Theoretical Foundations of Large-scale Machine Learning

ECE 826

Lecture 1

Intros

- *Instructor*

Dimitris Papailiopoulos
dimitris@papail.io
dimitris@ece.wisc.edu

- *Times*

Tu. + Thu. @ 9:30 - 10:45pm

- *Office Hours:*

Just send an email,
and we'll arrange a mutually convenient time.

- *Webpage:*

papail.io/826

Material

The course is based on recent books and text on ML, OPT, statistics, and systems

- Everything online:

  papail.io/826

  - Books
  - Papers
  - Lecture Notes
  - Slides

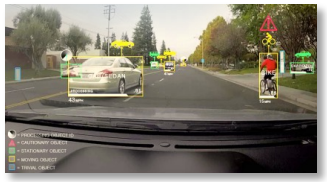
Overview

- Why Machine Learning?
- Why Large-Scale?
- Where does Optimization fit in?
- What we will cover?
- Who is this class for?
- Grading

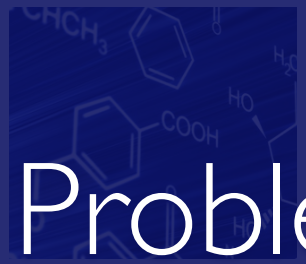
Why Machine Learning?

Real and Important Problems

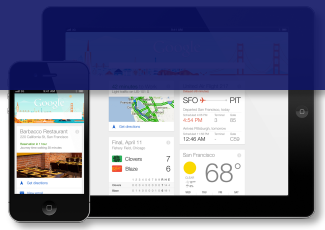
Automation and Robotics

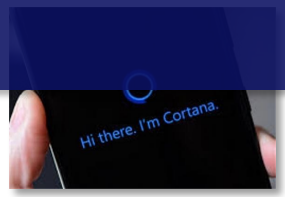


Drug Discovery and Healthcare



Intelligent Personal Assistants



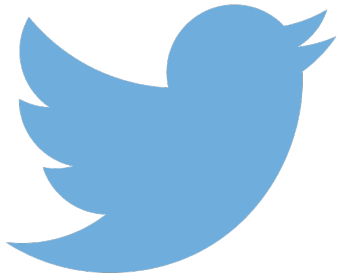


Recommender Systems

amazon

NETFLIX


facebook




amazon

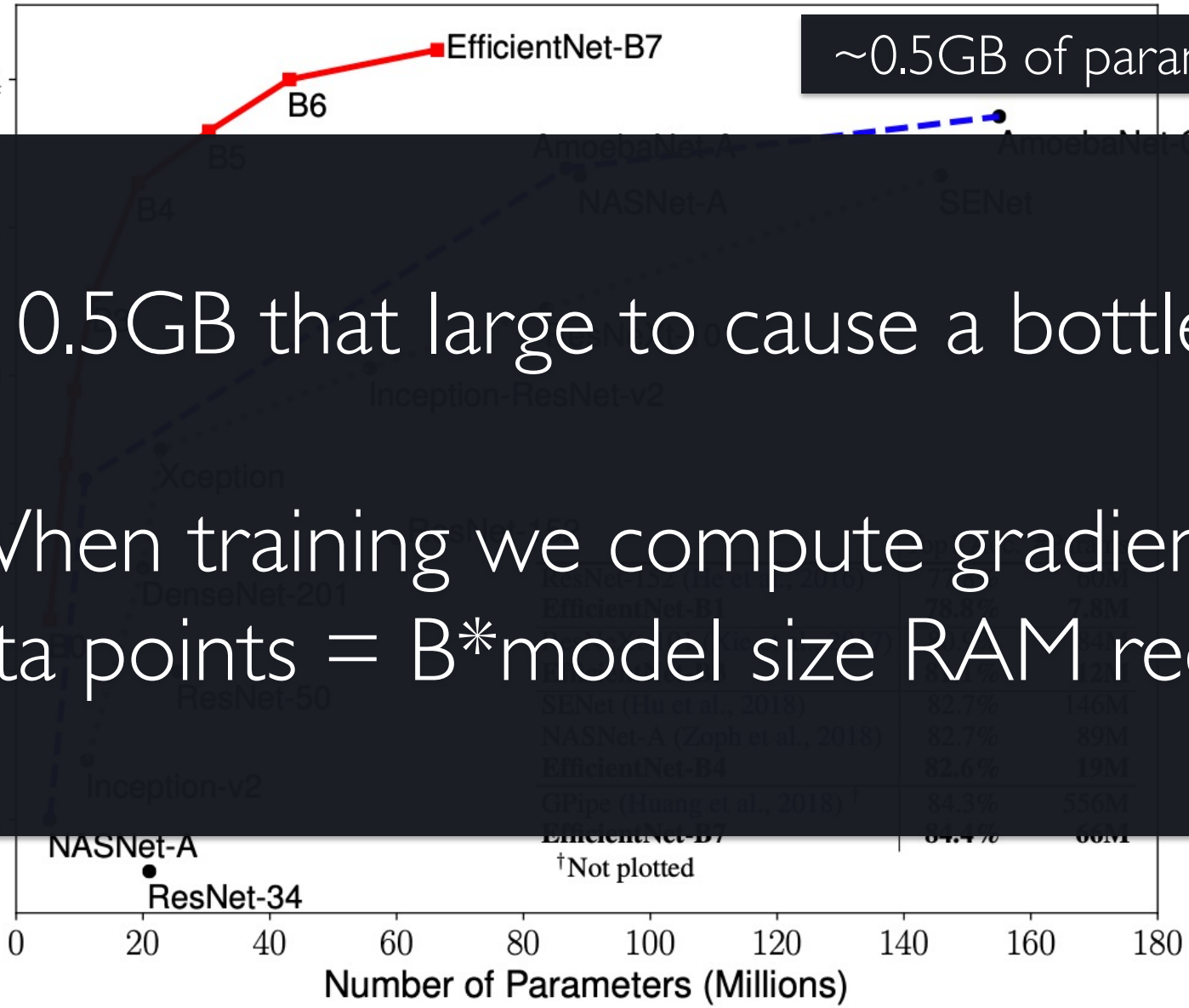

IBM


Taken in 2016
Get alerts for this search
Taken in 2022
Turn on job alerts
163,000+ Machine Learning Jobs in United States (3,117 new)

Real Jobs

Why Large-scale?

Model sizes (vision)

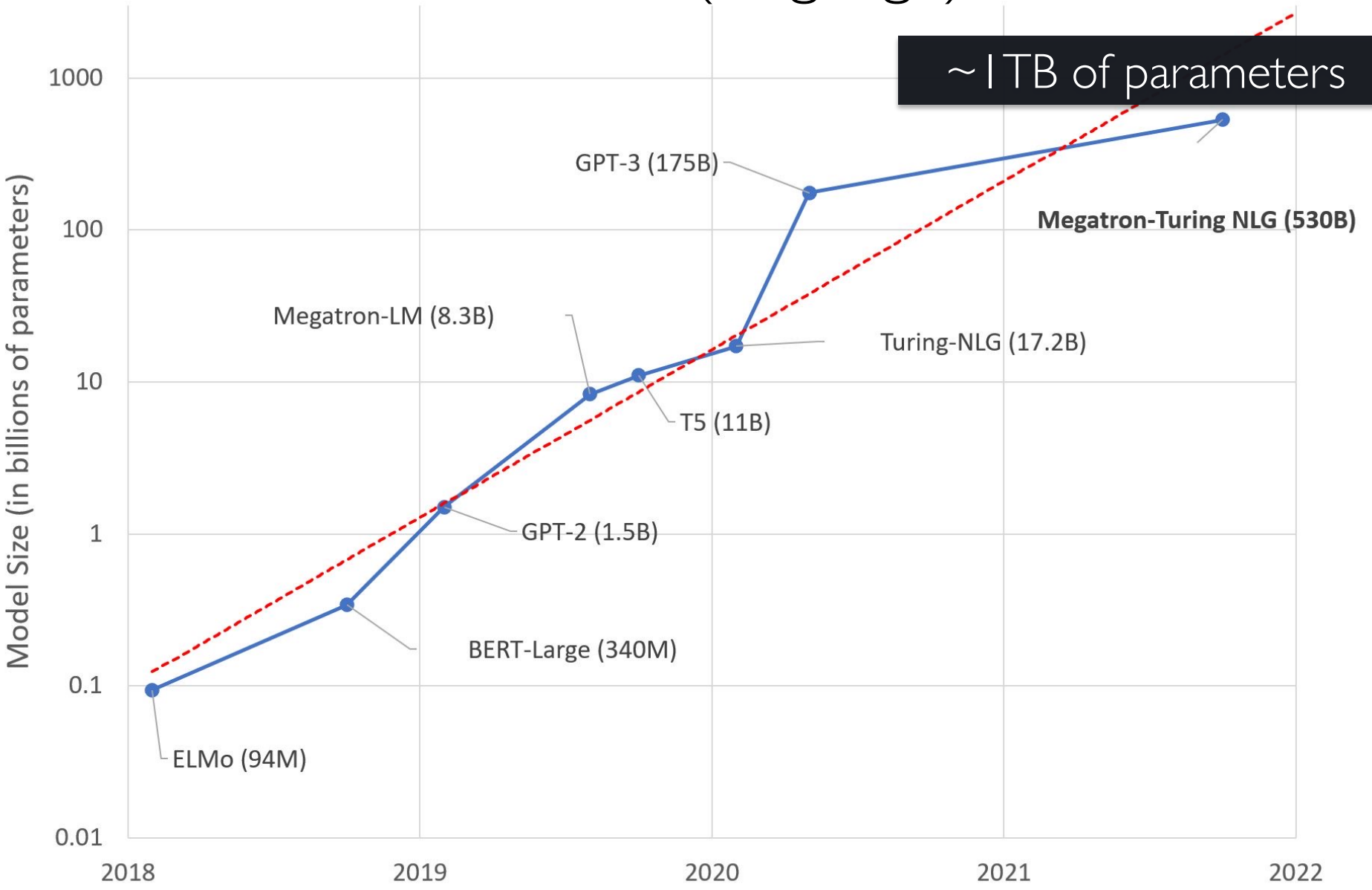


Q: is 0.5GB that large to cause a bottleneck?

A: When training we compute gradients on B data points = B*model size RAM required

[Source: http://proceedings.mlr.press/v97/tan19a/tan19a.pdf]

Model sizes (language)



[**Source**: https://tinyurl.com/Megatron-Turing-NLG]

Data set sizes

IMAGENET (ILSVRC) 1.2 mil 256 x 256 images = 150 GB

Name	Megatron MT-NLG
Lab	NVIDIA and Microsoft
Parameters	530B (530,000M)
Dataset sources	Trained with The Pile v1 + more, totalling 15 datasets:
	Books3
	PubMed Abstracts
	BookCorpus2
	NIH ExPorter
	Pile-CC
	ArXiv
	GitHub
	+ Common Crawl 2020
	+ Common Crawl 2021
	+ RealNews, from 5000 news domains (120GB).
	+ CC-Stories, 1M story documents from the CC (31GB).
Dataset total size	>825GB

We need to train huge models on enormous data sets

[Source: https://www.image-net.org, https://lifearchitect.ai/megatron/]

Let's unpack things a bit

The Machine Learning Pipeline

What is a "regular" ML pipeline?

- *Running example*: **image classification**
- Goal:
  "train a computer to recognize a cat from a dog"

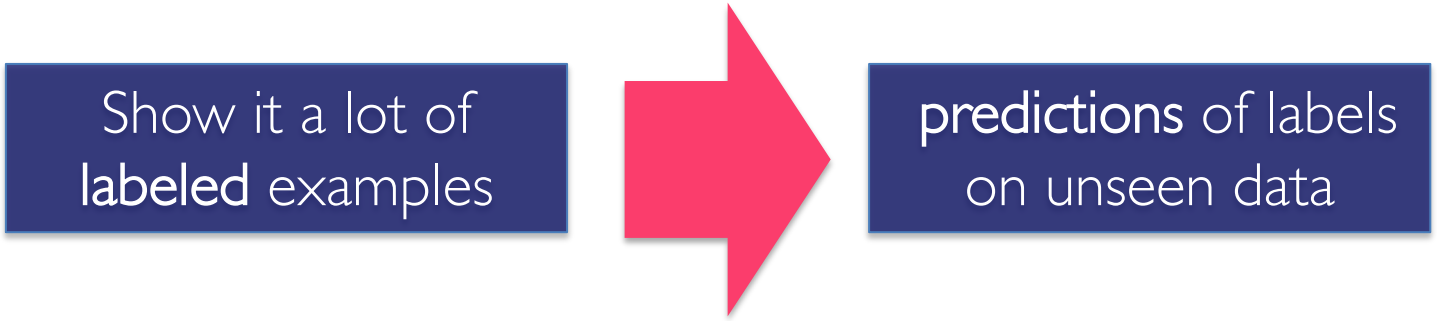
What is a "regular" ML pipeline?

- *Running example*: **image classification**
- Goal:
  "train a computer to recognize a cat from a dog"
- How?

What is a "regular" ML pipeline?

- *Running example*: **image classification**
- Goal:
  "train a computer to recognize a cat from a dog"
- How?

  Simple idea, inspired by inductive human learning



From This

To this

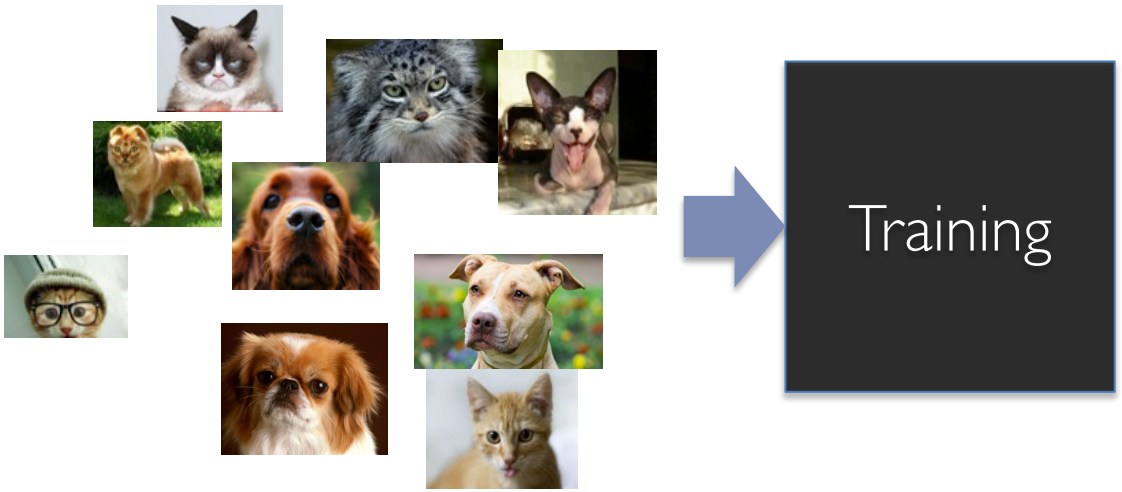
What is a "regular" ML pipeline?

- *Running example*: **image classification**
- Goal:
  "train a computer to recognize a cat from a dog"
- How?

Input Data

What is a "regular" ML pipeline?

- *Running example*: **image classification**
- Goal:
  "train a computer to recognize a cat from a dog"
- How?

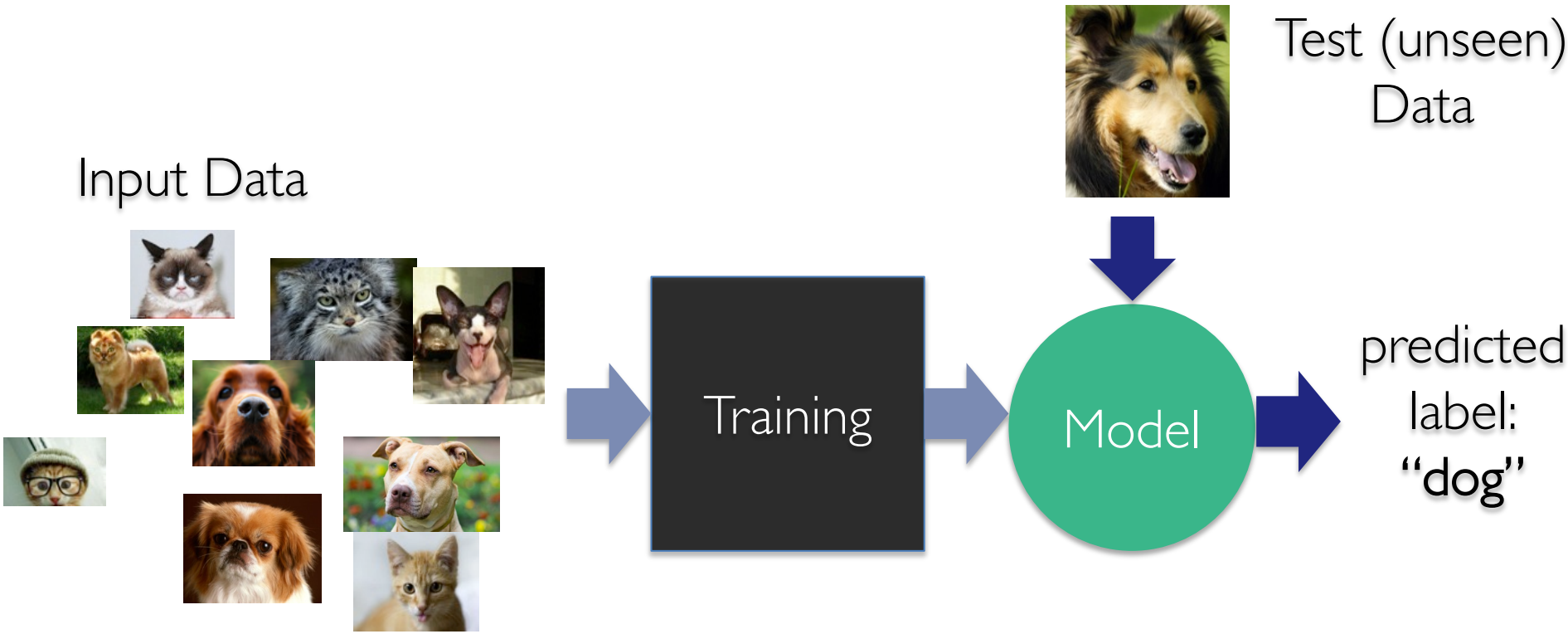


What is a "regular" ML pipeline?

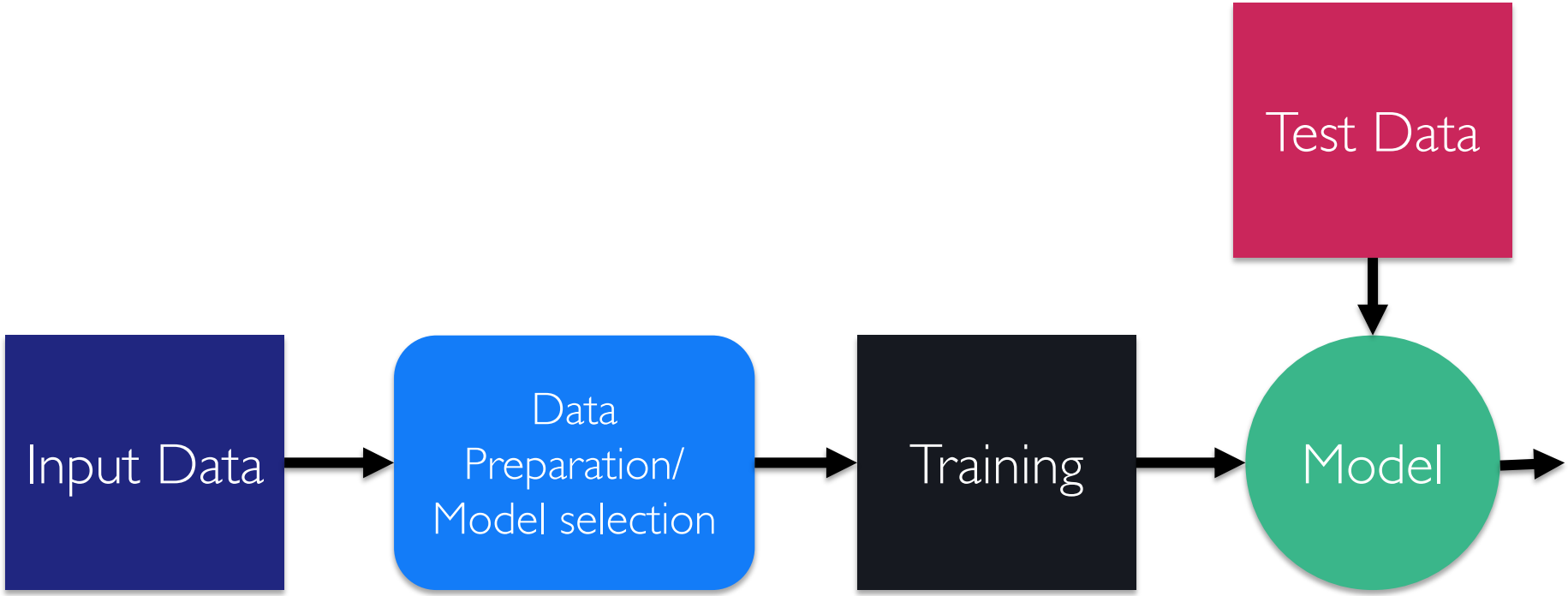
- *Running example*: **image classification**
- Goal:
  "train a computer to recognize a cat from a dog"
- How?

What is a "regular" ML pipeline?

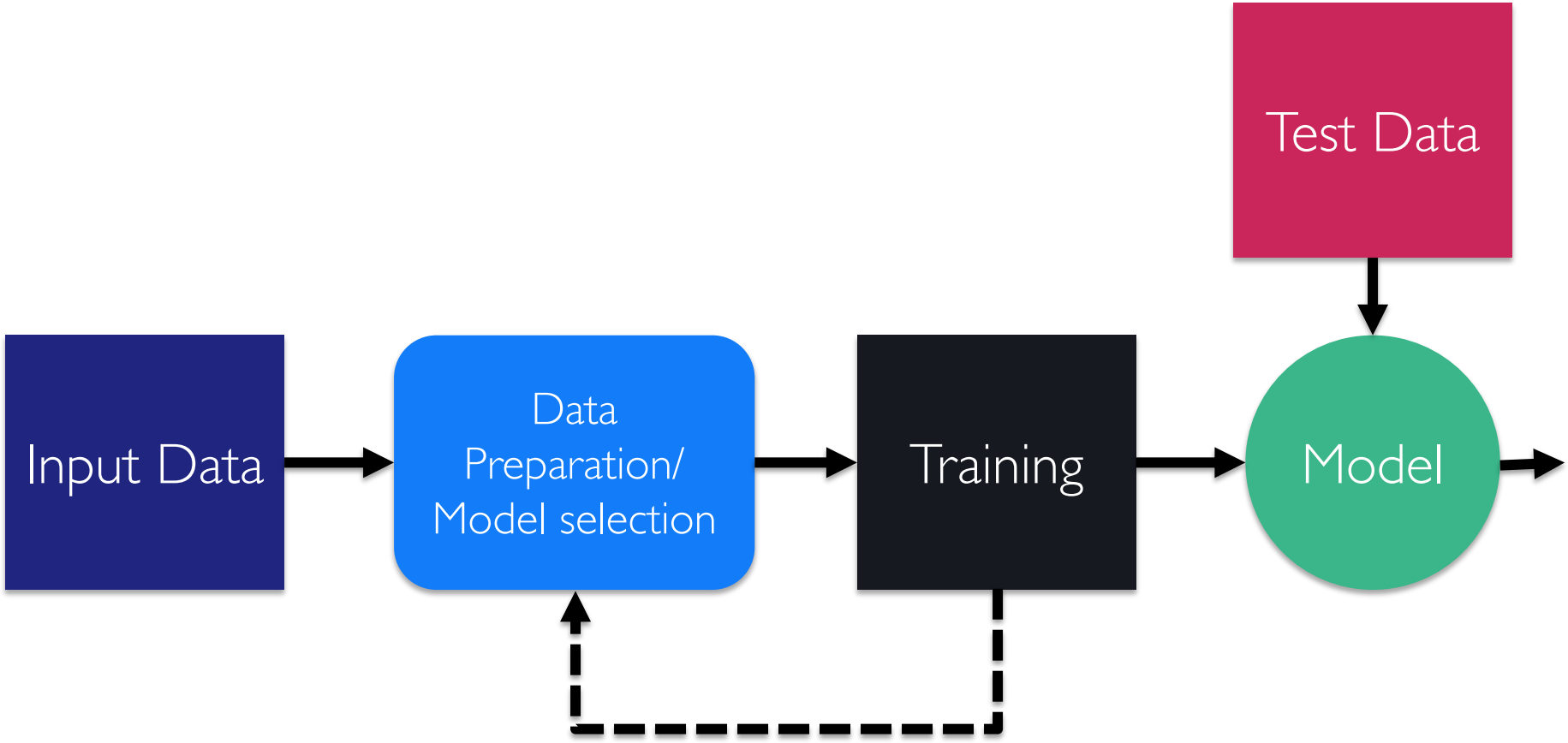
- *Running example*: **image classification**
- Goal:
  "train a computer to recognize a cat from a dog"
- How?



ML Pipelines



ML Pipelines

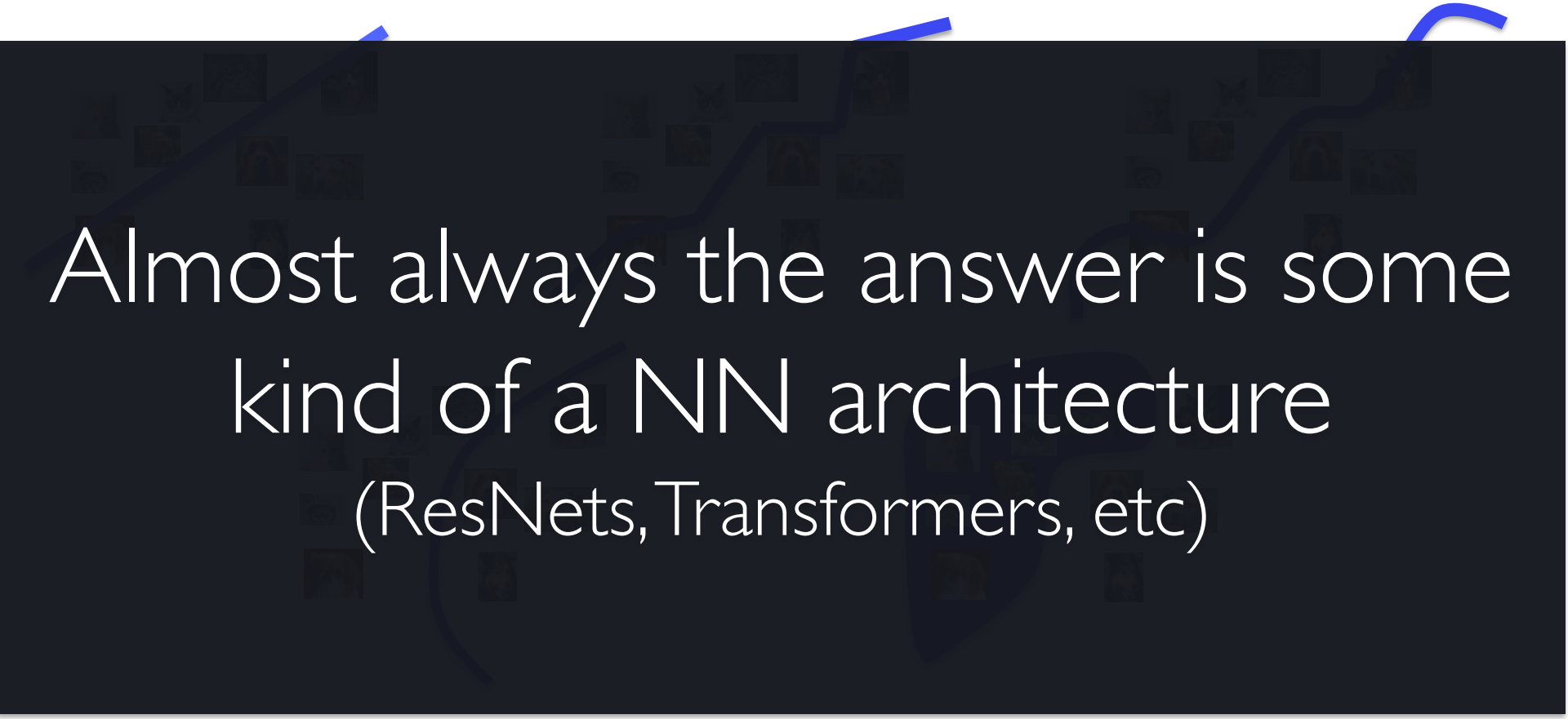


Data Preparation

- Data cleaning
- Data normalization
- Handling missing/noisy data/outliers
- Removal of unwanted features
- Putting in the appropriate format
- Data Augmentation

Goal: Make the most out of your data

Choose your predictor (hypothesis class)



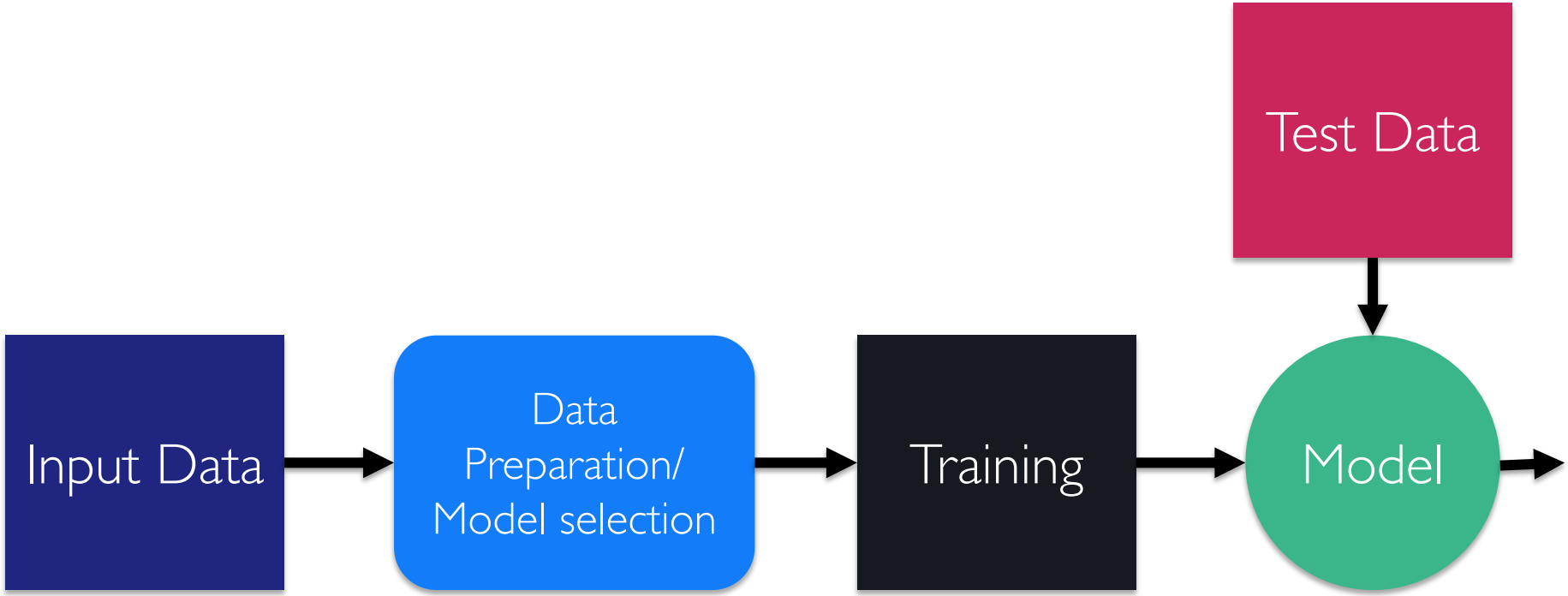
Almost always the answer is some kind of a NN architecture

(ResNets, Transformers, etc)

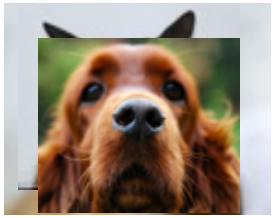
Goal: Pick a predictor that

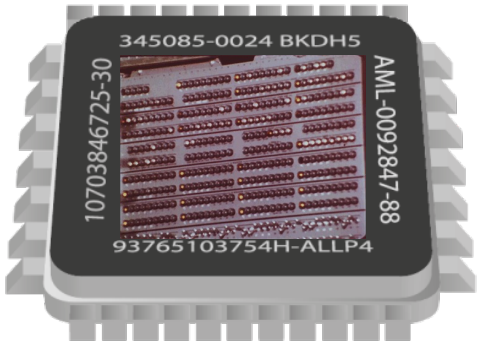
1) Is expressive 2) is easy to train 3) doesn't overfit

ML Pipelines



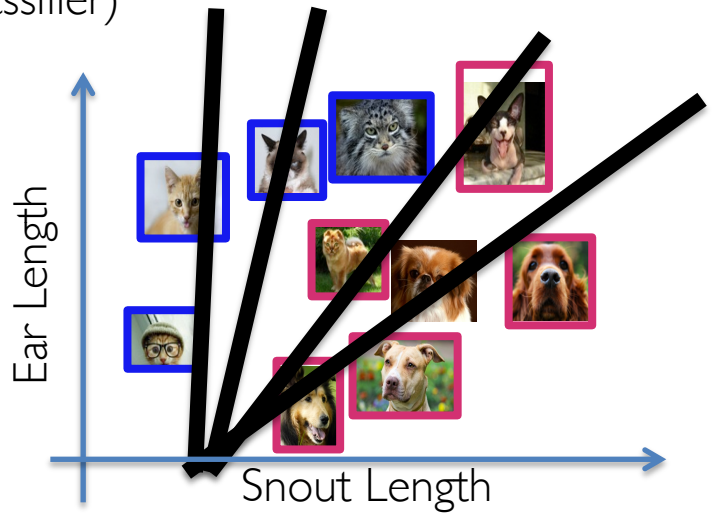
Now we train





Goal: Train a model to minimizes **training error**

Line = predictive model (classifier)



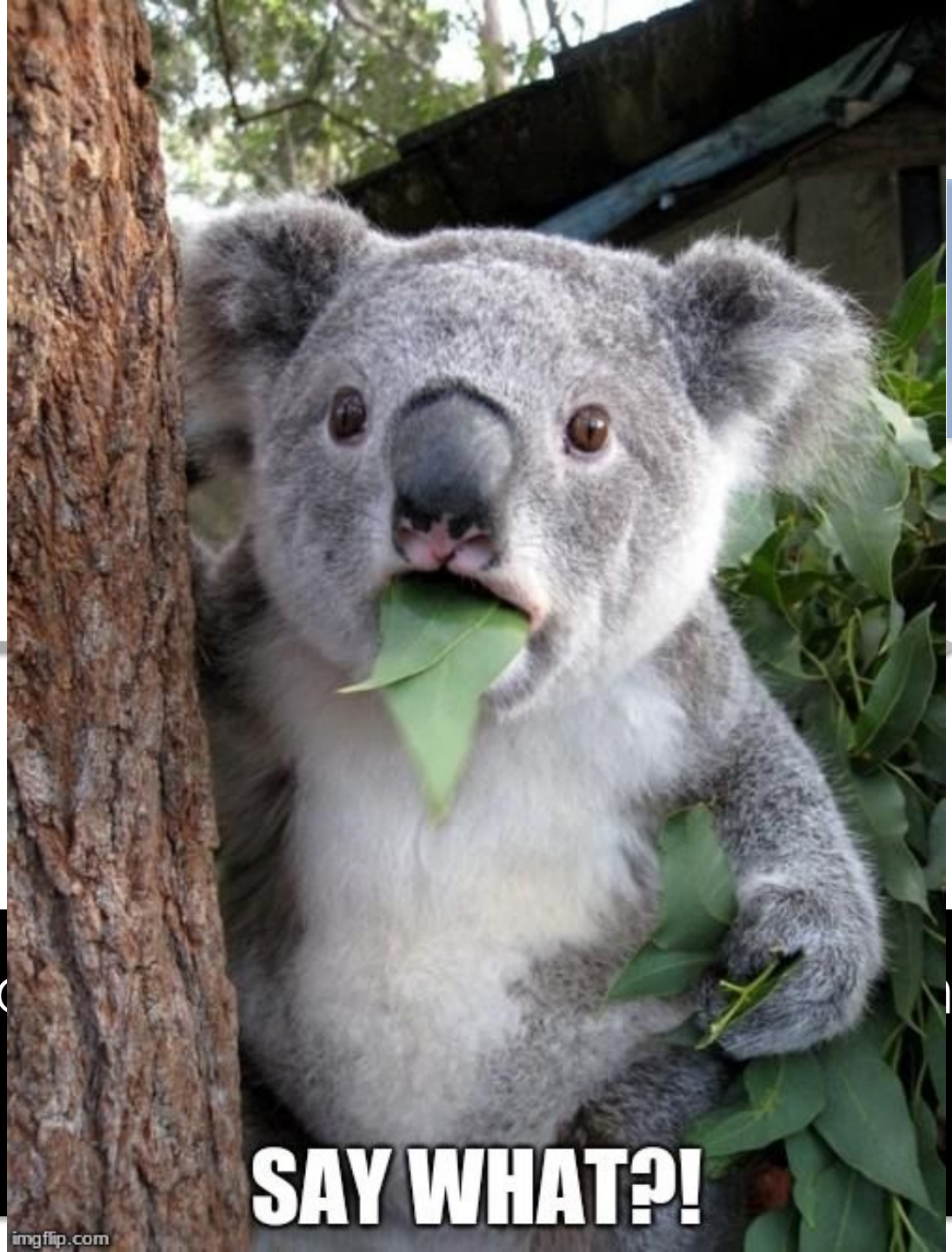
Input Data

Test Data

Model

GPT-3 wo

a single



Questions?

What is the role of Optimization?

Training via Optimization

- Many Problems in ML can be written as

$$\min_{\mathbf{w} \in \mathcal{W}} \sum_{i=1}^{n} \ell_i(\mathbf{w}) + \lambda \cdot \mathcal{R}(\mathbf{w})$$

Why?

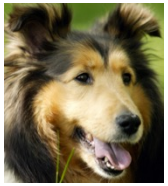
The goal of Training

- What we have: Labeled examples presented as (**features**, **label**)

$$\mathbf{z}_i = (\mathbf{x}_i, y_i)$$

- E.g., features = image, label = "cat" (-1), or "dog" (+1)

$$\left(\quad , \quad +1 \right)$$



- Assumption:

  all **labeled** (train/test) examples come from **unknown** distribution

$$\mathbf{z}_i = (\mathbf{x}_i, y_i) \sim \mathcal{D}$$

The goal of Training

- What we have: *n* **labeled** examples drawn i.i.d. from $\mathcal{D}$

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$$

- What we want: Train a **model** (a predictor function *f*)

$$f : \text{feature vector} \rightarrow \text{label}$$

That performs well on **unseen** data.

The goal of Training

- What we have: *n* **labeled** examples drawn i.i.d. from $\mathcal{D}$

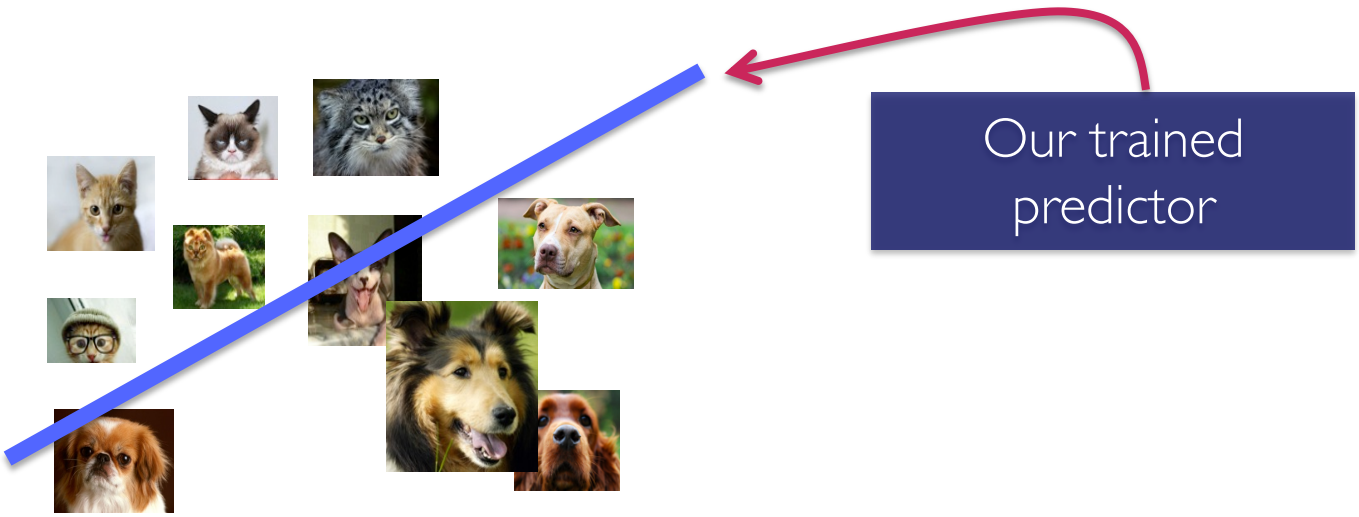
$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$$

- What we want: Train a **model** (a predictor function *f*)

$$f : \text{feature vector} \rightarrow \text{label}$$

That performs well on **unseen** data.

- E.g.,



The goal of Training

- How to measure performance?

$$\text{loss}(f(\mathbf{x}); y)$$

Measures **disagreement** between predicted and true label

Examples:
$\mathbf{1}_{f(\mathbf{x}) \neq y}$
$(f(\mathbf{x}) - y)^2$
$|f(\mathbf{x}) - y|$
Cross entropy
Logistic loss

- **Goal:**

We want a predictor *f* with **small loss on unseen data** (e.g., on the test set)

$$\sum_{(\mathbf{x},y) \text{ is an unseen example}} \text{loss}(f(\mathbf{x}); y)$$

But, we haven't seen… unseen examples

The goal of Training

- The loss on the "unseen" examples converges to the **expected loss**

$$\sum_{(\mathbf{x},y)\ \text{is an unseen example}} \text{loss}(f(\mathbf{x}); y) \longrightarrow \mathbb{E}_{(\mathbf{x},y)}\{\text{loss}(f(\mathbf{x}); y)\}$$

Expected/True risk

- What else converges to the expected loss?

$$\frac{1}{n}\sum_{i=1}^{n} \text{loss}(f(\mathbf{x}_i); y_i)$$

Empirical risk

Note: More details in next lecture

Training via Optimization

- We want to solve:

$$\min_{f \in \mathcal{F}} \mathbb{E}\{\text{loss}(f(\mathbf{x}); y)\}$$

- $\mathcal{F}$ can be:
  - NNs of depth 1
  - NNs of depth 2
  - NNs of depth 3
  - ...
- We instead solve the ERM:

$$\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \text{loss}(f(\mathbf{x}_i); y_i)$$

Generalization is another great mystery, we will come back to soon

Note: More details in next lecture

Training via Optimization

- More often than not, we minimize the empirical risk through

$$\min_{\mathbf{w} \in \mathcal{W}} \sum_{i=1}^{n} \ell_i(\mathbf{w}) + \lambda \cdot \mathcal{R}(\mathbf{w})$$

Measures model fit for data point i *(avoids under-fitting)*

Measures model "complexity" *(enhances generalization)*

Note: Several examples in next lectures.

Questions?

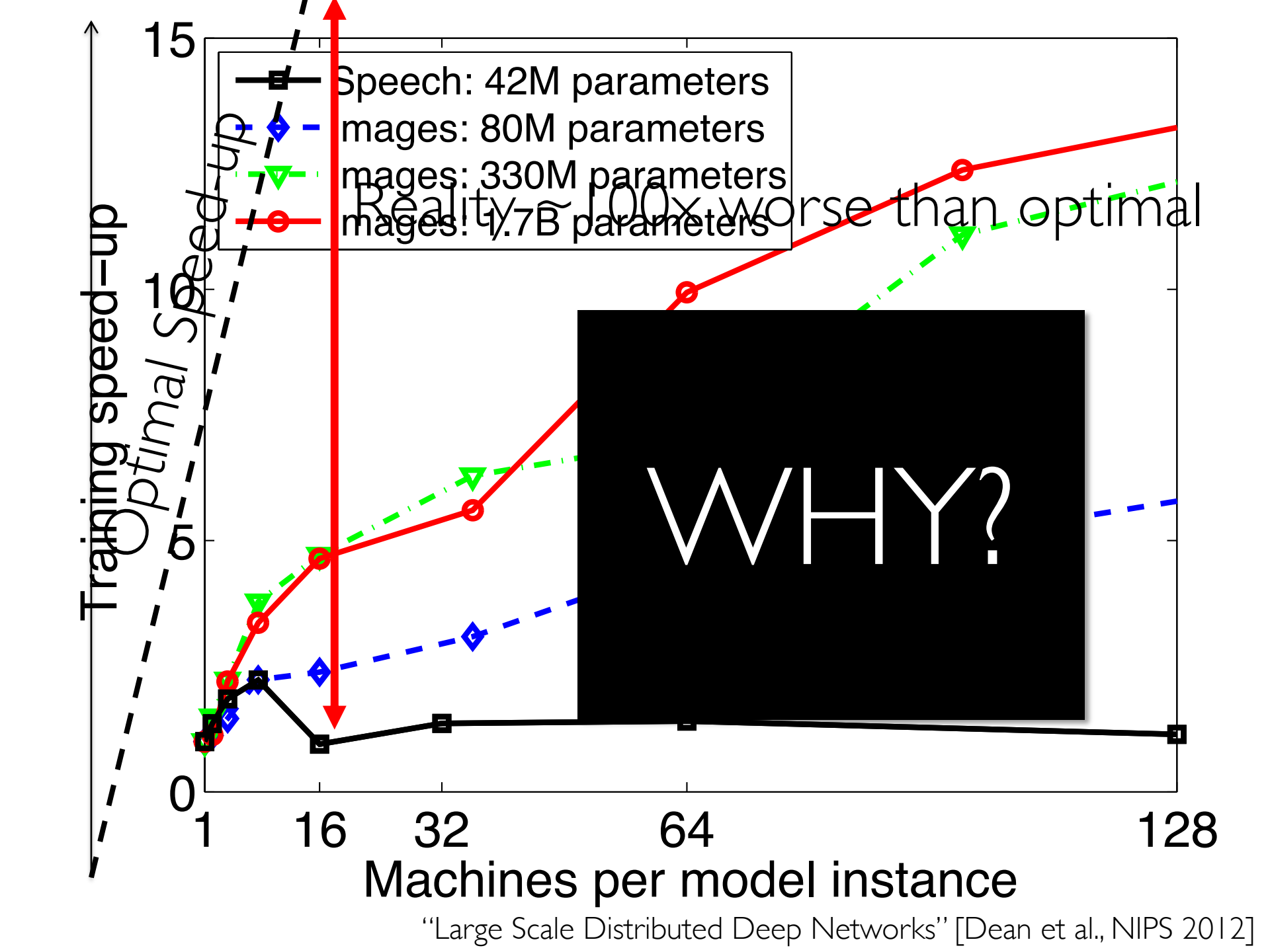
Scalability & System Aspects

Solvability ≠ **Scalability**

- Some of the problems that we visit, can be solved in **polynomial time**, but that's not enough.
  E.g.,
  O(#examples^4 * dimension^5) not scalable

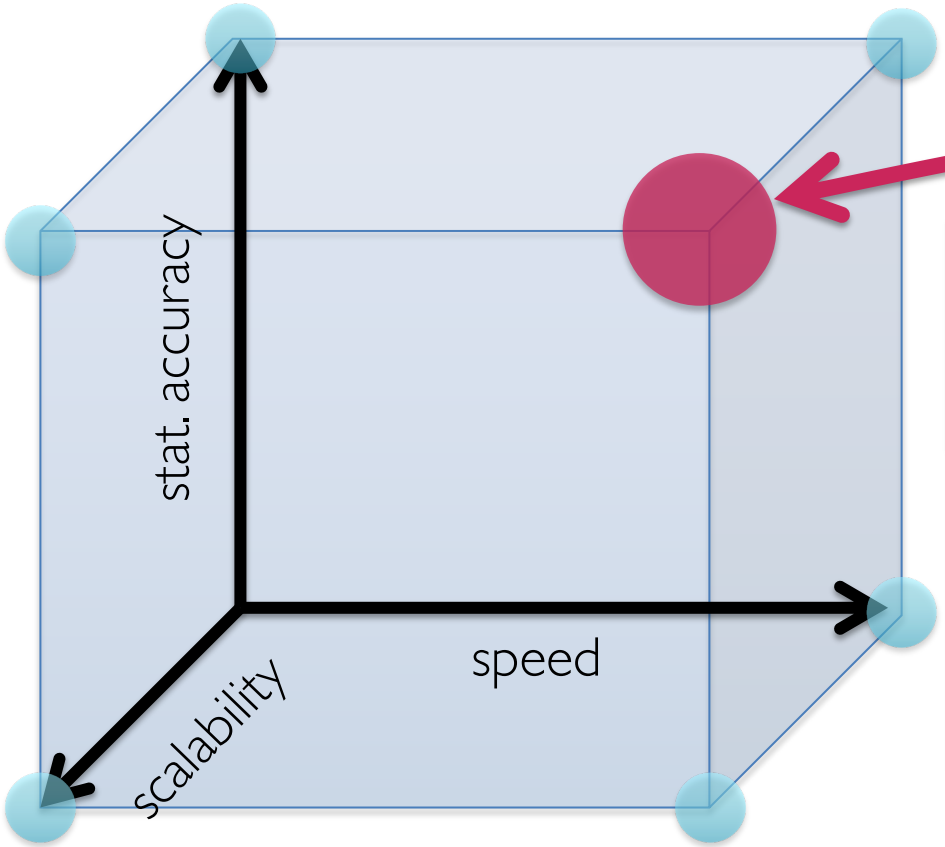
- We need **fast** algorithms!
  *Ideally O(#examples * #dimension) time*

- We want algorithms amenable to **parallelization**!
  if serial runs in O(T) time, we want O(T/P) on P cores.



"Large Scale Distributed Deep Networks" [Dean et al., NIPS 2012]

Performance Trade-off



We want our algorithms & models to be here

How?

By using both theory and practice to guide our design

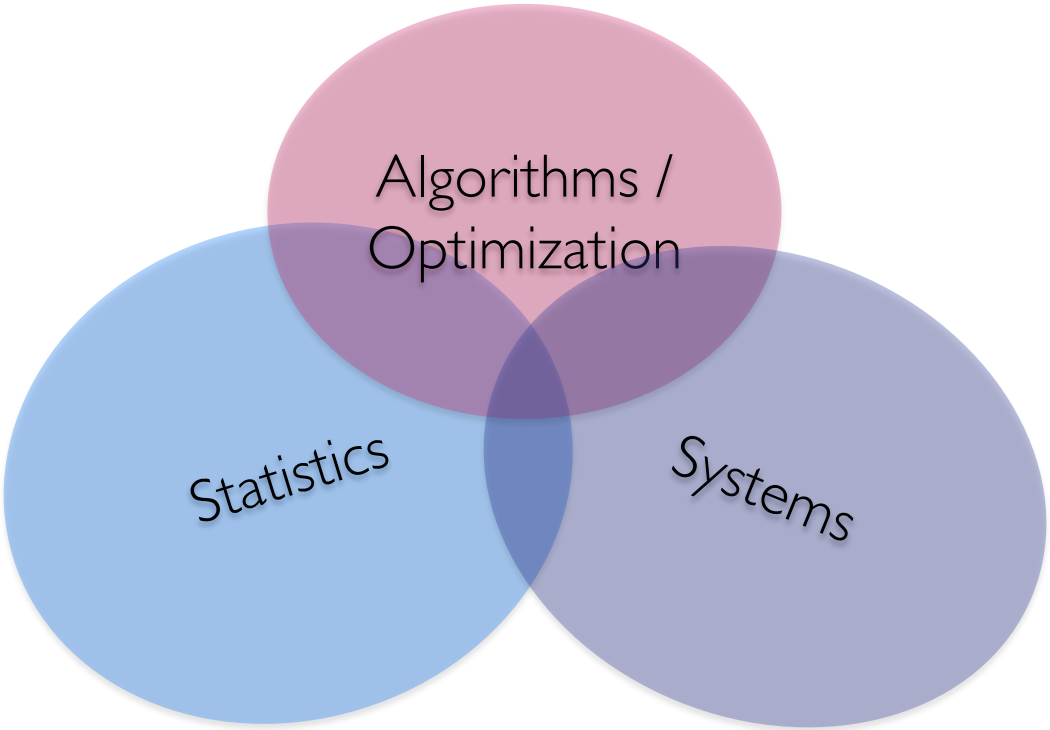
We want even more from a model

Moar!

- Robustness to attacks/hardware failures
- Low Memory footprint
- Easy to store
- Fast to do Inference on
- Easy to Personalize to individual users
- …

Contents and Goals

This course



What will we cover?

- Part 1: *(~6 weeks) (ML+OPT theory)*
  - When does a classifier generalize/why?
  - Memorization bounds for NNs
  - Generalization through Algorithmic Stability
  - Stochastic Optimization for Convex/Nonconvex Problems
  - Optimizing Neural Nets, Hardness, Expressivity, Recent results

- Part 2: *(~ 6 weeks) (large-scale ML practical aspects)*
  - Distributed/Parallel Learning
  - Federated/Decentralized Learning
  - Model & Gradient Compression/Quantization/Low precision
  - Sparse Updates and the Lottery Ticket Hypothesis
  - Dealing with worst-case Failures (Delays, Adversaries)
  - Modern Architectures: ResNets, Vision/Language Transformers, etc

What we won't cover

- Unsupervised Learning/RL/Generative Models
- Tensorflow/PyTorch
- How to train Imagenet in 1hr
- How to tune the batchnorm layers of ResNet-101
- Building the next Vision Transformer

Goals of this course

Learn useful theoretical tools and current trends for large-scale ML

Produce a research-quality project report

Questions?

Grading Logistics

Grading

No Written Exams

Grading

1. Research Project – 60%

2. Homework – 20%

3. Scribing – 10%

4. Student Participation – 10%

Semester Project *(60%)*

- Milestones:

Part 1: (20%)

Mid-semester proposal (due ~week #9)

Proposal presentation and feedback day.

Part 2: (40%)

End-semester report (due May 5)

- During the semester:

Weekly Slack updates, starting after the proposal

- What can it be?

An open problem directly relevant to the course.

Homework *(20%)*

- *Frequency:*

Every ~3 weeks
starting ~ week 4
Total of ~3

- *Content:*

- Exercises related to the class
- Paper reviews
- Some implementation component

Scribing *(10%)*

- *Frequency:*

Every lecture starting lecture #2
Due 1 week after each lecture

- *Format:*

Groups of #students / #lectures
One group / lecture
(everyone has to scribe once)
LaTeX template

Student Participation *(10%)*

- Students expected to participate with questions, and comments.
- Small project updates during semester.
- Meet with instructor before project proposal
- Other in-class activities.

Grading

1. Research Project – 60%

2. Homework – 20%

3. Scribing – 10%

4. Student Participation – 10%

Pre-Reqs

- Linear Algebra
- Probability
- Grad course in ML

Questions?

Quiz