Full Stack Deep Learning

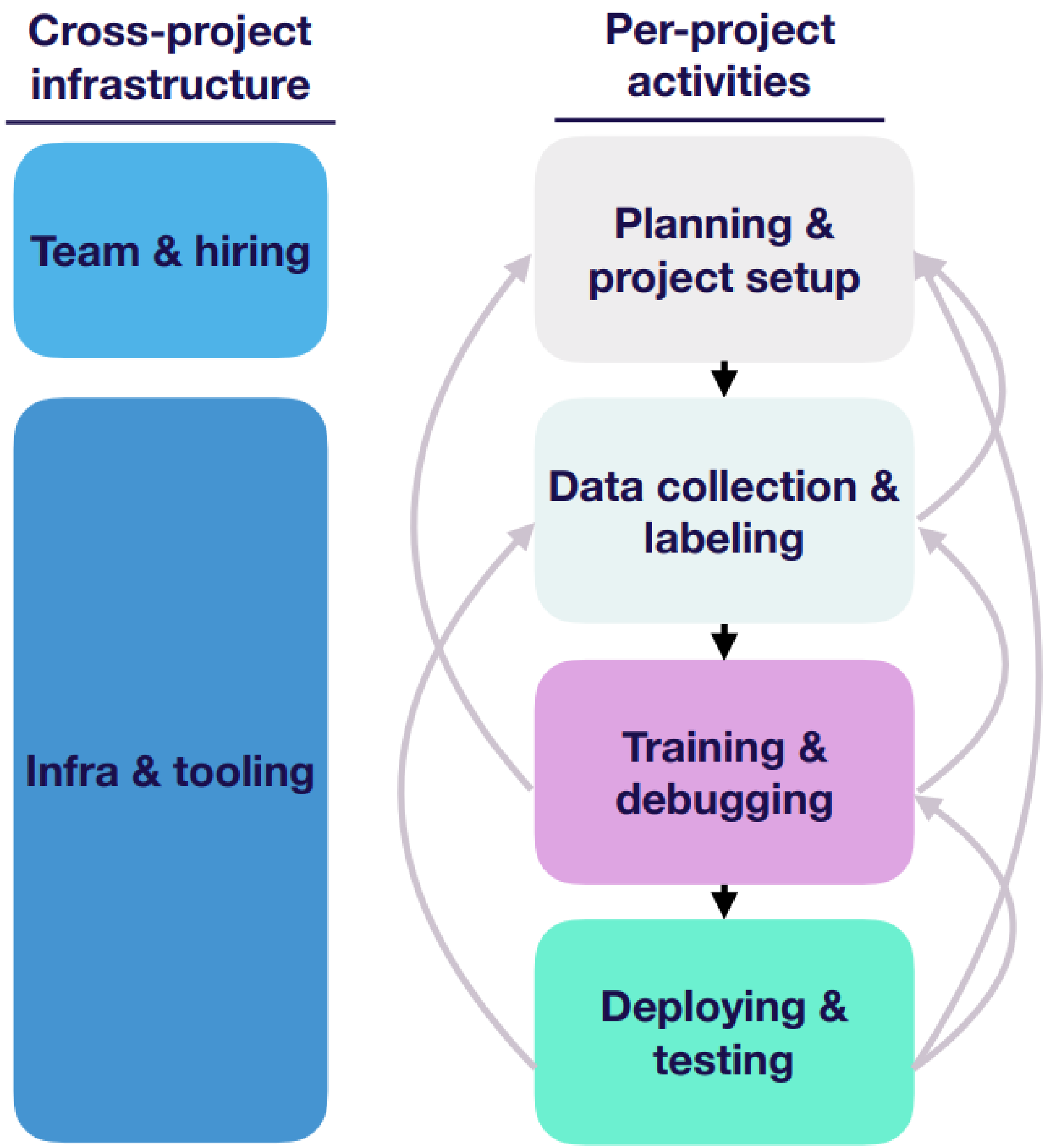
WU WEI

Outline

1. Setting up Machine Learning Projects
2. Infrastructure & Tooling
3. Data Management
4. Machine Learning Teams
5. Troubleshooting Deep Neural Networks
6. Testing & Deployment

1. Setting up Machine Learning Projects

Lifecycle of a ML project

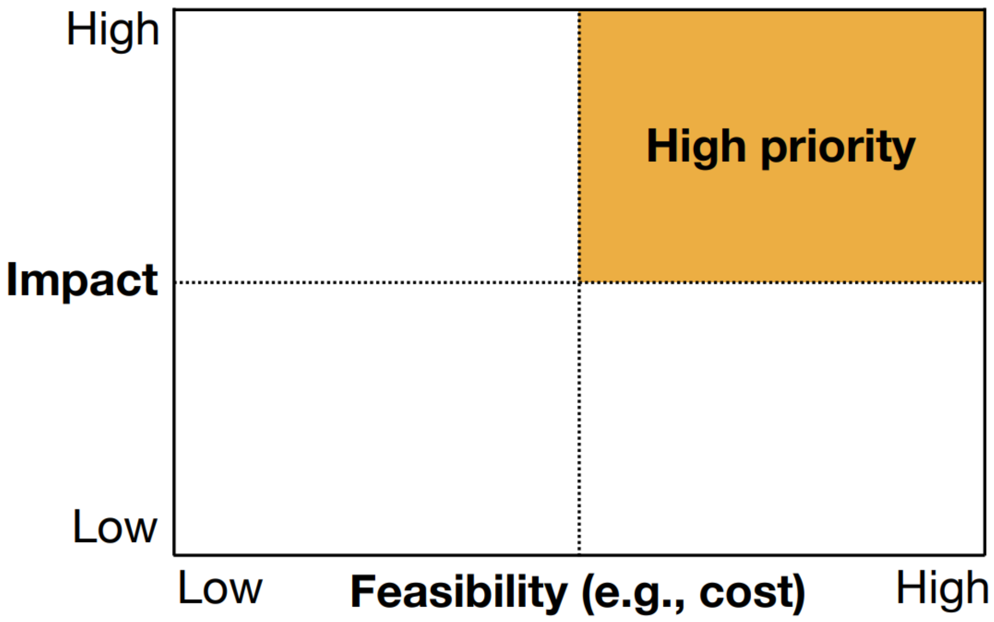


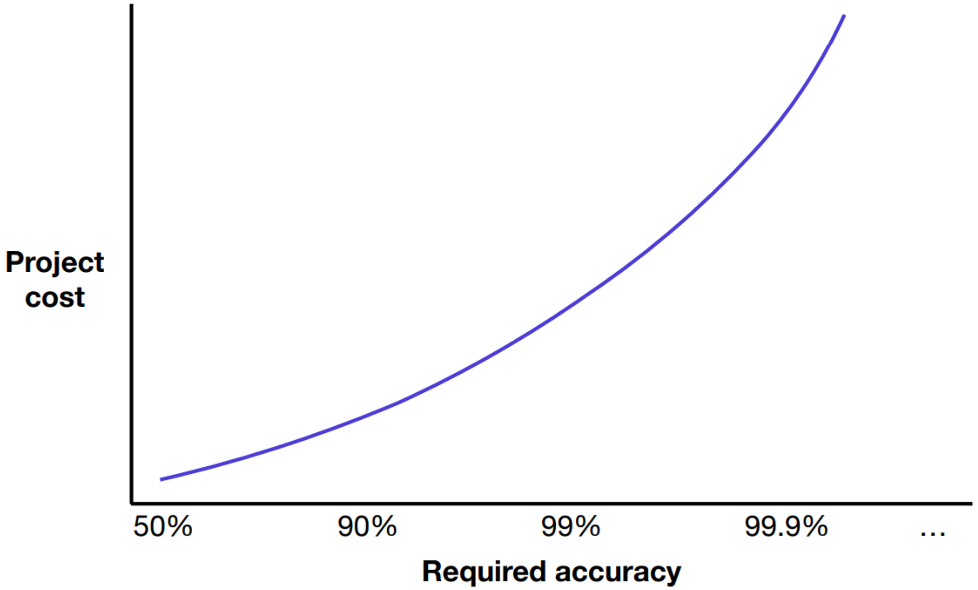
Note:

We need an excellent team and reliable infrastructure, and then we will face the question: Should we divide the entire project into multiple links, or should a member be responsible for the entire process? These two methods have their own advantages and disadvantages. According to my experience, small research projects are generally completed independently, while large online projects require teamwork. The industry's practice is generally to divide a large system into layers. For example, the recommendation system is divided into recall, rank, rerank and other links.

Another problem is that we always ignore the early preparation of the project. As a result, we spend a lot of unnecessary time and energy in places that are not expected. Maybe the goal is not clear, maybe the data does not meet the requirements, etc.

Prioritizing Projects





Note:

It is very important to clarify the priority and cost of the project, we should give priority to the important and low-cost projects.

At the same time, as a engineer, we need to give a trajectory of performance and cost as a reference for the business side when specifying business goals.

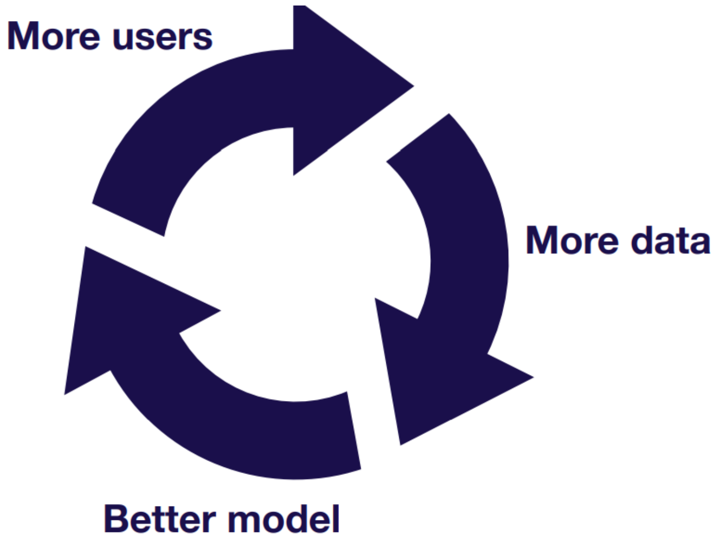
Archetypes

	Examples
Improve an existing process	• Improve code completion in an IDE • Build a customized recommendation system • Build a better video game AI
Augment a manual process	• Turn sketches into slides • Email auto-completion • Help a radiologist do their job faster
Automate a manual process	• Full self-driving • Automated customer support • Automated website design

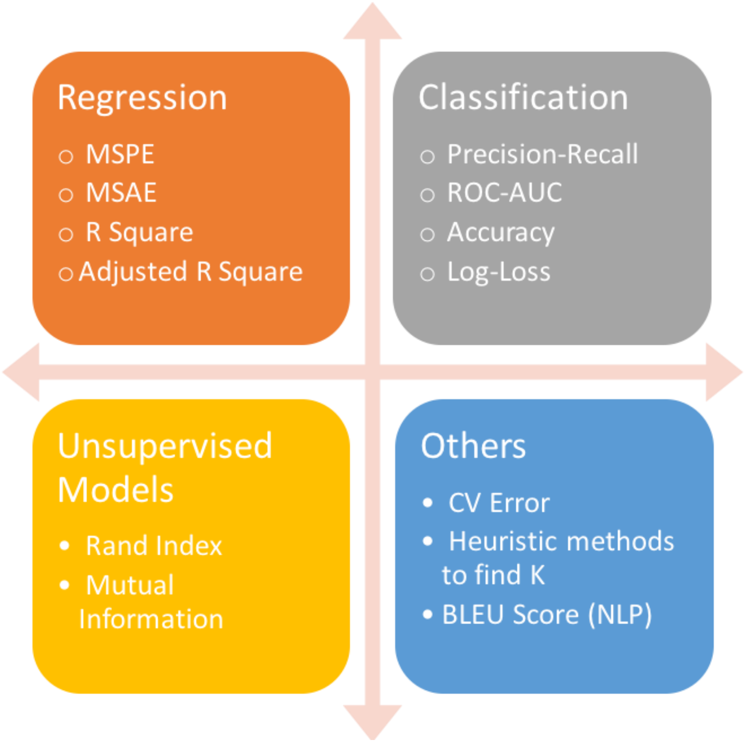
Note:

Clarifying the type of ML project can help us to better plan ahead.

Establishing highly automated Data flywheels allows us to do more with less.

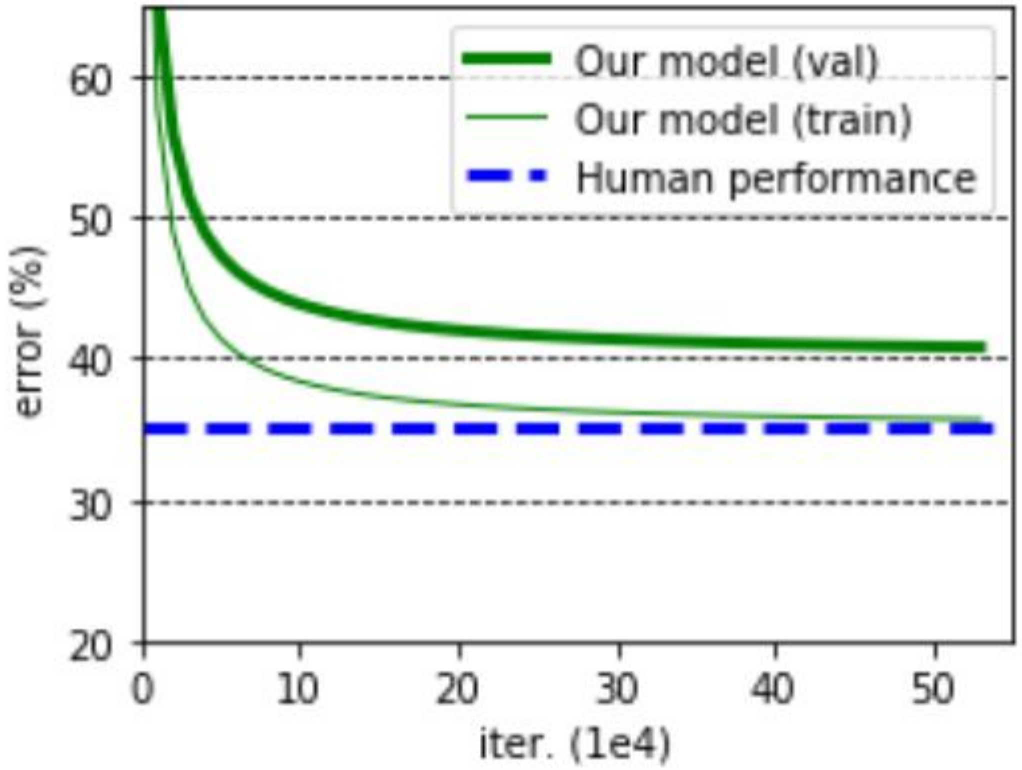


Metrics



The most important part of this part is a sentence: How to pick a single number to optimize, we know that there are many indicators to evaluate the model, how to choose the most suitable according to business goals Metric, not only requires knowledge of machine learning, but also requires a deep understanding of the business.

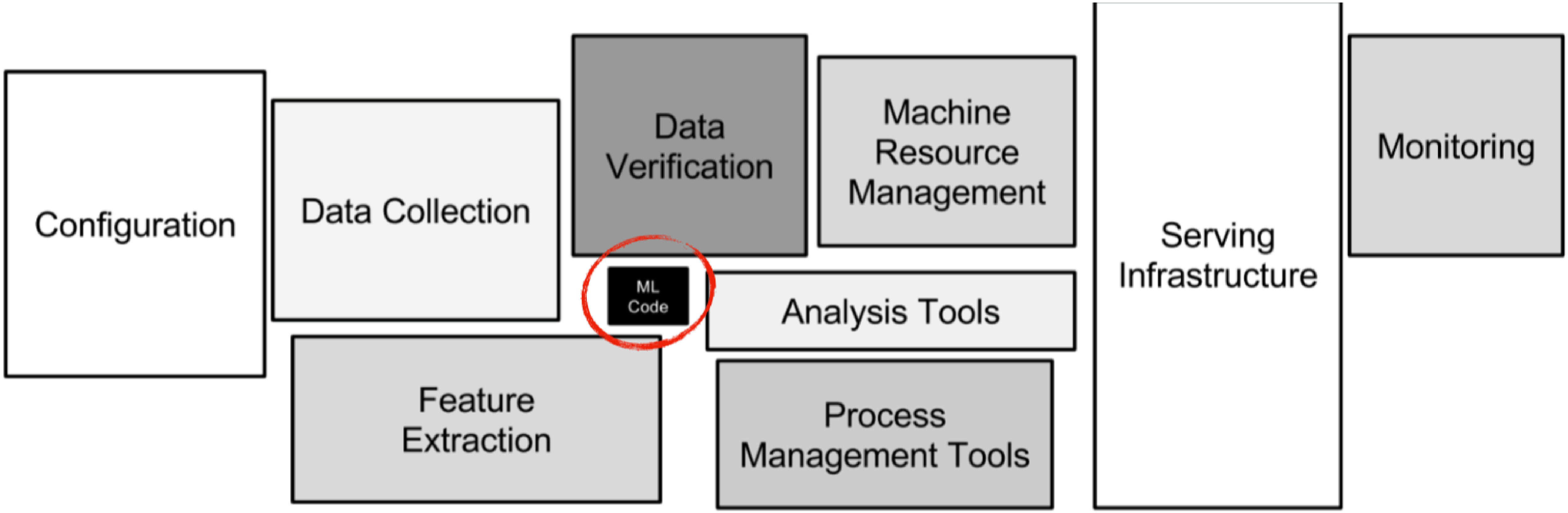
Baselines



Sometimes, when we take over a machine learning project, in order to shorten the time, we often ignore the baseline and directly try the model we think is the best.

However, this is a wrong approach. A good baseline will have a positive impact on the project, such as clear optimization goals, quick trial, etc. If we skip this link, we may be like a headless fly that cannot find the right direction.

2. Infrastructure & Tooling


Configuration
Data Collection
Data Verification
Machine Resource Management
Serving Infrastructure
Monitoring
ML Code
Analysis Tools
Feature Extraction
Process Management Tools

GPU Comparison Table

Card	Release	Arch	Use-case	RAM (Gb)	32bit TFlops	Tensor TFlops	16bit	Cost	Cloud
K80	2014H2	Kepler	Server	24	5	N/A	No	used	AWS, GCP, MS
Titan X	2015H1	Maxwell	Enthusiast	12	6	N/A	No	used	
P100	2016H1	Pascal	Server	16	10	N/A	Yes	used	GCP, MS
1080 Ti	2017H1	Pascal	Consumer	11	13	N/A	No	used	
V100	2017H1	Volta	Server	16	14	120	Yes	$10000	AWS, GCP, MS
Titan V	2017H2	Volta	Enthusiast	12	14	110	Yes	used	
2080 Ti	2018H2	Turing	Consumer	11	13	60	Yes	$1000	
Titan RTX	2018H2	Turing	Enthusiast	24	16	130	Yes	$2500	
RTX 8000	2018H2	Turing	Enthusiast	48	16	160	Yes	$5500	

- New NVIDIA architecture every year: Kepler —> Maxwell —> Pascal —> Volta -> Turing
- RAM: should fit meaningful batches of your model
- 32bit vs Tensor Tflops: Tensor Cores are specifically for deep learning operations (mixed precision)• Good for convolutional/transformer models

Resource Management

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

3. Data Management

1. Sources

- Most DL applications require lots of labelled data
  Exceptions: RL, GANs, "semi-supervised" learning
- Publicly available datasets = No competitive advantage
  But can serve as starting point

2. Data Labelling

- User Interfaces
- Sources of labour
- Service companies

3. Data Storage

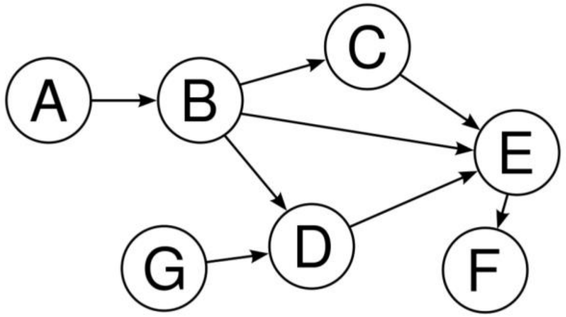
- Building blocks
  - Filesystem
  - Object Storage
  - Database
  - "Data Lake“
- What goes where
- Where to learn more

4. Data Versioning

- Level 0: unversioned
- Level 1: versioned via snapshot at training time
- Level 2: versioned as a mix of assets and code(recommend)
- Level 3: specialized data versioning solution

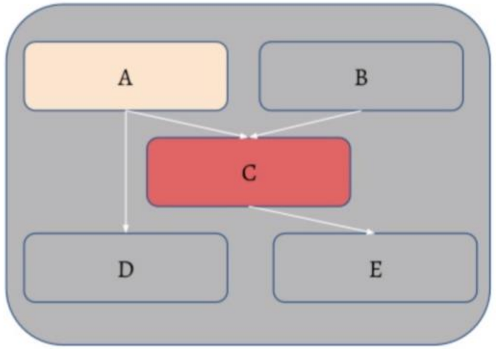
5. Data Processing

- Task Dependencies

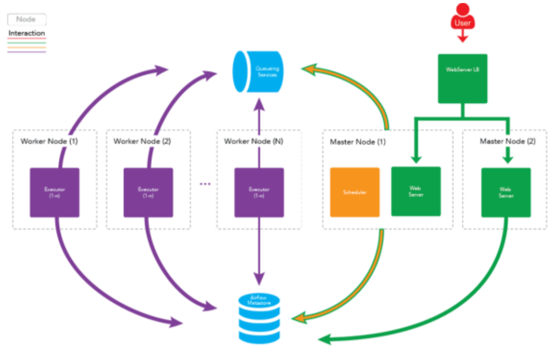


Some tasks can't be started until other tasks are finished.
Finishing a task should "kick off" its dependencies

- Airflow



- Distributing work

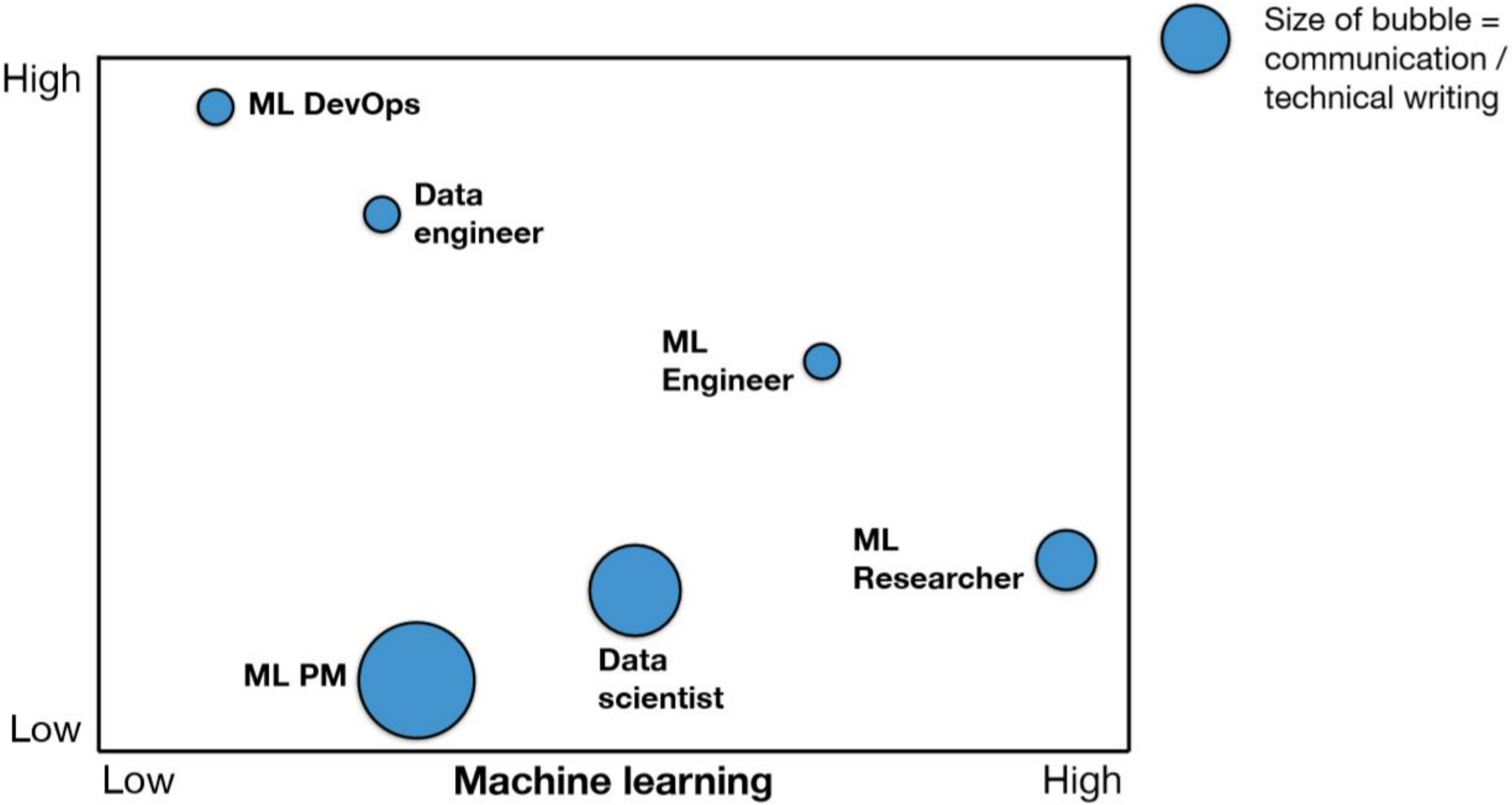


4. Machine Learning Teams

Breakdown of job function by role

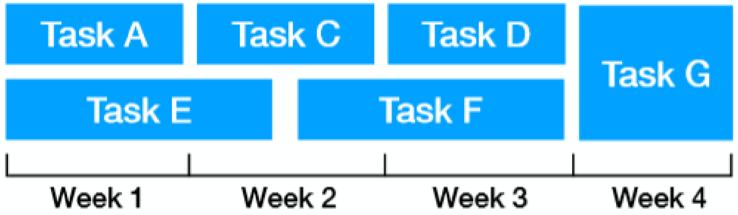
Role	Job Function	Work product	Commonly used tools
ML product manager	Work with ML team, business, users, data owners to prioritize & execute projects	Design docs, wireframes, work plans	Jira, etc
DevOps engineer	Deploy & monitor production systems	Deployed product	AWS, etc.
Data engineer	Build data pipelines, aggregation, storage, monitoring	Distributed system	Hadoop, Kafka, Airflow
ML engineer	Train & deploy prediction models	Prediction system running on real data (often in production)	Tensorflow, Docker
ML researcher	Train prediction models (often forward looking or not production-critical)	Prediction model & report describing it	Tensorflow, pytorch, Jupyter
Data scientist	Blanket term used to describe all of the above. In some orgs, means answering business questions using analytics	Prediction model or report	SQL, Excel, Jupyter, Pandas, SKLearn, Tensorflow

What skills are needed for the roles?

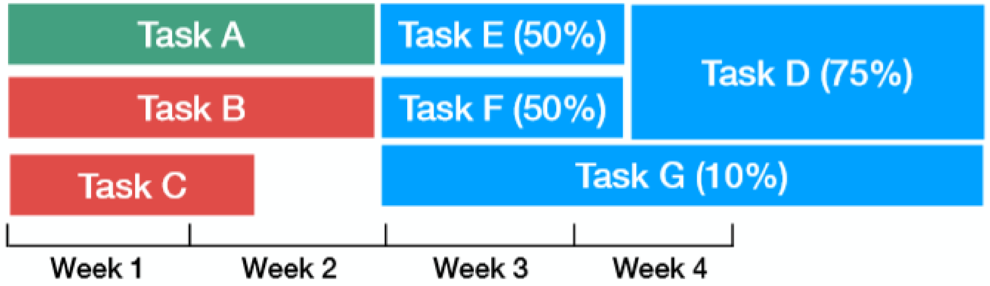


How to manage ML teams better

- Do ML project planning probabilistically
  - From:

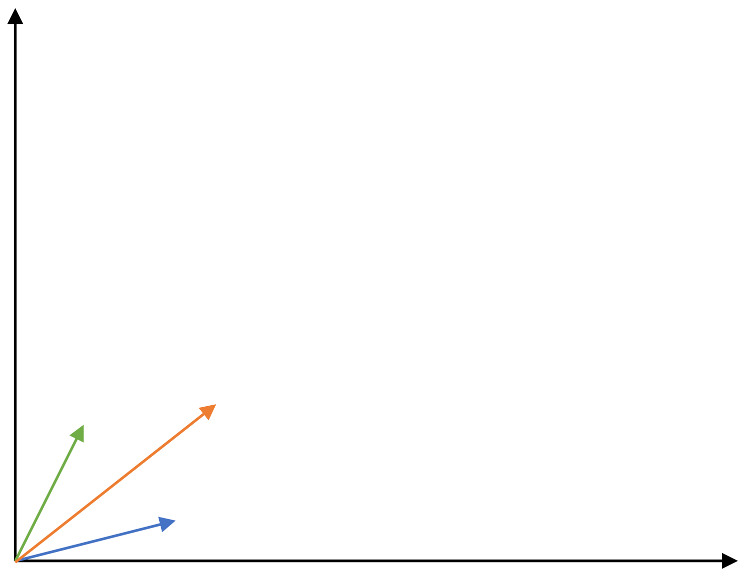


  - To:



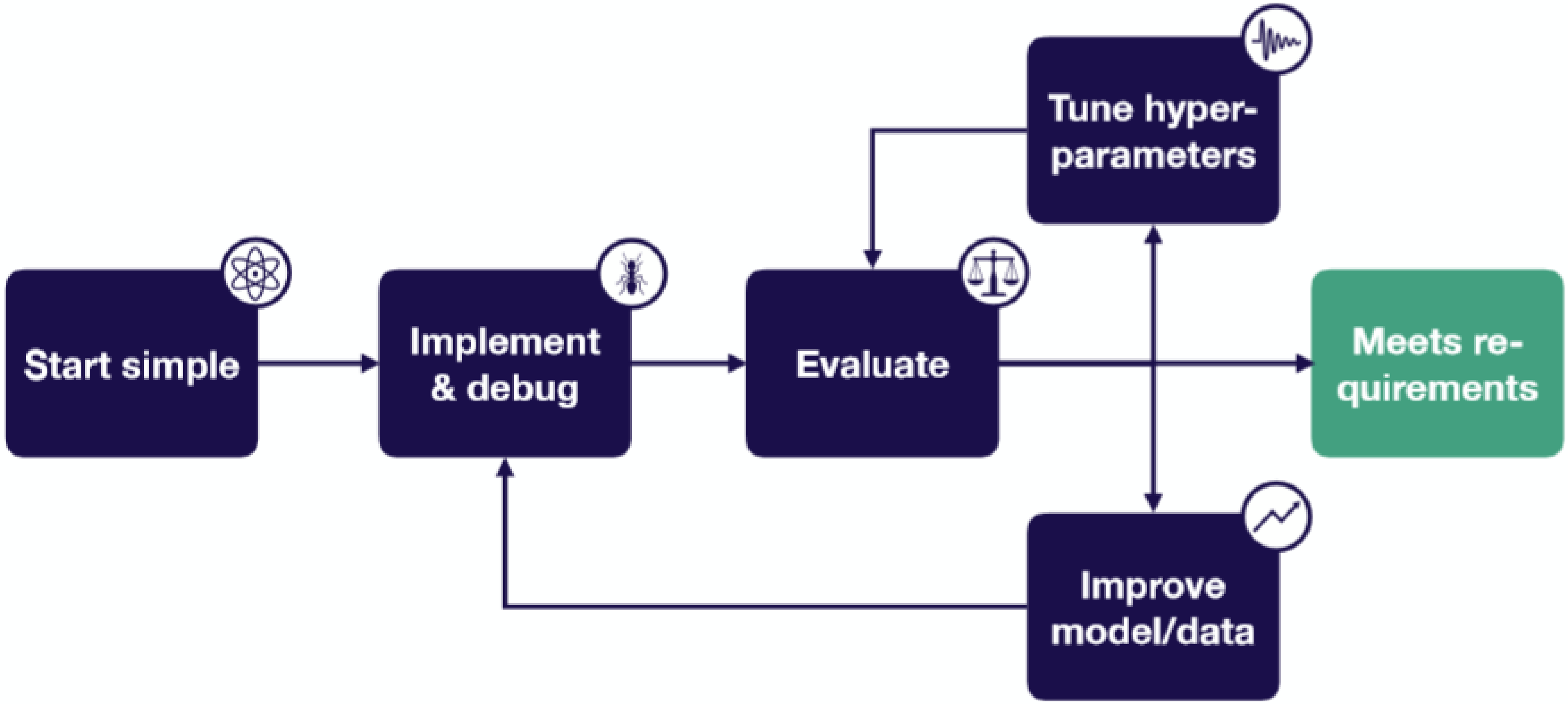
- Attempt a portfolio of approaches
- Measure progress based on inputs, not results
- Have researchers and engineers work together
- Get end-to-end pipelines together quickly to demonstrate quick wins
- Educate leadership on ML timeline uncertainty

The output of the team is the vector sum of individual efforts!



5. Troubleshooting Deep Neural Networks

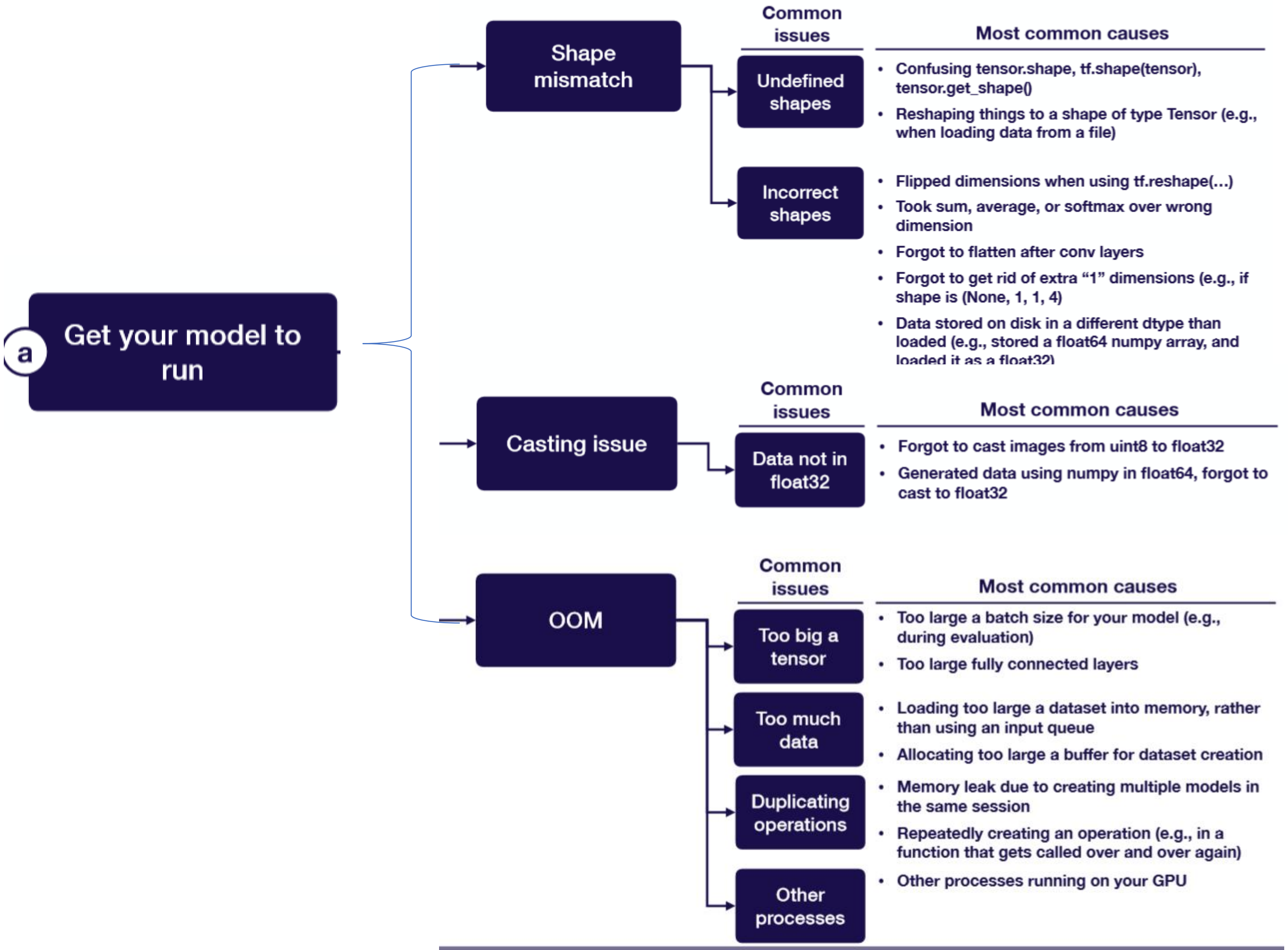
Strategy for DL troubleshooting



Starting simple

Steps	Summary
a Choose a simple architecture	• LeNet, LSTM, or fully connected
b Use sensible defaults	• Adam optimizer & no regularization
c Normalize inputs	• Subtract mean and divide by std, or just divide by 255 (ims)
d Simplify the problem	• Start with a simpler version of your problem (e.g., smaller dataset)

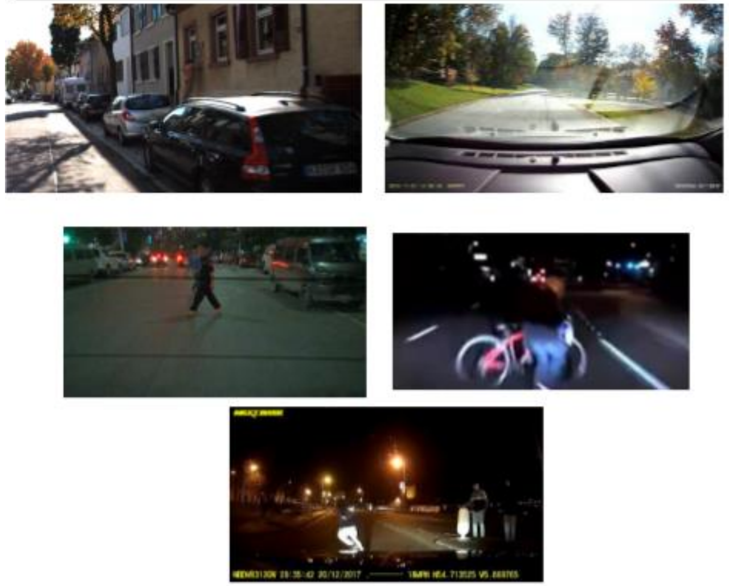
Implementing bug-free DL models



Error analysis

Test error = irreducible error + bias + variance + distribution shift + val overfitting

Test-val set errors (no pedestrian detected)



Train-val set errors (no pedestrian detected)

Error type	Error % (train-val)	Error % (test-val)	Potential solutions	Priority
1. Hard-to-see pedestrians	0.1%	0.1%	• Better sensors	Low
2. Reflections	0.3%	0.3%	• Collect more data with reflections • Add synthetic reflections to train set • Try to remove with pre-processing • Better sensors	Medium
3. Nighttime scenes	0.1%	1%	• Collect more data at night • Synthetically darken training images • Simulate night-time data • Use domain adaptation	High

Hyperparameter optimization

Which hyper-parameters to tune?

Choosing hyper-parameters

- More sensitive to some than others
- Depends on choice of model
- Rules of thumb (only) to the right
- Sensitivity is relative to default values! (e.g., if you are using all-zeros weight initialization or vanilla SGD, changing to the defaults will make a big difference)

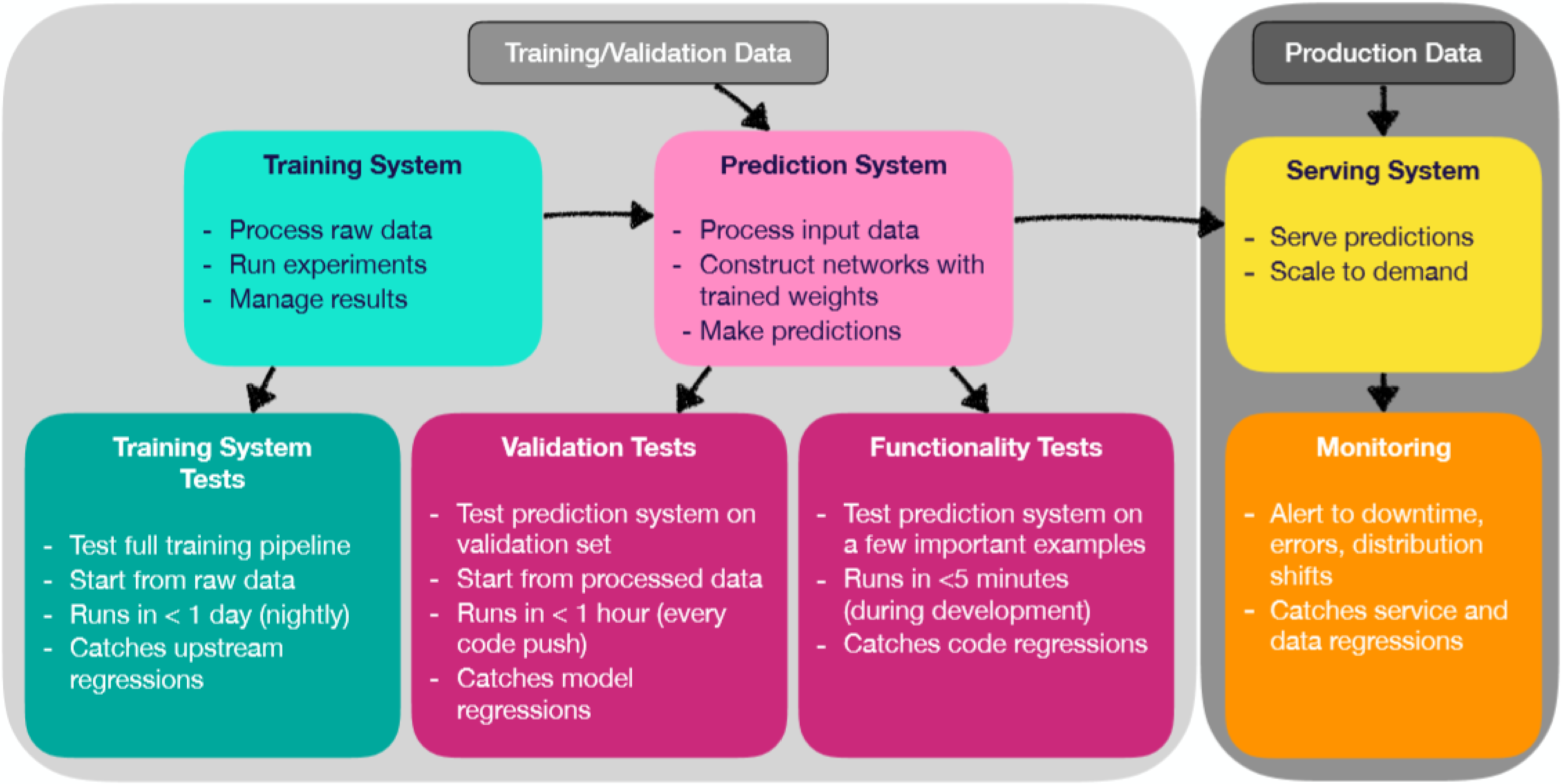
Hyperparameter	Approximate sensitivity
Learning rate	High
Learning rate schedule	High
Optimizer choice	Low
Other optimizer params (e.g., Adam beta1)	Low
Batch size	Low
Weight initialization	Medium
Loss function	High
Model depth	Medium
Layer size	High
Layer params (e.g., kernel size)	Medium
Weight of regularization	Medium
Nonlinearity	Low

How to tune the hyperparameters?

- manual hyperparam optimization
- grid search
- random search
- coarse-to-fine
- Bayesian hyperparam opt
- …

6. Testing & Deployment

Project structure



ML Test Score

1	Feature expectations are captured in a schema.
2	All features are beneficial.
3	No feature's cost is too much.
4	Features adhere to meta-level requirements.
5	The data pipeline has appropriate privacy controls.
6	New features can be added quickly.
7	All input feature code is tested.

Data Tests

1	Model specs are reviewed and submitted.
2	Offline and online metrics correlate.
3	All hyperparameters have been tuned.
4	The impact of model staleness is known.
5	A simpler model is no[illegible]
6	Model quality is suffic[illegible] slices.
7	The model is tested for considerations of inclusion.

Model Tests

1	Training is reproducible.
2	Model specs are unit tested.
3	The ML pipeline is Integration tested.
4	Model quality is validated before serving.
5	The model is debuggable.
6	Models are canaried before serving.
7	Serving models can be rolled back.

ML Infrastructure Tests

1	Dependency changes result in notification.
2	Data invariants hold for inputs.
3	Training and serving are not skewed.
4	Models are not too stale.
5	Models are numerically stable.
6	Computing performance has not regressed.
7	Prediction quality has not regressed.

Monitoring Tests

Testing / CI

Unit / Integration Tests

- Tests for individual module functionality and for the whole system

Continuous Integration

- Tests are run every time new code is pushed to the repository, before updated model is deployed.

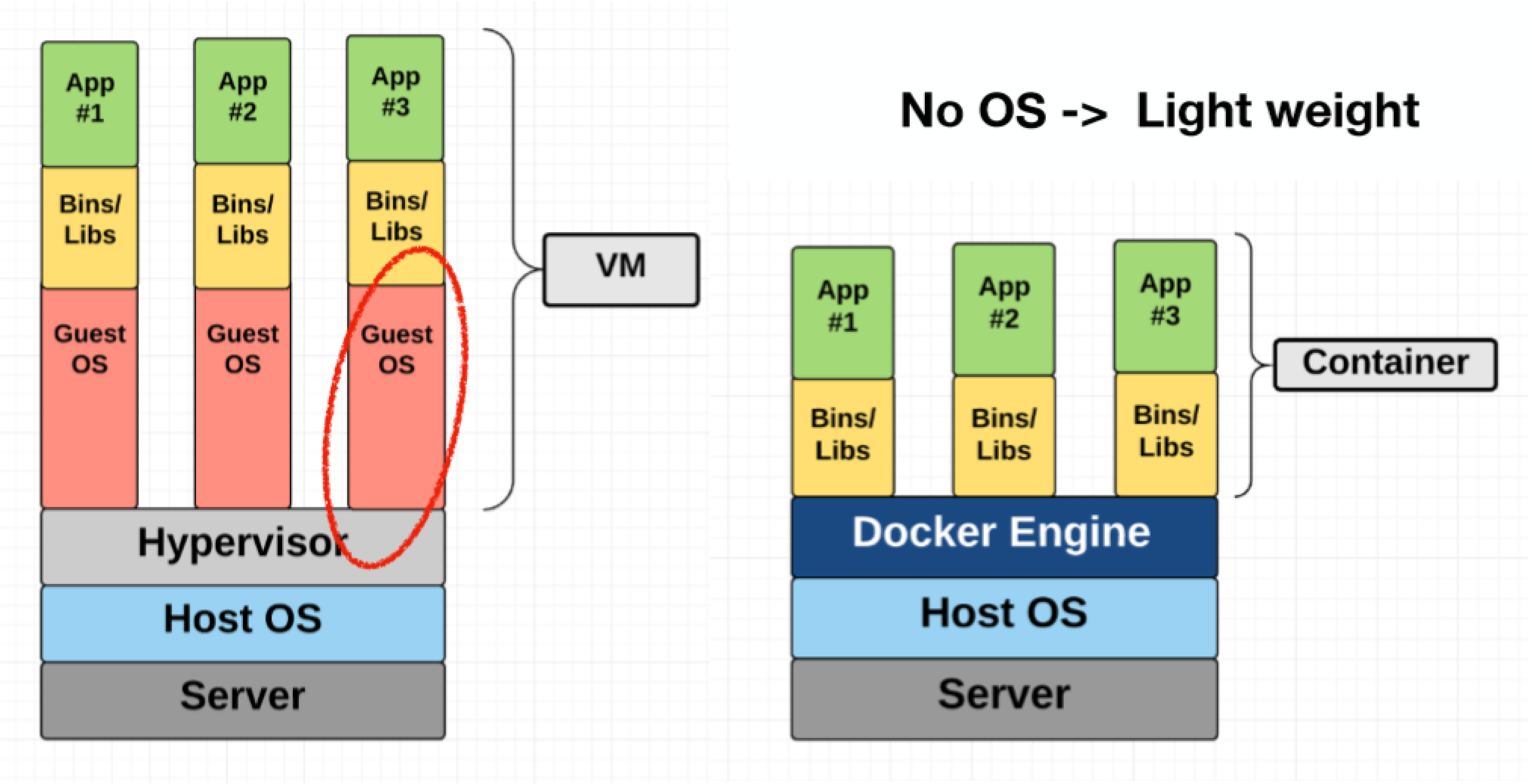
SaaS for CI

- CircleCI, Travis, Jenkins, Buildkite

Containerization (via Docker)

- A self-enclosed environment for running the tests

What is docker?



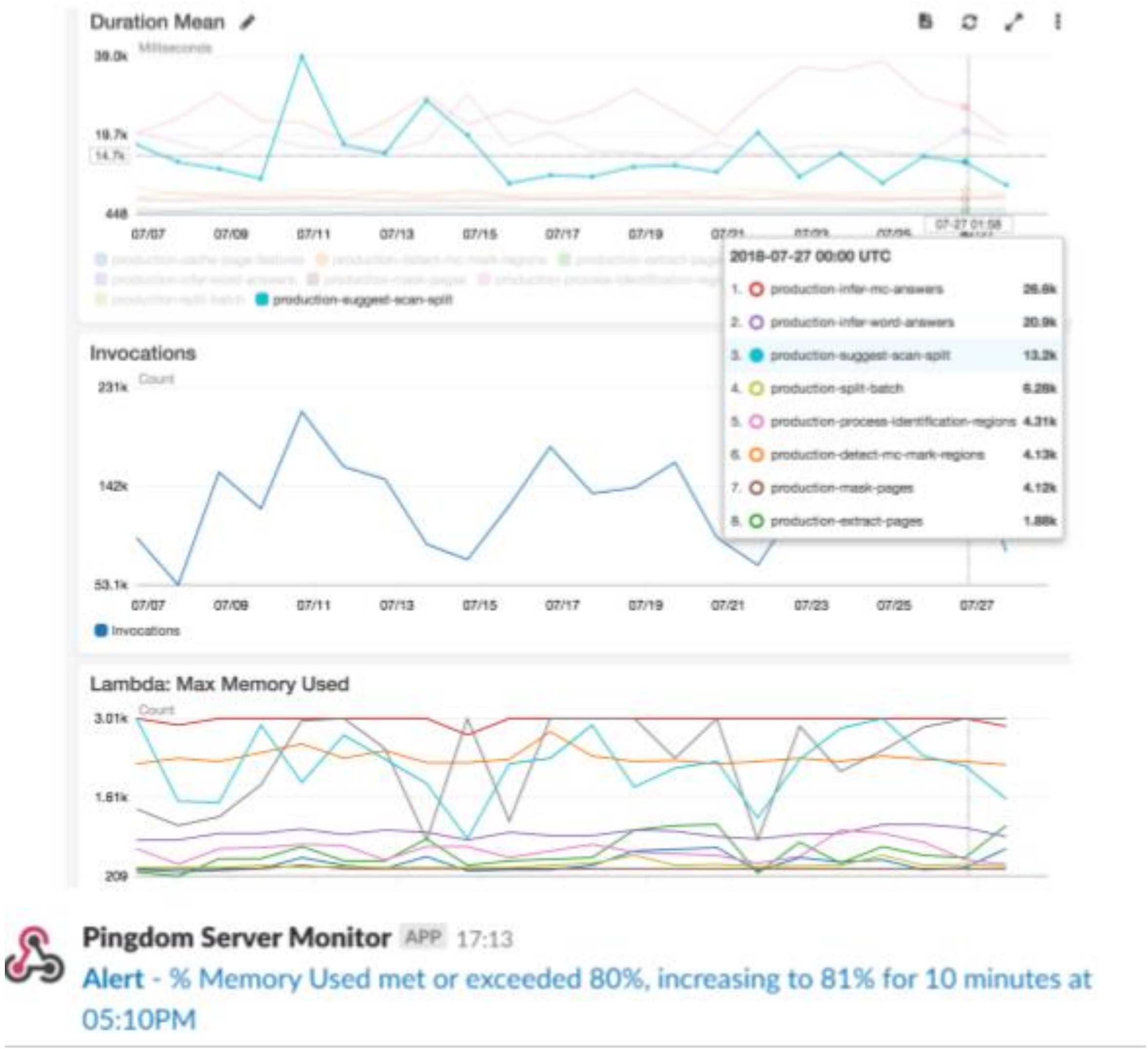
Web Deployment

REST API
- Serving predictions in response to canonically-formatted HTTP requests
- The web server is running and calling the prediction system

Options
- Deploying code to VMs, scale by adding instances
- Deploy code as containers, scale via orchestration
- Deploy code as a “serverless function”
- Deploy via a model serving solution

Methods
- DEPLOY CODE TO YOUR OWN BARE METAL
- DEPLOY CODE TO CLOUD INSTANCES
- DEPLOY DOCKER CONTAINERS TO CLOUD INSTANCES
- DEPLOY SERVERLESS FUNCTIONS

Monitoring



- Alarms for when things go wrong, and records for tuning things.
- Cloud providers have decent monitoring solutions.
- Anything that can be logged can be monitored (i.e. data skew)
- Will explore in lab

Data Distribution Monitoring

Window Size
NA/6 predictions

○ since last Scheduled Test ◉ by Time ○ by Data

Till Date	Y	M	W	D	H	MI	S

Domino Data Lab

Model Drift

Search

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

Problems of hardware Mobile

Embedded and mobile frameworks are less fully featured than full PyTorch/Tensorflow

- Have to be careful with architecture(Tensorflow Lite/PyTorch Mobile)
- Interchange format(ONNX model)

Embedded and mobile devices have little memory and slow/expensive compute

- Have to reduce network size / quantize weights / distill knowledge