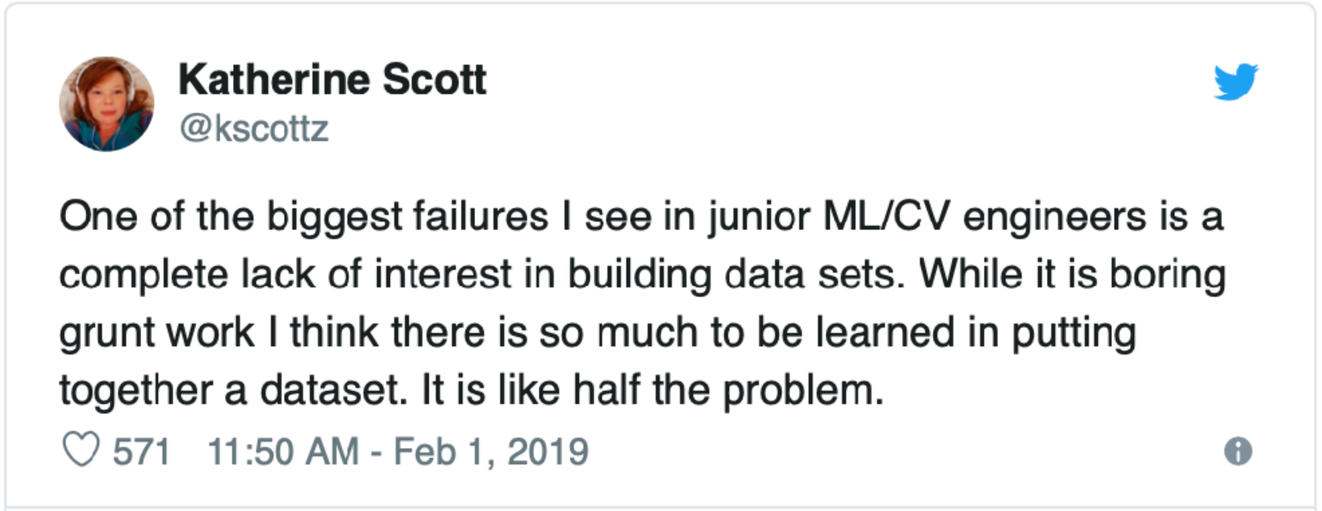
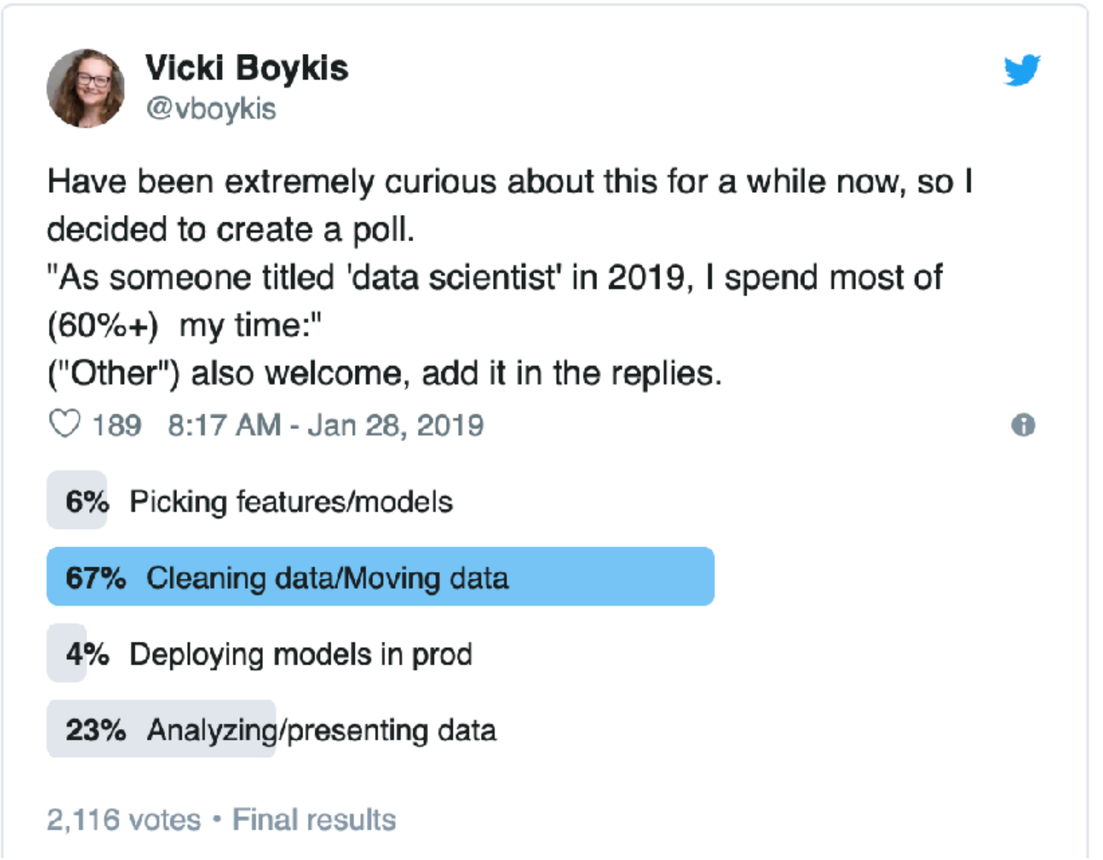
FSDL 2022

Data Management

Sergey Karayev

August 29, 2022





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 by Paperspace

DOMINO DATA LAB

Data

Versioning: Pachyderm, DVC, LIQUIDATA

Labeling: Label Studio, scale, prodigy

Processing: dbt, DAGSTER

Exploration: pandas, dask, RAPIDS

Datasets

Sources: S3, databricks, Parquet, snowflake

Development

Frameworks & Distributed Training: PYTORCH, fast.ai, DeepSpeed, FairScale

Experiment and Model Management: comet, Weights & Biases, TensorBoard, mlflow, neptune.ai

Resource Management: Determined AI, slurm

Software Engineering: jupyter, Streamlit, git

Compute: CoreWeave, Lambda

Deployment

Feature Store: tecton

Monitoring: fiddler

Edge: OctoML, NVIDIA TensorRT, TensorFlow Lite, ONNX

Web: ALGORITHMIA, SELDON

CI / Testing: Buildkite, great_expectations

or or

"All-in-one"

Amazon SageMaker

gradient by Paperspace

DOMINO DATA LAB

Data

Versioning: Pachyderm, DVC, LIQUIDATA

Labeling: Label Studio, scale, prodigy

Processing: dbt, DAGSTER

Exploration: pandas, dask, RAPIDS

Datasets

Sources: S3, databricks, Parquet, snowflake

Development

Frameworks & Distributed Training: PYTORCH, fast.ai, DeepSpeed, FairScale

Experiment and Model Management: comet, Weights & Biases, TensorBoard, mlflow, neptune.ai

Resource Management: Determined AI, slurm

Software Engineering: jupyter, Streamlit, git

Compute: CoreWeave, Lambda

Deployment

Feature Store: tecton

Monitoring: fiddler

Edge: OctoML, NVIDIA TensorRT, TensorFlow Lite, ONNX

Web: ALGORITHMIA, SELDON

CI / Testing: Buildkite, great_expectations

or or

Many possibilities

Data Sources

- Images
- Text Corpus
- Logs
- DB records

Different for every project / company!

Training

Local Filesystem

+

GPU

Many possibilities

Data Sources

Training

Images

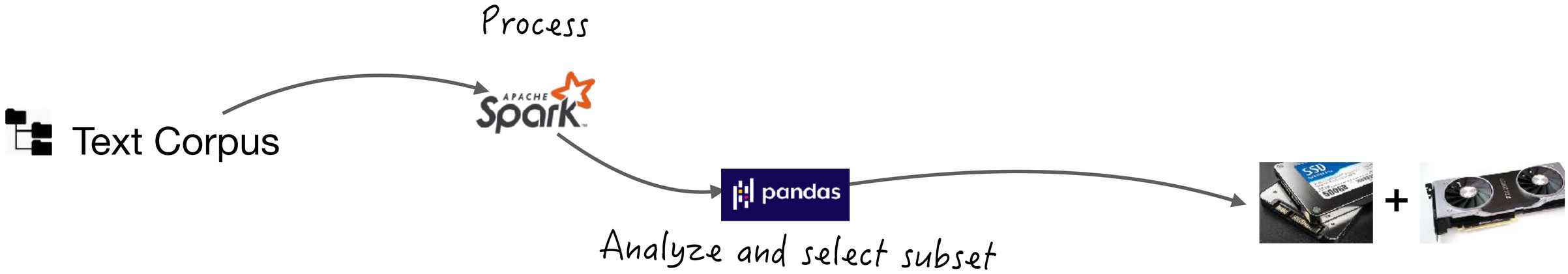
Simply Download

+

Many possibilities

Data Sources

Training



Many possibilities

Data Sources

Training

Aggregate and process

snowflake

Logs

DB records

+

Many possibilities

Data Sources

- Images
- Text Corpus
- Logs
- DB records

Different for every project / company!

Training

Local Filesystem

+

GPU

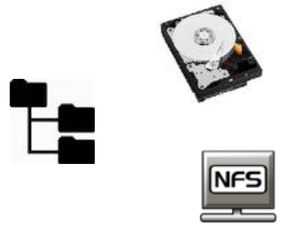
The Basics

- 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)

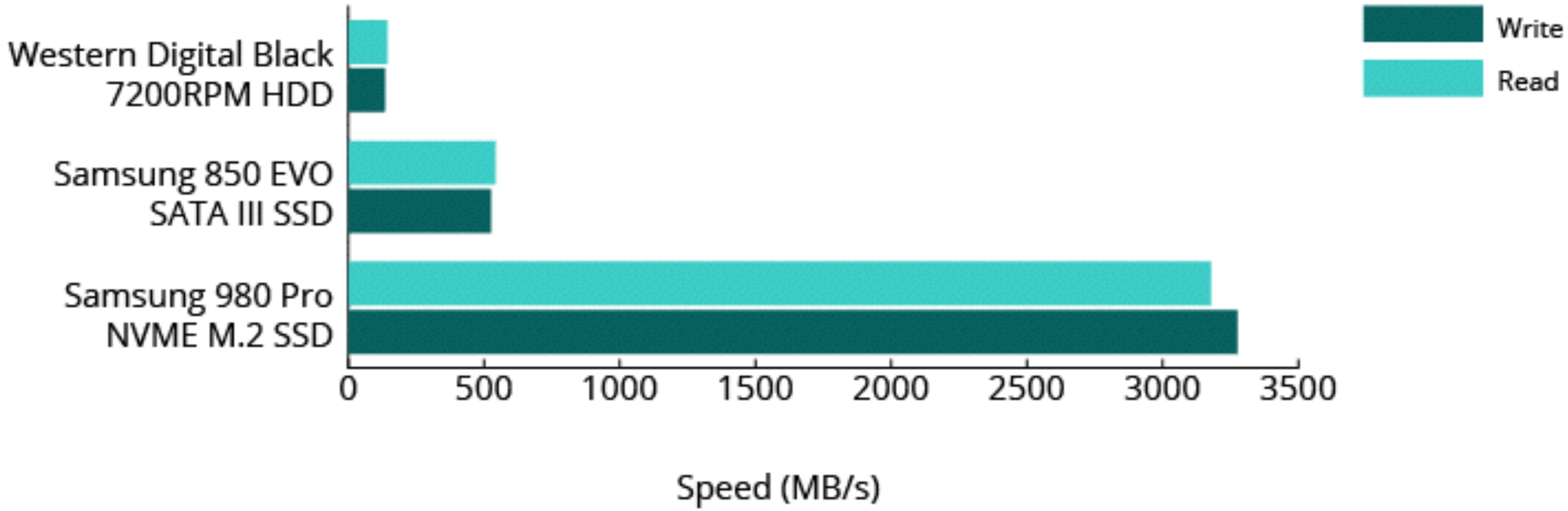
Local Disk Speeds

Almost 2 orders of magnitude difference!

HDD

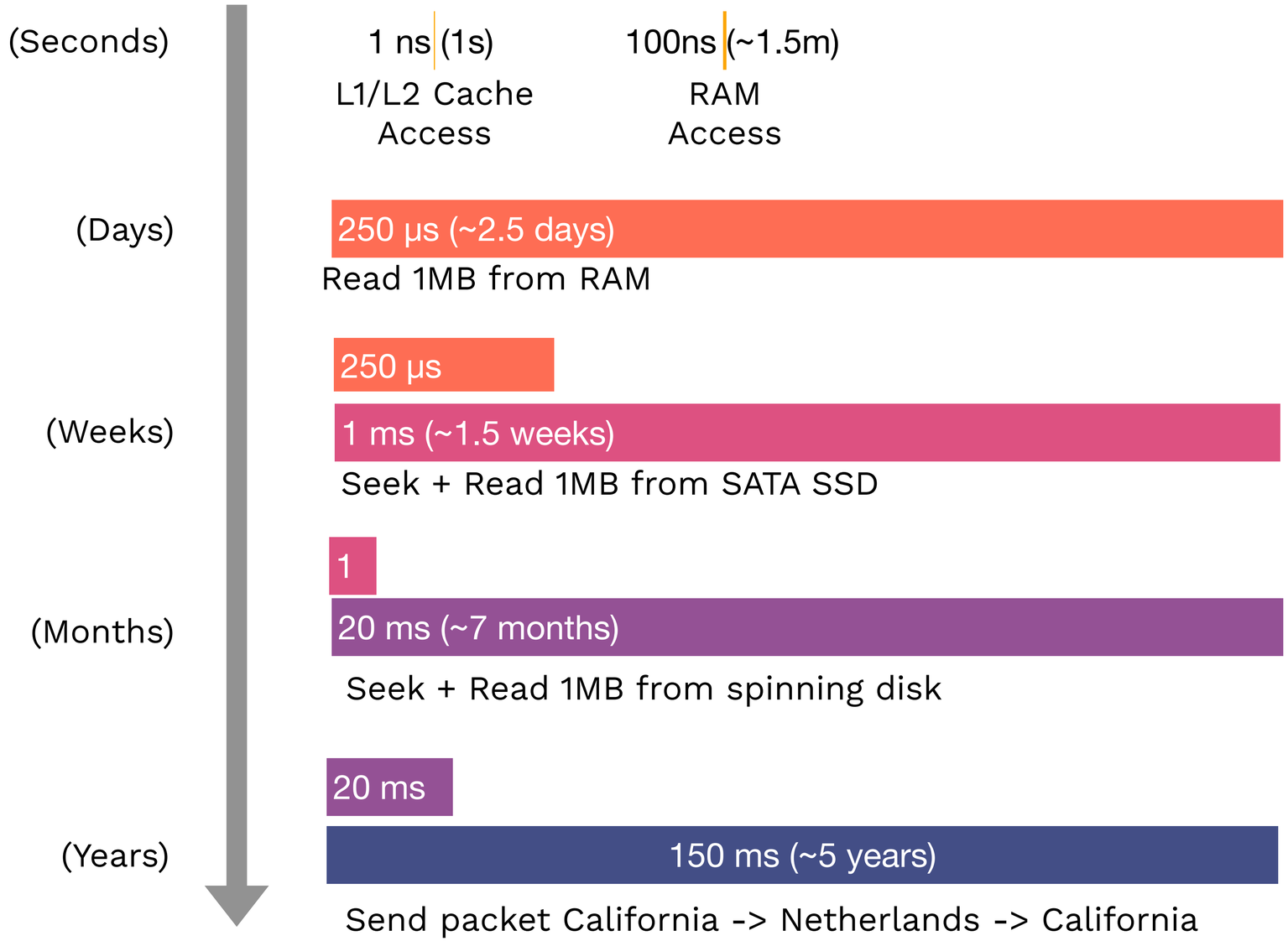
SATA SSD

NVMe SSD



https://voltcave.com/ssd-vs-hdd/

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



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

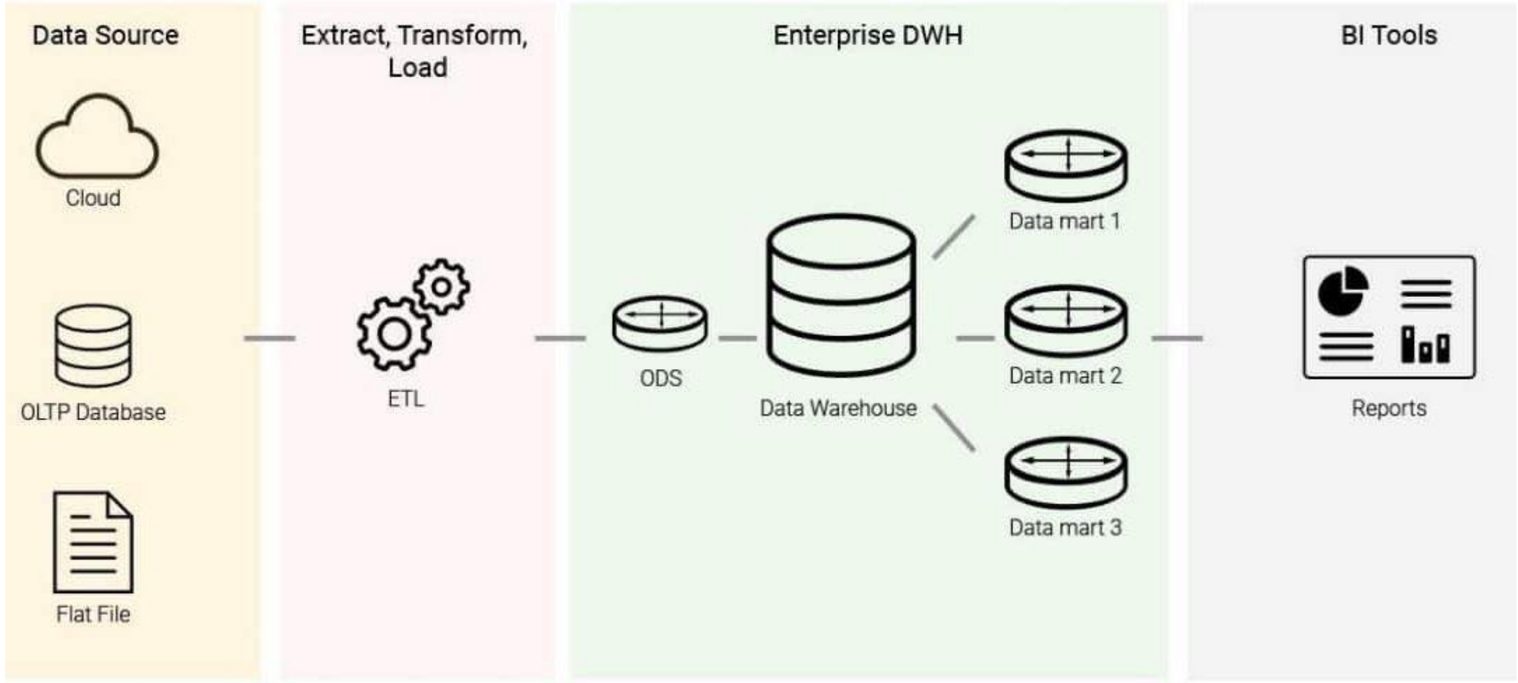
- Code that deals with collections of objects that reference each other (e.g. a Text is from a Document, which has an Author) **will eventually implement a crappy database**
- Using a database from the beginning will likely save time
- Many MLOps tools are databases at their core (e.g W&B is a DB of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

Data Warehouse

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

Google Cloud BigQuery snowflake amazon REDSHIFT

- 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`

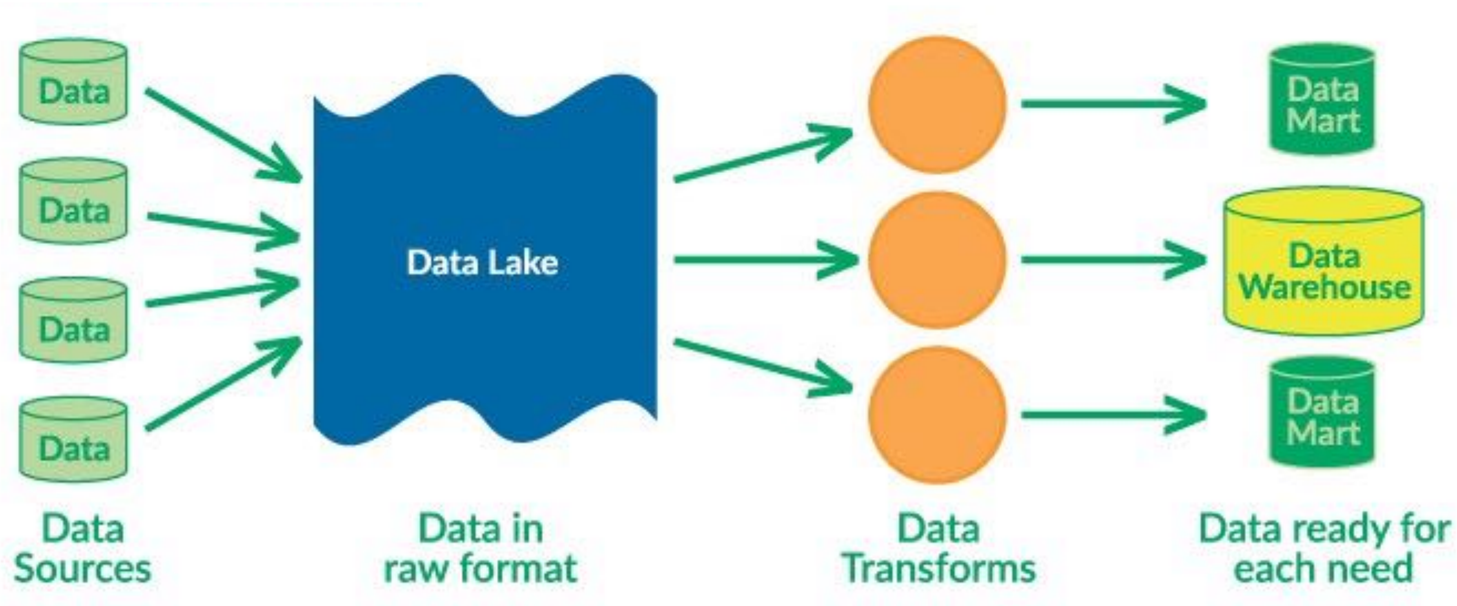


https://addepto.com/implement-data-warehouse-business-intelligence/

Data Lake

- Unstructured aggregation of data from multiple sources, e.g. databases, logs, expensive data transformations.

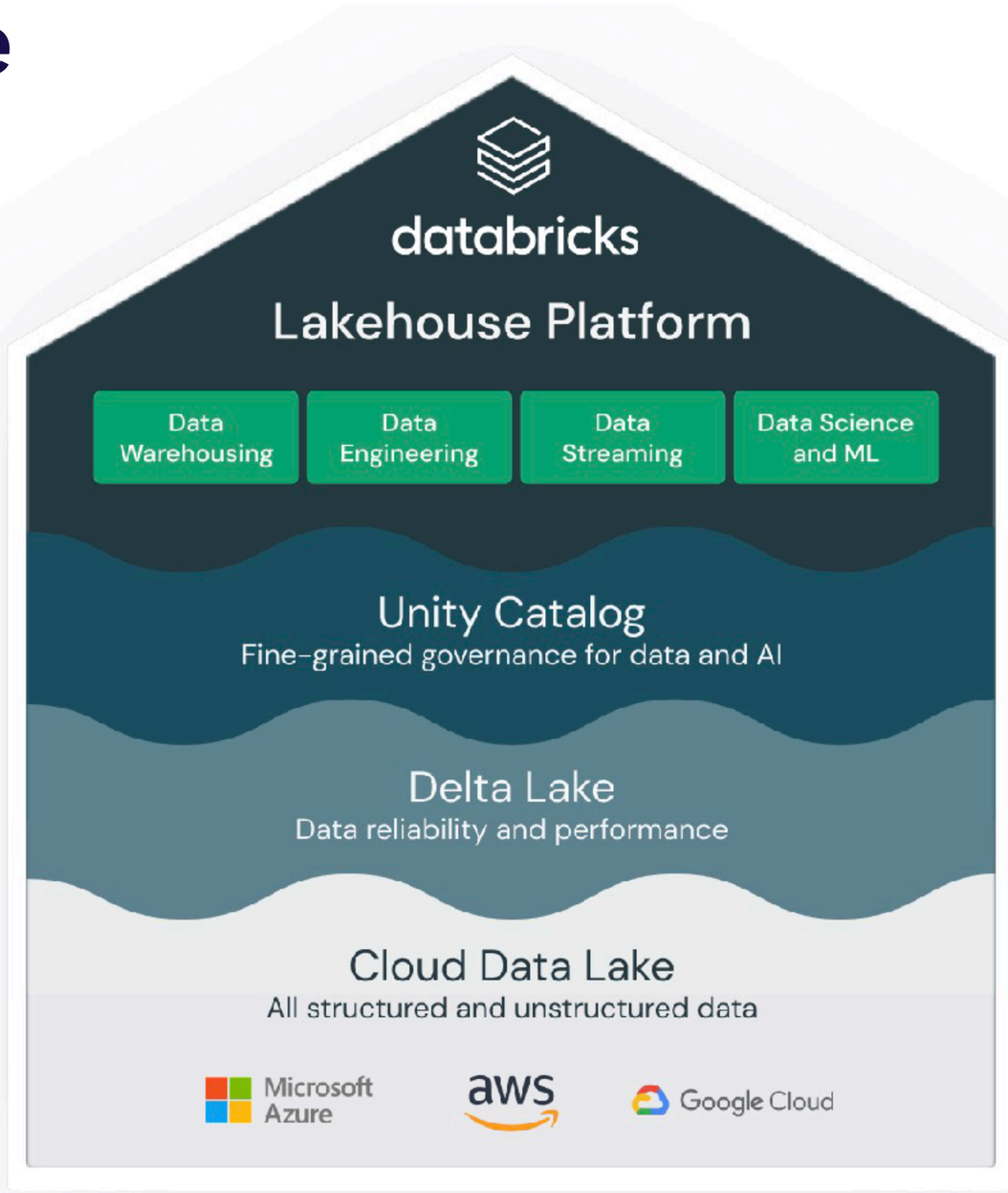
- **ELT**: dump everything in, then transform for specific needs later.



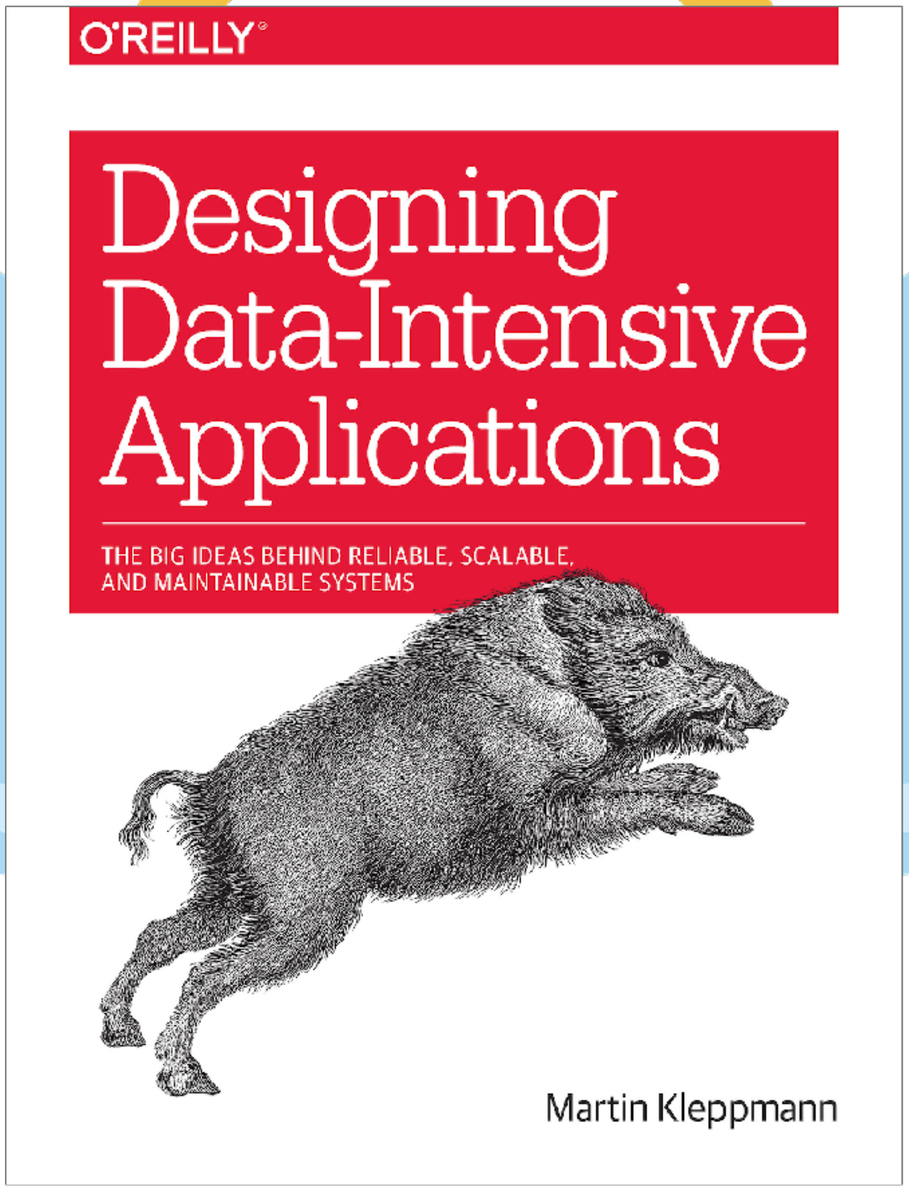
https://medium.com/data-ops/throw-your-data-in-a-lake-32cd21b6de02

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 by Paperspace

DOMINO DATA LAB

Data

Versioning: Pachyderm, DVC, LIQUIDATA

Labeling: Label Studio, scale, prodigy

Processing: dbt, DAGSTER

Exploration: pandas, dask, RAPIDS

Datasets

Sources: S3, databricks, Parquet, snowflake

Development

Frameworks & Distributed Training: PYTORCH, fast.ai, DeepSpeed, FairScale

Experiment and Model Management: comet, Weights & Biases, TensorBoard, mlflow, neptune.ai

Resource Management: Determined AI, slurm

Software Engineering: jupyter, Streamlit, git

Compute: CoreWeave, Lambda

Deployment

Feature Store: tecton

Monitoring: fiddler

Edge: OctoML, NVIDIA TensorRT, TensorFlowLite, ONNX

Web: ALGORITHMIA, SELDON

CI / Testing: Buildkite, great_expectations

or or

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

```sql
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
```

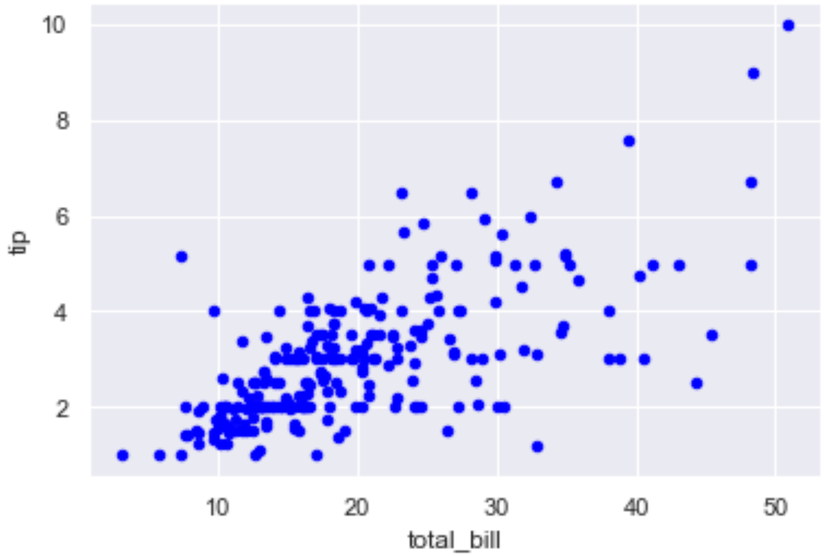
https://pandas.pydata.org/docs/getting_started/comparison/comparison_with_sql.html

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]: <matplotlib.axes._subplots.AxesSubplot at 0xc5120b0>
```



dask

RAPIDS
Open GPU Data Science

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

“All-in-one”

Amazon SageMaker

gradient by Paperspace

DOMINO DATA LAB

Data

Versioning
Pachyderm, DVC, LIQUIDATA

Labeling
Label Studio, scale, prodigy

Processing
dbt, DAGSTER

Exploration
pandas, dask, RAPIDS

Datasets

Sources
S3, databricks, Parquet, snowflake

Development

Frameworks & Distributed Training
PYTORCH, fast.ai, DeepSpeed, FairScale

Experiment and Model Management
comet, Weights & Biases, TensorBoard, mlflow, neptune.ai

Resource Management
Determined AI, slurm

Software Engineering
jupyter, Streamlit, git

Compute
CoreWeave, Lambda

Deployment

Feature Store
tecton

Monitoring
fiddler

Edge
ML, OctoML, NVIDIA TensorRT, TensorFlow Lite, ONNX

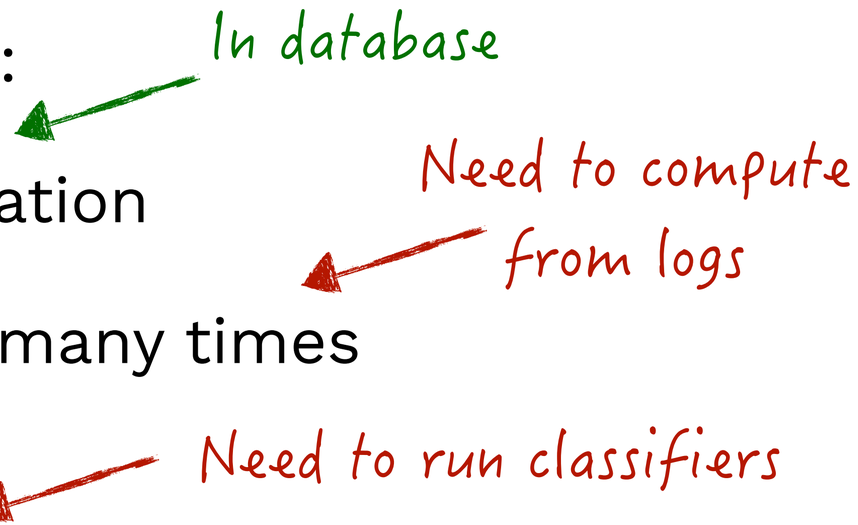
Web
ALGORITHMIA, SELDON

CI / Testing
Buildkite, great_expectations

or or

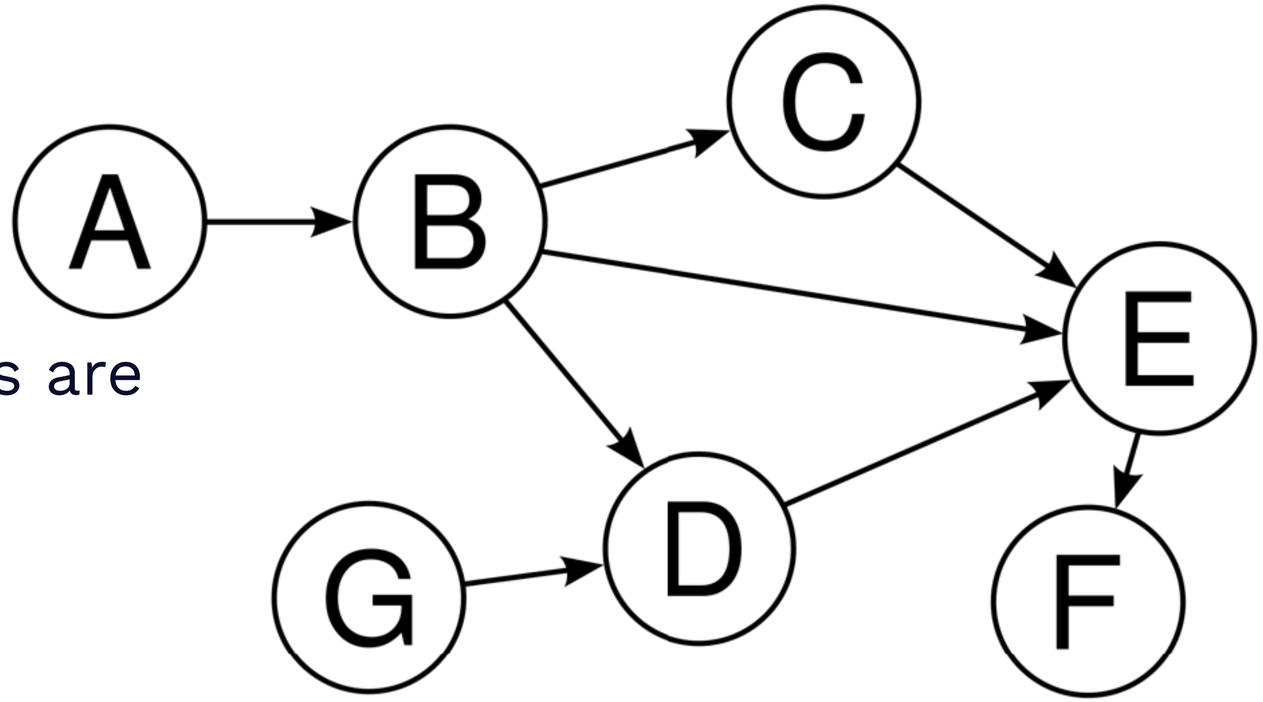
Motivational Example

- 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)



Task Dependencies

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

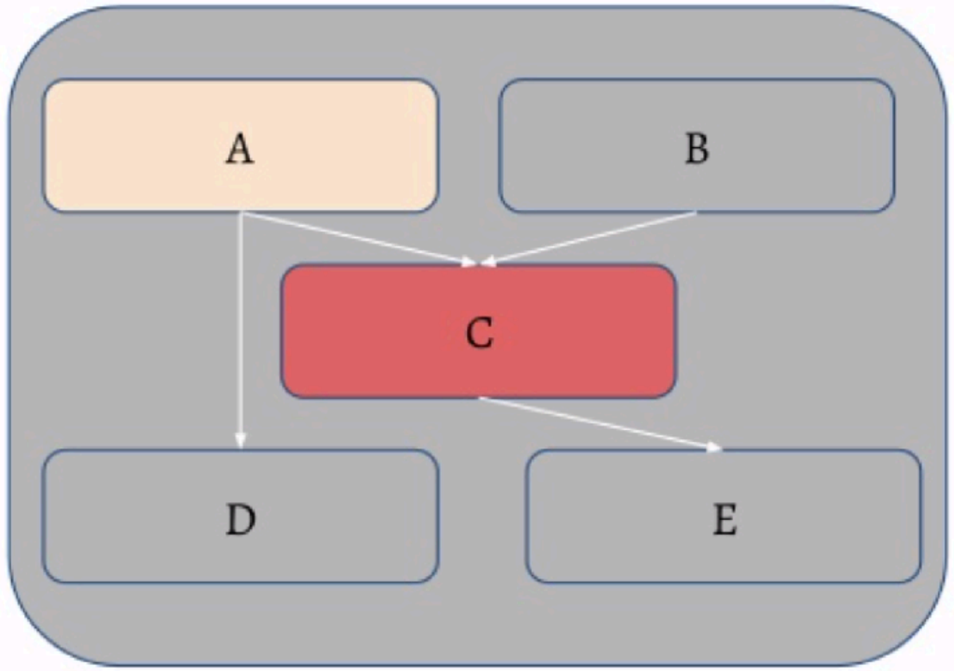


Ideally

- 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

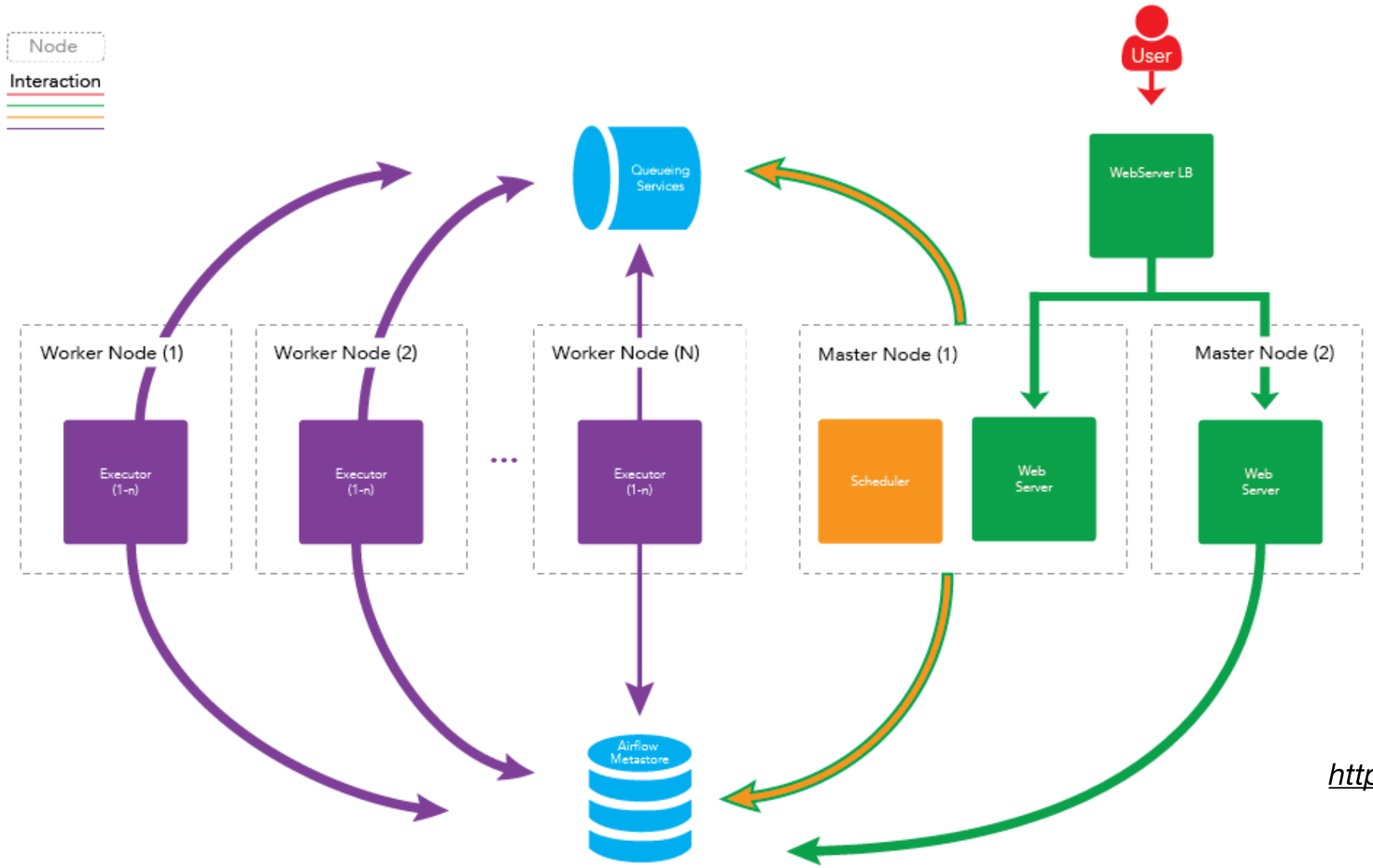



```
# Airflow
dag = DAG(schedule_interval=
              timedelta(days=1),
            start_date=
              datetime(2015,10,6))

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)
```

https://www.slideshare.net/PyData/how-i-learned-to-time-travel-or-data-pipelining-and-scheduling-with-airflow-67650418

Distributing work

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



http://site.clairvoyantsoft.com/making-apache-airflow-highly-available/

Prefect

- Improvements over Airflow

```python
from prefect import task, Flow

@task
def say_hello():
    print("Hello, world!")


with Flow("My First Flow") as flow:
    say_hello()


flow.run() # "Hello, world!"
```

PREFECTCORE

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

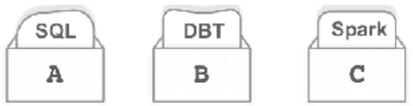
Monitor and Manage

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

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

January 18, 2014

hadoop 26 minutes

in parallel

```
cat *.pgn | grep "Result" | sort | uniq -c
```

70 seconds

in parallel

18 seconds

```
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 }'
```

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

“All-in-one”

Amazon SageMaker

gradient by Paperspace

DOMINO DATA LAB

Data

Versioning: Pachyderm, DVC, LIQUIDATA

Labeling: Label Studio, scale, prodigy

Processing: dbt, DAGSTER

Exploration: pandas, dask, RAPIDS

Datasets

Sources: databricks, S3, Parquet, snowflake

Development

Frameworks & Distributed Training: PYTORCH, fast.ai, DeepSpeed, FairScale

Experiment and Model Management: comet, Weights & Biases, TensorBoard, mlflow, neptune.ai

Resource Management: Determined AI, slurm

Software Engineering: jupyter, Streamlit, git

Compute: CoreWeave, Lambda

Deployment

Feature Store: tecton

Monitoring: fiddler

Edge: OctoML, NVIDIA TensorRT, TensorFlow Lite, ONNX

Web: ALGORITHMIA, SELDON

CI / Testing: Buildkite, great_expectations

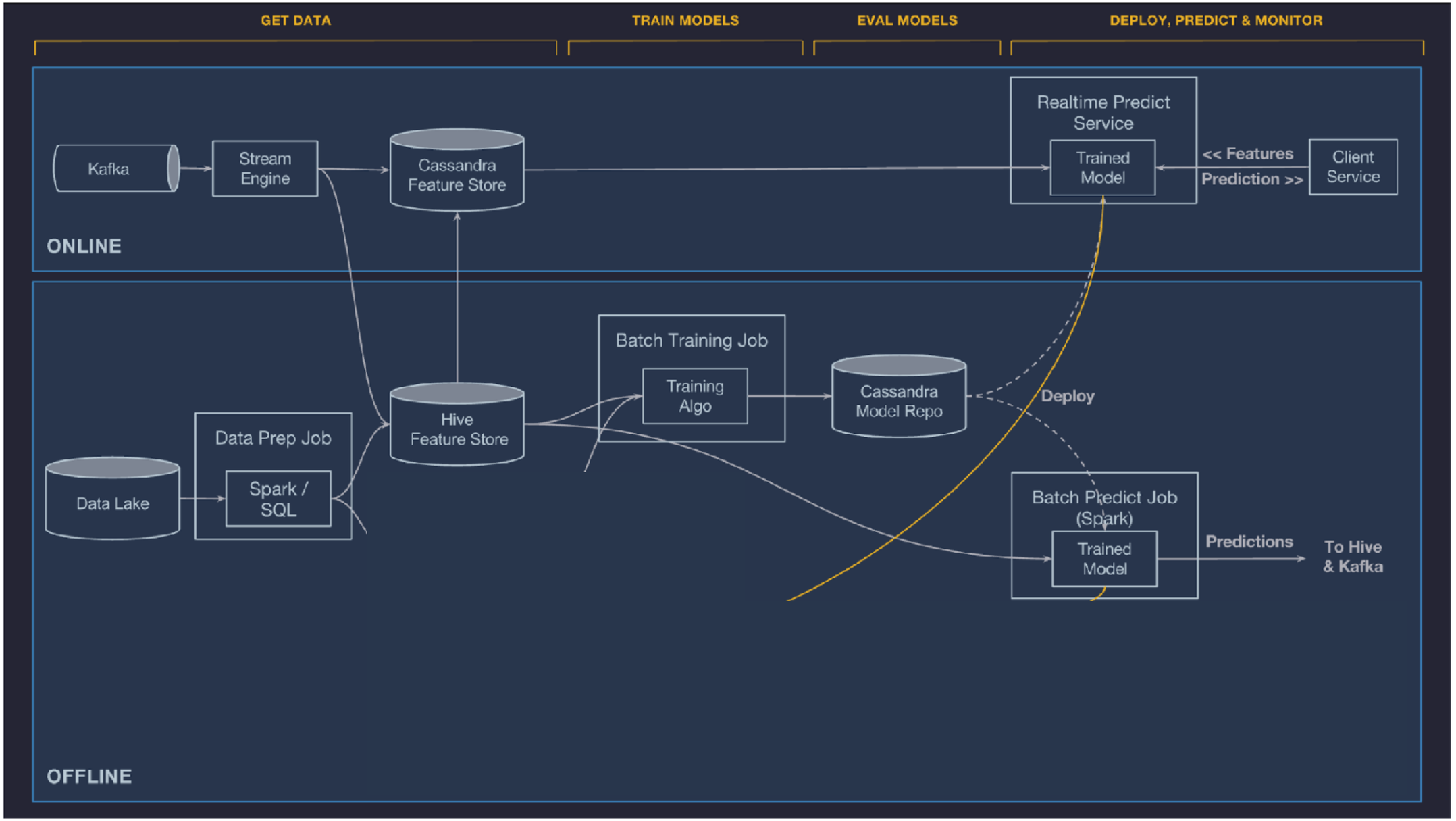
or or

Why feature stores?

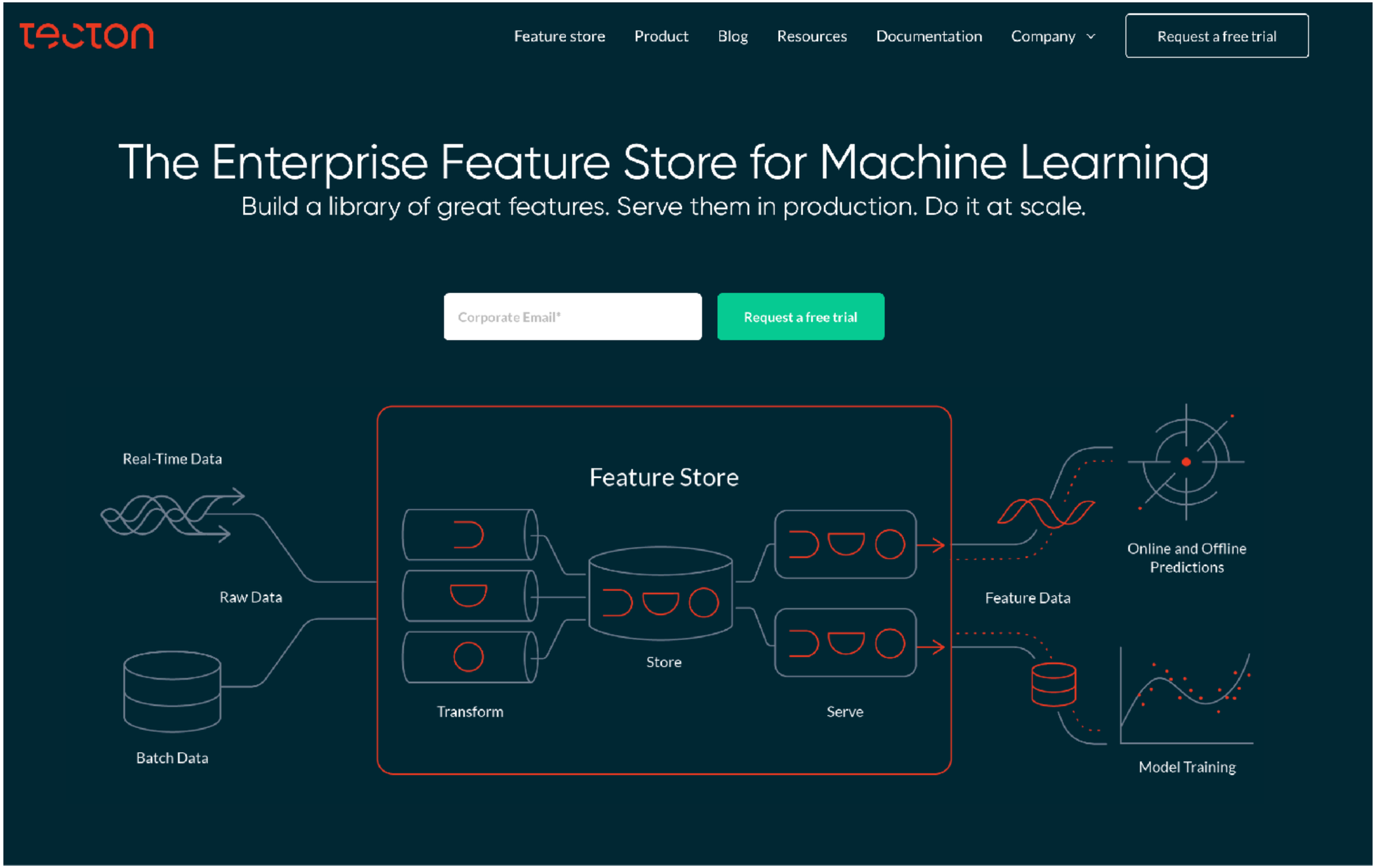
- All the data processing generates artifacts for training
- How do we
  - Make sure that in production, the same processing takes place?
  - Avoid recomputation when we retrain?
- Feature stores are a solution (that you may not need!)

Meet Michelangelo: Uber's Machine Learning Platform

Jeremy Hermann and Mike Del Balso September 5, 2017



https://eng.uber.com/michelangelo-machine-learning-platform/



https://www.tecton.ai

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

```python
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.

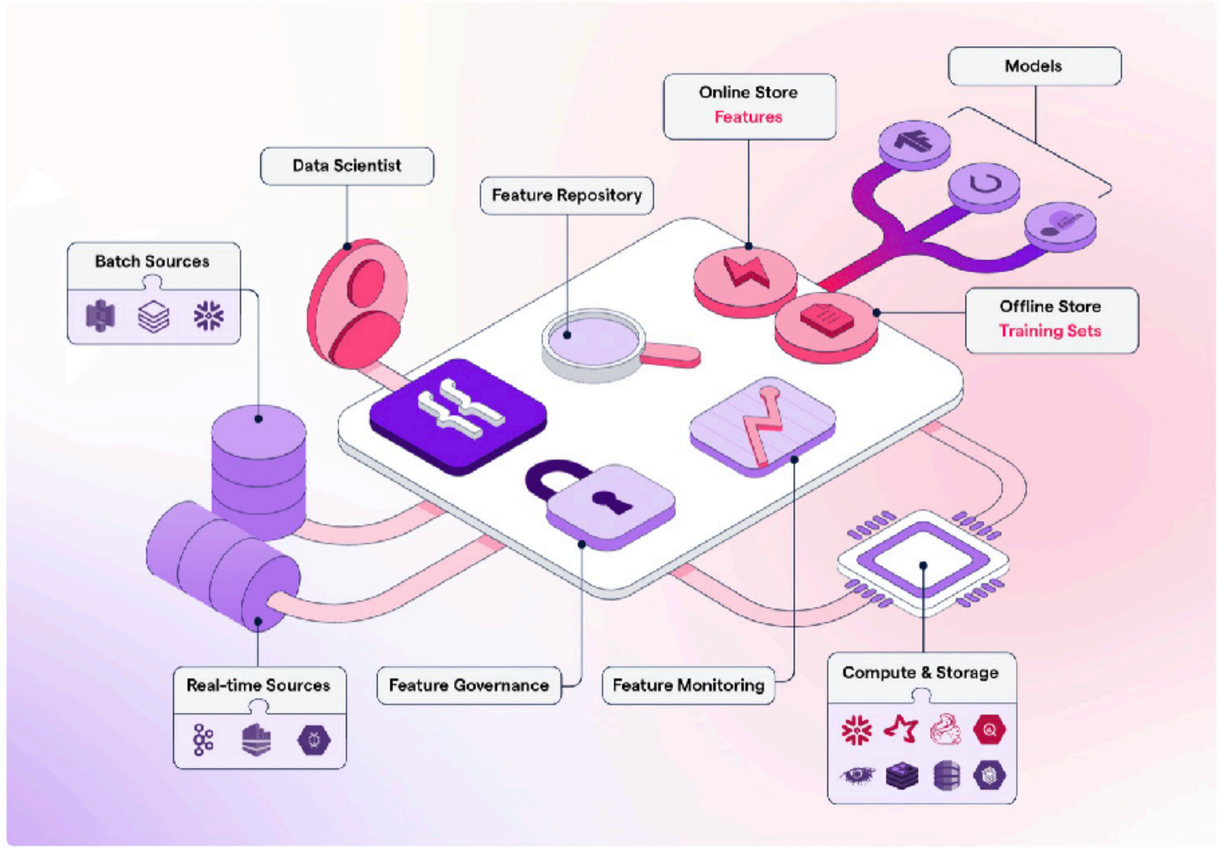
Serve

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



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 by Paperspace

DOMINO DATA LAB

Data

Versioning
Pachyderm, DVC, LIQUIDATA

Labeling
Label Studio, scale, prodigy

Processing
dbt, DAGSTER

Exploration
pandas, dask, RAPIDS

Datasets

Sources
S3, databricks, Parquet, snowflake

Development

Frameworks & Distributed Training
PYTORCH, fast.ai, DeepSpeed, FairScale

Experiment and Model Management
comet, Weights & Biases, TensorBoard, mlflow, neptune.ai

Resource Management
Determined AI, slurm

Software Engineering
jupyter, Streamlit, git

Compute
CoreWeave, Lambda

Deployment

Feature Store
tecton

Monitoring
fiddler

Edge
OctoML, NVIDIA TensorRT, TensorFlowLite, ONNX

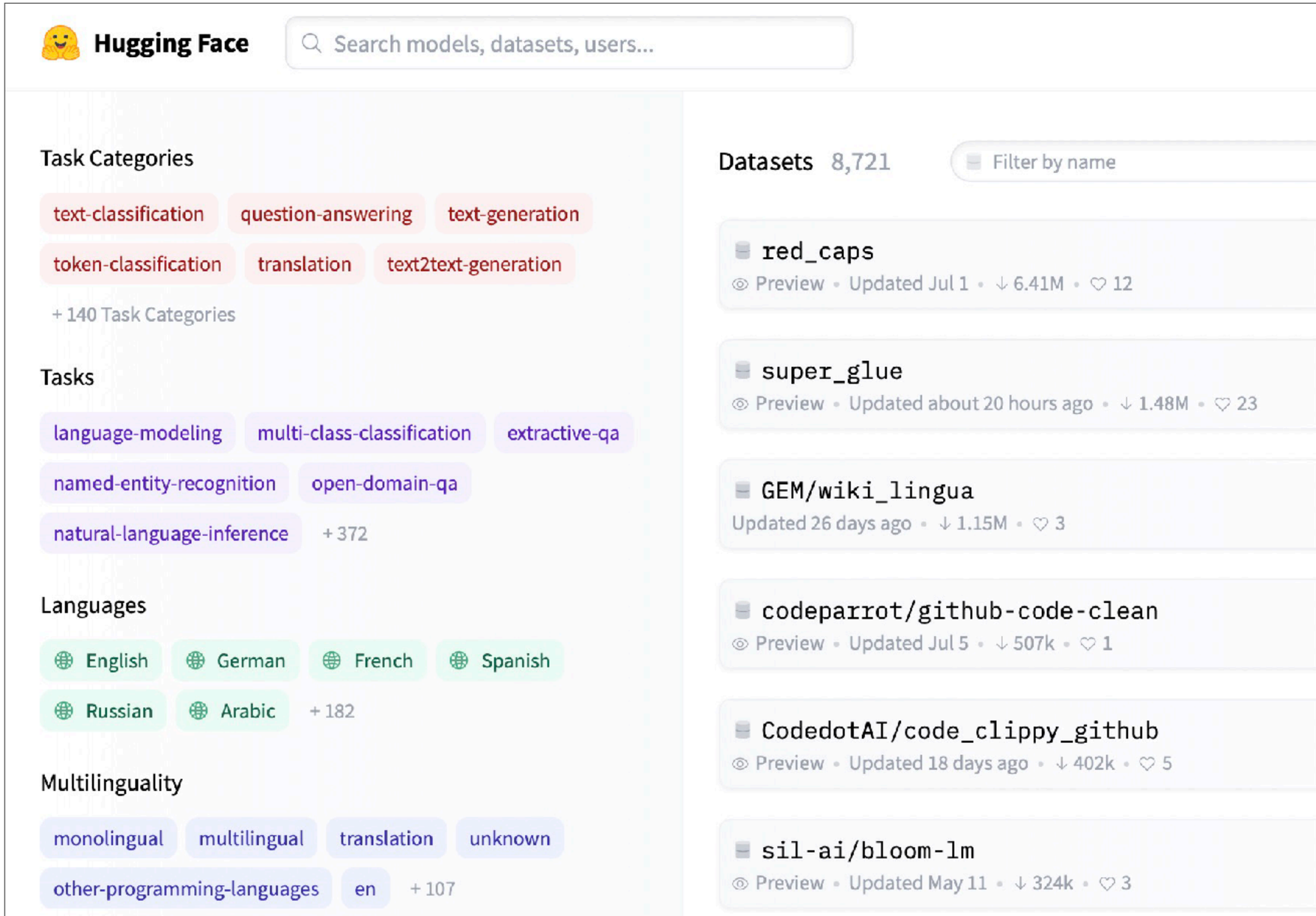
Web
ALGORITHMIA, SELDON

CI / Testing
Buildkite, great_expectations

or or

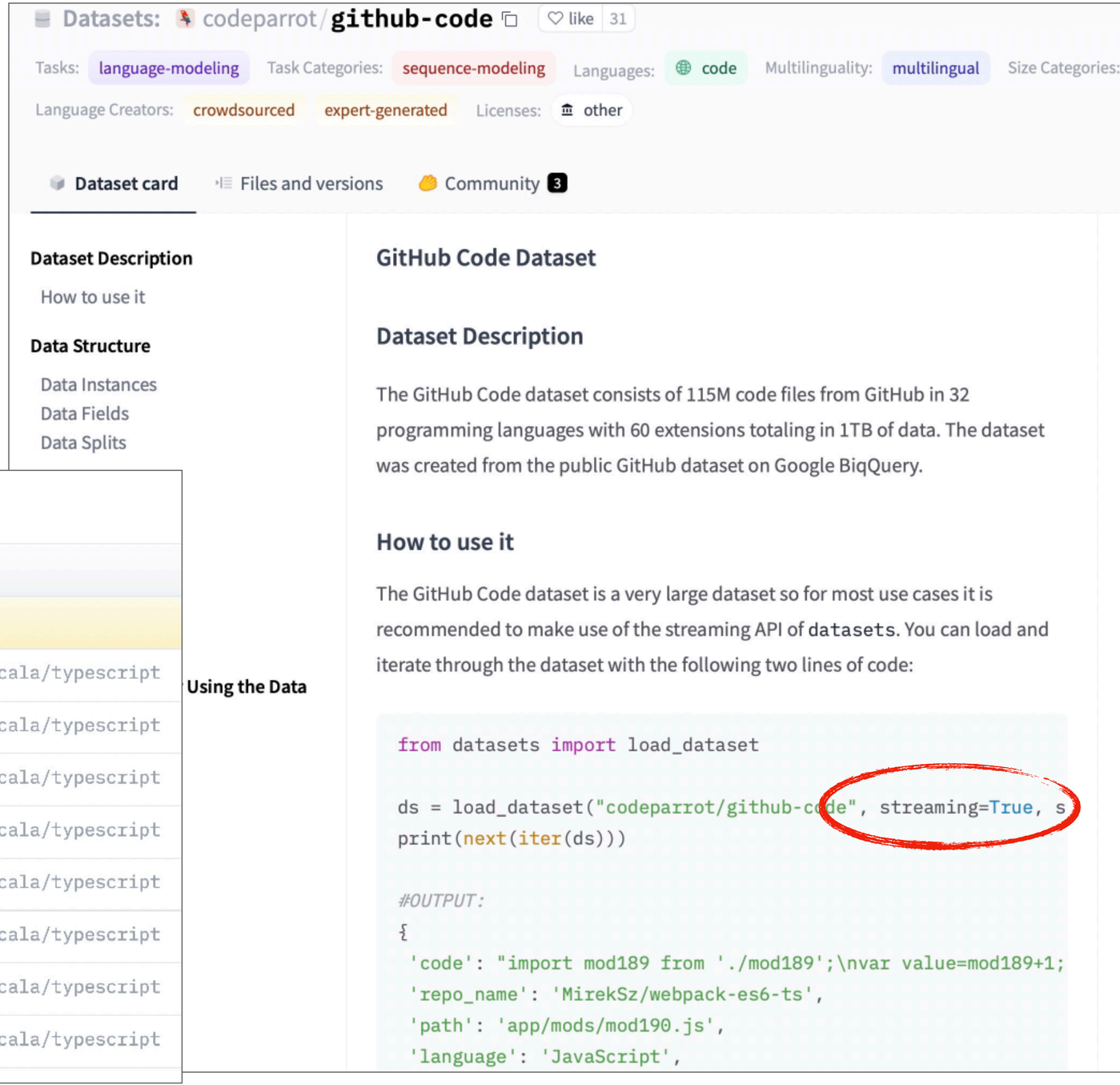
Huggingface Datasets

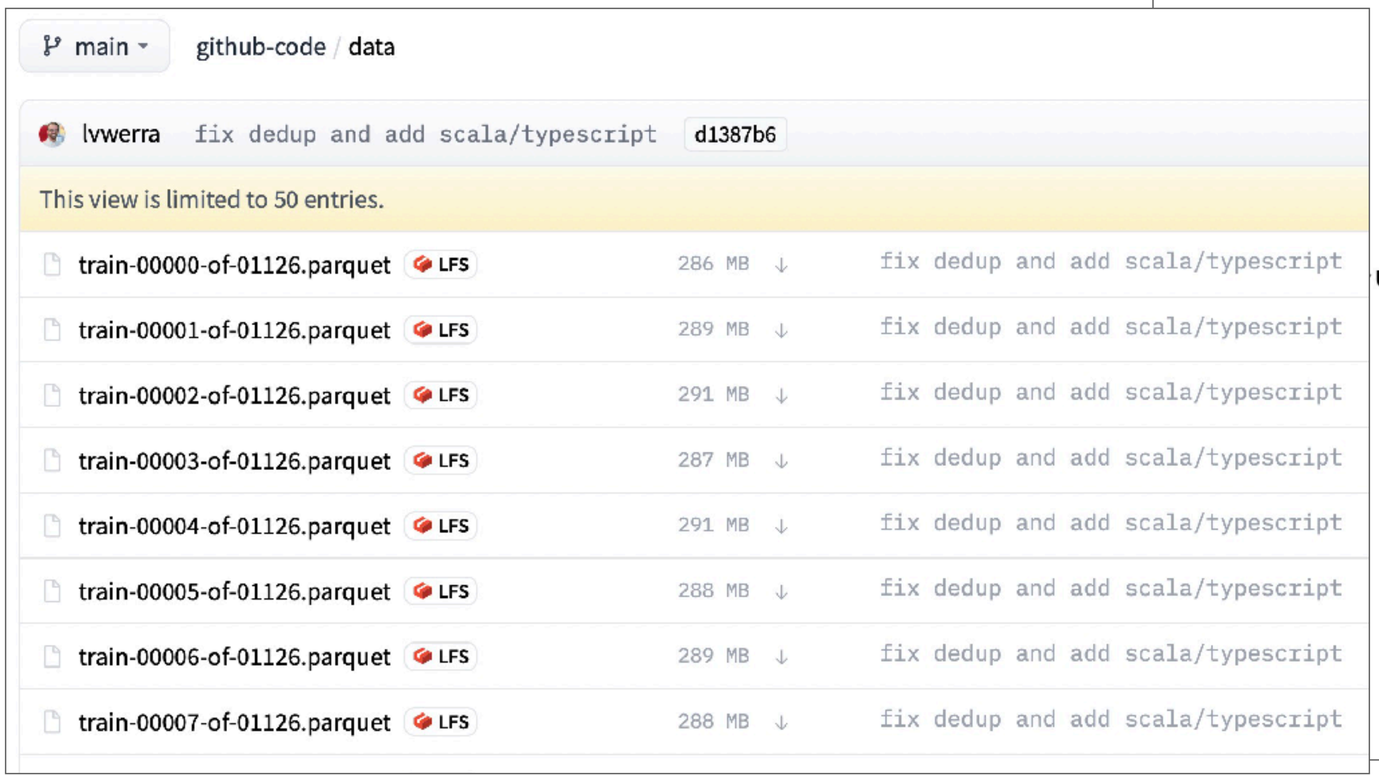
- Over 8K datasets for vision, NLP, etc



Example Dataset

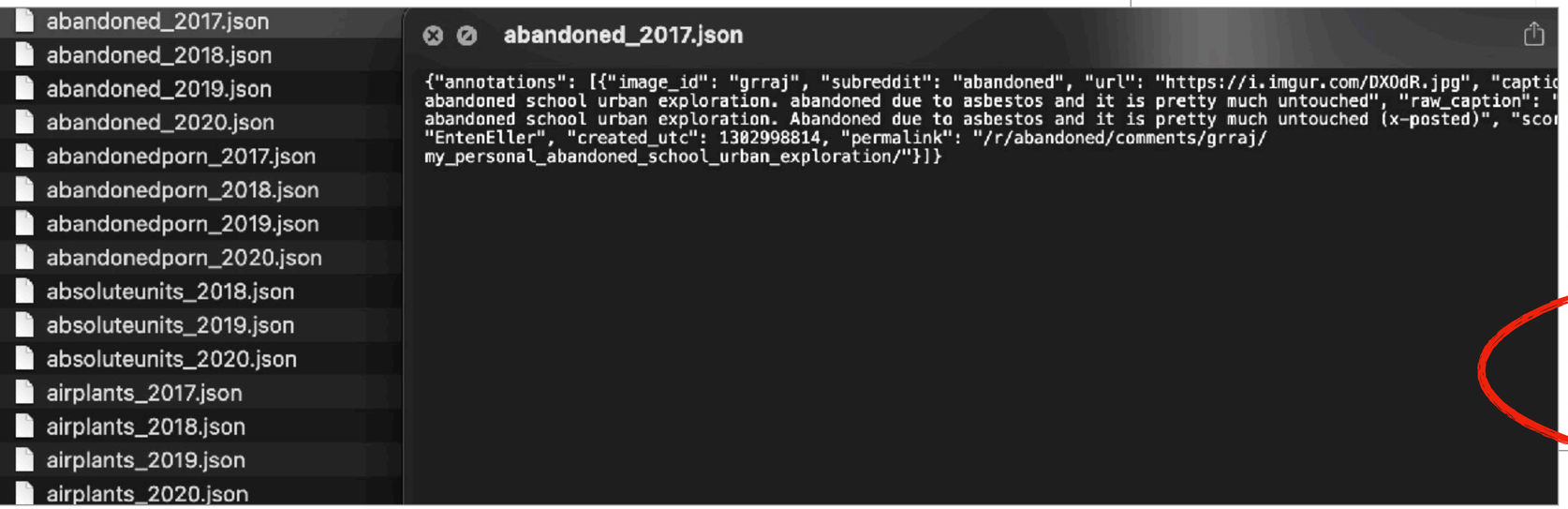
- Github-Code: >1TB of text
- Library allows you to stream it
- Underlying format: Parquet

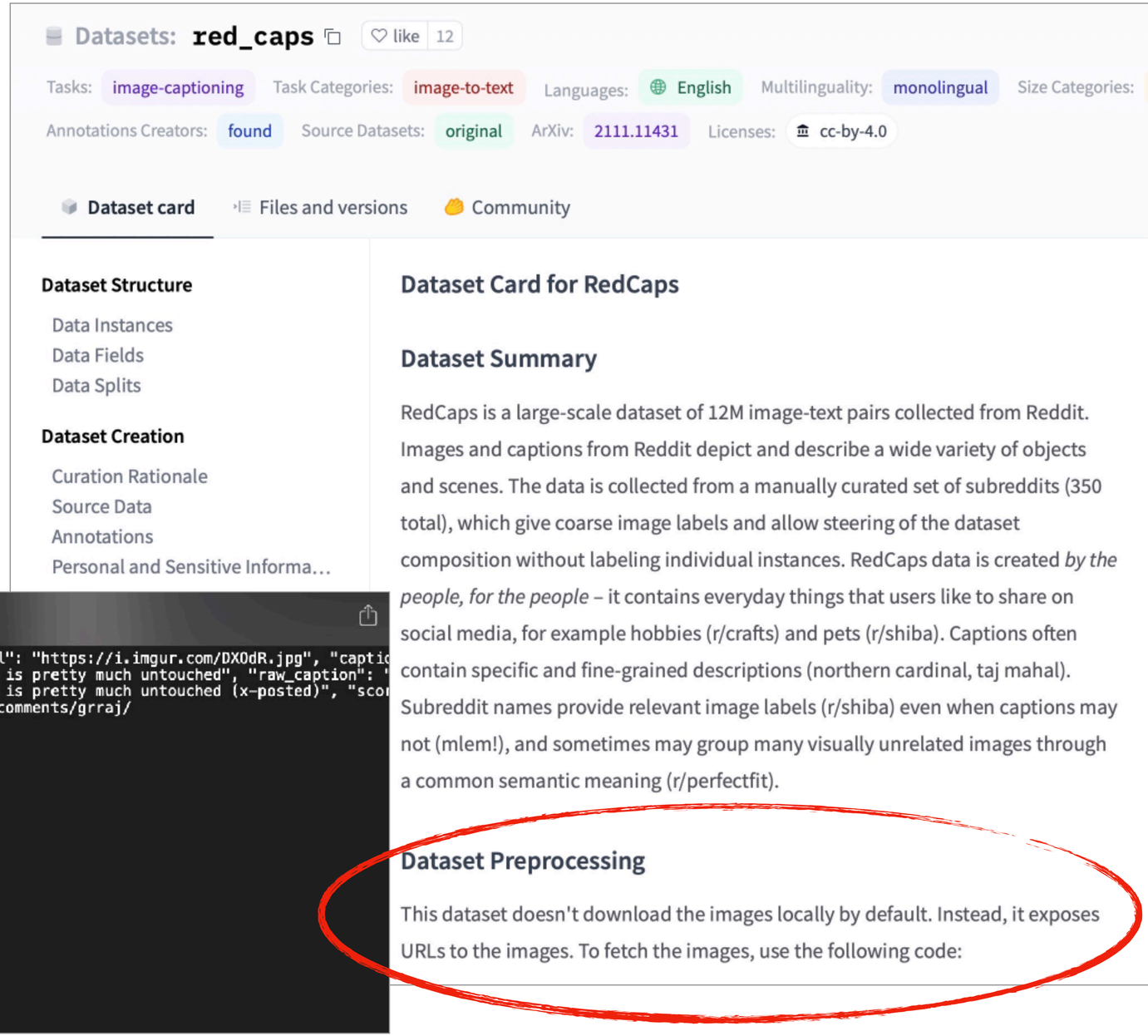




Example Dataset

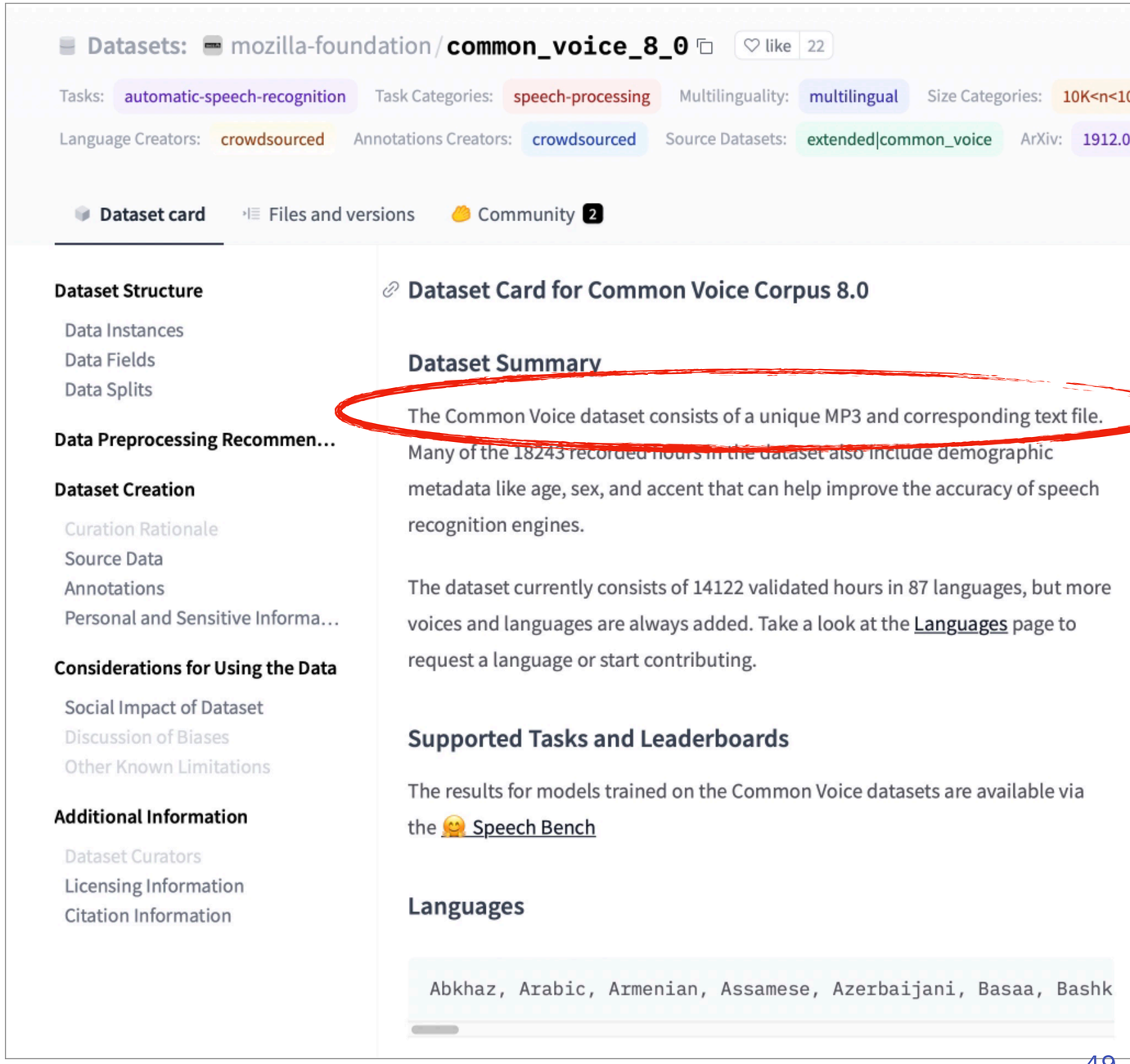
- RedCaps: 12M image-text pairs
- Need to download images yourself (multi-threaded!)
- Underlying format: images + JSON files





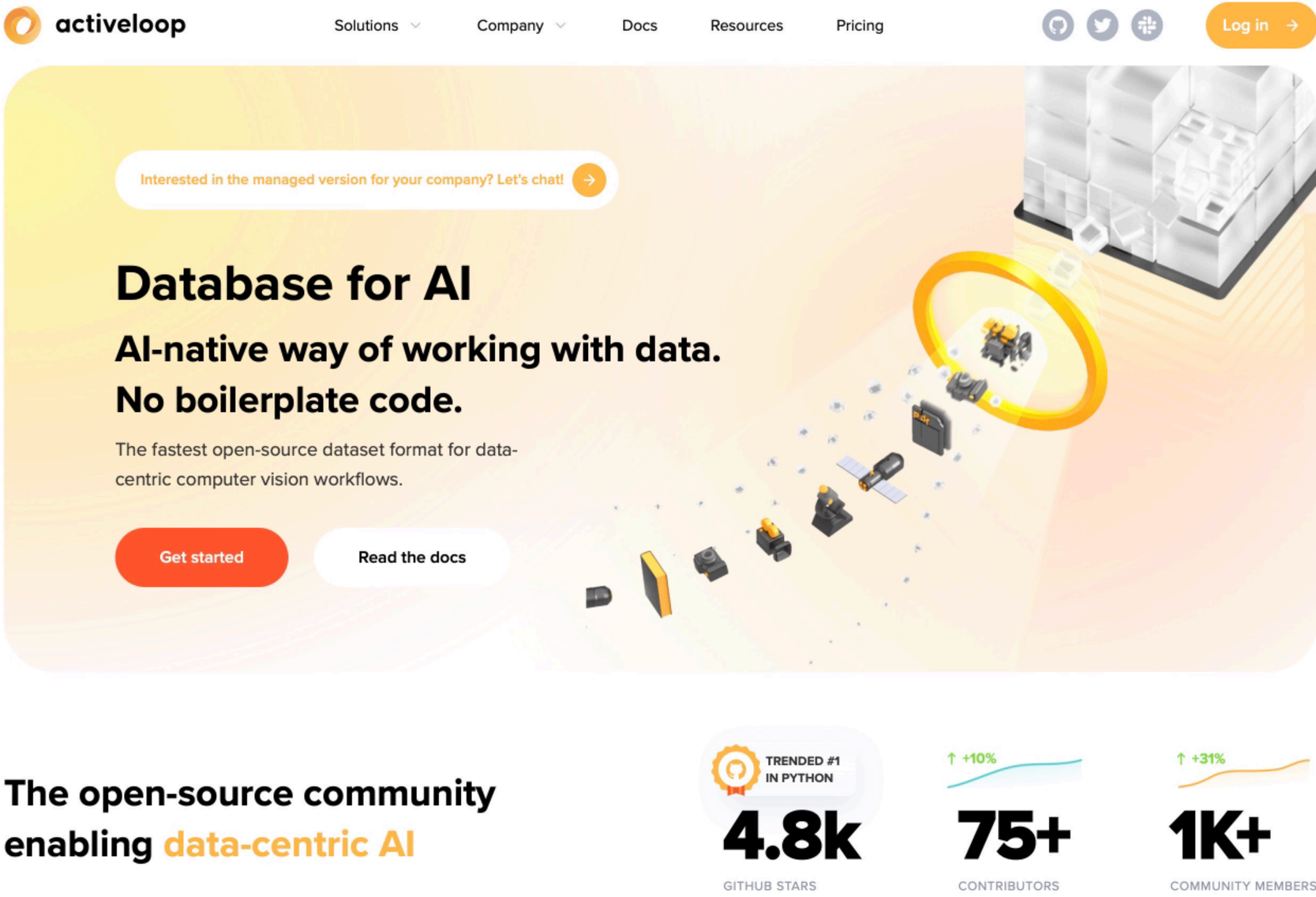
Example Dataset

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



Activeloop

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



DETAILS ANALYTICS

COCO Training

13

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

object detection | image | pose estimation | segmentation

Created by	Last updated:	Size of dataset:	Number of tensors:
activeloop	11/02/2021	Not Available	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 by Paperspace

DOMINO DATA LAB

Data

Versioning: Pachyderm, DVC, LIQUIDATA

Labeling: Label Studio, scale, prodigy

Processing: dbt, DAGSTER

Exploration: pandas, dask, RAPIDS

Datasets

Sources: databricks, S3, Parquet, snowflake

Development

Frameworks & Distributed Training: PYTORCH, fast.ai, DeepSpeed, FairScale

Experiment and Model Management: comet, Weights & Biases, TensorBoard, mlflow, neptune.ai

Resource Management: Determined AI, slurm

Software Engineering: jupyter, Streamlit, git

Compute: CoreWeave, Lambda

Deployment

Feature Store: tecton

Monitoring: fiddler

Edge: OctoML, NVIDIA TensorRT, TensorFlow Lite, ONNX

Web: ALGORITHMIA, SELDON

CI / Testing: Buildkite, great_expectations

or or

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.

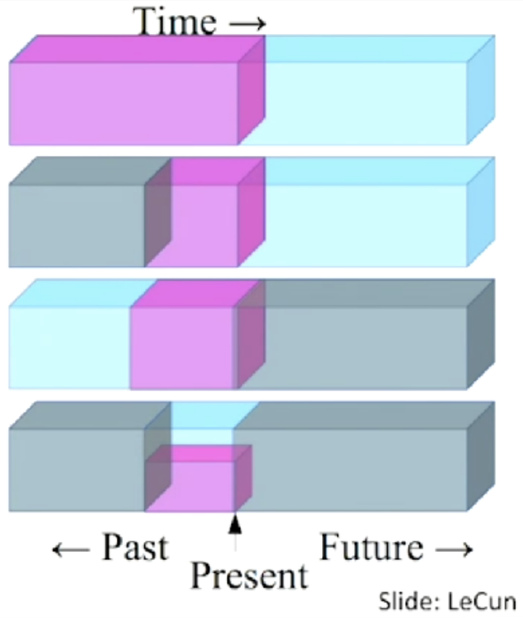


Fig. 1. A great summary of how self-supervised learning tasks can be constructed (Image source: LeCun's talk)

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

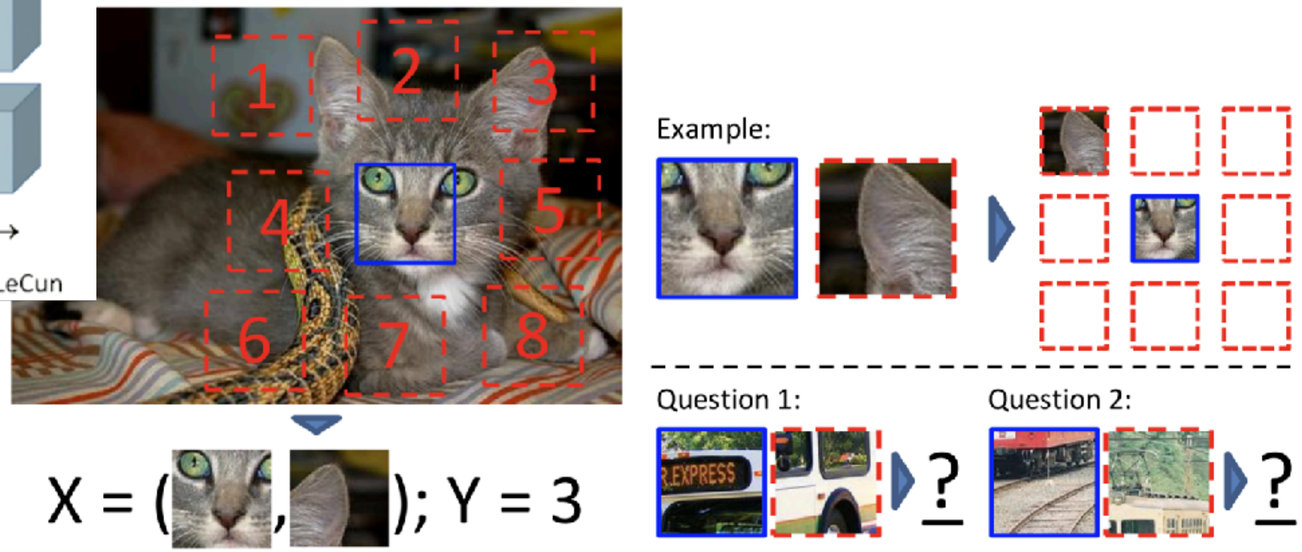


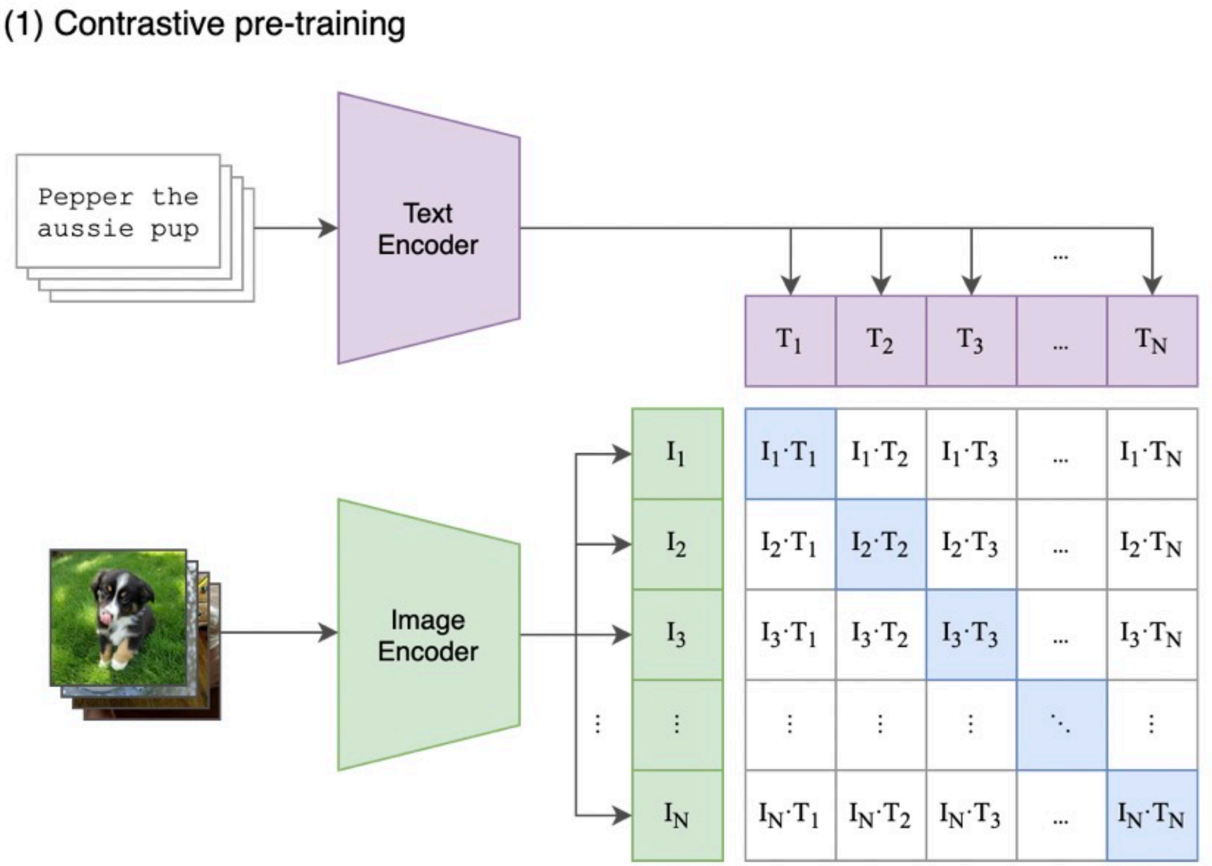
Fig. 4. Illustration of self-supervised learning by predicting the relative position of two random patches. (Image source: Doersch et al., 2015)

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

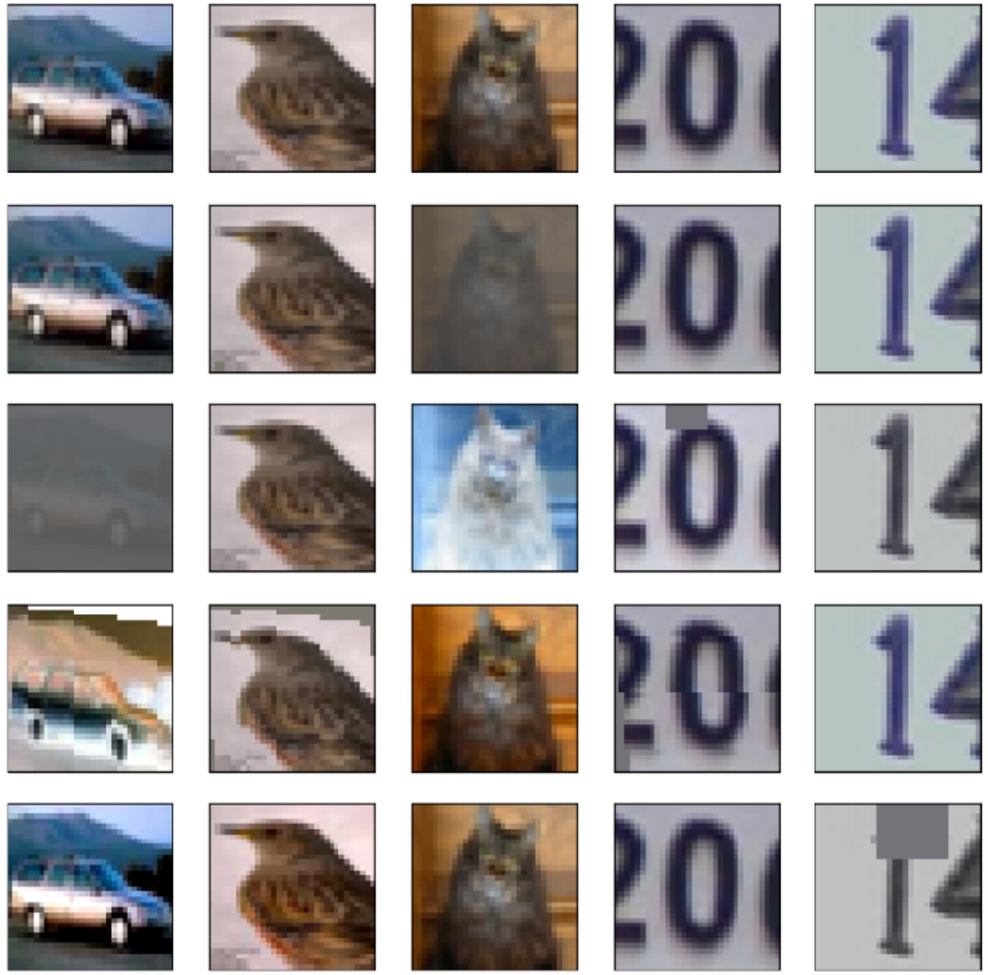
- Works across modalities, too
- Note the "contrastive" training:
  - Minimize distance between image and its text
  - Maximize distance between image and other texts



https://github.com/openai/CLIP

Image data augmentation

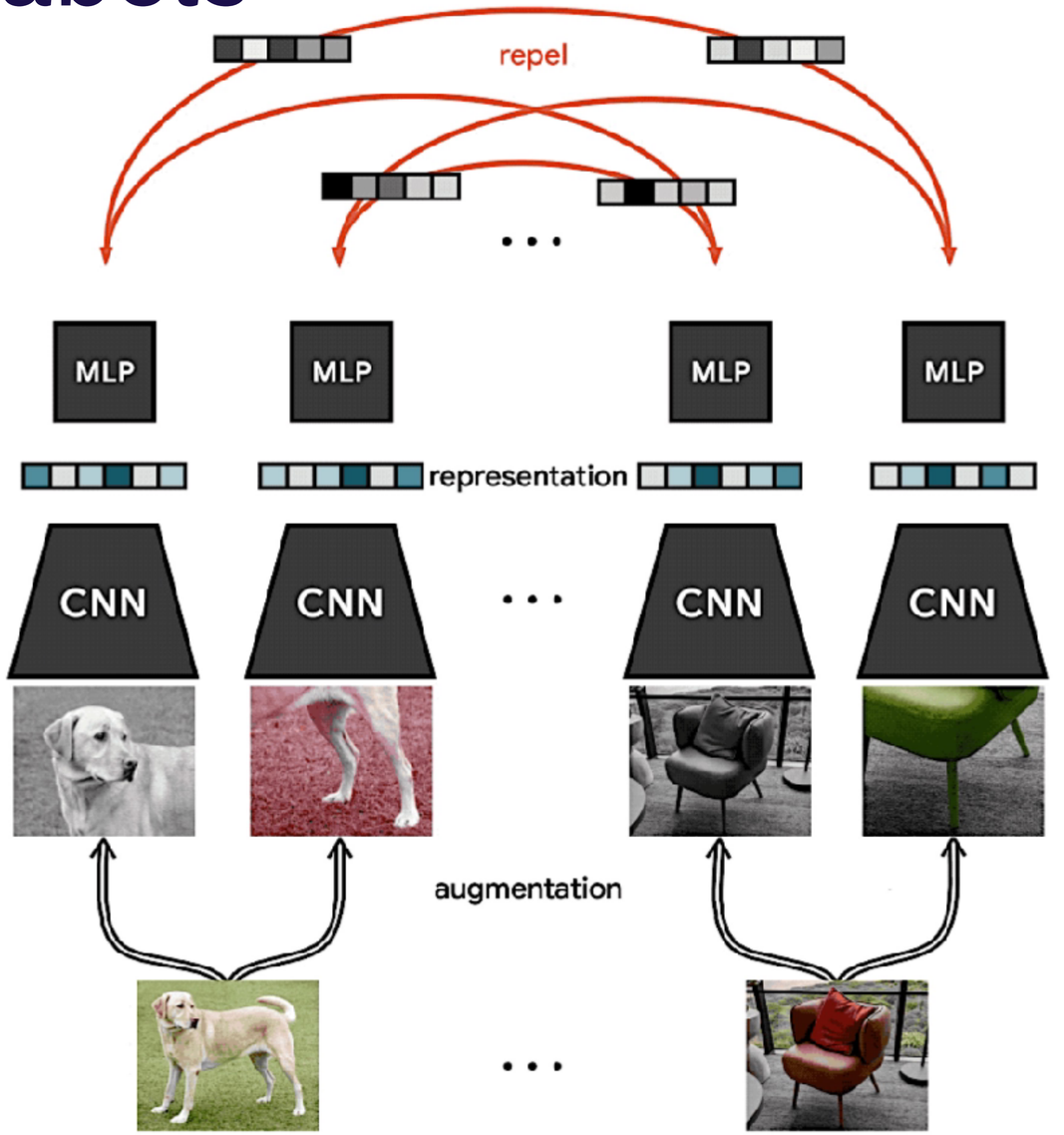
- **Must do** for training vision models
- Frameworks (e.g. torchvision) provide functions that do this
- Done in parallel to GPU training on the CPU



https://towardsdatascience.com/1000x-faster-data-augmentation-b91bafee896c

Augmentation can replace labels

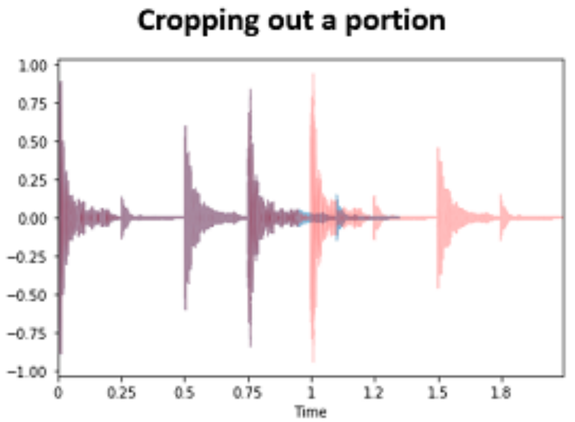
- SimCLR: learning objective is to
  - a) maximize agreement between augmented views of the same image
  - b) minimize agreement between different images

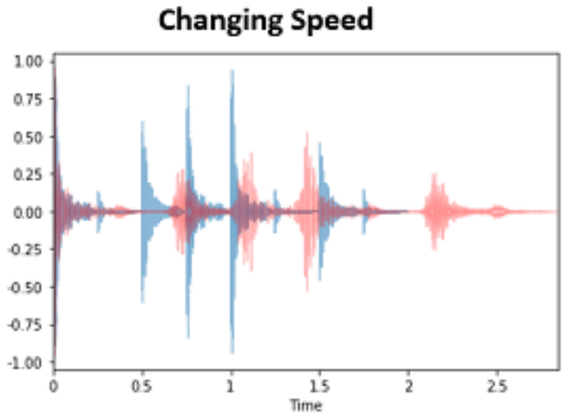


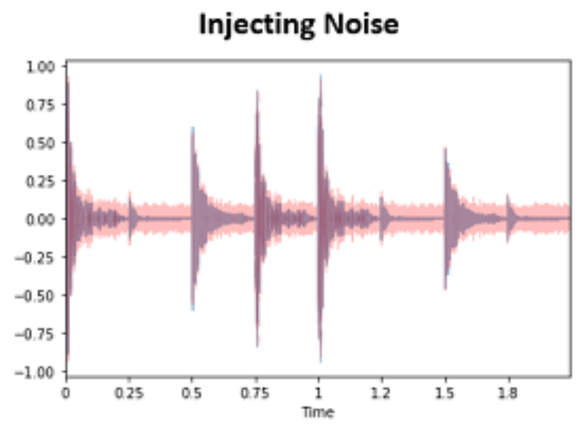
https://ai.googleblog.com/2020/04/advancing-self-supervised-and-semi.html

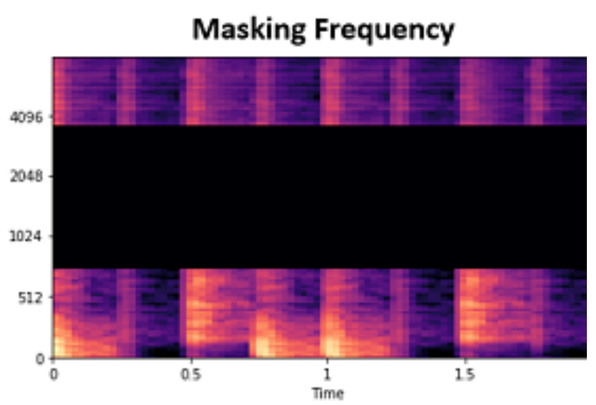
Other data augmentation

- 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





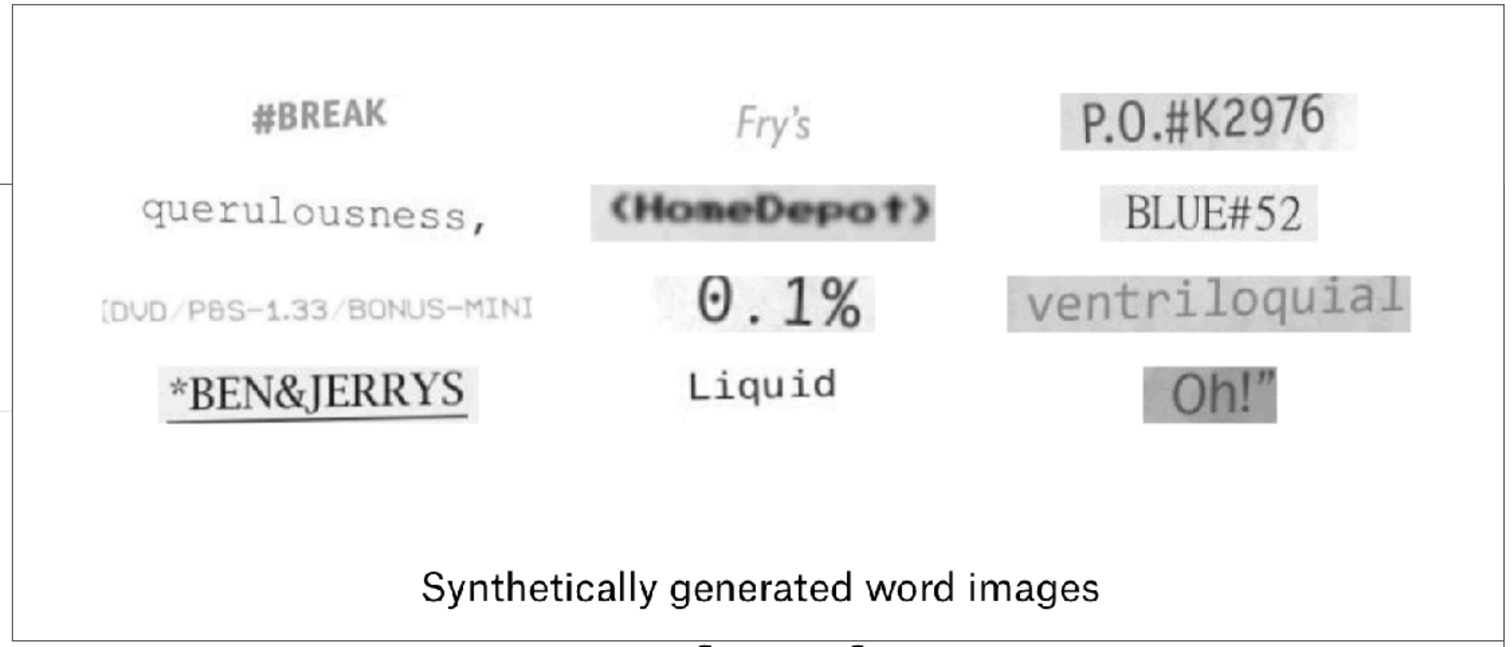




https://github.com/makcedward/nlpaug

Synthetic data

Underrated idea that is often worth starting with



Synthetically generated word images

Dropbox

Creating a Modern OCR Pipeline Using Computer Vision and Deep Learning

Brad Neuberg | April 12, 2017

728 0

https://blogs.dropbox.com/tech/2017/04/creating-a-modern-ocr-pipeline-using-computer-vision-and-deep-learning/

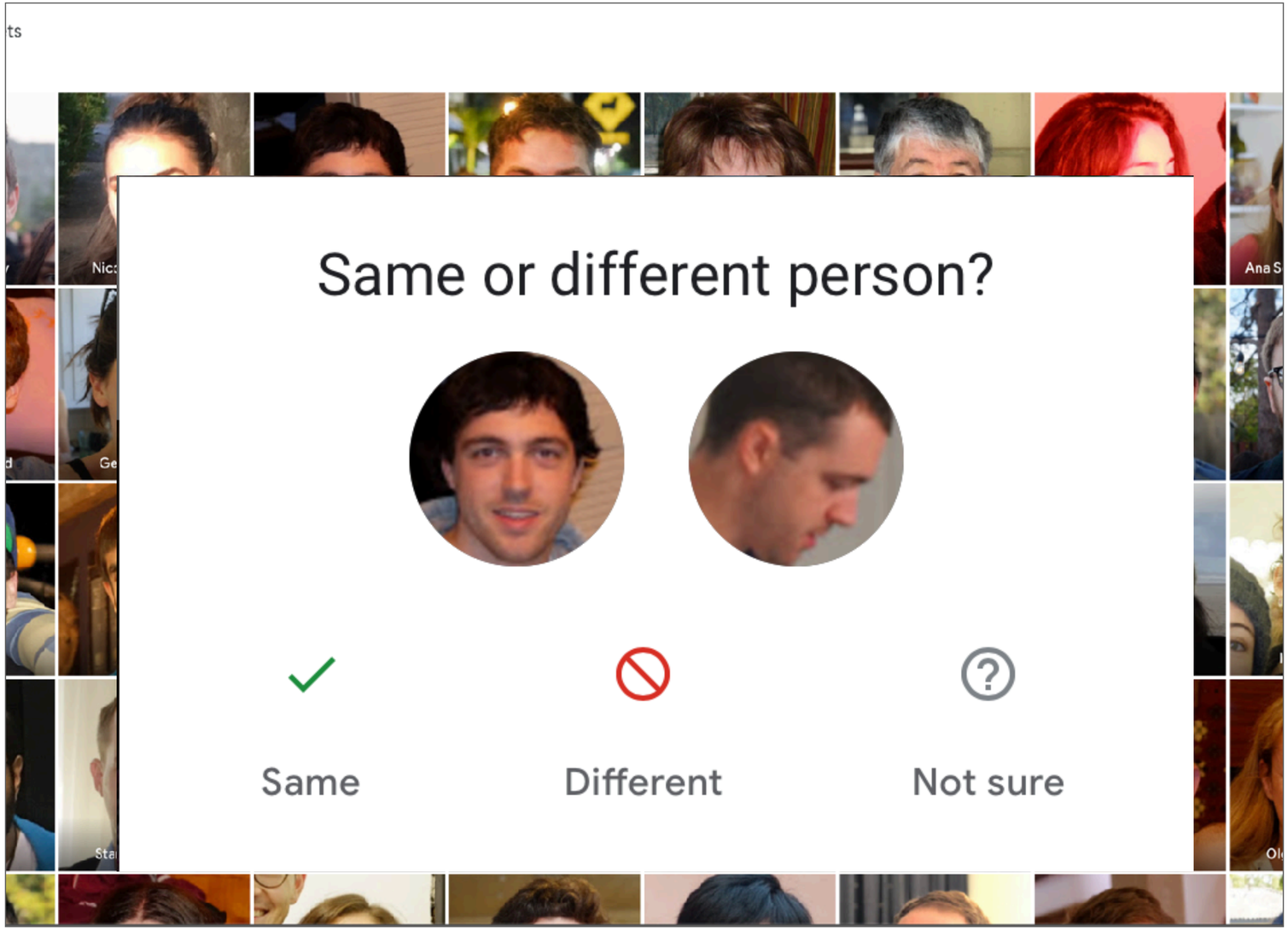
This can get pretty deep!

- 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

Andrew Moffat - https://github.com/amoffat/metabrite-receipt-tests

Ask your users to label data for you!

Enables rapid improvement with user labels



But usually: label data...

https://cdn-sv1.deepsense.ai/wp-content/uploads/2017/04/sample_image_from_the_training_set.jpg

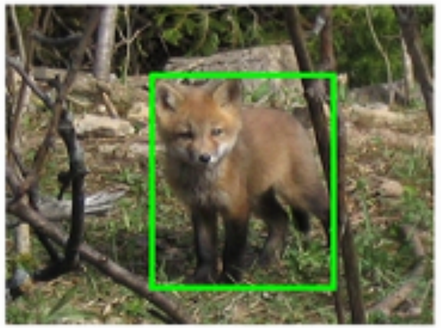
Standard set of features:

- bounding boxes, segmentations, keypoints, cuboids

- set of applicable classes

Training the annotators is crucial

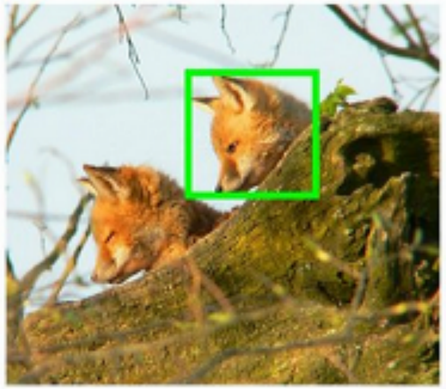
Rule 1: Include all visible part and draw as tightly as possible.



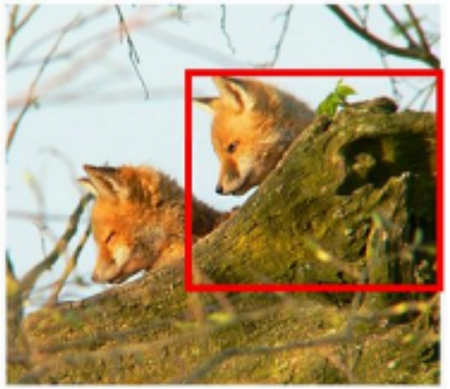
CORRECT

WRONG: must be as tight as possible!

WRONG: must include al parts!

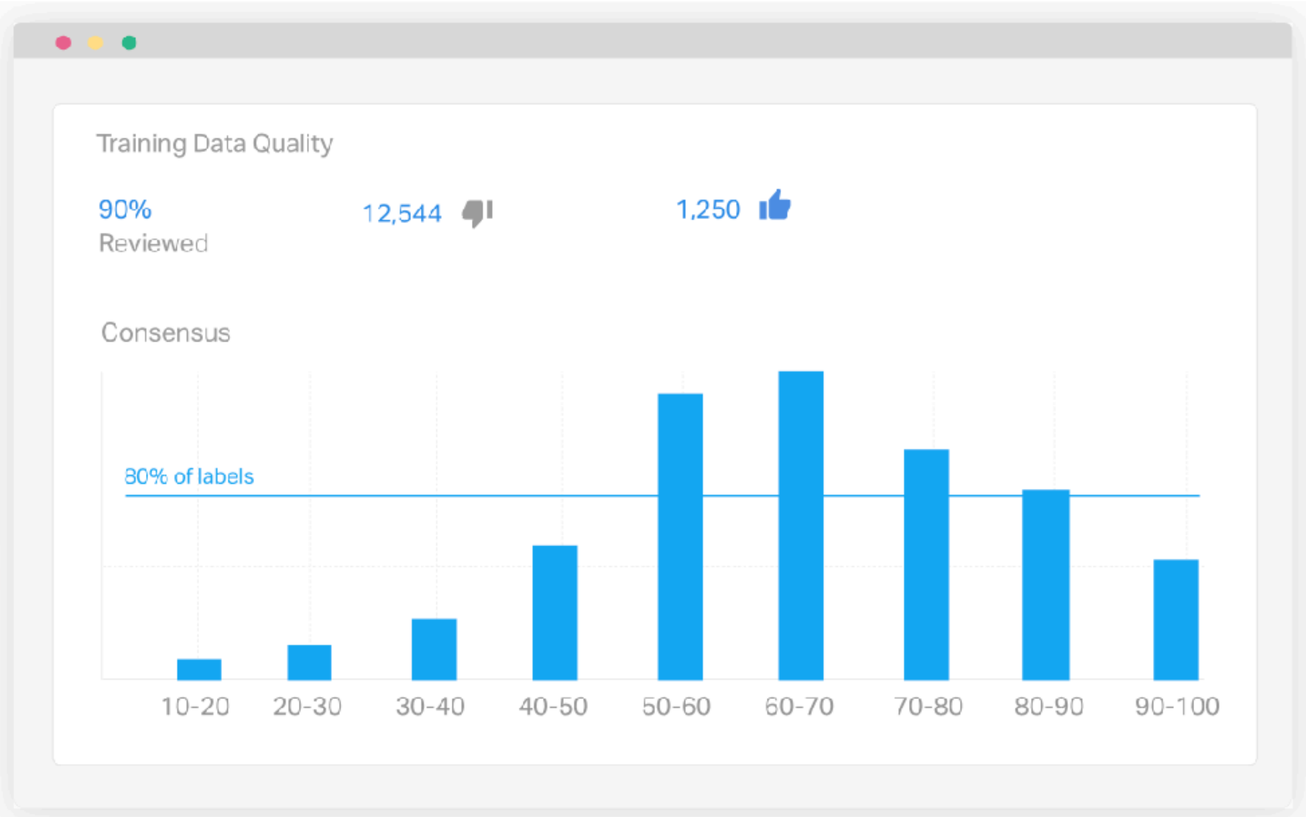


CORRECT



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

Source: cs.stanford.edu



Quality assurance is key

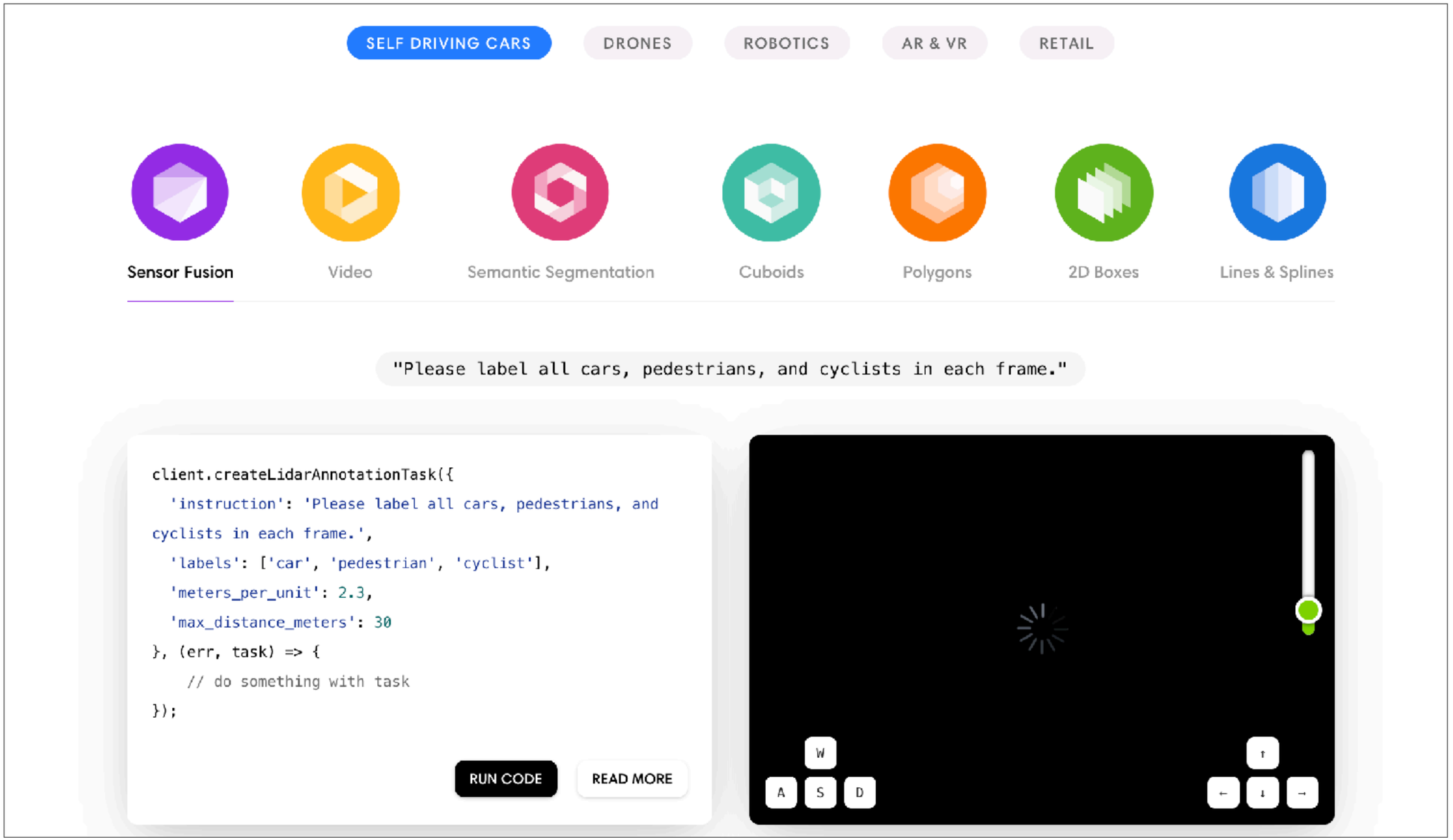
Sources of Labor

- 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



And there are many others

Labelbox

Product Solutions Company Pricing Support Sign in

NEW Introducing One-click Outsourcing

The best way to cr
manage training da

Labelbox is a collaborative traini
platform for artificial intelligence

CREATE ACCOUNT CONTACT SALES

CONDÉ NAST AIRBUS

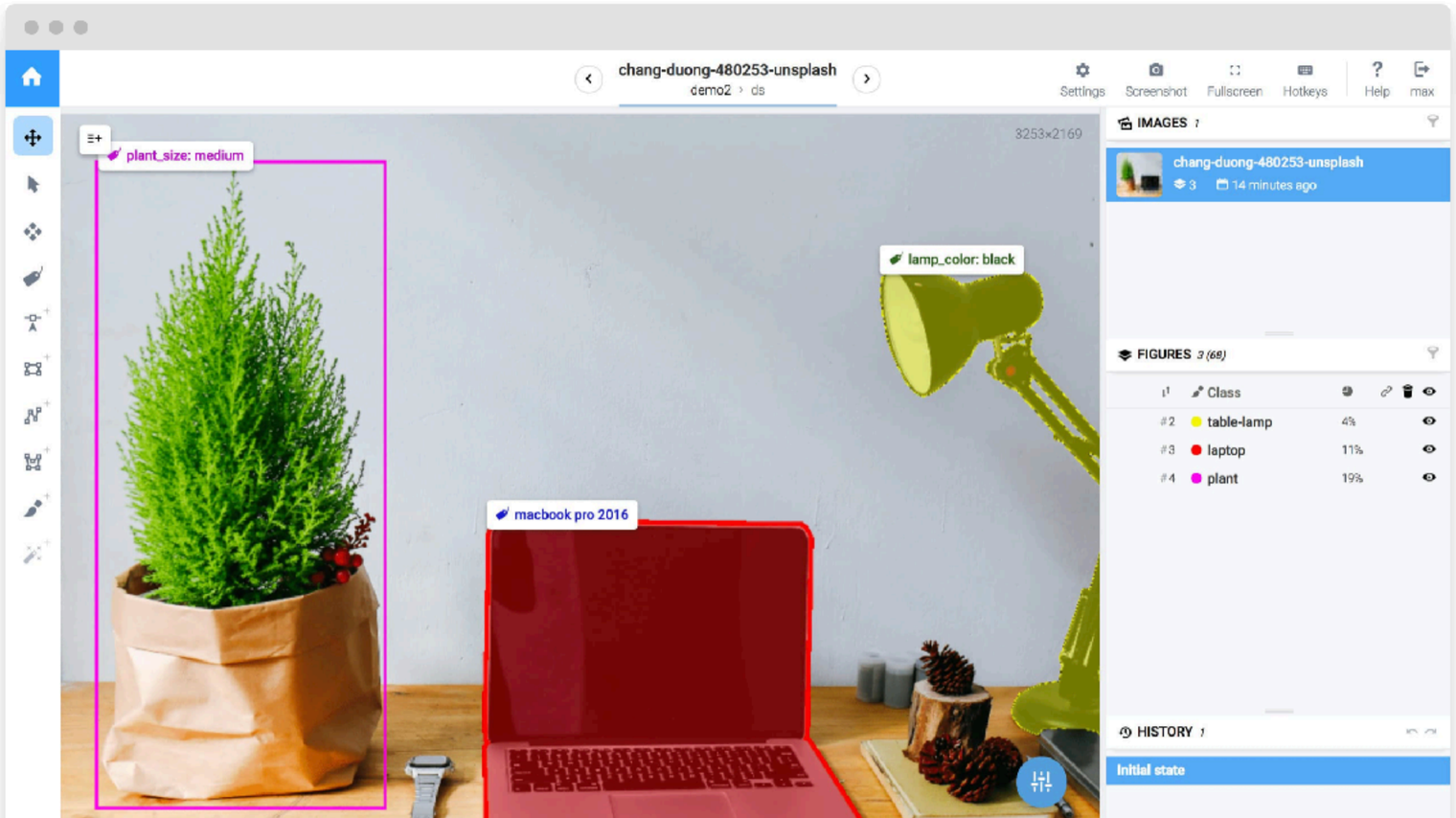
KEEP TRUCKIN STOCKWELL

SUPERVISELY

Product Enterprise Pricing About Learn More Docs Log in

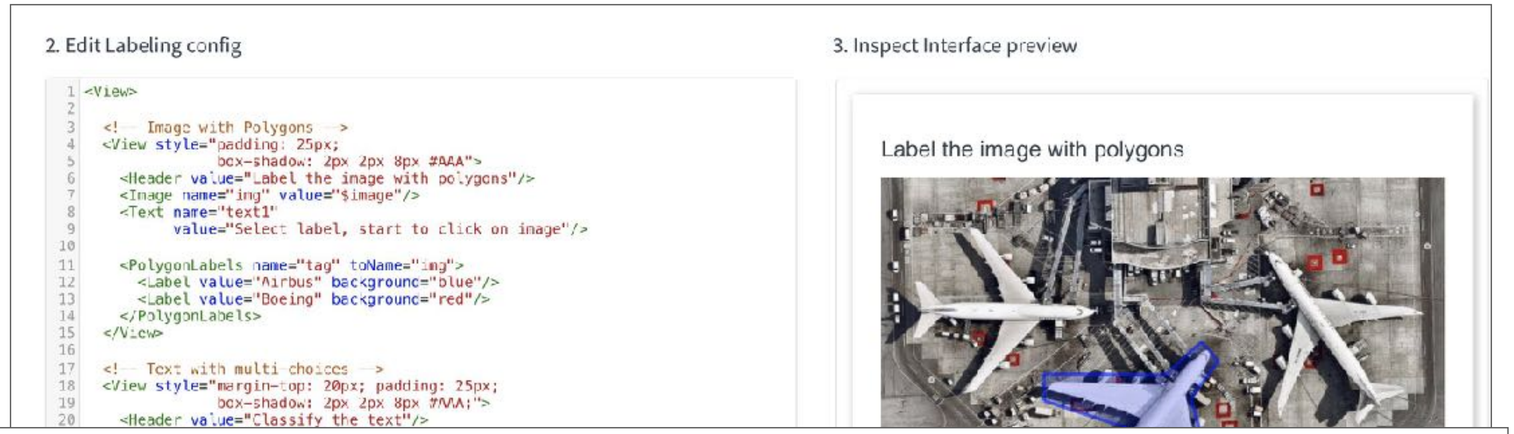
Leverage best in class annotation platform

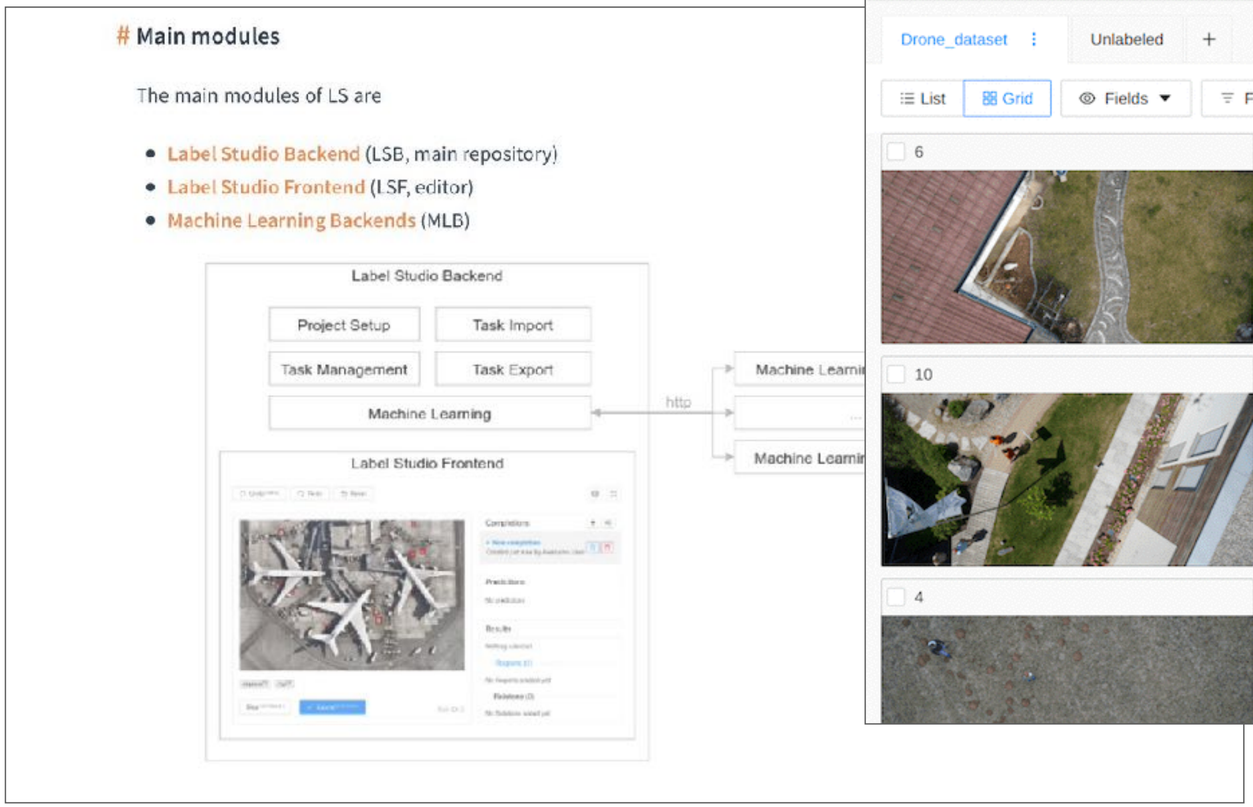
BOUNDING BOX POLYGON PIXELWISE TAG HOTKEYS LABELING HISTORY

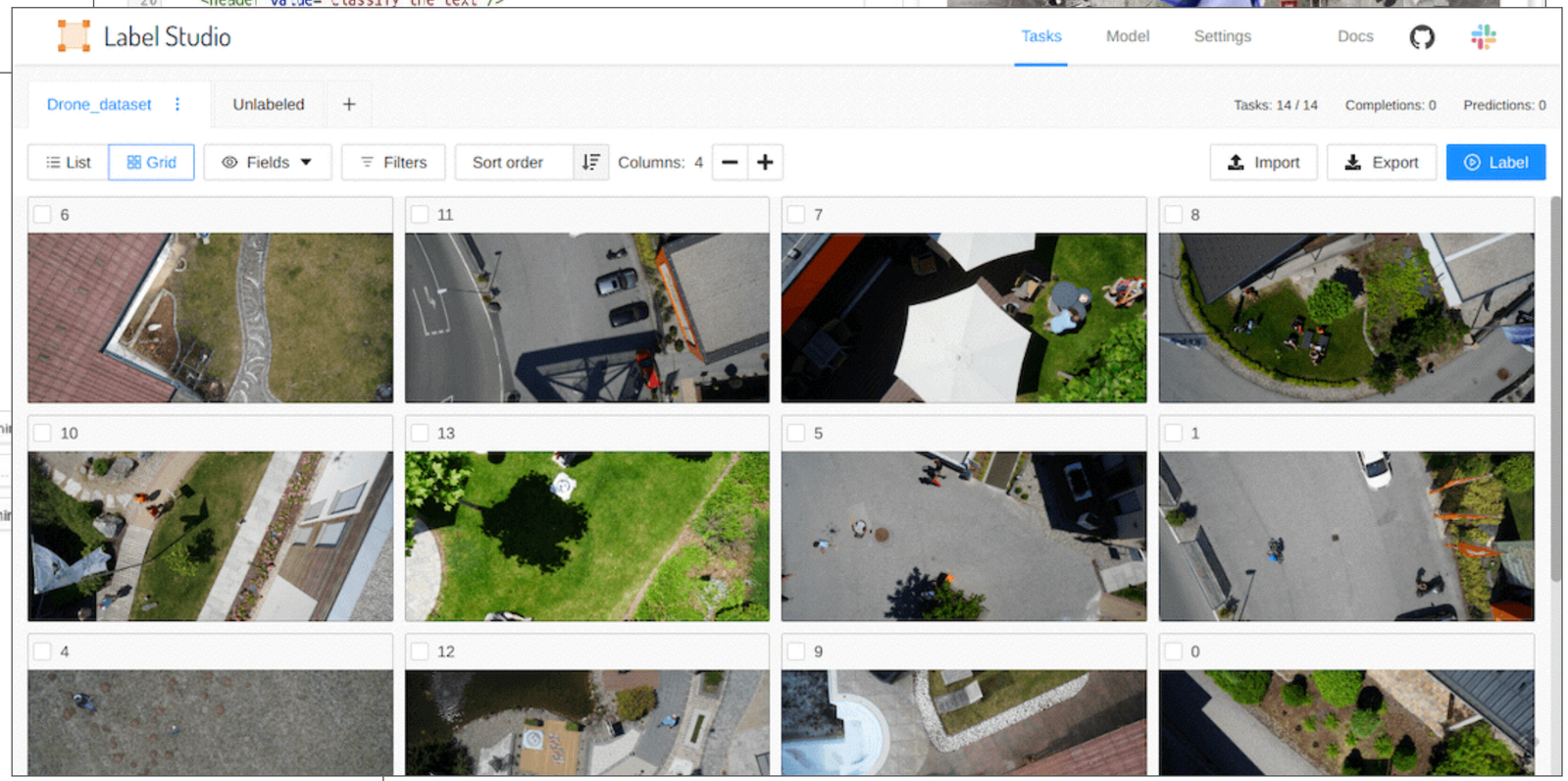


Label Studio

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







Diffgram

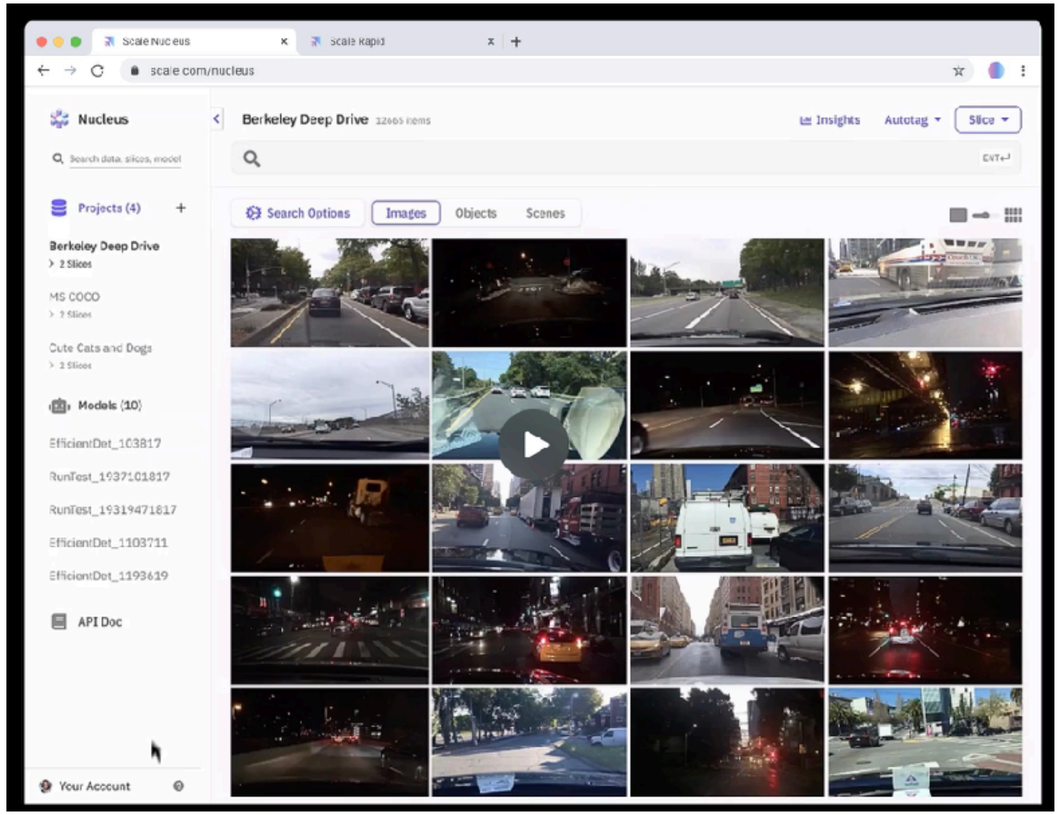
- Another open-source solution may be even better

Overall Diffgram has:

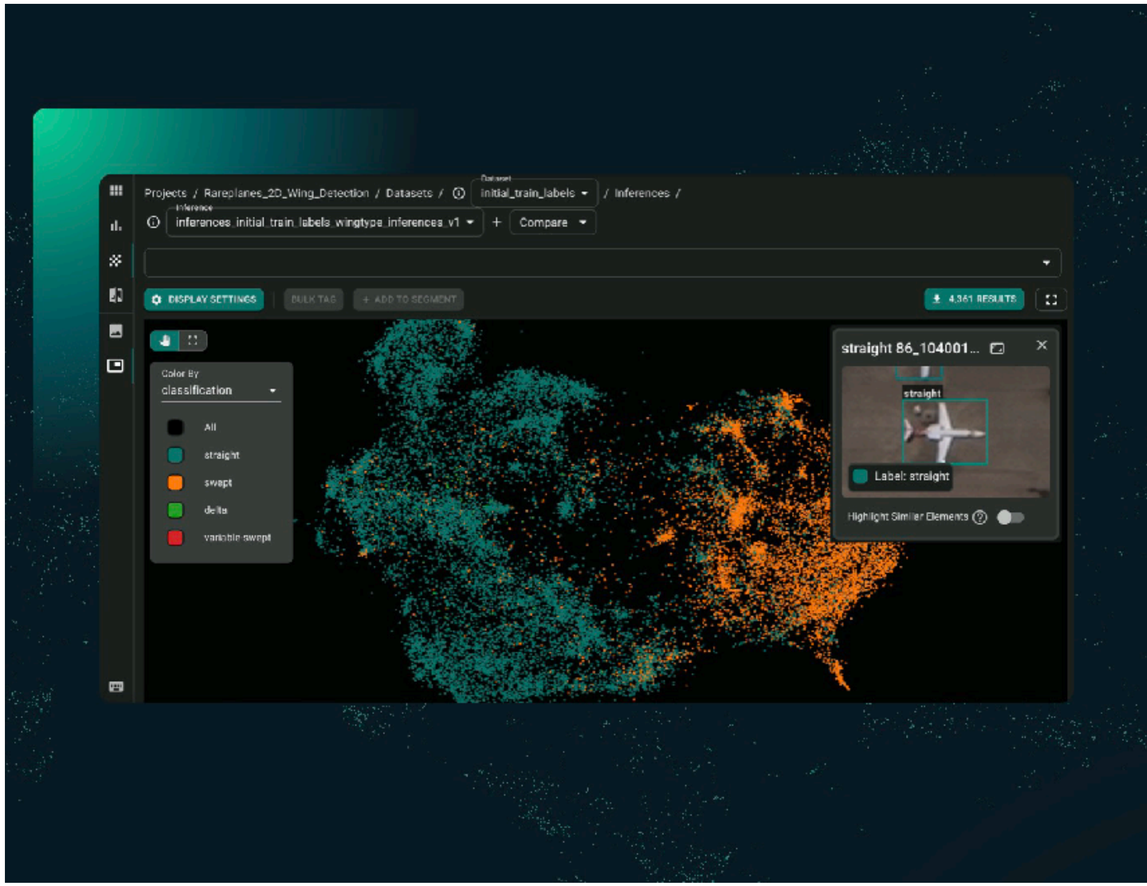
- Easier UI based customization
- More extensive automations
- Similar media type coverage
- Greater depth of image features
- Greater depth of video features
- Greater depth of text features
- 3D Support (Missing in Labelstudio)
- Feature Store – Ingest, Store, Query, Stream (Missing in Labelstudio)
- 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



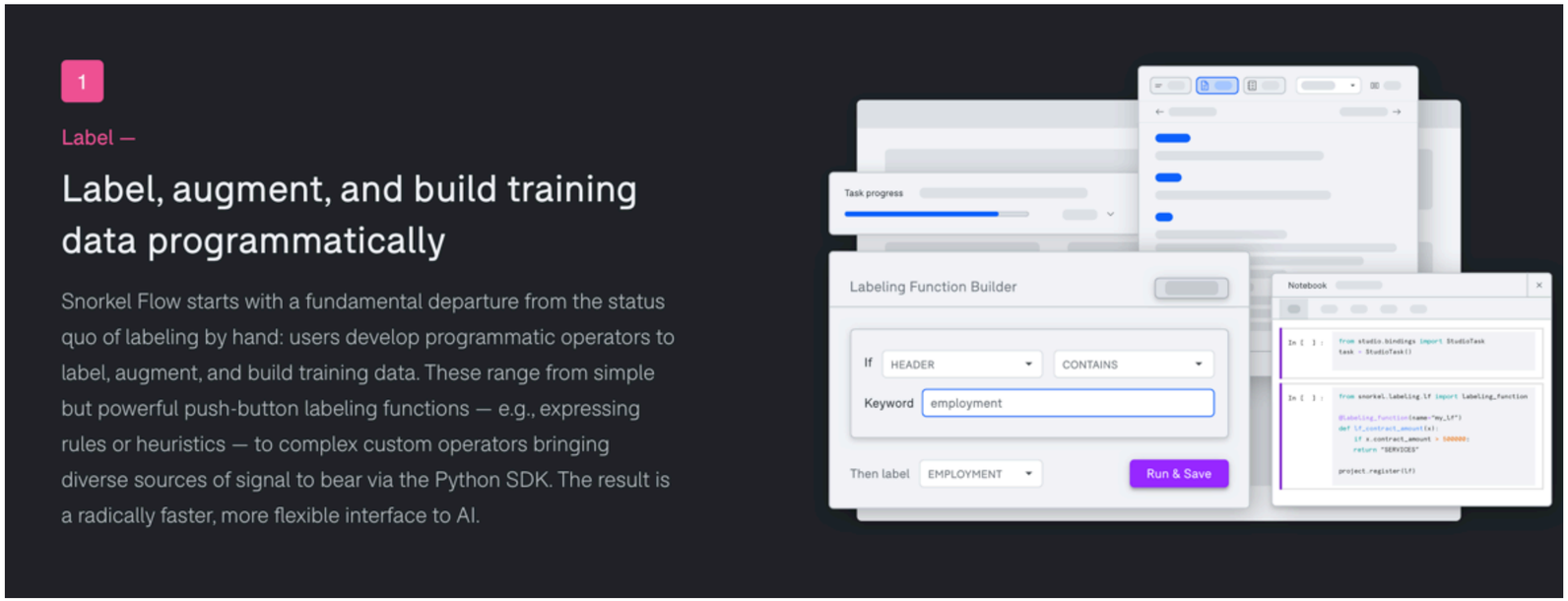
https://scale.com/nucleus



https://www.aquariumlearning.com

Weak supervision

- Snorkel
  - Open-source project snorkel.org
  - Commercial platform snorkel.ai
- Rubrix: open-source solution




```python
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 by Paperspace

DOMINO DATA LAB

Data

Versioning: DVC, Pachyderm, LIQUIDATA

Labeling: Label Studio, scale, prodigy

Processing: dbt, DAGSTER

Exploration: pandas, dask, RAPIDS

Datasets

Sources: databricks, S3, Parquet, snowflake

Development

Frameworks & Distributed Training: PYTORCH, fast.ai, DeepSpeed, FairScale

Experiment and Model Management: comet, Weights & Biases, TensorBoard, mlflow, neptune.ai

Resource Management: Determined AI, slurm

Software Engineering: jupyter, Streamlit, git

Compute: CoreWeave, Lambda

Deployment

Feature Store: tecton

Monitoring: fiddler

Edge: OctoML, NVIDIA TensorRT, TensorFlowLite, ONNX

Web: ALGORITHMIA, SELDON

CI / Testing: Buildkite, great_expectations

or or

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
DVC	✅ (Apache 2.0)	✅	✅	✅	❌
DELTA LAKE	✅ (Apache 2.0)	❌	✅	❌	✅
Git Large File Storage	✅ (MIT)	✅	❌	✅	❌
Pachyderm	✅⚠ (Non-standard license)	✅	✅	❌	✅
DOLT	✅ (Apache 2.0)	❌	❌	❌	❌
lakeFS	✅ (Apache 2.0)	✅	❌	❌	✅
DAGsHub					

Warning: This is biased toward DVC

https://dagshub.com/blog/data-version-control-tools/

DVC

Open-source
Version Control System
for Machine Learning Projects

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"
```

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:

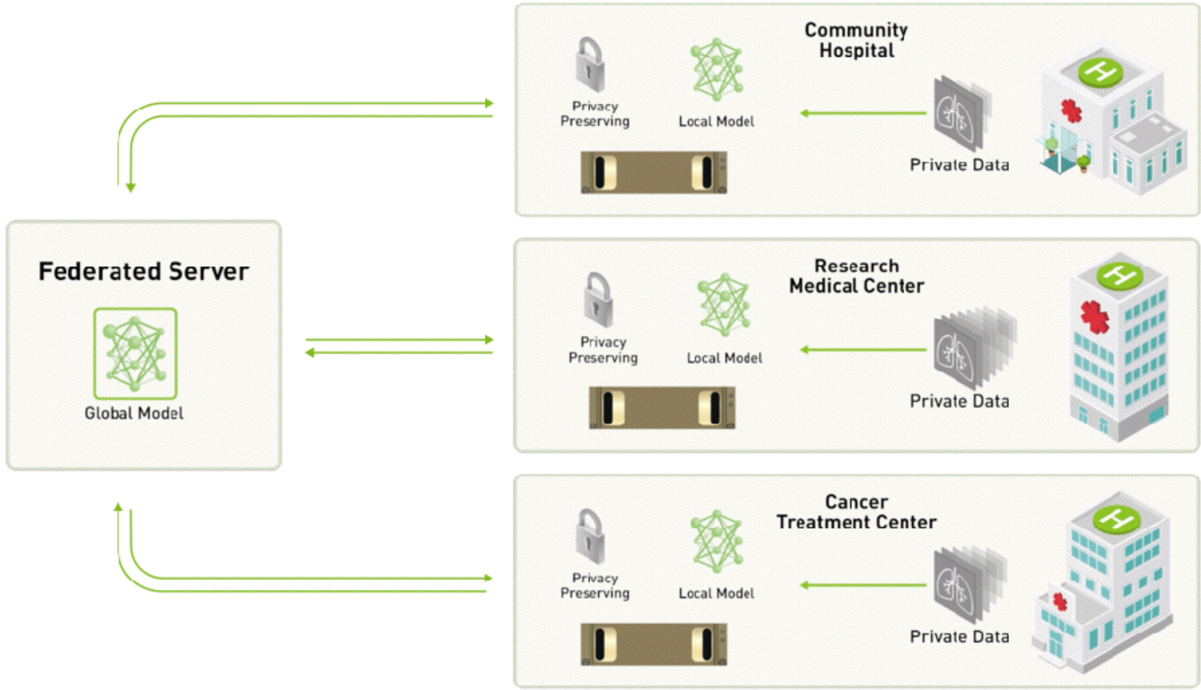
```
$ git add .gitignore matrix.pkl.dvc model.pkl.dvc
$ git commit -m "add featurization and train steps to the pipe
```

4

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

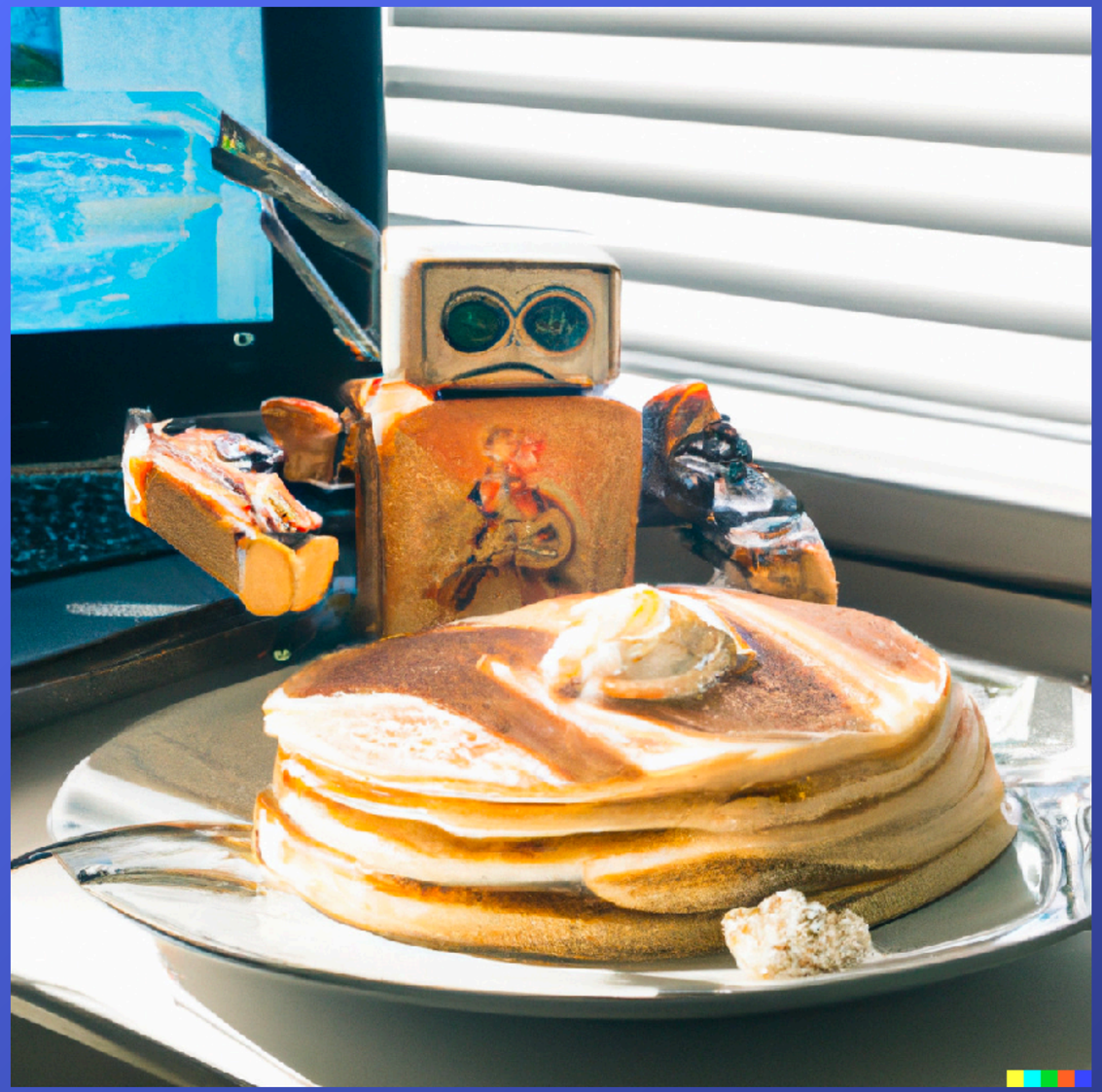
Research Area: Privacy

- 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!