



FSDL 2022

# Data Management

Sergey Karayev

AUGUST 29, 2022



**Katherine Scott**  
@kscottz



One of the biggest failures I see in junior ML/CV engineers is a complete lack of interest in building data sets. While it is boring grunt work I think there is so much to be learned in putting together a dataset. It is like half the problem.

11:50 AM - Feb 1, 2019



**abhishek**  
@abhi1thakur

...

Everyone wants to train models, no one wants to create data. When was the last time you labeled/collected data yourself to train a model? :)

3:36 AM · Dec 17, 2021 · Twitter Web App

149 Retweets 56 Quote Tweets 1,523 Likes



**Vicki Boykis**  
@vboykis



Have been extremely curious about this for a while now, so I decided to create a poll.

"As someone titled 'data scientist' in 2019, I spend most of (60%+) my time:"

("Other") also welcome, add it in the replies.

189 8:17 AM - Jan 28, 2019



6% Picking features/models

67% Cleaning data/Moving data

4% Deploying models in prod

23% Analyzing/presenting data

2,116 votes • Final results

<https://veekaybee.github.io/2019/02/13/data-science-is-different/>



# Key Points

- Spend 10x as much time exploring the data as you would like to
- Fixing/adding/augmenting data is **usually** the best way to improve performance
- Keep it simple!



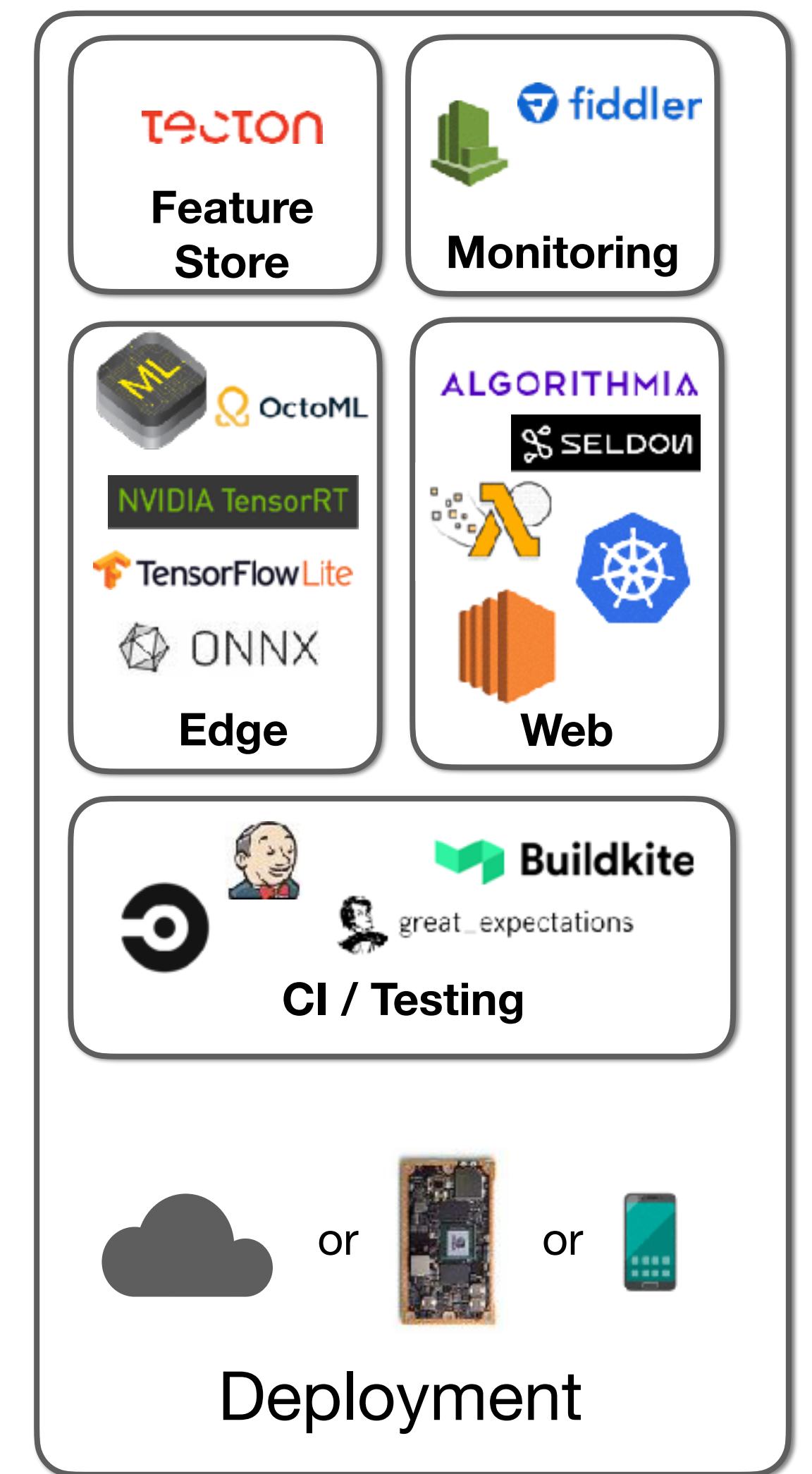
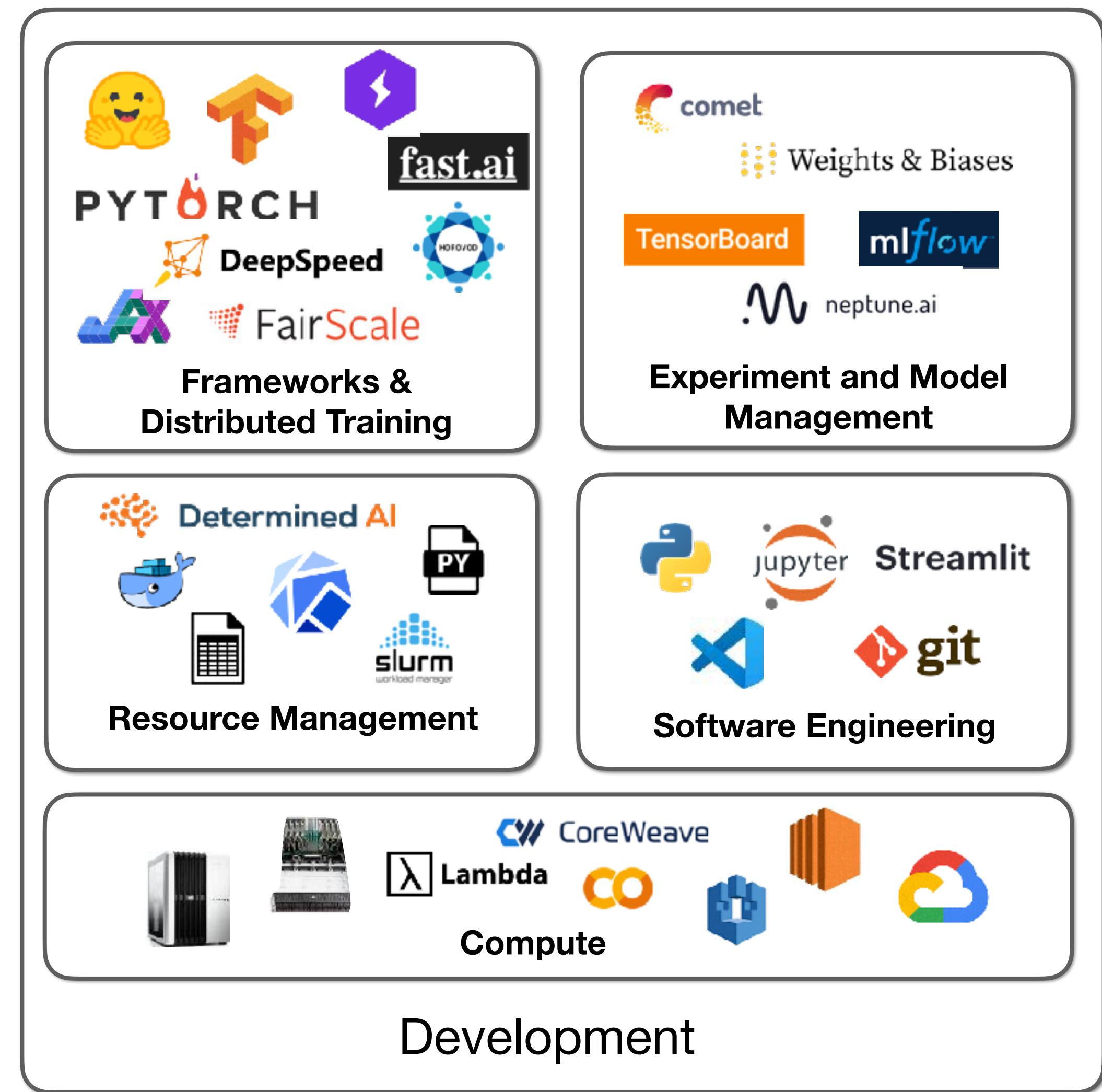
“All-in-one”



Amazon SageMaker

gradient<sup>o</sup>  
by Paperspace

DOMINO  
DATA LAB



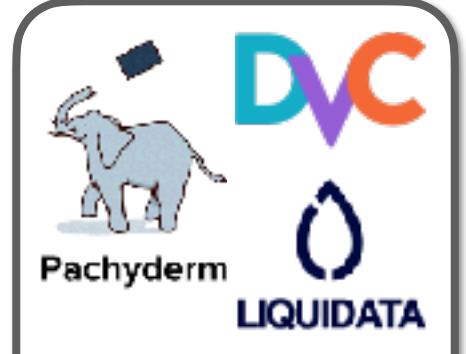
“All-in-one”



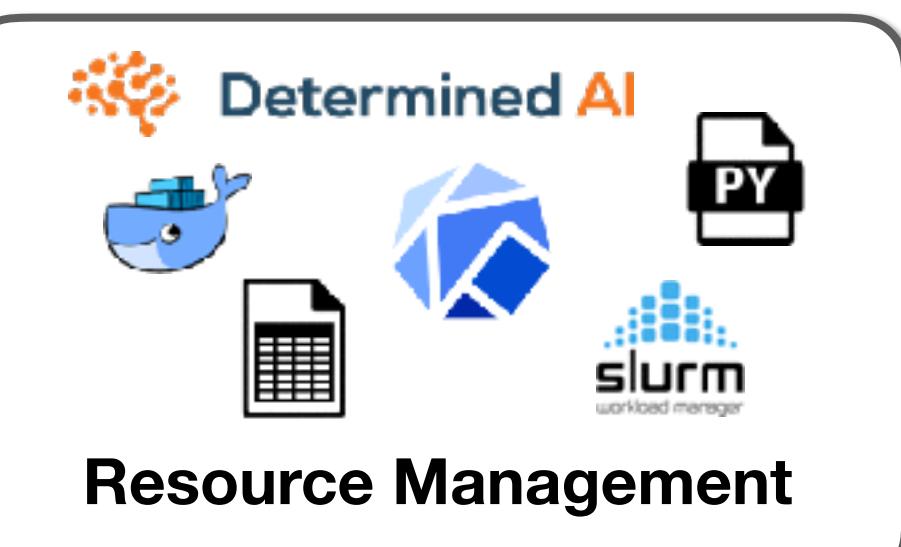
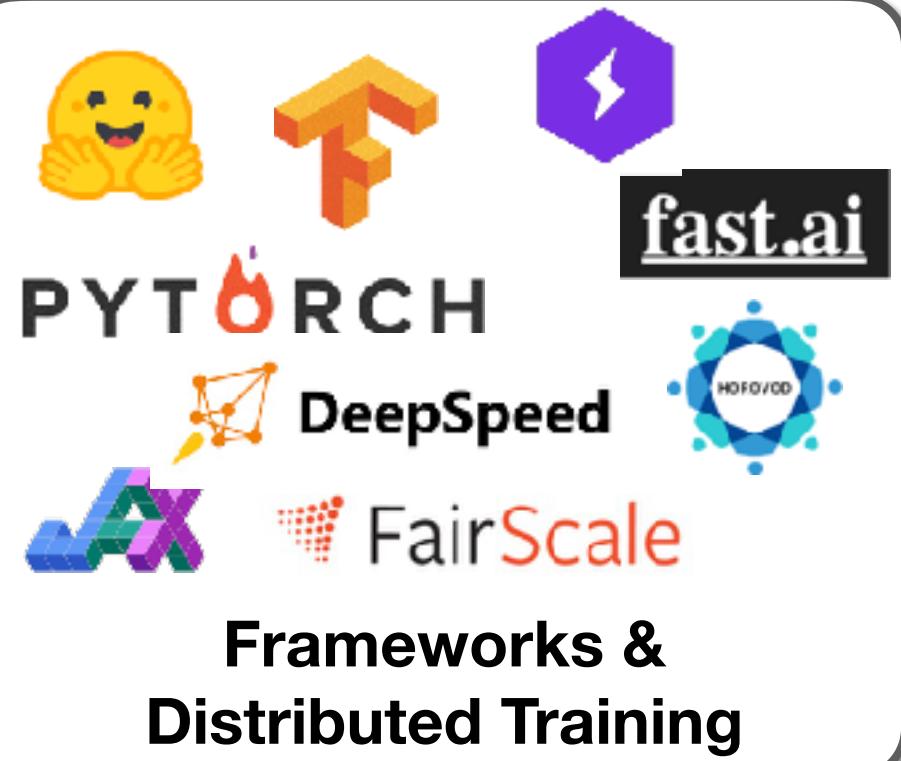
Amazon SageMaker



DOMINO  
DATA LAB



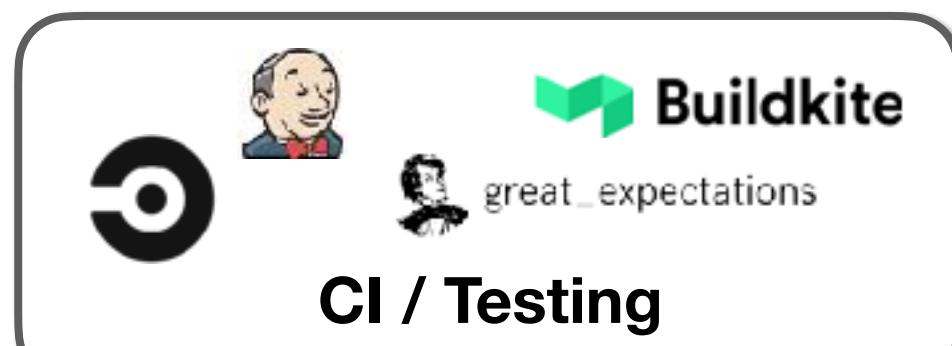
Data



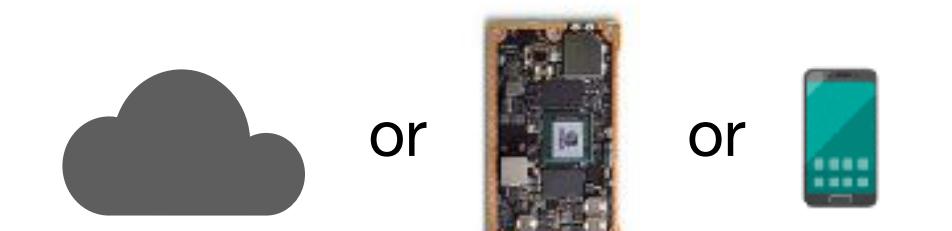
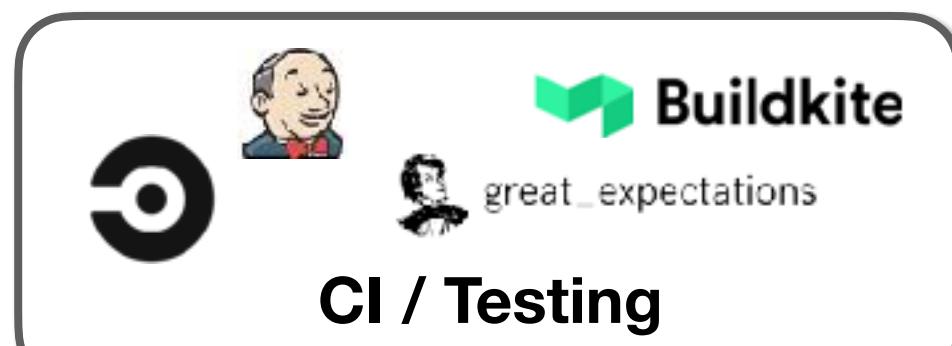
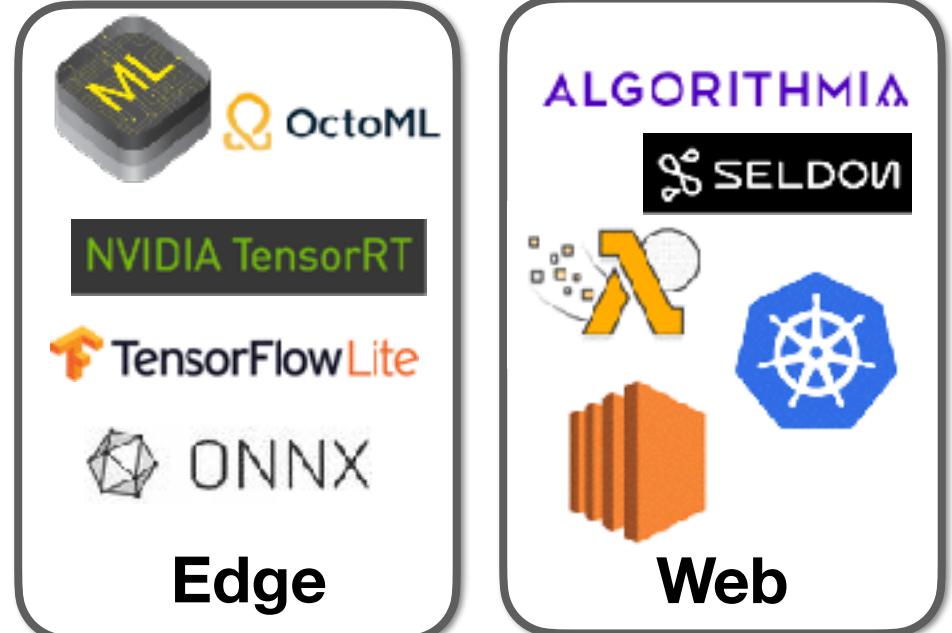
Development



Experiment and Model Management



Deployment



# Many possibilities

## Data Sources

 Images

 Text Corpus

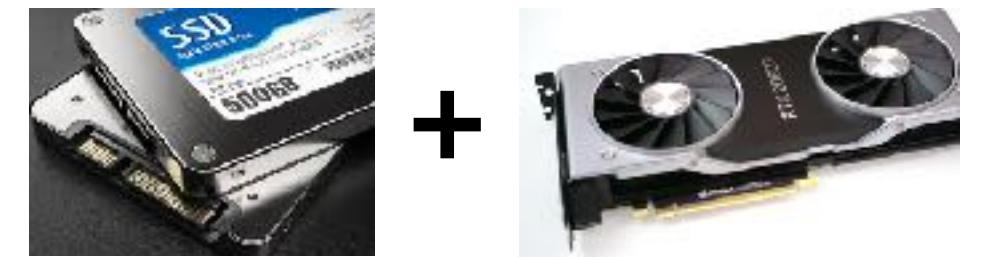
 Logs

 DB records

Different for every project / company!

## Training

Local Filesystem



GPU

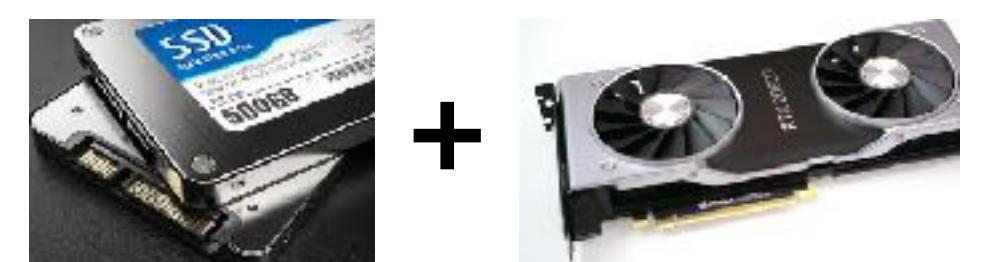
# Many possibilities

## Data Sources

 Images

## Training

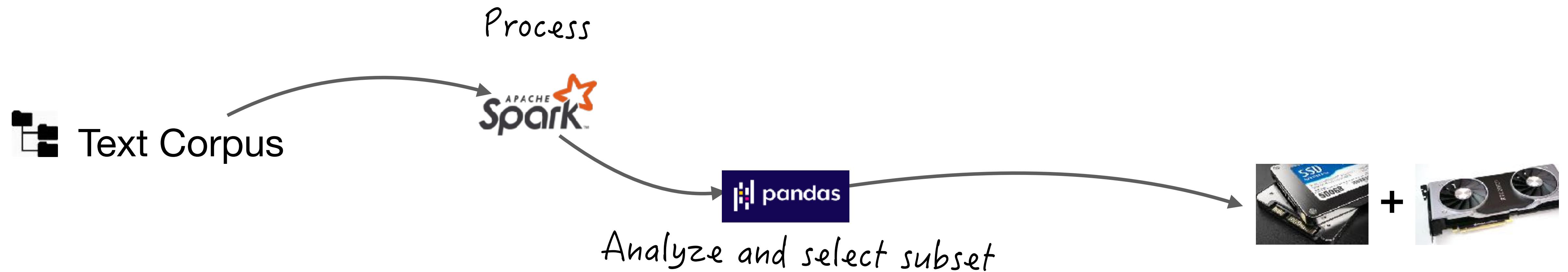
*Simply Download*



# Many possibilities

## Data Sources

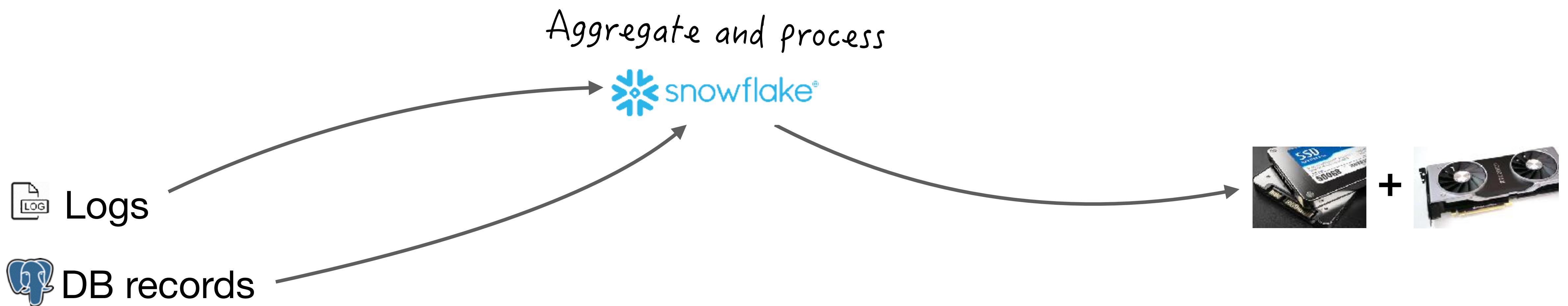
## Training



# Many possibilities

## Data Sources

## Training



# Many possibilities

## Data Sources

 Images

 Text Corpus

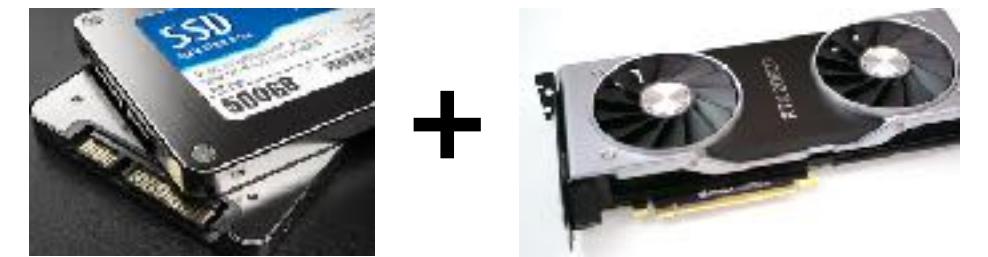
 Logs

 DB records

Different for every project / company!

## Training

Local Filesystem



GPU



# The Basics

- Filesystem
- Object Storage
- Databases

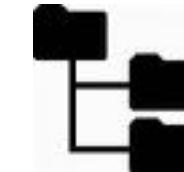


# The Basics

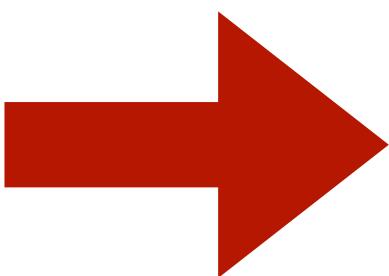
- **Filesystem**
- Object Storage
- Databases



# Filesystem



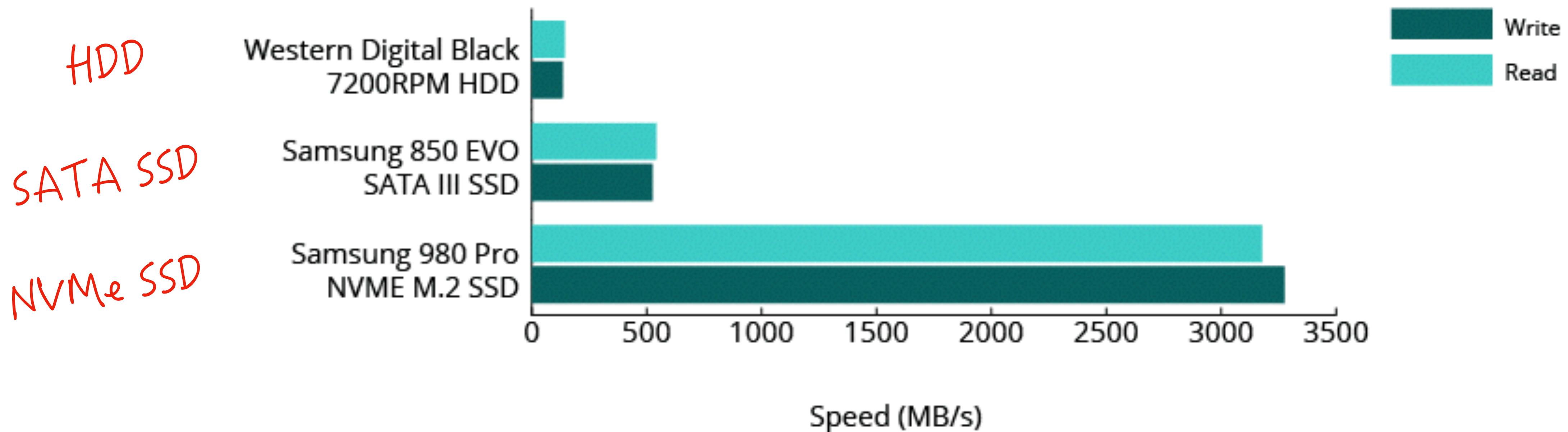
- Fundamental unit is a "file", which can be text or binary, is not versioned, and is easily overwritten.
- On a disk that's connected to your machine
  - Physically connected on-prem
  - "Attached" in the cloud
  - Or even distributed (e.g. HDFS)



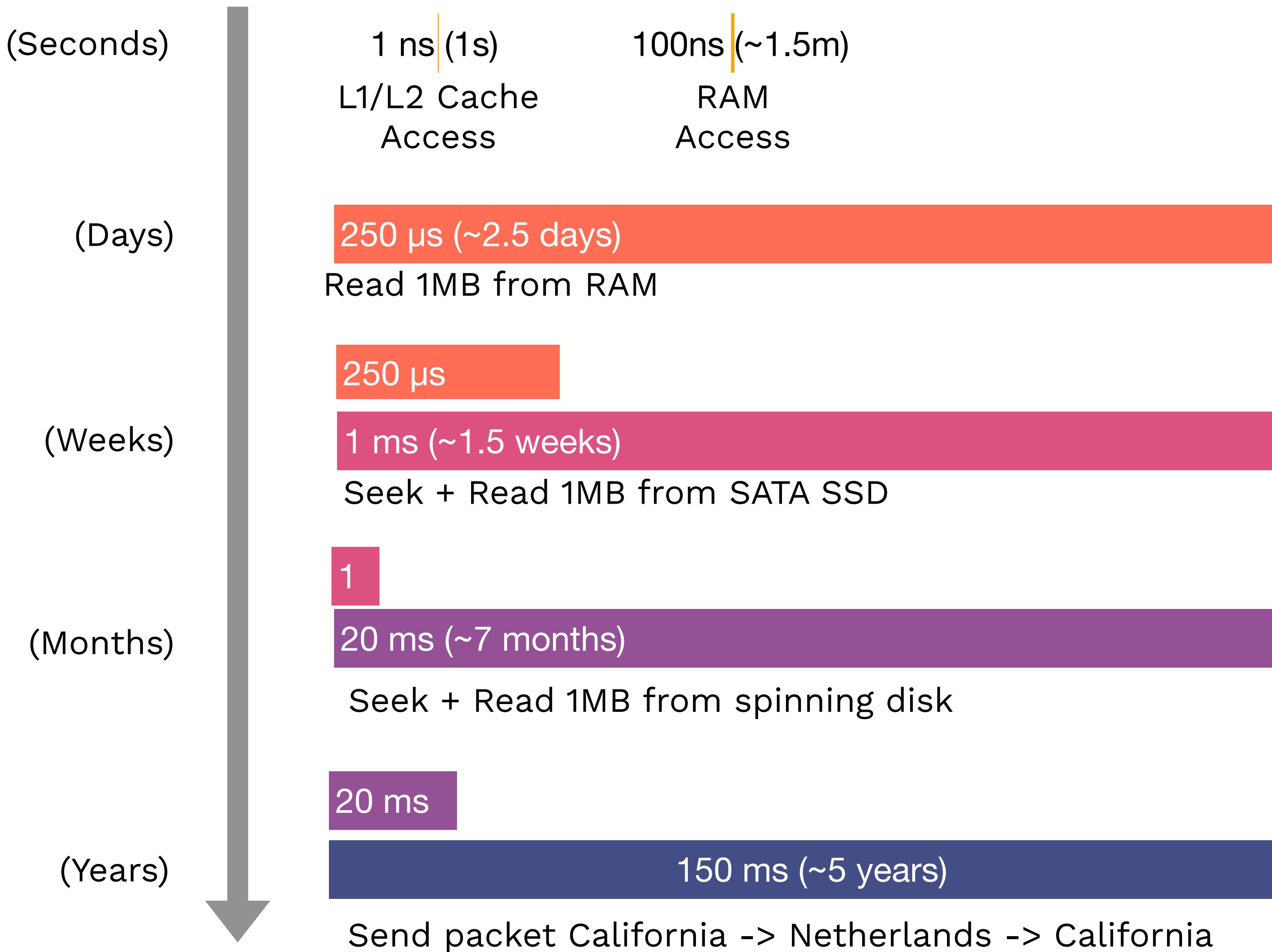


# Local Disk Speeds

Almost 2 orders of magnitude difference!



# Latency numbers you should know (with human-scale numbers in paren)



Please send GPU timing  
info!



# Local Data Format

- Binary data (images, audio):
  - Just use standard formats (e.g. JPEG)
- For metadata (labels) / tabular data / text data:
  - Compressed json/txt file(s) are just fine
  - Parquet is a table format that's fast, compact, and widely used



# The Basics

- Filesystem
- **Object Storage**
- Databases

# Object Storage



- An API over the filesystem.
- Fundamental unit is an "object". Usually binary: image, sound file, etc.
- Versioning, redundancy can be built into the service.
- Not as fast as local, but fast enough within the cloud

e.g. s3://my-bucket-name/my-file-name.jpg



# The Basics

- Filesystem
- Object Storage
- **Databases**

# Database



- Persistent, fast, scalable storage and retrieval of structured data
- Mental model: everything is actually in RAM, but software ensures that everything is persisted to disk.
- Not for binary data! Store object-store URLs instead.
- Postgres is the right choice most of the time. Supports unstructured JSON.
  - SQLite is perfectly good for small projects.



# You should probably be using a database

- Code that deals with collections of objects that reference each other (e.g. a Text is from a Document, which has an Author) **will eventually implement a crappy database**
- Using a database from the beginning will likely save time
- Many MLOps tools are databases at their core (e.g W&B is a DB of experiments, HuggingFace Hub is a DB of models, Label Studio is a DB of labels)

# Data Warehouse

- Store for Online **Analytical** Processing (OLAP)

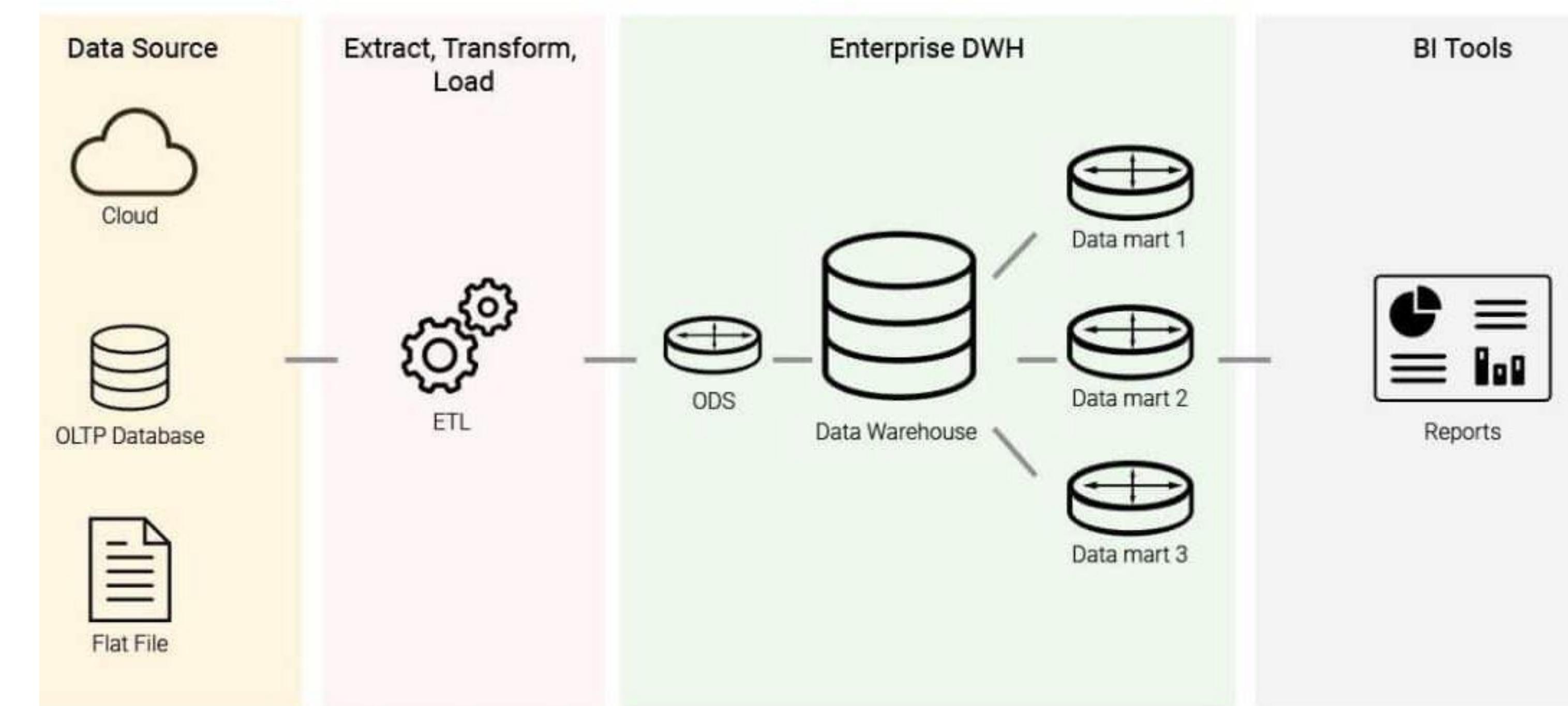
Google Cloud  
BigQuery

snowflake®

amazon  
REDSHIFT

- vs Databases for Online **Transaction** Processing (OLTP)
- Extract-Transform-Load (**ETL**) data in

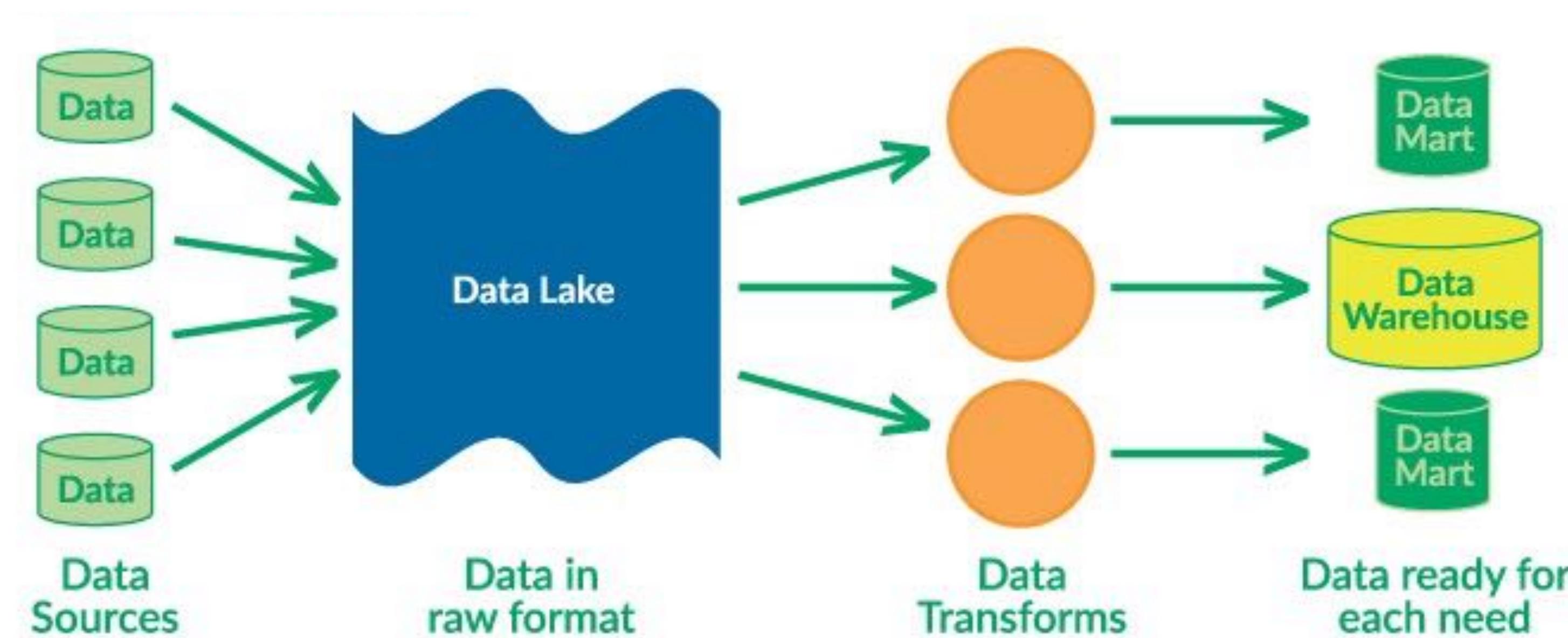
- OLAPs: usually column-oriented, for queries like mean length of comments.text over last 30 days
- OLTPs: usually row-oriented, for queries like select comments where user\_id=123





# Data Lake

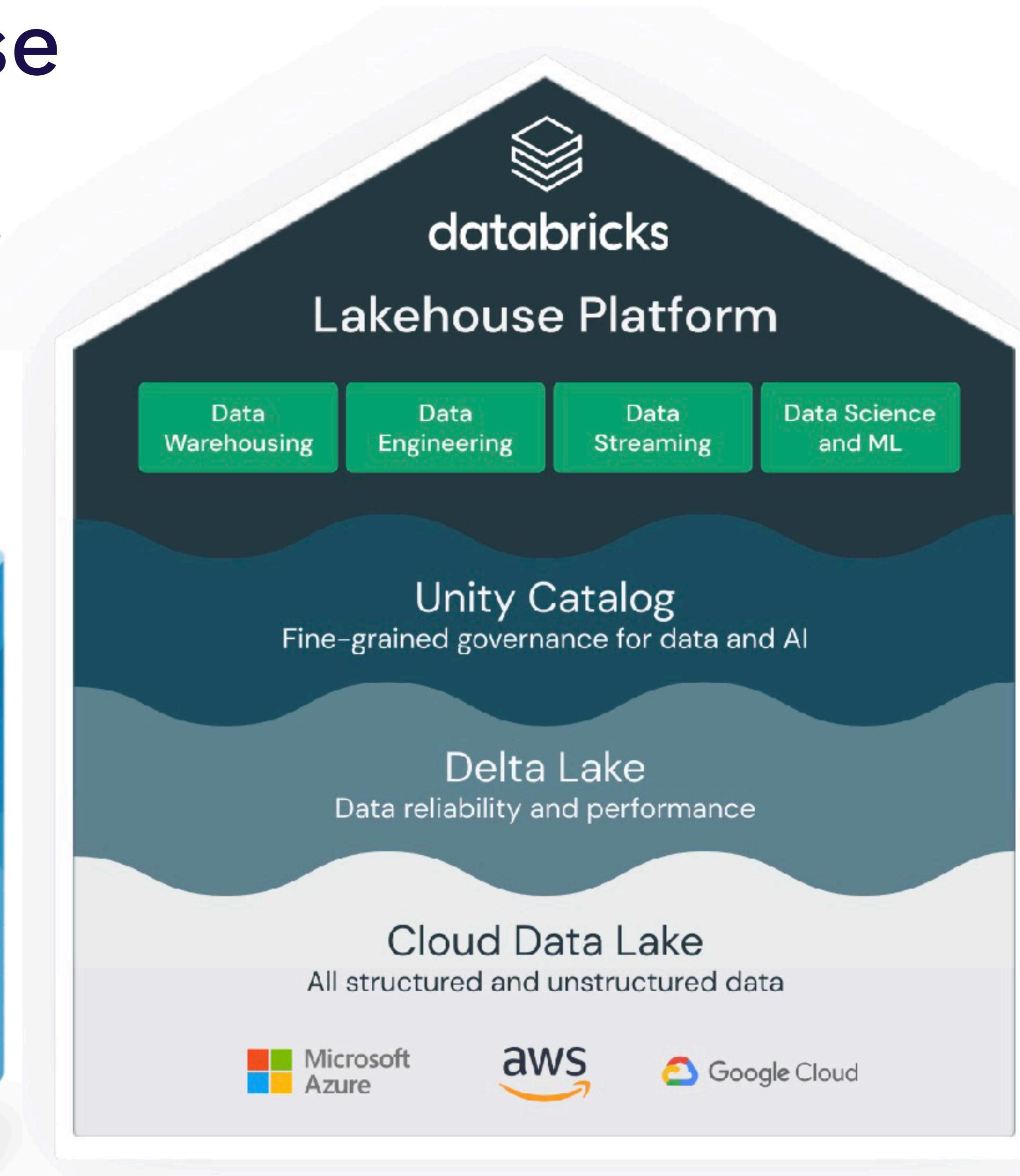
- Unstructured aggregation of data from multiple sources, e.g. databases, logs, expensive data transformations.
- **ELT**: dump everything in, then transform for specific needs later.



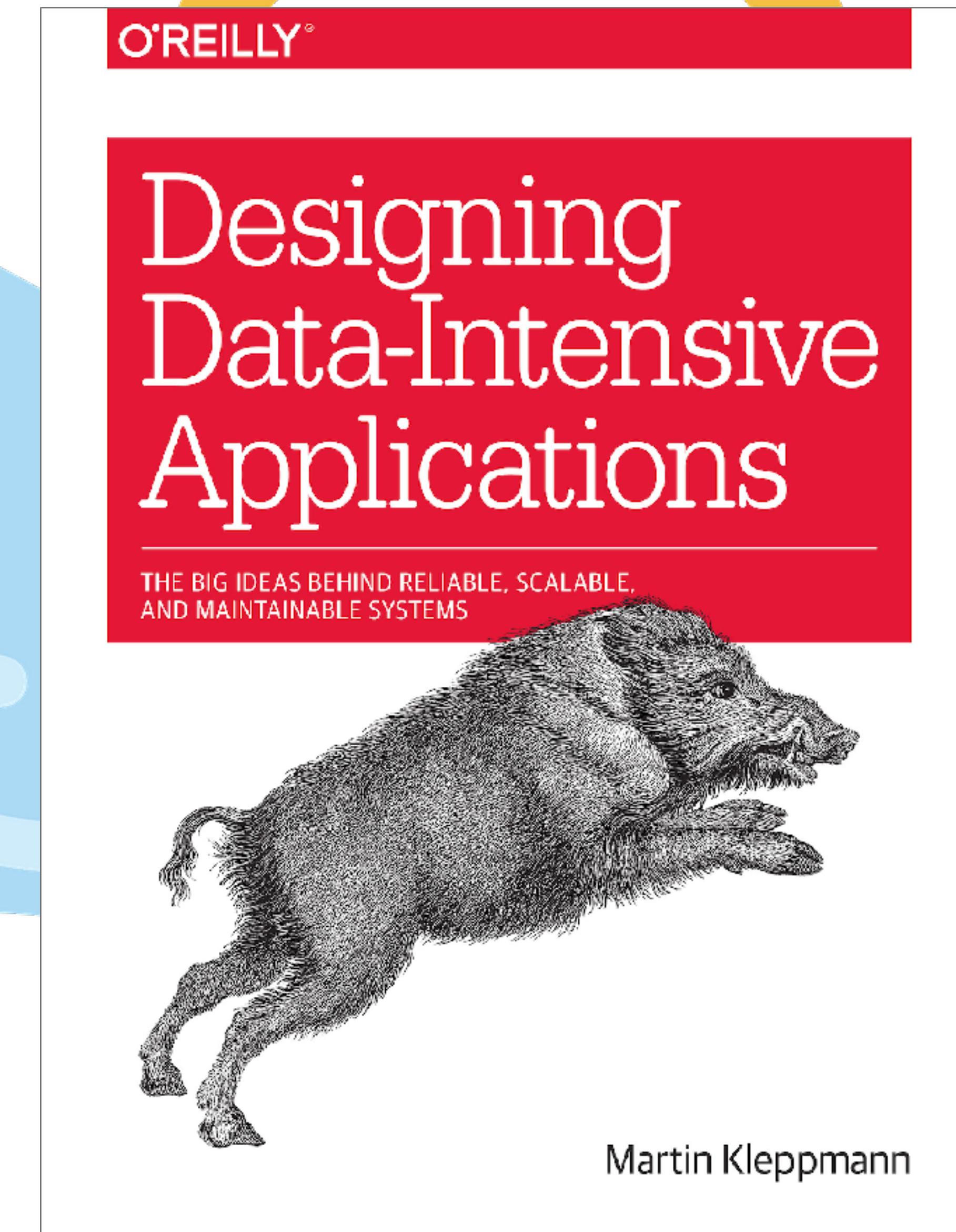


# Trend: both Lake and House

- Both structured and unstructured data together



# If you're interested in this stuff



<https://dataintensive.net>

“All-in-one”



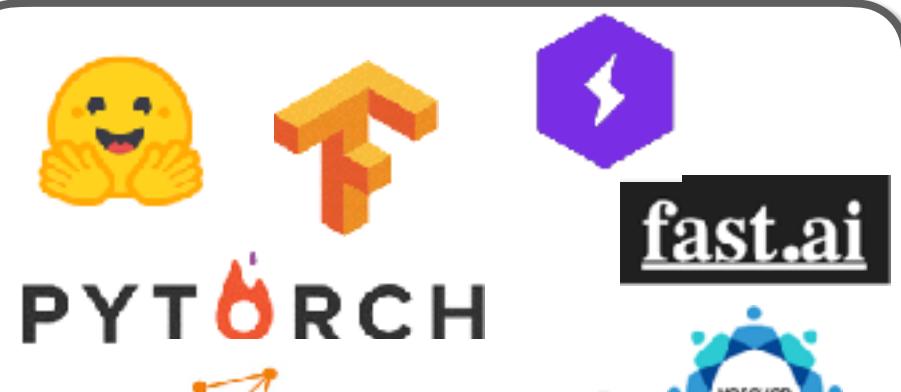
Amazon SageMaker

gradient<sup>o</sup>  
by Paperspace

DOMINO  
DATA LAB



Data



Frameworks &  
Distributed Training



Resource Management



CoreWeave

Compute

Development



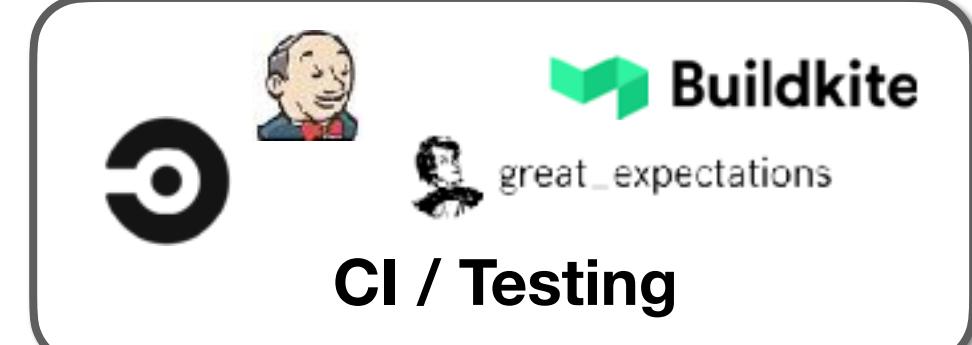
Experiment and Model  
Management



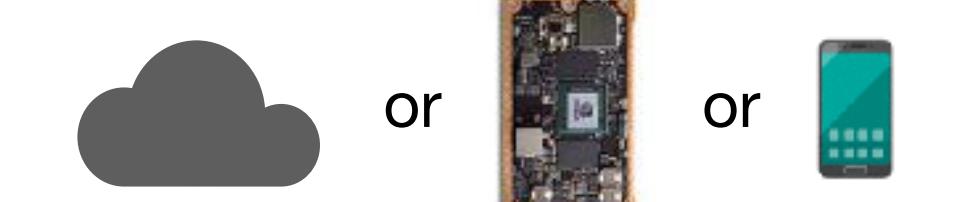
Software Engineering



Edge



CI / Testing



Deployment





# SQL and DataFrames

- Most data solutions use SQL.  
Some, like Databricks, use  
DataFrames.
- SQL is the standard interface  
for structured data.
- Pandas is the main DataFrame  
in the Python ecosystem.
- Our advice: become fluent in  
both

```
SELECT smoker, day, COUNT(*), AVG(tip)
FROM tips
GROUP BY smoker, day;
/*
smoker day
No   Fri      4  2.812500
      Sat     45  3.102889
      Sun     57  3.167895
      Thur    45  2.673778
Yes  Fri      15  2.714000
      Sat     42  2.875476
      Sun     19  3.516842
      Thur    17  3.030000
*/
```

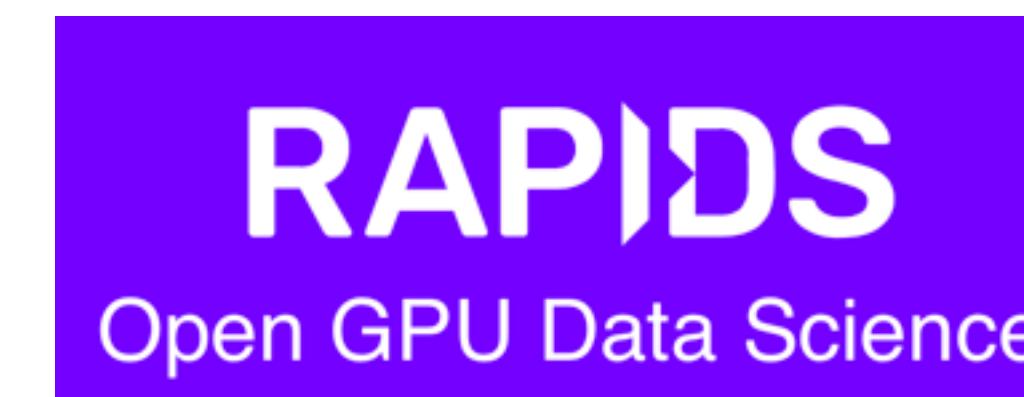
```
In [22]: tips.groupby(["smoker", "day"]).agg({"tip": [np.size, np.mean]})  
Out[22]:
```

		tip	
		size	mean
smoker	day		
No	Fri	4.0	2.812500
	Sat	45.0	3.102889
	Sun	57.0	3.167895
	Thur	45.0	2.673778
Yes	Fri	15.0	2.714000
	Sat	42.0	2.875476
	Sun	19.0	3.516842
	Thur	17.0	3.030000



# Pandas

- The workhorse of Python data science
- + DASK DataFrames parallelize Pandas operations over cores
- + RAPIDS to do Pandas operations on GPUs

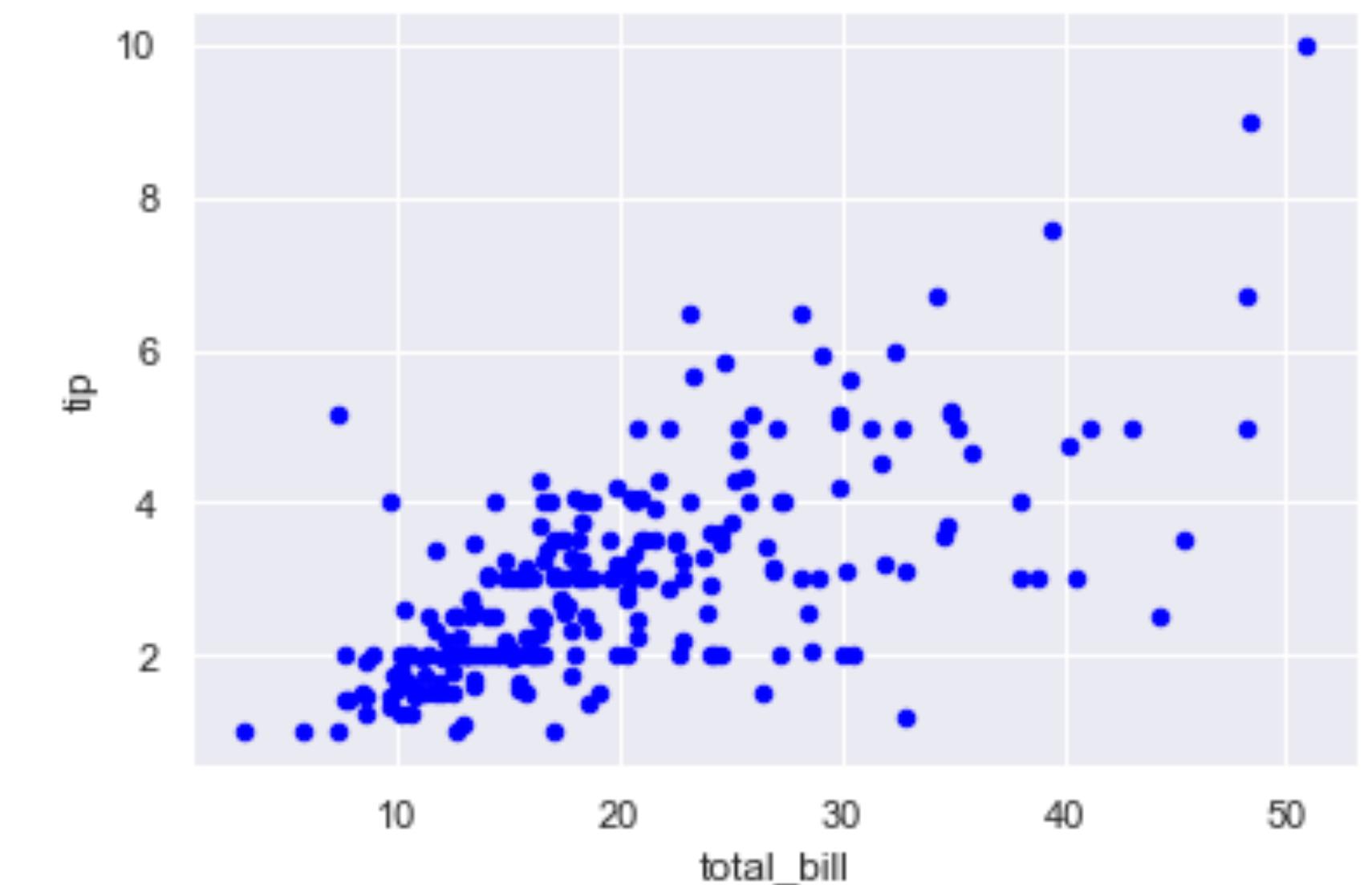


In [12]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

tips = pd.read_csv("tips.csv")
tips.plot.scatter(x="total_bill", y="tip", c="Blue")
```

Out[12]: &lt;matplotlib.axes.\_subplots.AxesSubplot at 0xc5120b0&gt;



<https://projectcodeed.blogspot.com/2019/08/setting-up-jupyter-notebooks-for-data.html>

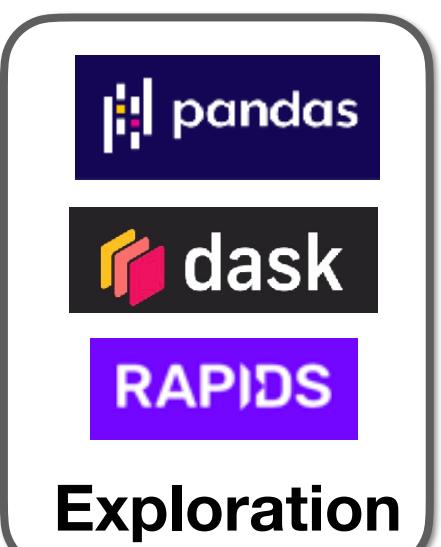
“All-in-one”



Amazon SageMaker

gradient<sup>o</sup>  
by Paperspace

DOMINO  
DATA LAB



Data



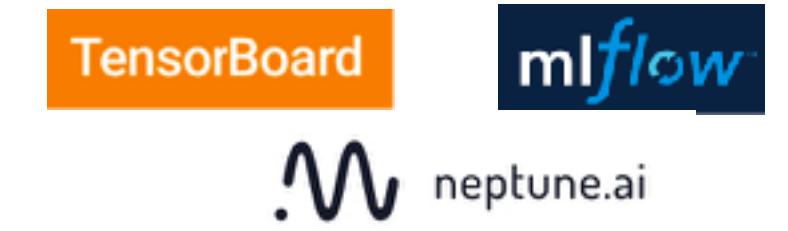
Frameworks &  
Distributed Training



Resource Management



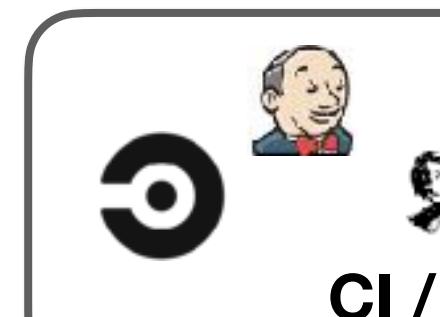
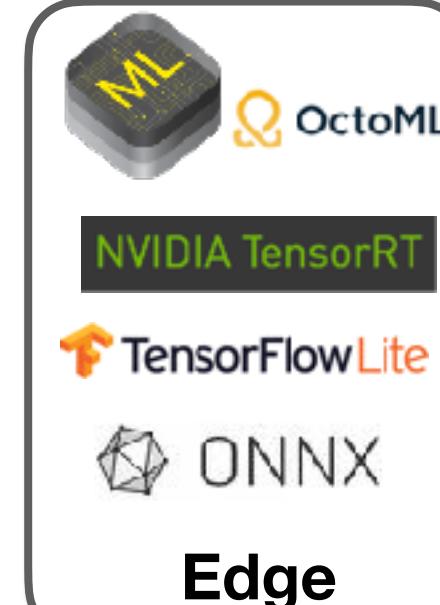
Development



Experiment and Model  
Management



Software Engineering





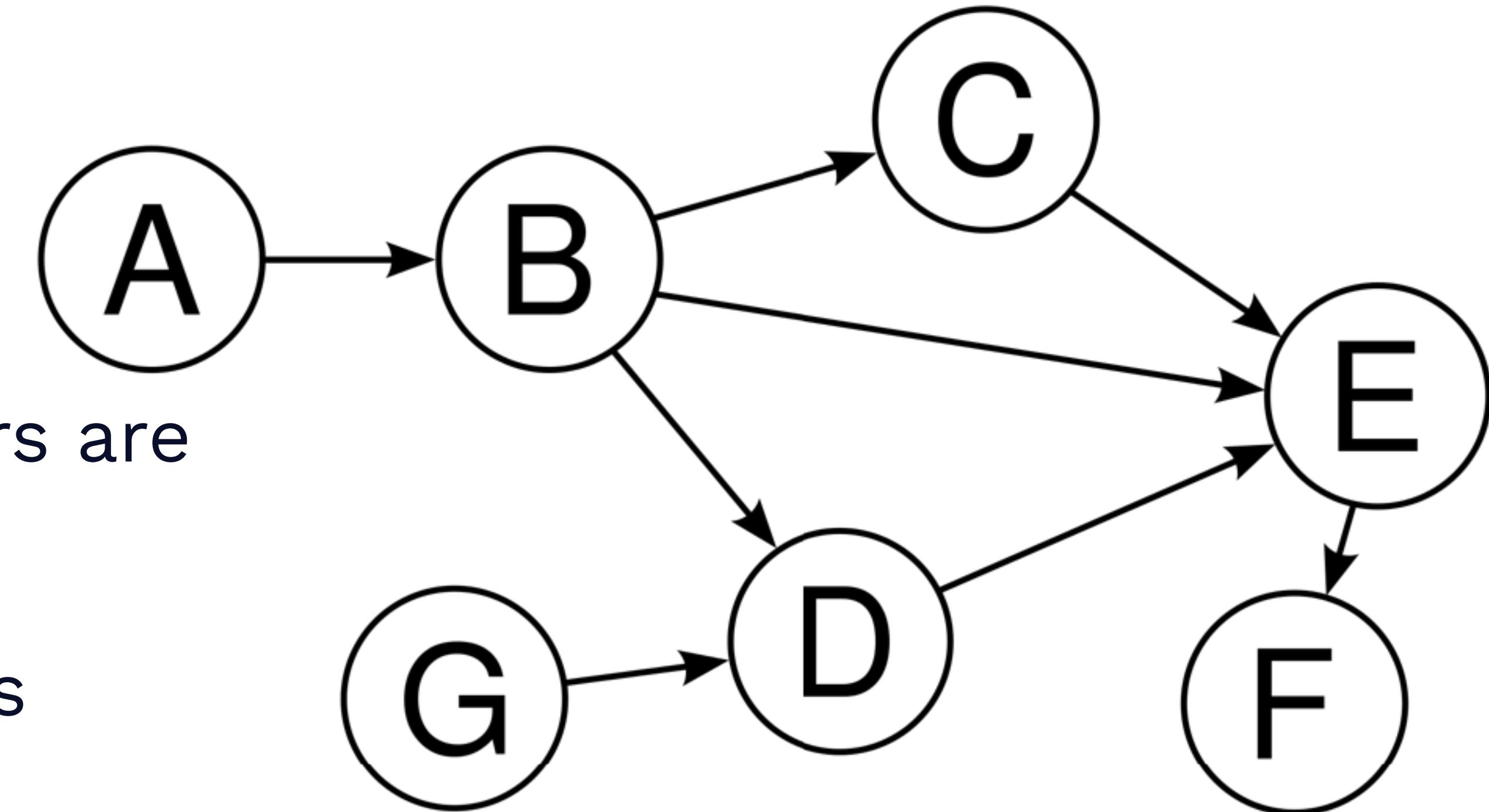
# Motivational Example

- We have to train a photo popularity predictor every night.
  - For each photo, training data must include:
    - Metadata such as posting time, title, location
    - Some features of the user, such as how many times they logged in today.
    - Outputs of photo classifiers (content, style)
- In database*
- Need to compute from logs*
- Need to run classifiers*



# Task Dependencies

- Some tasks can't start until others are finished.
- Finishing a task should kick off its dependencies.





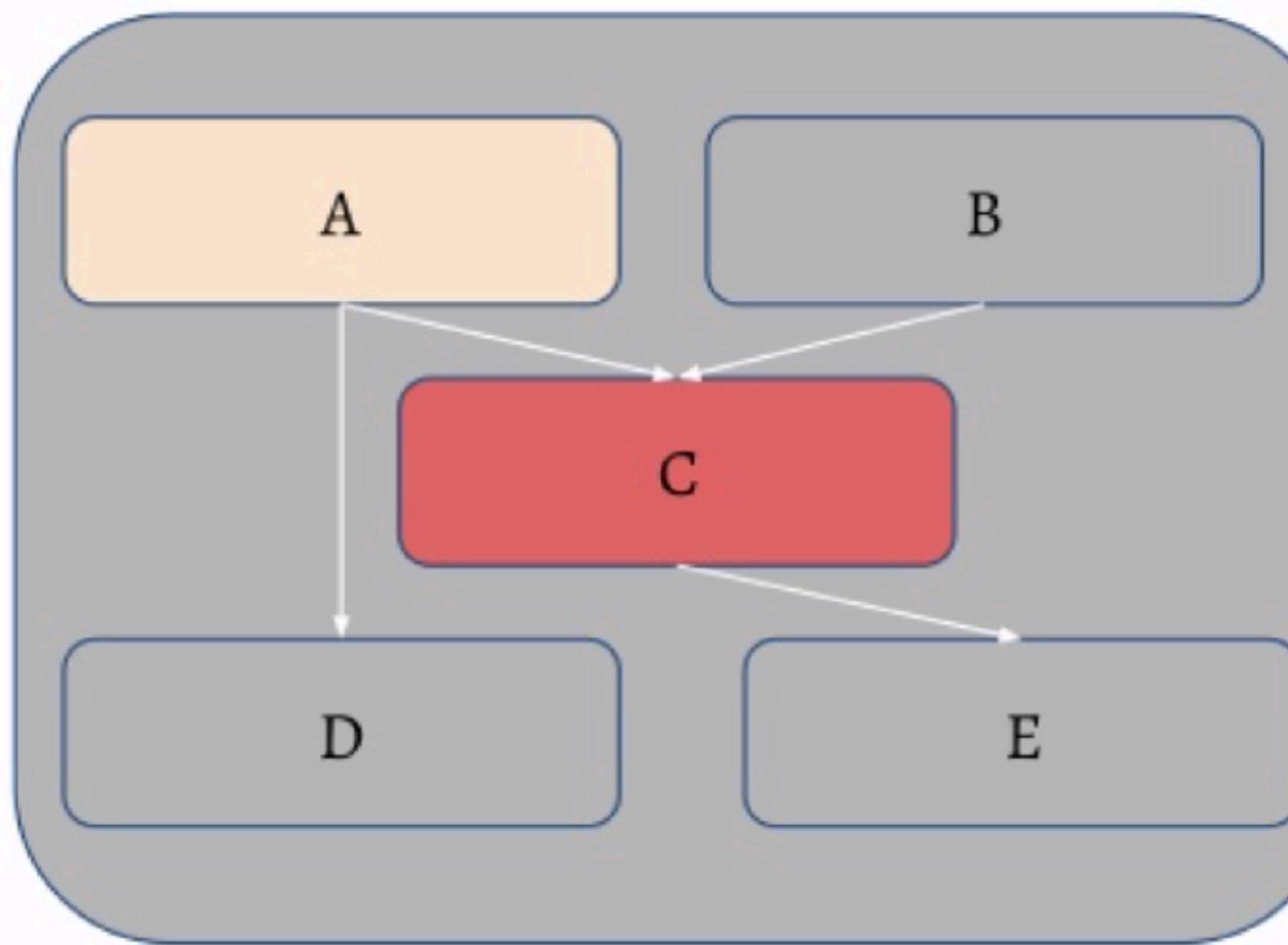
# Ideally

- Dependencies are not always files, but programs and databases
- Work needs to be spread over many machines
- Many dependency graphs are executing all at once



# Airflow

- Specify the DAG of tasks using Python



```
# Airflow
dag = DAG(schedule_interval=
            timedelta(days=1),
           start_date=
           datetime(2015,10,6))

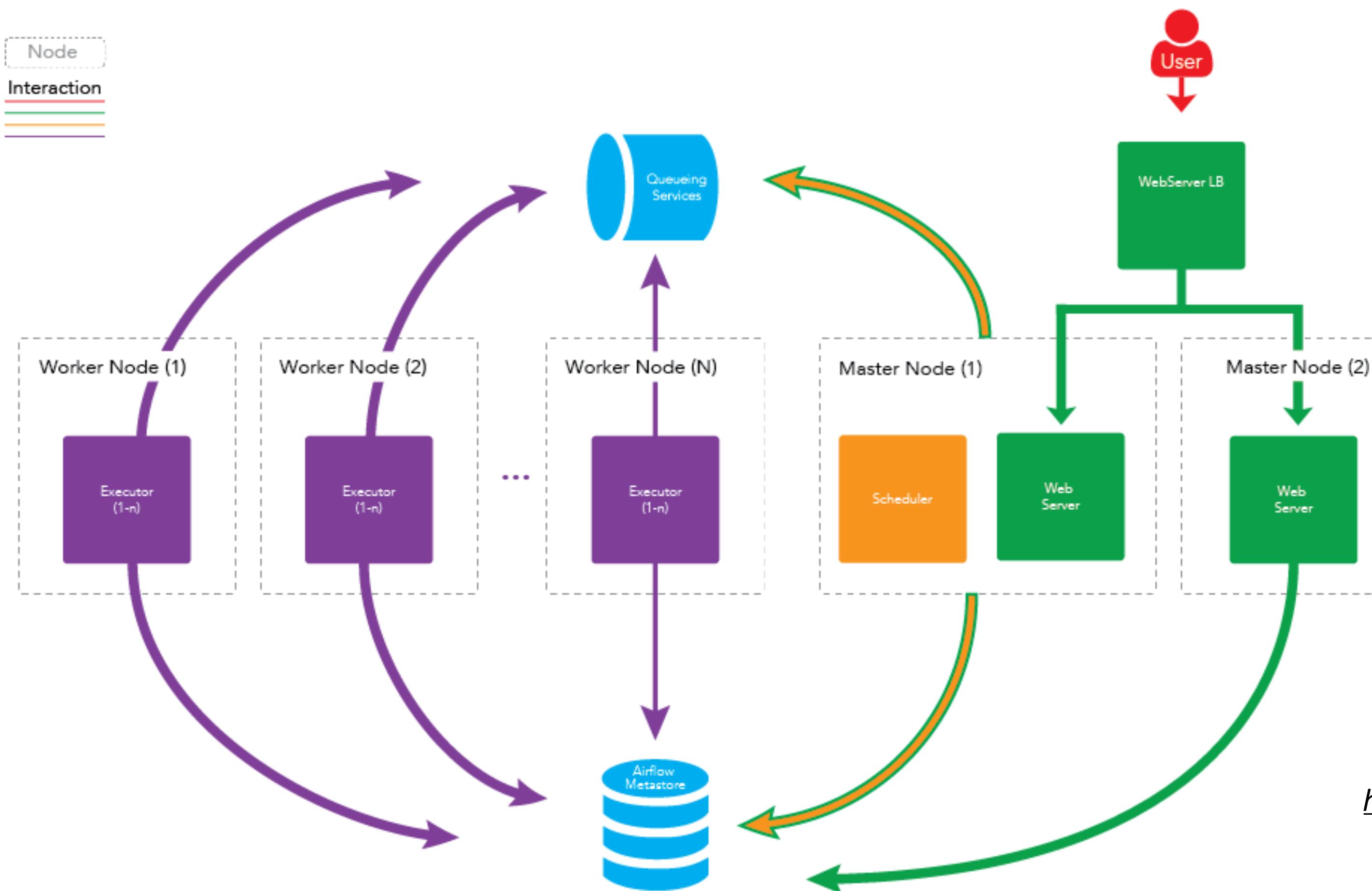
a = PythonOperator(
    task_id="A",
    python_callable=ClassA,
    dag=dag)

c = MySQLOperator(
    task_id="B",
    sql="DROP TABLE hello",
    dag=dag)

c.set_upstream(a)
```

# Distributing work

- The workflow manager has a queue for the tasks, and manages workers that pull from it, restarting jobs if they fail.





# Prefect

- Improvements over Airflow

```
1 from prefect import task, Flow
2
3 @task
4 def say_hello():
5     print("Hello, world!")
6
7
8 with Flow("My First Flow") as flow:
9     say_hello()
10
11 flow.run() # "Hello, world!"
```



## Focus on Your Code

Execution in Your Cloud;  
Orchestration in Ours



STEP 01 - CORE

### Build Your Flow

Design and test your workflow with our open-source Prefect Core framework.



STEP 02 - CLOUD

### Register Your Flow

Send metadata (but never code!) to Prefect Cloud in order to register your flow for scheduling and execution.



STEP 03 - CORE

### Run the Flow

Flow runs are executed on your private infrastructure, keeping data secure. State updates are sent to Cloud as metadata.



STEP 04 - CLOUD

### Monitor and Manage

Use the Cloud UI to monitor all of your flows, no matter where or how often they run.

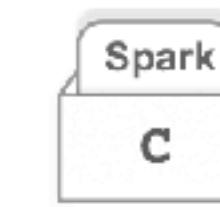
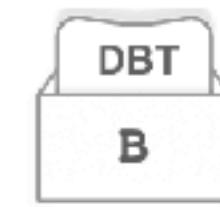


# Dagster

- Another contender



**Dagster is a data orchestrator for machine learning, analytics, and ETL**



Implement components in any tool, such as Pandas, Spark, SQL, or DBT.



Define your pipelines in terms of the data flow between reusable, logical components.



Test locally and run anywhere with a unified view of data pipelines and assets.



# Keep things simple whenever possible

- Don't overengineer
- We have many CPU cores and a lot of RAM nowadays
- For example, UNIX has powerful parallelism, streaming, highly optimized tools

```
find . -type f -name '*.pgn' -print0 |  
xargs -0 -n4 -P4 mawk '/Result/ { split($0, a, "-"); res = substr(a[1], length(a[1]), 1); if (res == 1) white++; if (res == 0) black++; if (res == 2) draw++ } END { print white+black+draw, white, black, draw }'  
mawk '{games += $1; white += $2; black += $3; draw += $4;} END { print games, white, black, draw }'
```

## Command-line Tools can be 235x Faster than your Hadoop Cluster

January 18, 2014



26 minutes

in parallel  
18 seconds

in parallel  
70 seconds

<https://adamdrake.com/command-line-tools-can-be-235x-faster-than-your-hadoop-cluster.html>

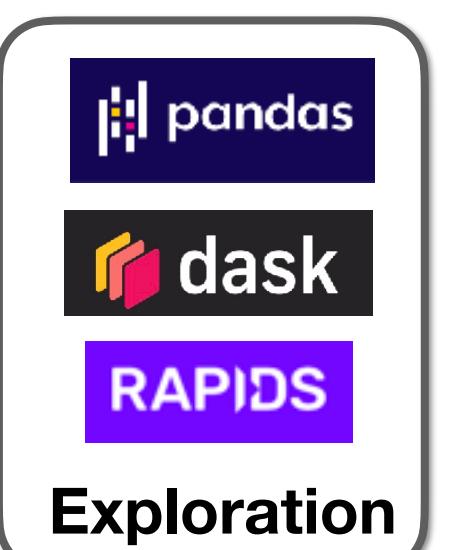
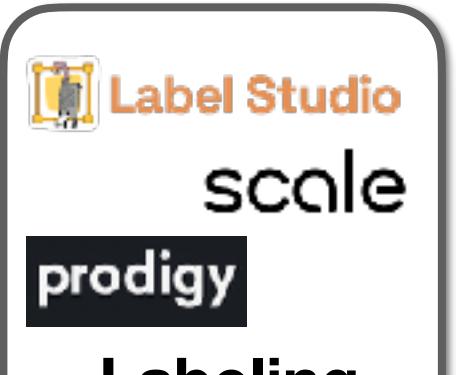
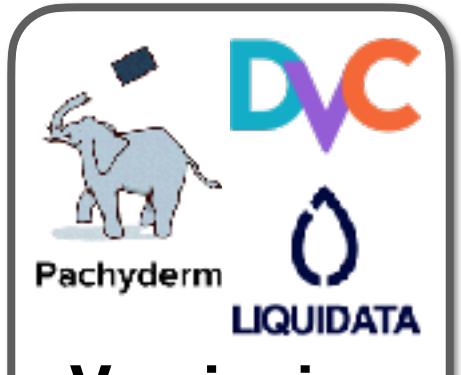
“All-in-one”



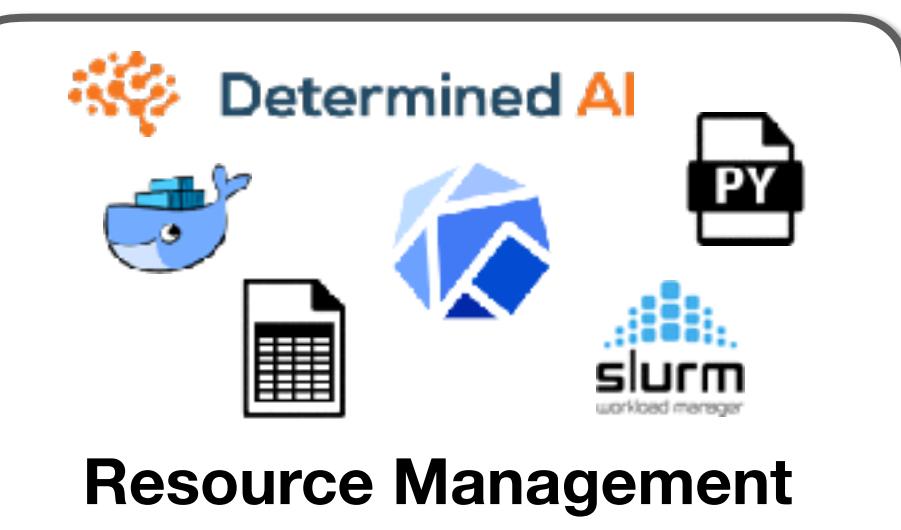
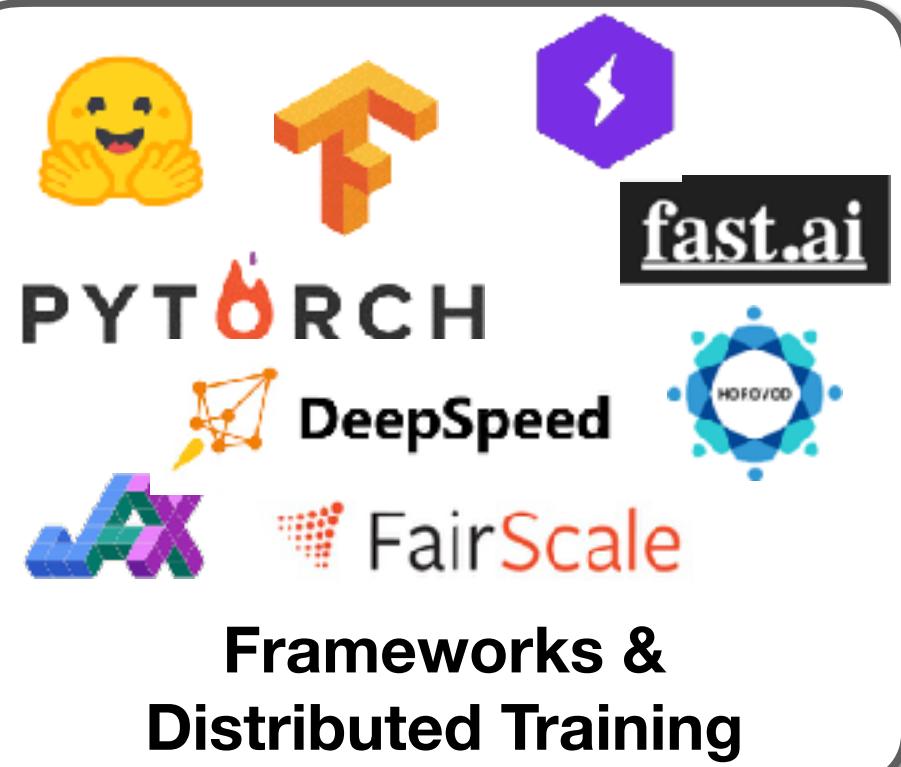
Amazon SageMaker

gradient<sup>o</sup>  
by Paperspace

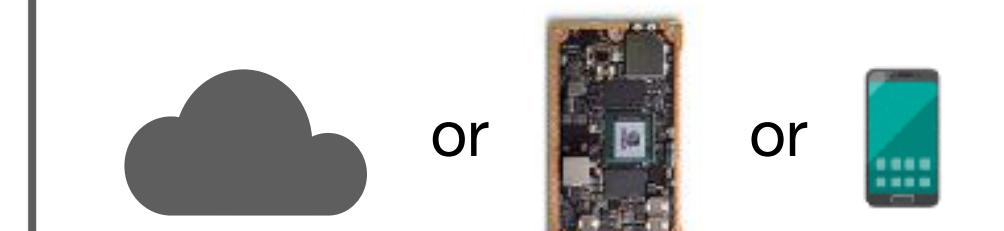
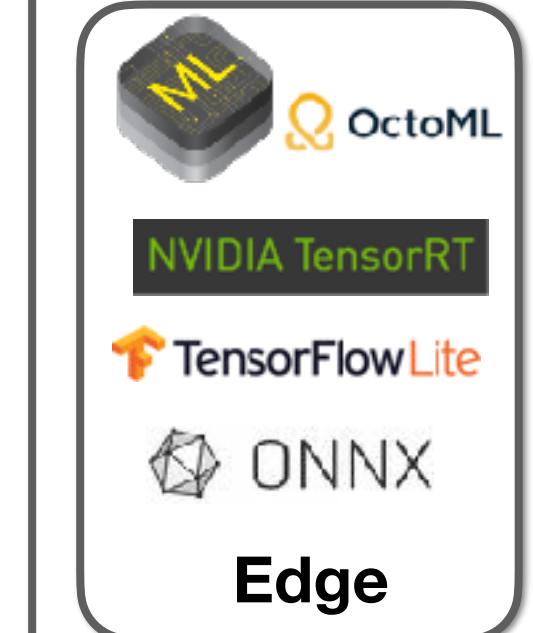
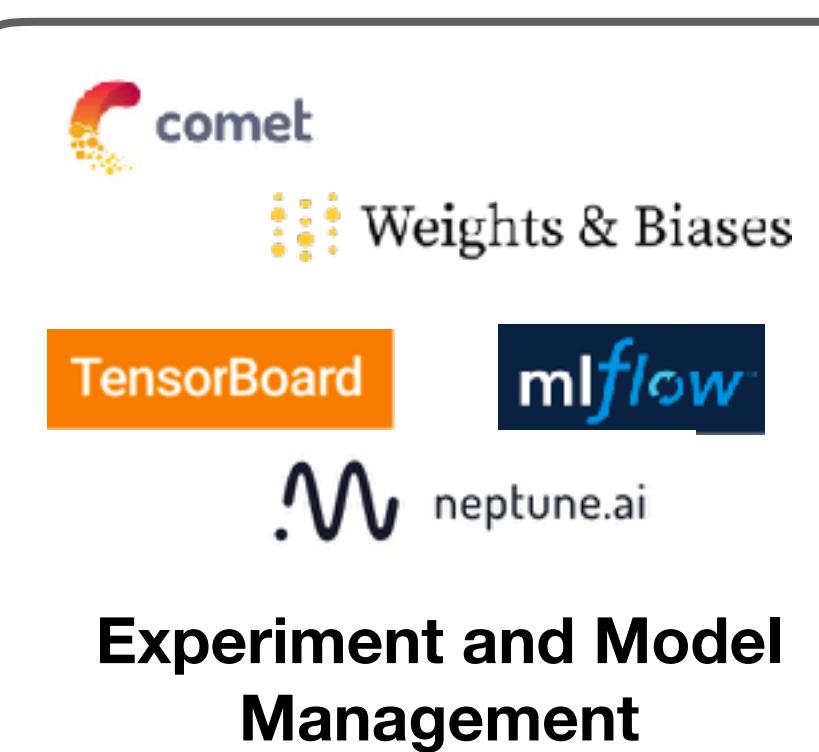
DOMINO  
DATA LAB



Data



Development





# Why feature stores?

- All the data processing generates artifacts for training
- How do we
  - Make sure that in production, the same processing takes place?
  - Avoid recomputation when we retrain?
- Feature stores are a solution (that you may not need!)

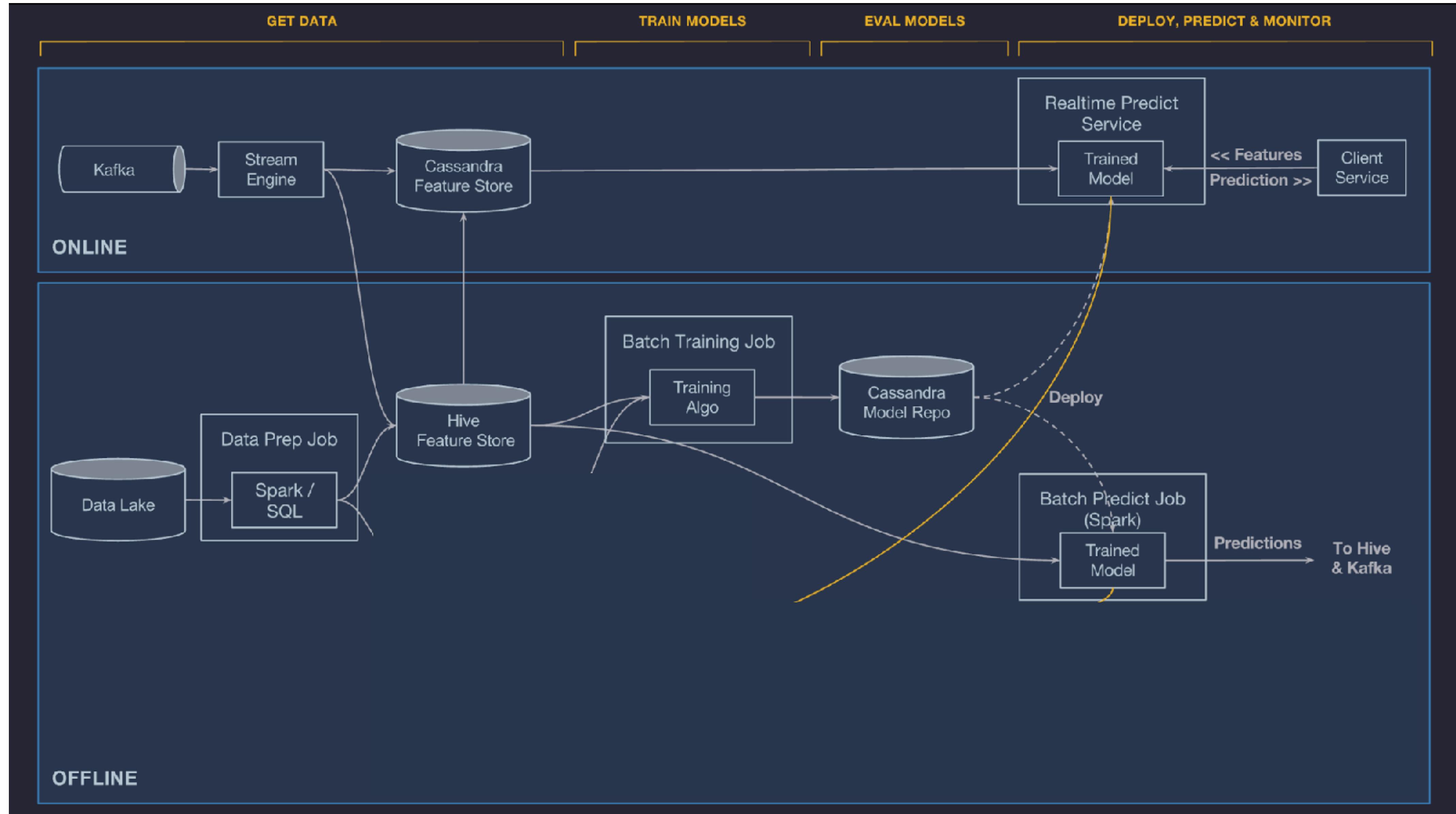
# Meet Michelangelo: Uber's Machine Learning Platform



FSDL 2022

Jeremy Hermann and Mike Del Balso

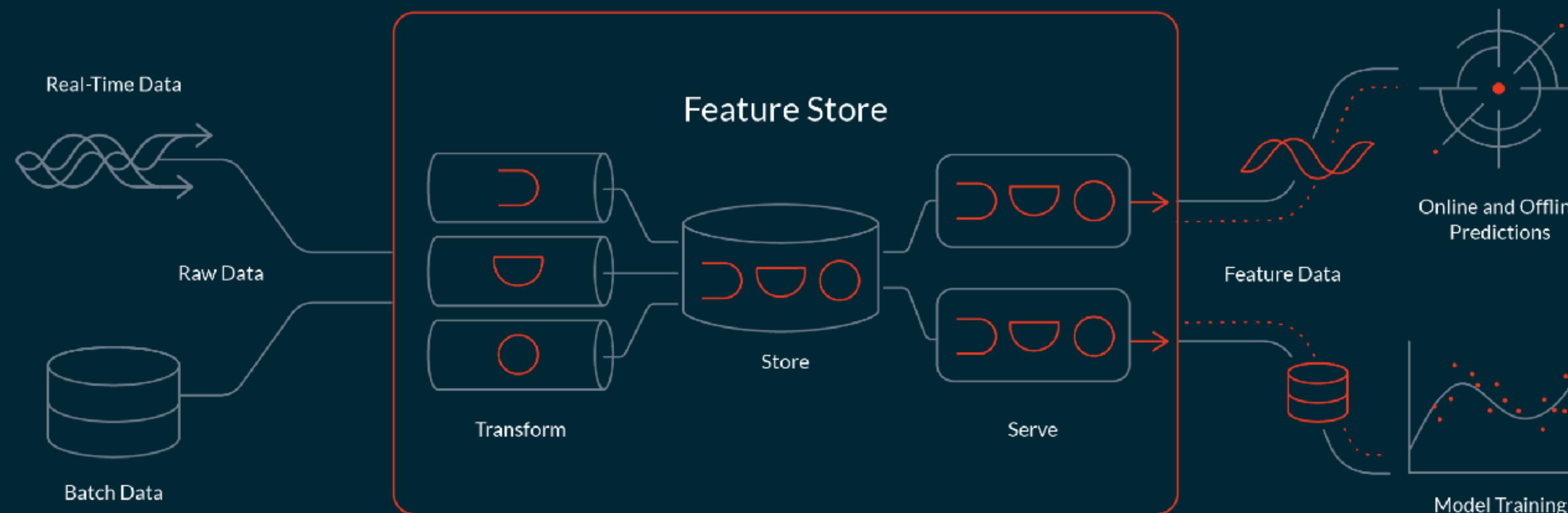
September 5, 2017





# The Enterprise Feature Store for Machine Learning

Build a library of great features. Serve them in production. Do it at scale.

[Request a free trial](#)

# An open source feature store for machine learning

Feast is the bridge between your data and your machine learning models. It allows teams to register, ingest, serve, and monitor features in production.

[Quickstart](#)[Learn More](#)

```
customer_features = [  
    'credit_score', 'balance', 'total_purchases', 'last_active'  
]  
  
# Fetch historical features  
historical_features_df = fs.get_historical_features(customer_ids, customer_features)  
  
# Train model  
my_model = ml.fit(historical_features_df)  
  
# Fetch online features  
online_features = fs.get_online_features(customer_ids, customer_features)  
  
# Predict using online features  
prediction = my_model.predict(online_features)
```

## Register

Feast provides a registry through which to explore, develop, collaborate on, and publish new feature definitions. The registry is the central interface for all interactions with the feature store.

## Ingest

Easily ingest data from both batch and streaming sources into both online and offline feature stores, automating data management and making features available for serving.

## Serve

A feature retrieval interface that provides a consistent view of features in stores. Feast provides a point-in-time correct interface for training data, and a low-latency API for online serving.

## Monitor

Feast allows teams to confidently operate machine learning systems by publishing operational metrics, statistics, and logs to their existing production monitoring infrastructure.

[What is Featureform](#)[Quickstart \(Local\)](#)[Quickstart \(Kubernetes\)](#)

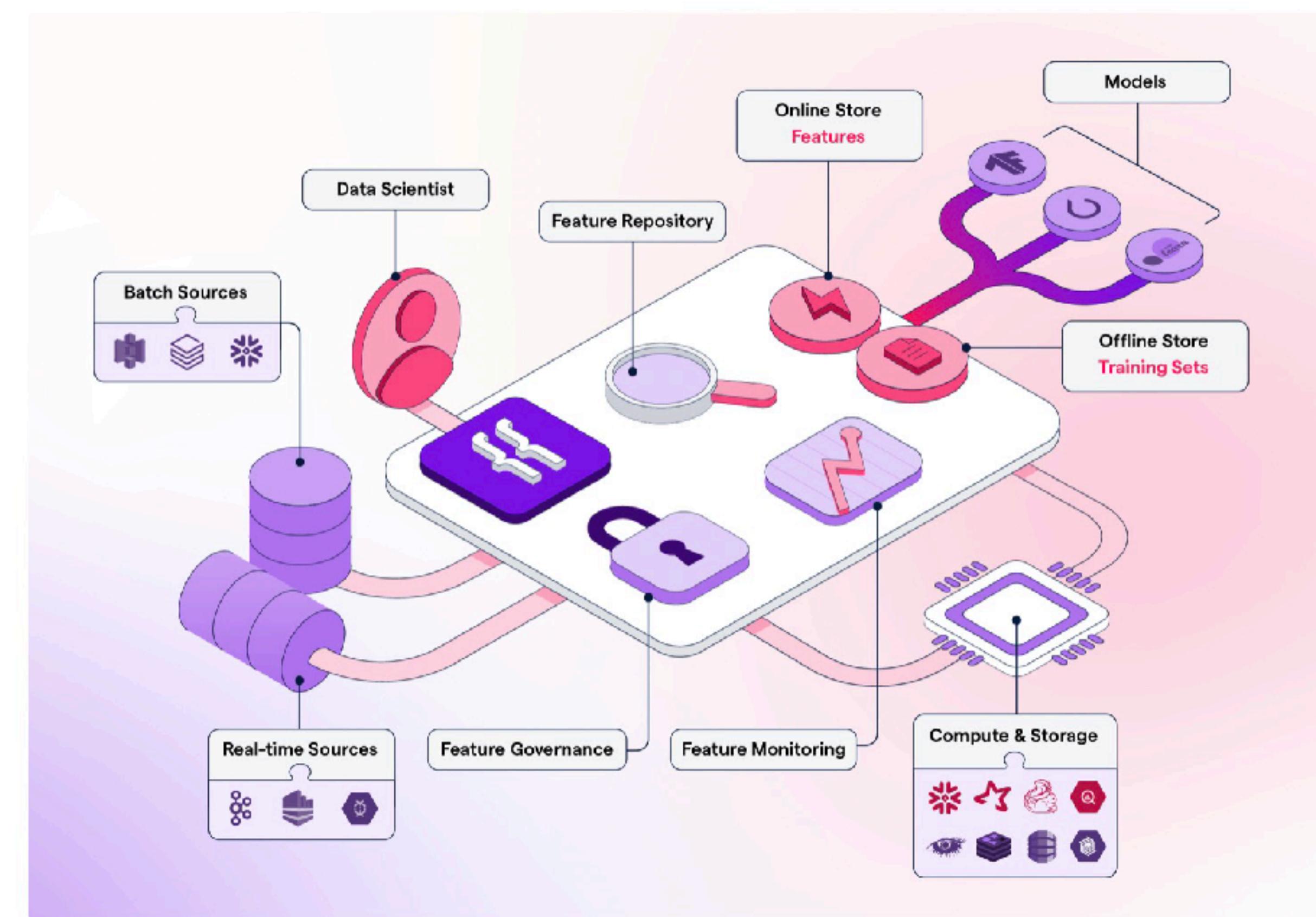
## GETTING STARTED

[Overview](#)[Registering Infrastructure Providers](#)[Transforming Data](#)[Defining Features, Labels, and Training Sets](#)[Serving for Inference and Training](#)[Interact with the CLI](#)[Exploring the Feature Registry](#)

## PROVIDERS

[Redis](#)[Cassandra](#)[DynamoDB](#)[Firestore](#)[Snowflake](#)[Postgres](#)

# What is Featureform



Featureform is a **virtual feature store**. It enables data scientists to **define, manage, and serve** their ML model's features. Featureform sits **atop your existing infrastructure** and orchestrates it to work like a **traditional feature store**.

By using Featureform, a data science team can solve the organizational problems:



# In summary

- Binary data (images, sound files, compressed texts) is stored as **objects**.
- Metadata (labels, user activity) is stored in **database**.
- Don't be afraid of **SQL**, and know there are accelerated **DataFrames**
- If dealing with logs and other sources of data, set up **data lake**
- Set up a repeatable **process** to aggregate data needed for training.
  - Depending on expense and complexity of processing, a **feature store** could be useful
- At training time, copy the data that is needed to a **filesystem** on a fast drive, and optimize **GPU transfer**.

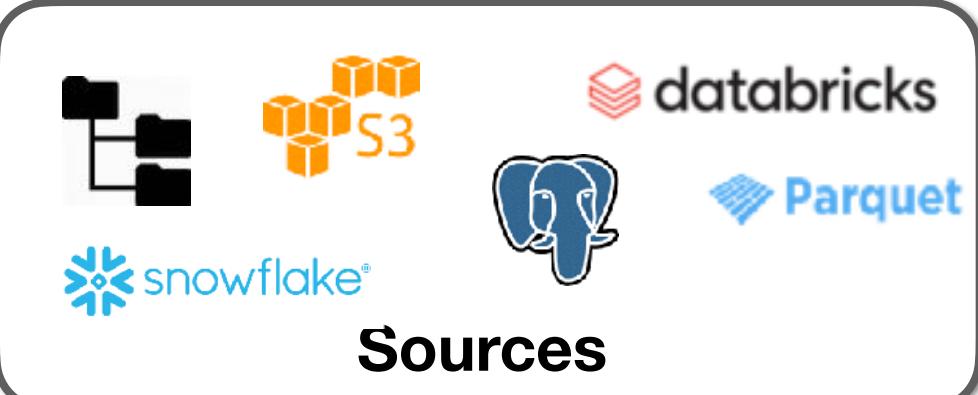
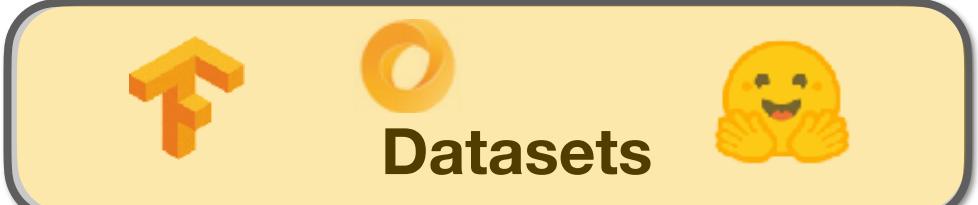
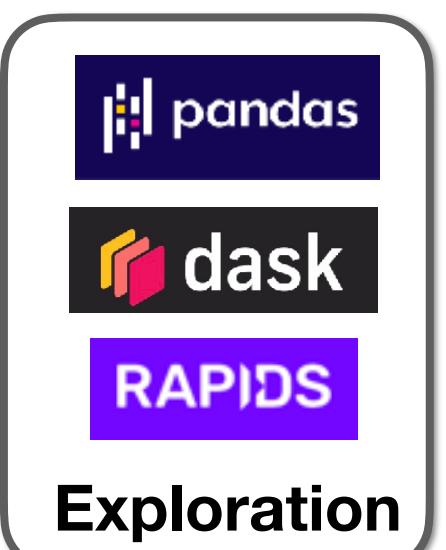
“All-in-one”



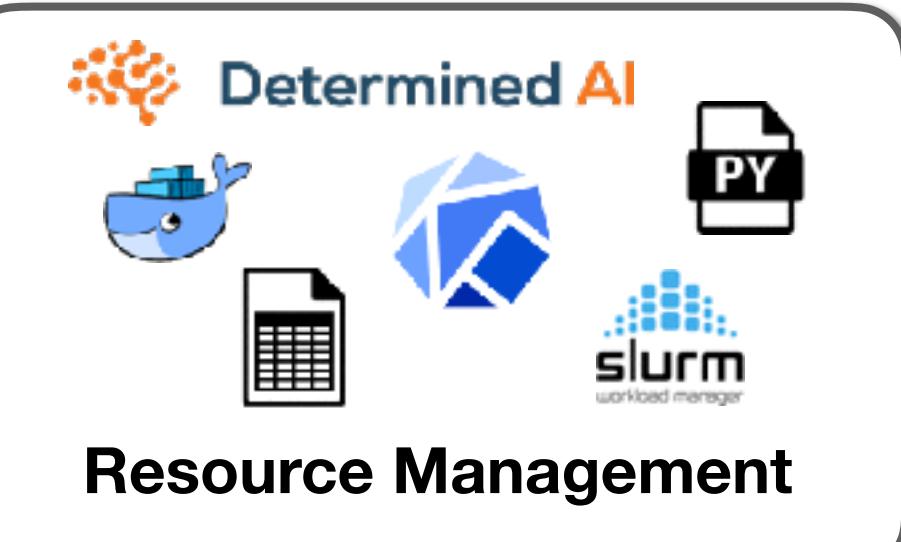
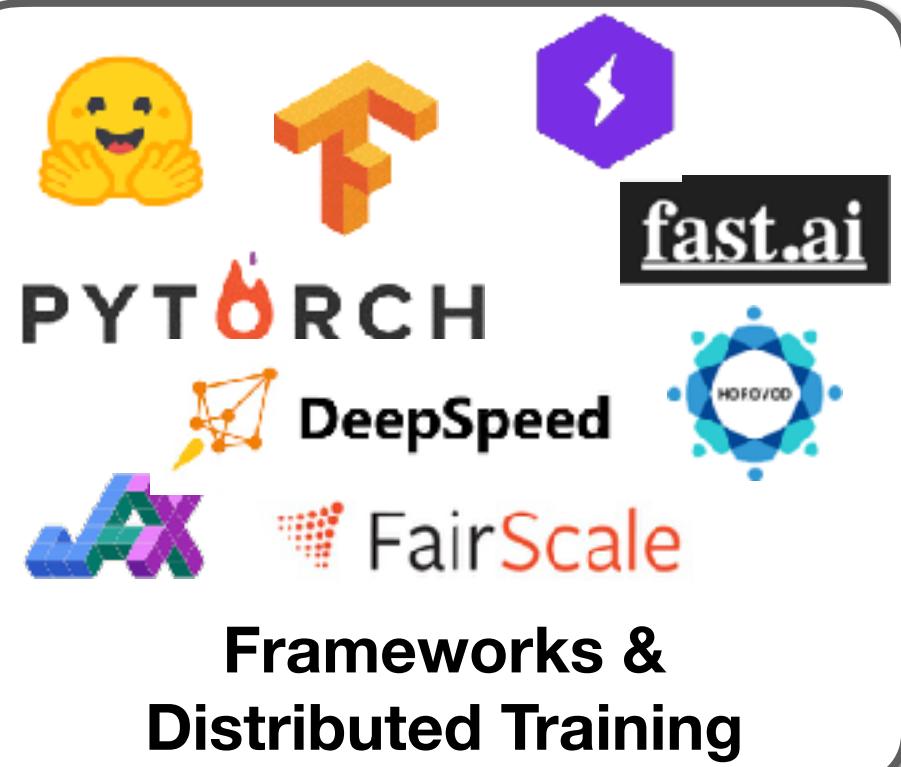
Amazon SageMaker

gradient<sup>o</sup>  
by Paperspace

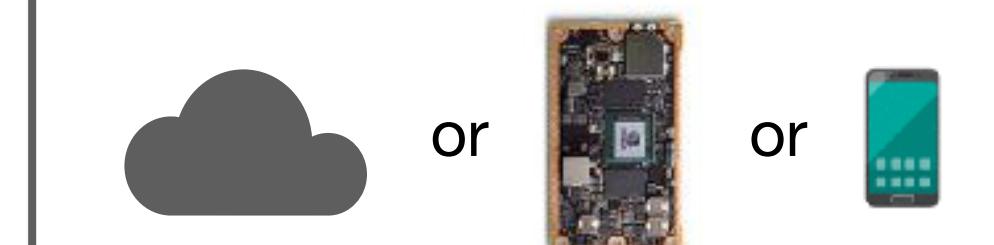
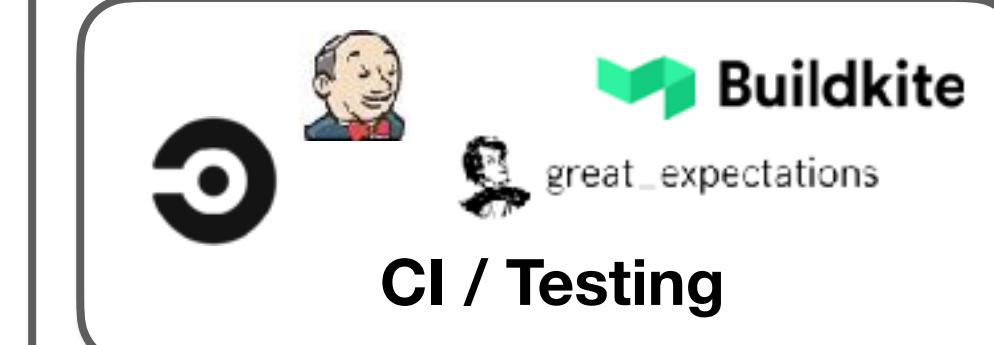
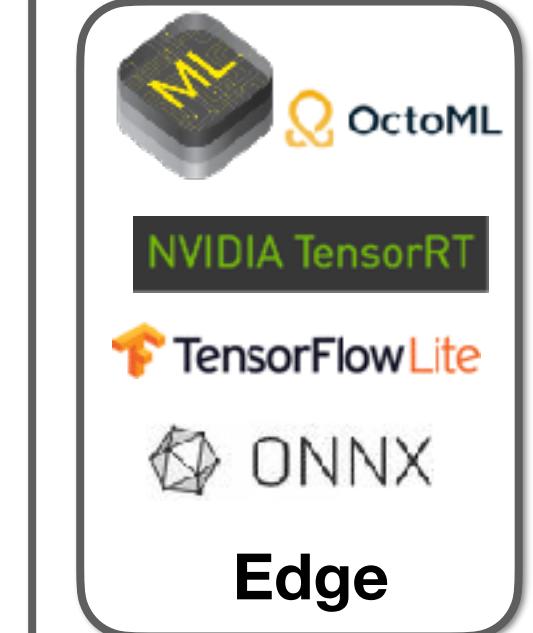
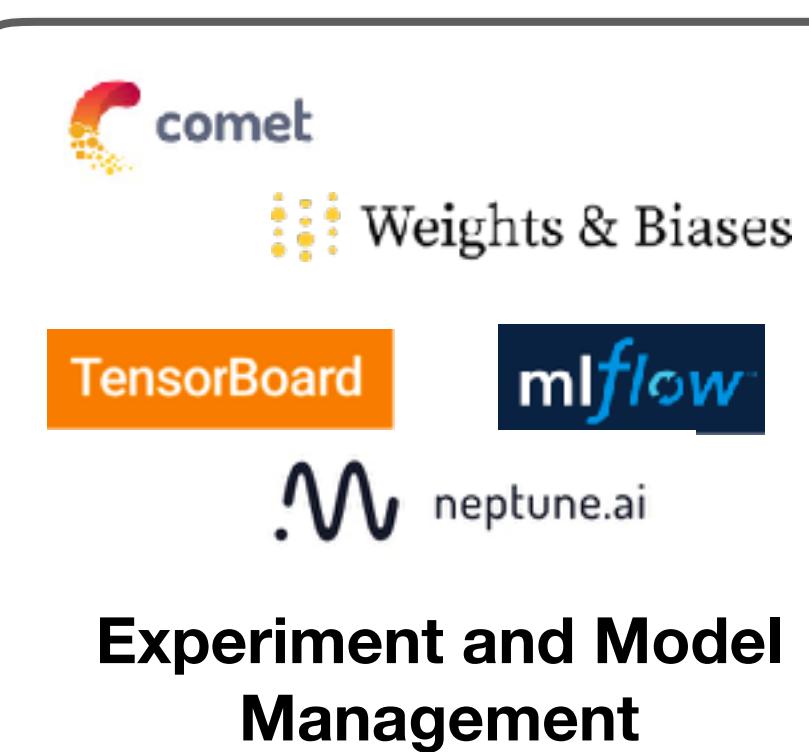
DOMINO  
DATA LAB



Data



Development



Deployment





# Huggingface Datasets

- Over 8K datasets for vision, NLP, etc

The screenshot shows the Huggingface Datasets homepage. At the top left is the Hugging Face logo with a smiling emoji. A search bar is at the top right. Below the header are sections for Task Categories, Tasks, Languages, and Multilinguality. The Task Categories section lists several categories like text-classification, question-answering, and text-generation. The Tasks section lists language-modeling, multi-class-classification, extractive-qa, named-entity-recognition, open-domain-qa, and natural-language-inference. The Languages section shows English, German, French, Spanish, Russian, and Arabic. The Multilinguality section shows monolingual, multilingual, translation, unknown, and other-programming-languages. To the right, a large list of datasets is shown, each with a thumbnail, name, preview link, update date, size, and a heart icon for likes. The datasets listed are red\_caps, super\_glue, GEM/wiki\_lingua, codeparrot/github-code-clean, CodedotAI/code\_clippy\_github, and sil-ai/bloom-lm.

Dataset	Preview	Last Updated	Size	Likes
red_caps	Preview	Jul 1	6.41M	12
super_glue	Preview	About 20 hours ago	1.48M	23
GEM/wiki_lingua	Updated	26 days ago	1.15M	3
codeparrot/github-code-clean	Preview	Jul 5	507k	1
CodedotAI/code_clippy_github	Preview	18 days ago	402k	5
sil-ai/bloom-lm	Preview	May 11	324k	3



# Example Dataset

- Github-Code: >1TB of text
- Library allows you to stream it
- Underlying format: Parquet

main ➔ github-code / data

lvwerra fix dedup and add scala/typescript d1387b6

This view is limited to 50 entries.

File	Size	Description
train-00000-of-01126.parquet	286 MB	fix dedup and add scala/typescript
train-00001-of-01126.parquet	289 MB	fix dedup and add scala/typescript
train-00002-of-01126.parquet	291 MB	fix dedup and add scala/typescript
train-00003-of-01126.parquet	287 MB	fix dedup and add scala/typescript
train-00004-of-01126.parquet	291 MB	fix dedup and add scala/typescript
train-00005-of-01126.parquet	288 MB	fix dedup and add scala/typescript
train-00006-of-01126.parquet	289 MB	fix dedup and add scala/typescript
train-00007-of-01126.parquet	288 MB	fix dedup and add scala/typescript

Datasets: codeparrot/github-code like 31

Tasks: language-modeling Task Categories: sequence-modeling Languages: code Multilinguality: multilingual Size Categories:

Language Creators: crowdsourced expert-generated Licenses: other

Dataset card Files and versions Community 3

**Dataset Description**

How to use it

**Data Structure**

Data Instances Data Fields Data Splits

**GitHub Code Dataset**

**Dataset Description**

The GitHub Code dataset consists of 115M code files from GitHub in 32 programming languages with 60 extensions totaling in 1TB of data. The dataset was created from the public GitHub dataset on Google BigQuery.

**Using the Data**

**How to use it**

The GitHub Code dataset is a very large dataset so for most use cases it is recommended to make use of the streaming API of datasets. You can load and iterate through the dataset with the following two lines of code:

```
from datasets import load_dataset
ds = load_dataset("codeparrot/github-code", streaming=True, s
print(next(iter(ds)))
```

#OUTPUT:

```
{
  'code': "import mod189 from './mod189';\nvar value=mod189+1;
  'repo_name': 'MirekSz/webpack-es6-ts',
  'path': 'app/mods/mod190.js',
  'language': 'JavaScript',
```



# Example Dataset

- RedCaps: 12M image-text pairs
- Need to download images yourself (multi-threaded!)
- Underlying format: images + JSON files

The screenshot shows a file explorer interface with a dark theme. On the left, there is a list of JSON files: abandoned\_2017.json, abandoned\_2018.json, abandoned\_2019.json, abandoned\_2020.json, abandonedporn\_2017.json, abandonedporn\_2018.json, abandonedporn\_2019.json, abandonedporn\_2020.json, absoluteunits\_2018.json, absoluteunits\_2019.json, absoluteunits\_2020.json, airplants\_2017.json, airplants\_2018.json, airplants\_2019.json, and airplants\_2020.json. The file 'abandoned\_2017.json' is selected and its content is displayed in the main pane. The content is a JSON object with an 'annotations' key containing an array of objects. One object in the array has a 'url' key pointing to a file named 'DX0dR.jpg'.

```
{"annotations": [{"image_id": "grraj", " subreddit": "abandoned", "url": "https://i.imgur.com/DX0dR.jpg", "caption": "abandoned school urban exploration. abandoned due to asbestos and it is pretty much untouched", "raw_caption": "abandoned school urban exploration. Abandoned due to asbestos and it is pretty much untouched (x-posted)", "score": "EntenEller", "created_utc": 1302998814, "permalink": "/r/abandoned/comments/grraj/my_personal_abandoned_school_urban_exploration/"}]}
```

The screenshot shows a dataset card for 'red\_caps'. The top navigation bar includes 'Datasets: red\_caps', 'Tasks: image-captioning', 'Task Categories: image-to-text', 'Languages: English', 'Multilinguality: monolingual', 'Size Categories:', 'Annotations Creators: found', 'Source Datasets: original', 'ArXiv: 2111.11431', and 'Licenses: cc-by-4.0'. Below the navigation, there are three tabs: 'Dataset card' (selected), 'Files and versions', and 'Community'. The 'Dataset Structure' section contains links to 'Data Instances', 'Data Fields', and 'Data Splits'. The 'Dataset Creation' section contains links to 'Curation Rationale', 'Source Data', 'Annotations', and 'Personal and Sensitive Informa...'. The main content area is titled 'Dataset Card for RedCaps' and describes the dataset as a large-scale collection of 12M image-text pairs from Reddit. It mentions the variety of objects and scenes, the manual curation of subreddits, and the coarse image labels used for steering. The text continues to describe the dataset's purpose as sharing everyday things on social media and its use in hobbies like crafts and pets. The 'Dataset Preprocessing' section at the bottom notes that the dataset does not download images locally by default but instead exposes URLs to them, with a call to action to fetch images using provided code.

Datasets: **red\_caps** like 12

Tasks: **image-captioning** Task Categories: **image-to-text** Languages: **English** Multilinguality: **monolingual** Size Categories:

Annotations Creators: **found** Source Datasets: **original** ArXiv: **2111.11431** Licenses: **cc-by-4.0**

**Dataset card** Files and versions Community

**Dataset Structure**

Data Instances  
Data Fields  
Data Splits

**Dataset Creation**

Curation Rationale  
Source Data  
Annotations  
Personal and Sensitive Informa...

## Dataset Card for RedCaps

### Dataset Summary

RedCaps is a large-scale dataset of 12M image-text pairs collected from Reddit. Images and captions from Reddit depict and describe a wide variety of objects and scenes. The data is collected from a manually curated set of subreddits (350 total), which give coarse image labels and allow steering of the dataset composition without labeling individual instances. RedCaps data is created *by the people, for the people* – it contains everyday things that users like to share on social media, for example hobbies (r/crafts) and pets (r/shiba). Captions often contain specific and fine-grained descriptions (northern cardinal, taj mahal). Subreddit names provide relevant image labels (r/shiba) even when captions may not (mlem!), and sometimes may group many visually unrelated images through a common semantic meaning (r/perfectfit).

### Dataset Preprocessing

This dataset doesn't download the images locally by default. Instead, it exposes URLs to the images. To fetch the images, use the following code:



# Example Dataset

- CommonVoice: 14K hours of speech
- Underlying format: mp3 + text file

Datasets: mozilla-foundation/common\_voice\_8\_0 like 22

Tasks: automatic-speech-recognition Task Categories: speech-processing Multilinguality: multilingual Size Categories: 10K<n<100K

Language Creators: crowdsourced Annotations Creators: crowdsourced Source Datasets: extended|common\_voice ArXiv: 1912.0

Dataset card Files and versions Community 2

**Dataset Structure**

- Data Instances
- Data Fields
- Data Splits

**Data Preprocessing Recommen...**

**Dataset Creation**

- Curation Rationale
- Source Data
- Annotations
- Personal and Sensitive Informa...

**Considerations for Using the Data**

- Social Impact of Dataset
- Discussion of Biases
- Other Known Limitations

**Additional Information**

- Dataset Curators
- Licensing Information
- Citation Information

**Dataset Card for Common Voice Corpus 8.0**

**Dataset Summary**

The Common Voice dataset consists of a unique MP3 and corresponding text file. Many of the 18243 recorded hours in the dataset also include demographic metadata like age, sex, and accent that can help improve the accuracy of speech recognition engines.

The dataset currently consists of 14122 validated hours in 87 languages, but more voices and languages are always added. Take a look at the [Languages](#) page to request a language or start contributing.

**Supported Tasks and Leaderboards**

The results for models trained on the Common Voice datasets are available via the [Speech Bench](#)

**Languages**

Abkhaz, Arabic, Armenian, Assamese, Azerbaijani, Basaa, Bashk

# Activeloop

- Another interesting dataset-focused solution
- Explore, stream, and transform data without saving it all locally

The screenshot shows the homepage of the Activeloop website. At the top, there is a navigation bar with the Activeloop logo, a search icon, and links for Solutions, Company, Docs, Resources, Pricing, and Log in. Below the navigation bar is a large yellow banner with the text "Database for AI" and "AI-native way of working with data. No boilerplate code." It also features a call-to-action button "Get started" and a link "Read the docs". To the right of the banner is a 3D illustration of a stack of white cubes and a satellite in space. At the bottom of the page, there is a section titled "The open-source community enabling data-centric AI" with icons for GitHub stars, contributors, and community members.

Interested in the managed version for your company? Let's chat! →

## Database for AI

AI-native way of working with data.  
No boilerplate code.

The fastest open-source dataset format for data-centric computer vision workflows.

Get started    Read the docs

### The open-source community enabling data-centric AI

TRENDED #1 IN PYTHON  
**4.8k**  
GITHUB STARS

+10%  
**75+**  
CONTRIBUTORS

+31%  
**1K+**  
COMMUNITY MEMBERS



Search datasets

Log in



Datasets



Docs

Run query

Query history

Version: main

DETAILS ANALYTICS

## COCO Training

```
ds = hub.load('hub://activeloop/coco-train')
```

object detection image pose estimation segmentation

A Created by activeloop

Last updated: 11/02/2021

Size of dataset: Not Available

Number of tensors: Not Available

### README

#### Description

COCO is large-scale object detection, segmentation, and captioning dataset.

This dataset is based on COCO 2017.

Read more in our [COCO dataset](#) docs.

#### License

The annotations in this dataset belong to the COCO Consortium and are licensed under a Creative Commons Attribution 4.0 License.

The COCO Consortium does not own the copyright of the images. Use of the images must abide by the Flickr Terms of Use. The users of the images accept full responsibility for the use of the dataset, including but not limited to the use of any copies of copyrighted images that they may create from the dataset.



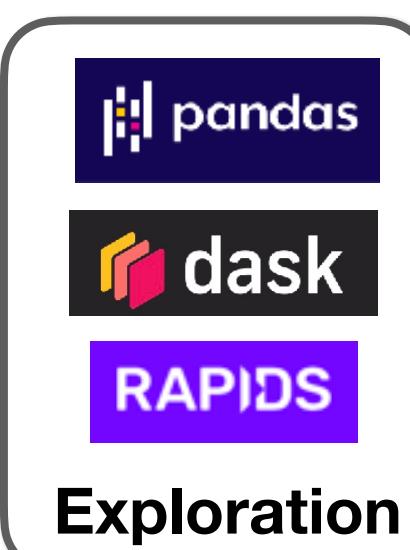
“All-in-one”



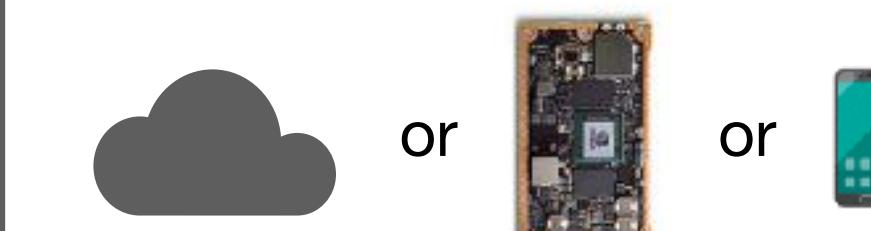
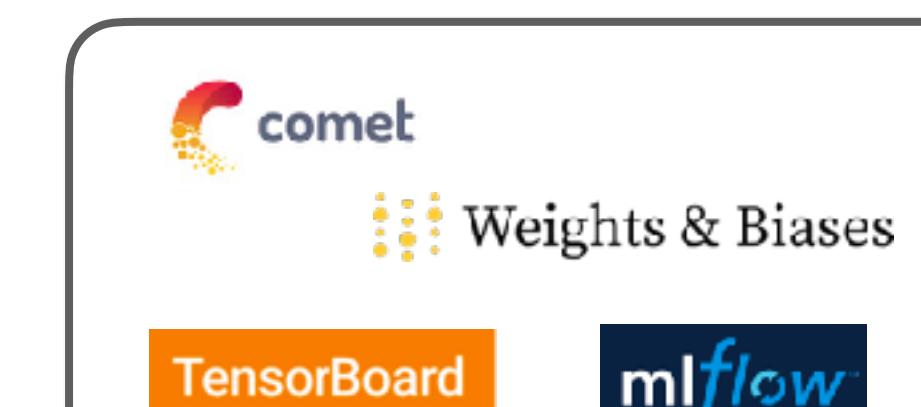
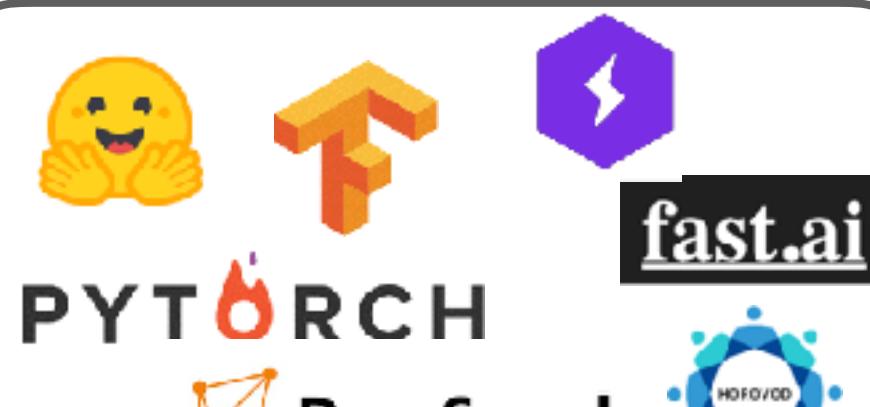
Amazon SageMaker

gradient<sup>o</sup>  
by Paperspace

DOMINO  
DATA LAB



Data



Deployment





# May not have to label data!

# Self-supervised learning

- ▶ Predict any part of the input from any other part.
- ▶ Predict the **future** from the **past**.
- ▶ Predict the **future** from the **recent past**.
- ▶ Predict the **past** from the **present**.
- ▶ Predict the **top** from the **bottom**.
- ▶ Predict the **occluded** from the **visible**
- ▶ Pretend there is a part of the input you don't know and predict that.

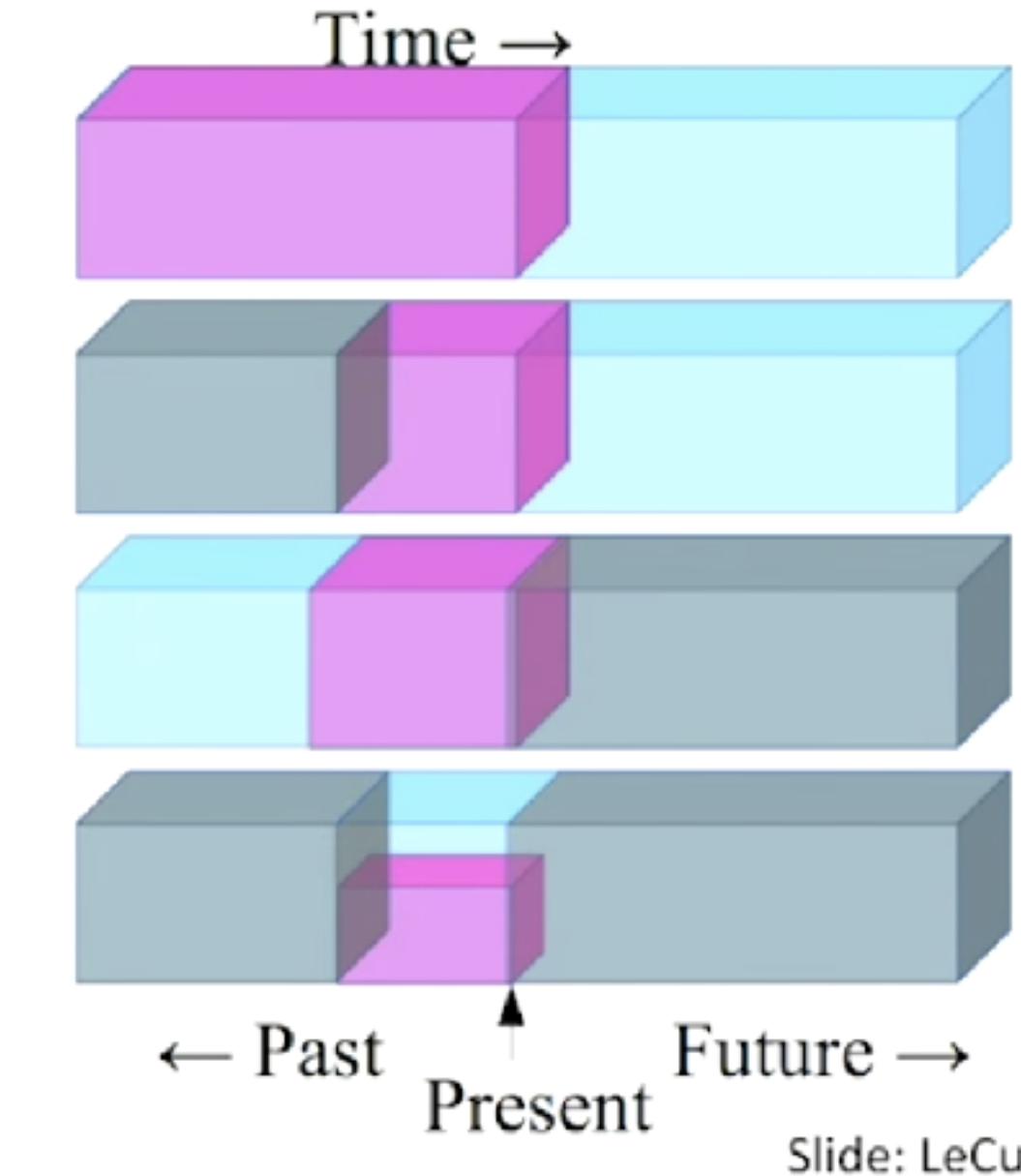
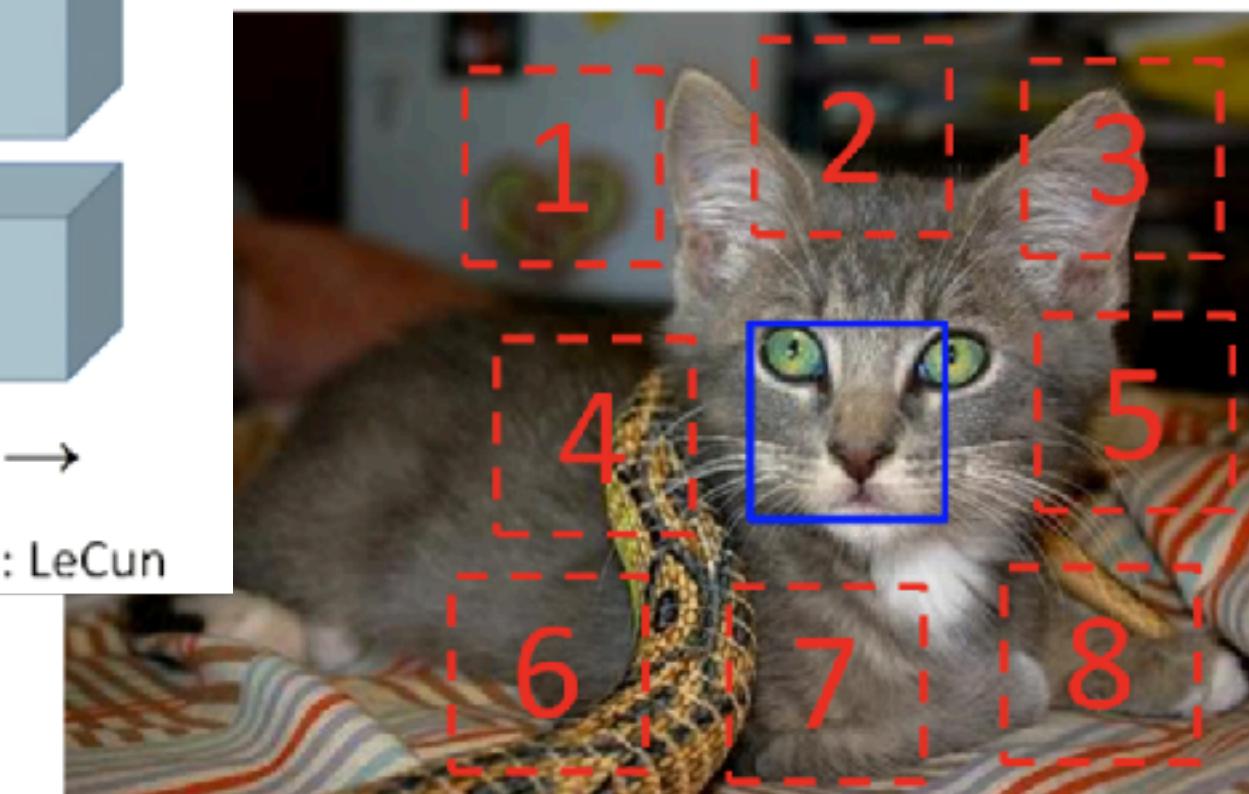


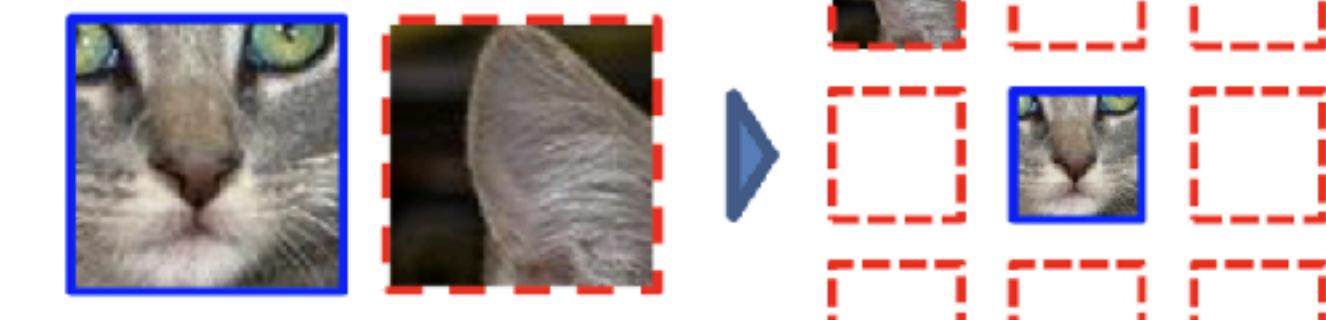
Fig. 1. A great summary of how self-supervised learning tasks can be constructed (Image source: LeCun's talk)

**Very important idea:** Use parts of data to label other parts



$$X = (\text{cat eye}, \text{ear}); Y = 3$$

Example:



Question 1:



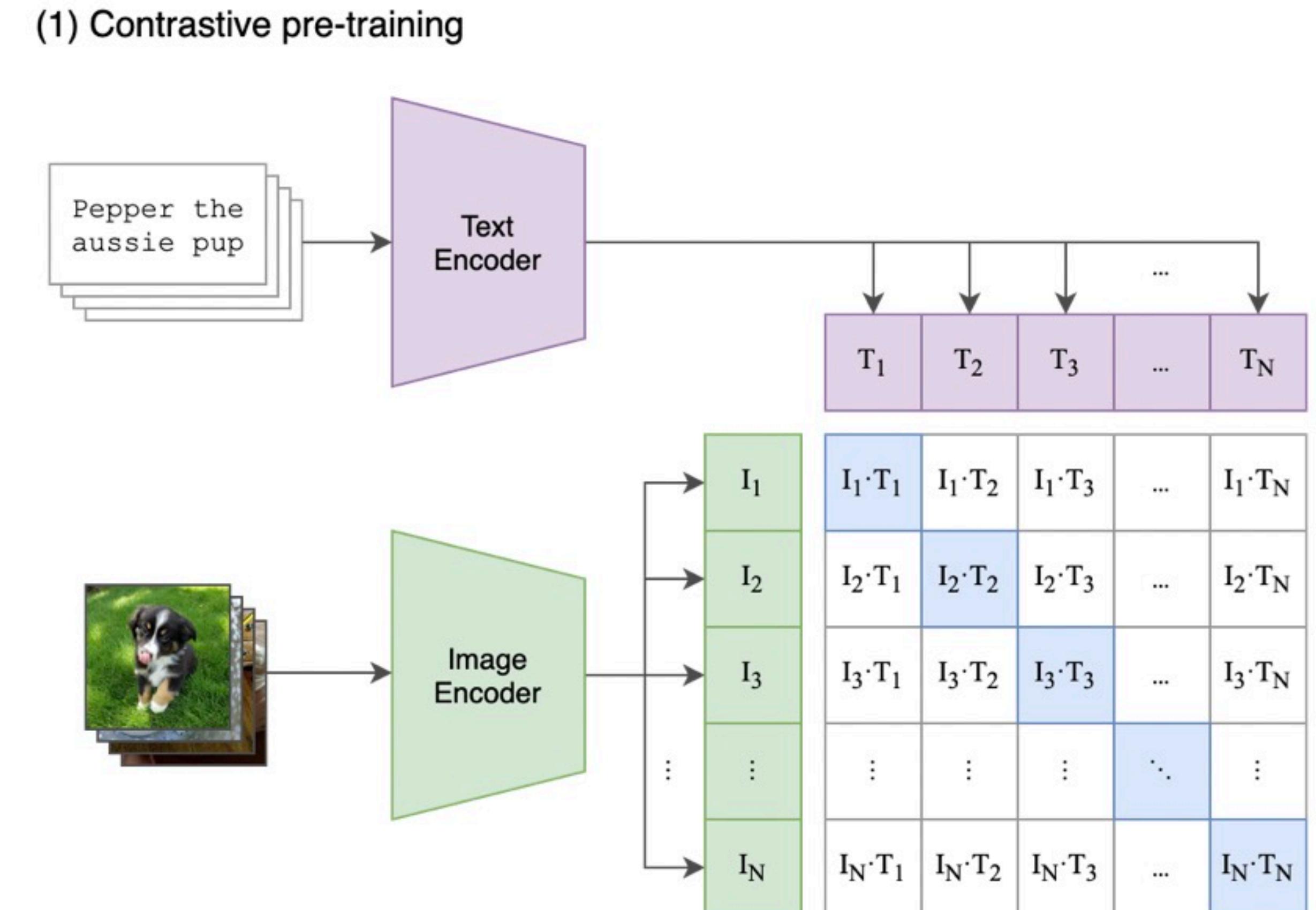
Fig. 4. Illustration of self-supervised learning by predicting the relative position of two random patches. (Image source: Doersch et al., 2015)

<https://ai.facebook.com/blog/self-supervised-learning-the-dark-matter-of-intelligence>

<https://lilianweng.github.io/lil-log/2019/11/10/self-supervised-learning.html>

# Self-supervised learning

- Works across modalities, too
- Note the "contrastive" training:
  - Minimize distance between image and its text
  - Maximize distance between image and other texts

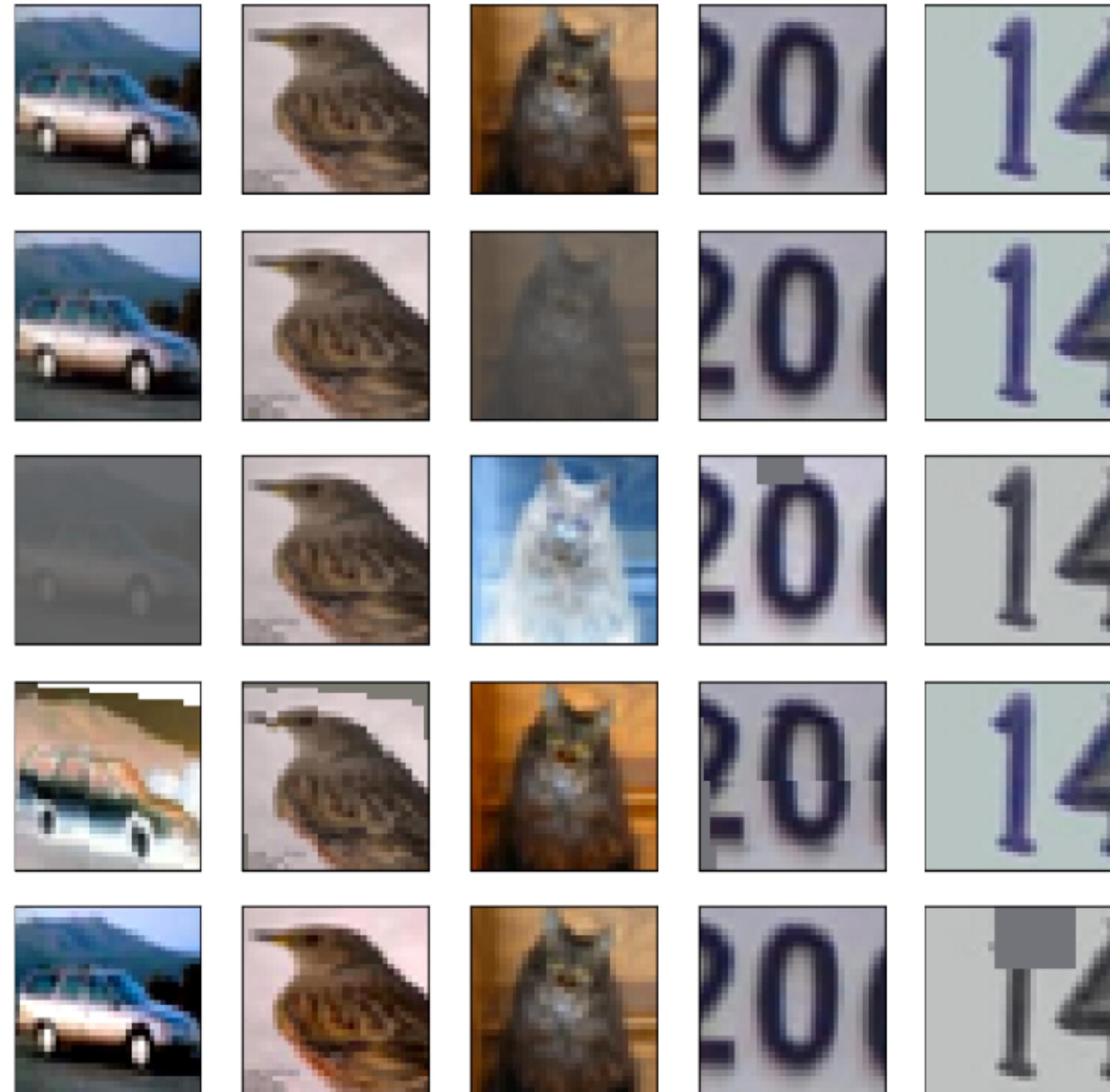


<https://github.com/openai/CLIP>



# Image data augmentation

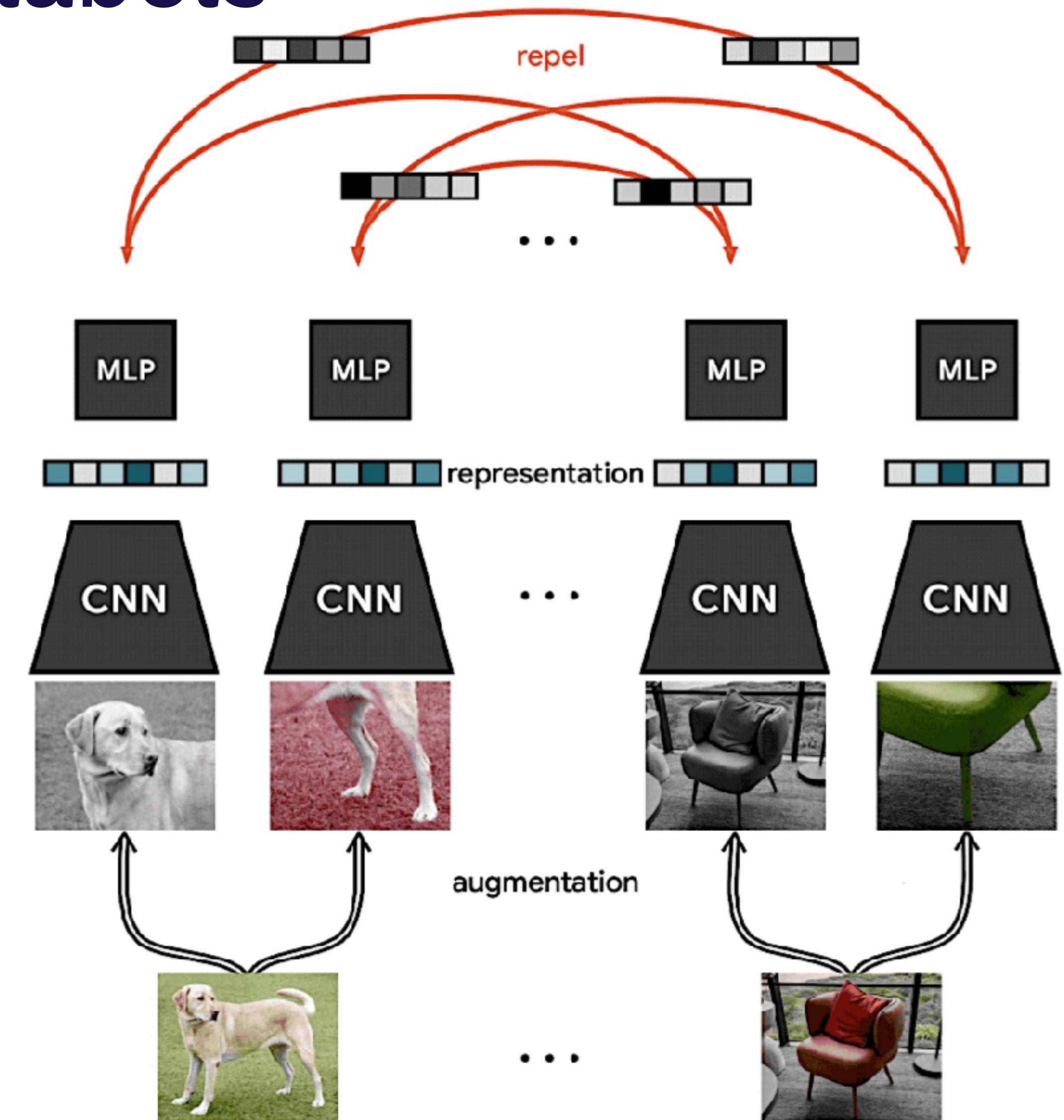
- **Must do** for training vision models
- Frameworks (e.g. torchvision) provide functions that do this
- Done in parallel to GPU training on the CPU



<https://towardsdatascience.com/1000x-faster-data-augmentation-b91baf8ee896c>

# Augmentation can replace labels

- SimCLR: learning objective is to
  - a) maximize agreement between augmented views of the same image
  - b) minimize agreement between different images

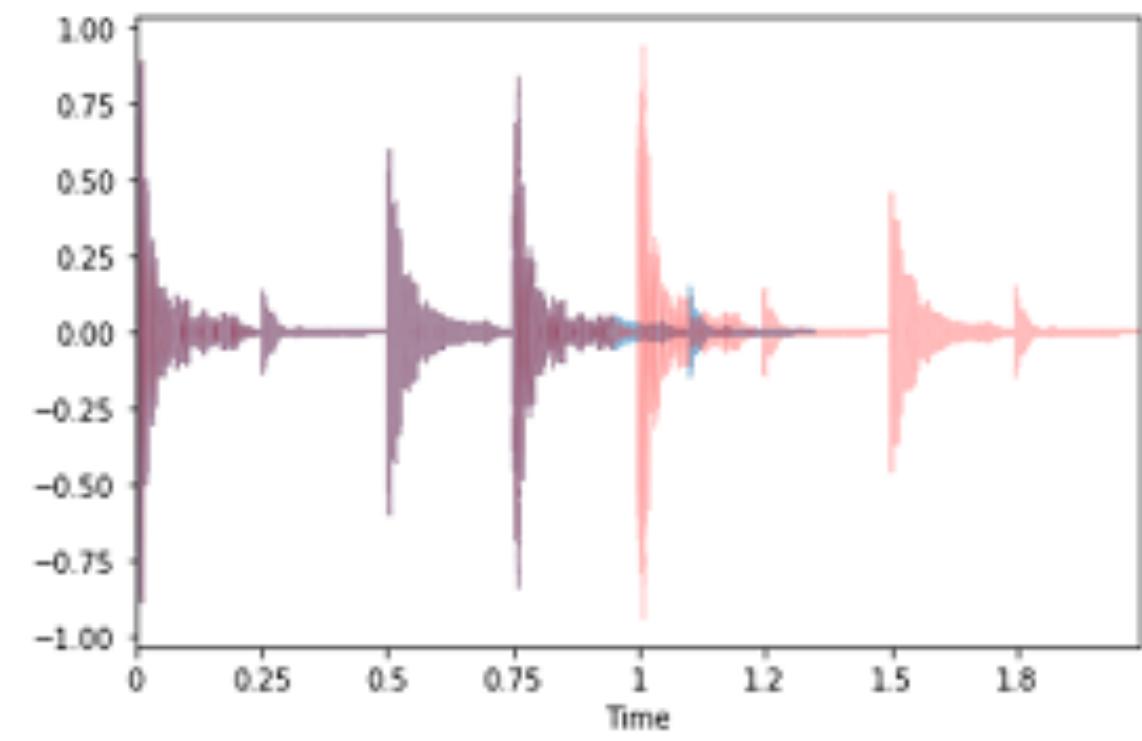


<https://ai.googleblog.com/2020/04/advancing-self-supervised-and-semi.html>

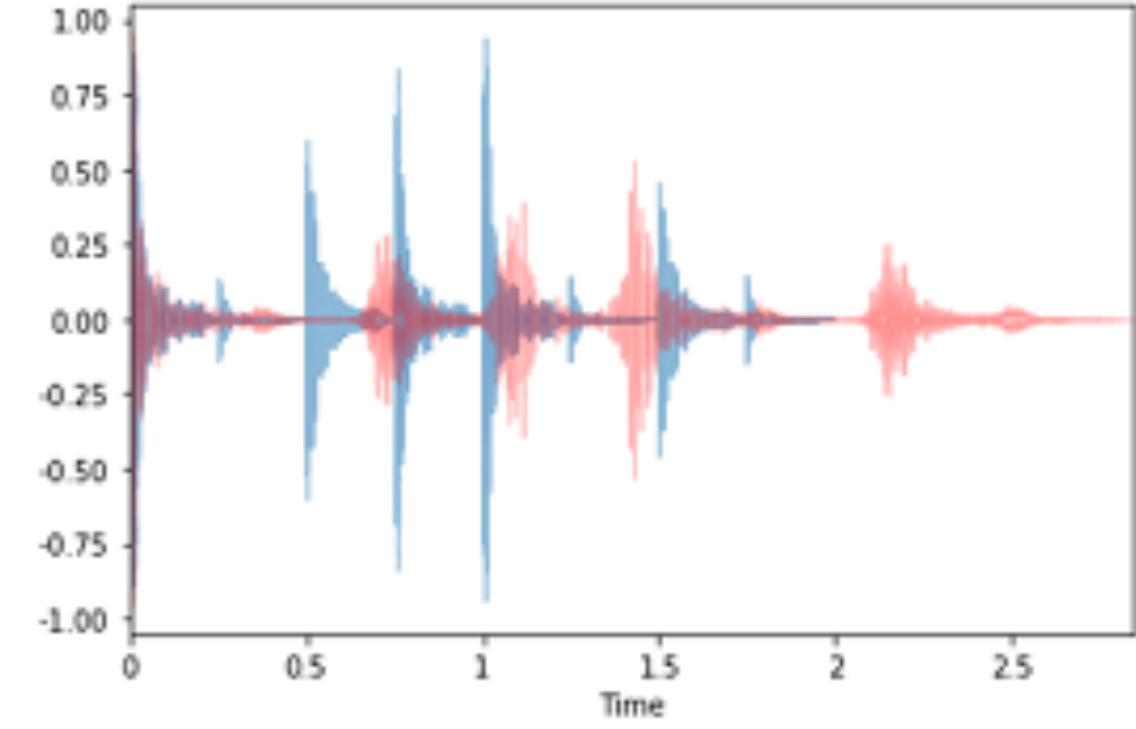
# Other data augmentation

- Tabular
  - Delete some cells to simulate missing data
- Text
  - No well established techniques, but replace words with synonyms, change order of things.
- Speech
  - Change speed, insert pauses, add audio effects

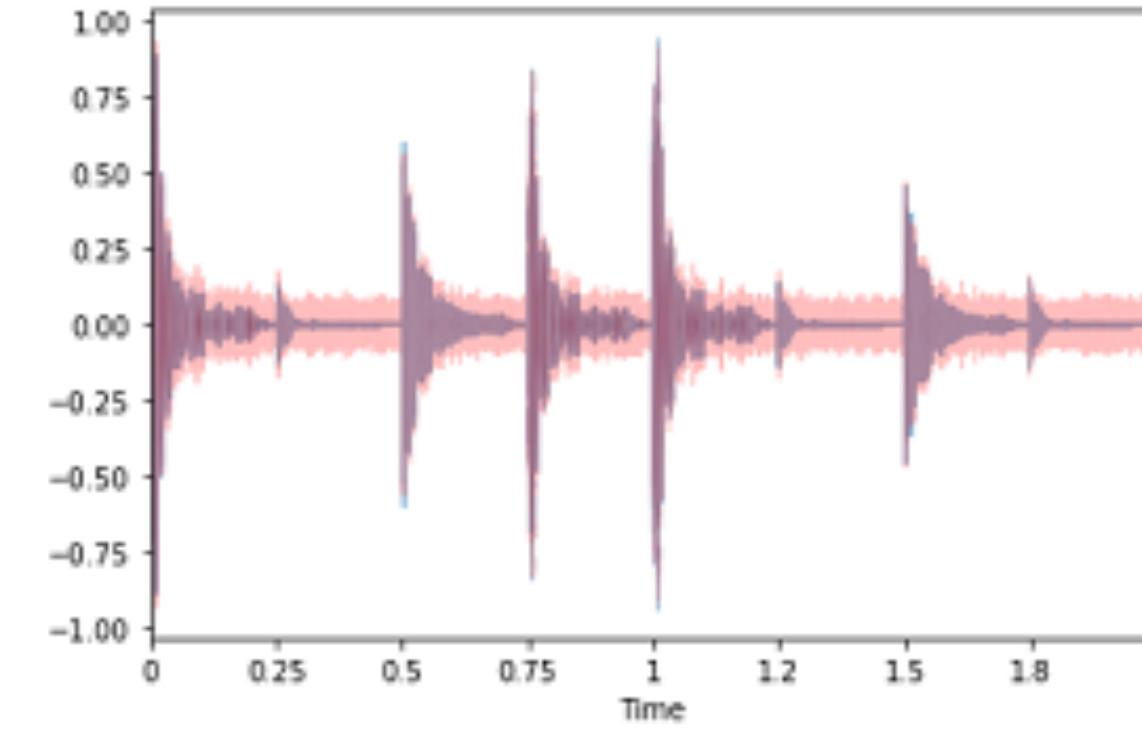
Cropping out a portion



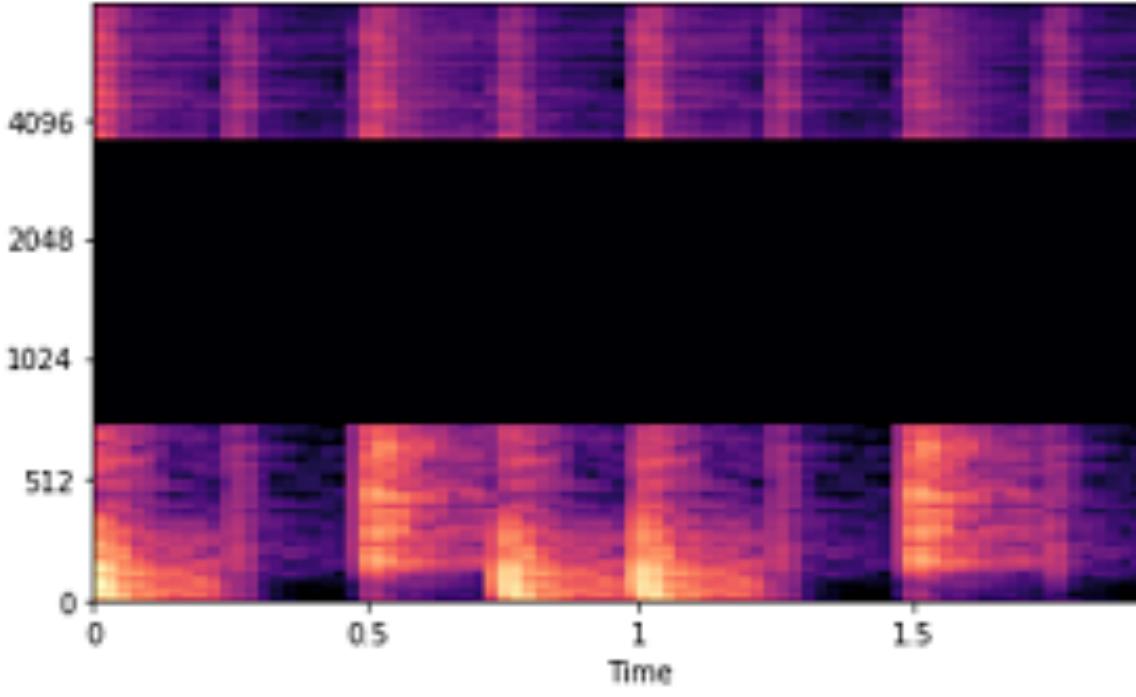
Changing Speed



Injecting Noise



Masking Frequency

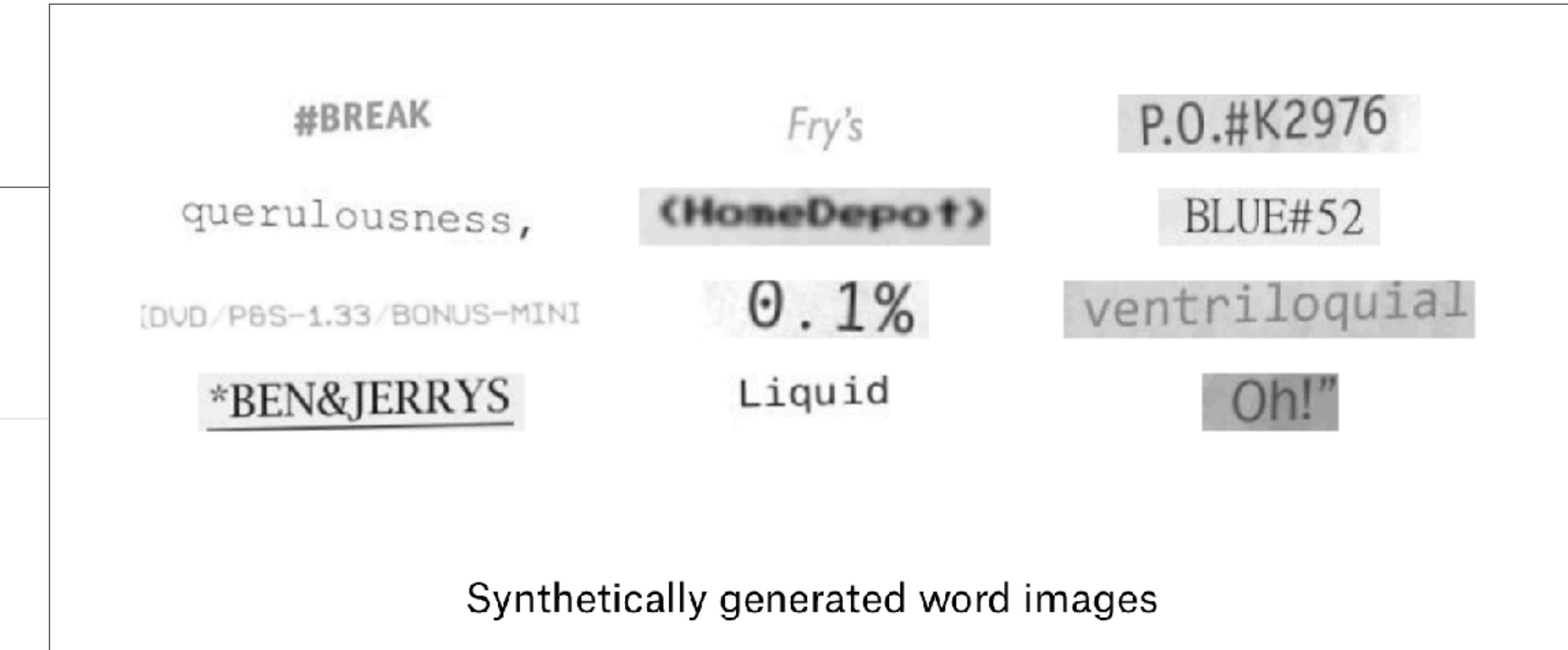


<https://github.com/makcedward/nlpaug>



# Synthetic data

Underrated idea that is often worth starting with



## Creating a Modern OCR Pipeline Using Computer Vision and Deep Learning

Brad Neuberg | April 12, 2017

728 0



# This can get pretty deep!



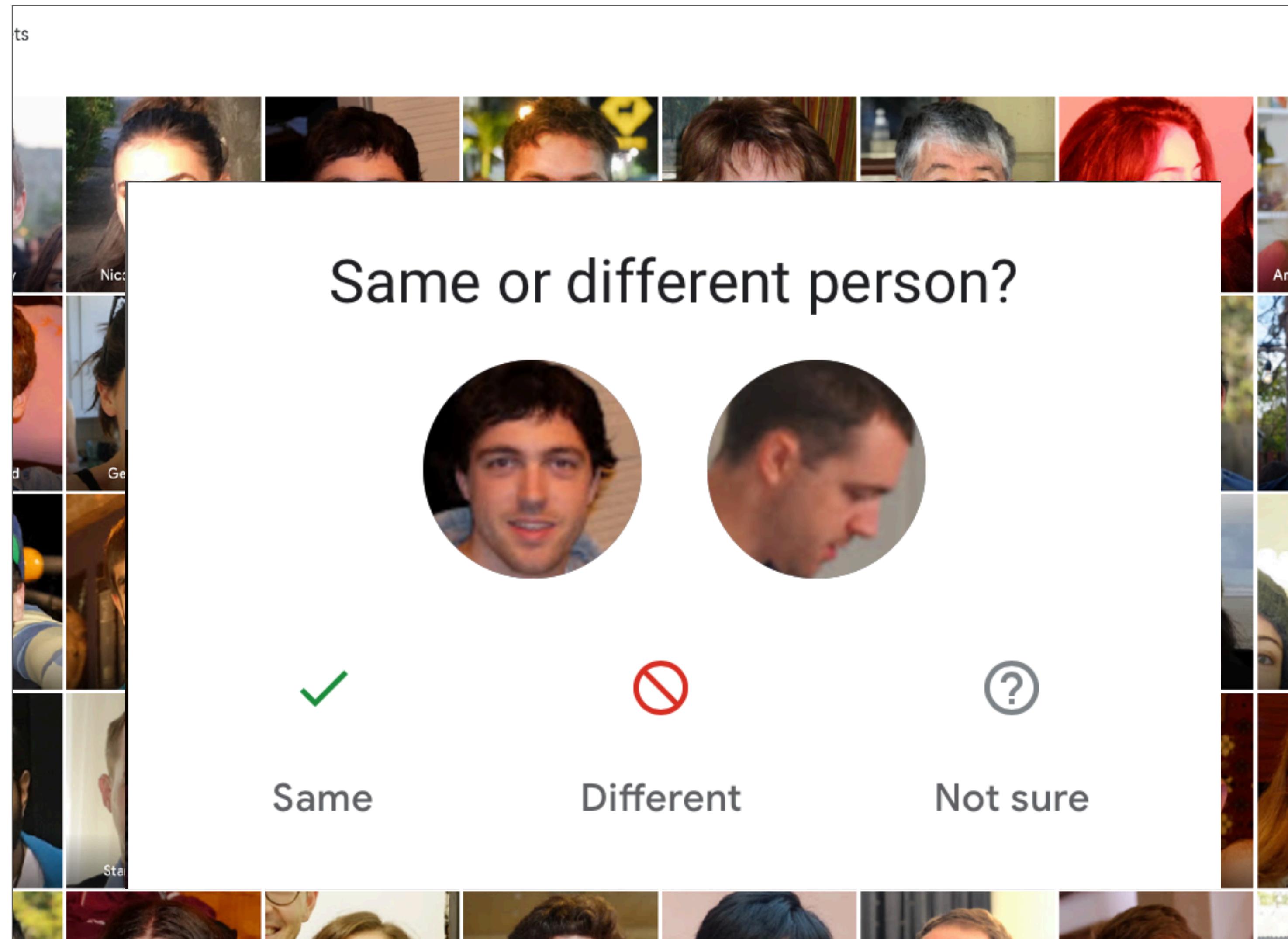
- Receipt crinkliness
- Receipt curvature
- Receipt alignment with camera
- Receipt ink fadedness
- Receipt paper glossiness
- Table material
- Camera flash
- Camera location
- Camera direction
- Camera exposure
- Camera focal distance
- Camera aperature size
- Environment ambient brightness
- Primary light location





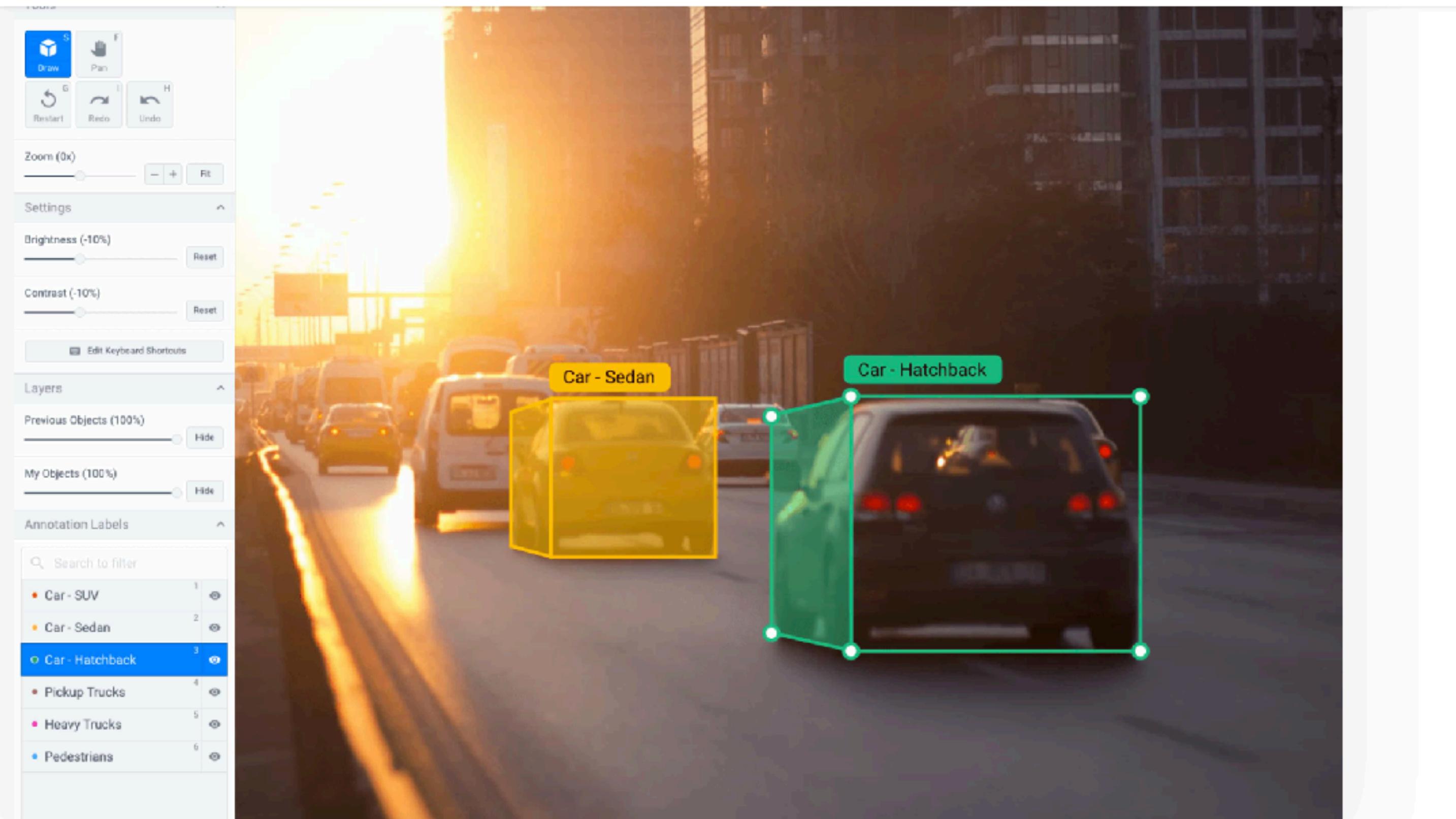
# Ask your users to label data for you!

Enables rapid improvement with user labels



# But usually: label data...





Standard set of features:

- bounding boxes, segmentations, keypoints, cuboids

- set of applicable classes



Categorization



Bounding Box



Semantic Segmentation



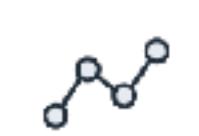
Line Annotation



Keypoint Annotation



Cuboid Annotation



Contour Annotation



Poly Mask Annotation

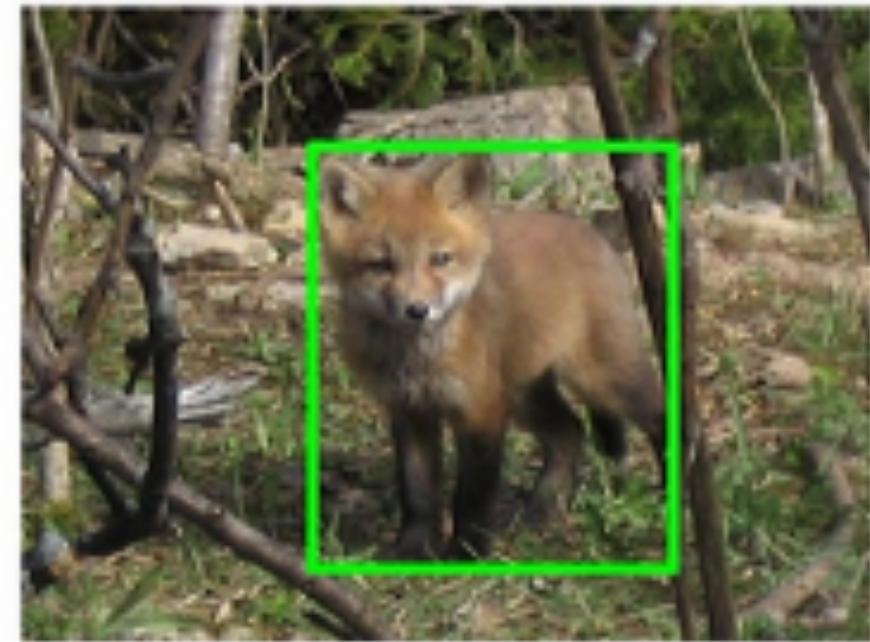


Polygonal Annotation



# Training the annotators is crucial

**Rule 1: Include all visible part and draw as tightly as possible.**



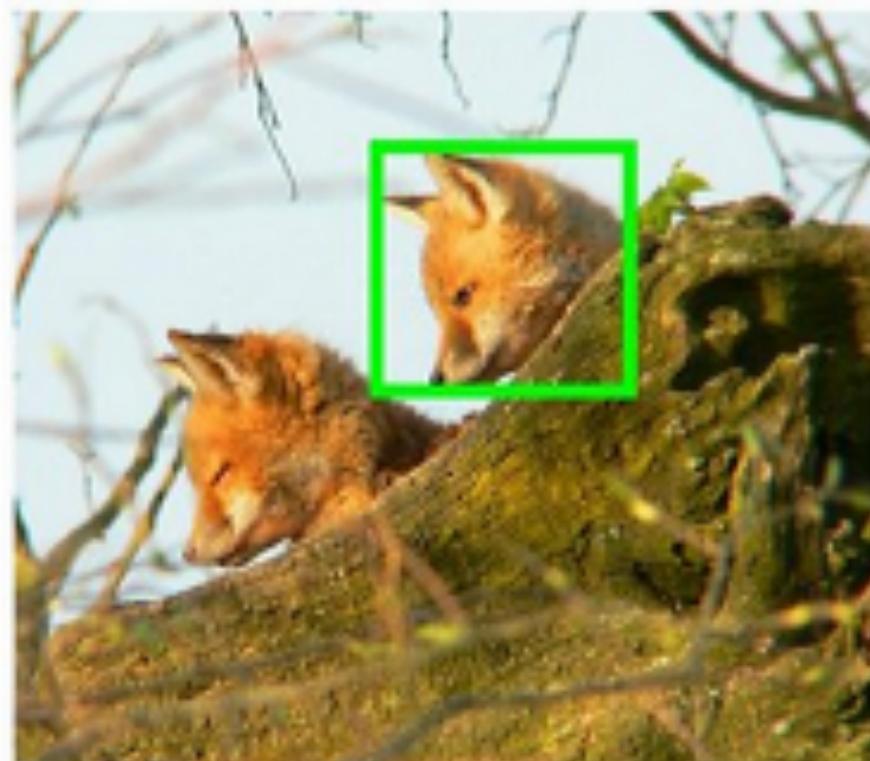
**CORRECT**



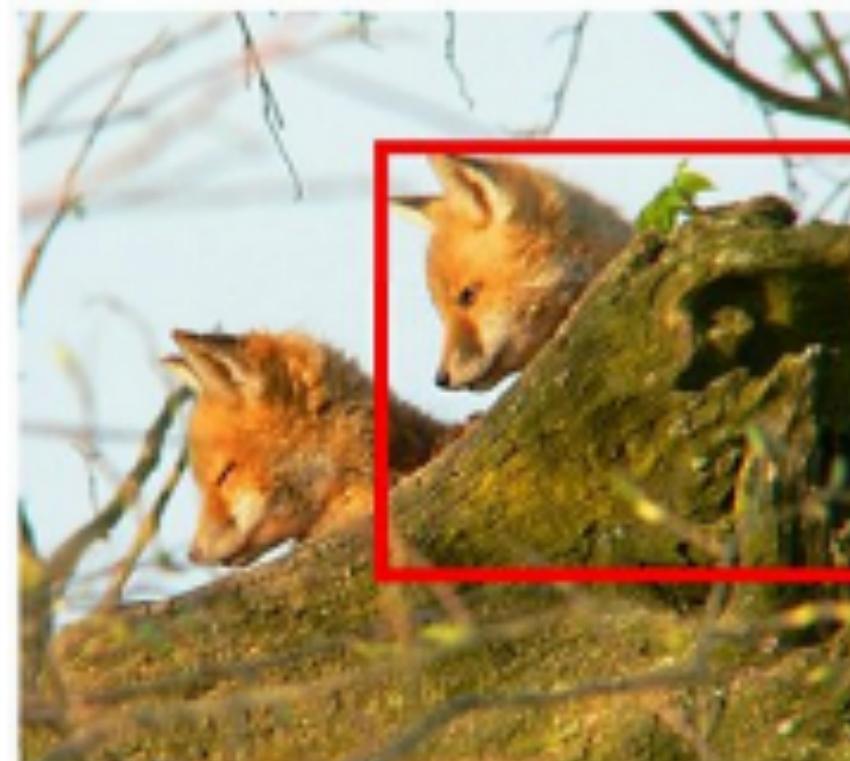
**WRONG: must be as tight as possible!**



**WRONG: must include all parts!**



**CORRECT**



**WRONG: occluded parts do not matter as long as all visible parts are included.**



Quality assurance is key

Source: cs.stanford.edu



# Sources of Labor

- Full-service data labeling
- Hire own annotators, promote best ones to quality control
- Crowdsource (Mechanical Turk)

# Full Service Companies

- Data labeling requires separate software stack, temporary labor, and quality assurance. Makes sense to outsource.
- Dedicate several days to selecting the best one for you:
  - Label gold standard data yourself
  - Sales calls with several contenders, ask for work sample on same data
  - Ensure agreement with your gold standard, and evaluate on value

# Scale.ai is a dominant data labeling solution

SELF DRIVING CARSDRONESROBOTICSAR & VRRETAIL

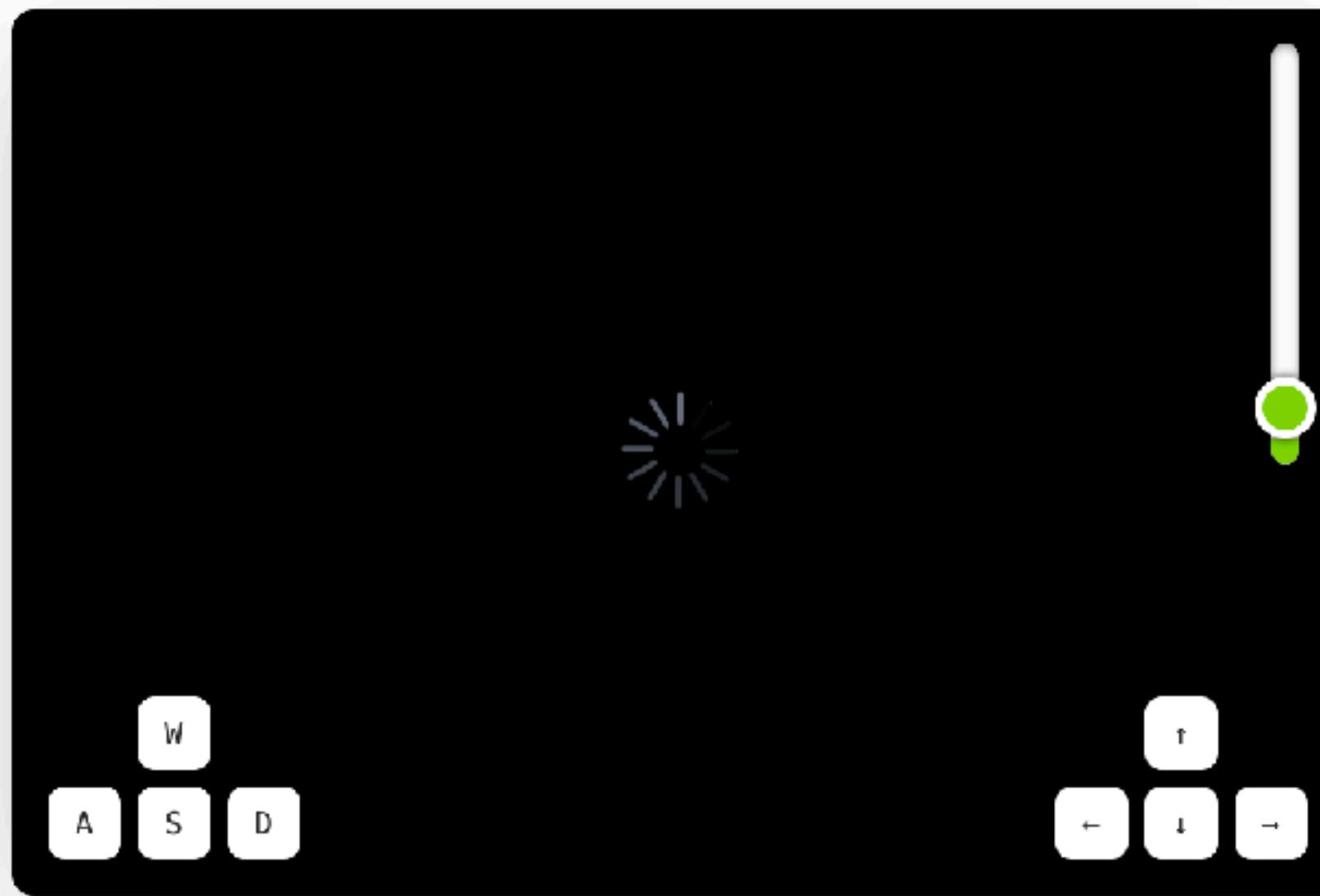


Sensor FusionVideoSemantic SegmentationCuboidsPolygons2D BoxesLines & Splines

"Please label all cars, pedestrians, and cyclists in each frame."

```
client.createLidarAnnotationTask({  
    'instruction': 'Please label all cars, pedestrians, and  
    cyclists in each frame.',  
    'labels': ['car', 'pedestrian', 'cyclist'],  
    'meters_per_unit': 2.3,  
    'max_distance_meters': 30  
}, (err, task) => {  
    // do something with task  
});
```

[RUN CODE](#)[READ MORE](#)





# And there are many others

Labelbox

Product Solutions Company Pricing Support Sign in ▾

**{BOT} SUPERVISELY**

NEW Introducing One-click Outsourcing

## The best way to create, manage training data

Labelbox is a collaborative training platform for artificial intelligence.

CREATE ACCOUNT CONTACT SALES

CONDÉ NAST AIRBUS

KEEP TRUCKIN STOCKWELL

Leverage best in class annotation platform

BOUNDING BOX POLYGON PIXELWISE TAG HOTKEYS LABELING HISTORY

chang-duong-480253-unplash  
demo2 > ds

3253x2169

Settings Screenshot Fullscreen Hotkeys Help max

IMAGES 1

chang-duong-480253-unplash 3 14 minutes ago

FIGURES 3 (68)

#	Class	%
#1	Class	19%
#2	table-lamp	4%
#3	laptop	11%
#4	plant	19%

HISTORY 1

Initial state



# Label Studio

- Open-source edition to run yourself
- Enterprise edition for managed hosting
- Using in lab!

2. Edit Labeling config

```
1 <View>
2   <!-- Image with Polygons -->
3   <View style="padding: 25px;
4     box-shadow: 2px 2px 8px #AAA">
5     <Header value="Label the image with polygons"/>
6     <Image name="img" value="$image"/>
7     <Text name="text1"
8       value="Select label, start to click on image"/>
9
10    <PolygonLabels name="tag" toName="img">
11      <Label value="Airbus" background="blue"/>
12      <Label value="Boeing" background="red"/>
13    </PolygonLabels>
14
15  </View>
16
17  <!-- Text with multi choices -->
18  <View style="margin-top: 20px; padding: 25px;
19    box-shadow: 2px 2px 8px #AAA">
20    <Header value="Classify the text"/>
```

3. Inspect Interface preview

Label the image with polygons

# Main modules

The main modules of LS are

- Label Studio Backend (LSB, main repository)
- Label Studio Frontend (LSF, editor)
- Machine Learning Backends (MLB)

Label Studio

Drone\_dataset : Unlabeled +

Tasks: 14 / 14 Completions: 0 Predictions: 0

Import Export Label

Grid Fields Filters Sort order Columns: 4 - +



# Diffgram



- Another open-source solution may be even better

The screenshot shows the Diffgram homepage. At the top, there's a navigation bar with links for Community, Docs, Product, Enterprise, Playground, Login, and Contact. The main heading is "Open Source Data Labeling Platform" with a subtitle "Image, Video, Text, 3D, Geo, & More. From Annotation to Feature Store." Below this are two buttons: "Try It Online" and "Install". A GitHub star icon indicates 1,458 stars. At the bottom, there's a section titled "Join the Open Source Movement" with a small illustration of a person interacting with a globe.

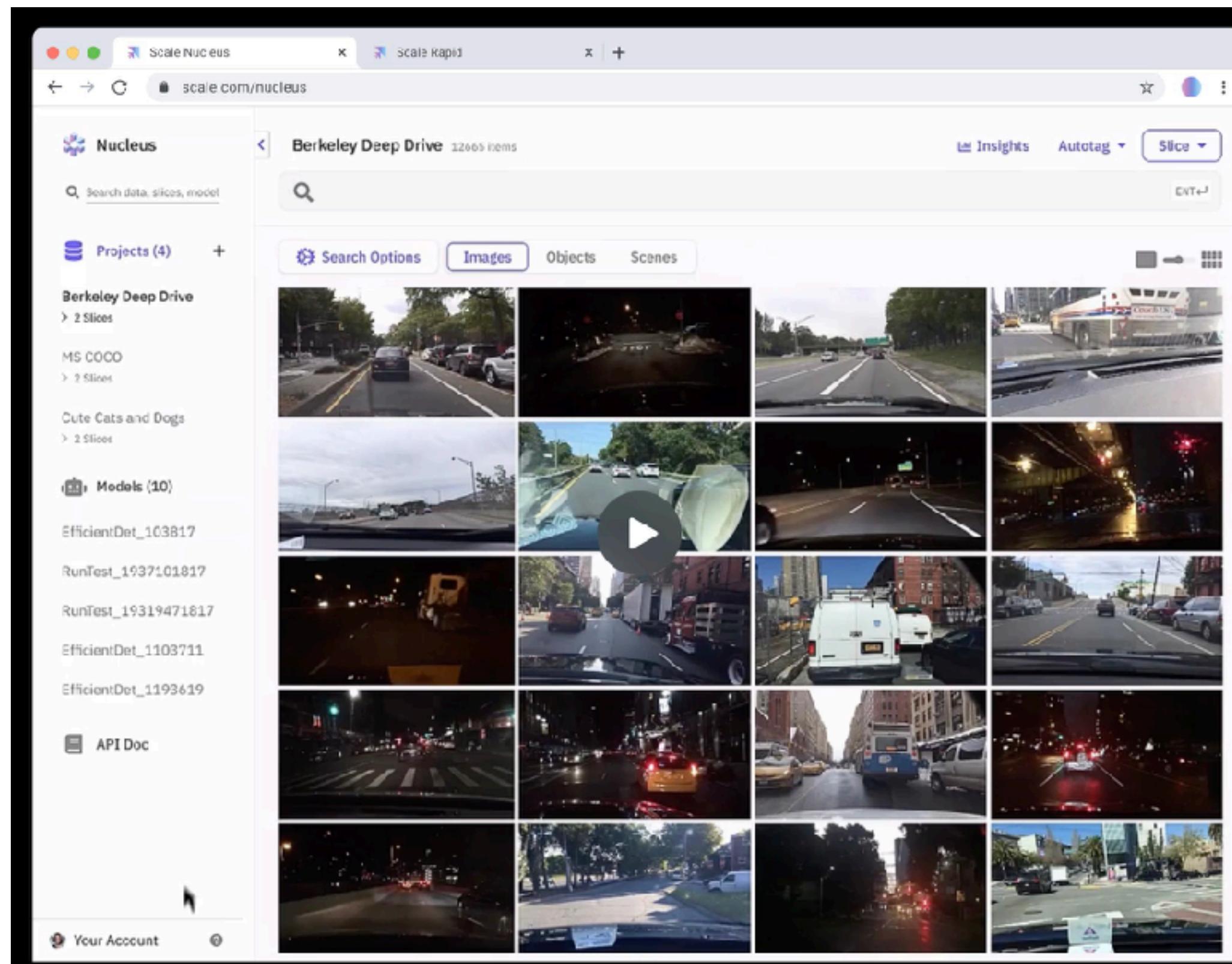
## Overall Diffgram has:

- Easier UI based customization
- More extensive automations
- Similar media type coverage
- Greater depth of image features
- Greater depth of video features
- Greater depth of text features
- 3D Support (Missing in Labelstudio)
- Feature Store – Ingest, Store, Query, Stream (Missing in Labelstudio)
- Better Scale
- Overall Richness of Features
- Ease of use, with everything available on the UI
- All in one product, truly open source

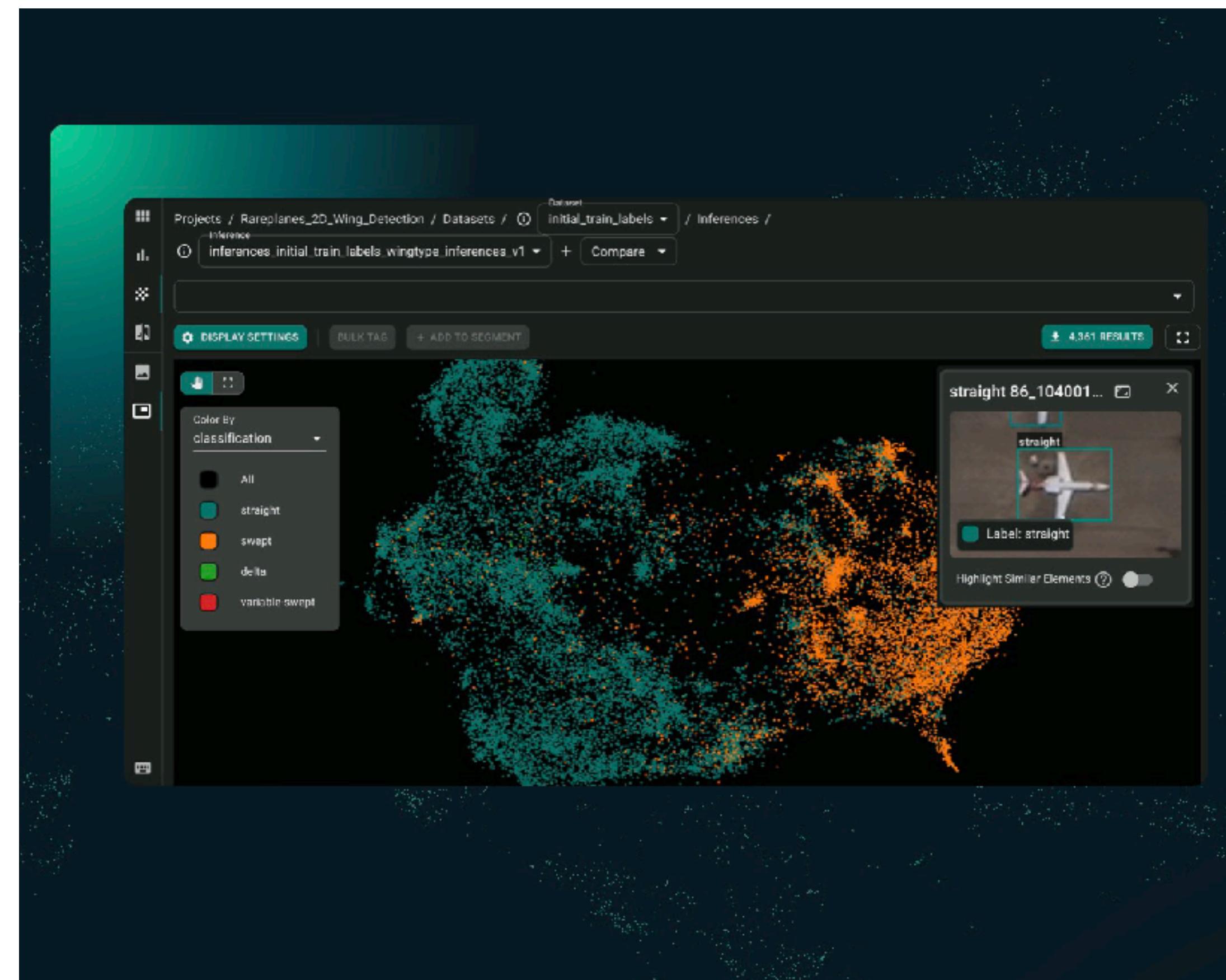


# Aquarium and Scale Nucleus

- Key feature: see where your current model performs poorly, and label that data



<https://scale.com/nucleus>



<https://www.aquariumlearning.com>



# Weak supervision

- Snorkel
  - Open-source project [snorkel.org](https://snorkel.org)
  - Commercial platform [snorkel.ai](https://snorkel.ai)
- Rubrix: open-source solution

The screenshot shows the Snorkel Flow interface. On the left, a card displays the text: "Label, augment, and build training data programmatically". Below this, a detailed description of Snorkel Flow's approach is provided. On the right, there is a "Labeling Function Builder" window with a configuration panel and a "Run & Save" button. To the right of the builder is a "Notebook" window containing Python code for defining a labeling function.

Label –

Label, augment, and build training data programmatically

Snorkel Flow starts with a fundamental departure from the status quo of labeling by hand: users develop programmatic operators to label, augment, and build training data. These range from simple but powerful push-button labeling functions — e.g., expressing rules or heuristics — to complex custom operators bringing diverse sources of signal to bear via the Python SDK. The result is a radically faster, more flexible interface to AI.

If HEADER CONTAINS Keyword employment Then label EMPLOYMENT

```
In [1]: from studio.bindings import StudioTask
task = StudioTask()

In [2]: from snorkel.Labeling.LF import Labeling_Function
@Labeling_Function(name="my_LF")
def lf_contract_amount(x):
    if x.contract.amount > 500000:
        return "SERVICES"
project.register(lf)
```

```
from snorkel.labeling import labeling_function

@labeling_function()
def lf_contains_link(x):

    # Return a label of SPAM if "http" in comment text, otherwise ABSTAIN

    return SPAM if "http" in x.text.lower() else ABSTAIN
```



# Conclusions

- Think of how you can do self-supervised learning
- Use labeling software and get to know your data by labeling it yourself for a while
- Write out detailed rules and outsource to full-service company if you can afford it
- Else, hiring part-time makes more sense than trying to make crowdsourcing work

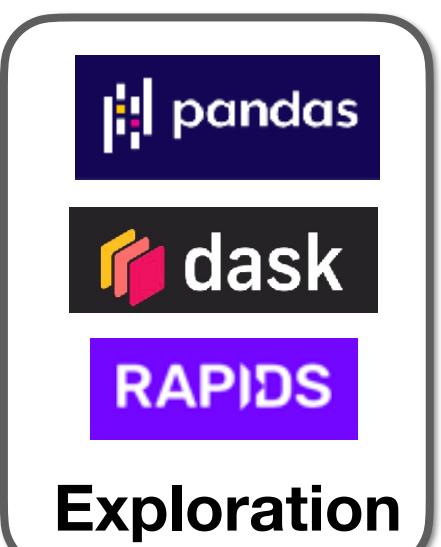
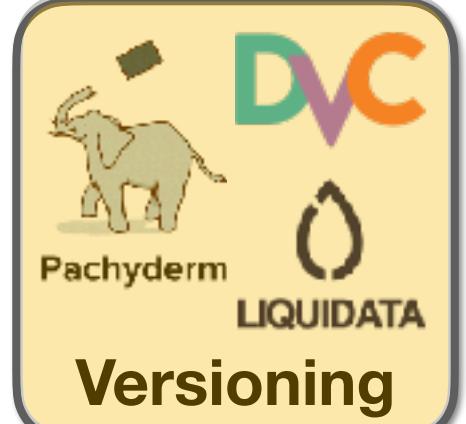
“All-in-one”



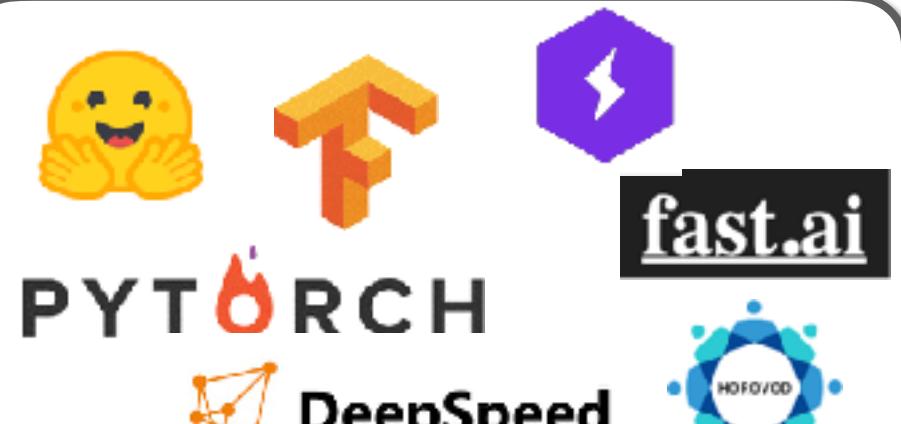
Amazon SageMaker

gradient<sup>o</sup>  
by Paperspace

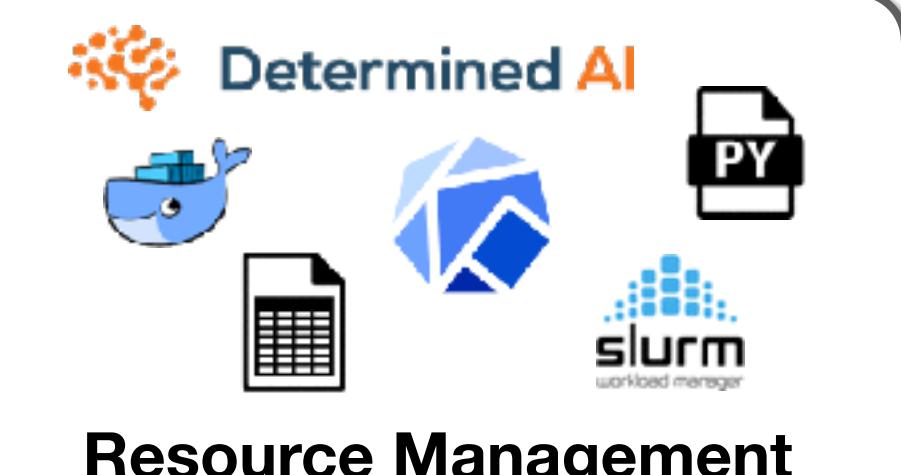
DOMINO  
DATA LAB



Data



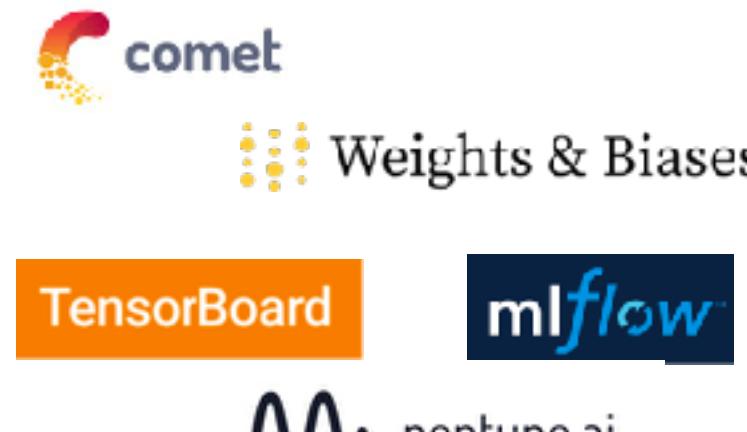
Frameworks &  
Distributed Training



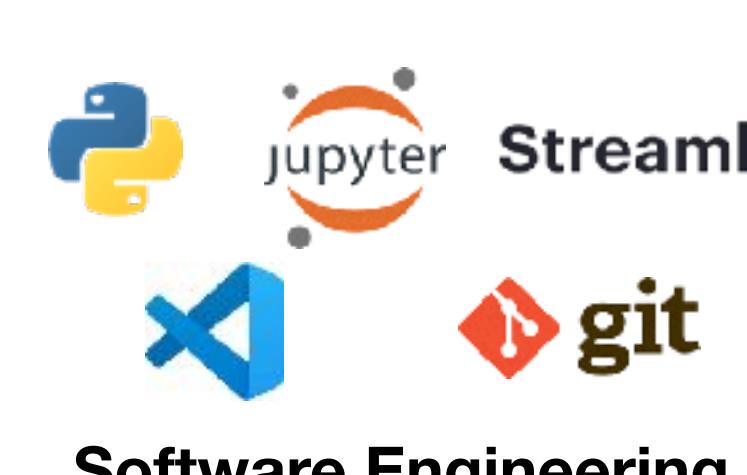
Resource Management



Development



Experiment and Model  
Management



Software Engineering



Edge



CI / Testing



Deployment





# Data Versioning

Level 0: unversioned

Level 1: versioned via snapshot at training time

Level 2: versioned as a mix of assets and code

Level 3: specialized data versioning solution



# Level 0



- Data lives on filesystem/S3 and database
- **Problem:** Deployed machine learning models are part code, part data. If data is not versioned, deployed models are not versioned.
- Problem you **will** face: inability to get back to a previous level of performance



# Level 1



- Data is versioned by storing a snapshot of everything at training time
- This kind of works, but would be far better to be able to version data just as easily as code.



## Level 2



- Data is versioned as a mix of assets and code.
- Heavy files stored in S3, with unique ids. Training data is stored as JSON or Parquet, referring to these ids and include relevant metadata (labels, user activity, etc).
- Data files can get big, but using **git-lfs** lets us store them just as easily as code.



# Level 3



- Specialized solutions for versioning data, usually helping you store large files.
- Could totally make sense, but don't assume you need it right away!
- Leading solution is DVC.



# Data Versioning Solutions

	Open Source	Data Format Agnostic	Cloud/Storage Agnostic* (Supports most common cloud and storage types)	Simple to Use	Easy Support for BIG Data
	 (Apache 2.0)				
	 (Apache 2.0)				
	 (MIT)				
	 ! (Non-standard license)				
	 (Apache 2.0)				
	 (Apache 2.0)				



Warning: This is biased toward DVC



# DVC

## Open-source Version Control System for Machine Learning Projects

2

```
$ dvc run \
  -d prepare.py -d data.xml \
  -o data.tsv -o data-test.tsv \
  python prepare.py data.xml
```

3 The first stage, feature extraction:

```
$ dvc run -d featurization.py -d data.tsv \
  -o matrix.pkl \
  python featurization.py data.tsv matrix.pkl
```

The second stage, training:

```
$ dvc run -d train.py -d matrix.pkl \
  -o model.pkl \
  python train.py matrix.pkl model.pkl
```

Let's commit meta-files that describe our pipeline:

```
$ git add .gitignore matrix.pkl.dvc model.pkl.dvc
$ git commit -m "add featurization and train steps to the pipeline"
```

1 \$ dvc add data.xml

DVC stores information about your data file in a special `.dvc` file, that has a human-readable [description](#) and can be committed to Git to track versions of your file:

```
$ git add .gitignore data.xml.dvc
$ git commit -m "add source data to DVC"
```

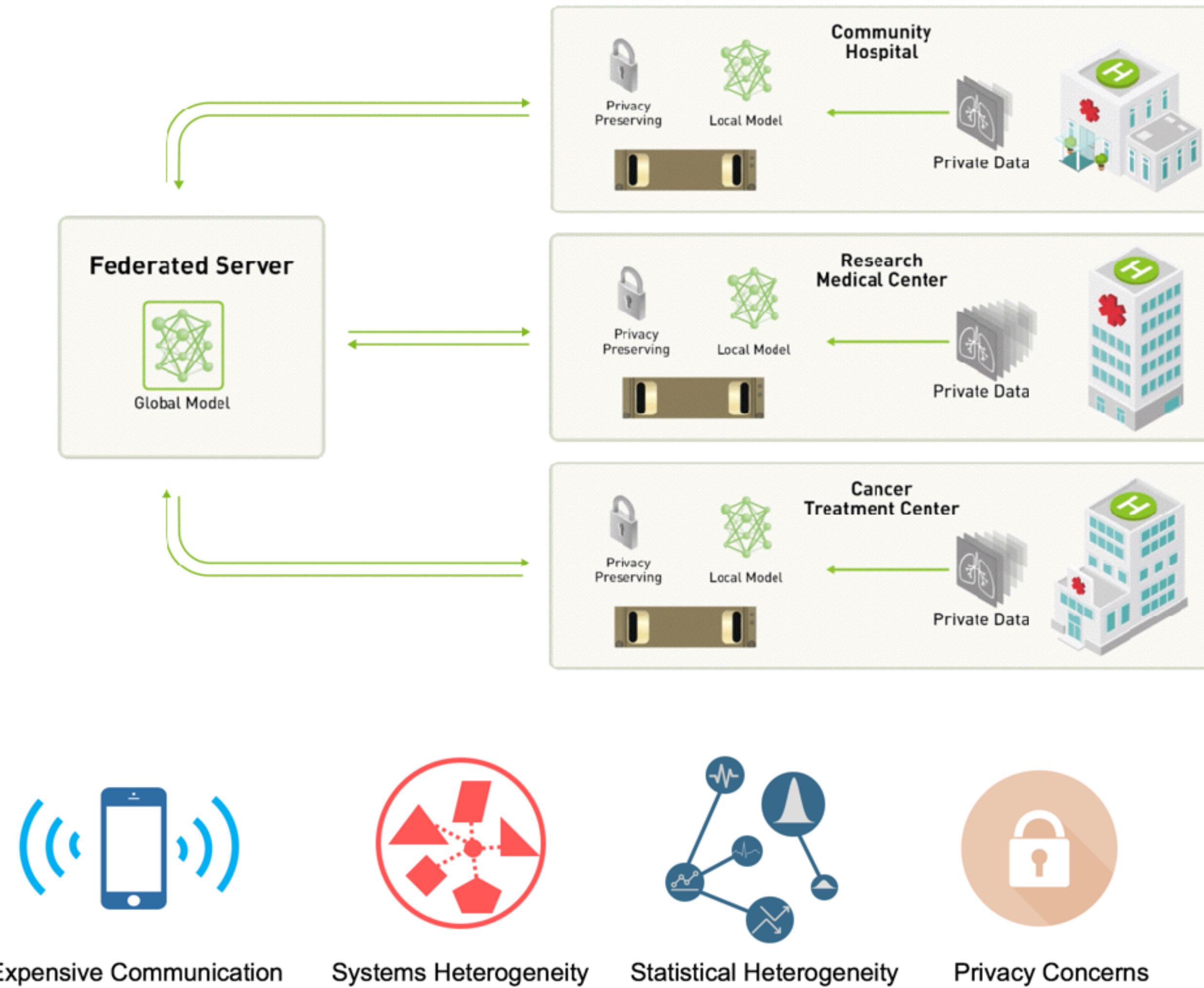
4

```
$ dvc pipeline show --ascii model.pkl.dvc --outs
  -----
  | data.xml |
  -----
  **      **
  **      **
  *      *
  -----
  | data.tsv |      | data-test.tsv |
  -----
  **      **
  **      **
  *      *
  -----
  | matrix.pkl |
  -----
  *      *
  *      *
  -----
  | model.pkl |
  -----
```



# Research Area: Privacy

- Federated Learning: training a global model from data on local devices, without ever having access to the data
- Differential privacy: aggregating data such that individual points cannot be identified
- Another topic: Learning on encrypted data



<https://blog.ml.cmu.edu/2019/11/12/federated-learning-challenges-methods-and-future-directions/>

<https://blogs.nvidia.com/blog/2019/10/13/what-is-federated-learning/>



Thank you!