Full Stack Deep Learning

Setting up Machine Learning Projects

Josh Tobin, Sergey Karayev, Pieter Abbeel

Goals for the lecture

1. Introduce our framework for understanding ML projects
2. Describe best practices for planning & setting up ML projects

Running case study - pose estimation

(x, y, z) **Position (L2 loss)**

(ϕ, θ, ψ) **Orientation (L2 loss)**

Xiang, Yu, et al. "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes." arXiv preprint arXiv:1711.00199 (2017).

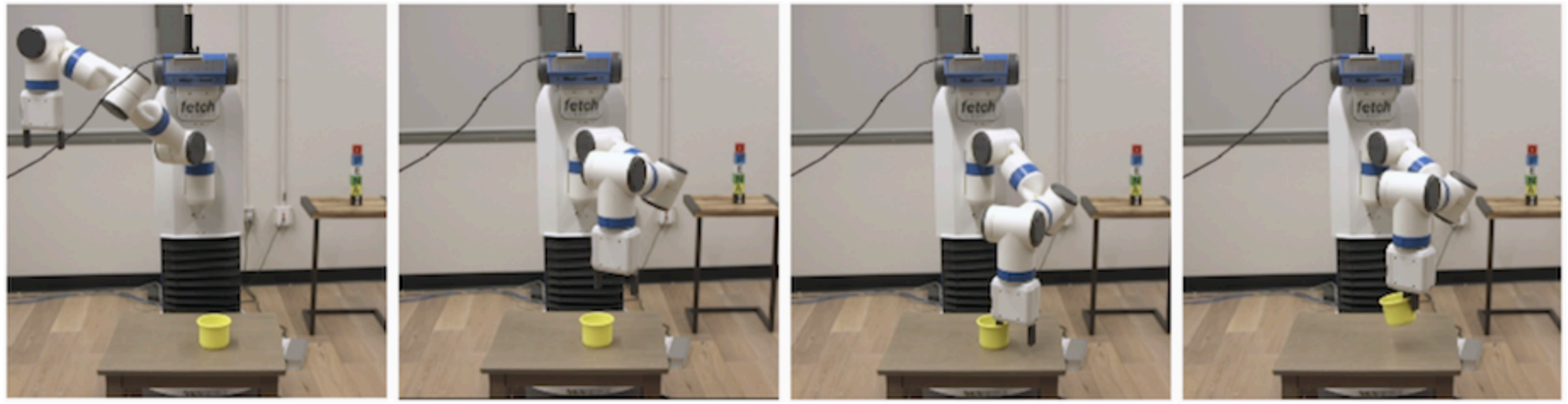
Hypothetical Co. *Full Stack Robotics (FSR)* wants to use pose estimation to enable grasping

(x, y, z)

(ϕ, θ, ψ)

Grasp model

Motor commands



Goals for the lecture

1. **Introduce our framework for understanding ML projects**

2. Describe best practices for planning & setting up ML projects

Lifecycle of a ML project

Planning & project setup

Lifecycle of a ML project

Planning & project setup

- Decide to work on pose estimation
- Determine requirements & goals
- Allocate resources
- Etc.

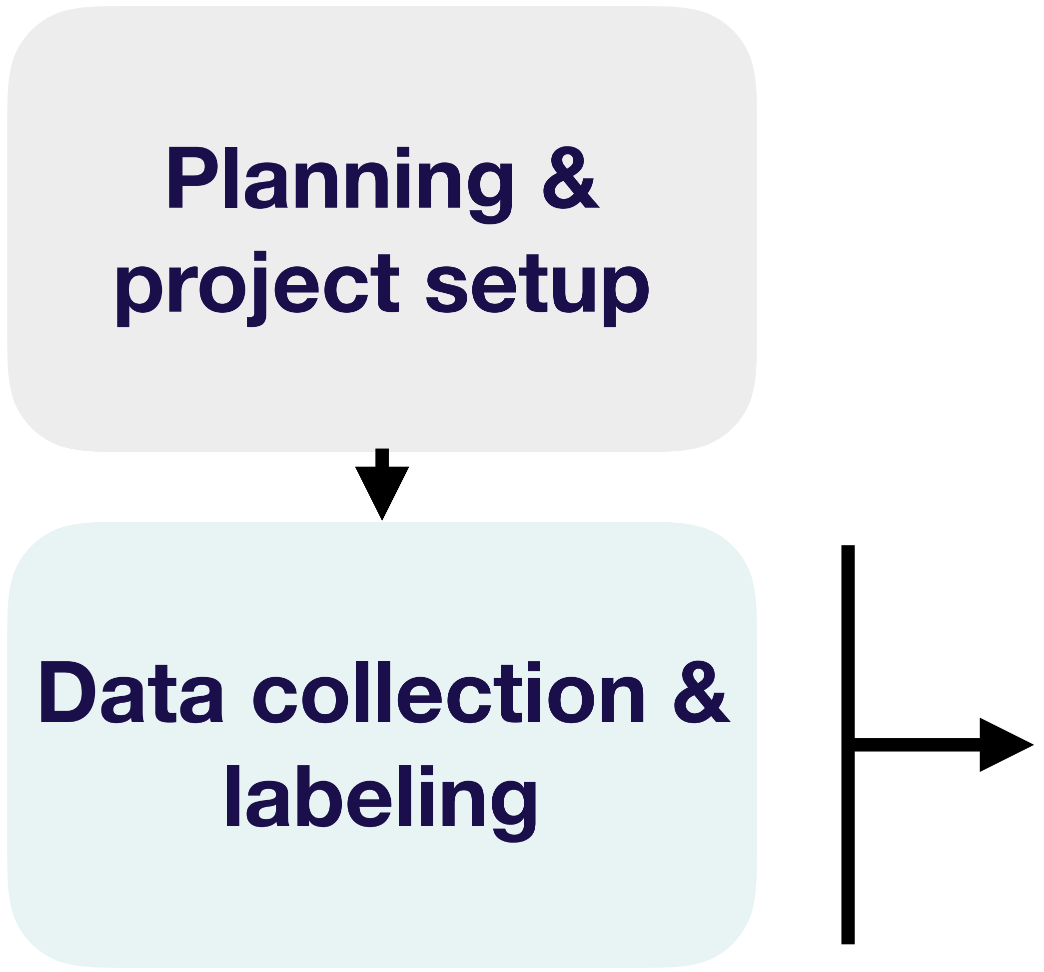
Lifecycle of a ML project

Planning & project setup

↓

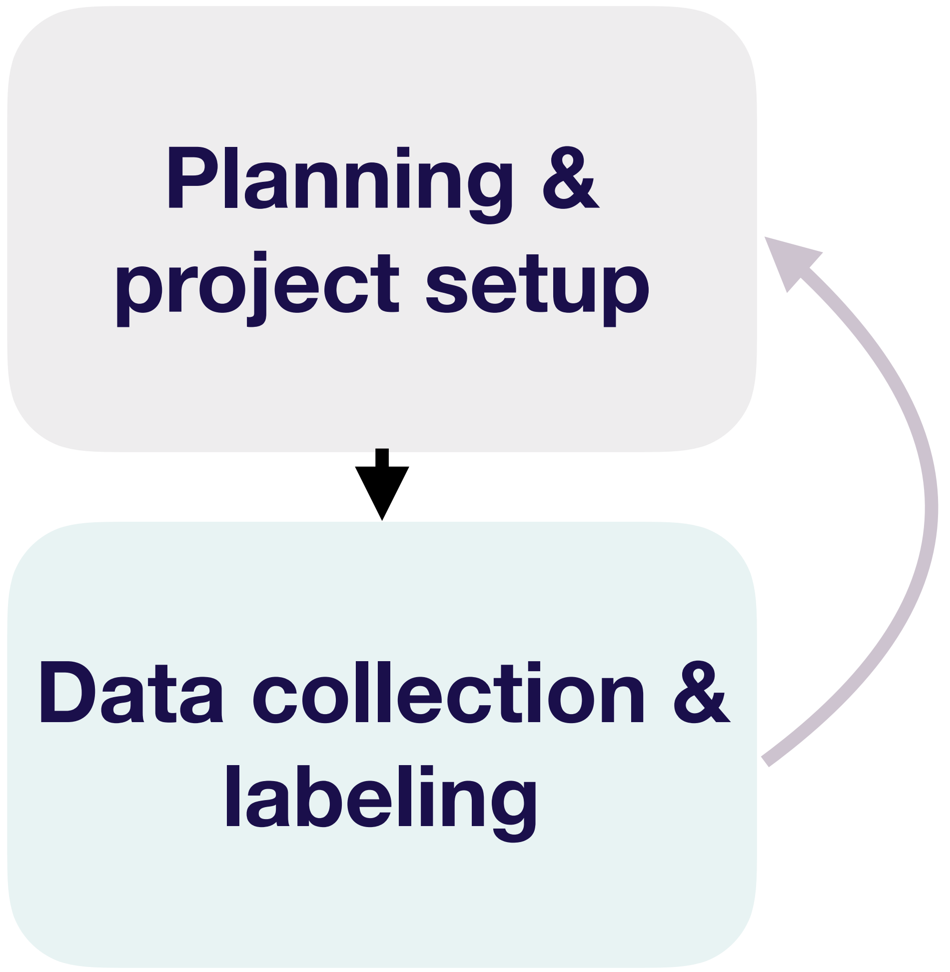
Data collection & labeling

Lifecycle of a ML project

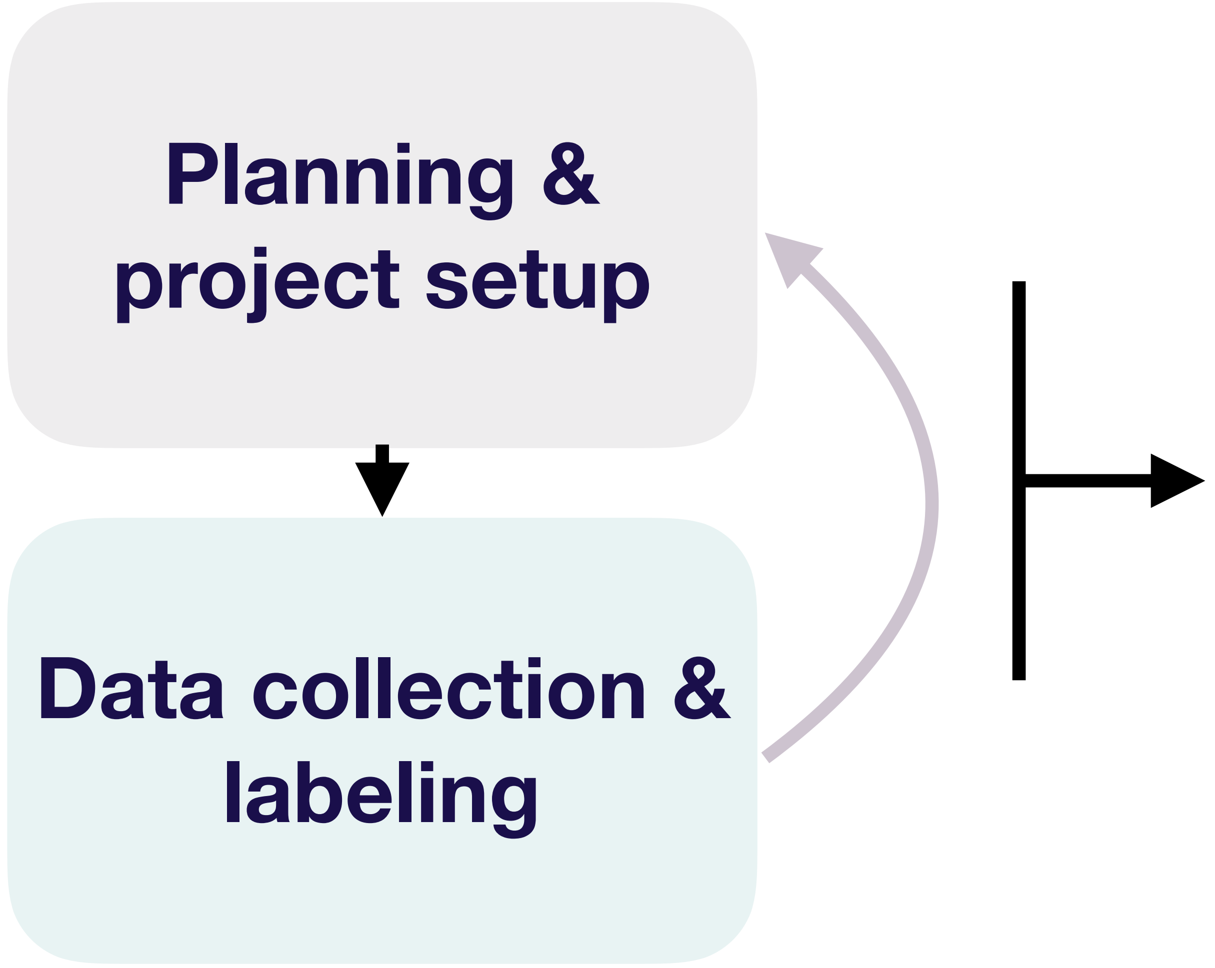


- Collect training objects
- Set up sensors (e.g., cameras)
- Capture images of objects
- Annotate with ground truth (how?)

Lifecycle of a ML project

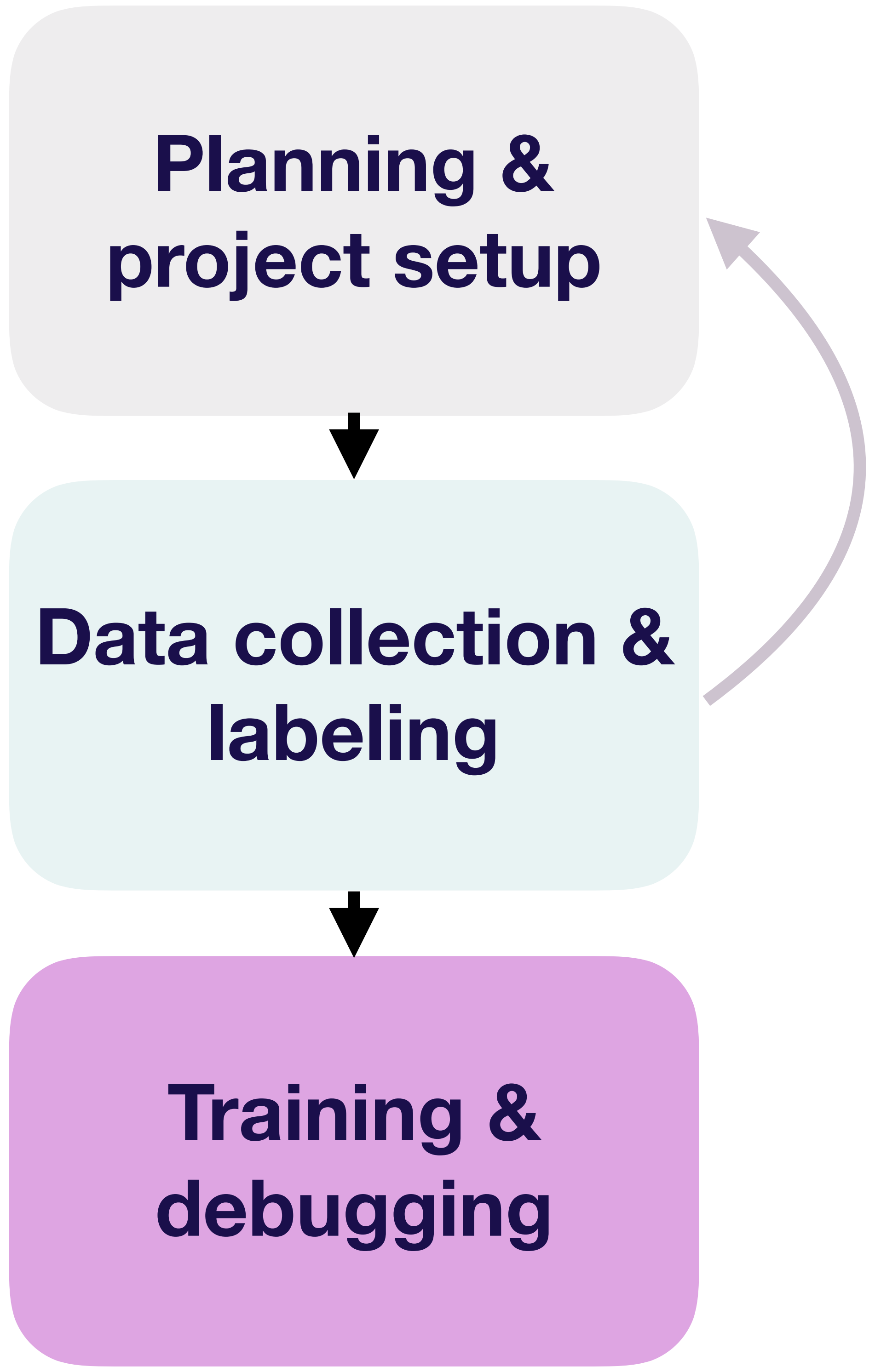


Lifecycle of a ML project

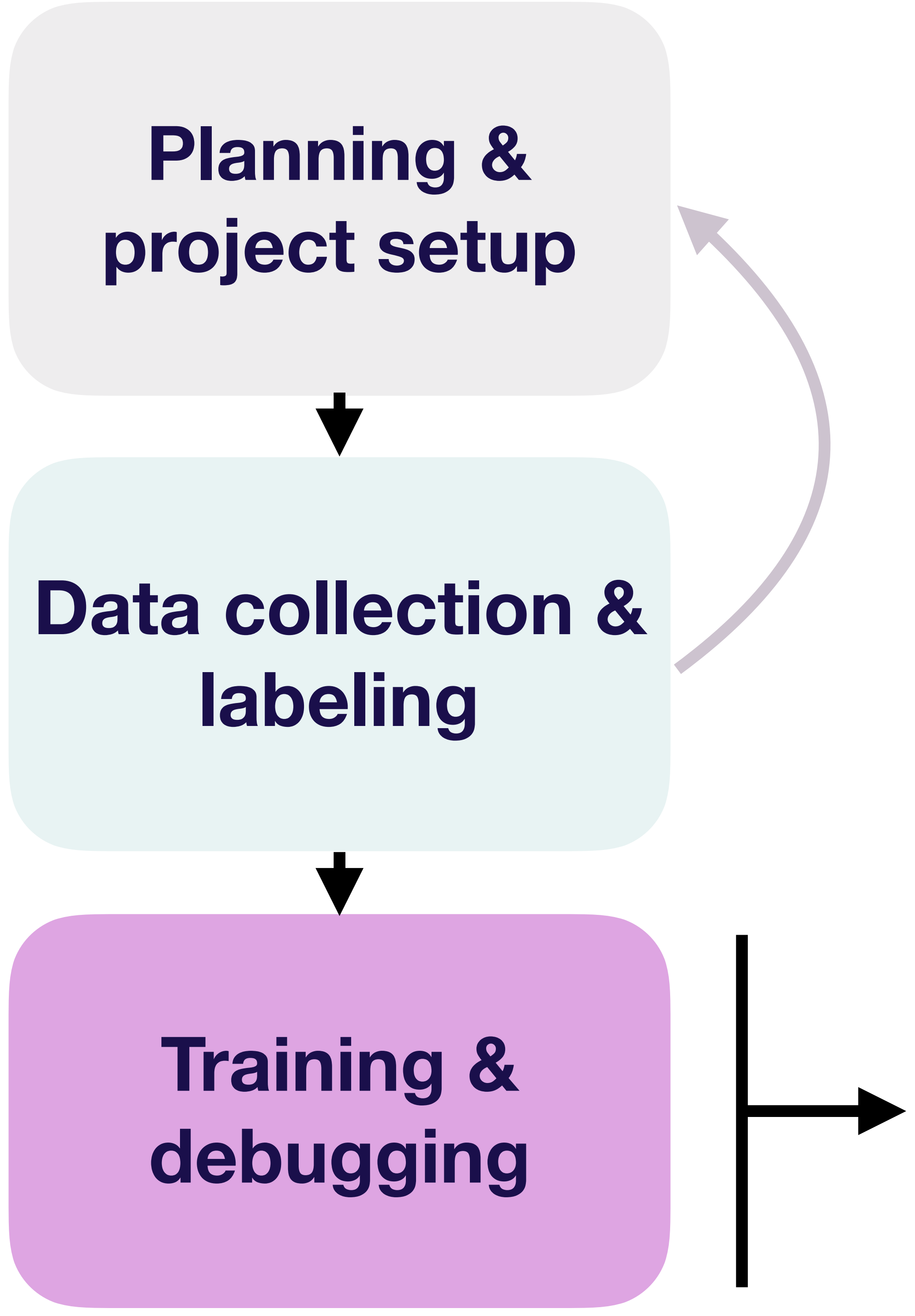


- Too hard to get data.
- Easier to label a different task (e.g., hard to annotate pose, easier to annotate per-pixel segmentation)

Lifecycle of a ML project

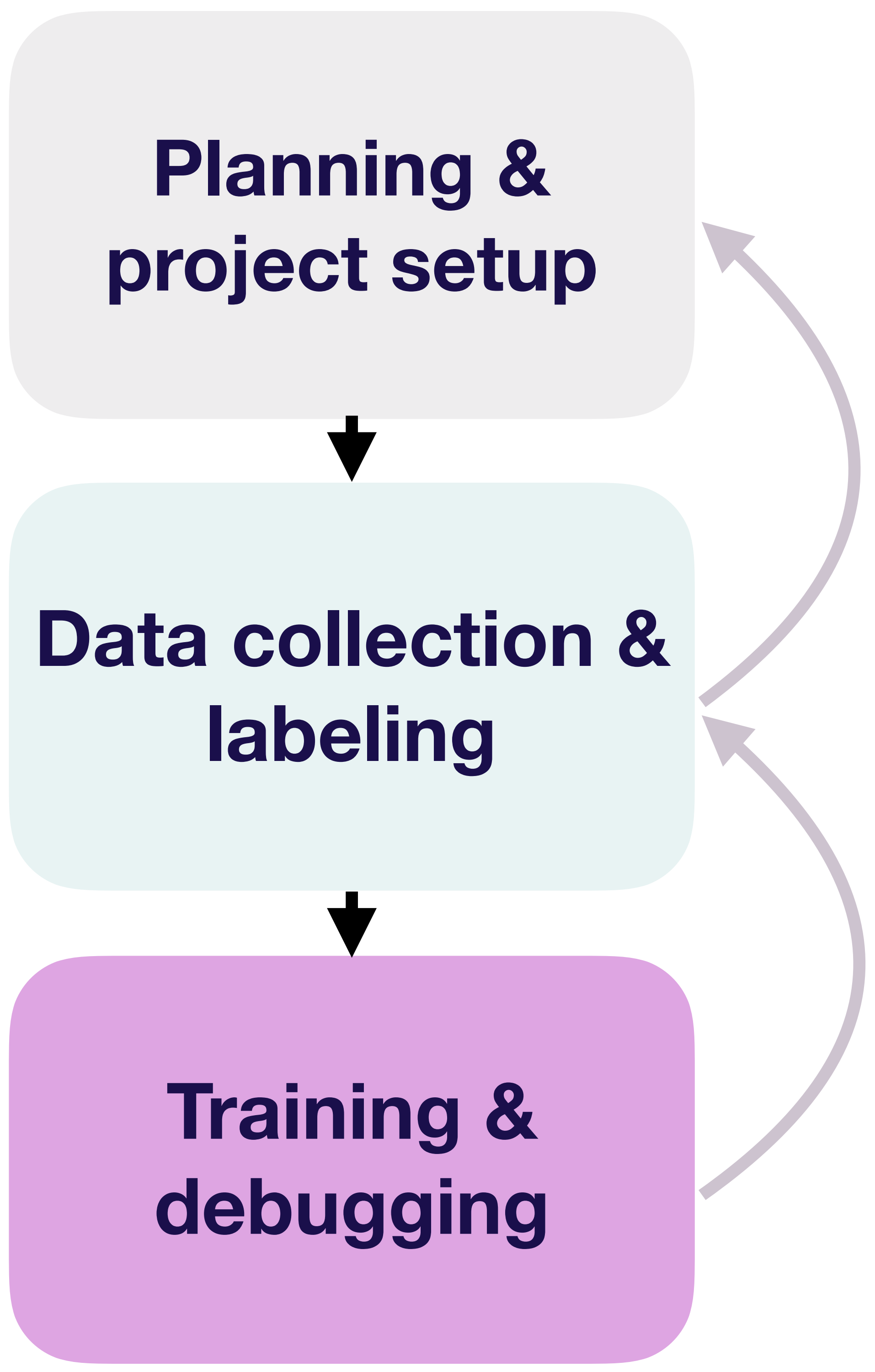


Lifecycle of a ML project

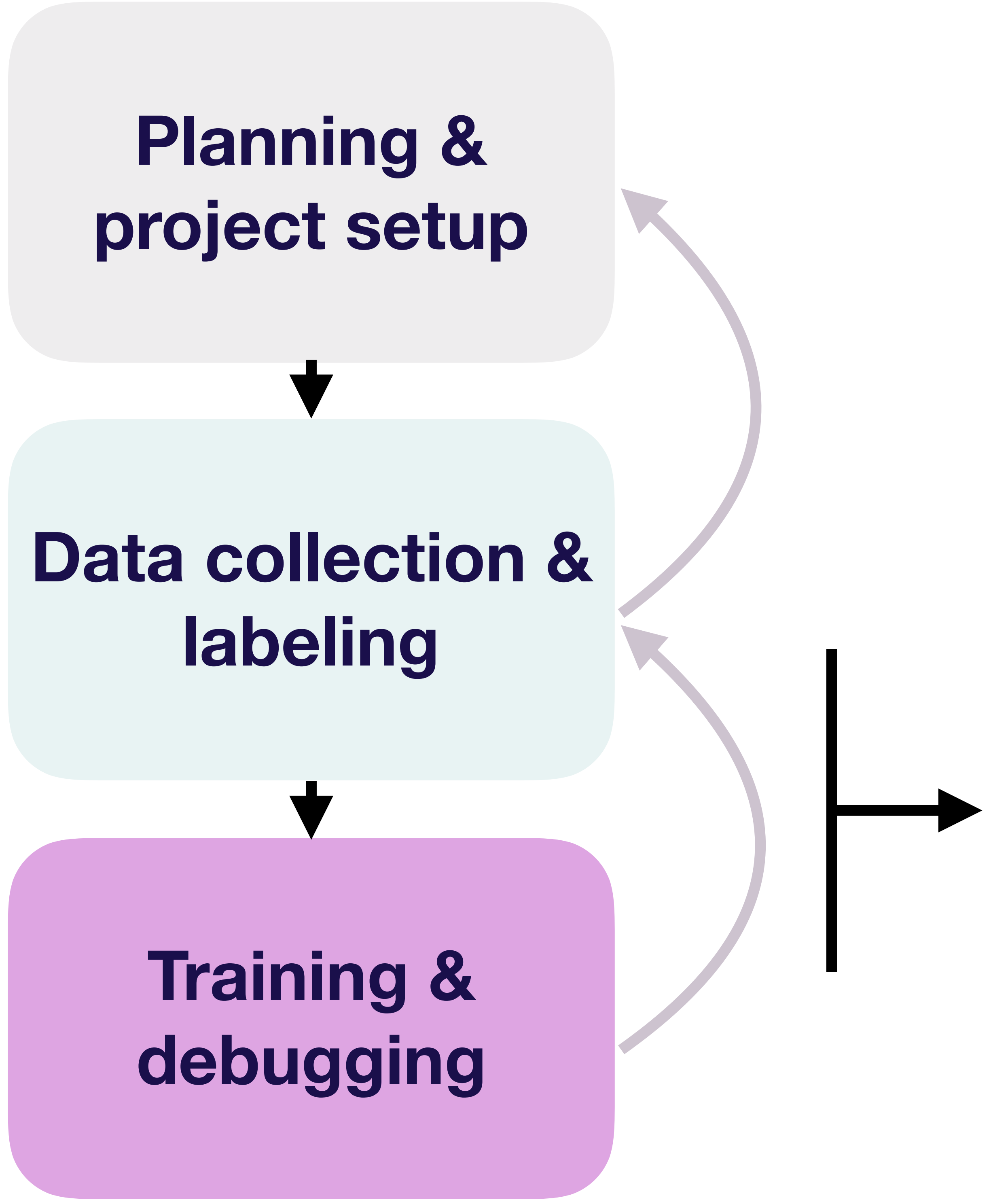


- Implement baseline in OpenCV
- Find SoTA model & reproduce
- Debug our implementation
- Improve model for our task

Lifecycle of a ML project

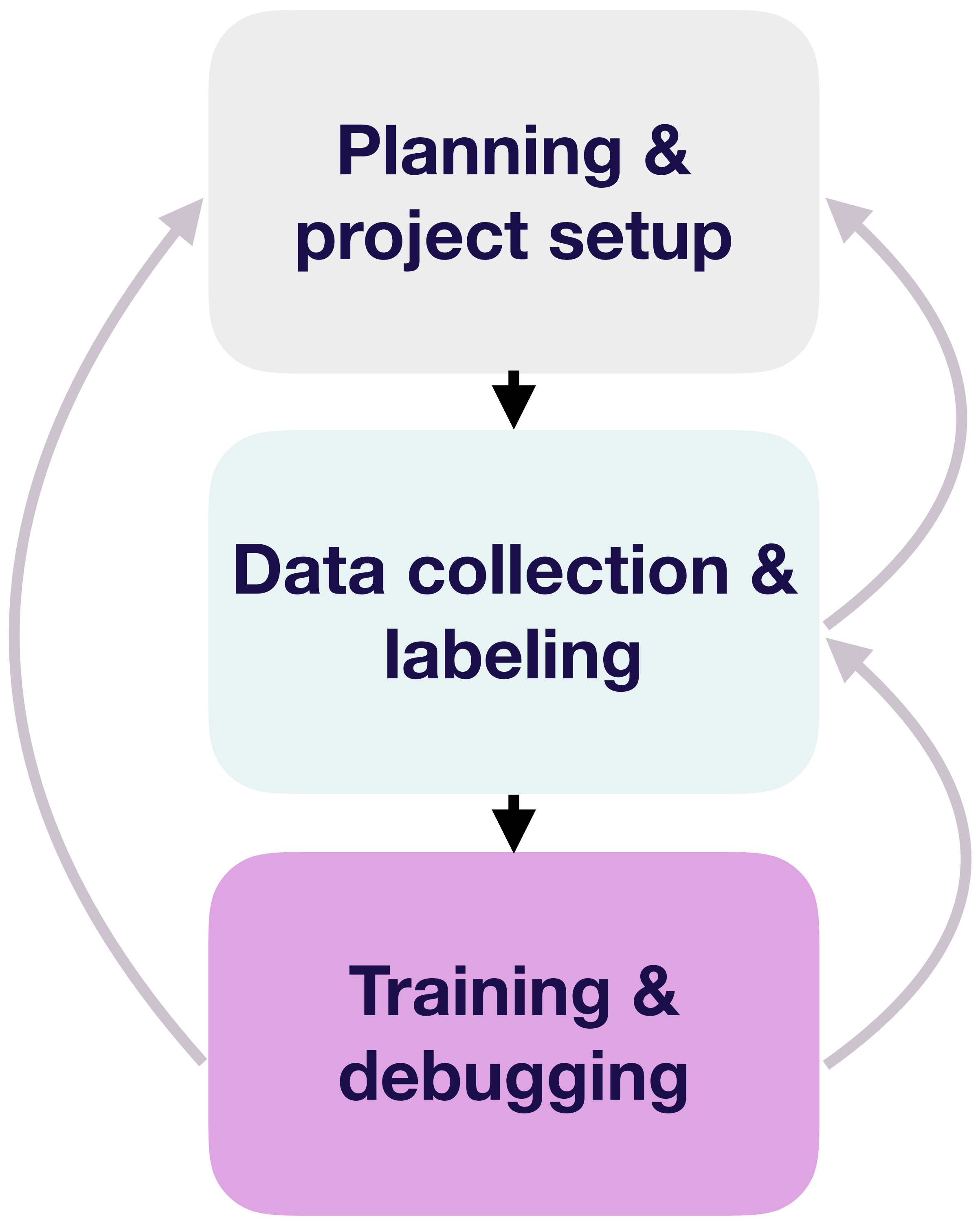


Lifecycle of a ML project



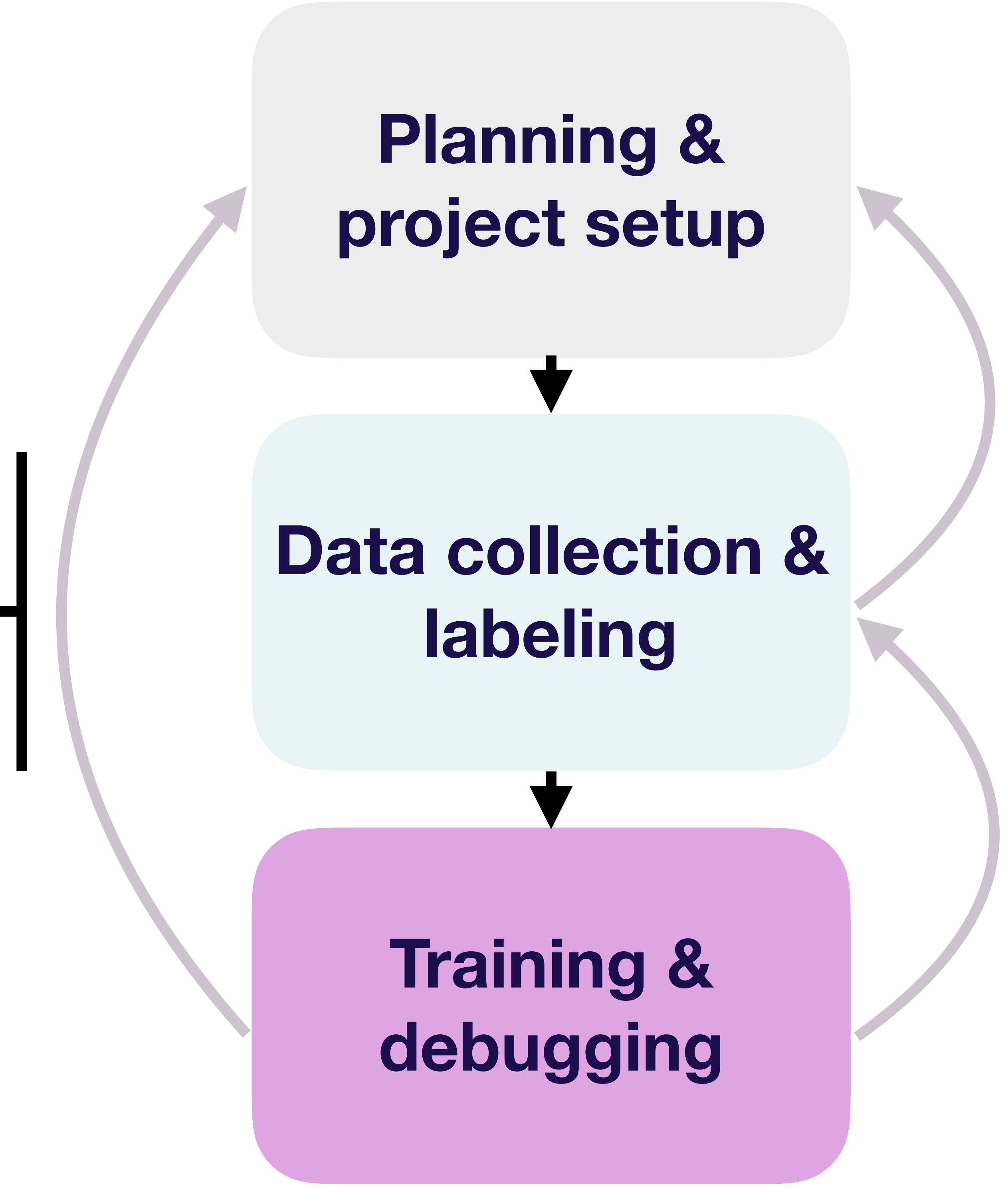
- Collect more data
- Realize data labeling is unreliable

Lifecycle of a ML project

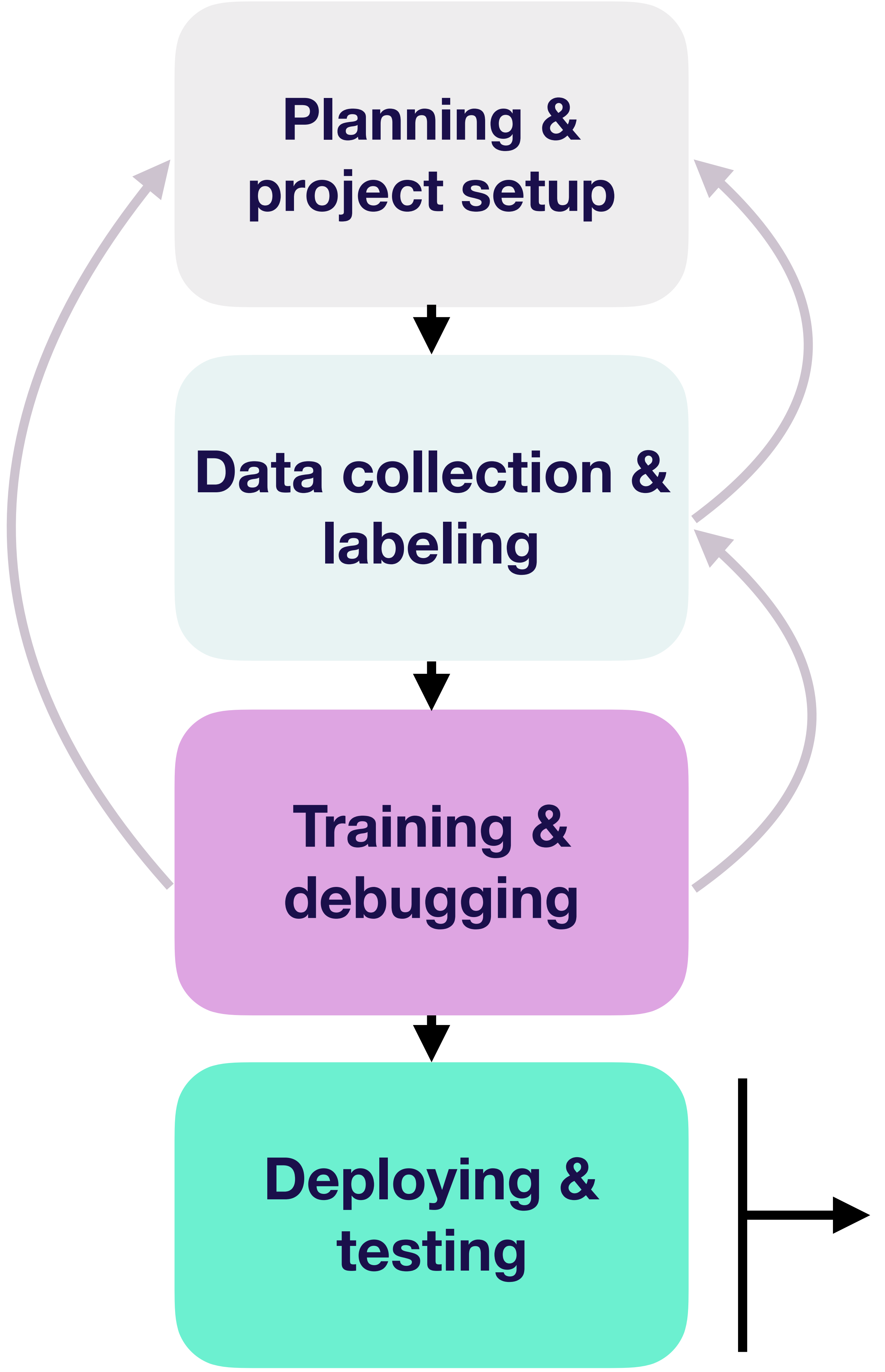


Lifecycle of a ML project

- Realize task is too hard
- Requirements trade off with each other - revisit which are most important

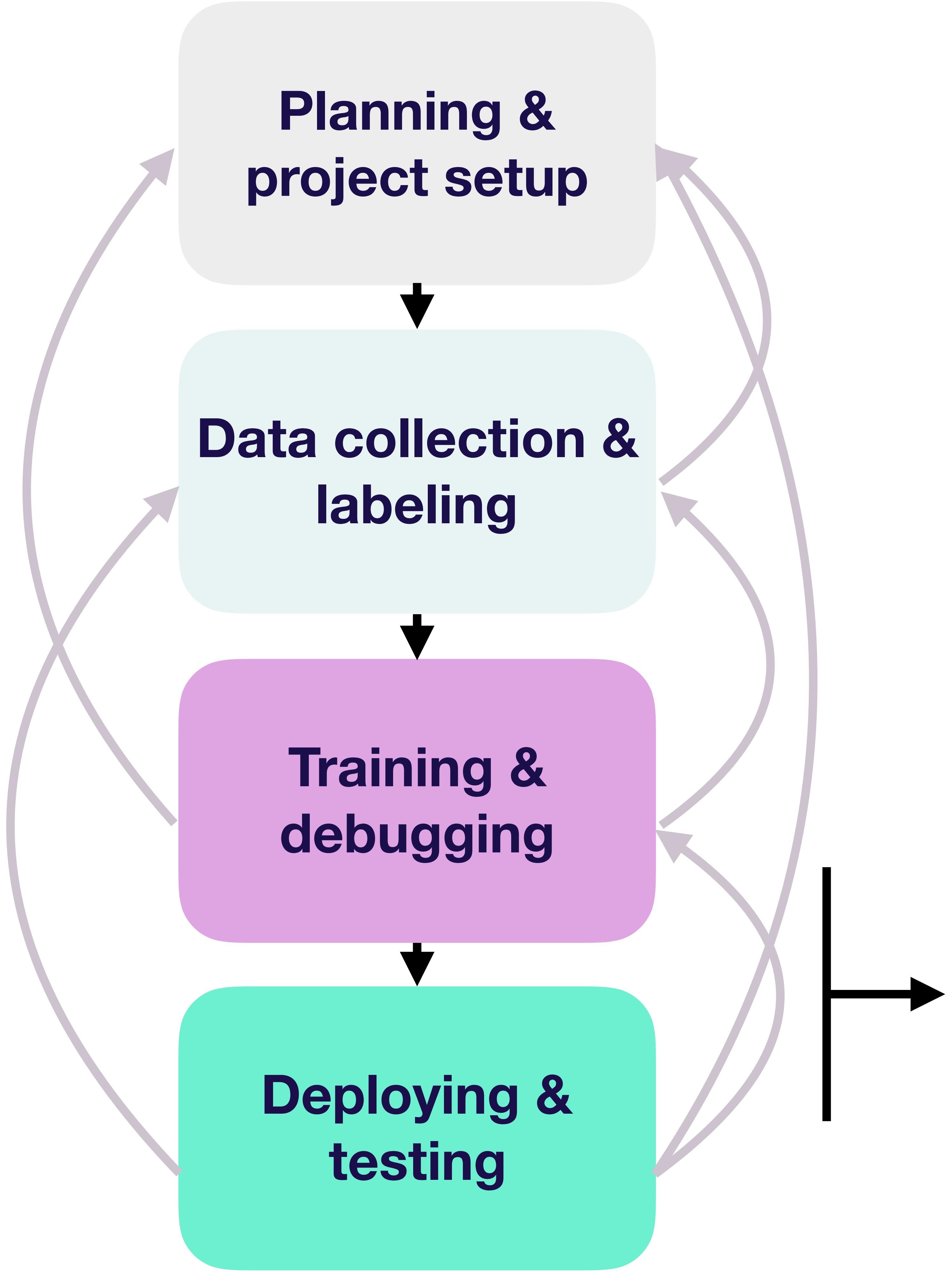


Lifecycle of a ML project



- Pilot in grasping system in the lab
- Write tests to prevent regressions
- Roll out in production

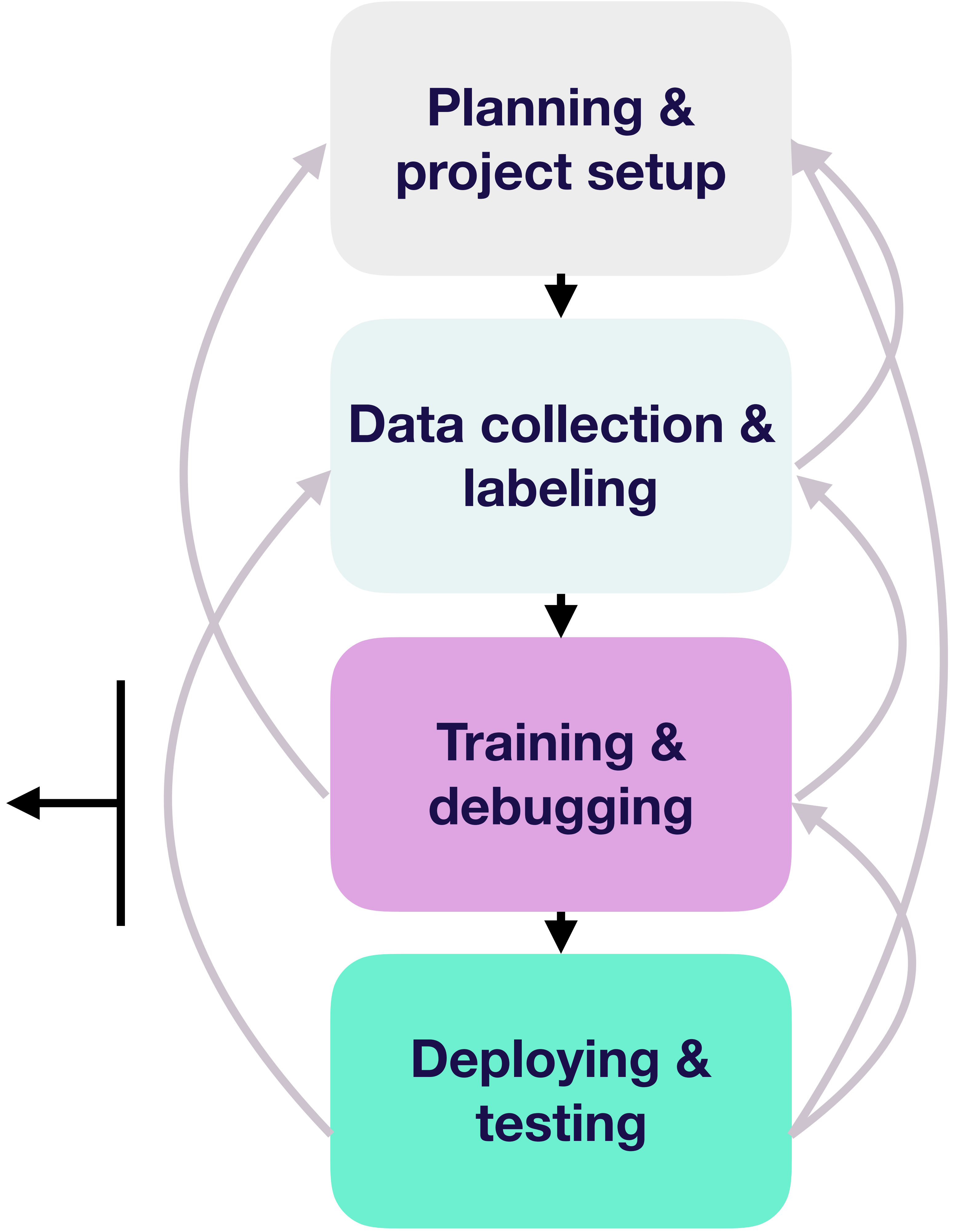
Lifecycle of a ML project



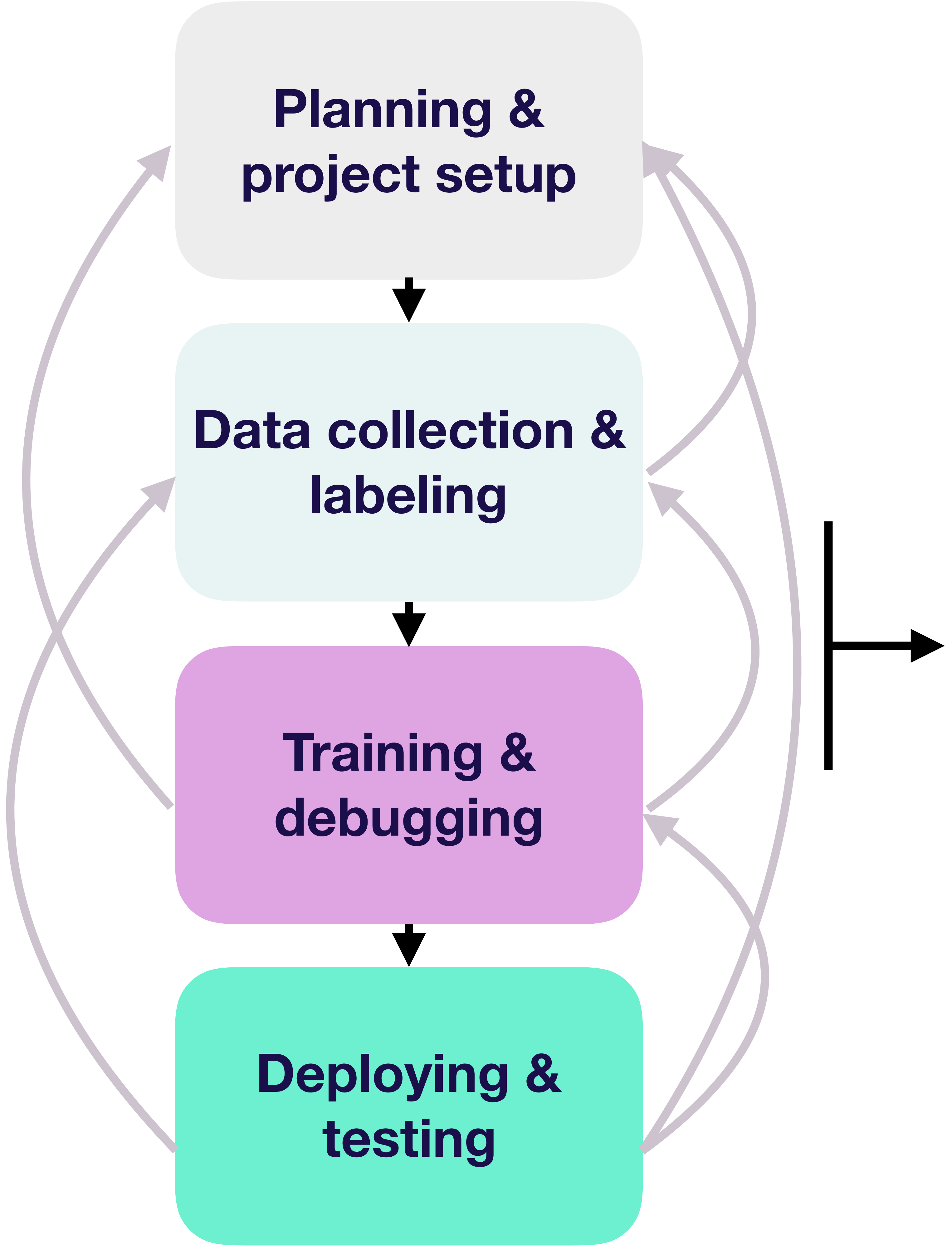
- Doesn’t work in the lab - keep improving accuracy of model

Lifecycle of a ML project

- Fix data mismatch between training data and data seen in deployment
- Collect more data
- Mine hard cases

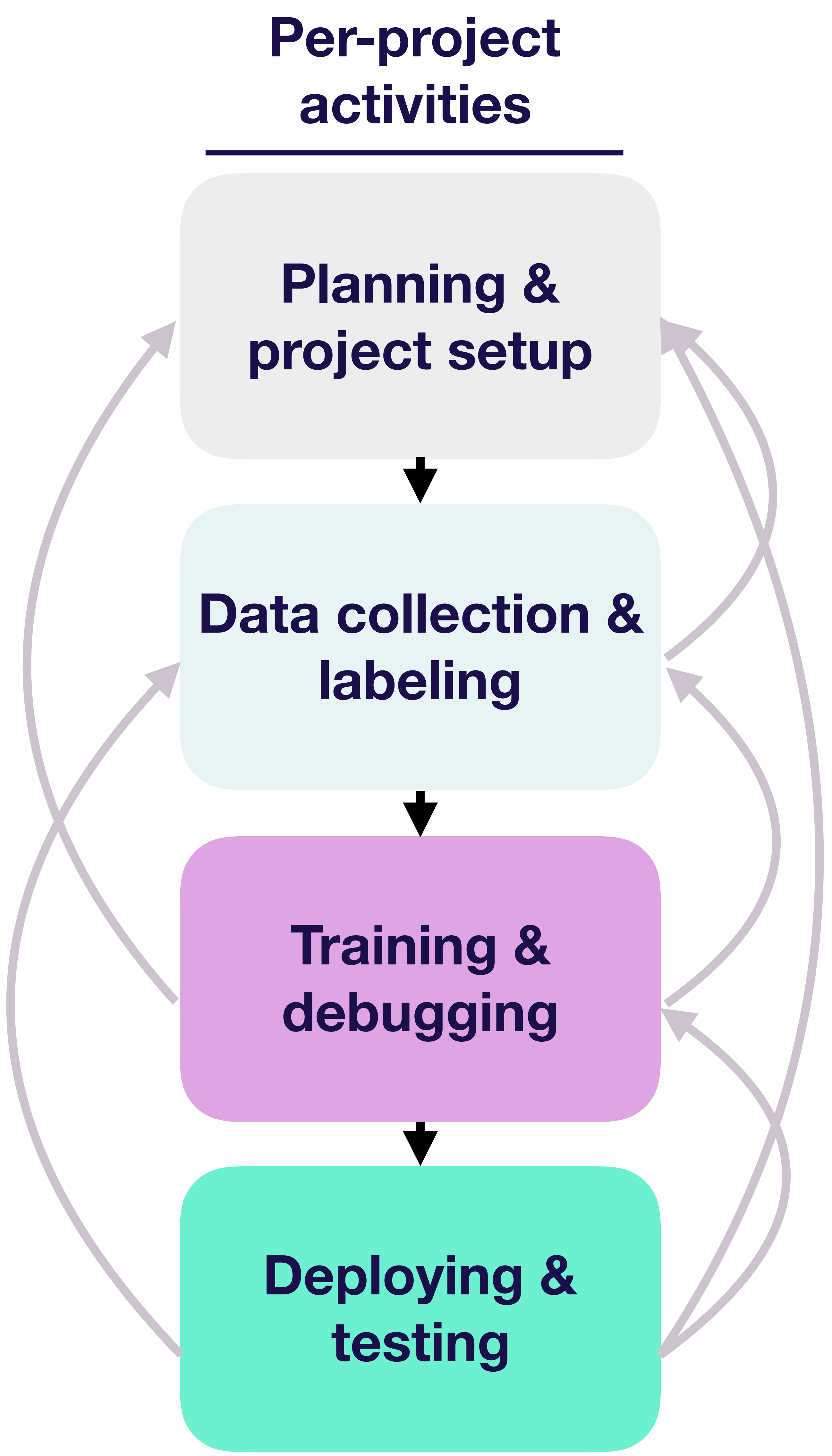


Lifecycle of a ML project

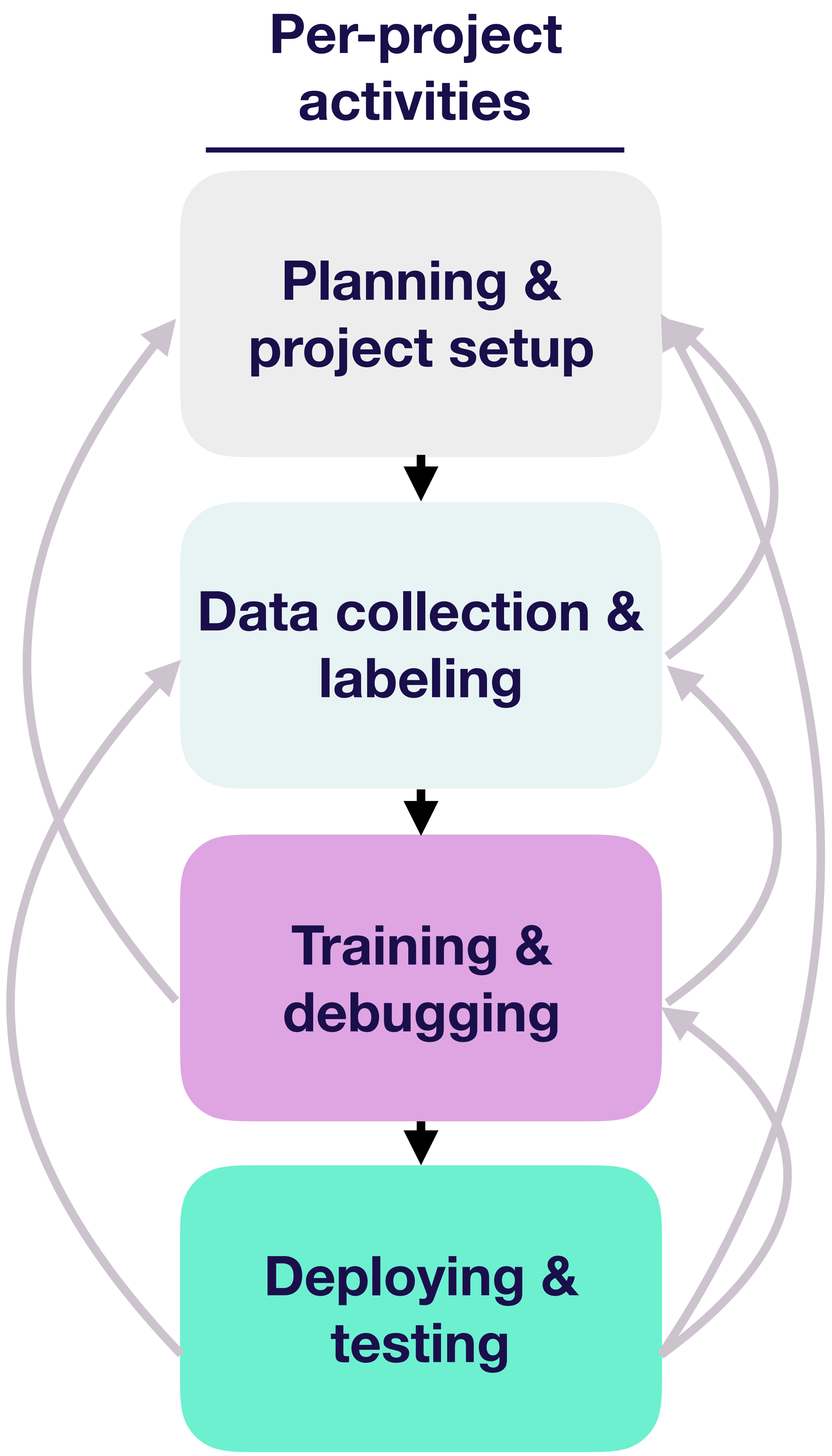


- The metric you picked doesn't actually drive downstream user behavior. Revisit the metric.
- Performance in the real world isn't great - revisit requirements (e.g., do we need to be faster or more accurate?

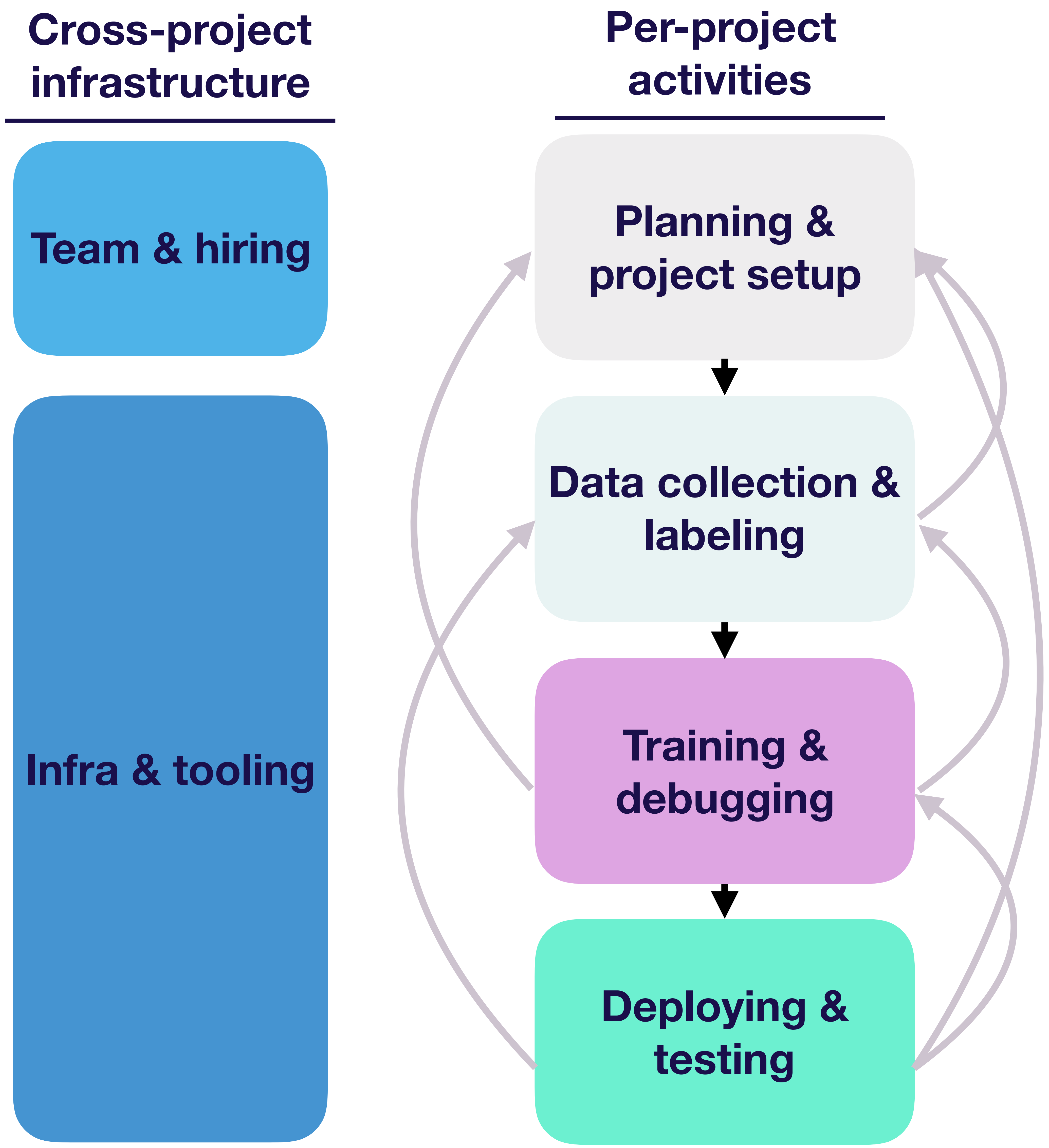
Lifecycle of a ML project



Lifecycle of a ML project



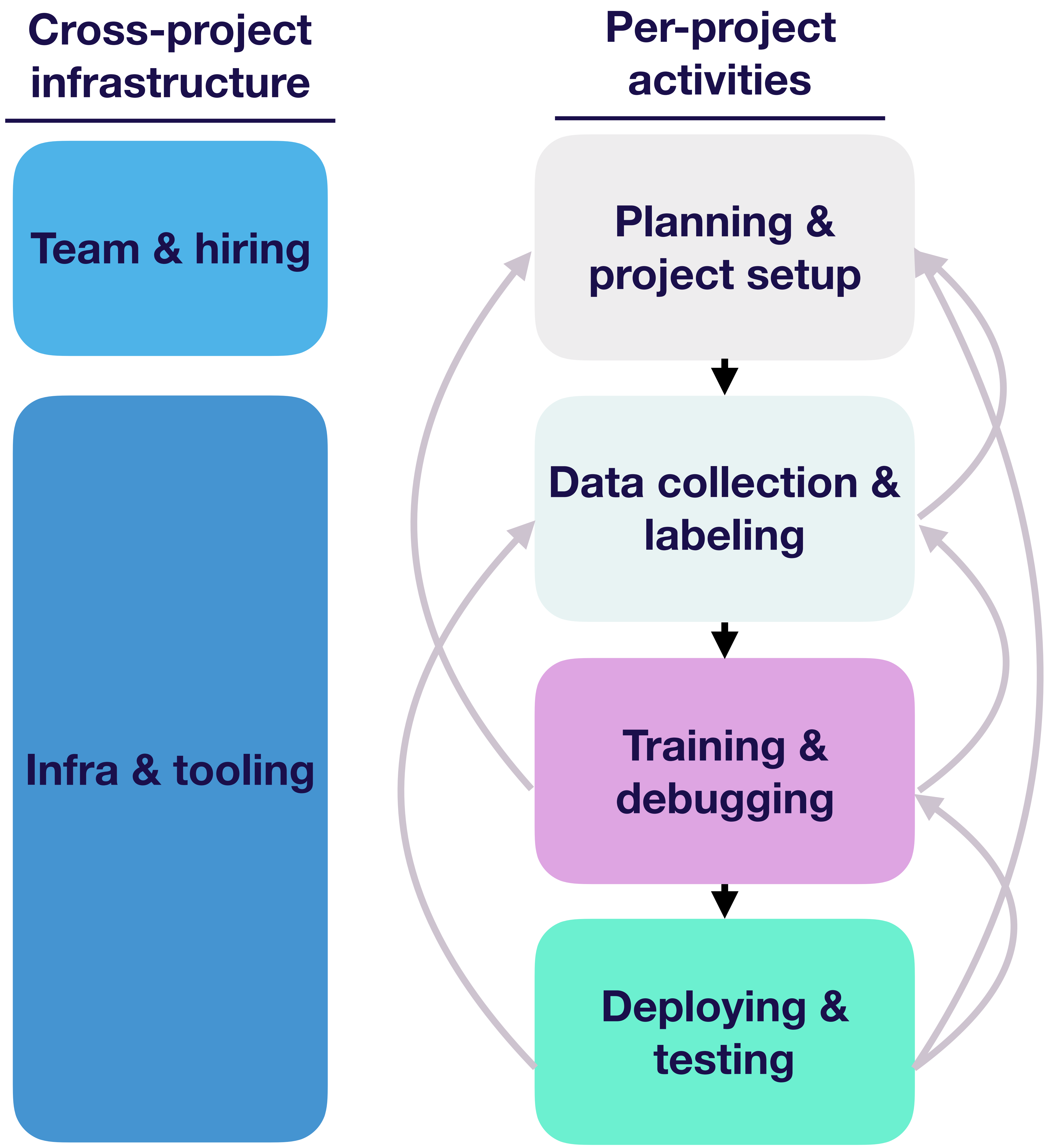
Lifecycle of a ML project



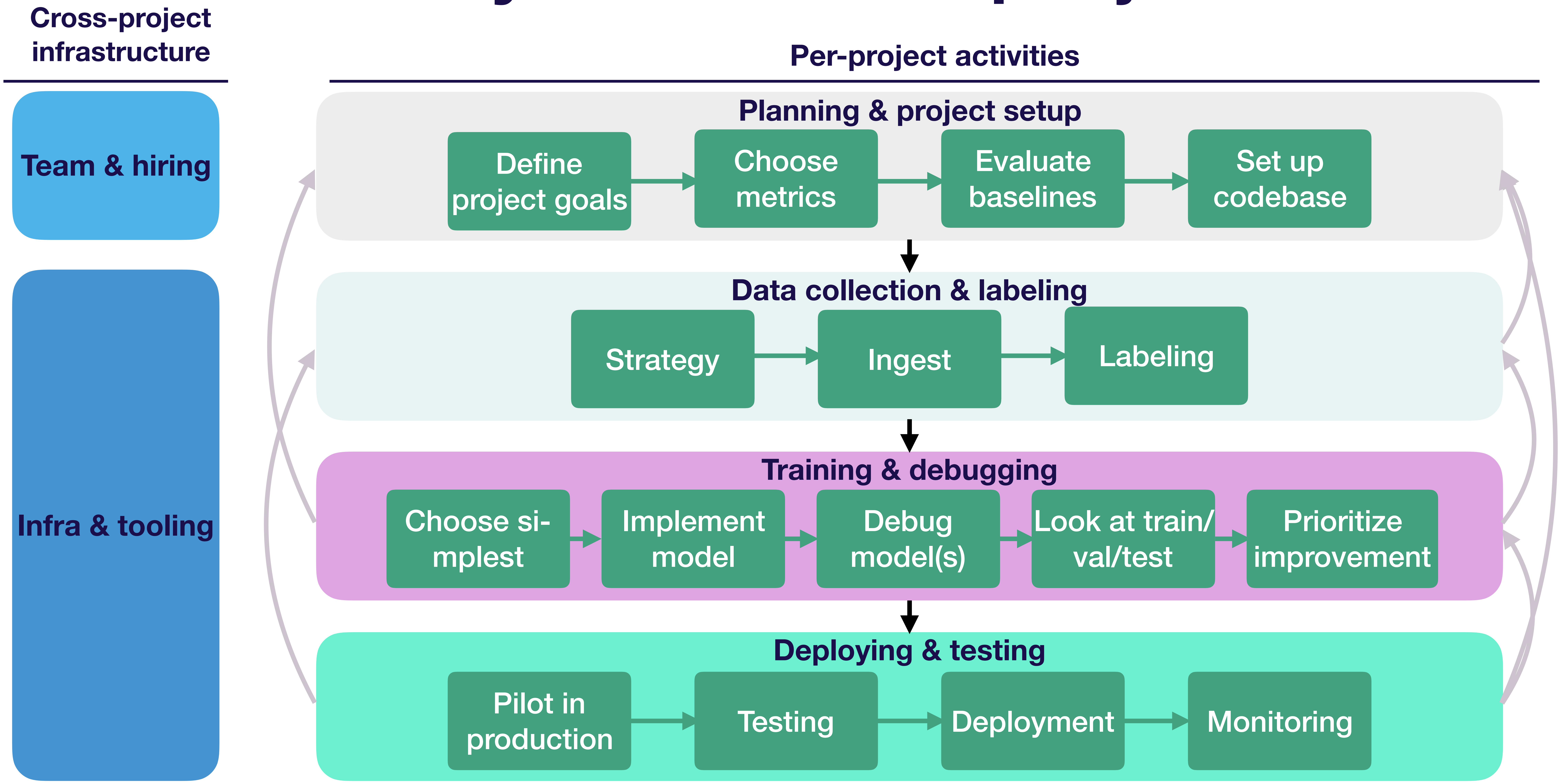
What else do you need to know?

- Understand state of the art in your domain
    - Understand what’s possible
    - Know what to try next
- We will introduce most promising research areas

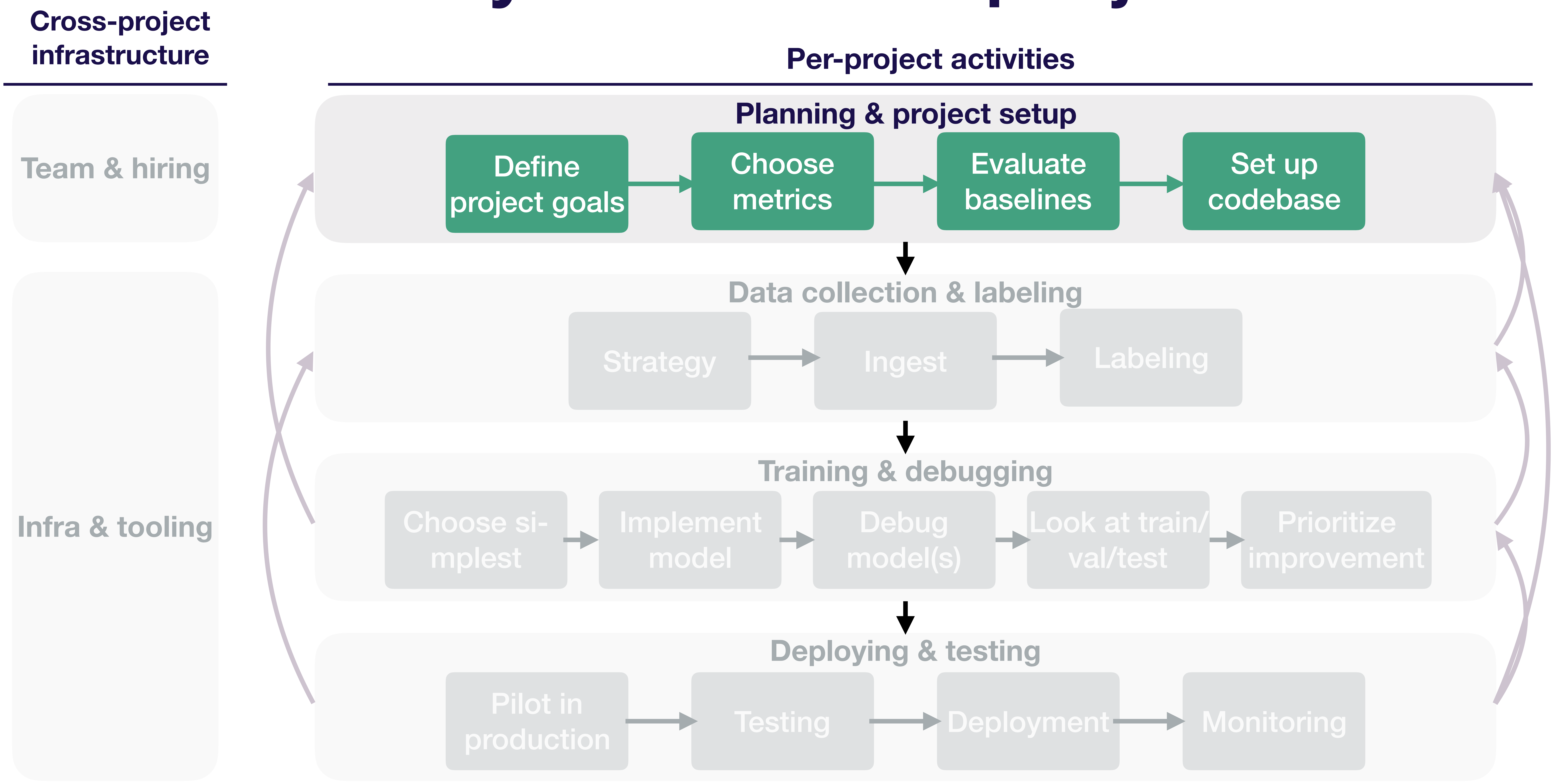
Lifecycle of a ML project



Lifecycle of a ML project



Lifecycle of a ML project



Outline of the rest of the lecture

1. Prioritizing projects & choosing goals
2. Choosing metrics
3. Choosing baselines

Outline of the rest of the lecture

1. **Prioritizing projects & choosing goals**
2. Choosing metrics
3. Choosing baselines

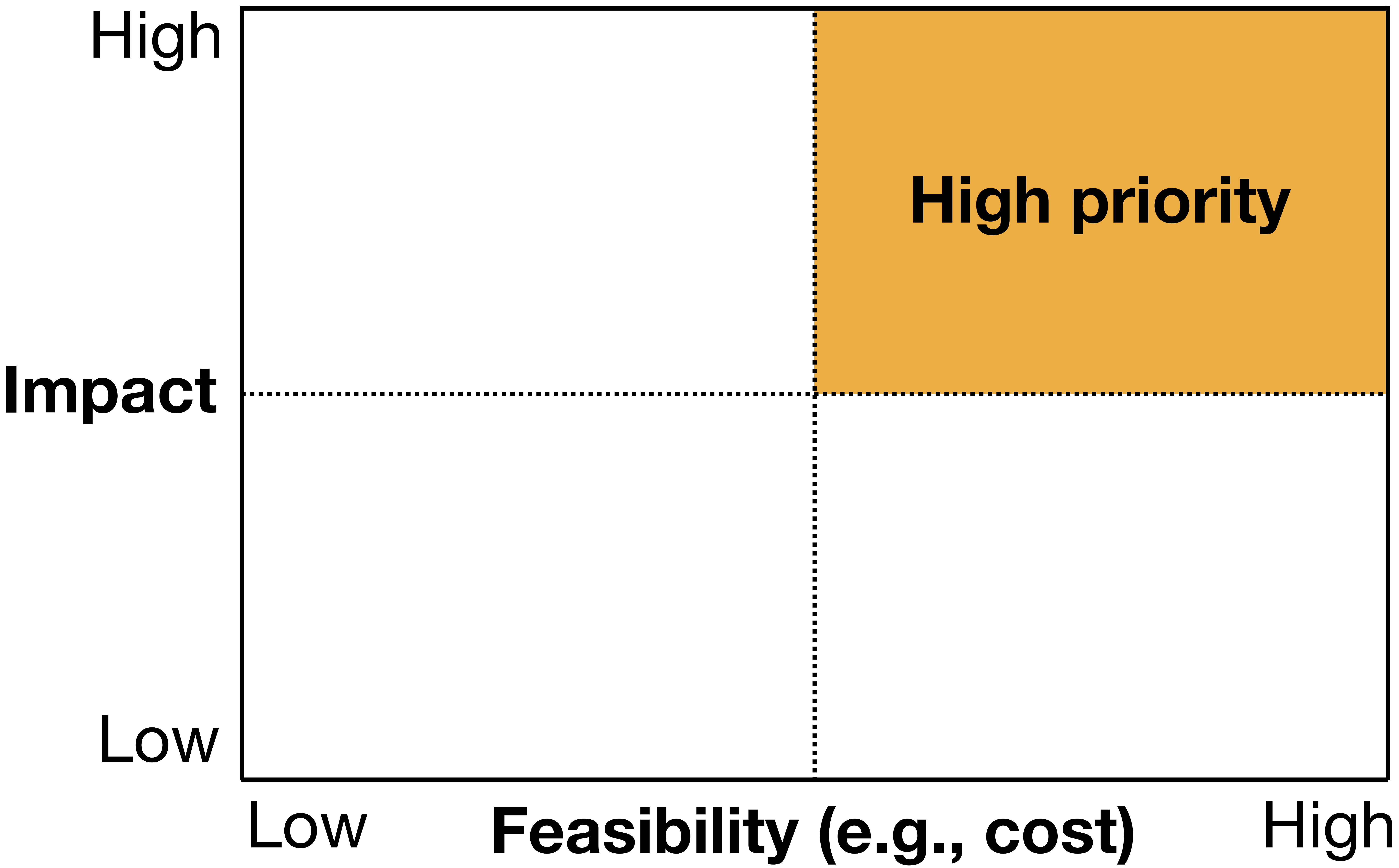
Key points for prioritizing projects

A. High-impact ML problems

- Complex parts of your pipeline
- Places where cheap prediction is valuable

B. Cost of ML projects is driven by data availability, but accuracy requirement also plays a big role

A (general) framework for prioritizing projects



Mental models for high-impact ML projects

1. Where can you take advantage of cheap prediction?

2. Where can you automate complicated manual software pipelines?

Mental models for high-impact ML projects

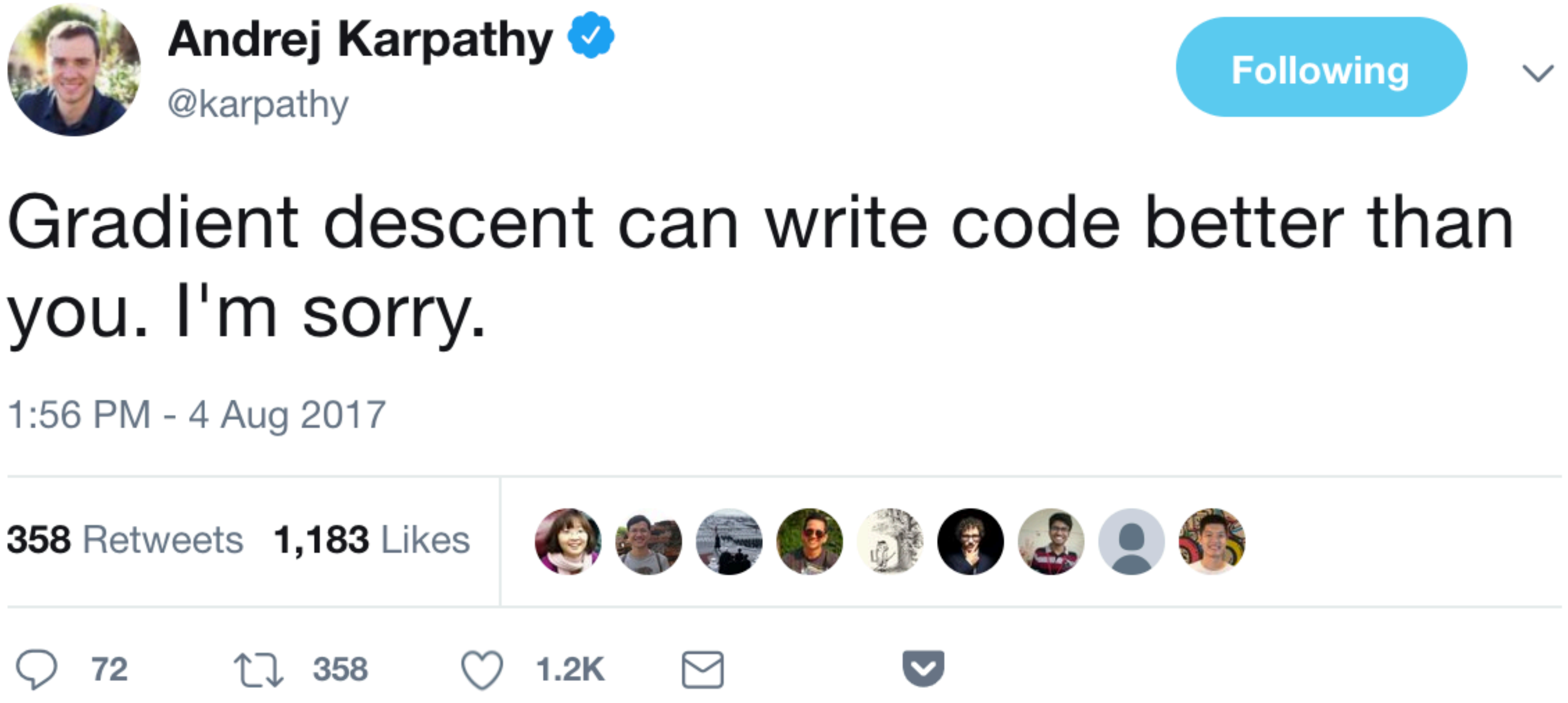
The economics of AI
(Agrawal, Gans, Goldfarb)

- AI reduces cost of prediction
- Prediction is central for decision making
- Cheap prediction means
  - Prediction will be everywhere
  - Even in problems where it was too expensive before (e.g., for most people, hiring a driver)
- **Implication:** Look for projects where cheap prediction will have a huge business impact

Prediction Machines: The Simple Economics of Artificial Intelligence (Agrawal, Gans, Goldfarb)

Mental models for high-impact ML projects

Software 2.0
(Andrej Karpathy)



Software 2.0 (Andrej Karpathy): https://medium.com/@karpathy/software-2-0-a64152b37c35

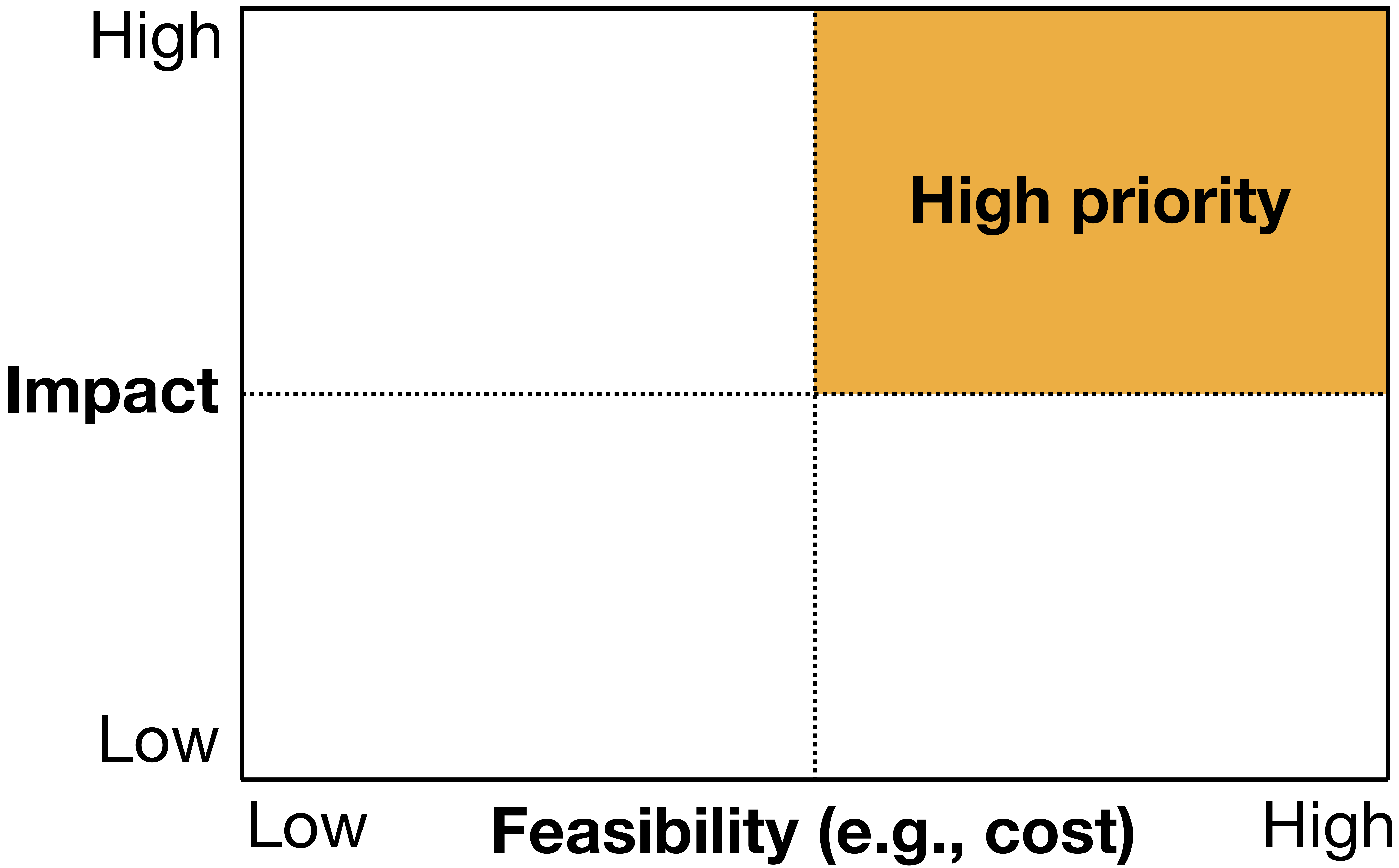
Mental models for high-impact ML projects

Software 2.0
(Andrej Karpathy)

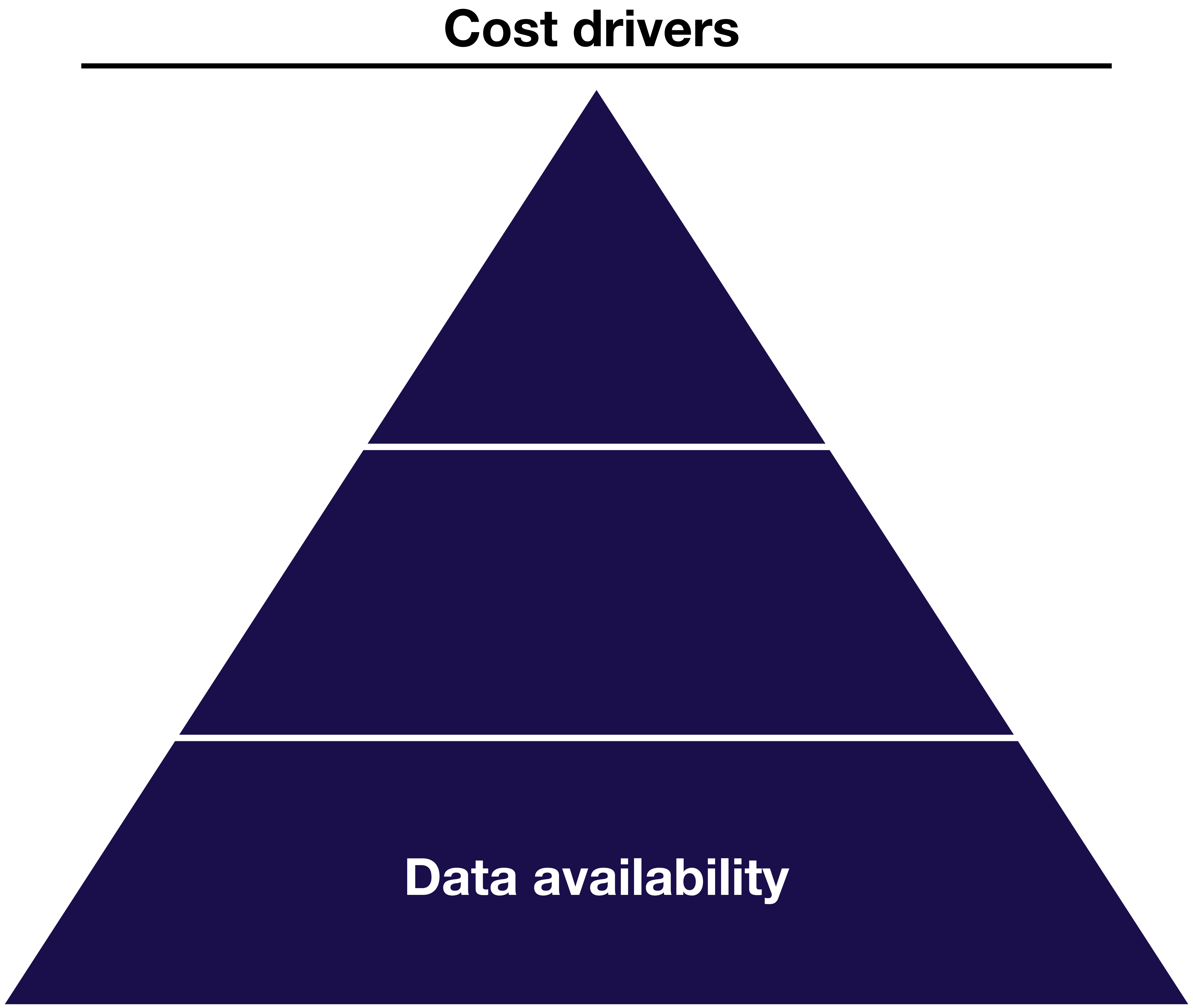
- *Software 1.0* = traditional programs with explicit instructions (python / c++ / etc)
- Software 2.0 = humans specify goals, and algorithm searches for a program that works
- 2.0 programmers work with datasets, which get compiled via optimization
- Why? Works better, more general, computational advantages
- **Implication:** look for complicated rule-based software where we can learn the rules instead of programming them

Software 2.0 (Andrej Karpathy): https://medium.com/@karpathy/software-2-0-a64152b37c35

A (general) framework for prioritizing projects

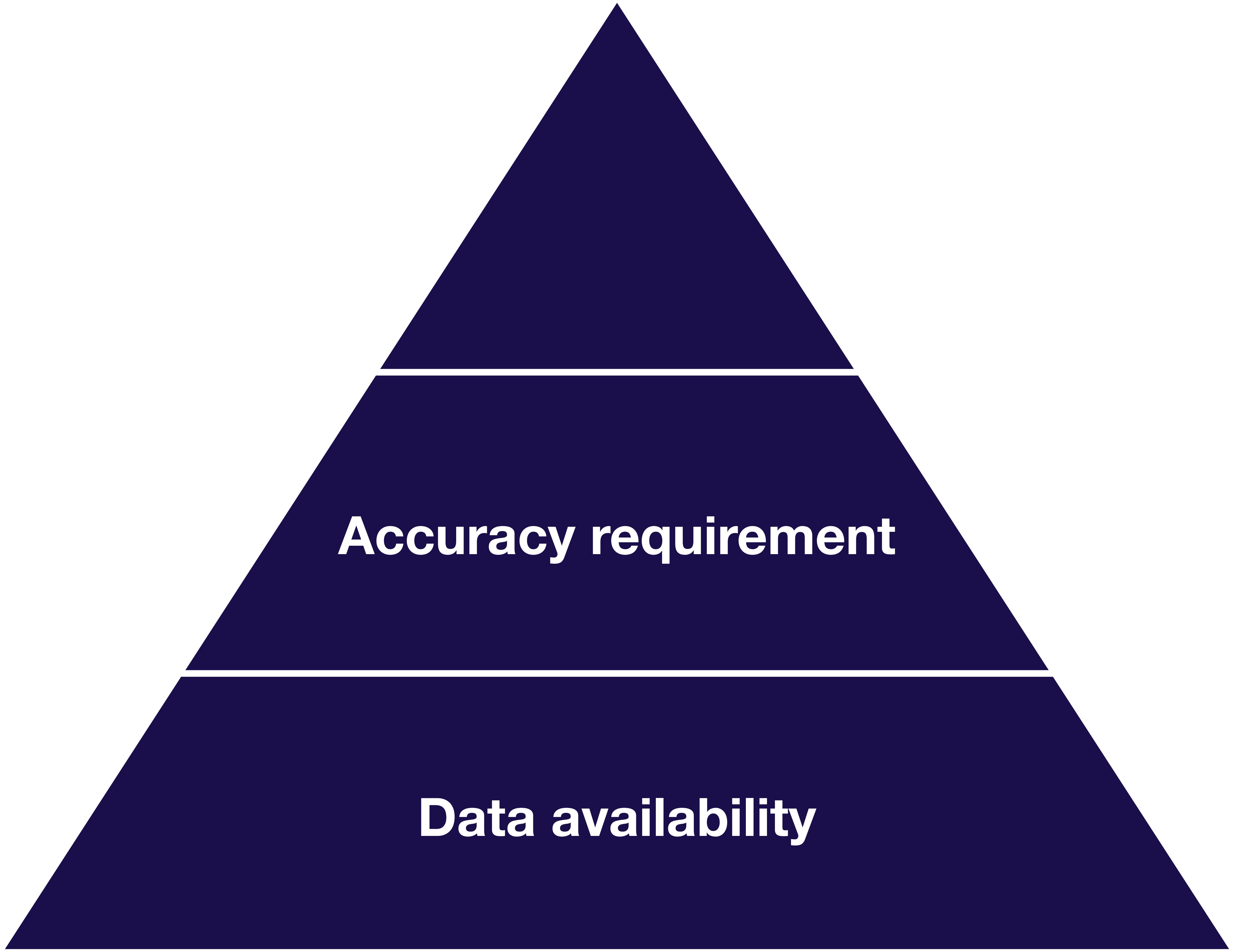


Assessing feasibility of ML projects



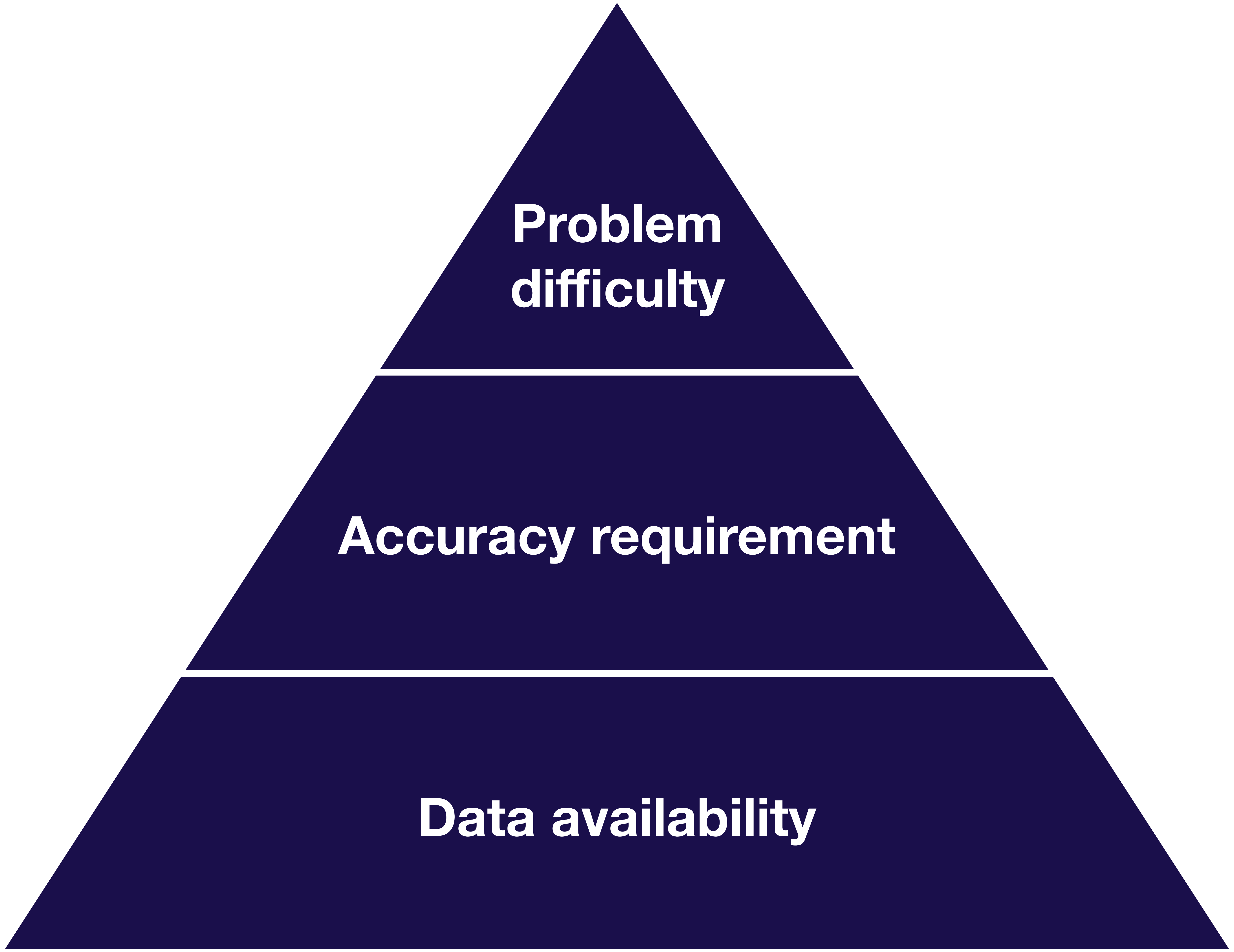
Assessing feasibility of ML projects

Cost drivers



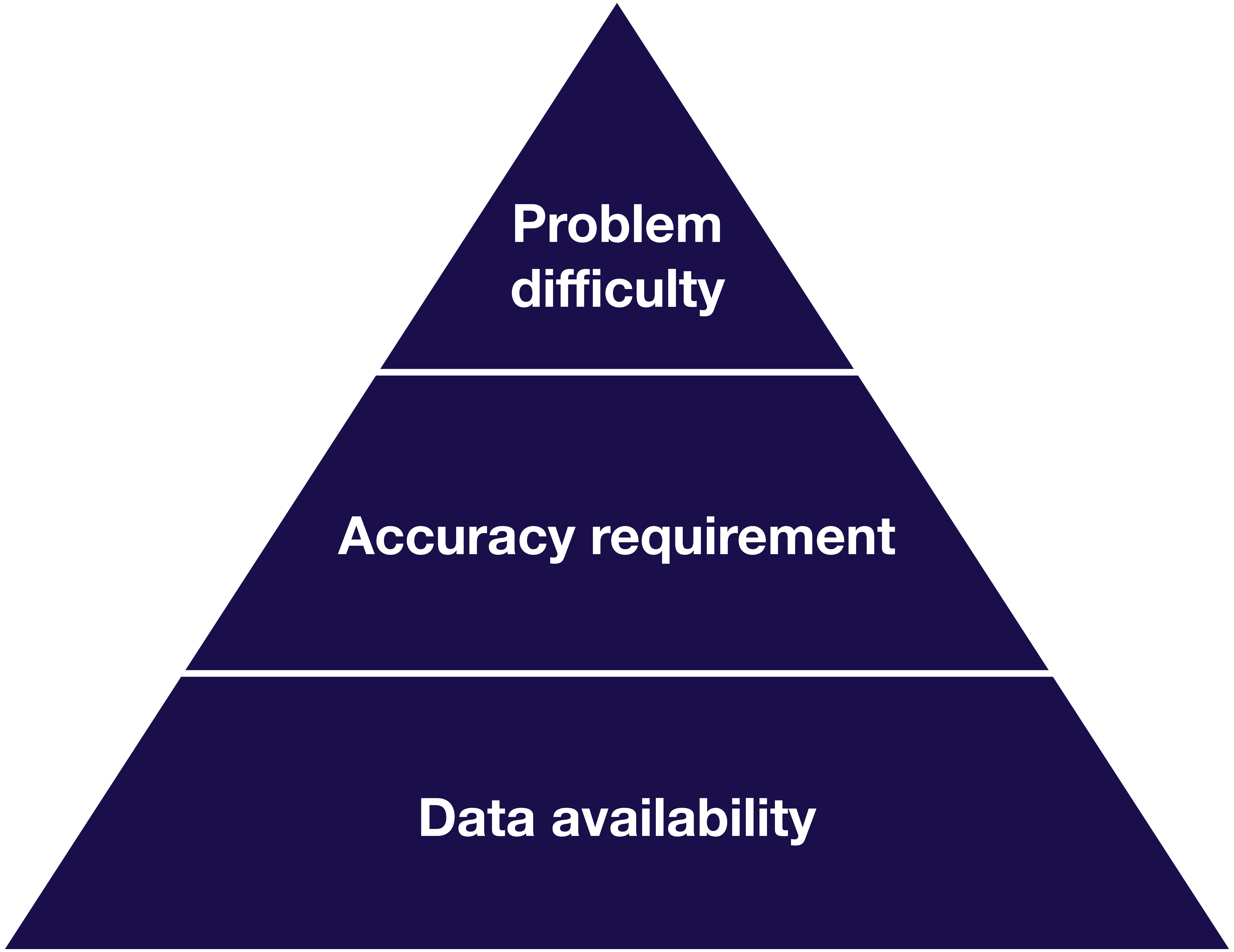
Assessing feasibility of ML projects

Cost drivers



Assessing feasibility of ML projects

Cost drivers

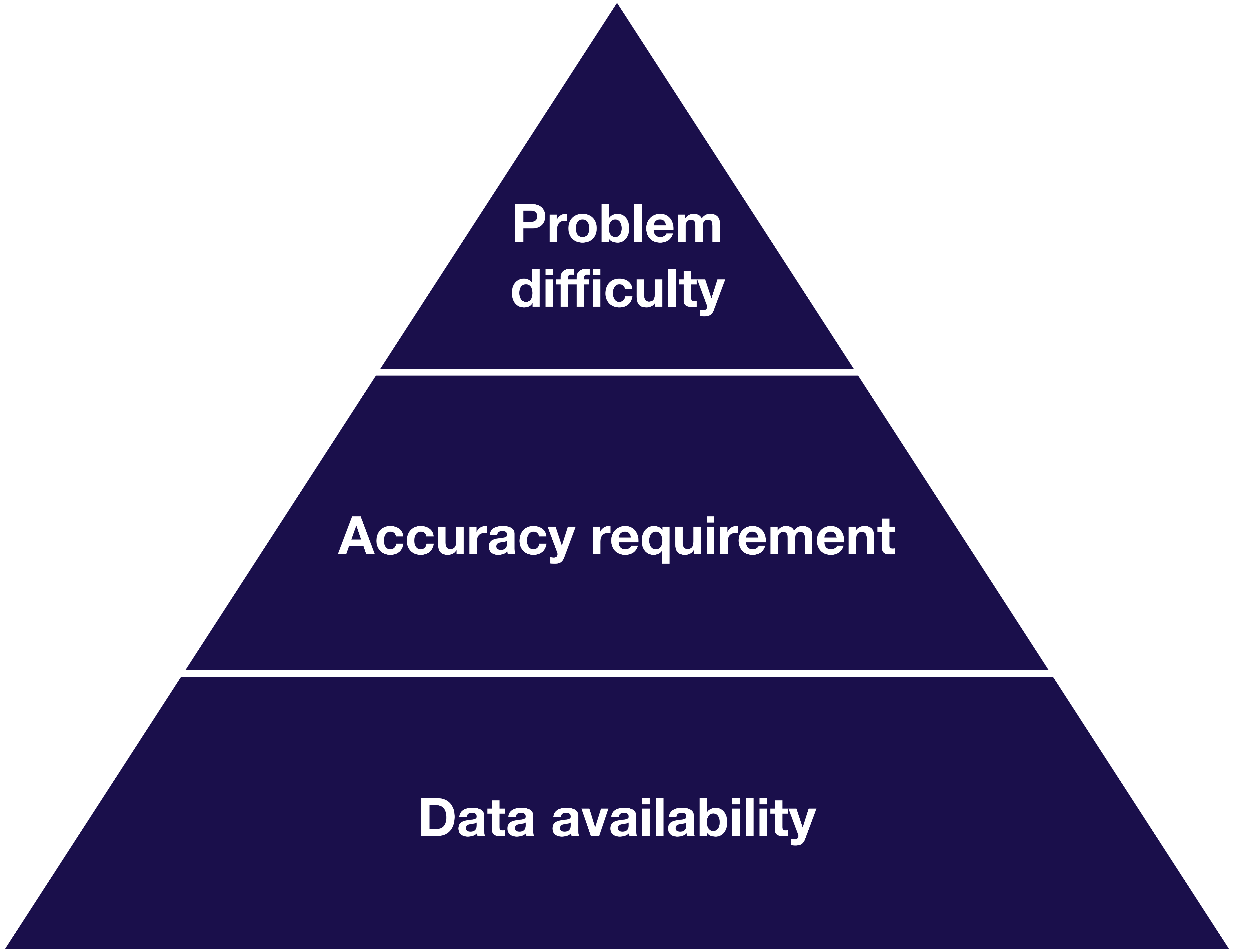


Main considerations

- How hard is it to acquire data?
- How expensive is data labeling?
- How much data will be needed?

Assessing feasibility of ML projects

Cost drivers



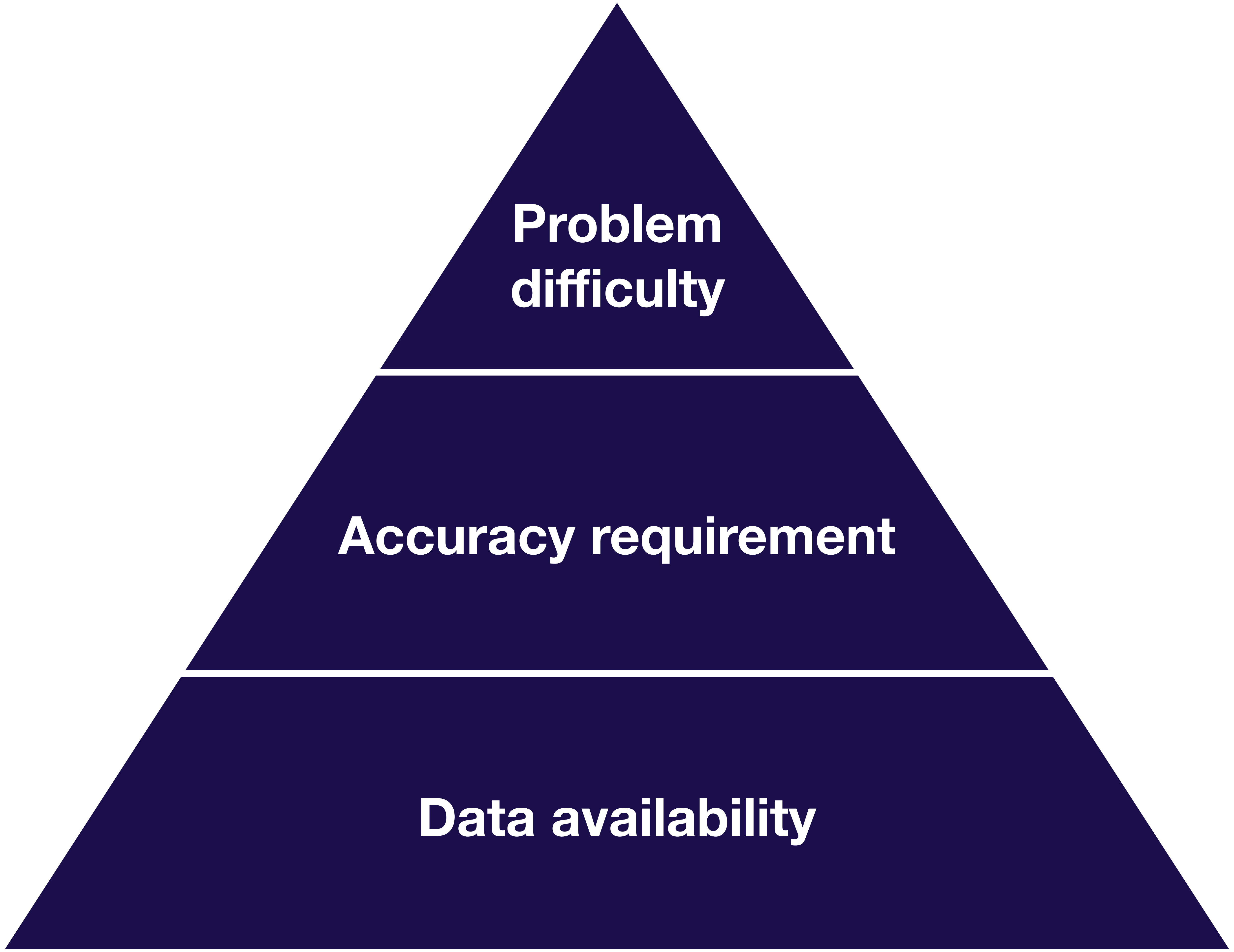
Main considerations

- How costly are wrong predictions?
- How frequently does the system need to be right to be useful?

- How hard is it to acquire data?
- How expensive is data labeling?
- How much data will be needed?

Assessing feasibility of ML projects

Cost drivers



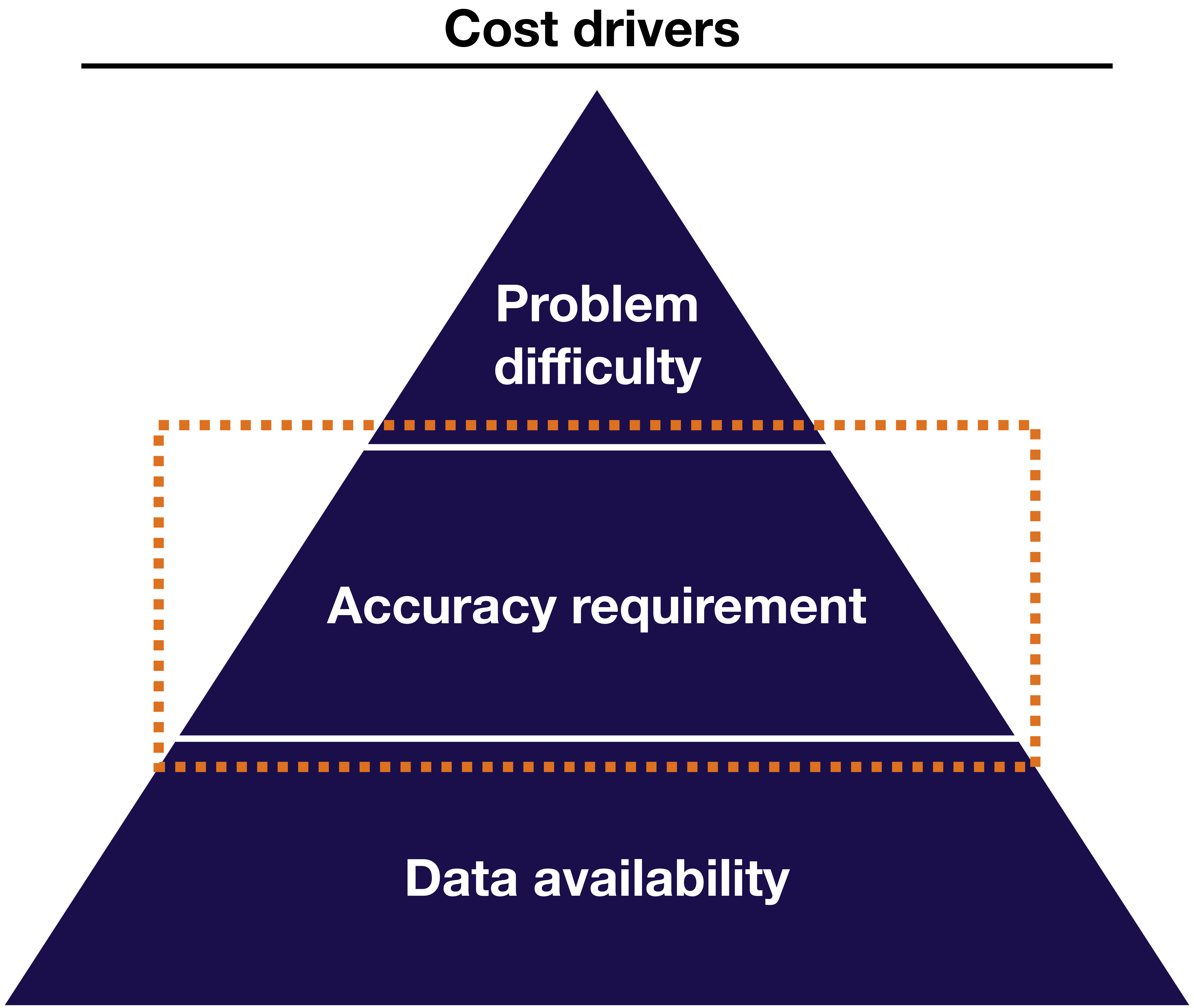
Main considerations

- Good published work on similar problems? (newer problems mean more risk & more technical effort)
- Compute needed for training?
- Compute available for deployment?

- How costly are wrong predictions?
- How frequently does the system need to be right to be useful?

- How hard is it to acquire data?
- How expensive is data labeling?
- How much data will be needed?

Assessing feasibility of ML projects



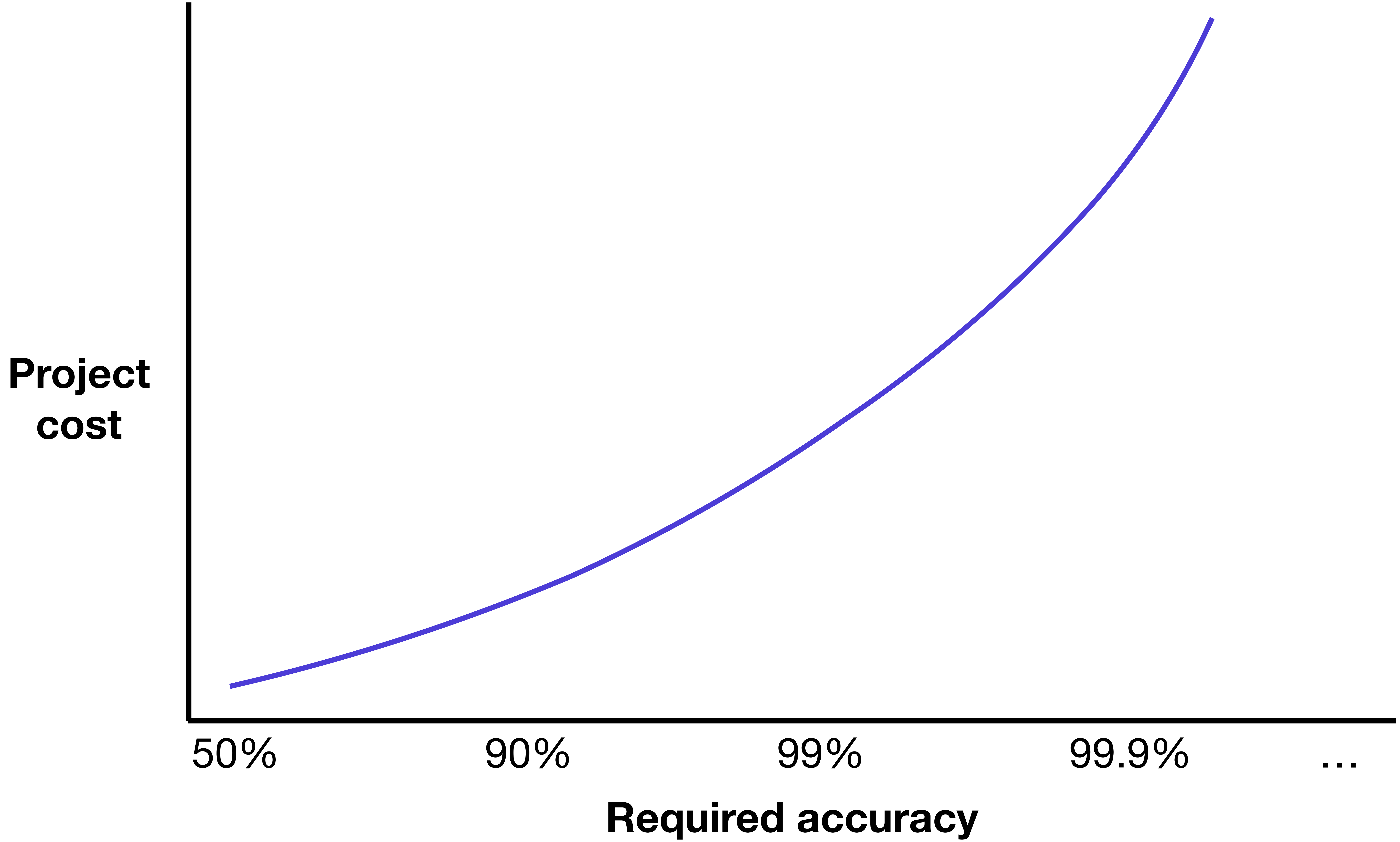
Main considerations

- Good published work on similar problems? (newer problems mean more risk & more technical effort)
- Compute needed for training?
- Compute available for deployment?

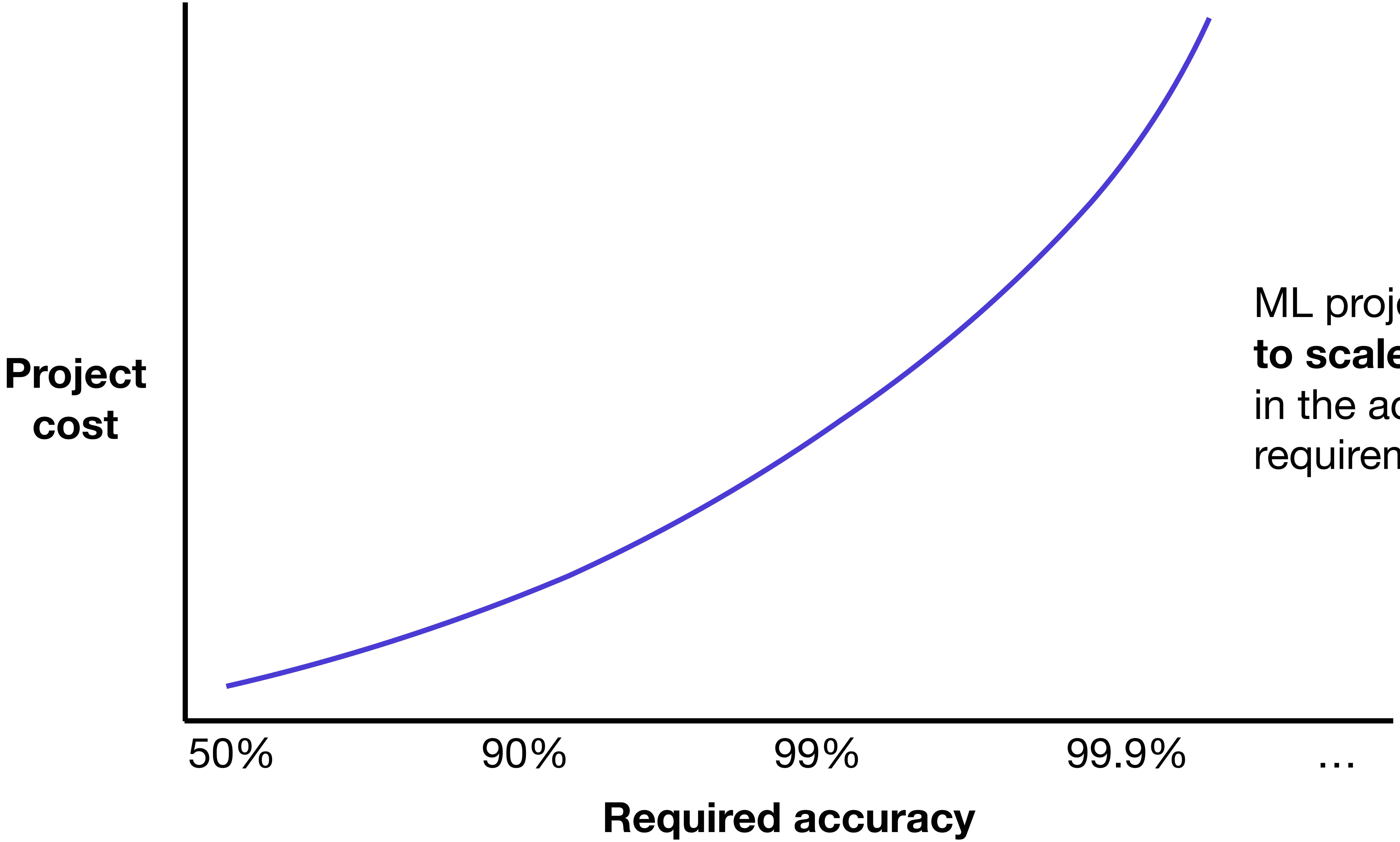
- How costly are wrong predictions?
- How frequently does the system need to be right to be useful?

- How hard is it to acquire data?
- How expensive is data labeling?
- How much data will be needed?

Why are accuracy requirements so important?

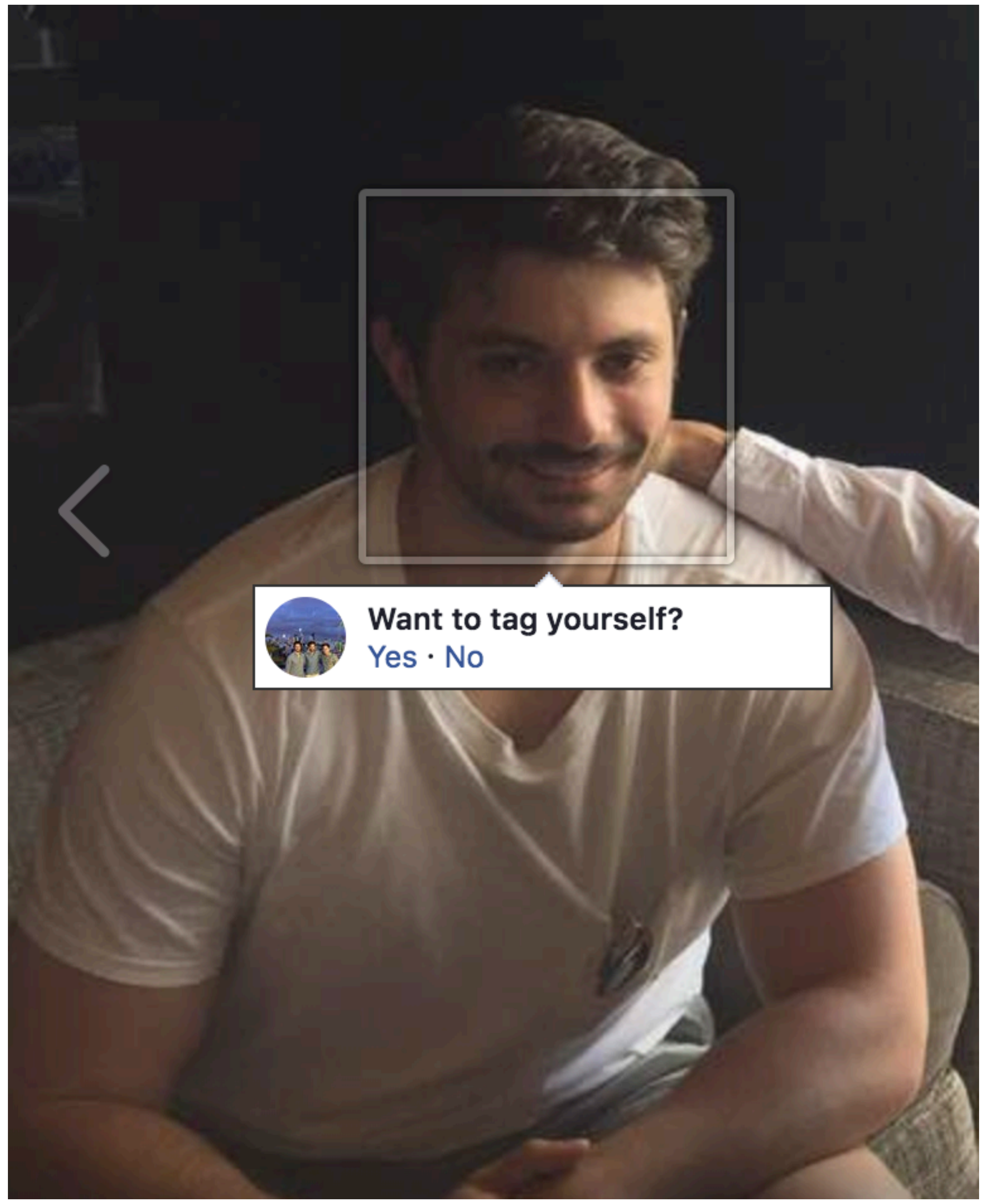


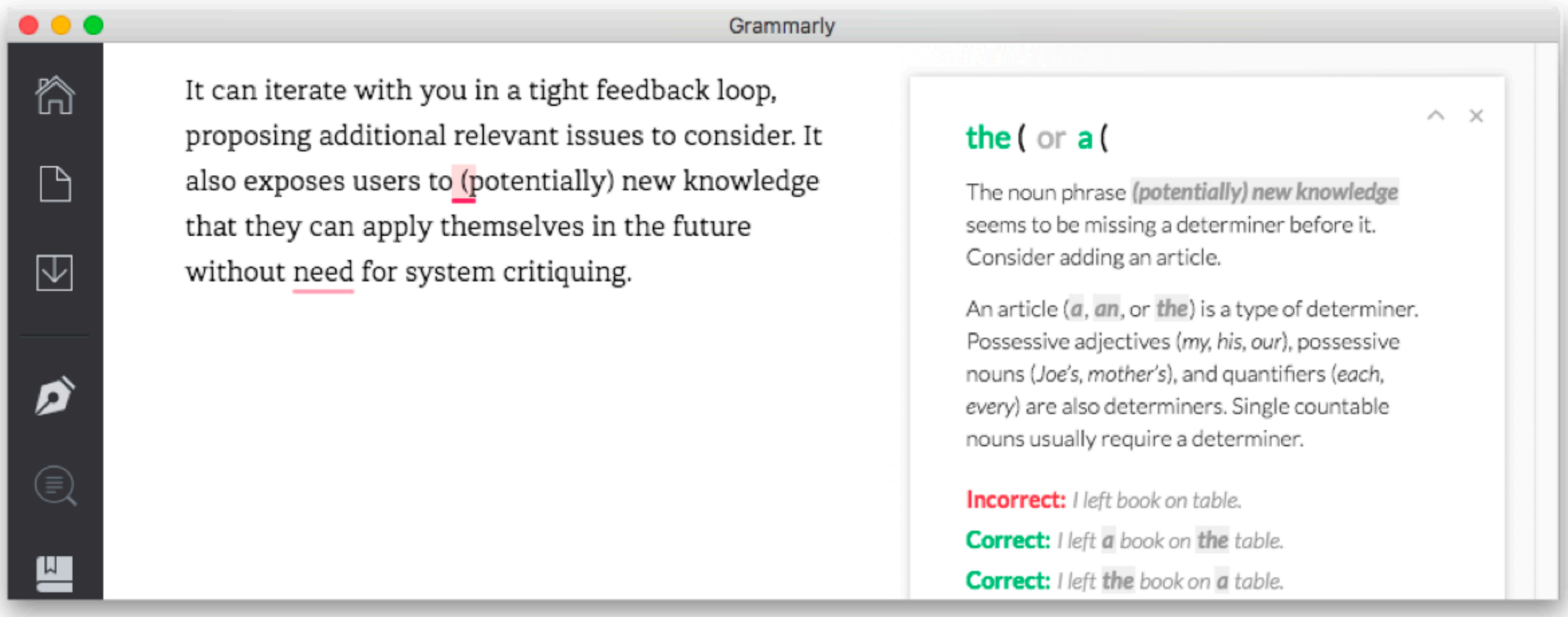
Why are accuracy requirements so important?

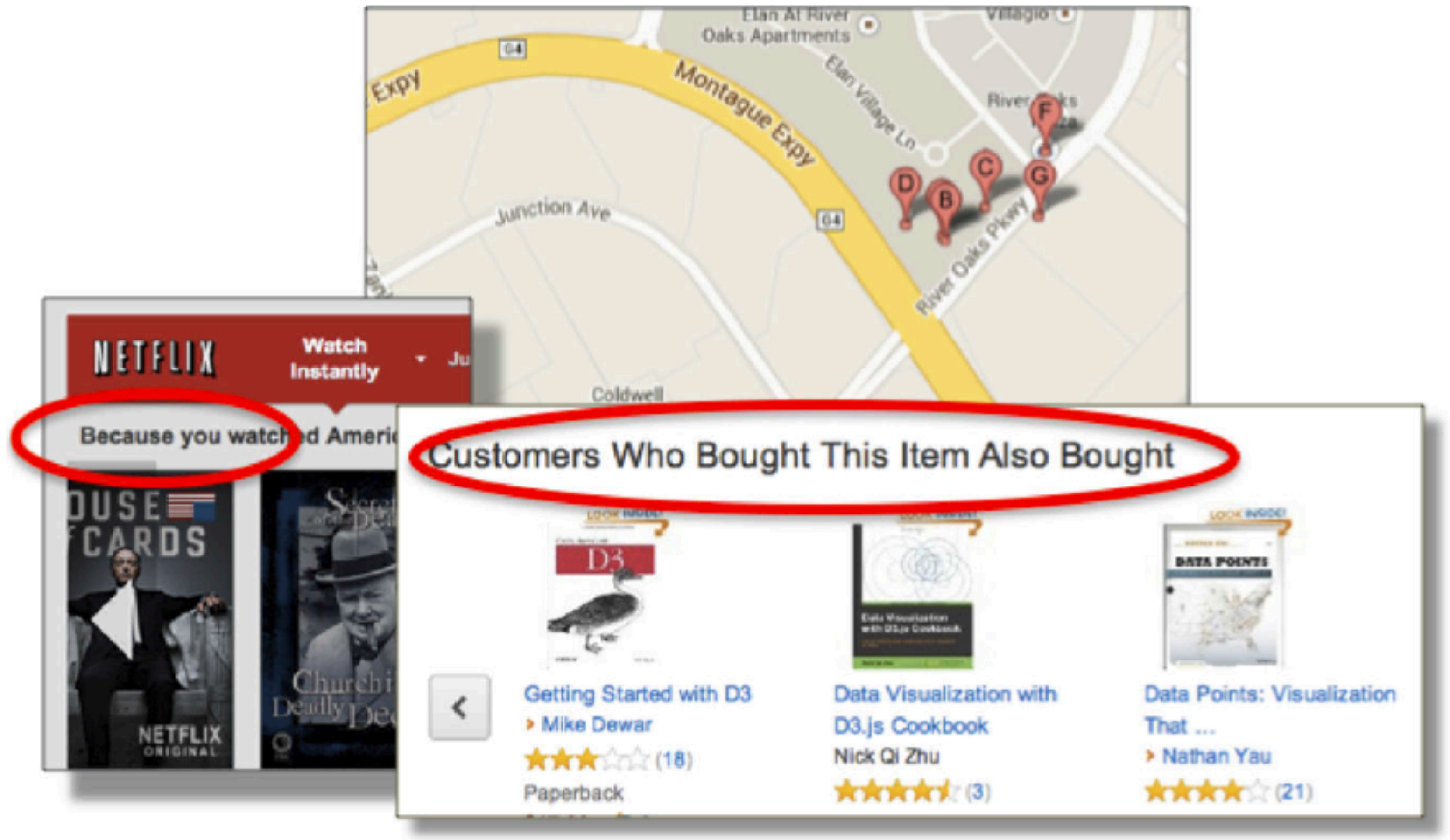


ML project **costs tend to scale super-linearly** in the accuracy requirement

Product design can reduce need for accuracy

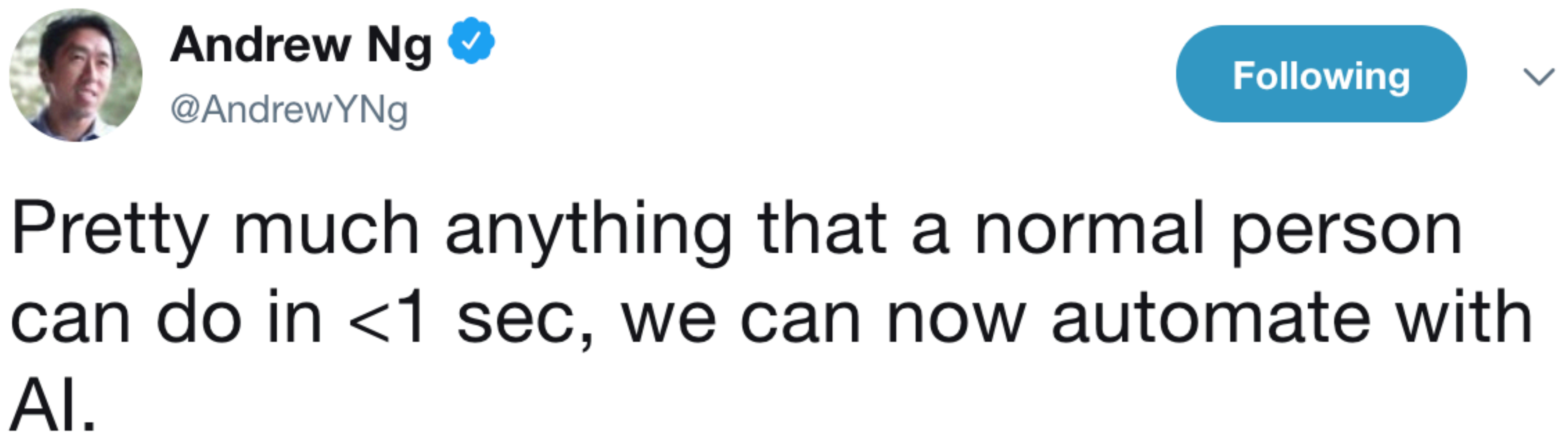






See “Designing Collaborative AI” (Ben Reinhardt and Belmer Negrillo):
https://medium.com/@Ben_Reinhardt/designing-collaborative-ai-5c1e8dbc8810

Another heuristic for assessing feasibility



Examples	Counter-examples?
• Recognize content of images	• Understand humor / sarcasm
• Understand speech	• In-hand robotic manipulation
• Translate speech	• Generalize to new scenarios
• Grasp objects	• etc.
• etc.	

Why is FSR focusing on pose estimation?

Impact

- FSR's goal is grasping - requires reliable pose estimation
- Traditional robotics pipeline uses hand-designed heuristics & online optimization
  - Slow
  - Brittle
  - Great candidate for Software 2.0!

Feasibility

- Data availability
  - Easy to collect data
  - Labeling data could be a challenge, but can instrument lab with sensors
- Accuracy requirement
  - Require high accuracy to grasp an object: <0.5cm
  - However, low cost of failure - picks per hour important, not % successes
- Problem difficulty
  - Similar published results exist but need to adapt to our objects and robot

Key points for prioritizing projects

A. To find high-impact ML problems, look for complex parts of your pipeline and places where cheap prediction is valuable

B. The cost of ML projects is primarily driven by data availability, but your accuracy requirement also plays a big role

Outline of the rest of the lecture

1. Prioritizing projects & choosing goals
2. **Choosing metrics**
3. Choosing baselines

Key points for choosing a metric

A. The real world is messy; you usually care about lots of metrics

B. However, ML systems work best when optimizing a single number

C. As a result, you need to pick a formula for combining metrics

D. This formula can and will change!

Review of accuracy, precision, and recall

Confusion matrix

n=100	Predicted: NO	Predicted: YES	
Actual: NO	5	5	**10**
Actual: YES	45	45	**90**
	50	**50**	

Review of accuracy, precision, and recall

Confusion matrix

n=100	Predicted: NO	Predicted: YES	
Actual: NO	5	5	**10**
Actual: YES	45	45	**90**
	50	**50**	

$$\textbf{Accuracy} \quad \frac{\text{Correct}}{\text{Total}}$$

50%

Review of accuracy, precision, and recall

Confusion matrix

n=100	Predicted: NO	Predicted: YES	
Actual: NO	5	5	**10**
Actual: YES	45	45	**90**
	50	**50**	

$$\textbf{Precision} \quad \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

90%

Review of accuracy, precision, and recall

Confusion matrix

n=100	Predicted: NO	Predicted: YES	
Actual: NO	5	5	**10**
Actual: YES	45	45	**90**
	50	**50**	

Recall

$$\frac{\text{true positives}}{\text{Actual YES}}$$

50%

Why choose a single metric?

	Precision	Recall
Model 1	0.9	0.5
Model 2	0.8	0.7
Model 3	0.7	0.9

Which is best?

How to combine metrics

- Simple average / weighted average

Combining precision and recall

	Precision	Recall
Model 1	0.9	0.5
Model 2	0.8	0.7
Model 3	0.7	0.9

Combining precision and recall

	Precision	Recall	(p + r) / 2
Model 1	0.9	0.5	0.7
Model 2	0.8	0.7	0.75
Model 3	0.7	0.9	0.8

Combining precision and recall

	Precision	Recall	(p + r) / 2
Model 1	0.9	0.5	0.7
Model 2	0.8	0.7	0.75
Model 3	0.7	0.9	0.8

How to combine metrics

- Simple average / weighted average

How to combine metrics

- Simple average / weighted average
- Threshold n-1 metrics, evaluate the nth

Thresholding metrics

Choosing which metrics to threshold	• Domain judgment (e.g., which metrics can you engineer around?) • Which metrics are least sensitive to model choice? • Which metrics are closest to desirable values?
Choosing threshold values	• Domain judgment (e.g., what is an acceptable tolerance downstream? What performance is achievable?) • How well does the baseline model do? • How important is this metric right now?

Combining precision and recall

	Precision	Recall	(p + r) / 2
Model 1	0.9	0.5	0.7
Model 2	0.8	0.7	0.75
Model 3	0.7	0.9	0.8

Combining precision and recall

	Precision	Recall	(p + r) / 2	p @ (r > 0.6)
Model 1	0.9	0.5	0.7	0.0
Model 2	0.8	0.7	0.75	0.8
Model 3	0.7	0.9	0.8	0.7

Combining precision and recall

	Precision	Recall	(p + r) / 2	p @ (r > 0.6)
Model 1	0.9	0.5	0.7	0.0
Model 2	0.8	0.7	0.75	0.8
Model 3	0.7	0.9	0.8	0.7

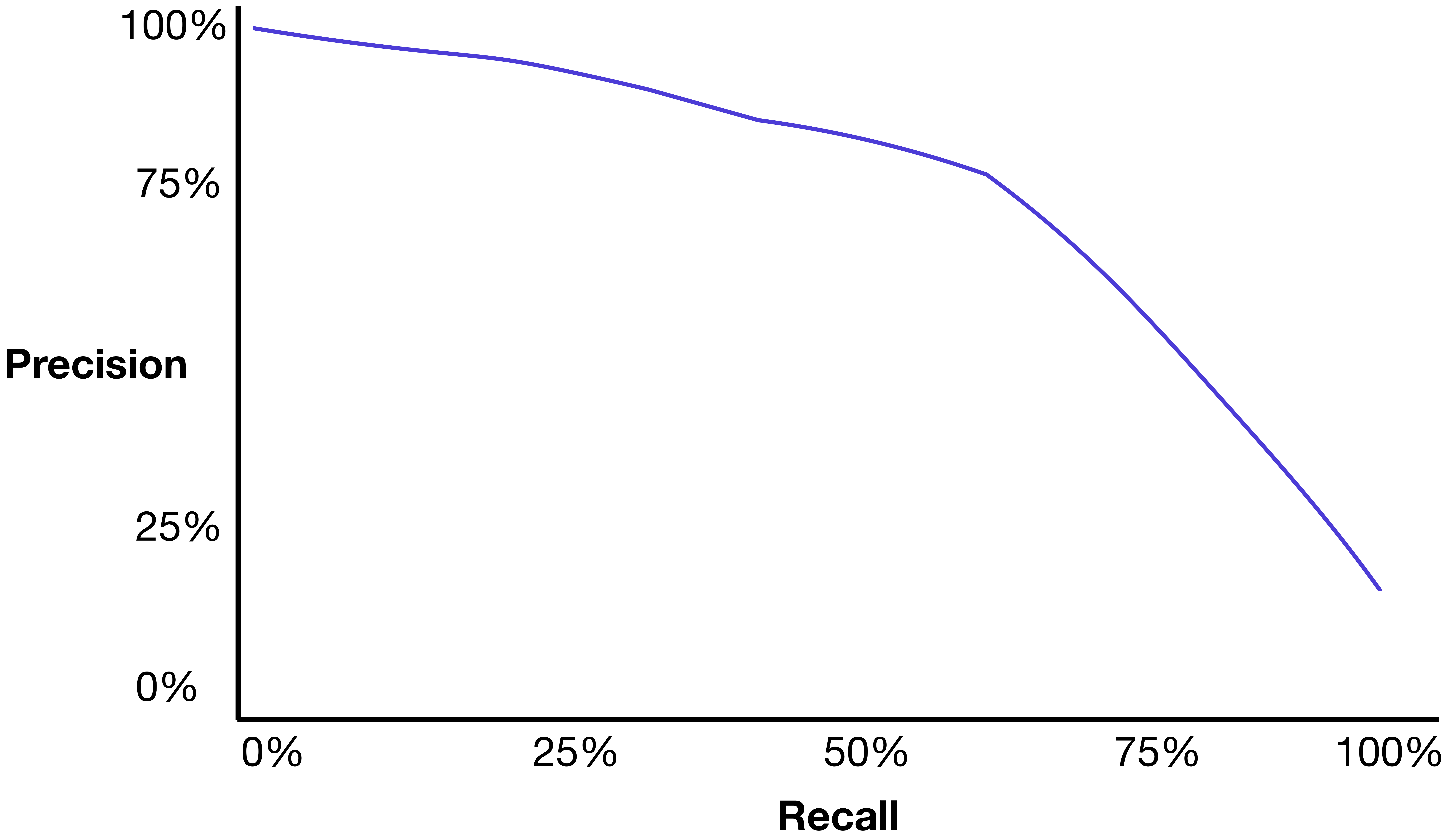
How to combine metrics

- Simple average / weighted average
- Threshold n-1 metrics, evaluate the nth

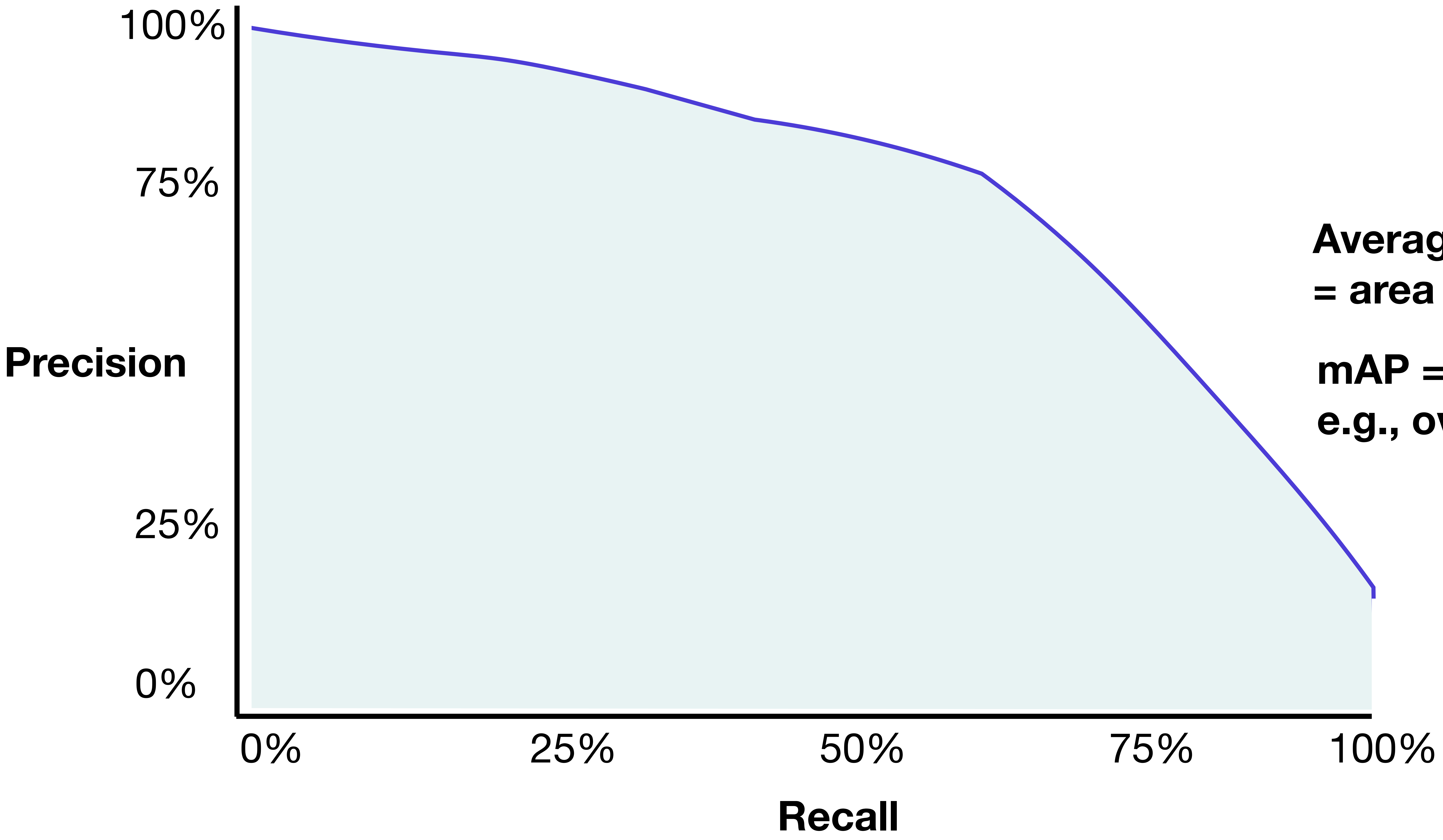
How to combine metrics

- Simple average / weighted average
- Threshold n-1 metrics, evaluate the nth
- More complex / domain-specific formula

Domain-specific metrics: mAP



Domain-specific metrics: mAP



Average precision (AP) = area under the curve

mAP = *mean* AP, e.g., over classes

Combining precision and recall

	Precision	Recall	(p + r) / 2	p @ (r > 0.6)
Model 1	0.9	0.5	0.7	0.0
Model 2	0.8	0.7	0.75	0.8
Model 3	0.7	0.9	0.8	0.7

Combining precision and recall

	Precision	Recall	(p + r) / 2	p @ (r > 0.6)	mAP
Model 1	0.9	0.5	0.7	0.0	0.7
Model 2	0.8	0.7	0.75	0.8	0.6
Model 3	0.7	0.9	0.8	0.7	0.6

Combining precision and recall

	Precision	Recall	(p + r) / 2	p @ (r > 0.6)	mAP
Model 1	0.9	0.5	0.7	0.0	0.7
Model 2	0.8	0.7	0.75	0.8	0.6
Model 3	0.7	0.9	0.8	0.7	0.6

Example: choosing a metric for pose estimation

Xiang, Yu, et al. "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes." arXiv preprint arXiv:1711.00199 (2017).

(x, y, z) **Position (L2 loss)**

(ϕ, θ, ψ) **Orientation (L2 loss)**

t **Prediction time**

Example: choosing a metric for pose estimation

- **Enumerate requirements**
  - Downstream goal is real-time robotic grasping
  - Position error must be <1cm, not sure exactly how precise is needed
  - Angular error <5 degrees
  - Must run in 100ms to work in real-time

Example: choosing a metric for pose estimation

- Enumerate requirements
- **Evaluate current performance**
  - Train a few models

Example: choosing a metric for pose estimation

- Enumerate requirements
- Evaluate current performance
- **Compare current performance to requirements**
  - Position error between 0.75 and 1.25cm (depending on hyperparameters)
  - All angular errors around 60 degrees
  - Inference time ~300ms

Example: choosing a metric for pose estimation

- Enumerate requirements
- Evaluate current performance
- **Compare current performance to requirements**
  - Prioritize angular error
  - Threshold position error at 1cm
  - Ignore run time for now

Example: choosing a metric for pose estimation

- Enumerate requirements
- Evaluate current performance
- Compare current performance to requirements
- **Revisit metric as your numbers improve**

Key points for choosing a metric

A. The real world is messy; you usually care about lots of metrics

B. However, ML systems work best when optimizing a single number

C. As a result, you need to pick a formula for combining metrics

D. This formula can and will change!

Outline

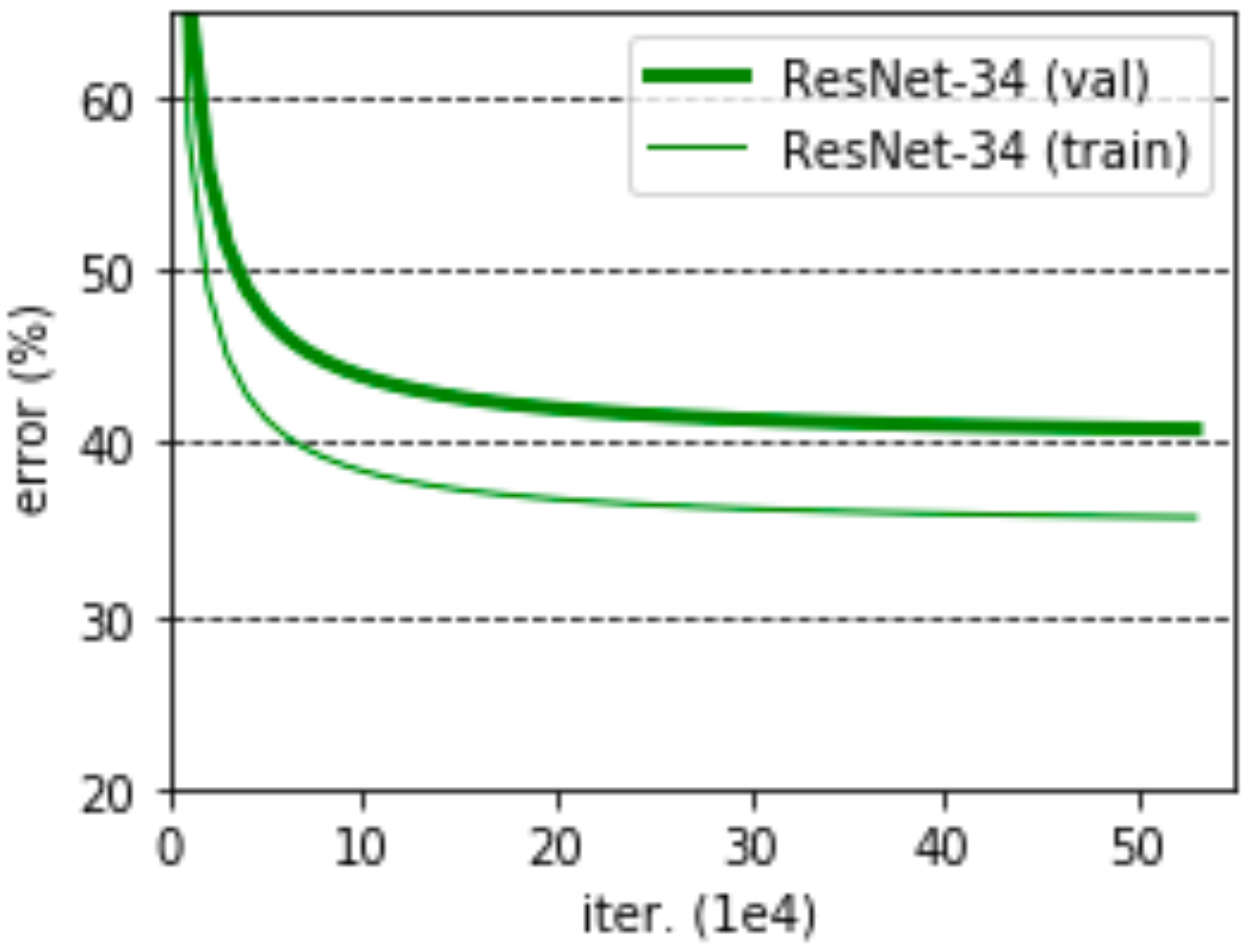
1. Prioritizing projects & choosing goals
2. Choosing metrics
3. **Choosing baselines**

Key points for choosing baselines

A. Baselines give you a lower bound on expected model performance

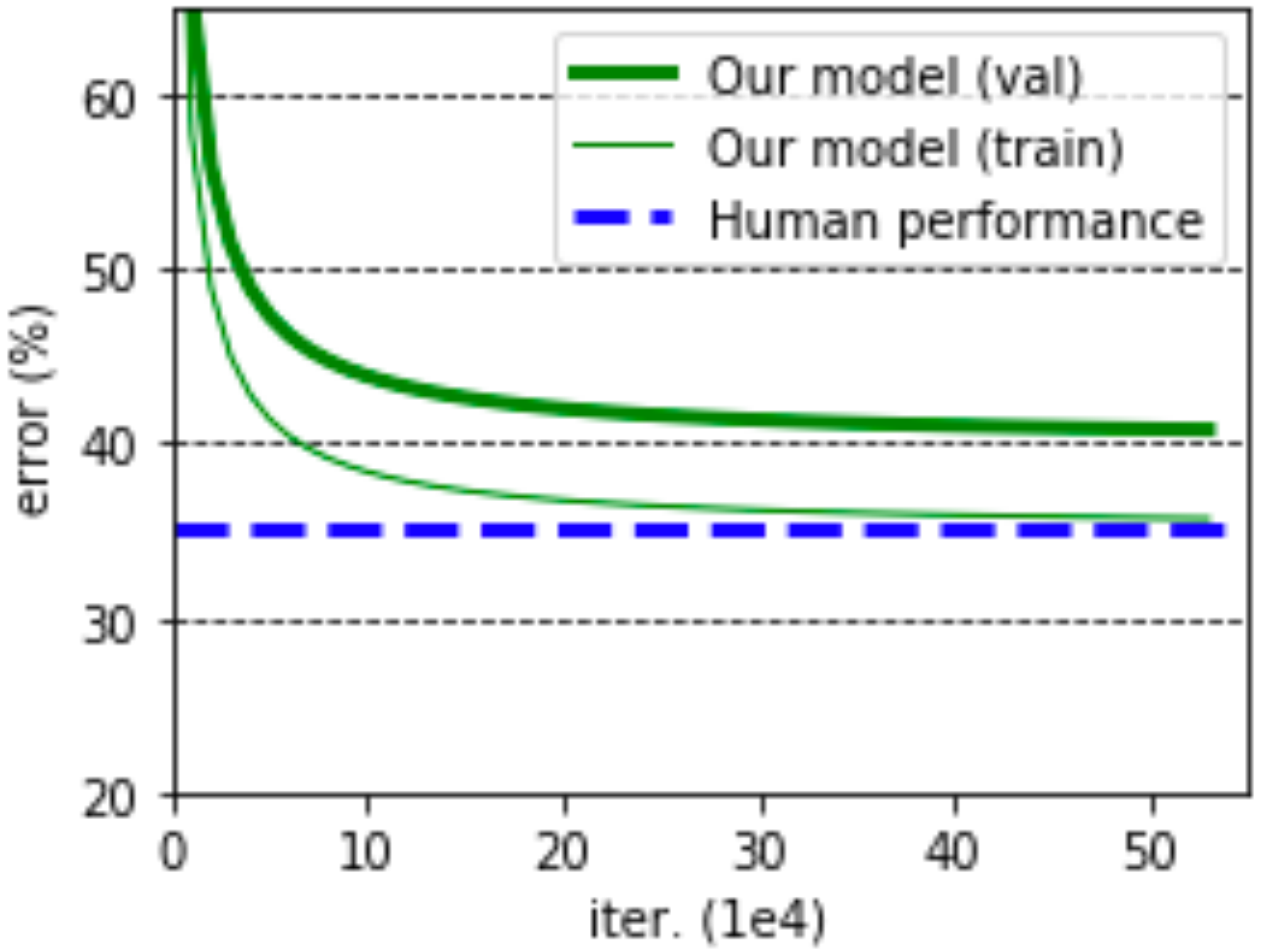
B. The tighter the lower bound, the more useful the baseline (e.g., published results, carefully tuned pipelines, & human baselines are better)

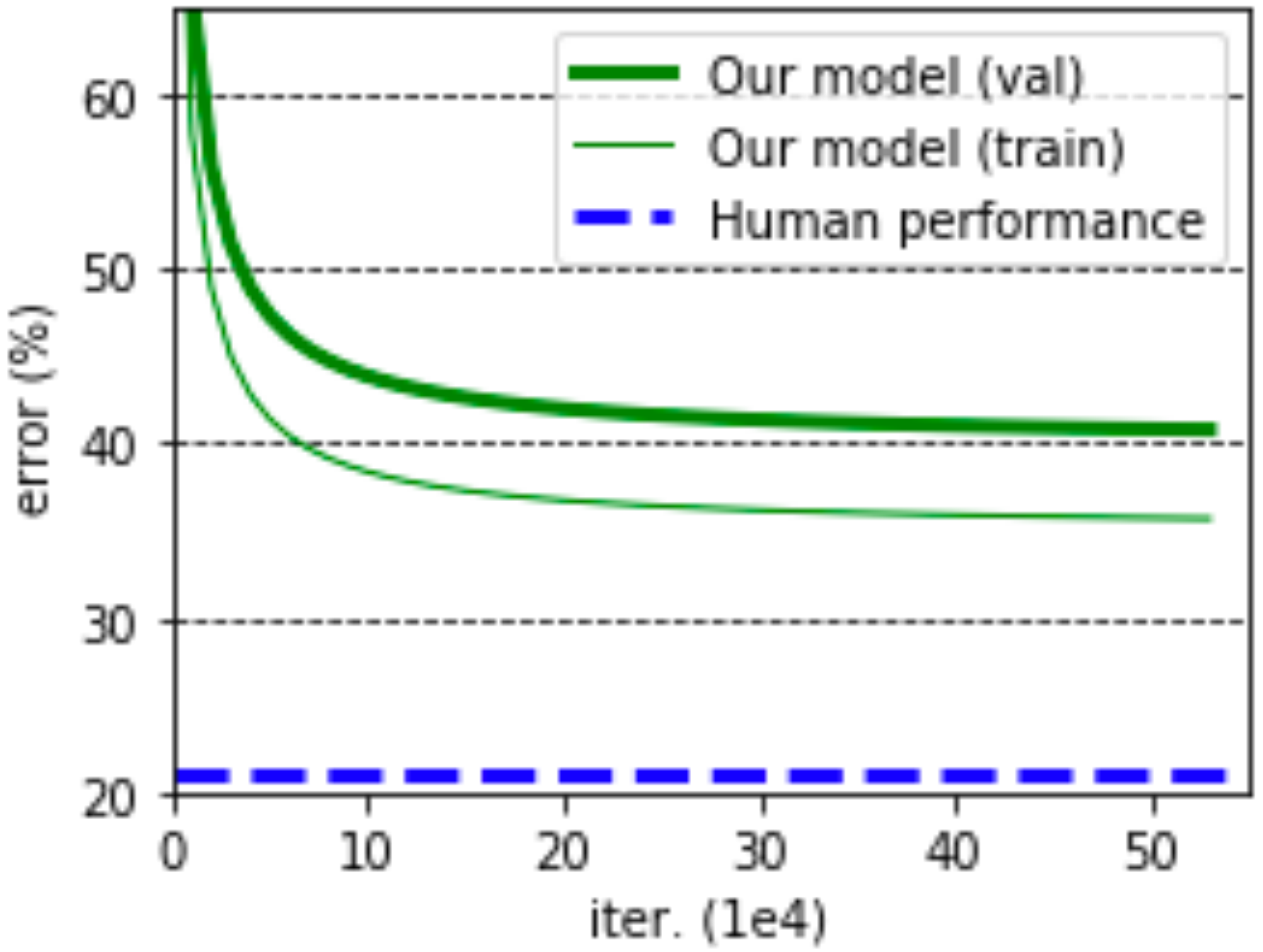
Why are baselines important?



Why are baselines important?

Same model, different baseline —> different next steps





Where to look for baselines

External baselines

- Business / engineering requirements

Where to look for baselines

External baselines

- Business / engineering requirements
- Published results ↦ **Make sure comparison is fair!**

Where to look for baselines

External baselines

- Business / engineering requirements
- Published results

Internal baselines

- Scripted baselines

Where to look for baselines

External baselines

- Business / engineering requirements
- Published results

Internal baselines

- Scripted baselines, e.g.,
  - OpenCV scripts
  - Rules-based methods

Where to look for baselines

External baselines

- Business / engineering requirements
- Published results

Internal baselines

- Scripted baselines
- Simple ML baselines

Where to look for baselines

External baselines

- Business / engineering requirements
- Published results

Internal baselines

- Scripted baselines
- Simple ML baselines, e.g.,
  - Standard feature-based models (e.g., bag-of-words classifier)
  - Linear classifier with hand-engineered features
  - Basic neural network model (e.g., VGG-like architecture without batch norm, weight norm, etc.)

How to create good human baselines

Quality of baseline		Ease of data collection
Low	Random people (e.g., Amazon Turk)	High
	Ensemble of random people	
	Domain experts (e.g., doctors)	
	Deep domain experts (e.g., specialists)	
High	Mixture of experts	Low

How to create good human baselines

- Highest quality that allows more data to be labeled easily
- More specialized domains need more skilled labelers
- Find cases where model performs worse and concentrate data collection

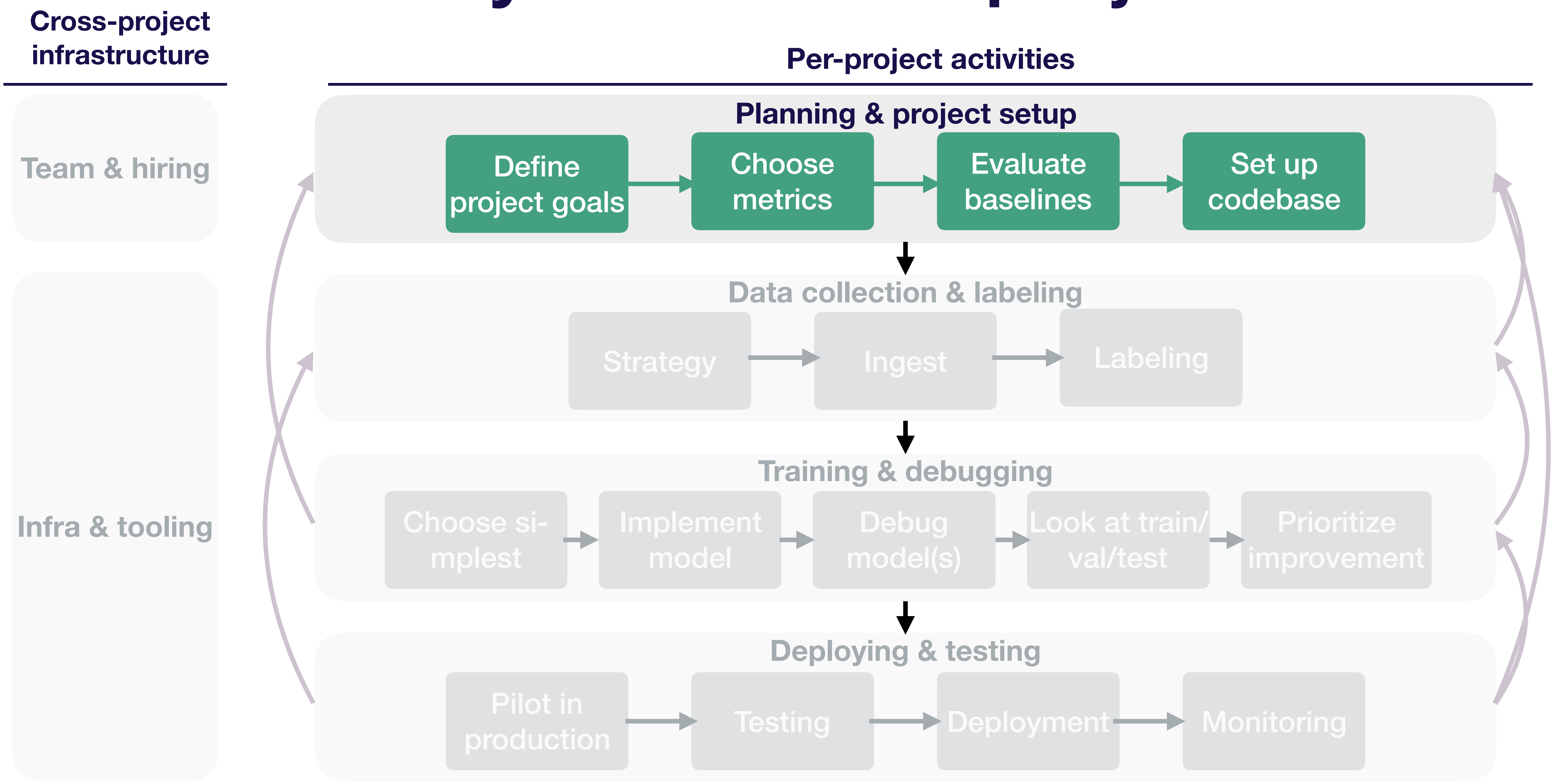
More on labeling in data lecture!

Key points for choosing baselines

A. Baselines give you a lower bound on expected model performance

B. The tighter the lower bound, the more useful the baseline (e.g., published results, carefully tuned pipelines, human baselines are better)

Lifecycle of a ML project



Questions?

Where to go to learn more

- Andrew Ng’s “Machine Learning Yearning”
- Andrej Karpathy’s “Software 2.0”
- Agrawal’s “The Economics of AI”