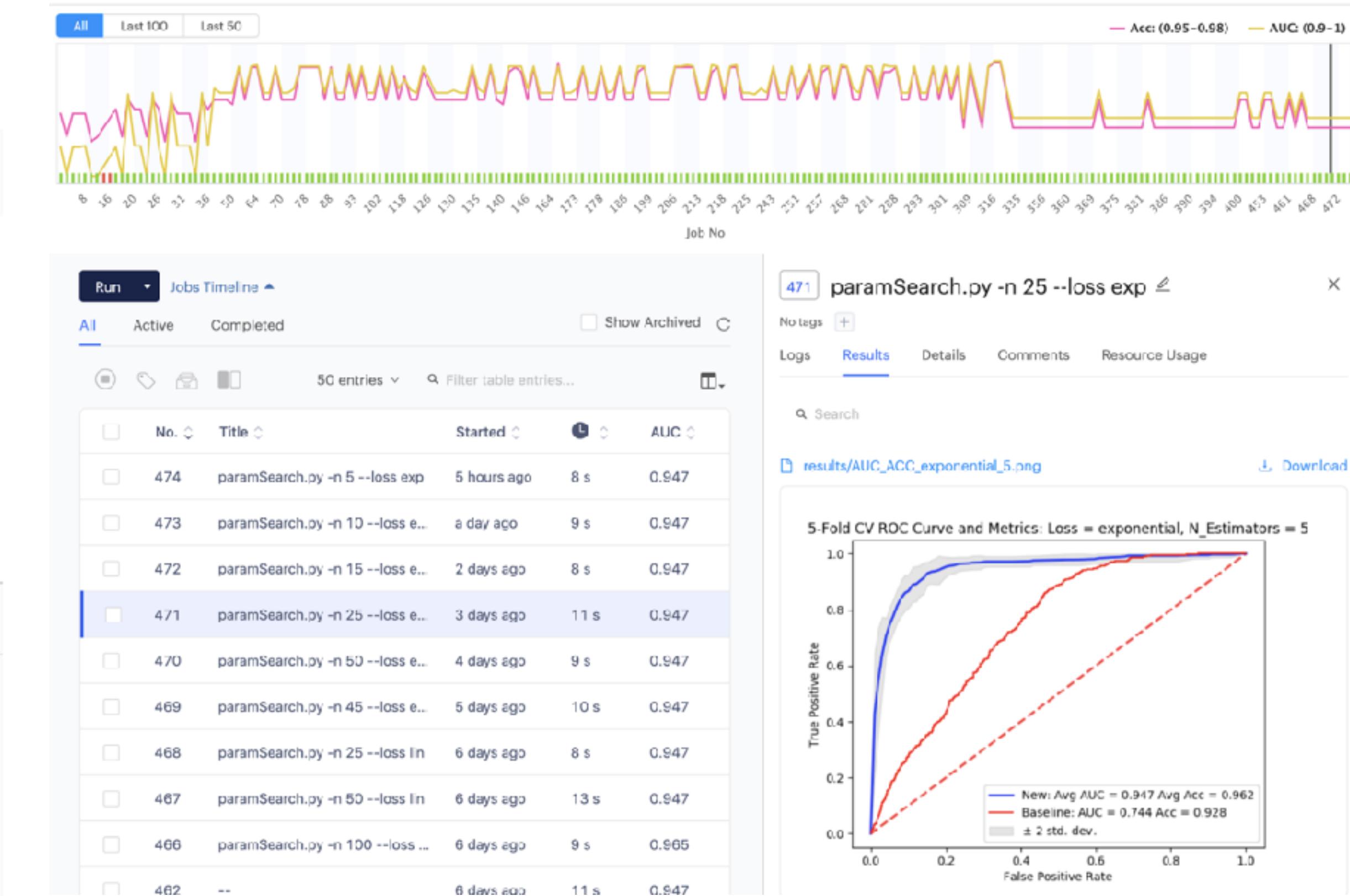
Small

4 cores · 16 GB RAM · \$0.0033/min

X1 32x Large

128 cores · 1952 GB RAM · \$0.0022/min

Track experiments



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
  }
}
```

Send

Window Size NA/6 predictions

since last Scheduled Test by Time by Data

Till Date Y M W D H MI S

Response | Instance Logs

```
model_time_items = 0,
"release": {
  "harness_version": "0.1",
  "model_version": "5d013eabc9e77c0007d4502e",
  "model_version_number": 8
},
"request_id": "QLEI7NWVMT2JG6AS",
"result": [
  0.9828162575212186
]
```

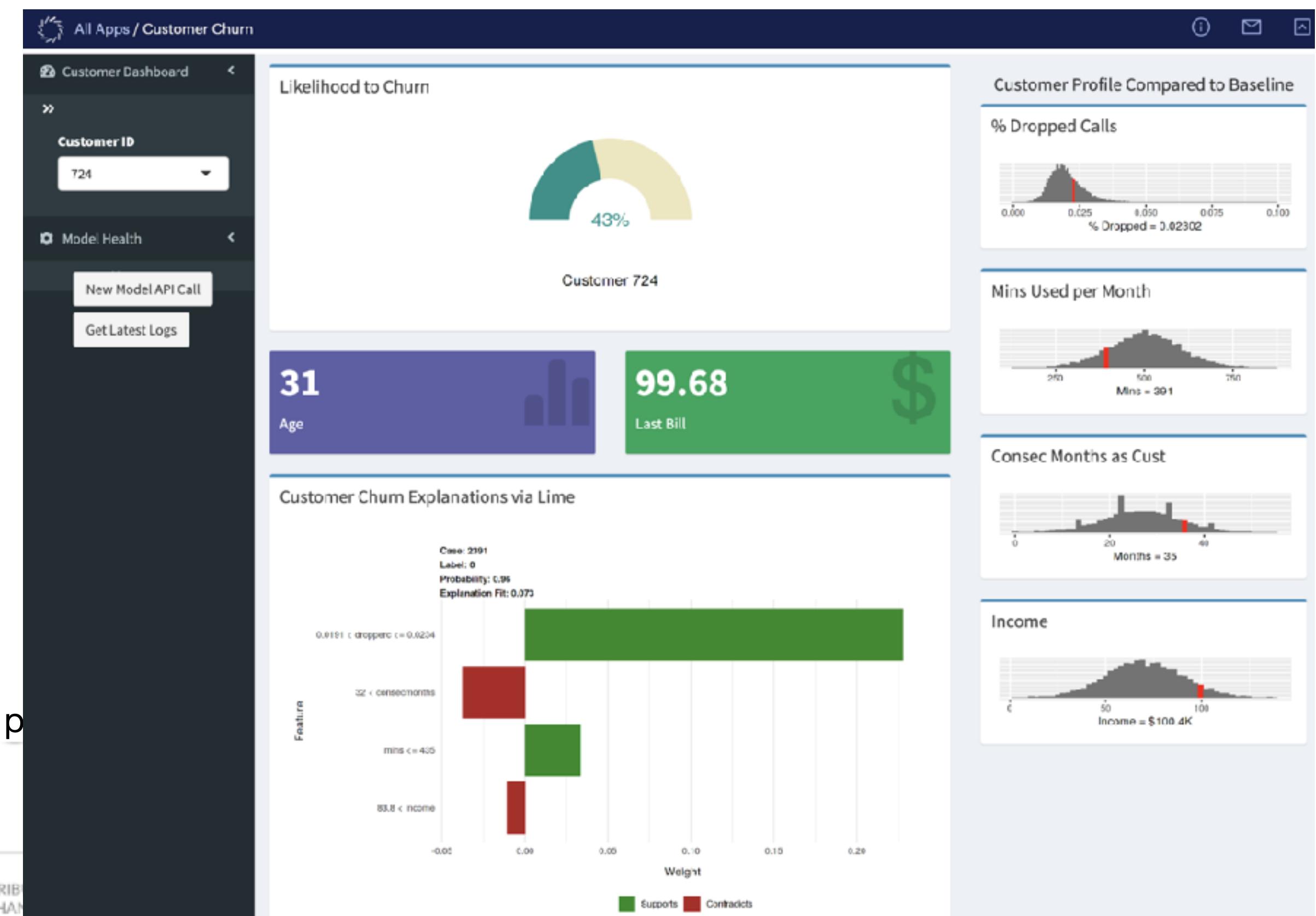
Model Drift

Search

| STATUS | FEATURE | TRAINING DATA | PREDICTION DATA | TEST TYPE | DISTRIBUTION CHANNEL | | |
|--------|-------------------------|---------------|-----------------|----------------------------|----------------------|--------------|-----|
| ● | 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 Infras | | | Kulback-Leibler Divergence | 0.1262 | Greater than | 0.3 |

Infras

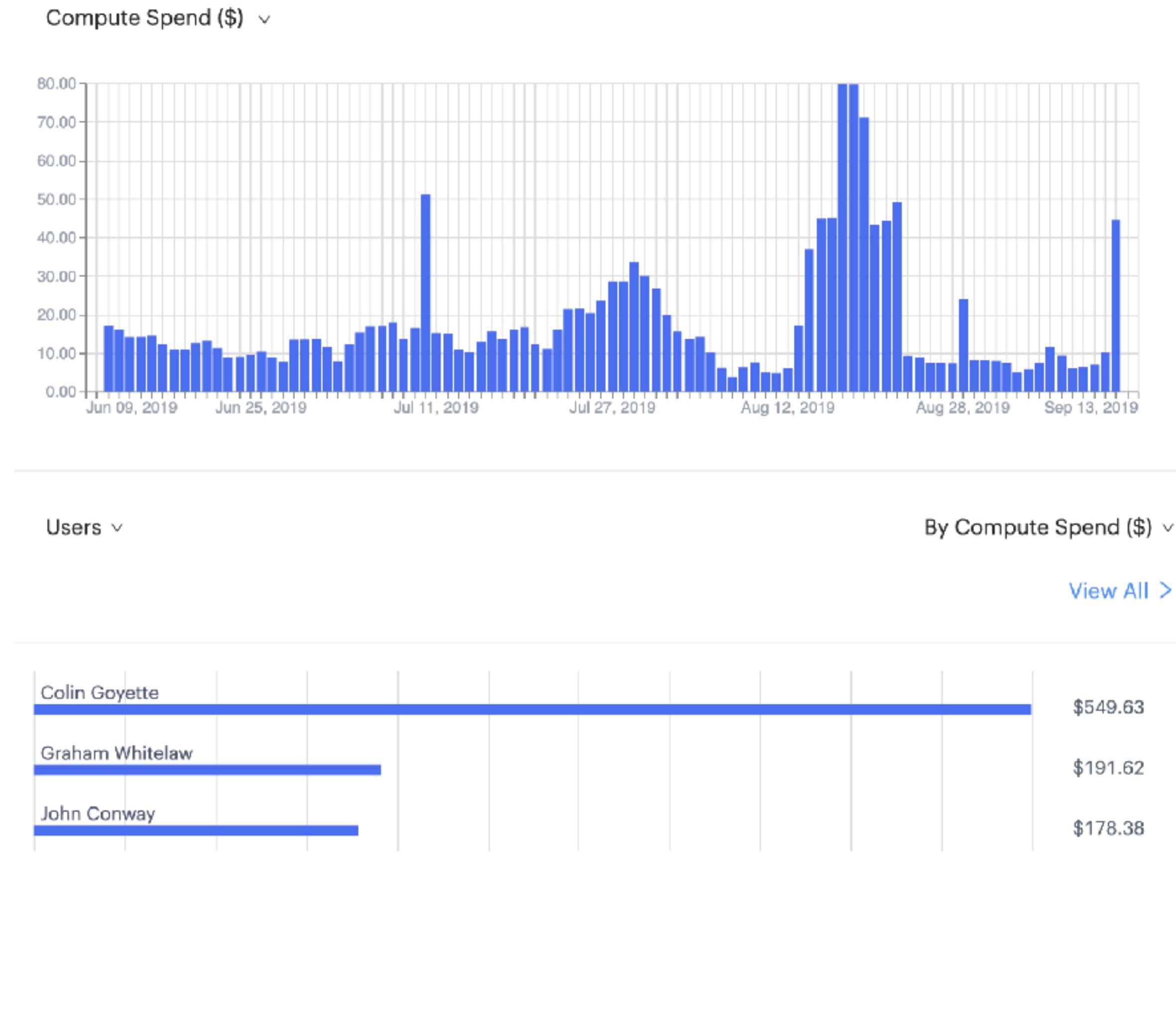
Publish applets



Monitor p



Overview



All projects in one place

Assets Portfolio

| Model APIs | Apps | Launchers | Schedulers |
|------------|------|-----------|------------|
| 02 | 09 | 13 | 05 |

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_Ma | 11 days ago | 4 | Dan S | | 10 views | 1k | 5k | Rob S |
| Auto MLPD | Model API | Character Re | 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 loca | 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 |



These could be good!

But let's wait to learn more about Data Management and Deployment in the weeks ahead.



Thank you!