# CS540 Introduction to Artificial Intelligence

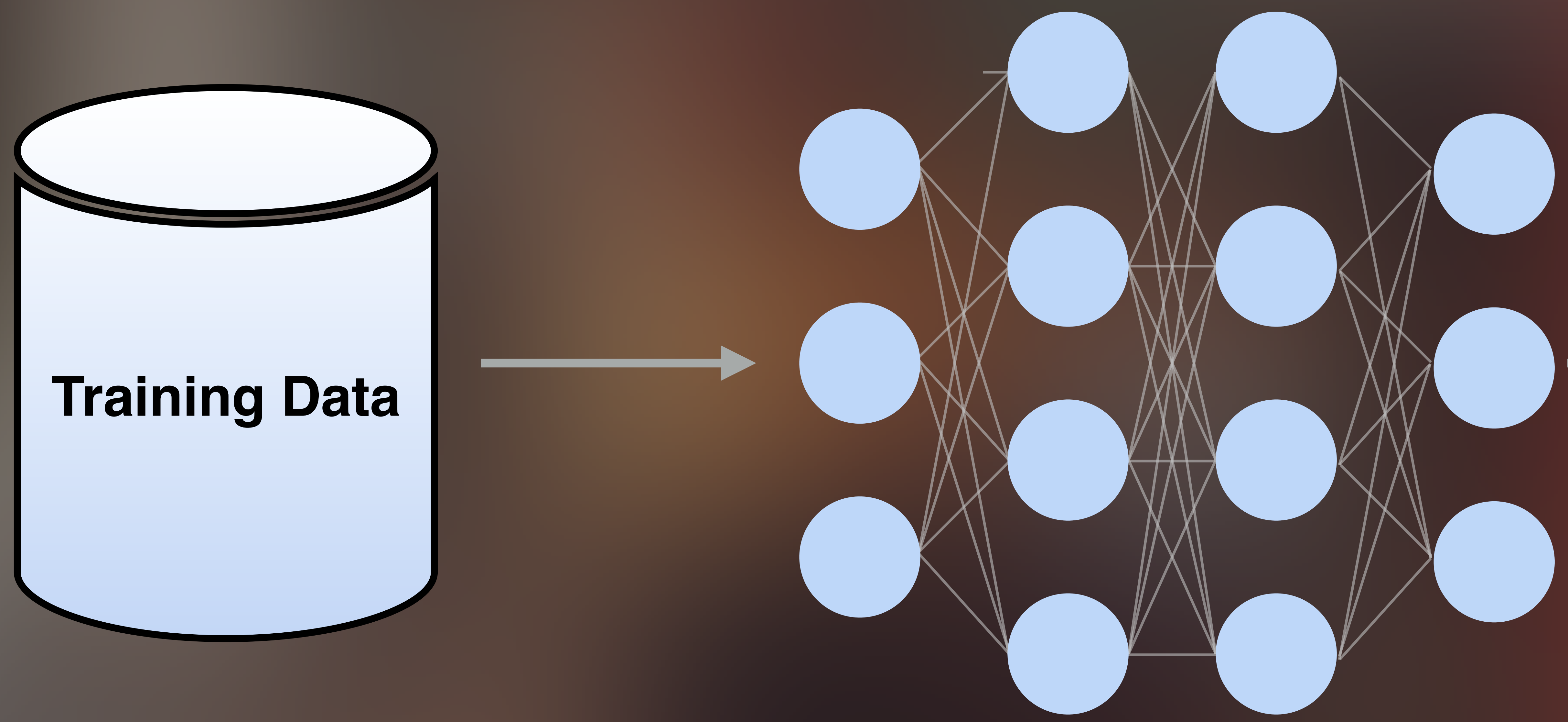
## **AI in the Real World**

University of Wisconsin-Madison

**Spring 2022**

# A running example

Food Image Classifier




# Basic steps to build an ML system

# The steps overview

- Step 1: collect data
- Step 2: look at your data
- Step 3: Create train/dev/test splits
- Step 4: build model
- Step 5: Evaluate your model
- Step 6: Diagnose error and repeat

# Acquire and annotate data

Data should be **diverse**

(annotation can be expensive)

# Data should be **realistic**

Ideal data sampled from the distribution your product will be run on.

Real photo taken by users

Professional ads photo

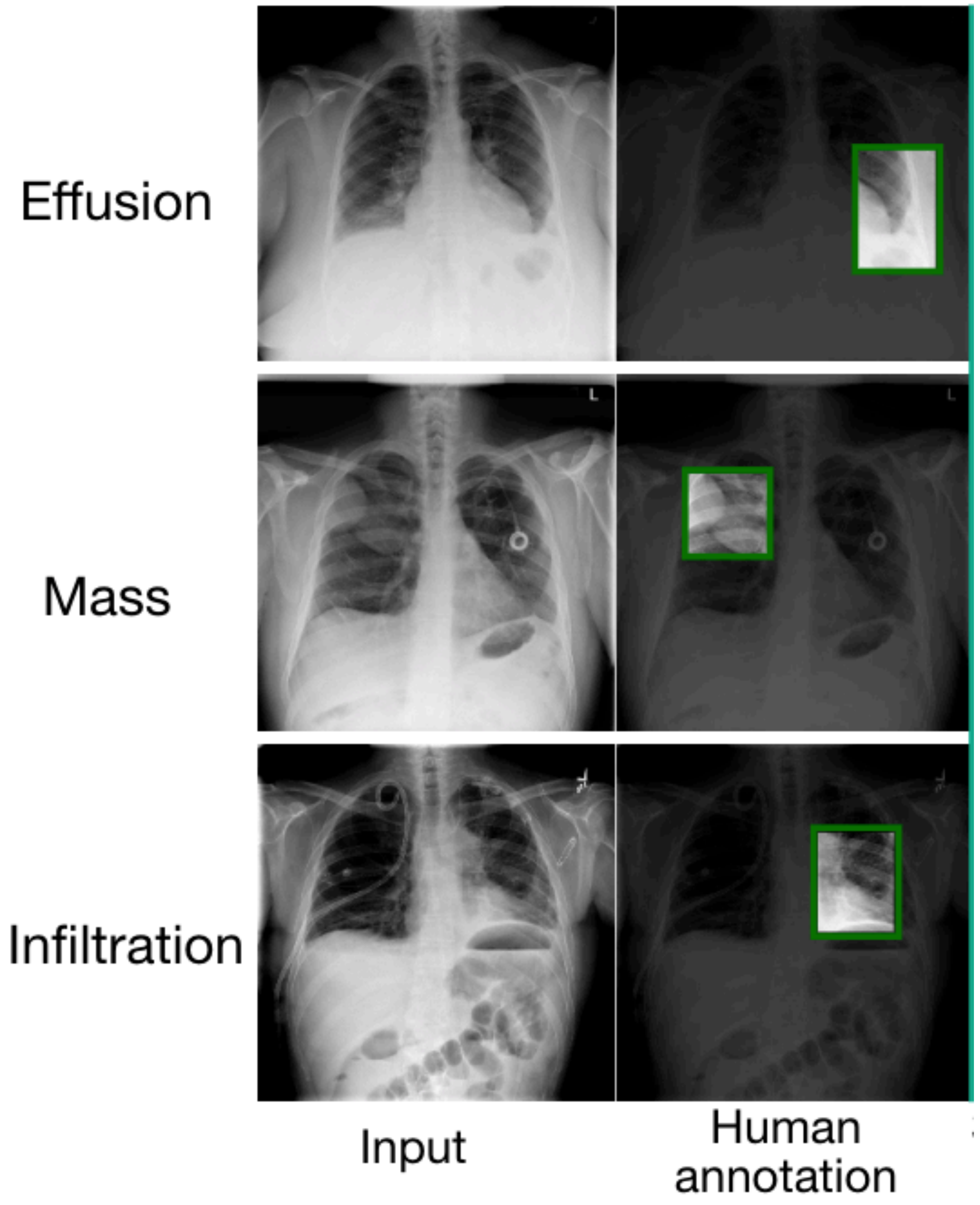
# Look at your data.

# Look at your data.

- You have some food images, take a closer look at them!
- Food from Europe different than from Africa? from Asia?
- Any potential bias in your data?
- Have the right people look at your data.
- Do this at every stage!

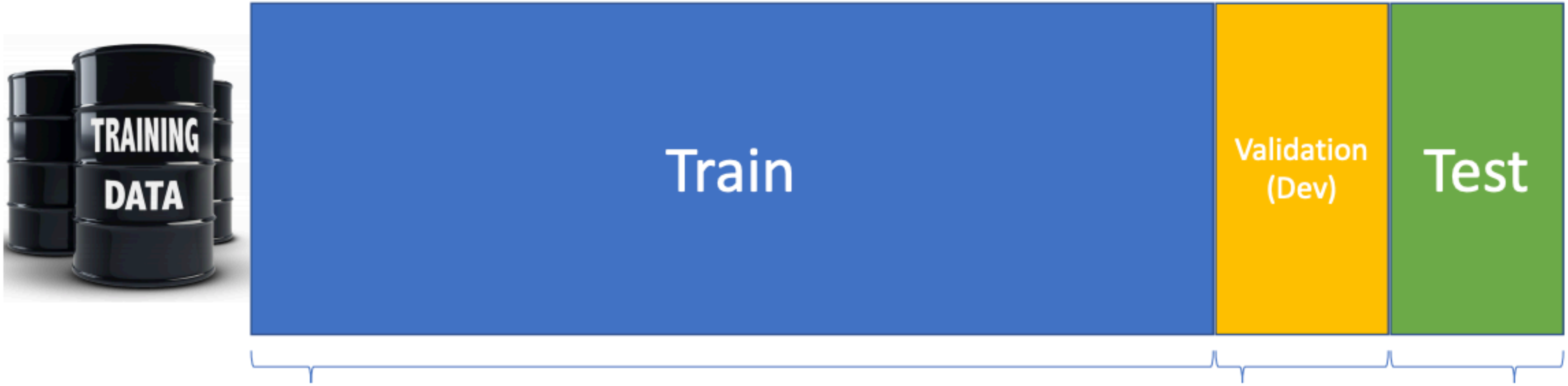
# Expertise sometimes can be required

- Biomedical imaging annotation can be expensive
- Professionally trained radiologists
- Domain knowledge



# Train/Dev/Test Split

# Partitioning Data: Train, Test, and Validation




(1) Fit model to the training dataset

(2) Fit hyperparameters to the *validation* (or *development*) dataset

(3) Test model performance on the test set

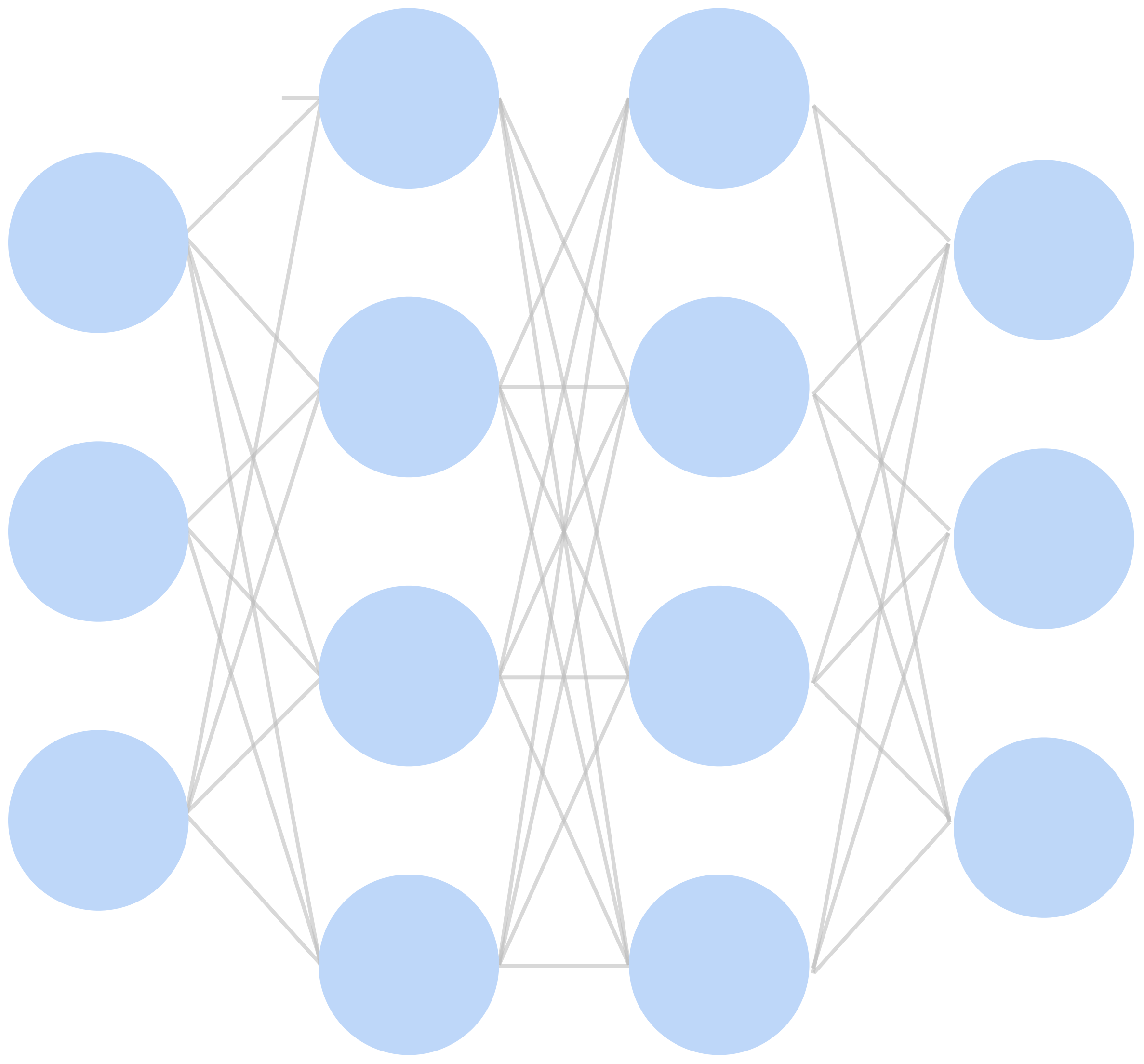
Slides credit: Chris Ré, Stanford CS229

# What makes a good split?

- **Ideal**: Train, test, & dev randomly sampled
  - Allows us to say train quality is approximately test quality
- Test is a **proxy** for the real world!
  - We'll talk more about this later...
- **Challenge**: Leakage.
  - (Nearly) same example in train and dev.
  - Causes performance to be overstated!
    - Eg., same senders in train and test?

Slides credit: Chris Ré, Stanford CS229

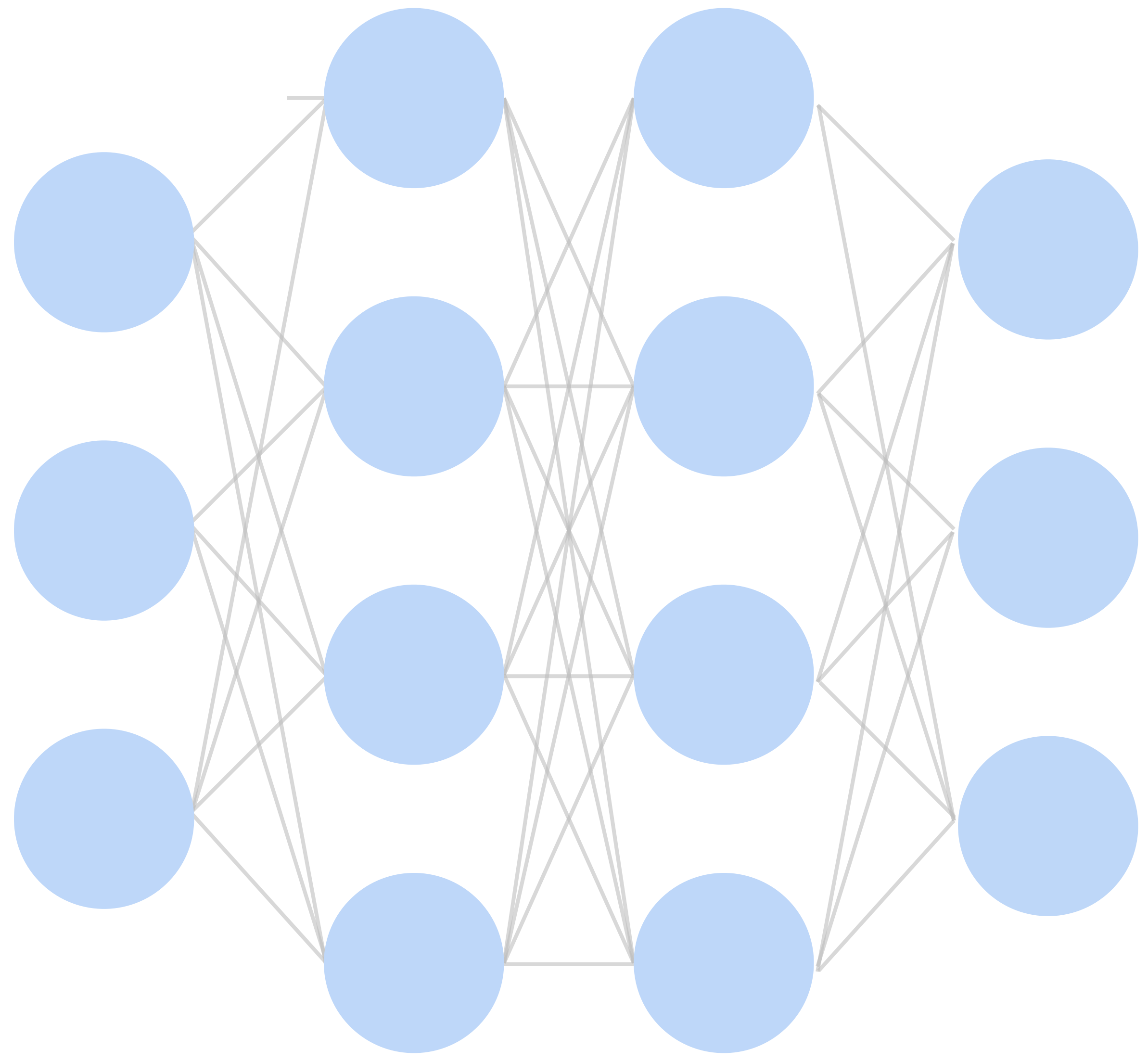
# Build your model.



# Build your model.

- A bag of learning algorithms learned from class.

- Simple model vs. deep models

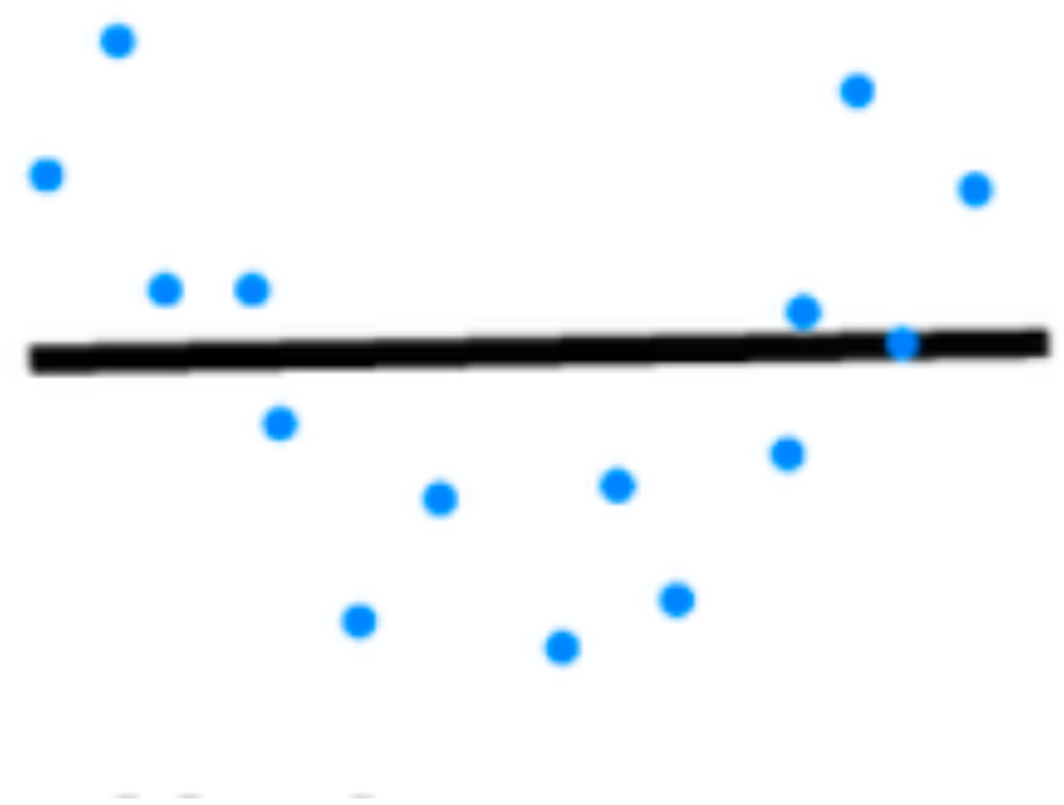


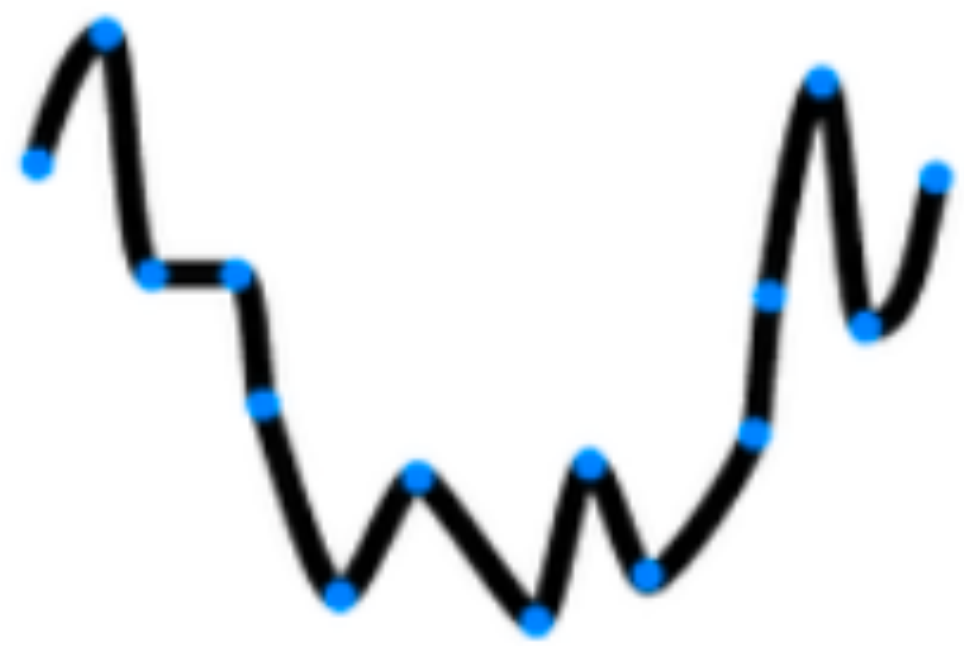
# Underfitting
# Overfitting


Image credit: hackernoon.com

# Model Capacity

- The ability to fit variety of functions
- Low capacity models struggles to fit training set
  - Underfitting
- High capacity models can memorize the training set
  - Overfitting





# Underfitting and Overfitting

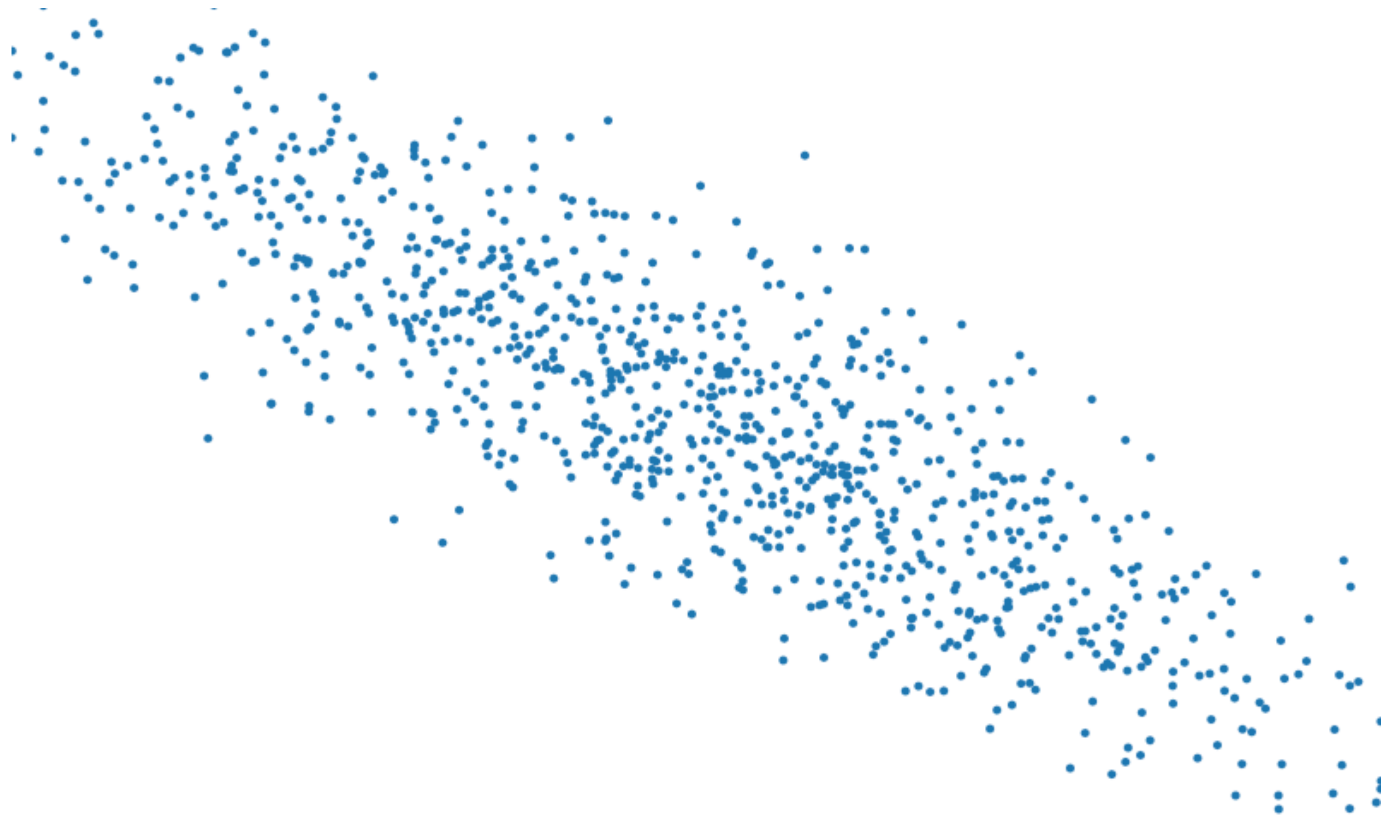
**Data complexity**

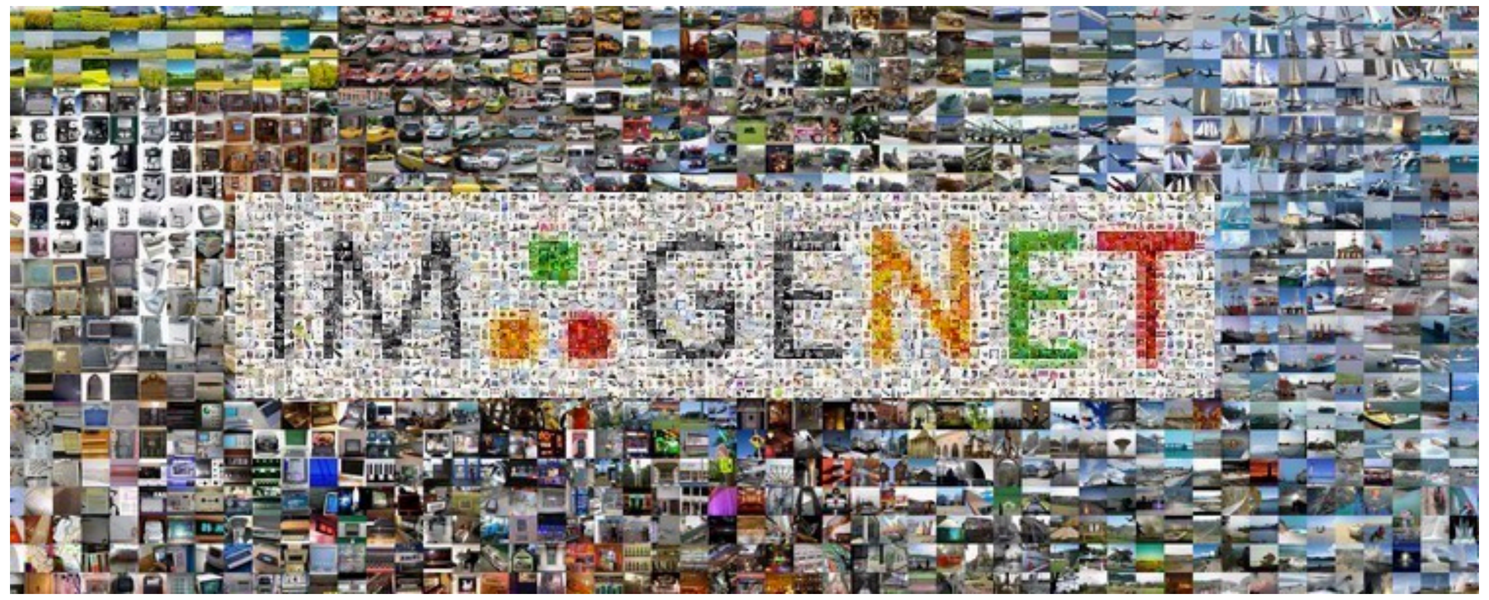
**Model capacity**

| | Simple | Complex |
|---|---|---|
| Low | Normal | Underfitting |
| High | Overfitting | Normal |

# Data Complexity

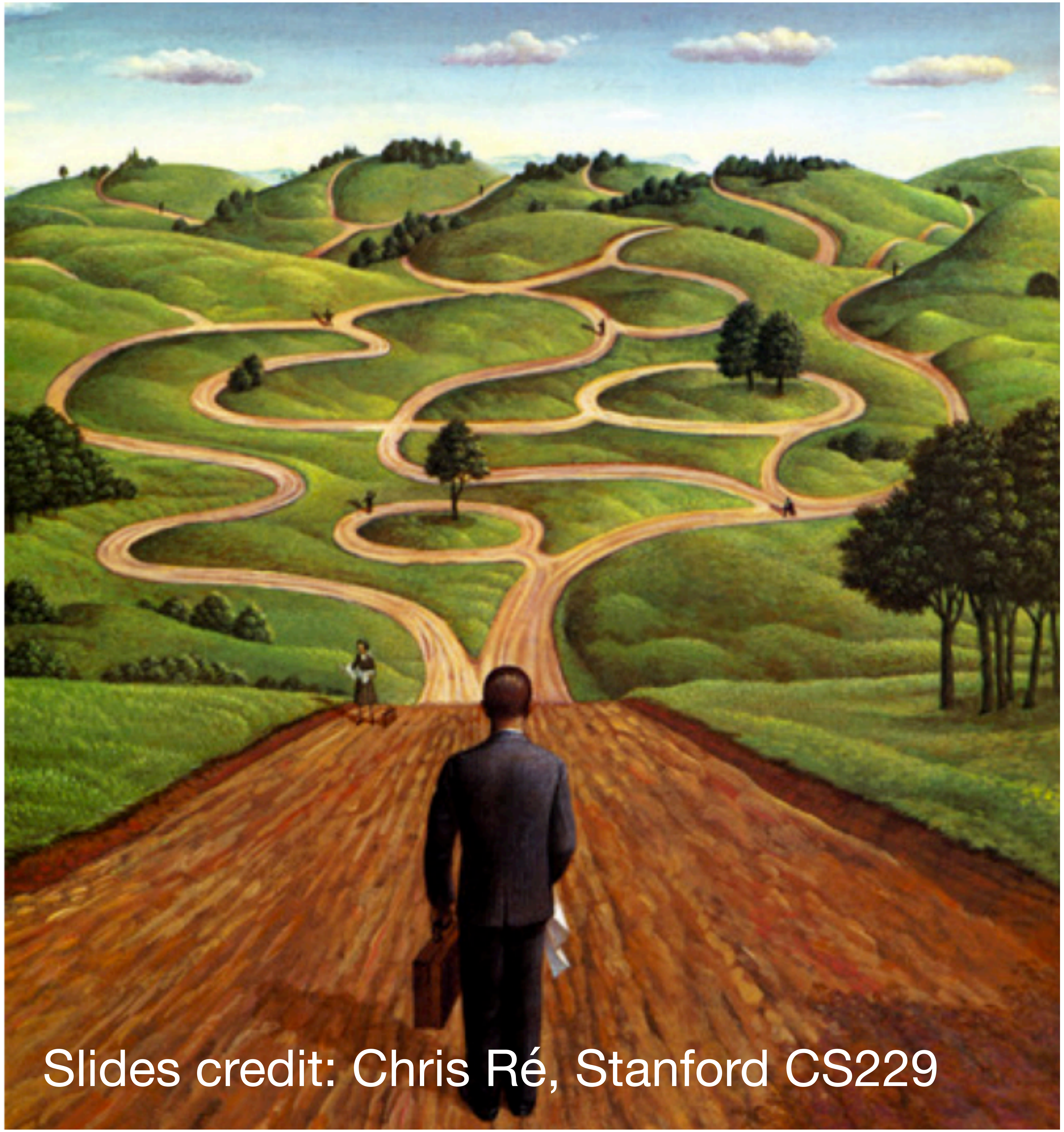
- Multiple factors matters
  - # of examples
  - # of features in each example
  - time/space structure
  - # of labels





# Ablation studies.

- You've built up a model, it has many different components.
  - Which matter?
  - which are stable?
- Remove one feature at a time!
  - *Adding* features + baseline could overestimate overlap. How?
- Measure performance.
  - Critical for research!

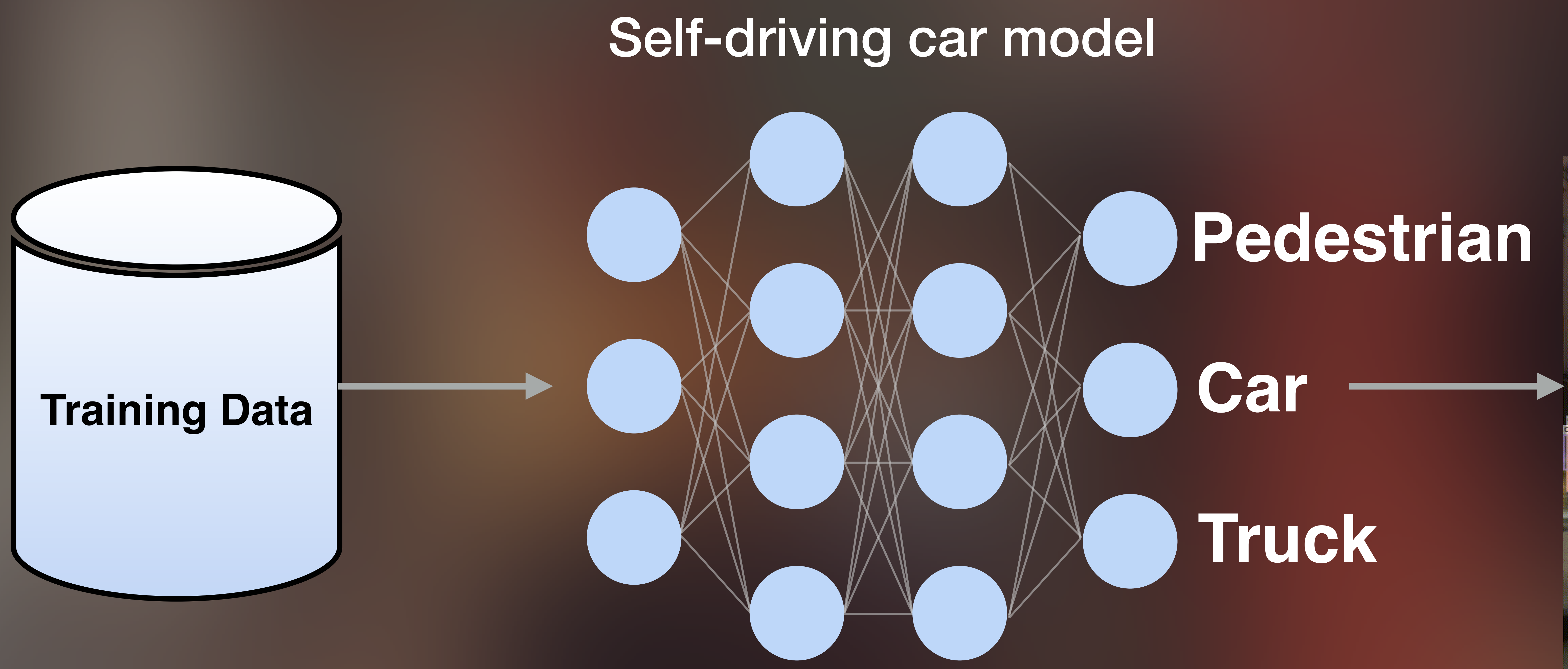


# Diagnose the error

**(inspect the data where the model makes mistakes)**

# Open-world Machine Learning


Video source: Berkeley Deep Drive-100k

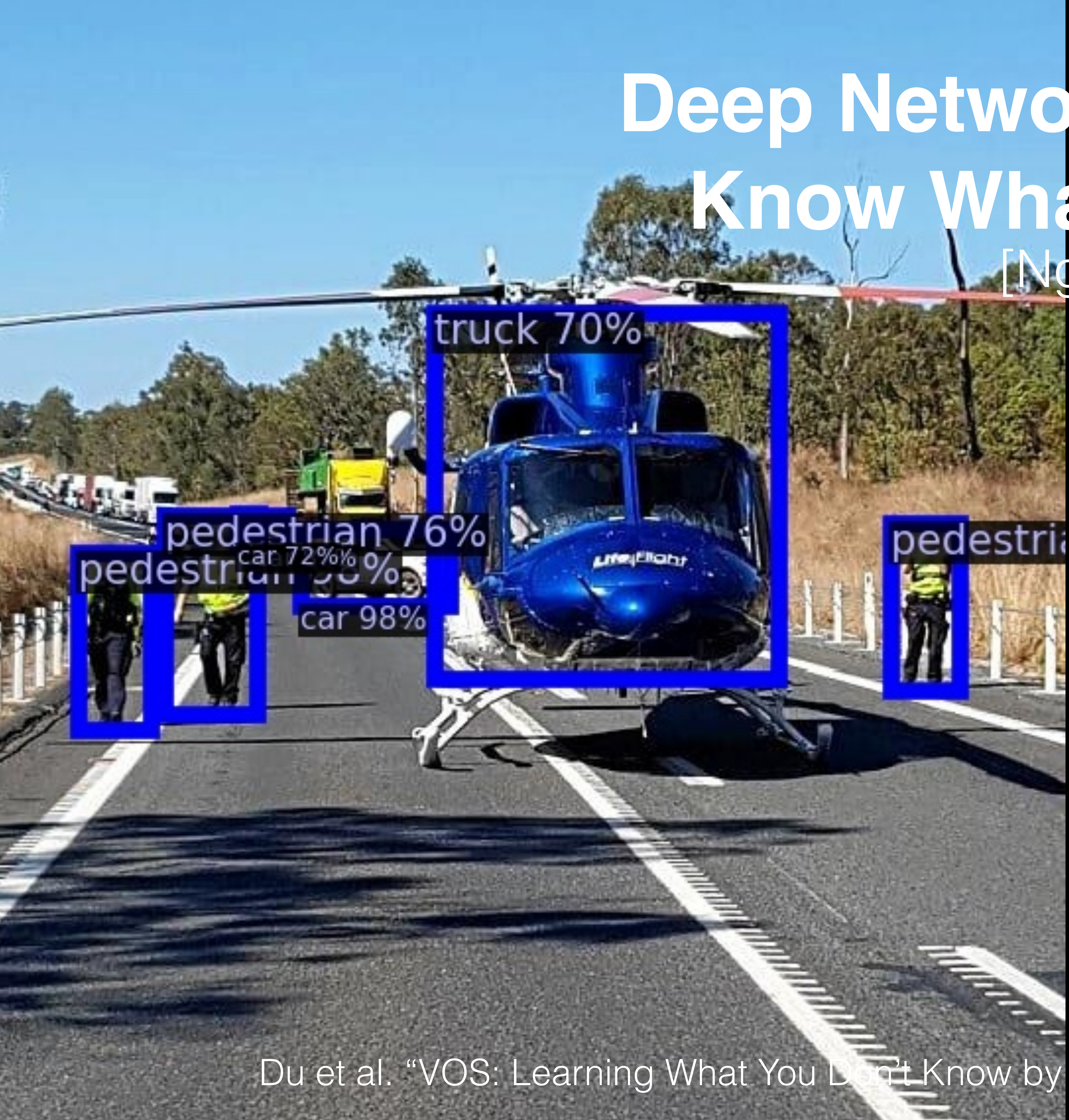


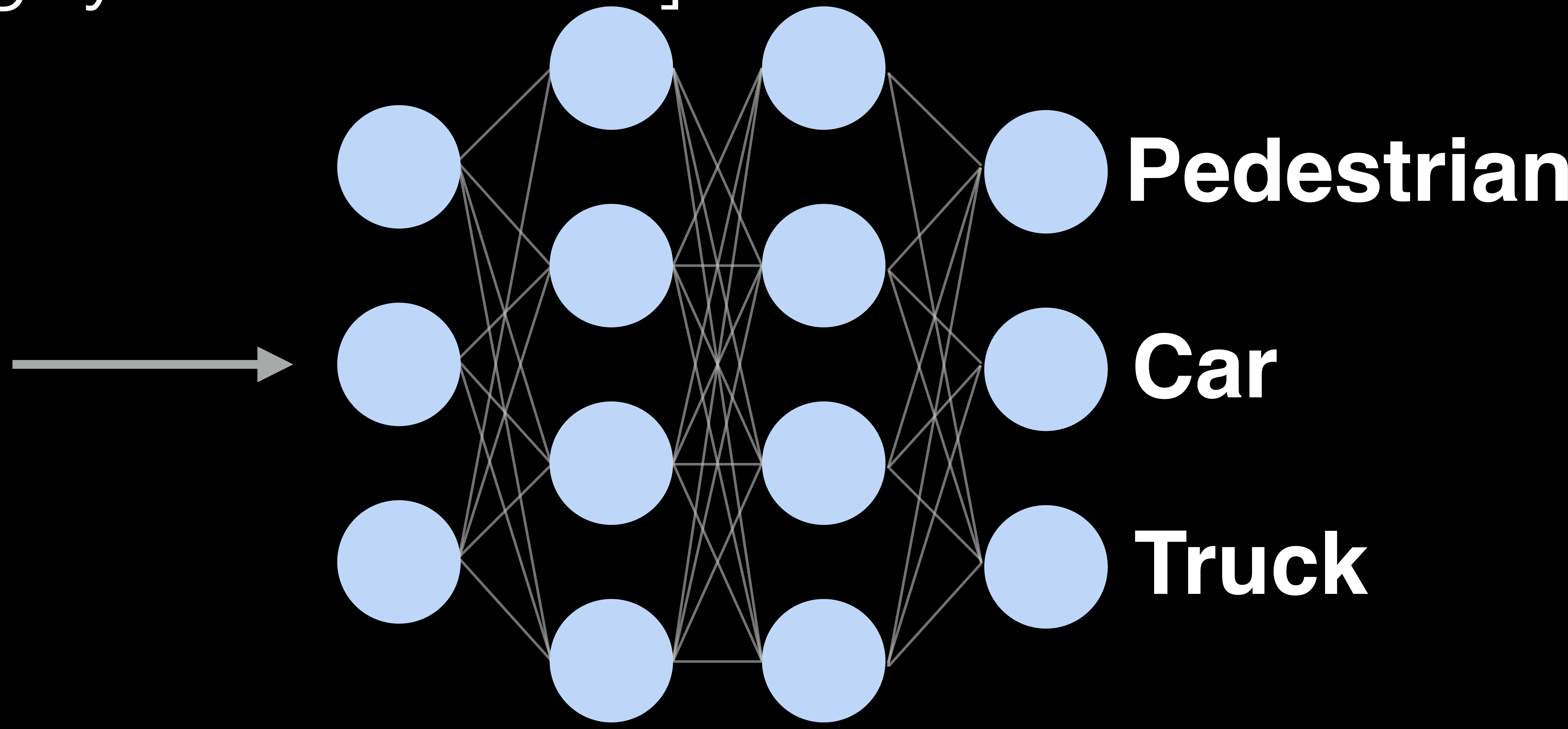

**Closed**-world**:** Training and testing distributions **match**

**Open**-world**:** Training and testing distributions **differ,** unknowns can emerge

# Deep Networks Do Not Necessarily Know What They Don't Know…

[Nguyen et al. 2015]

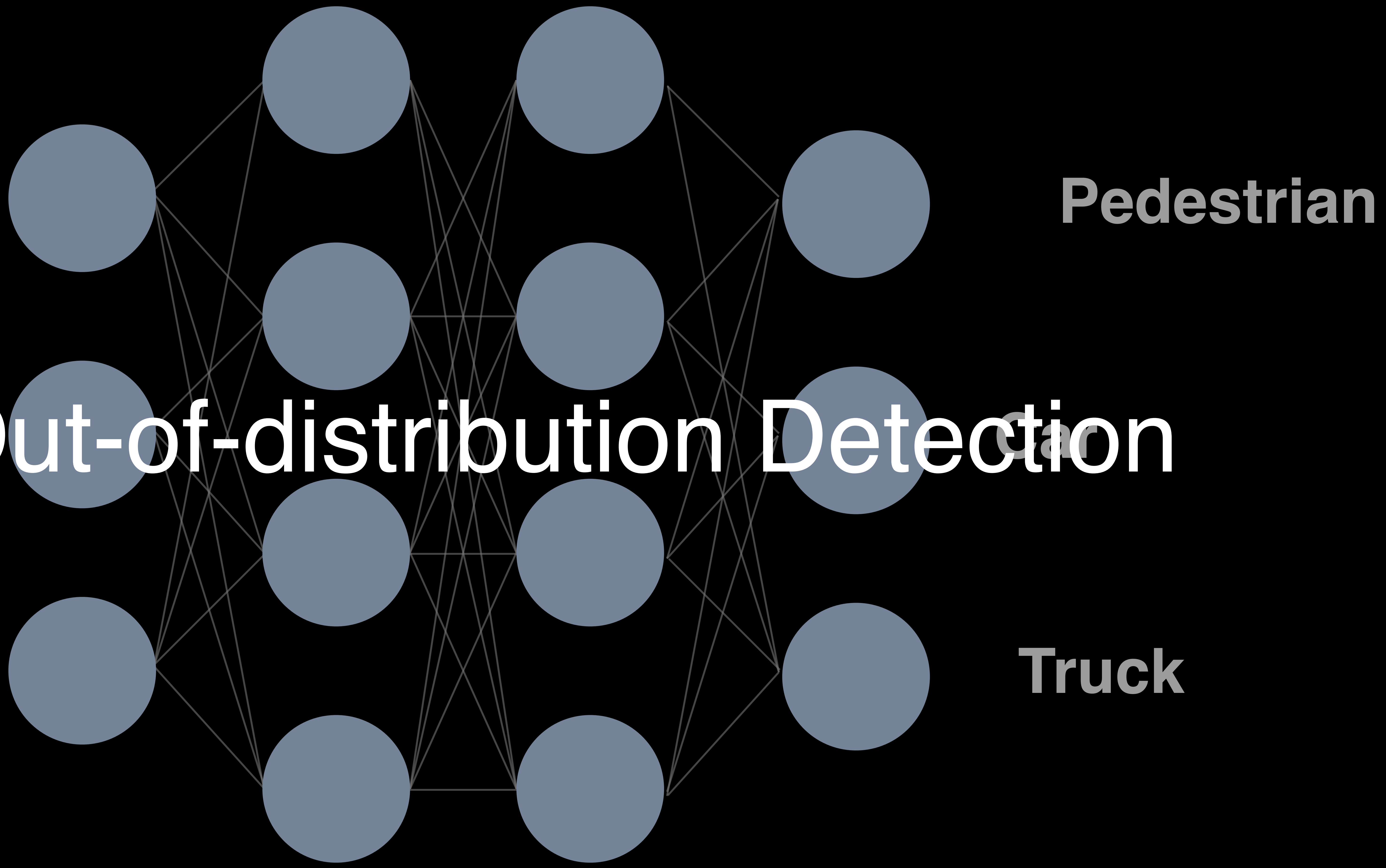



Model trained on BDD dataset produces overconfident predictions for unknown object "helicopter"

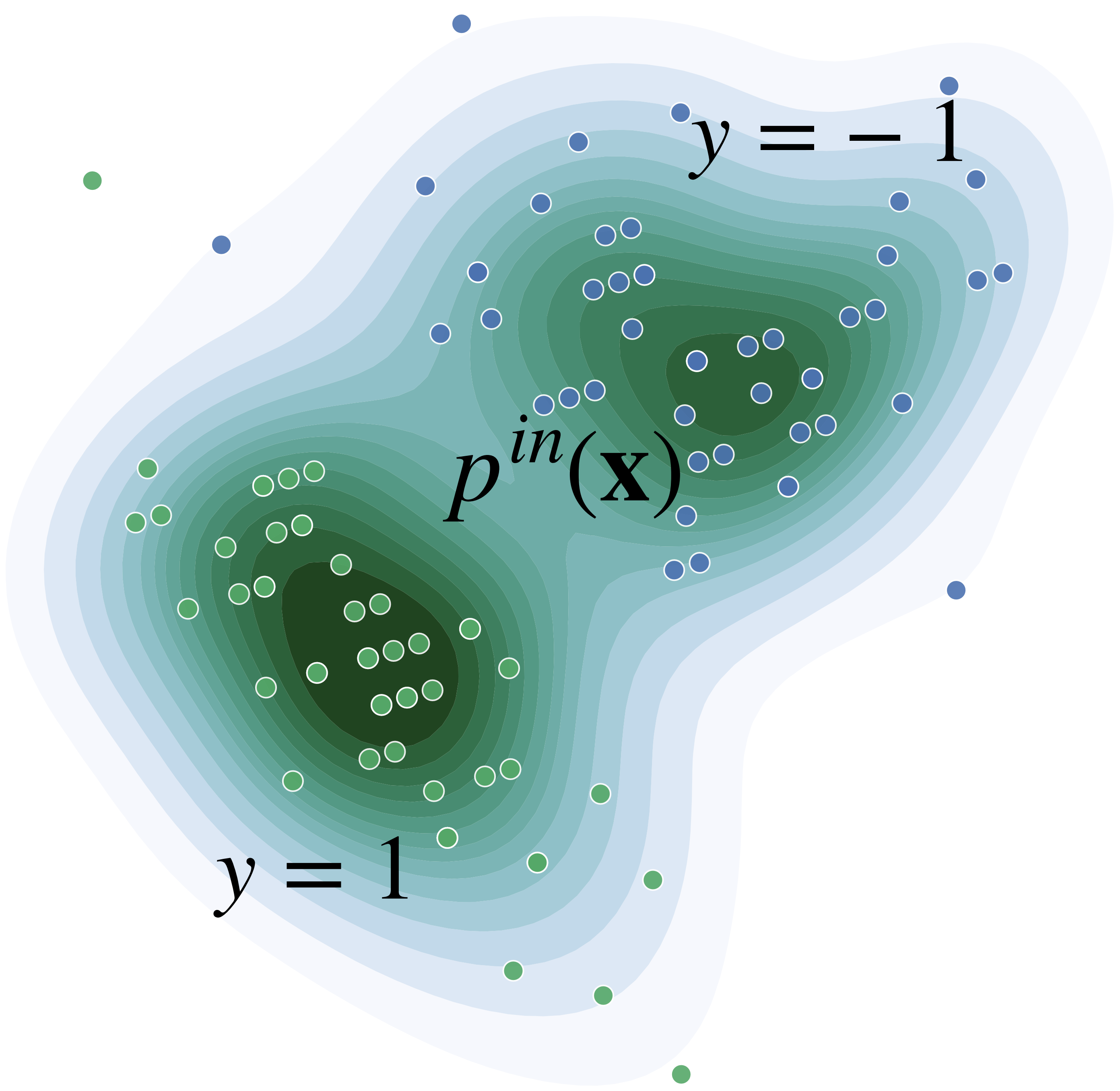
Du et al. "VOS: Learning What You Don't Know by Virtual Outlier Synthesis", ICLR, 2022

# Out-of-distribution Detection



# Out-of-distribution Detection: A Simple View

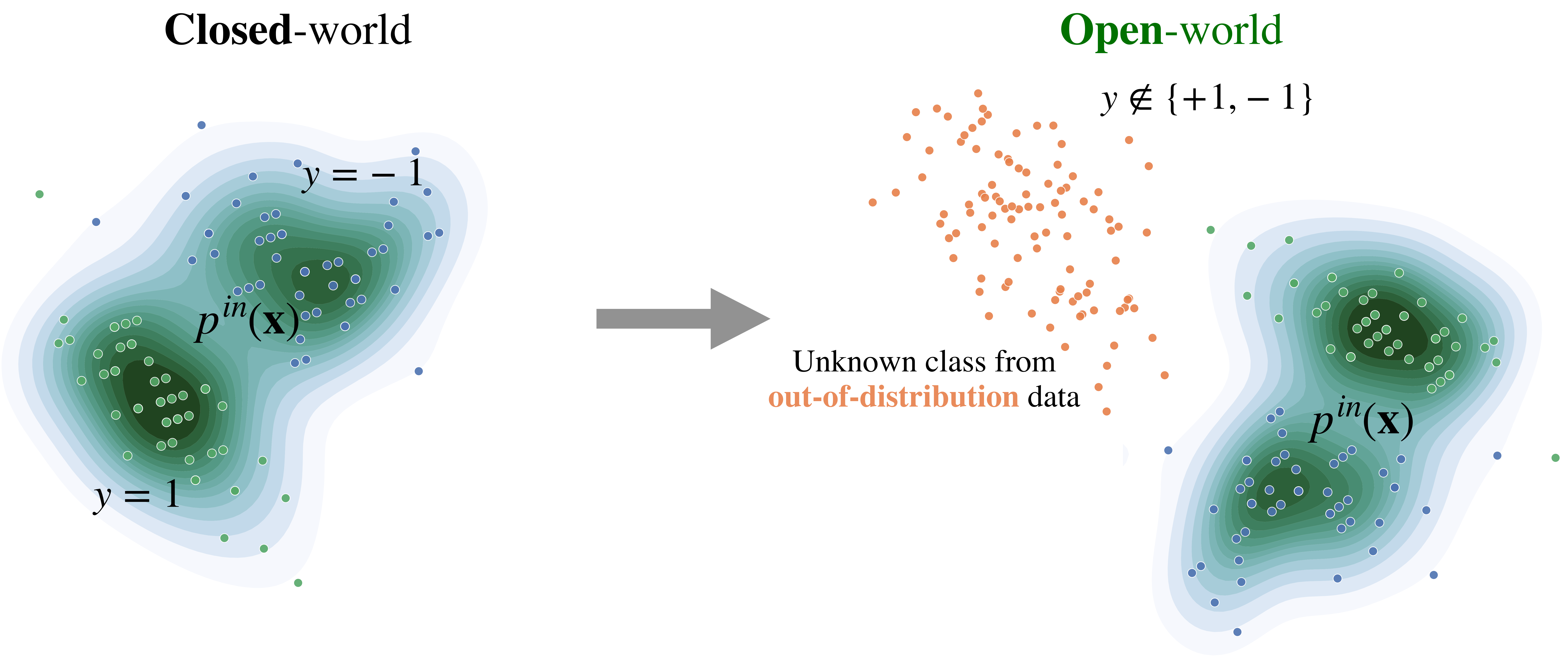
**Closed**-world



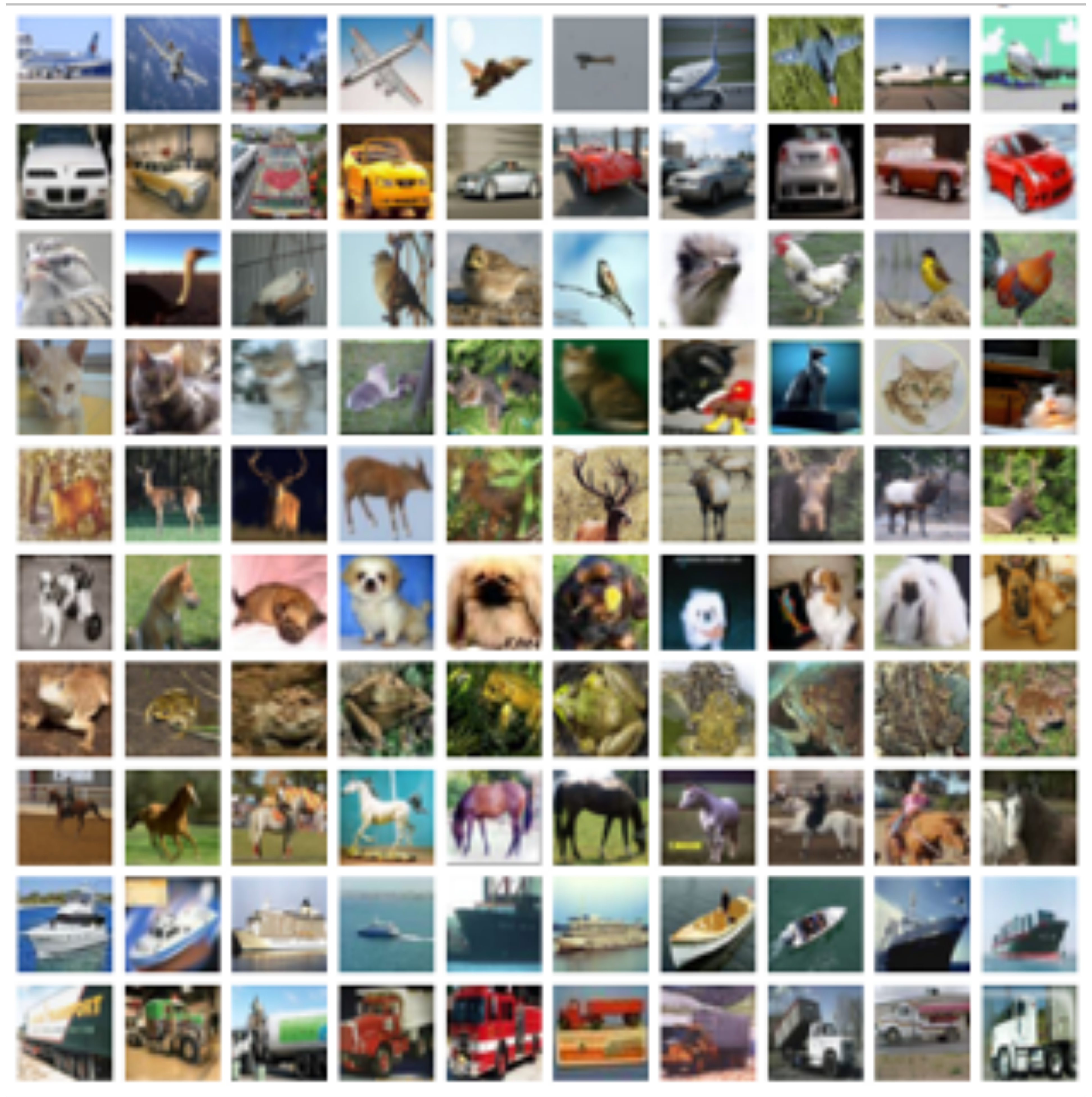

Input space: $\mathcal{X} = \mathbb{R}^d$

Label space: $\mathcal{Y} = \{1, -1\}$

# Out-of-distribution Detection: A Simple View



# Out-of-distribution Detection: A Simple View

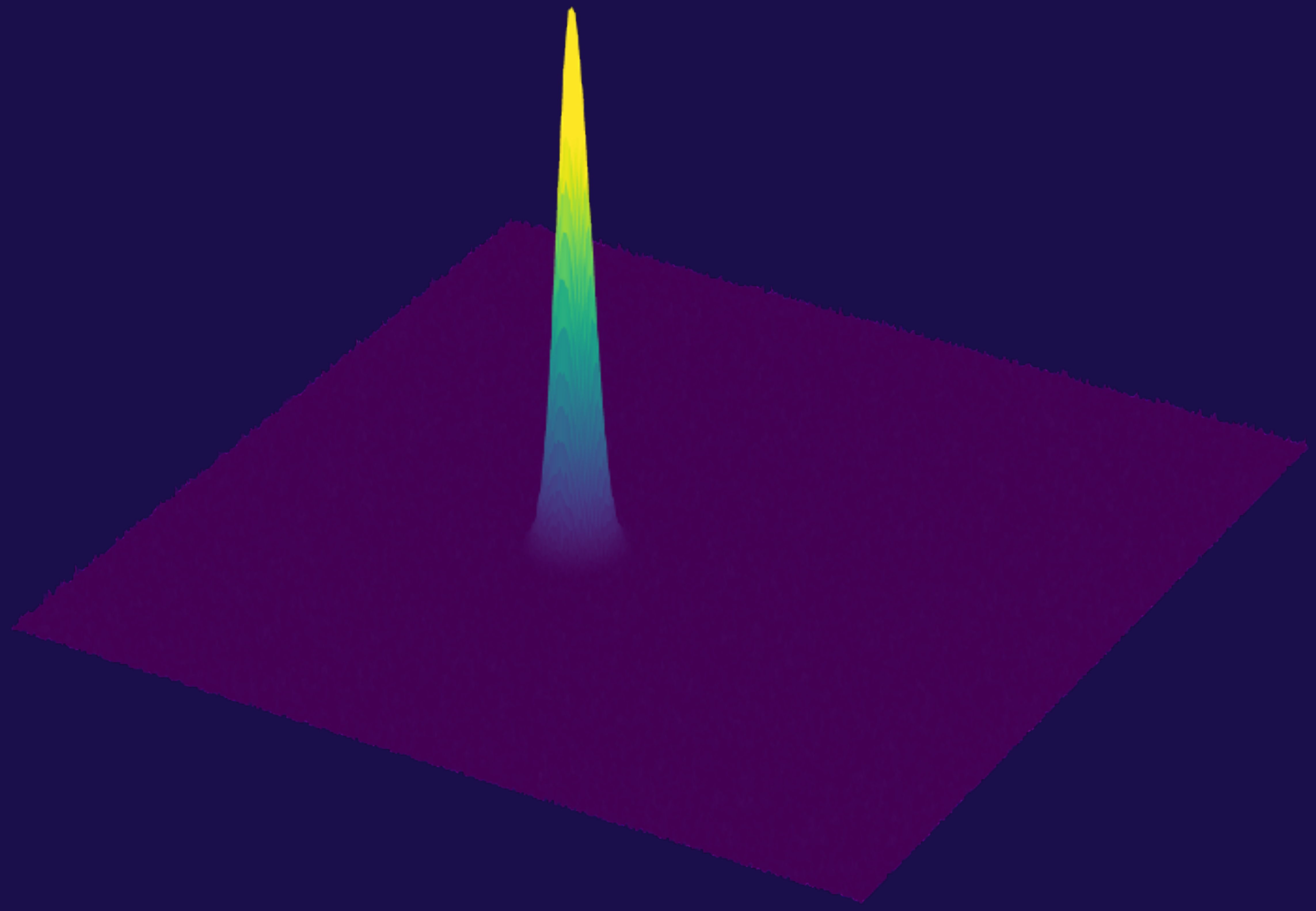


CIFAR-10 (in-distribution)

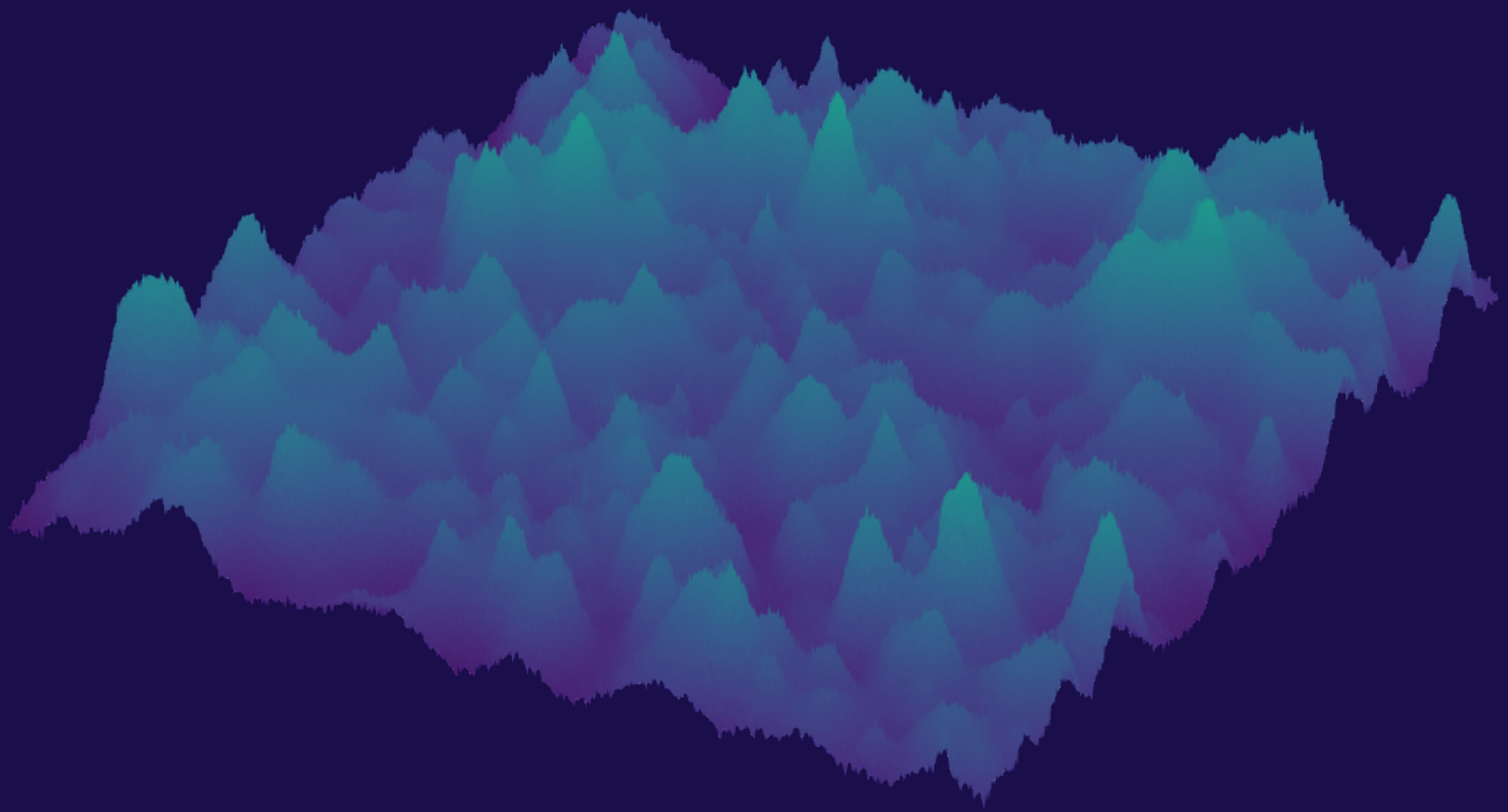
SVHN (OOD)

# Out-of-distribution Detection: A Simple View

CIFAR-10



The Internet



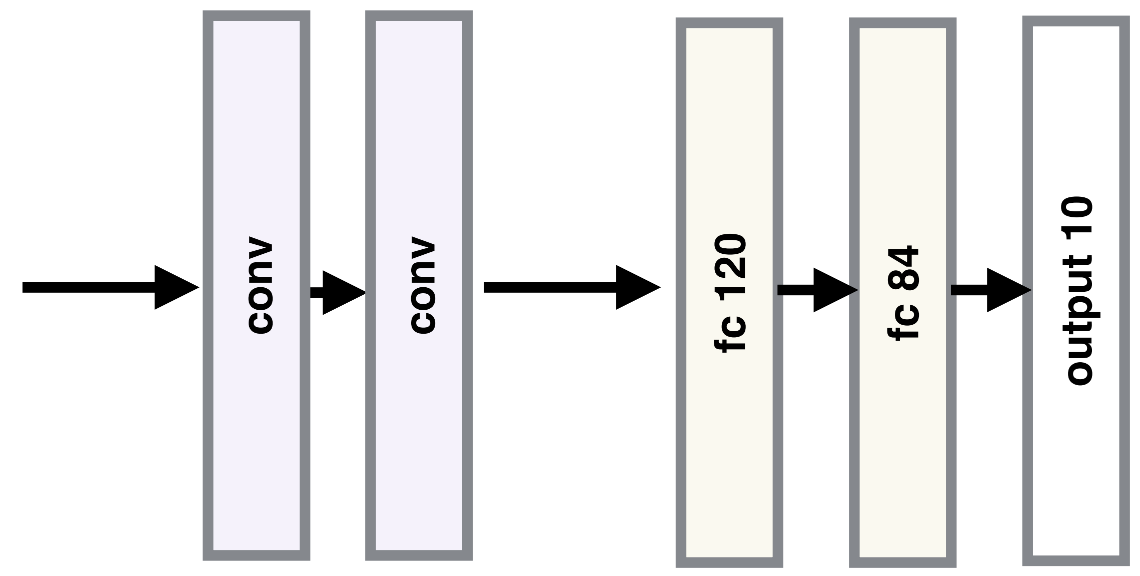
Slide from OpenAI

# The steps overview

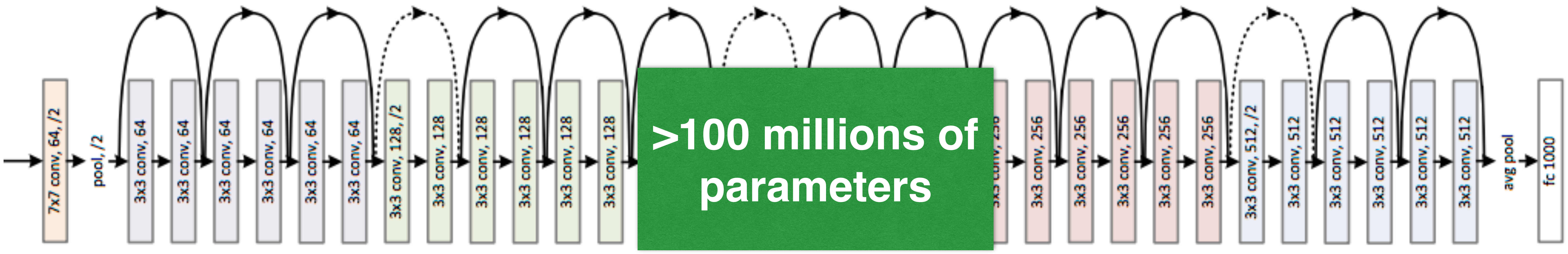
- Step 1: collect data
- Step 2: look at your data
- Step 3: Create train/dev/test splits
- Step 4: build model
- Step 5: Evaluate your model
- Step 6: Diagnose error and repeat

# Industry-scale Machine Learning

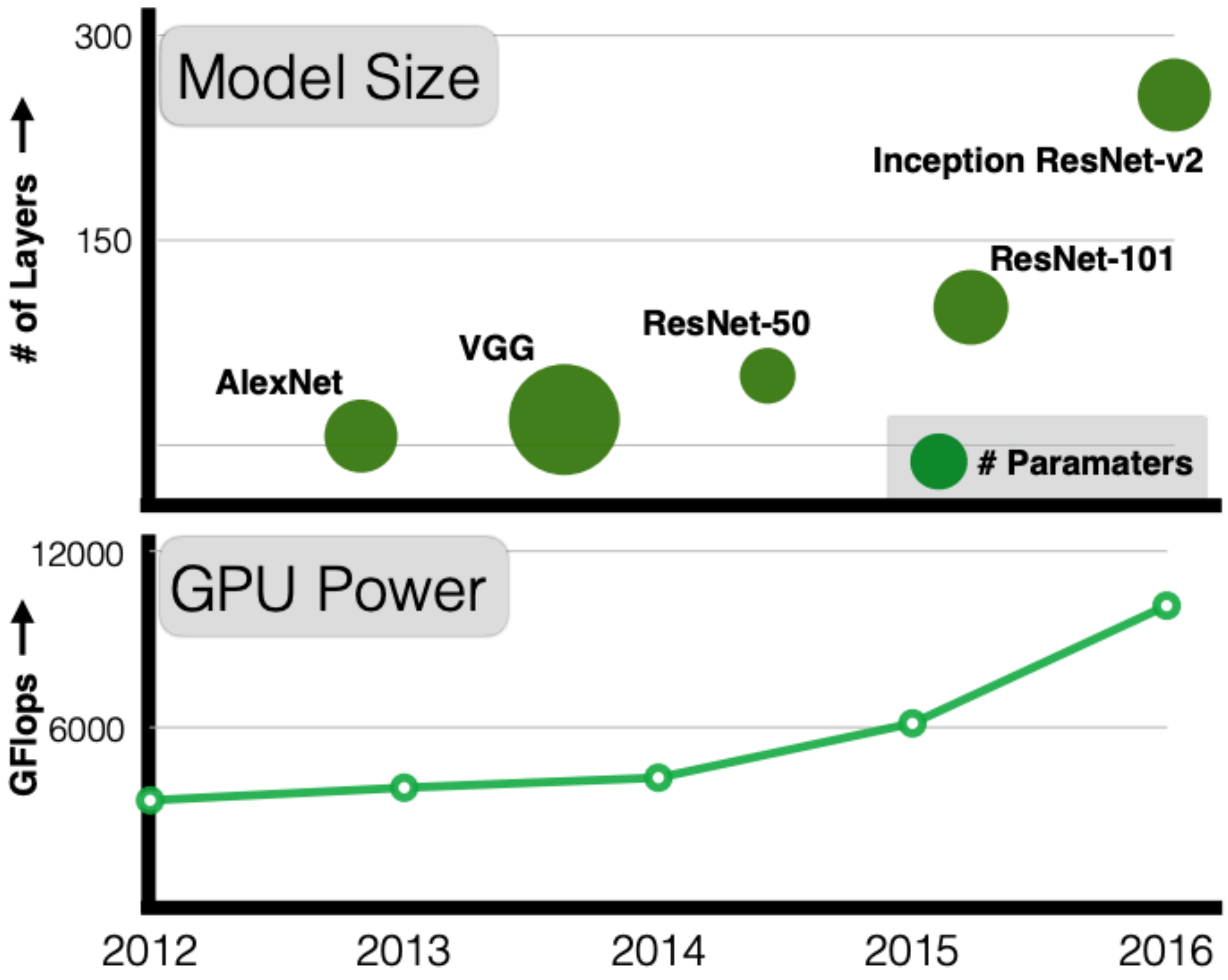
# Model Complexity Keeps Increasing



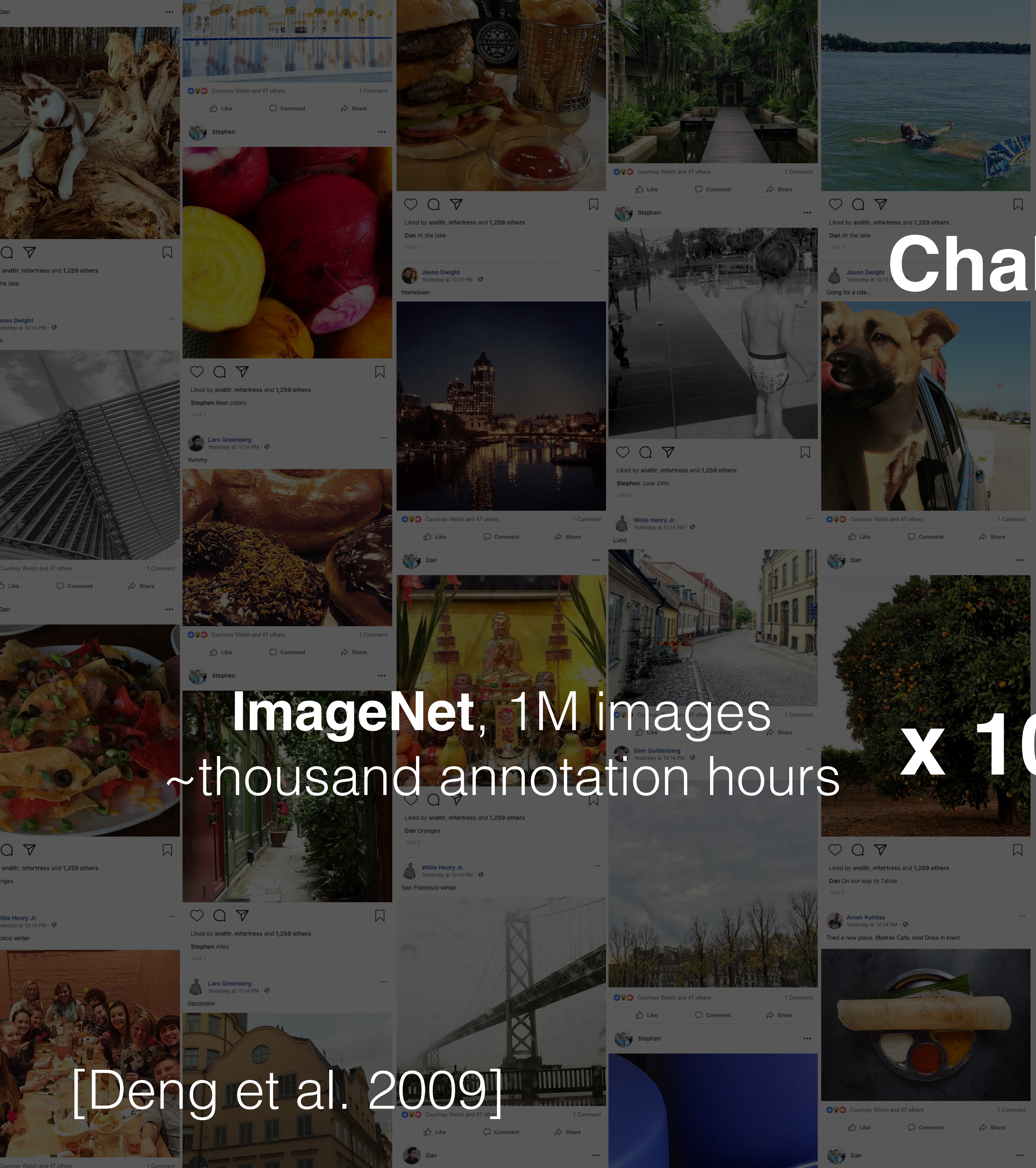

LeNet (Lecun et al. 1998)




ResNet (He et al. 2016)




[Sun et al. 2017]



# Challenge: Limited labeled data

**ImageNet**, 1M images
~thousand annotation hours

**x 1000**

1B images
~million annotation hours

[Deng et al. 2009]

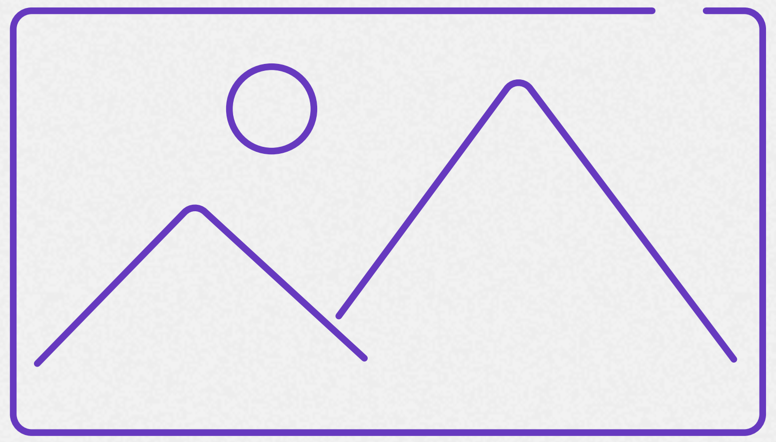
TRAINING AT SCALE

# Levels of Supervision

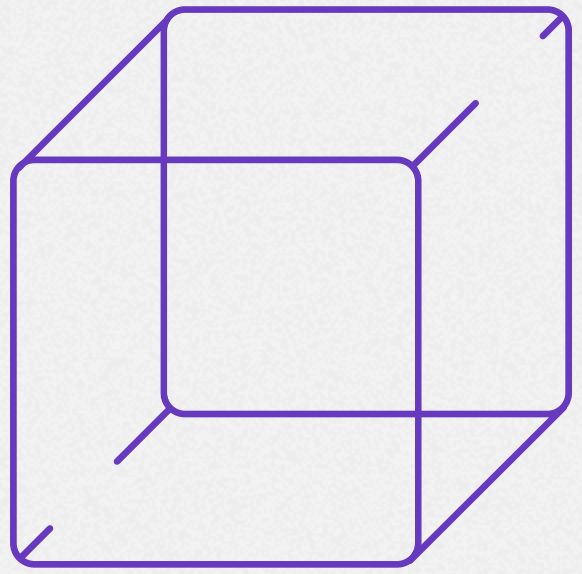
TRAINING AT SCALE

# Noisy Data

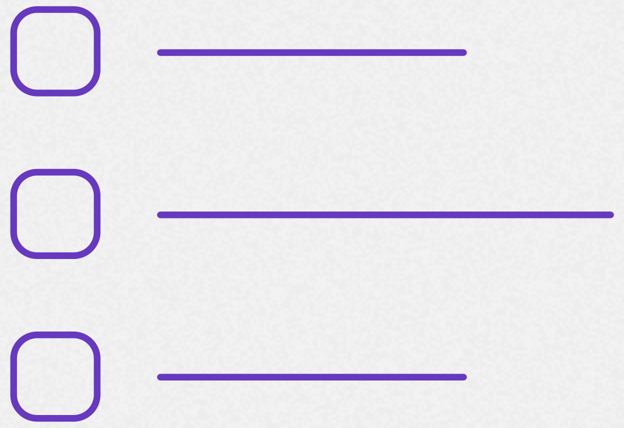
# Weakly Supervised Training



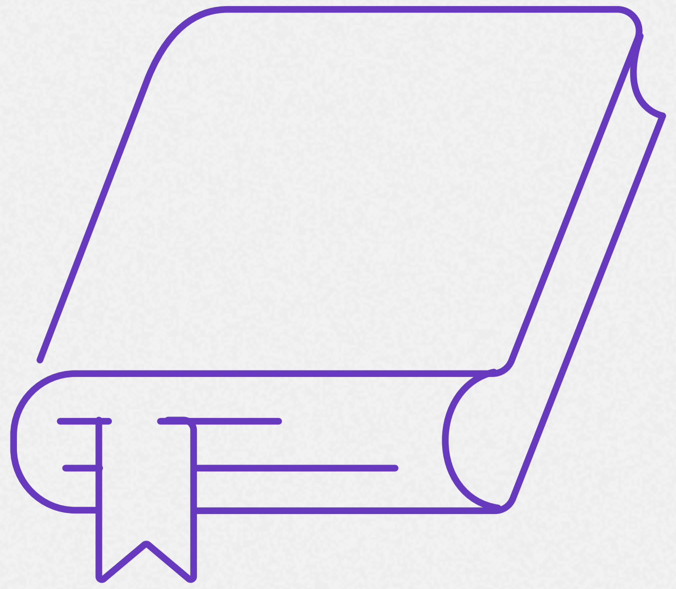
3.5B
PUBLIC INSTAGRAM
IMAGES



LARGE CAPACITY MODEL
(RESNEXT101-32X48)



17K UNIQUE LABELS



DISTRIBUTED
TRAINING
(350 GPUS)

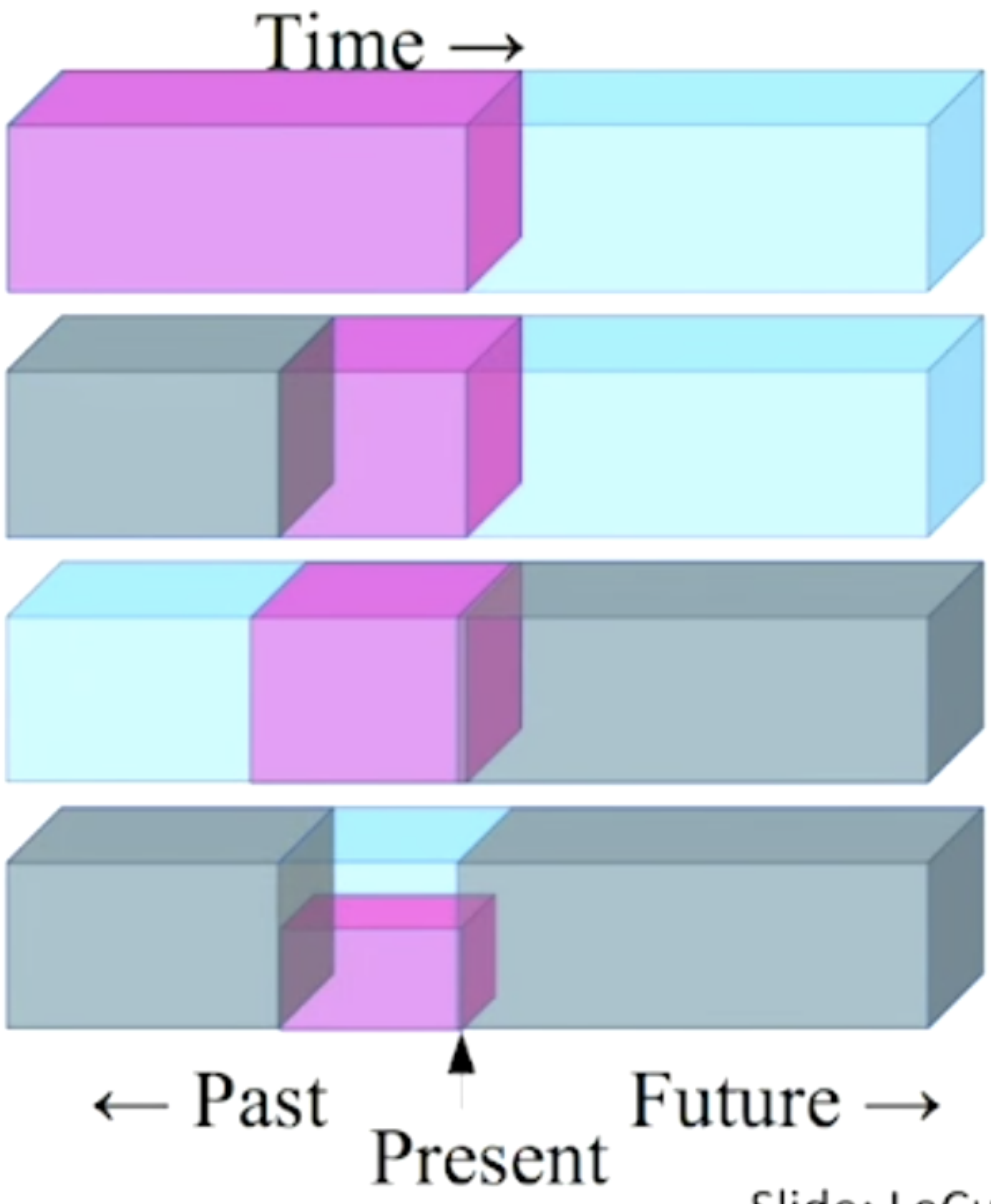
[Mahajan et al. 2018]

# Self-supervised Learning
# (no label)

What if we can get labels **for free** from unlabelled data and train unsupervised dataset in a supervised manner?

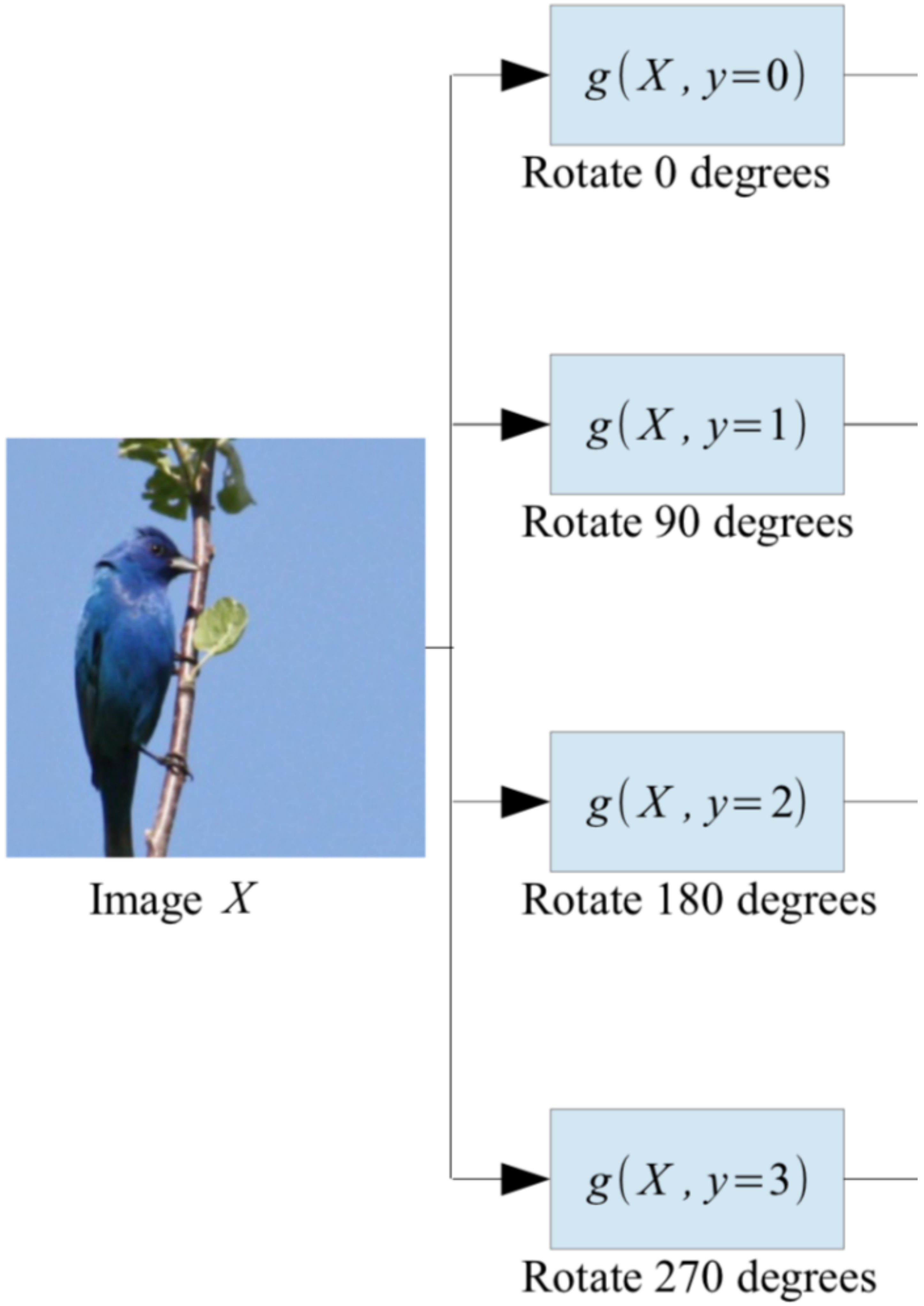
# Pretext Tasks

- 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.



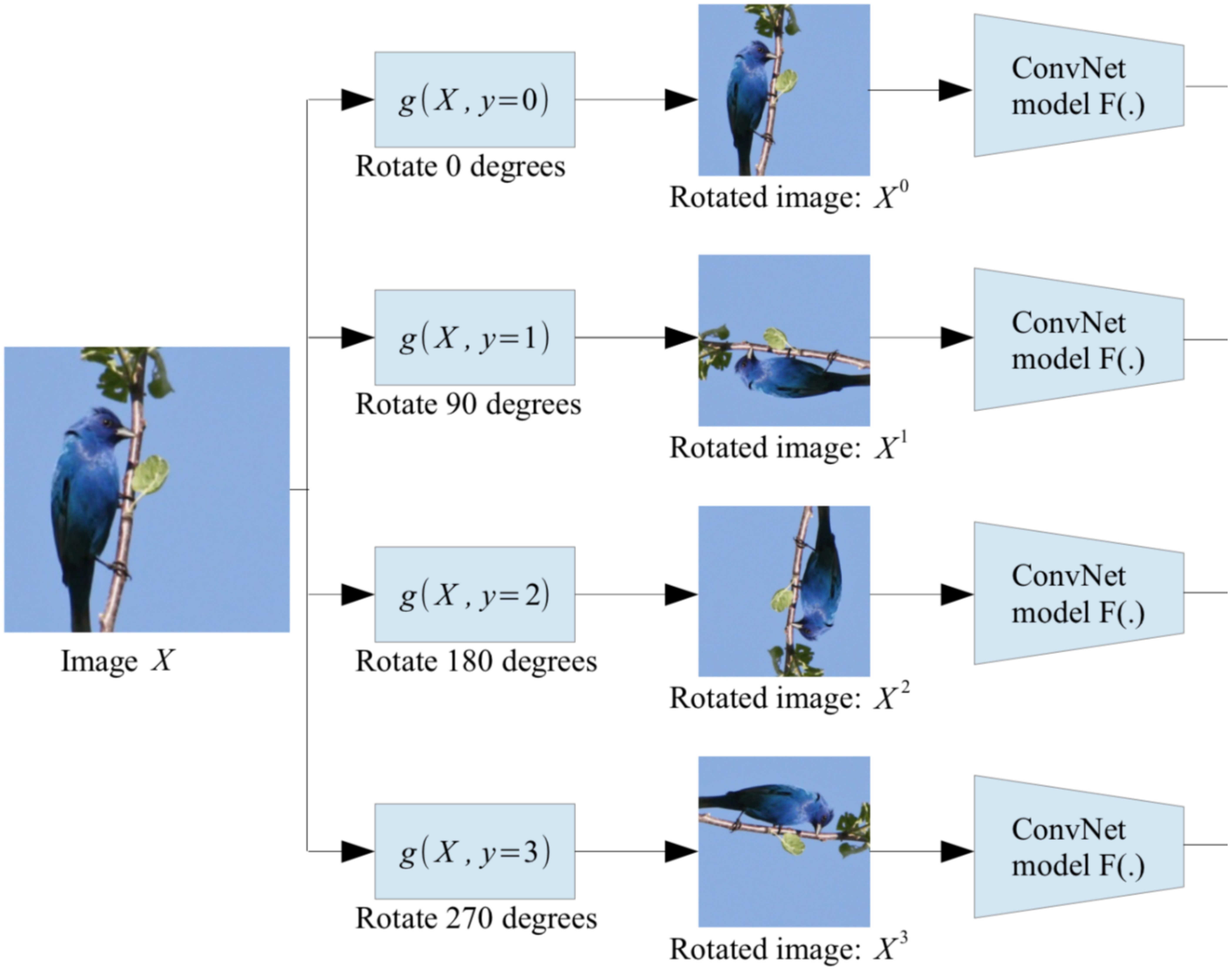

Slide: LeCun

# Rotation



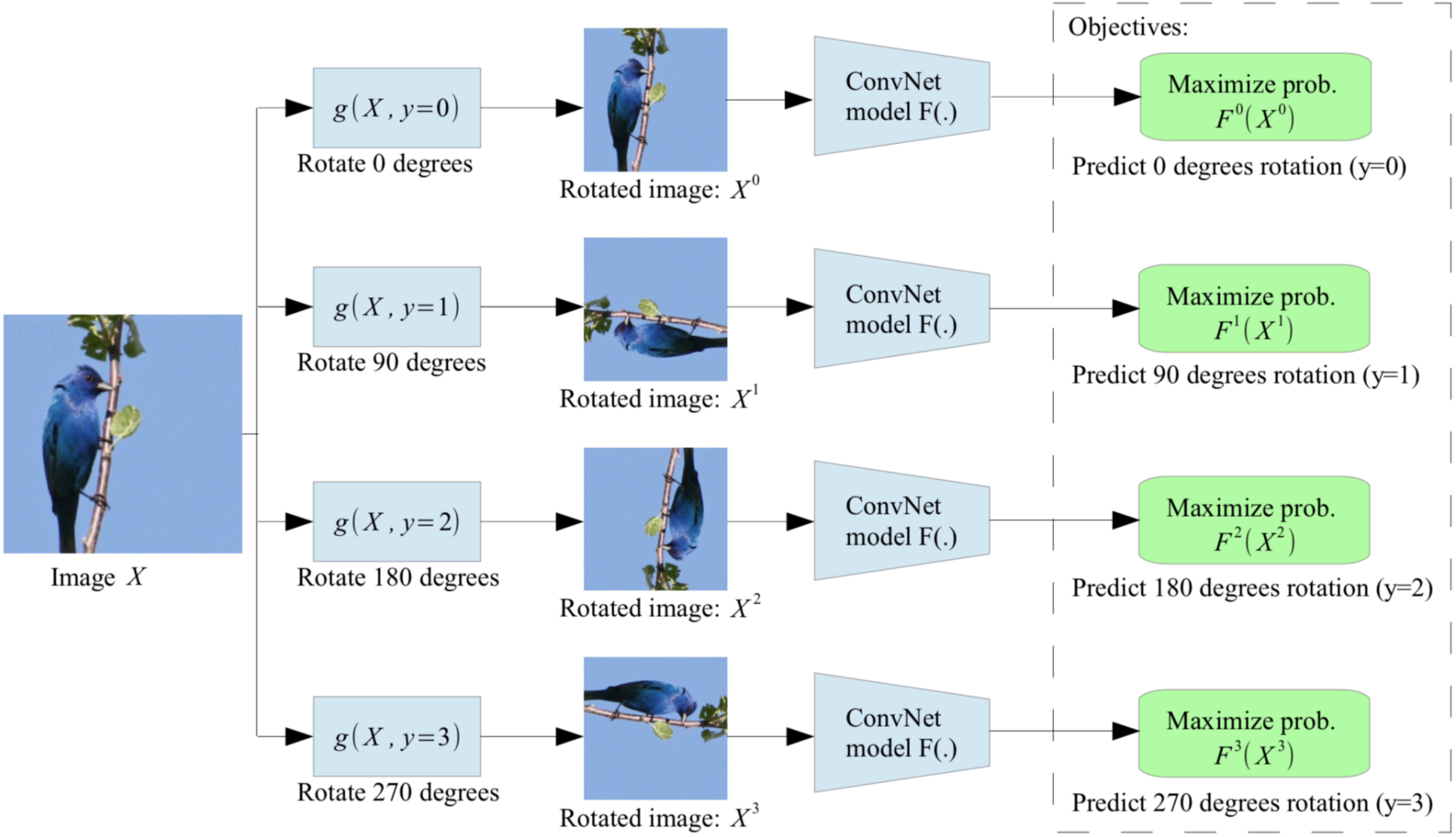

[Gidaris et al. 2018]

# Rotation



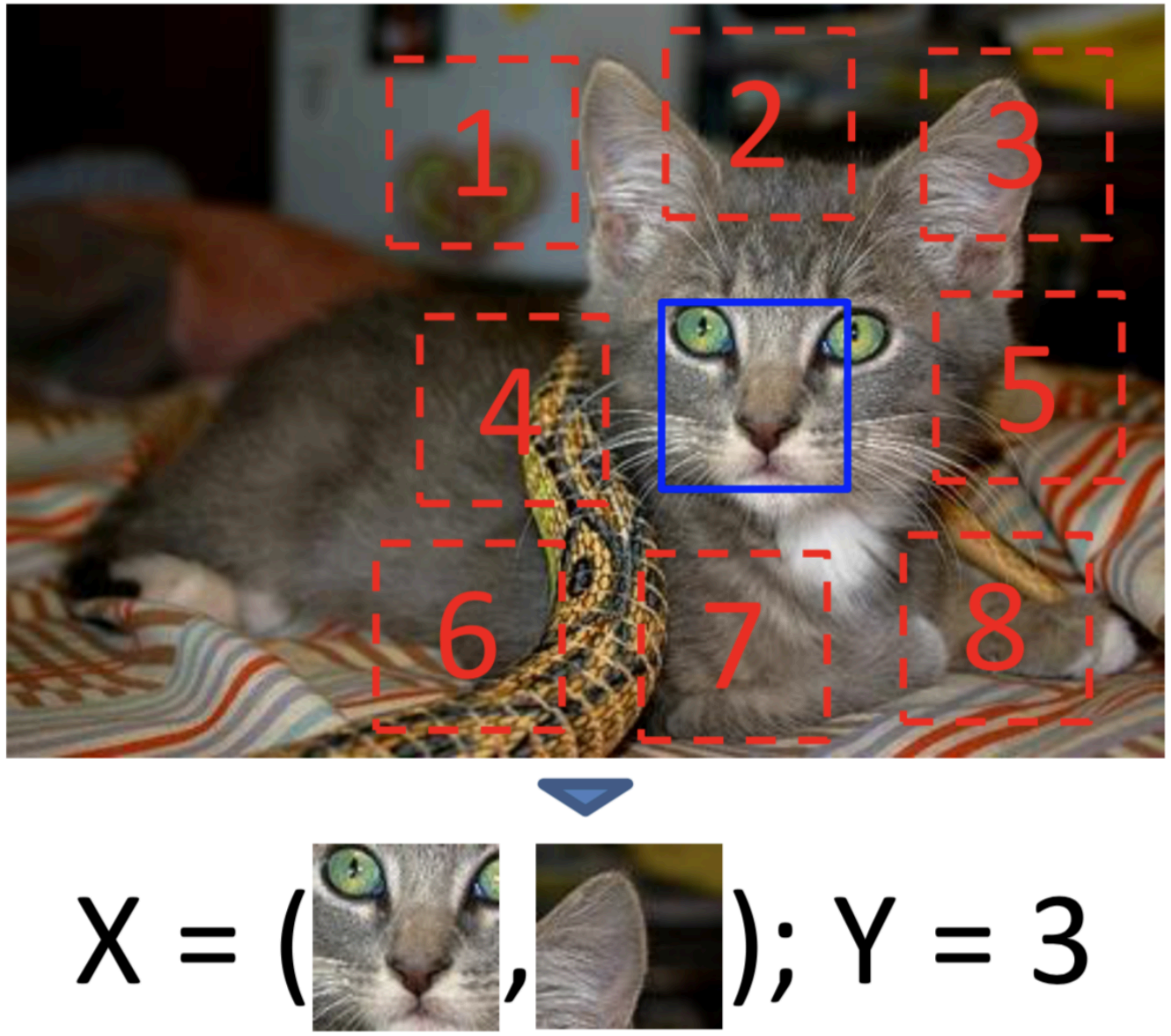

Gidaris et al. 2018

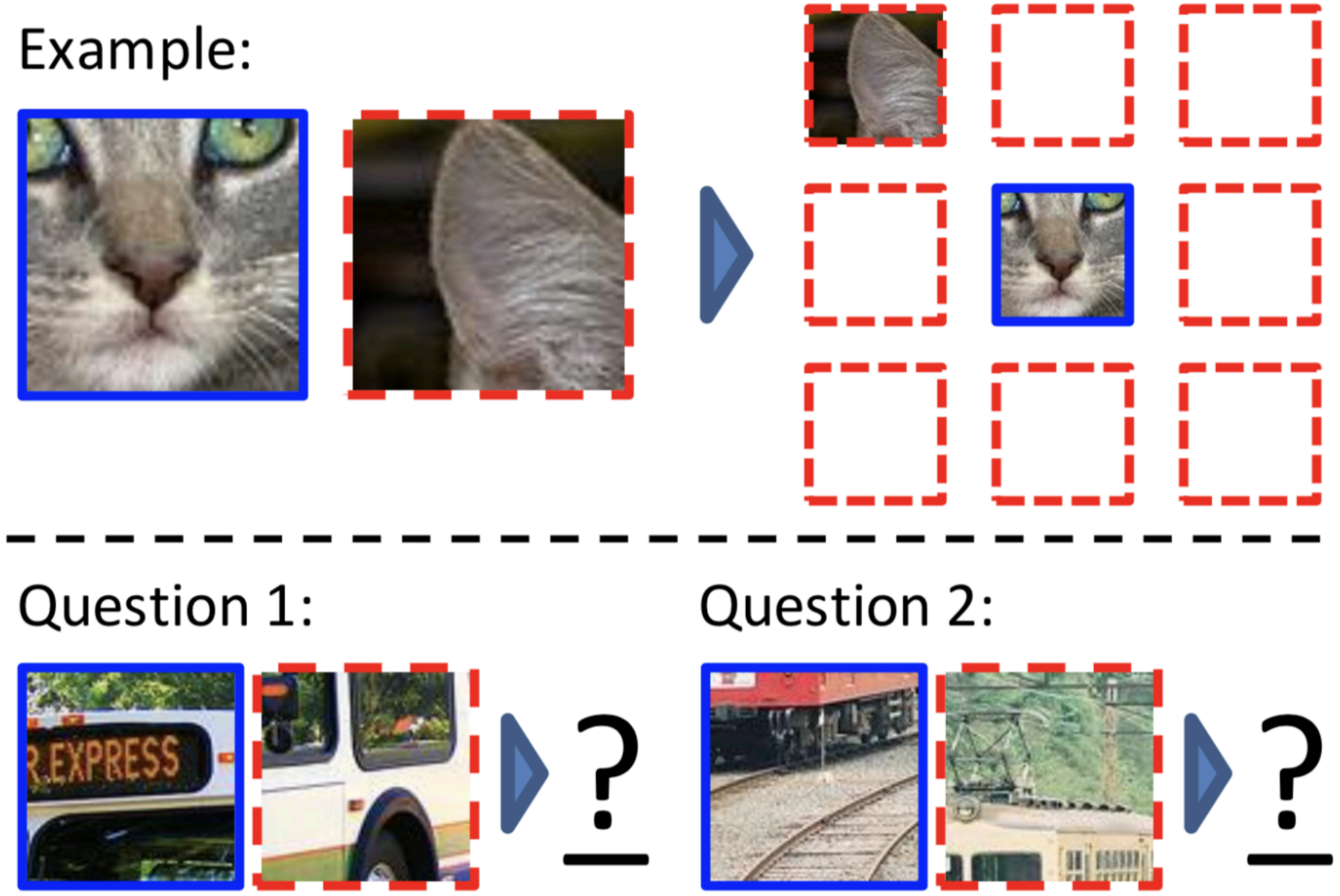
# Rotation




Gidaris et al. 2018

# Patches




[Doersch et al., 2015]

# Summary

- Basic steps to build an ML system
- Open-world machine learning
- Industry-scale machine learning

Thank you!