Non-Parametric Few-Shot Learning

CS 330

Course Reminders

Homework 1 due **tonight**.

Homework 2 released, due Mon 10/24.

Project mentors to be assigned this week.

Project proposal due next Weds 10/19.

(graded lightly, for your benefit)

Plan for Today

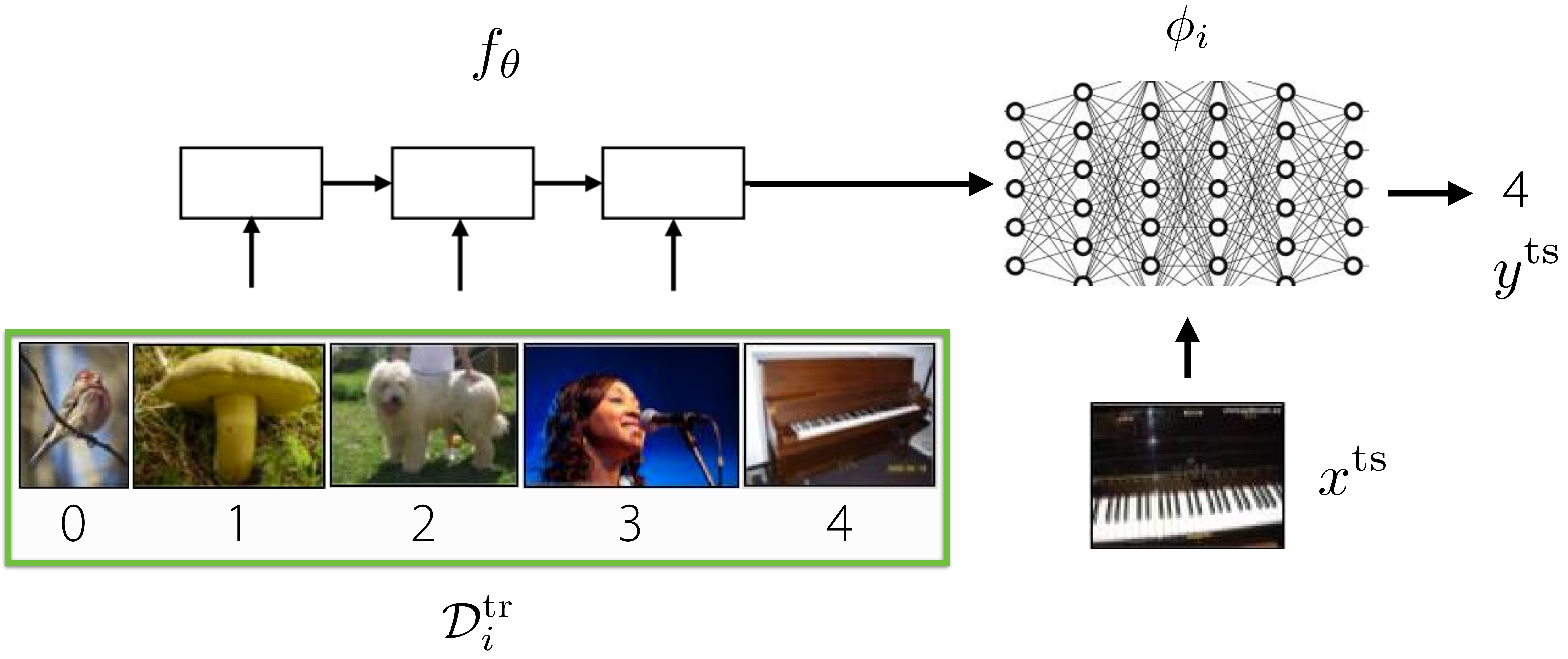
Non-Parametric Few-Shot Learning
- Siamese networks, matching networks, prototypical networks
- Case study of few-shot student feedback generation

Properties of Meta-Learning Algorithms
- Comparison of approaches

Example Meta-Learning Applications
- Imitation learning, drug discovery, motion prediction, language generation

Goals for by the end of lecture:
- Basics of non-parametric few-shot learning techniques (& how to implement)
- Trade-offs between black-box, optimization-based, and non-parametric meta-learning
- Familiarity with applied formulations of meta-learning

Recap: Black-Box Meta-Learning

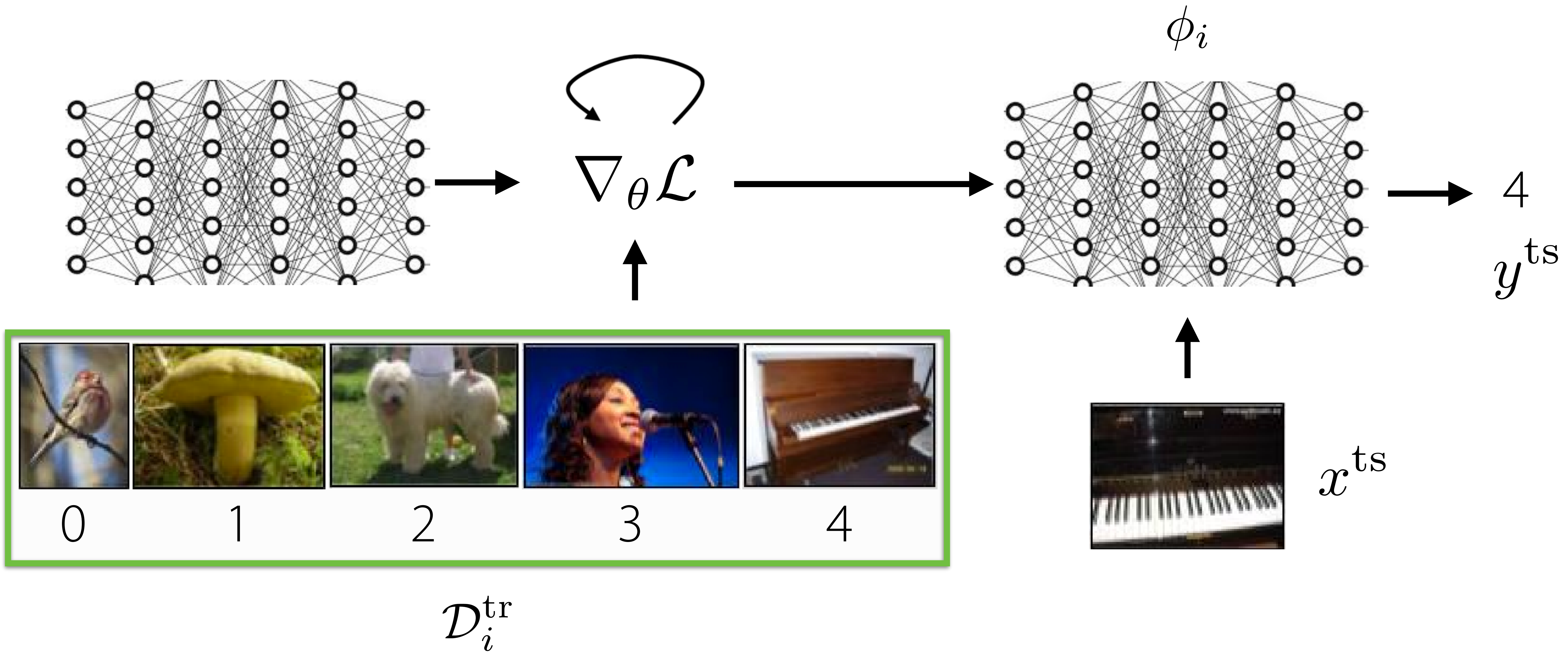


Key idea: parametrize learner as a neural network

+ **expressive**

- **challenging optimization** problem

Recap: Optimization-Based Meta-Learning



Key idea: embed optimization inside the inner learning process

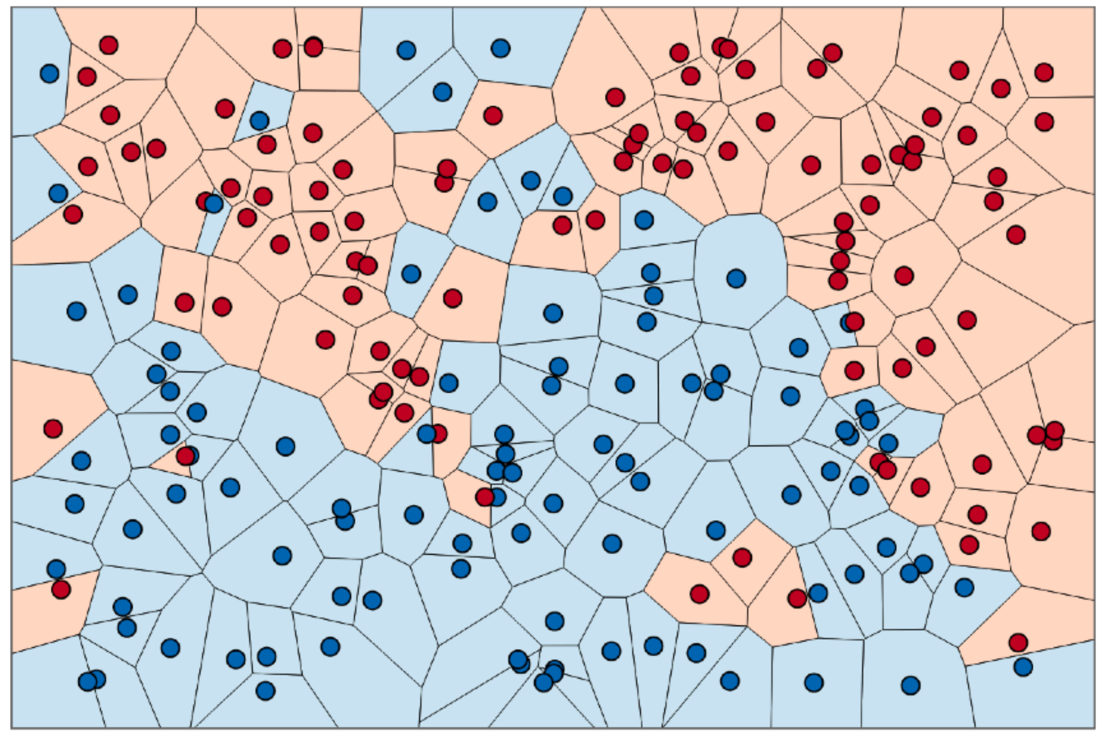
\+ **structure** of **optimization** embedded into meta-learner

\- typically requires **second-order optimization**

Today: Can we embed a learning procedure *without* a second-order optimization?

So far: Learning parametric models.

In low data regimes, **non-parametric** methods are simple, work well.



During **meta-test time**: few-shot learning <-> low data regime

During **meta-training**: still want to be parametric

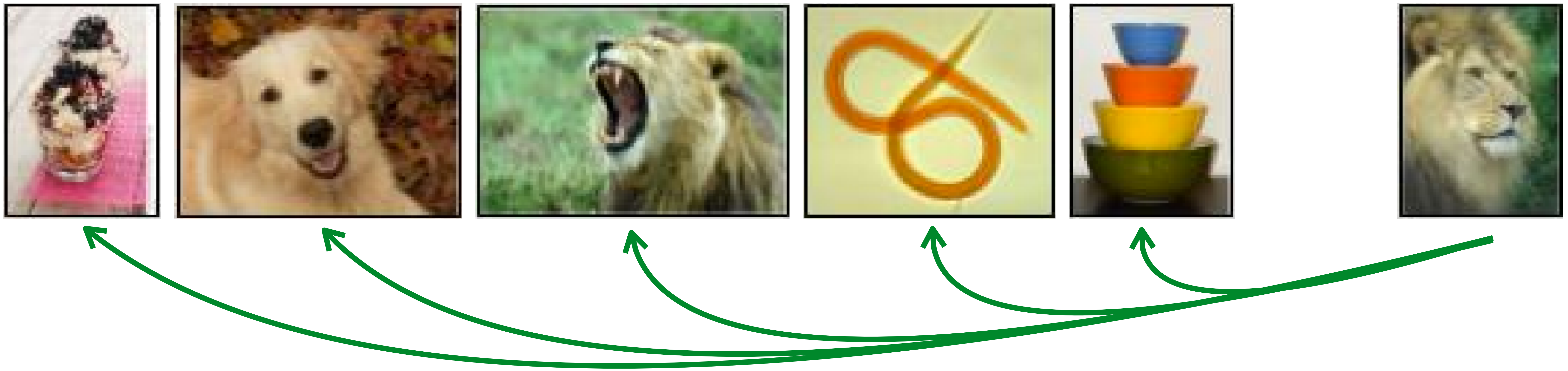
Can we use **parametric meta-learners** that produce effective **non-parametric learners**?

Note: some of these methods precede parametric approaches

Non-parametric methods

Key Idea: Use non-parametric learner.

training data $\mathcal{D}_i^{\mathrm{tr}}$ test datapoint x^{ts}



Compare test image with training images

In what space do you compare? With what distance metric?

ℓ_2 distance in pixel space?

In what space do you compare? With what distance metric?

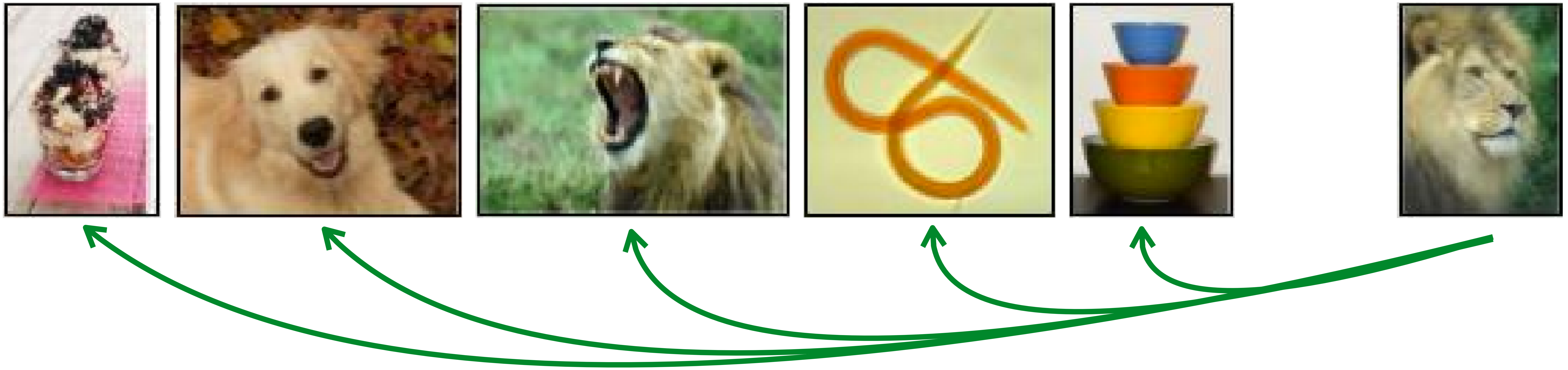
ℓ_2 distance in pixel space?

Zhang et al. (arXiv 1801.03924)

Non-parametric methods

Key Idea: Use non-parametric learner.

training data $\mathcal{D}_i^{\text{tr}}$ test datapoint x^{ts}



Compare test image with training images

In what space do you compare? With what distance metric?

~~ℓ_2 distance in pixel space?~~

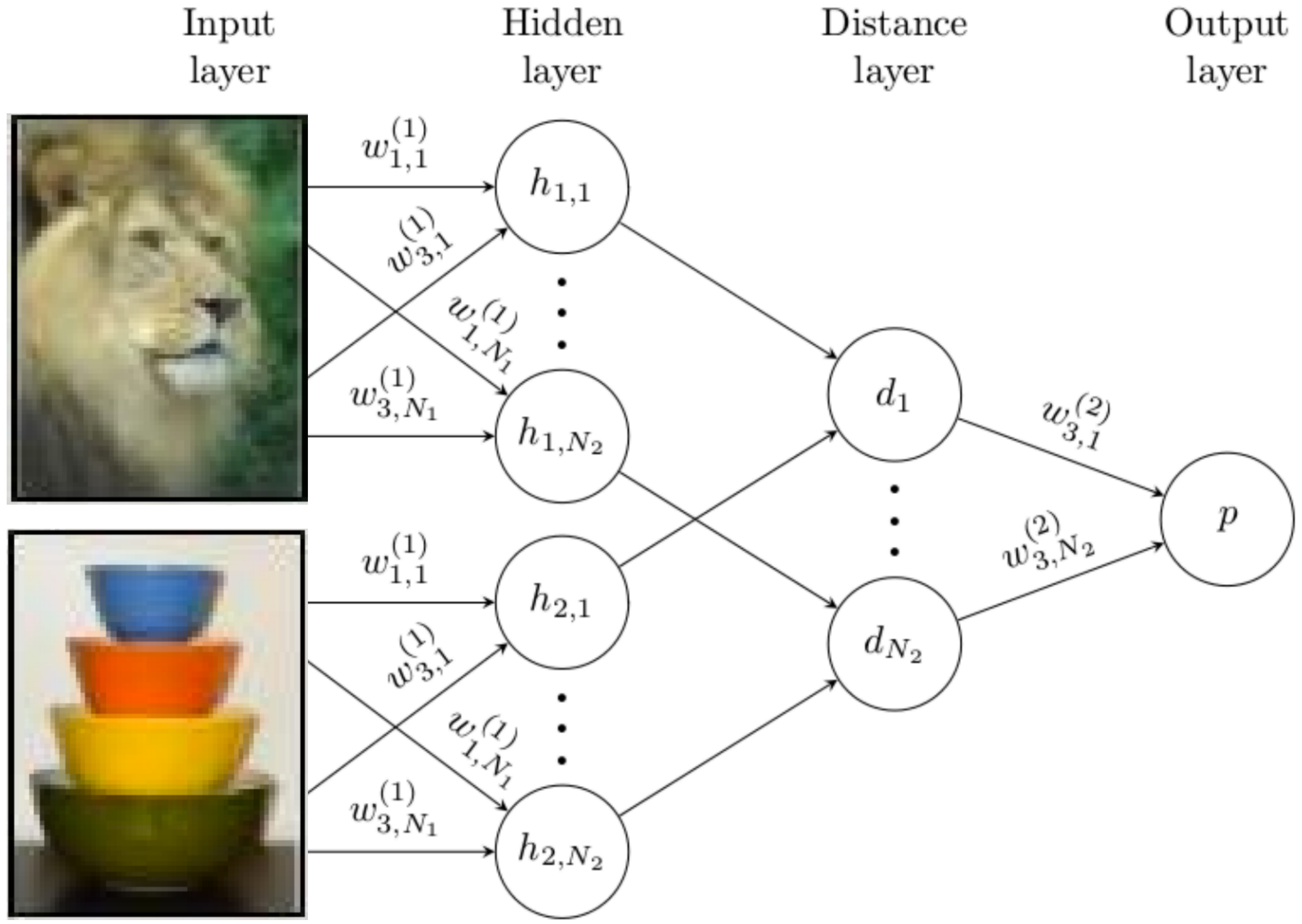
Question: What distance metric would you use instead?

Idea: Learn to compare using meta-training data

Non-parametric methods

Key Idea: Use non-parametric learner.

train Siamese network to predict whether or not two images are the same class



label

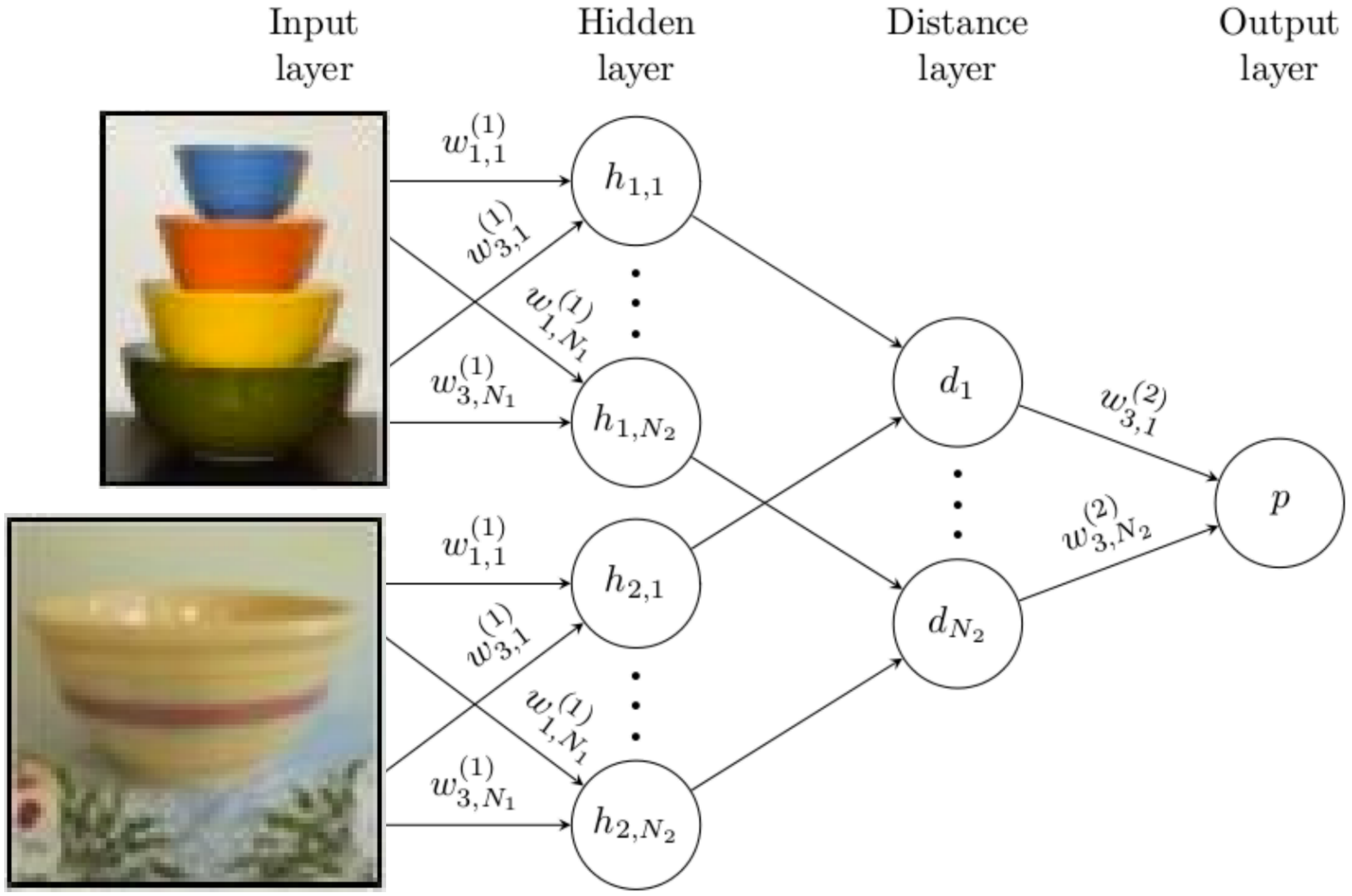
0

Koch et al., ICML '15

Non-parametric methods

Key Idea: Use non-parametric learner.

train Siamese network to predict whether or not two images are the same class



label

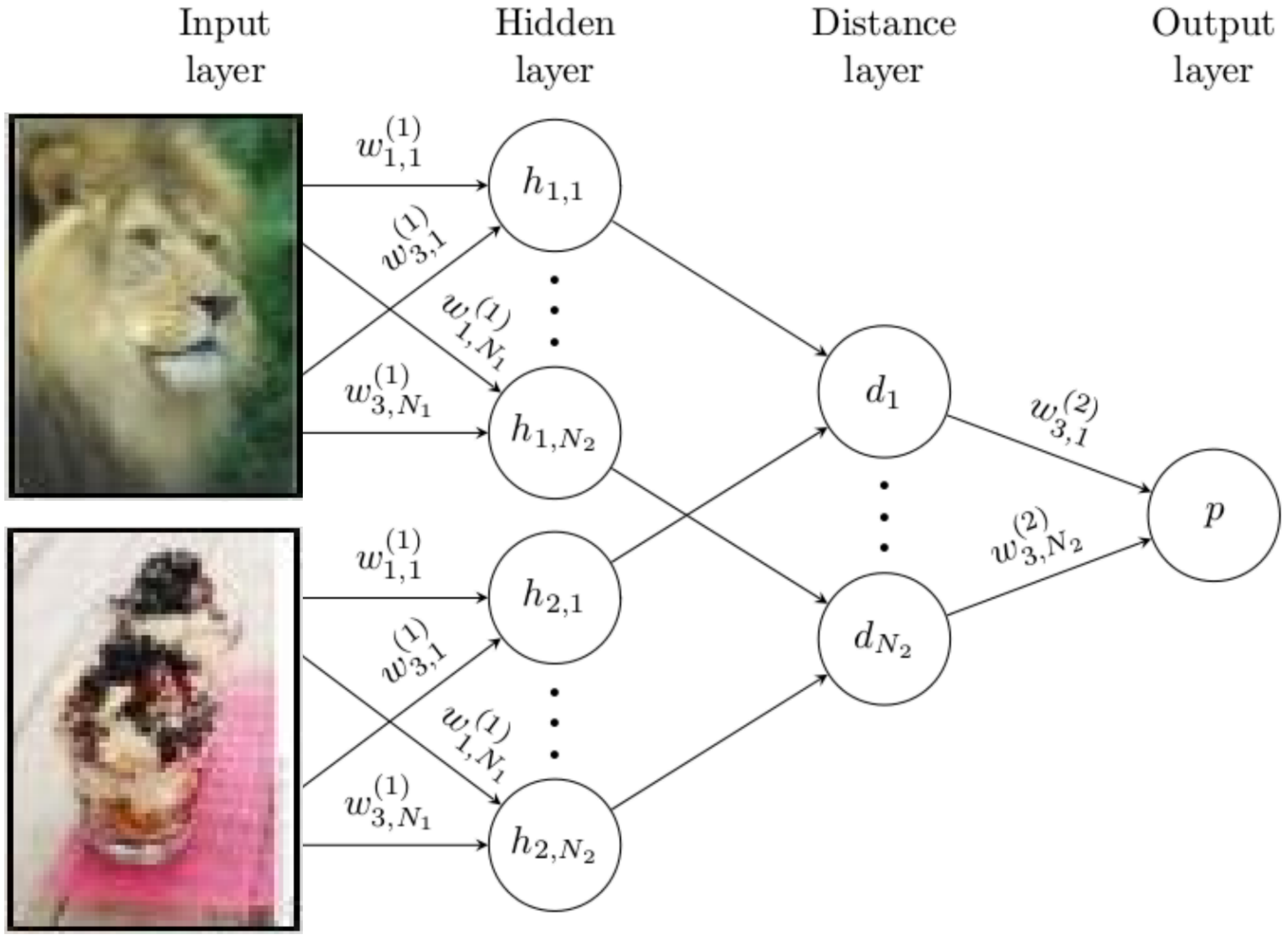
1

Koch et al., ICML ‘15

Non-parametric methods

Key Idea: Use non-parametric learner.

train Siamese network to predict whether or not two images are the same class



label

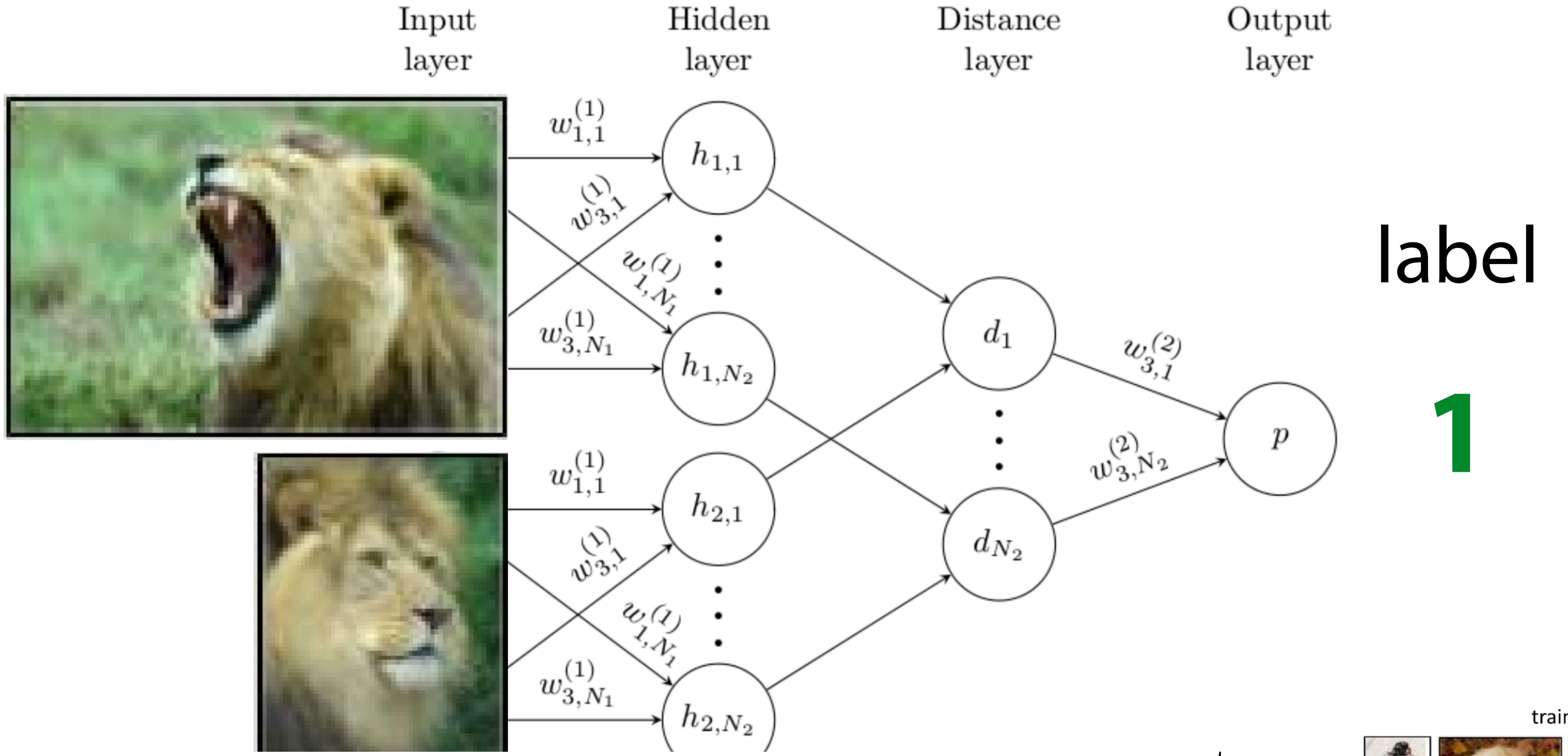
0

Koch et al., ICML '15

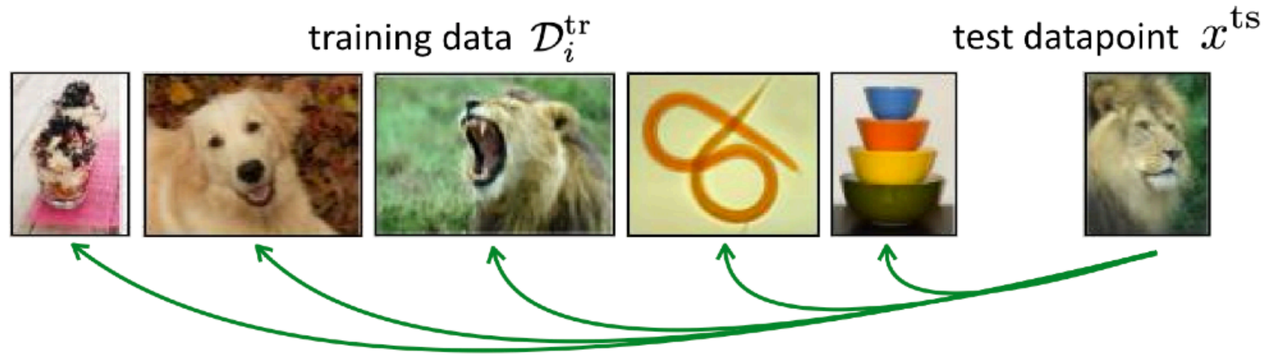
Non-parametric methods

Key Idea: Use non-parametric learner.

train Siamese network to predict whether or not two images are the same class



Meta-test time: compare image $\mathbf{x}_{\text{test}}$ to each image in $\mathcal{D}_j^{\text{tr}}$



Meta-training: Binary classification
Meta-test: N-way classification

Can we **match** meta-train & meta-test?

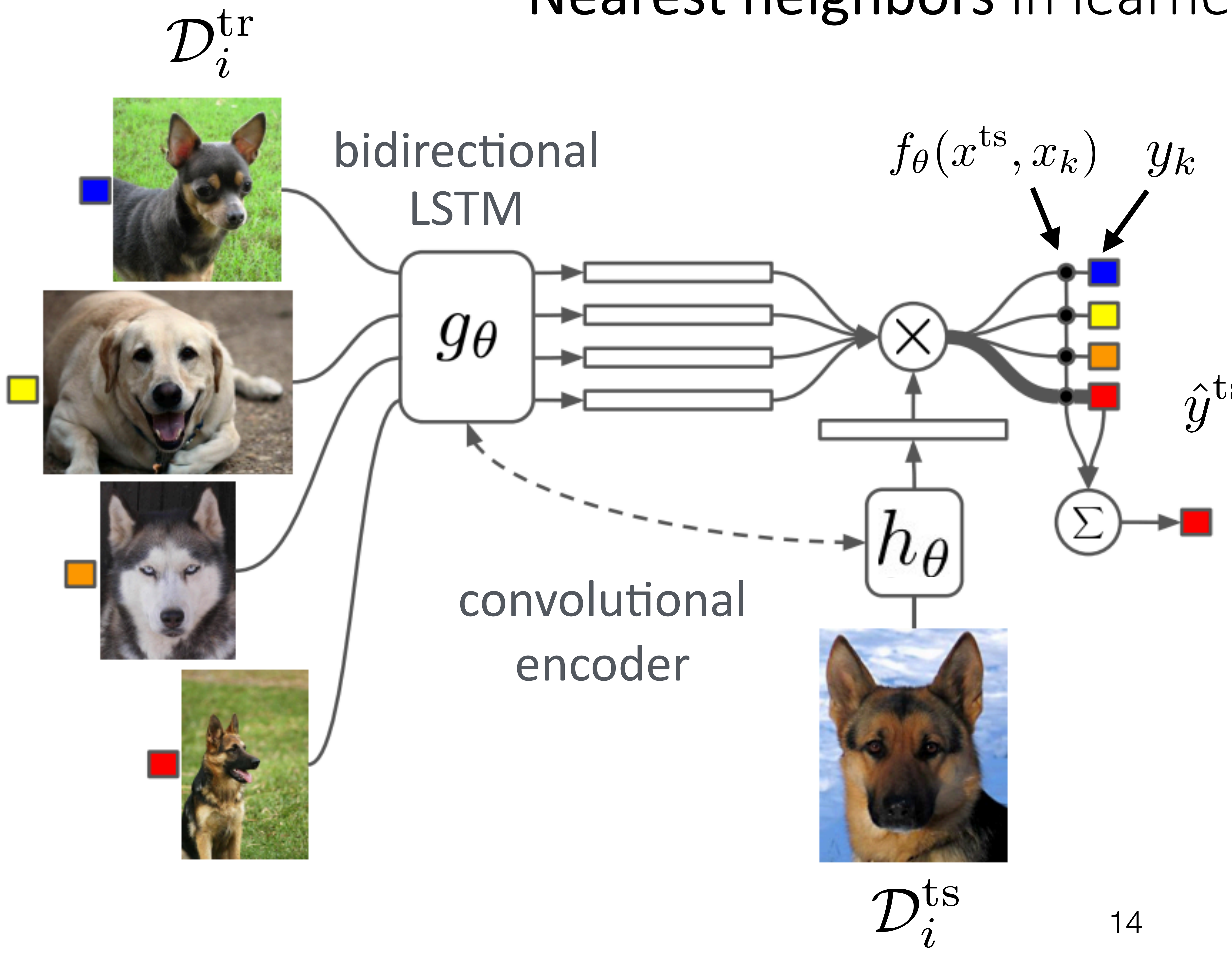
Koch et al., ICML '15

Non-parametric methods

Key Idea: Use non-parametric learner.

Can we **match** meta-train & meta-test?

Nearest neighbors in learned embedding space



$$\hat{y}^{\mathrm{ts}} = \sum_{x_k, y_k \in \mathcal{D}^{\mathrm{tr}}} f_\theta(x^{\mathrm{ts}}, x_k) y_k$$

Trained **end-to-end**.

Meta-train & **meta-test** time **match**.

Vinyals et al. Matching Networks, NeurIPS '16

Non-parametric methods

Key Idea: Use non-parametric learner.

General Algorithm:

~~Black box approach~~ Non-parametric approach (matching networks)

1. Sample task $\mathcal{T}_i$ *(or mini batch of tasks)*
2. Sample disjoint datasets $\mathcal{D}_i^{\text{tr}}, \mathcal{D}_i^{\text{test}}$ from $\mathcal{D}_i$
3. $\cancel{\text{Compute } \phi_i \leftarrow f_\theta(\mathcal{D}_i^{\text{tr}})}$ Compute $\hat{y}^{\text{ts}} = \sum_{x_k, y_k \in \mathcal{D}^{\text{tr}}} f_\theta(x^{\text{ts}}, x_k) y_k$
4. $\cancel{\text{Update } \theta \text{ using } \nabla_\theta \mathcal{L}(\phi_i, \mathcal{D}_i^{\text{test}})}$ Update θ using $\nabla_\theta \mathcal{L}(\hat{y}^{\text{ts}}, y^{\text{ts}})$

(Parameters ϕ integrated out, hence **non-parametric**)

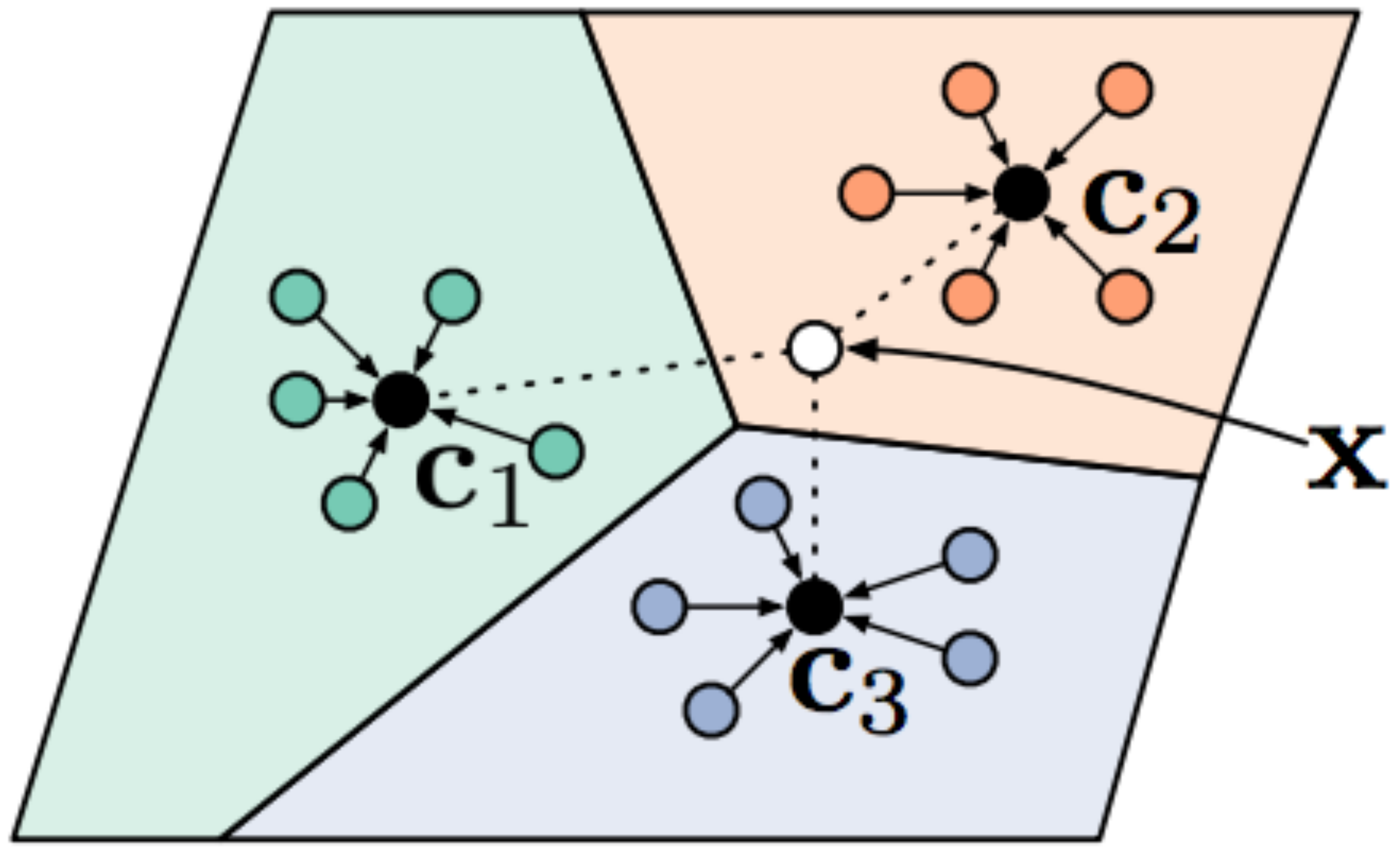
What if **>1 shot**?

Matching networks will perform comparisons **independently**

Can we **aggregate class information** to create a **prototypical embedding**?

Non-parametric methods

Key Idea: Use non-parametric learner.



$$\mathbf{c}_n = \frac{1}{K} \sum_{(x,y)\in\mathcal{D}_i^{\mathrm{tr}}} \mathbb{1}(y=n) f_\theta(x)$$

$$p_\theta(y=n|x) = \frac{\exp(-d\left(f_\theta(x), \mathbf{c}_n\right))}{\sum_{n'} \exp(d(f_\theta(x), \mathbf{c}_{n'}))}$$

d: Euclidean, or cosine distance

Snell et al. Prototypical Networks, NeurIPS '17

Non-parametric methods

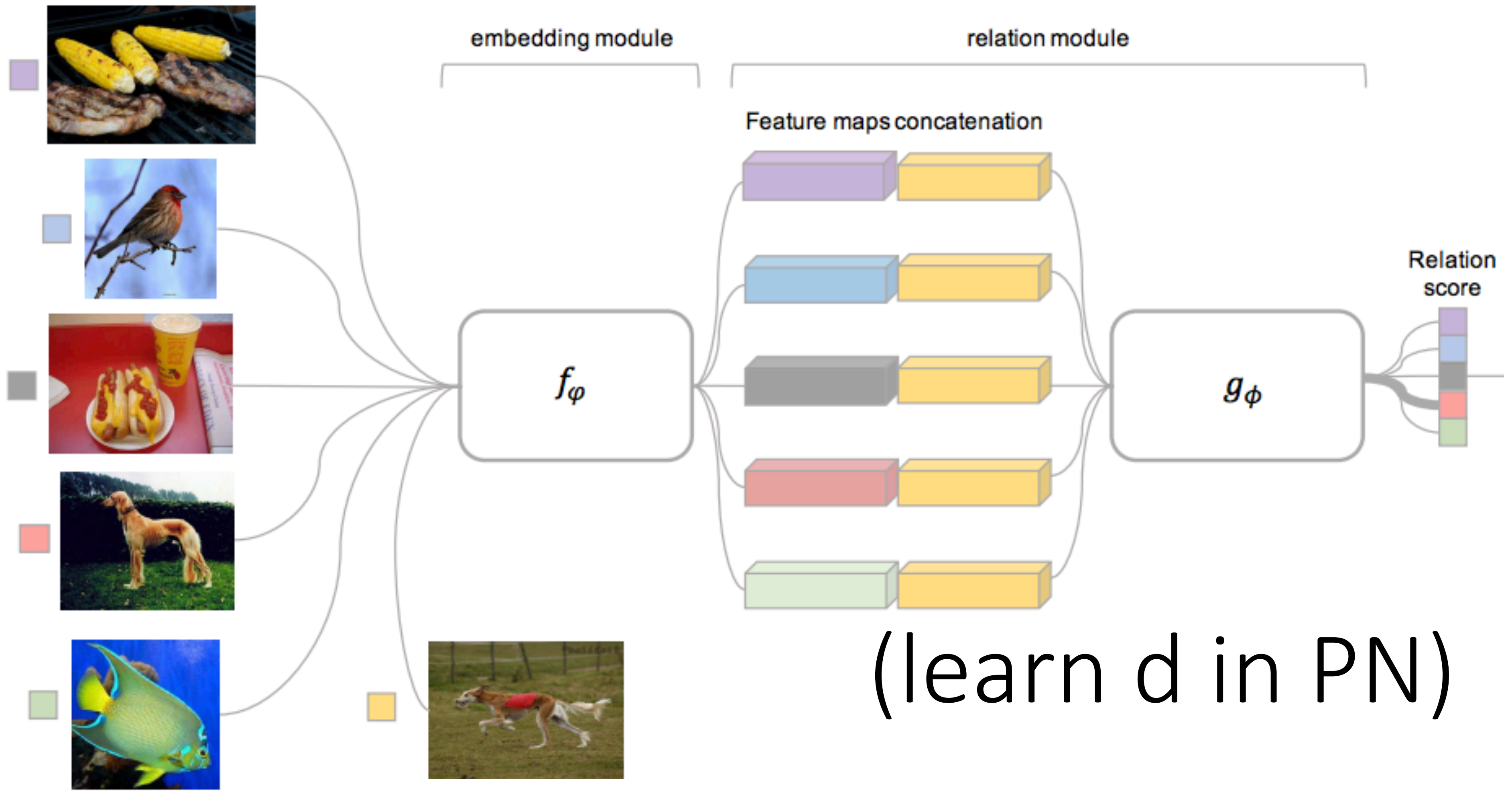
So far: Siamese networks, matching networks, prototypical networks

Embed, then nearest neighbors.

Challenge

What if you need to reason about more complex relationships between datapoints?

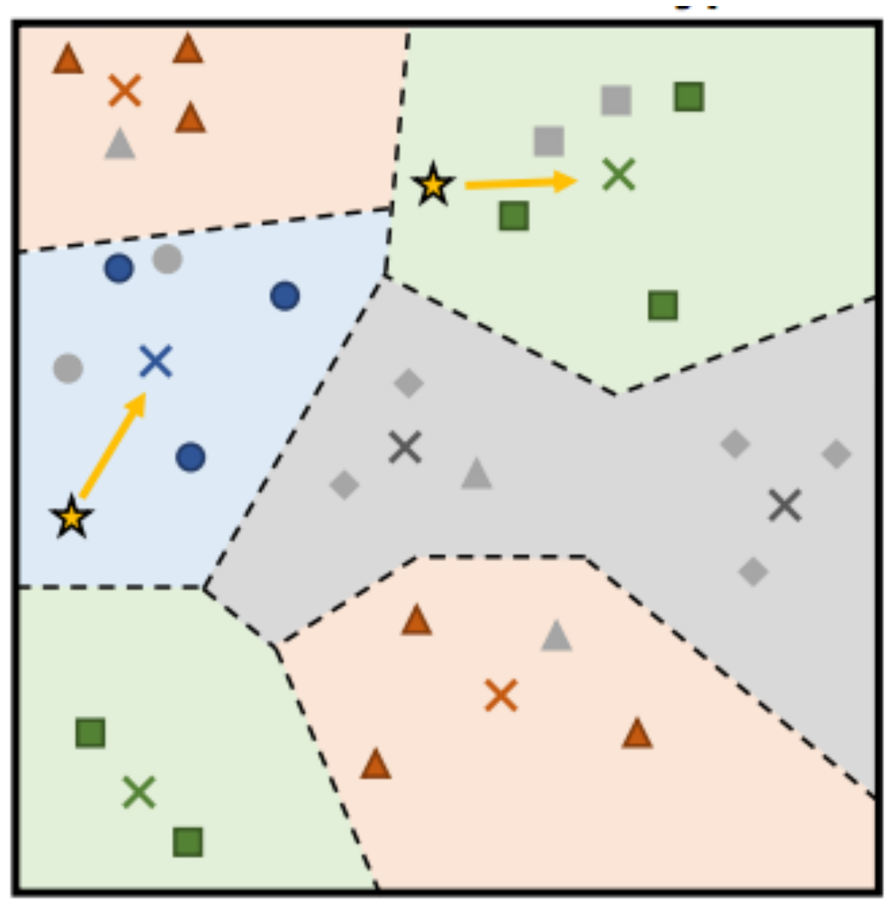
Idea: Learn non-linear relation module on embeddings



(learn d in PN)

Sung et al. Relation Net

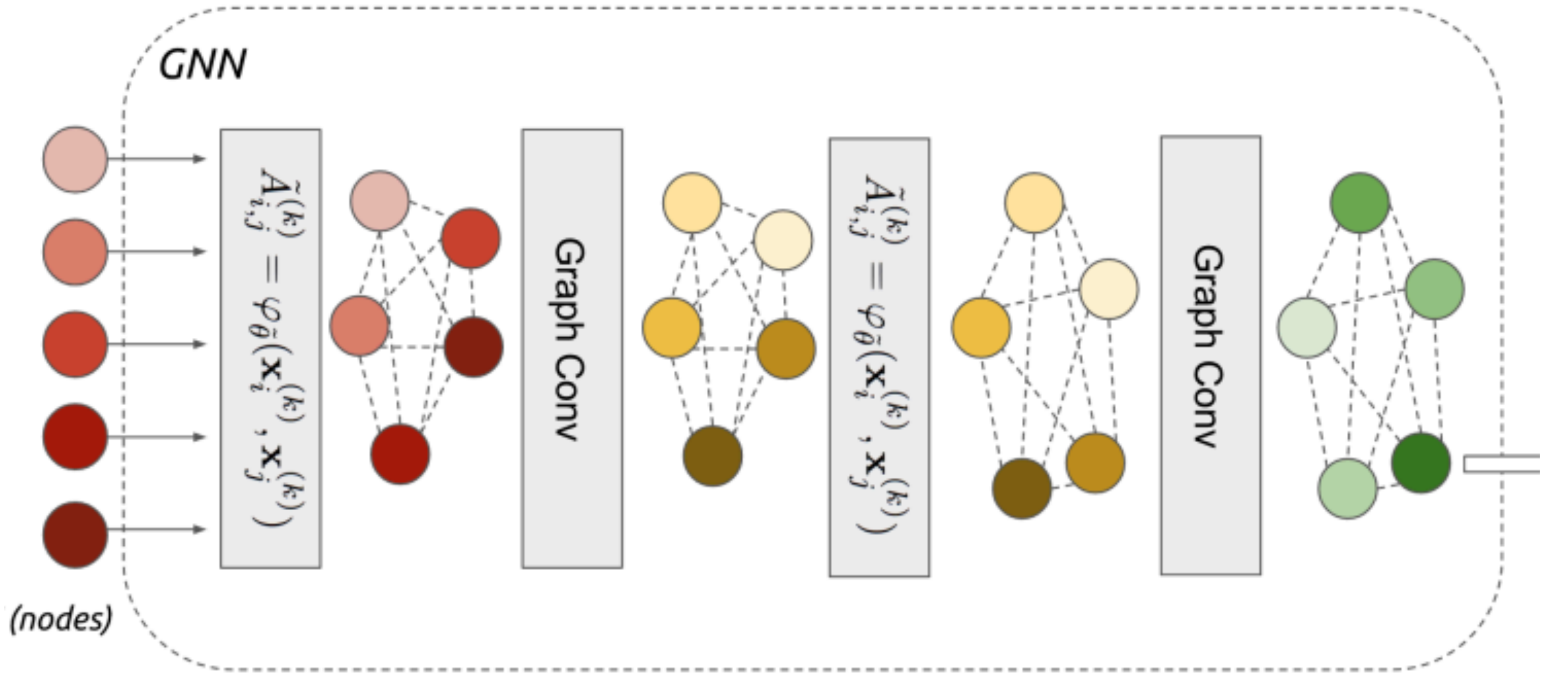
Idea: Learn infinite mixture of prototypes.



adaptive number of clusters

Allen et al. IMP, ICML '19

Idea: Perform message passing on embeddings



Garcia & Bruna, GNN

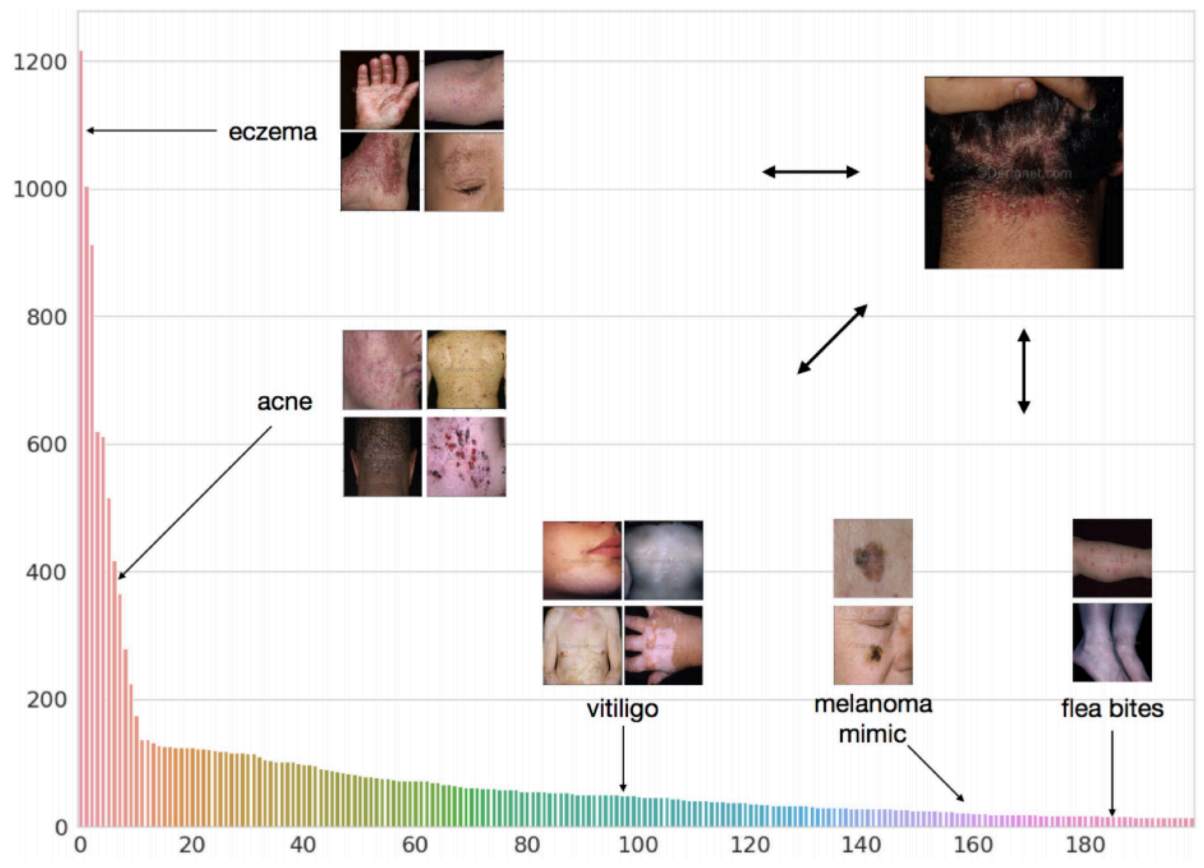
Previous Year's Case Study

Prototypical Clustering Networks for Dermatological Image Classification

Viraj Prabhu *,[1] Anitha Kannan[3] Murali Ravuri[3] Manish Chablani[3]
David Sontag[2] Xavier Amatriain[3]
[1]Georgia Tech [2]MIT [3]Curai
virajp@gatech.edu dsontag@mit.edu {anitha, murali, manish, xavier}@curai.com

Machine Learning for Healthcare Conference 2019

Link: https://arxiv.org/abs/1811.03066



Challenges:
- hard to get data
- data is long-tailed

Goal: Acquire accurate classifier on all classes

This Year's Case Study

Meta-Learning Student Feedback to 16,000 Solutions

Mike Wu, Chris Piech, and Chelsea Finn

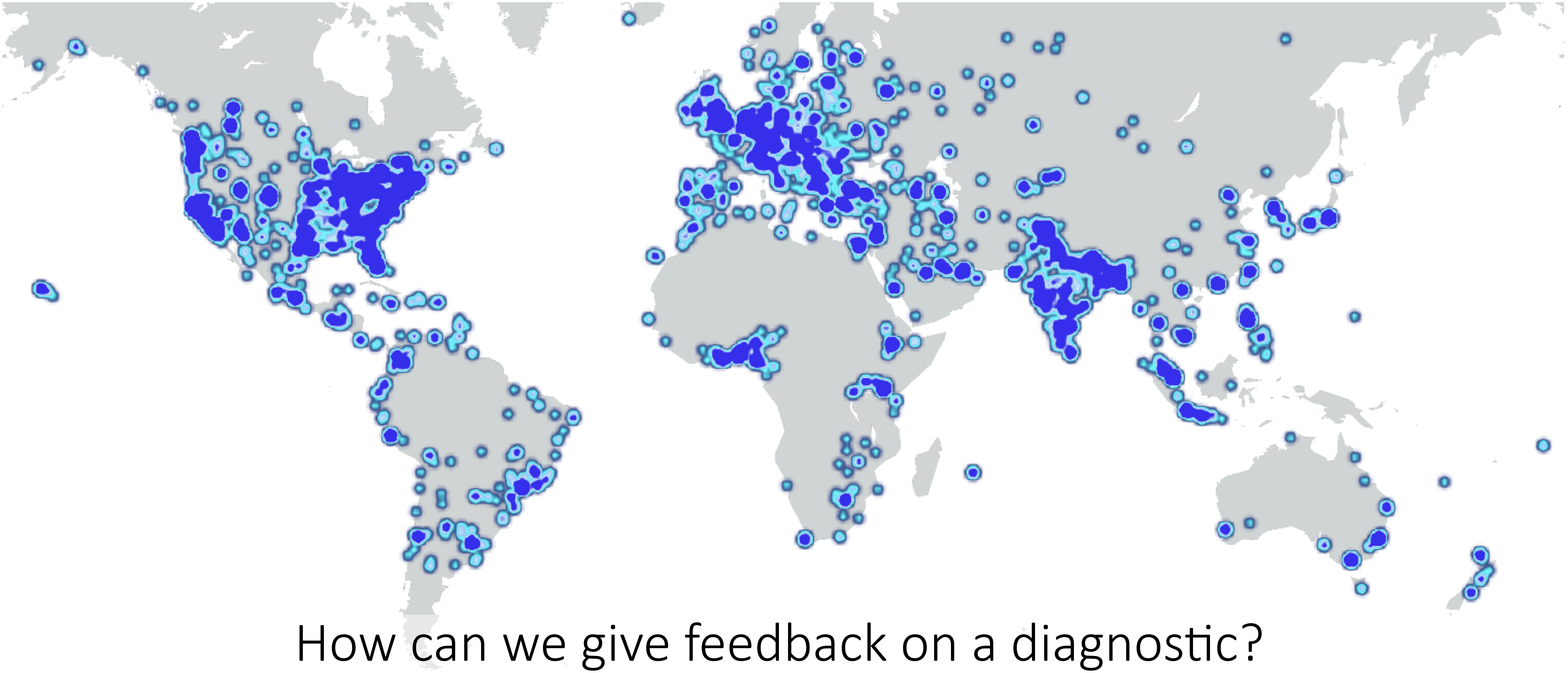
July 20, 2021

Links: https://ai.stanford.edu/blog/prototransformer/

https://arxiv.org/abs/2107.14035

The Feedback Problem

Code-in-Place 2021: Free intro to CS course, 12,000+ students from 150+ countries



How can we give feedback on a diagnostic?

Submissions: open-ended Python code snippets

Estimated 8+ months of human labor

Piech & Sahami & Zelenski,
Stanford University

The Feedback Challenge

- Train a model to infer student misconceptions, **y**, from the student solution, **x**.

```
# print 1 to n w/ loop
def my_solution(n)
      print(1)
      print(2)
      print(3)
```

[x] Incorrect Syntax

[x] Did not loop

[ ] Uses “print” fn

Predict!

Same rubrics that instructors use to give their feedback.

The Feedback Challenge

Why is this a hard problem for ML?

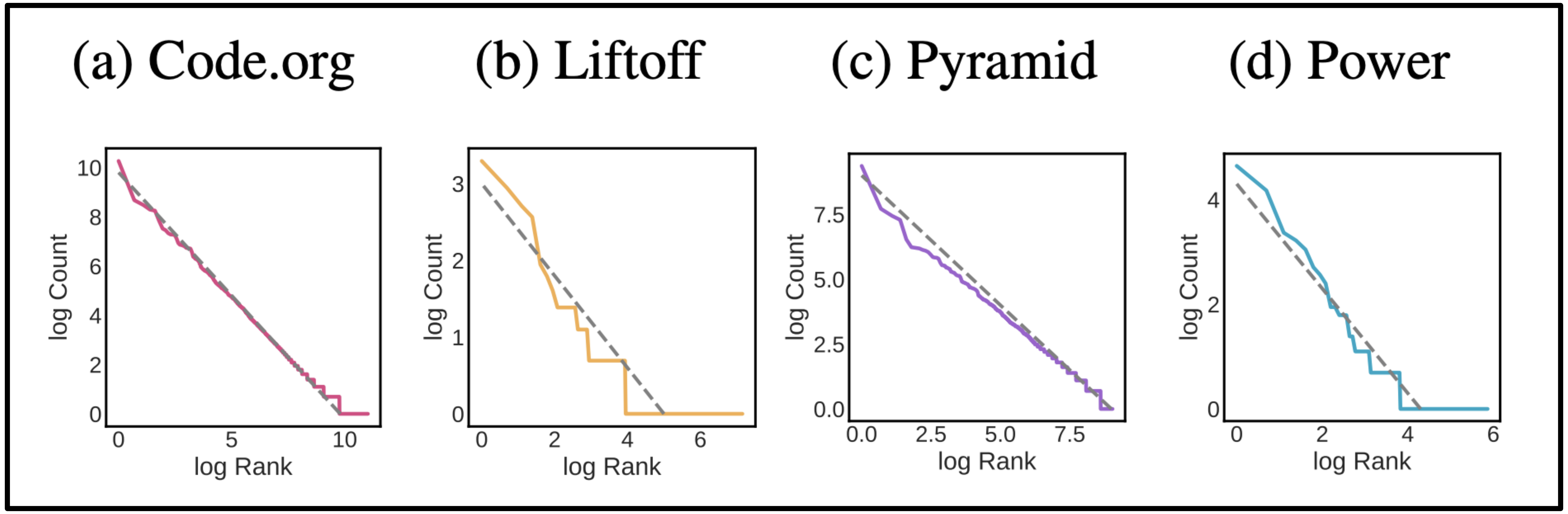
- **Limited annotation**: grading student work takes expertise and is very time consuming.

Example: annotating 800 blockly codes took **25 hrs**

The Feedback Challenge

Why is this a hard problem for ML?

- **Limited annotation**: grading student work takes expertise and is very time consuming.
- **Long tailed distribution**: students solve the same problem in many *many* ways.



Generative Grading: Neural Approximate Parsing for Verifiable Automated Student Feedback (Malik et. al. 2020)

The Feedback Challenge

Why is this a hard problem for ML?

- **Limited annotation**: grading student work takes expertise and is very time consuming.
- **Long tailed distribution**: students solve the same problem in many *many* ways.
- **Changing curriculums:** instructors constantly <u>edit</u> assignments and exams. Student solutions and instructor feedback look <u>different</u> year to year.

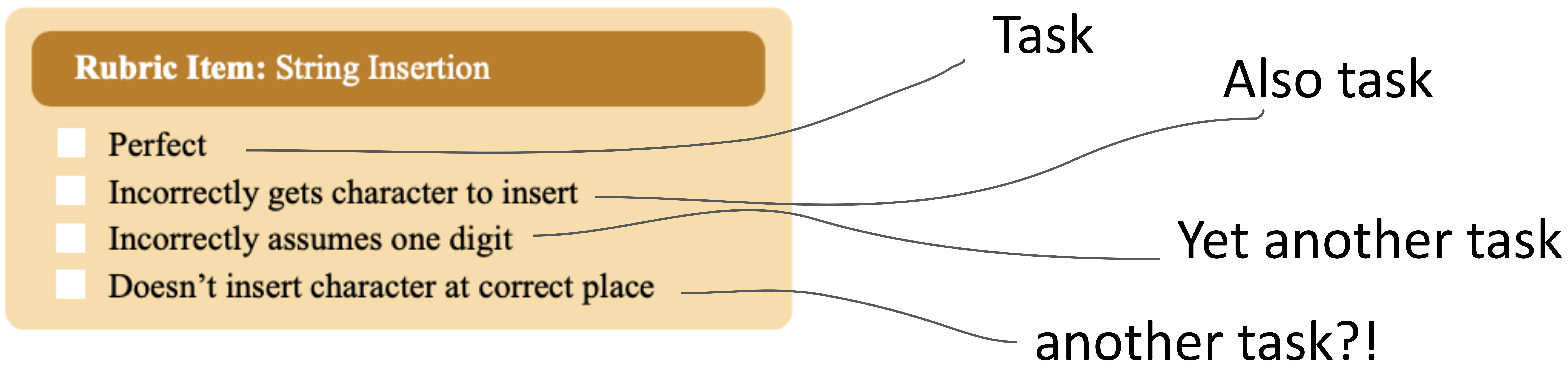
Framing it as a Meta-Learning Problem

Meta-Training Dataset: 4 final exams and 4 midterm exams from CS106.

- 63 questions and 24.8k student solutions.
- Every student solution has feedback via a <u>rubric</u>.

A <u>rubric</u> has several <u>items</u>. Each item has several <u>options</u> that you may pick as true.

- More than one option can be true.
- Every problem has its own (possibly unique) rubric items and options.



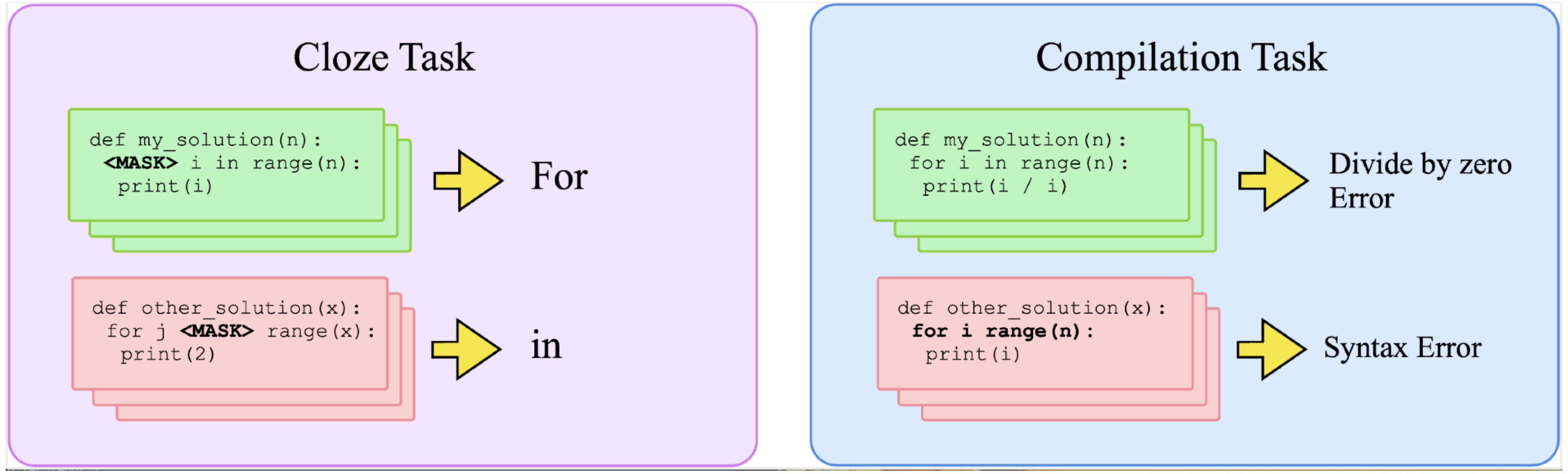
ProtoTransformer

$$p_\theta(y = n|x) = \frac{\exp(-d\left(f_\theta(x), \mathbf{c}_n\right))}{\sum_{n'} \exp(d(f_\theta(x), \mathbf{c}_{n'}))}$$

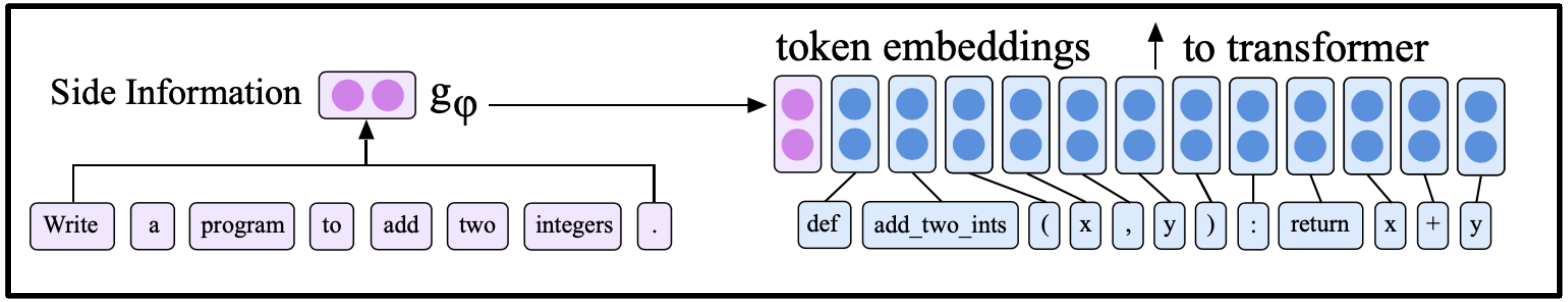
- $x = (x_1, x_2, \ldots, x_T)$ is a **sequence of discrete tokens** (e.g. code, language).
- The embedding $f_\Theta: X \rightarrow R^d$ is **a RoBERTa model** (stacked transformers) where token embeddings are averaged into a single vector.
- Applying this out of the box fails.

Attention is **not** all you need.😮

Trick #1: Augment rubric tasks with self-supervised tasks

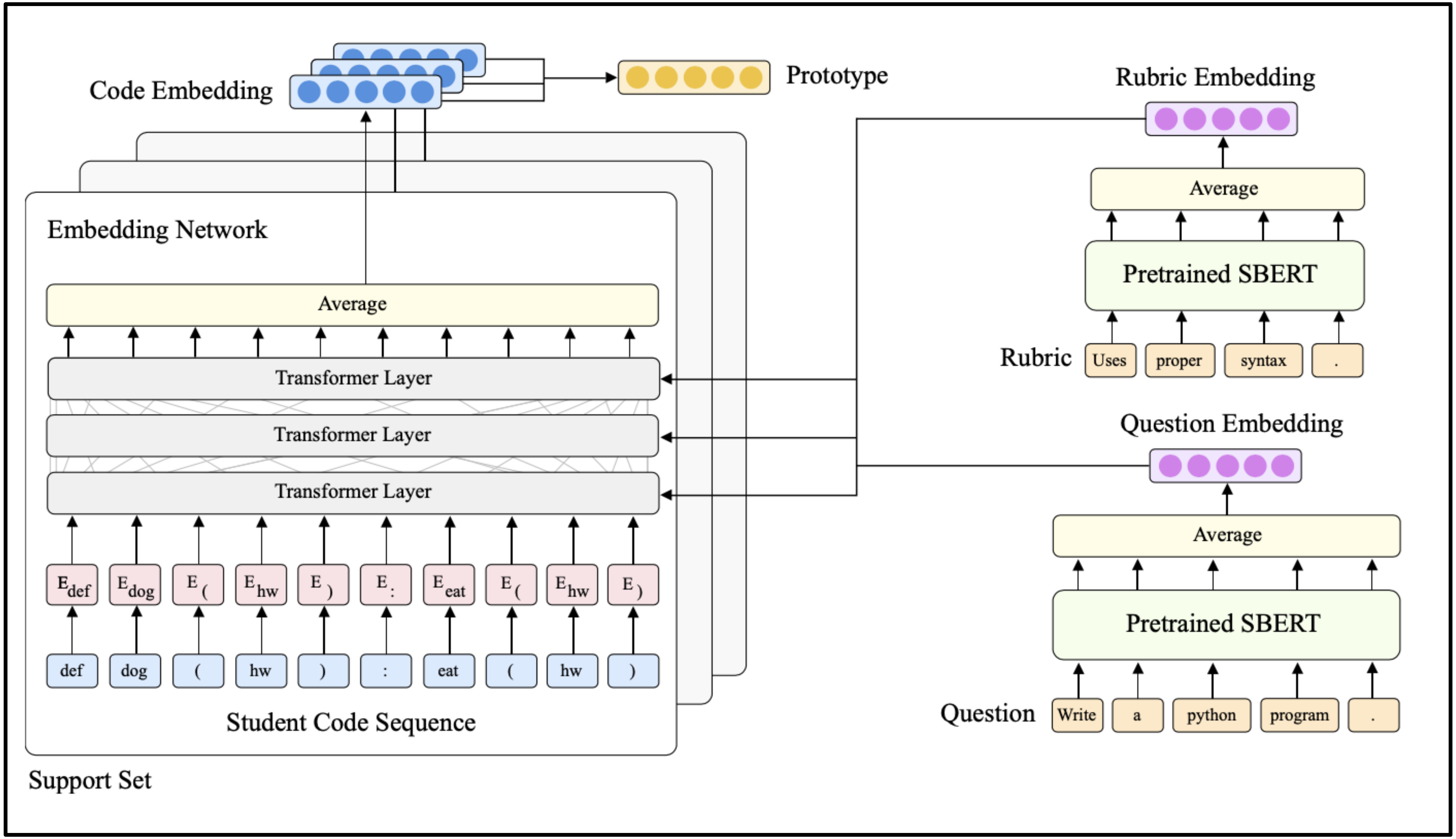


Trick #2: Reduce few-shot ambiguity by incorporating side information (rubric option name, question text)



Trick #3: Pre-train on unlabeled Python code.

CodeBERT: A Pre-Trained Model for Programming and Natural Languages (Feng et. al. 2020
CodeSearchNet Challenge: Evaluating the State of Semantic Code Search (Husain et.al. 2020)



Code Embedding
Prototype
Rubric Embedding
Embedding Network
Average
Transformer Layer
Transformer Layer
Transformer Layer
E def
E dog
E (
E hw
E)
E :
E eat
E (
E hw
E)
def
dog
(
hw
)
:
eat
(
hw
)
Student Code Sequence
Support Set
Average
Pretrained SBERT
Rubric
Uses
proper
syntax
.
Question Embedding
Average
Pretrained SBERT
Question
Write
a
python
program
.

Main Offline Results

	Held-out rubric			
Model	AP	P@50	P@75	ROC-AUC
ProtoTransformer	**84.2** (±1.7)	**85.2** (±3.8)	**74.2** (±1.4)	**82.9** (±1.3)
Supervised	66.9 (±2.2)	59.1 (±1.7)	53.9 (±1.5)	61.0 (±2.1)
Human TA	82.5	–	–	–
	Held-out exam			
Model	AP	P@50	P@75	ROC-AUC
ProtoTransformer	**74.2** (±1.6)	**77.3** (±2.7)	**67.3** (±2.0)	**77.0** (±1.4)
Supervised	65.8 (± 2.1)	60.1 (±3.0)	54.3 (±1.8)	60.7 (±1.6)
Human TA	82.5	–	–	–

- Outperforms supervised learning by 8-17%
- More accurate than human TA on held-out rubric
- Room to grow on held-out exam

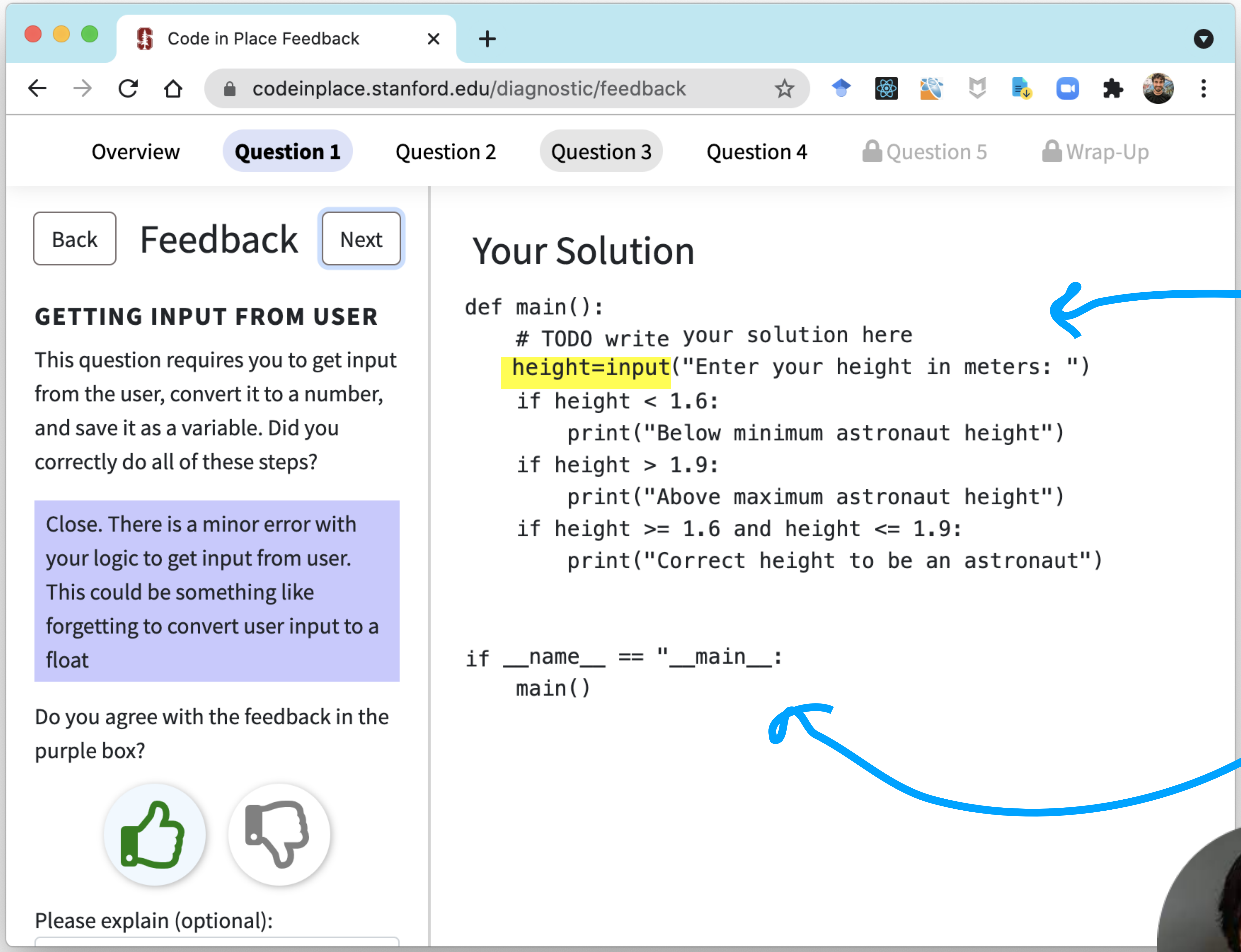
Live Deployment to Code-in-Place Students

May 10th, 2021: Students took diagnostic.

Algorithm uses attention to highlight where the error arises

AI generated feedback

Students evaluate the feedback



Syntax error here would prevent unit tests from being useful

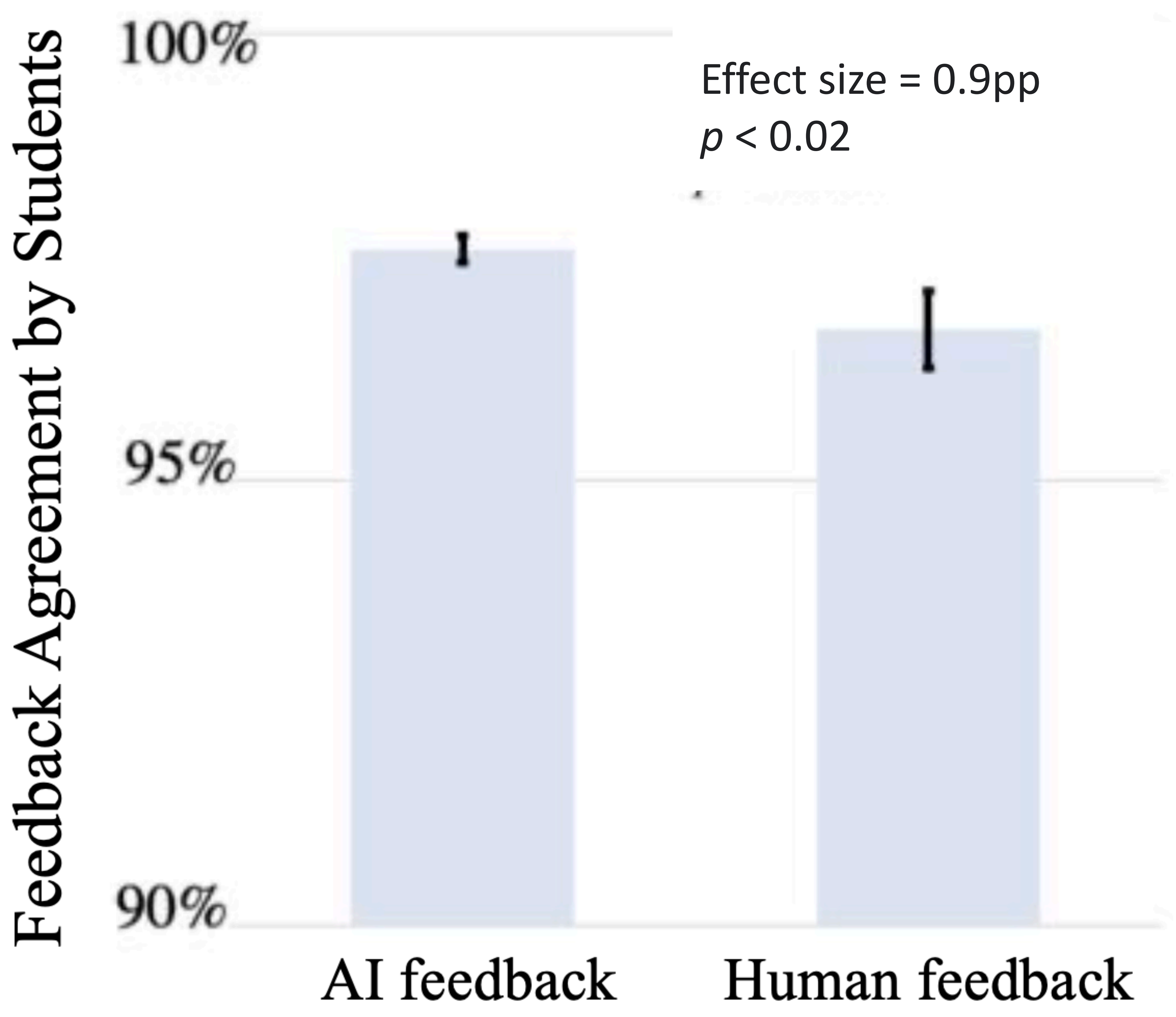
UI designed by Alan Cheng & Chris Piech

Blind, randomized trial *with real students*

Humans gave feedback ~1k answers.
AI gave feedback on the remaining ~**15k**.

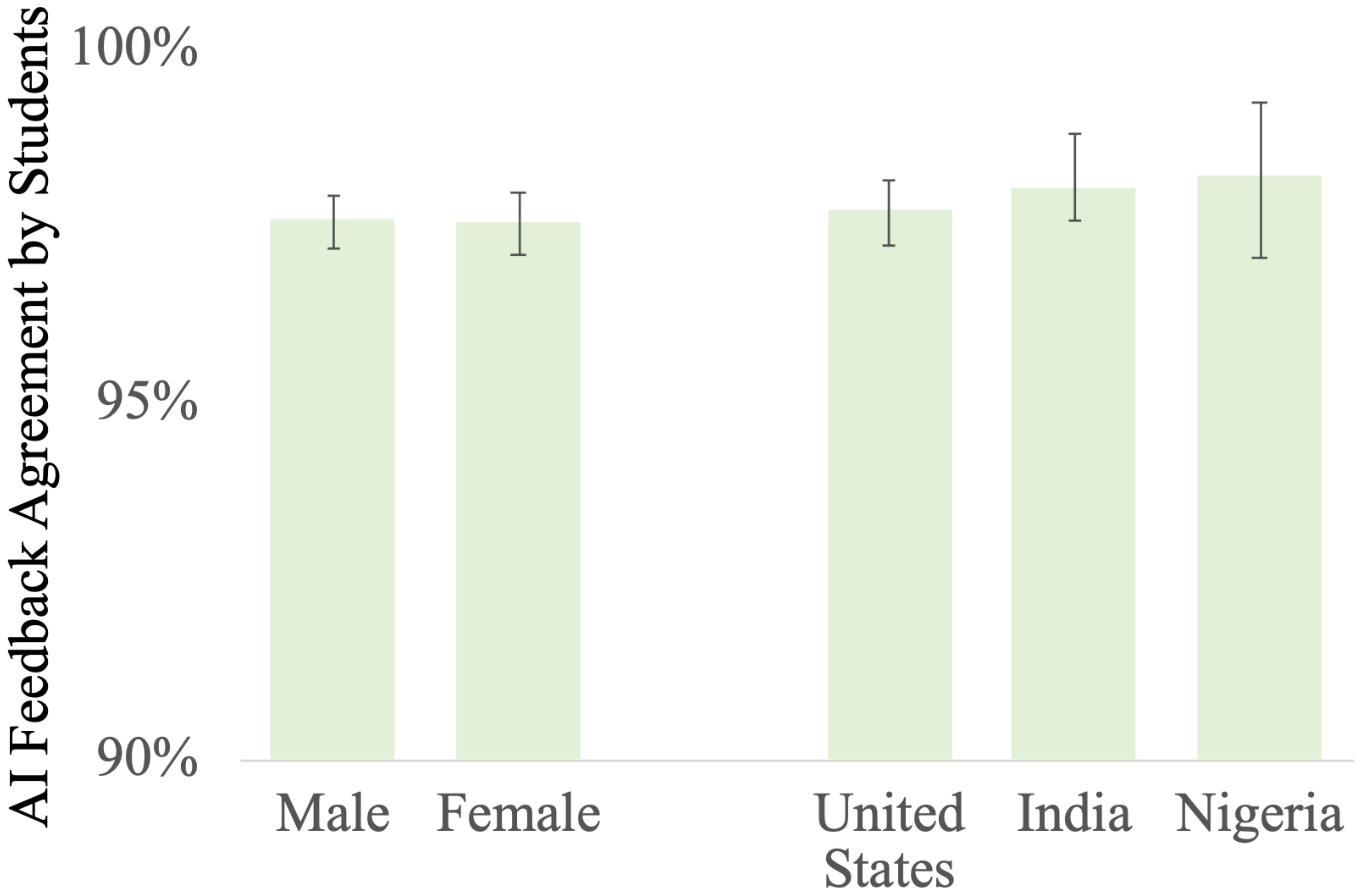
~2k could be auto-graded and were not included in analysis.

Humans gave good feedback.
ML model gave slightly better feedback.



Average holistic rating of usefulness by students was **4.6 ± 0.018 out of 5**.

No signs of bias by demographics



Plan for Today

Non-Parametric Few-Shot Learning
- Siamese networks, matching networks, prototypical networks
- Case study of few-shot student feedback generation

Properties of Meta-Learning Algorithms
- Comparison of approaches

Example Meta-Learning Applications
- Imitation learning, drug discovery, motion prediction, language generation

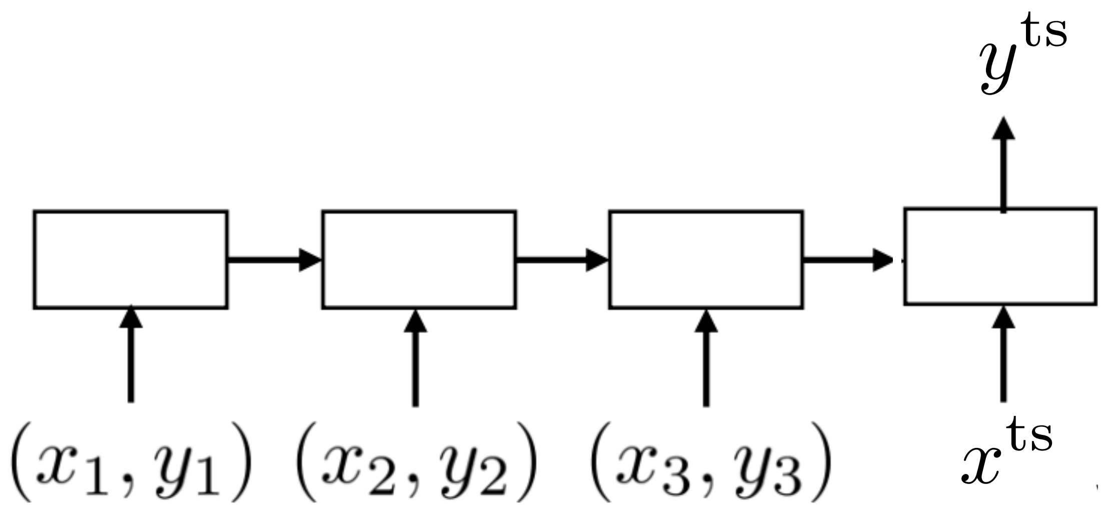
How can we think about how these methods compare?

Black-box vs. Optimization vs. Non-Parametric

Computation graph **perspective**

Black-box

$$y^{\text{ts}} = f_\theta(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}})$$



Optimization-based

$$y^{\text{ts}} = f_{\text{MAML}}(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}})$$
$$= f_{\phi_i}(x^{\text{ts}})$$
$$\text{where } \phi_i = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{\text{tr}})$$

Non-parametric

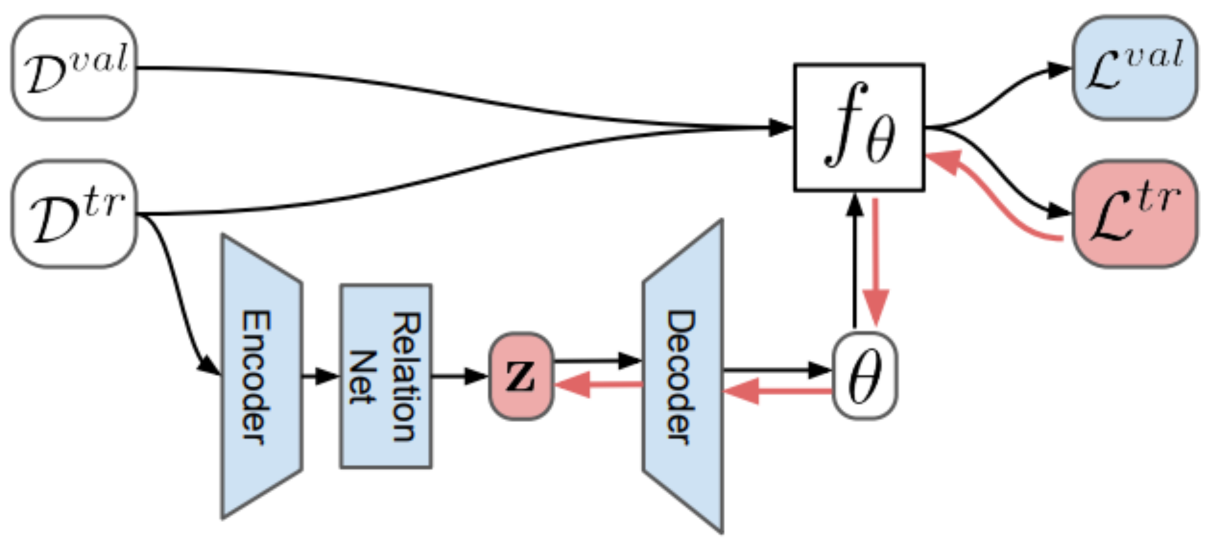
$$y^{\text{ts}} = f_{\text{PN}}(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}})$$
$$= \text{softmax}(-d\left(f_\theta(x^{\text{ts}}), \mathbf{c}_n\right))$$
$$\text{where } \mathbf{c}_n = \frac{1}{K} \sum_{(x,y) \in \mathcal{D}_i^{\text{tr}}} \mathbb{1}(y = n) f_\theta(x)$$

Note: (again) Can mix & match components of computation graph

Both condition on data & run gradient descent.

Jiang et al. CAML '19

Gradient descent on relation net embedding.



Rusu et al. LEO '19

MAML, but initialize last layer as ProtoNet during meta-training

Triantafillou et al. Proto-MAML '19

Black-box vs. Optimization vs. Non-Parametric

***Algorithmic properties* perspective**

Expressive power

the ability for f to represent a range of learning procedures

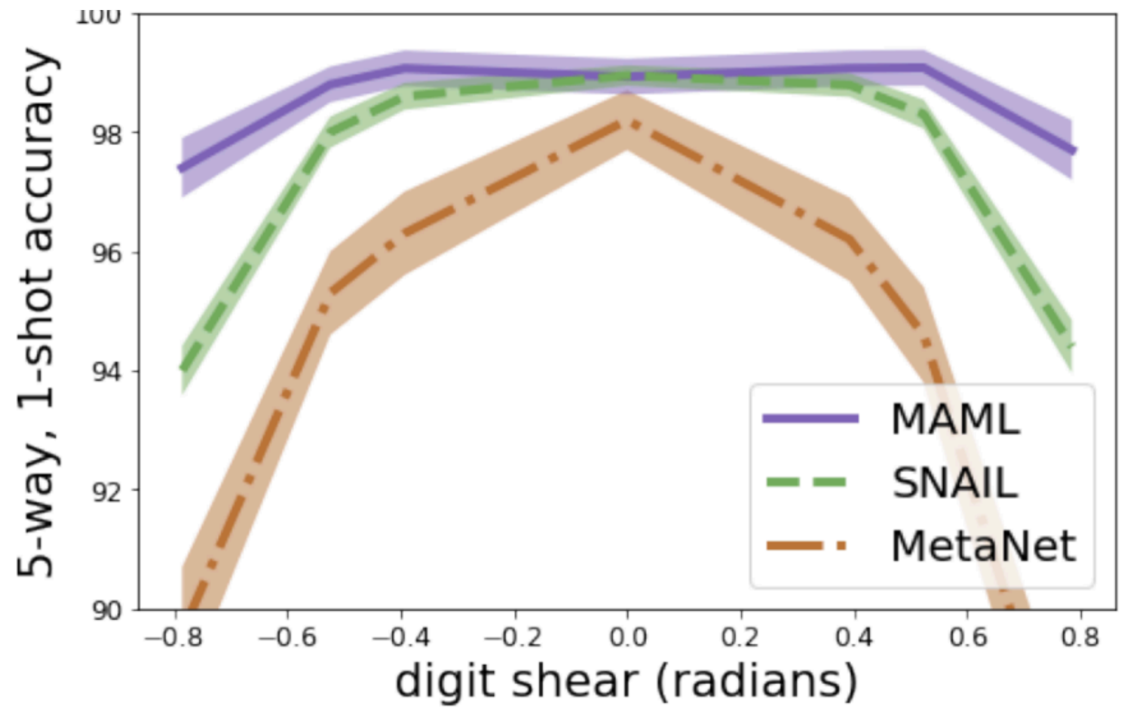
Why? scalability, applicability to a range of domains

Consistency

learned learning procedure will monotonically improve with more data

Why? reduce relliance on meta-training tasks,
good OOD task performance

Recall:



These properties are important for most applications!

Black-box vs. Optimization vs. Non-Parametric

Black-box

+ **complete expressive power**

- **not consistent**

+ easy to combine with **variety of learning problems** (e.g. SL, RL)

- **challenging optimization** (no inductive bias at the initialization)

- often **data-inefficient**

Optimization-based

+ **consistent, reduces** to **GD**

~ **expressive** for **very deep models***

+ **positive inductive bias** at the start of meta-learning

+ handles **varying** & **large K** well

+ **model-agnostic**

- **second-order optimization**

- usually **compute** and **memory** intensive

Non-parametric

+ **expressive** for **most architectures**

~ **consistent** under **certain conditions**

+ entirely **feedforward**

+ **computationally fast** & **easy to optimize**

- **harder to generalize** to **varying K**

- hard to scale to **very large K**

- so far, **limited to classification**

Generally, well-tuned versions of each perform **comparably** on many few-shot benchmarks!

(likely says more about the benchmarks than the methods)

Which method to use depends on your **use-case**.

*for supervised learning settings

Black-box vs. Optimization vs. Non-Parametric

***Algorithmic properties* perspective**

Expressive power

the ability for f to represent a range of learning procedures

Why? scalability, applicability to a range of domains

Consistency

learned learning procedure will monotonically improve with more data

Why? reduce reliance on meta-training tasks,
good OOD task performance

Uncertainty awareness

ability to reason about ambiguity during learning

Why? active learning, calibrated uncertainty, RL
principled Bayesian approaches

We'll discuss this in 2 weeks!

Plan for Today

Non-Parametric Few-Shot Learning

- Siamese networks, matching networks, prototypical networks
- Case study of few-shot student feedback generation

Properties of Meta-Learning Algorithms

- Comparison of approaches

Example Meta-Learning Applications

- Imitation learning, drug discovery, motion prediction, language generation

Application: One-Shot Imitation Learning

(Yu*, Finn* et al. One-Shot Imitation from Observing Humans. RSS 2018)

Tasks:

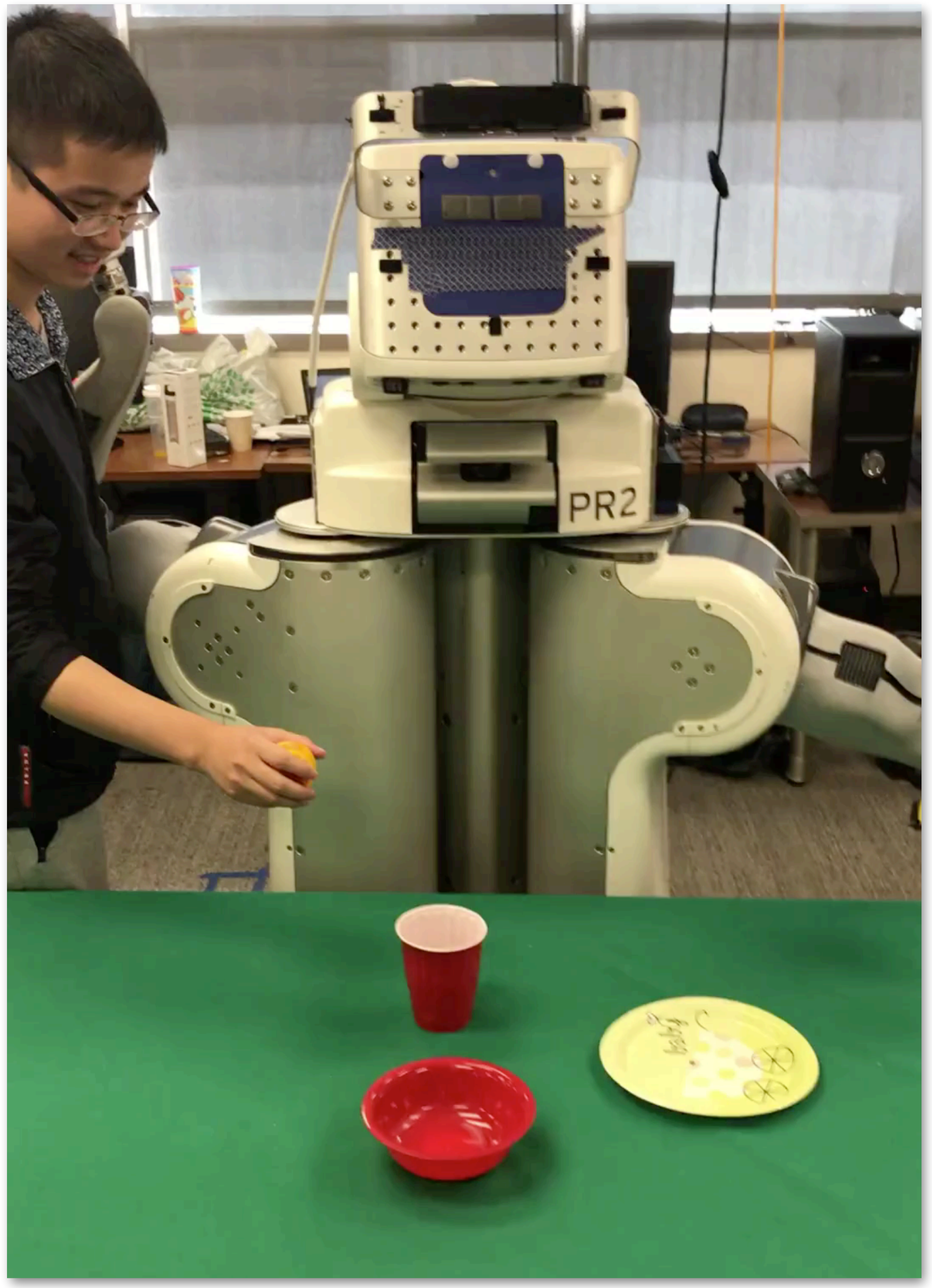
manipulating different objects

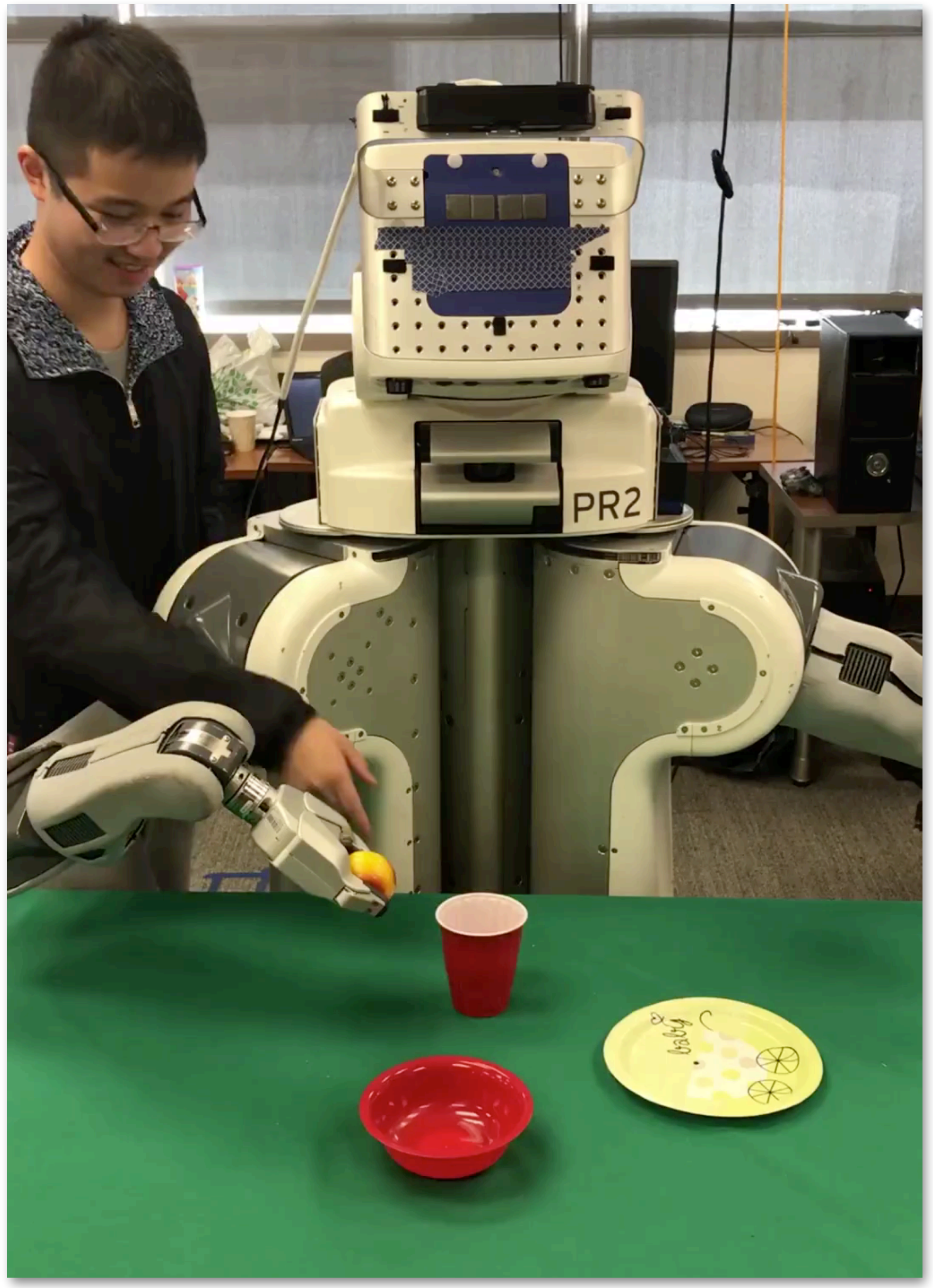
$\mathscr{D}_i^{\mathrm{tr}}$: video of a human

$\mathscr{D}_i^{\mathrm{ts}}$: teleoperated demonstration

Model: optimization-based

MAML with *learned* inner loss





Application: Low-Resource Molecular Property Prediction

(Nguyen et al. Meta-Learning GNN Initializations for Low-Resource Molecular Property Prediction. 2020)

[potentially useful for low-resource drug discovery problems]

Tasks:

Predicting properties & activities of different molecules

$\mathscr{D}_i^{\text{tr}}, \mathscr{D}_i^{\text{ts}}$: different instances

Model: optimization-based

MAML, first-order MAML, ANIL

Gated graph neural net base model

ChEMBL ID	k-NN	Finetune-All	Finetune-Top	FO-MAML	ANIL	MAML
2363236	0.316 ± 0.007	0.328 ± 0.028	0.329 ± 0.023	$\mathbf{0.337 \pm 0.019}$	0.325 ± 0.008	0.332 ± 0.013
1614469	0.438 ± 0.023	0.470 ± 0.034	0.490 ± 0.033	0.489 ± 0.019	0.446 ± 0.044	$\mathbf{0.507 \pm 0.030}$
2363146	0.559 ± 0.026	0.626 ± 0.037	$\mathbf{0.653 \pm 0.029}$	0.555 ± 0.017	0.506 ± 0.034	0.595 ± 0.051
2363366	0.511 ± 0.050	0.567 ± 0.039	0.551 ± 0.048	0.546 ± 0.037	0.570 ± 0.031	$\mathbf{0.598 \pm 0.041}$
2363553	$\mathbf{0.739 \pm 0.007}$	0.724 ± 0.015	0.737 ± 0.023	0.694 ± 0.011	0.686 ± 0.020	0.691 ± 0.013
1963818	0.607 ± 0.041	0.708 ± 0.036	0.595 ± 0.142	0.677 ± 0.026	0.692 ± 0.081	$\mathbf{0.745 \pm 0.048}$
1963945	0.805 ± 0.031	$\mathbf{0.848 \pm 0.034}$	0.835 ± 0.036	0.779 ± 0.039	0.753 ± 0.033	0.836 ± 0.023
1614423	0.503 ± 0.044	0.628 ± 0.058	0.642 ± 0.063	0.760 ± 0.024	0.730 ± 0.077	$\mathbf{0.837 \pm 0.036}^*$
2114825	0.679 ± 0.027	0.739 ± 0.050	0.732 ± 0.051	0.837 ± 0.042	0.759 ± 0.078	$\mathbf{0.885 \pm 0.014}^*$
1964116	0.709 ± 0.042	0.758 ± 0.044	0.769 ± 0.048	0.895 ± 0.023	0.903 ± 0.016	$\mathbf{0.912 \pm 0.013}$
2155446	0.471 ± 0.008	0.473 ± 0.017	0.476 ± 0.013	0.497 ± 0.024	0.478 ± 0.020	$\mathbf{0.500 \pm 0.017}$
1909204	0.538 ± 0.023	0.589 ± 0.031	0.577 ± 0.039	0.592 ± 0.043	0.547 ± 0.029	$\mathbf{0.601 \pm 0.027}$
1909213	0.694 ± 0.009	0.742 ± 0.015	$\mathbf{0.759 \pm 0.012}$	0.698 ± 0.024	0.694 ± 0.025	0.729 ± 0.013
3111197	0.617 ± 0.028	0.663 ± 0.066	0.673 ± 0.071	0.636 ± 0.036	0.737 ± 0.035	$\mathbf{0.746 \pm 0.045}$
3215171	0.480 ± 0.042	0.552 ± 0.043	0.551 ± 0.045	0.729 ± 0.031	0.700 ± 0.050	$\mathbf{0.764 \pm 0.019}$
3215034	0.474 ± 0.072	0.540 ± 0.156	0.455 ± 0.189	$\mathbf{0.819 \pm 0.048}$	0.681 ± 0.042	0.805 ± 0.046
1909103	0.881 ± 0.026	$\mathbf{0.936 \pm 0.013}$	0.921 ± 0.020	0.877 ± 0.046	0.730 ± 0.055	0.900 ± 0.032
3215092	0.696 ± 0.038	0.777 ± 0.039	0.791 ± 0.042	0.877 ± 0.028	0.834 ± 0.026	$\mathbf{0.907 \pm 0.017}$
1738253	0.710 ± 0.048	0.860 ± 0.029	0.861 ± 0.025	0.885 ± 0.033	0.758 ± 0.111	$\mathbf{0.908 \pm 0.011}$
1614549	0.710 ± 0.035	0.850 ± 0.041	0.860 ± 0.051	0.930 ± 0.022	0.860 ± 0.034	$\mathbf{0.947 \pm 0.014}$
Avg. Rank	5.4	3.5	3.5	3.1	4.0	**1.7**

Application: Few-Shot Human Motion Prediction

(Gui et al. Few-Shot Human Motion Prediction via Meta-Learning. ECCV 2018)

[potentially useful for human-robot interaction, autonomous driving]

Tasks:

Different human users & motions

$\mathscr{D}_i^{\text{tr}}$: past K time steps of motion

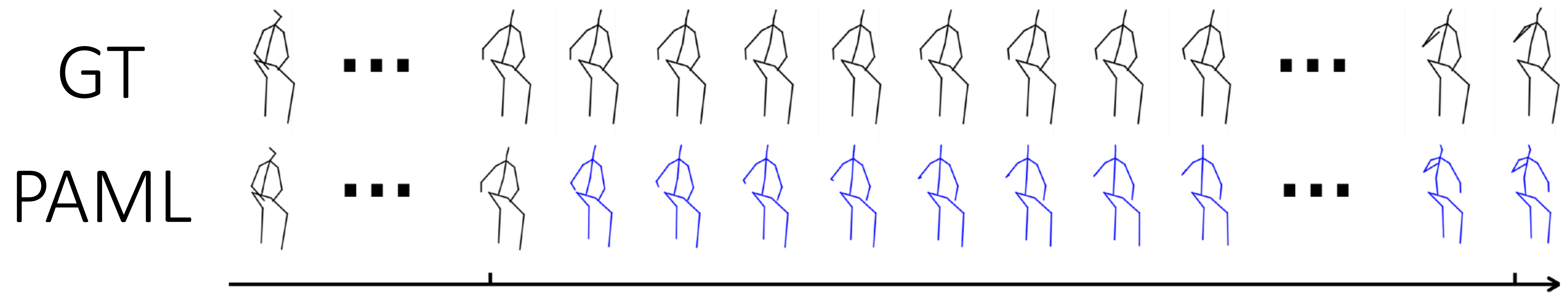
$\mathscr{D}_i^{\text{ts}}$: future second(s) of motion

Model:

optimization-based/black-box hybrid

MAML with additional learned update rule

Recurrent neural net base model



		Walking						Eating					
milliseconds		80	160	320	400	560	1000	80	160	320	400	560	1000
residual sup. [32] w/ (Baselines)	$\text{Scratch}_{\text{spec}}$	1.90	1.95	2.16	2.18	1.99	2.00	2.33	2.31	2.30	2.30	2.31	2.34
	$\text{Scratch}_{\text{agn}}$	1.78	1.89	2.20	2.23	2.02	2.05	2.27	2.16	2.18	2.27	2.25	2.31
	$\text{Transfer}_{\text{ots}}$	0.60	0.75	0.88	0.93	1.03	1.26	0.57	0.70	0.91	1.04	1.19	1.58
	Multi-task	0.57	0.71	0.79	0.85	0.96	1.12	0.59	0.68	0.83	0.93	1.12	1.33
	$\text{Transfer}_{\text{ft}}$	0.44	0.55	0.85	0.95	0.74	1.03	0.61	0.65	0.74	0.78	0.86	1.19
Meta-learning (Ours)	PAML	**0.35**	**0.47**	**0.70**	**0.82**	**0.80**	**0.83**	**0.36**	**0.52**	**0.65**	**0.70**	**0.71**	**0.79**

		Smoking						Discussion					
milliseconds		80	160	320	400	560	1000	80	160	320	400	560	1000
residual sup. [32] w/ (Baselines)	$\text{Scratch}_{\text{spec}}$	2.88	2.86	2.85	2.83	2.80	2.99	3.01	3.13	3.12	2.95	2.62	2.99
	$\text{Scratch}_{\text{agn}}$	2.53	2.61	2.67	2.65	2.71	2.73	2.77	2.79	2.82	2.73	2.82	2.76
	$\text{Transfer}_{\text{ots}}$	0.70	0.84	1.18	1.23	1.38	2.02	0.58	0.86	1.12	1.18	1.54	2.02
	Multi-task	0.71	0.79	1.09	1.20	1.25	1.23	0.53	0.82	1.02	1.17	1.33	1.97
	$\text{Transfer}_{\text{ft}}$	0.87	1.02	1.25	1.30	1.45	2.06	0.57	0.82	1.11	1.11	1.37	2.08
Meta-learning (Ours)	PAML	**0.39**	**0.66**	**0.81**	**1.01**	**1.03**	**1.01**	**0.41**	**0.71**	**1.01**	**1.02**	**1.09**	**1.12**

mean angle error w.r.t. prediction horizon

Closing note for today

$\mathscr{D}_i^{\text{tr}}$ and $\mathscr{D}_i^{\text{ts}}$ do not need to be sampled independently from $\mathscr{D}_i$.

$\mathscr{D}_i^{\text{tr}}$ could have:

- noisy labels
- weakly supervised
- domain shift
- etc.

Plan for Today

Non-Parametric Few-Shot Learning

- Siamese networks, matching networks, prototypical networks
- Case study of few-shot student feedback generation

Properties of Meta-Learning Algorithms

- Comparison of approaches

Example Meta-Learning Applications

- Imitation learning, drug discovery, motion prediction, language generation

Goals for by the end of lecture:

- Basics of non-parametric few-shot learning techniques (& how to implement)
- Trade-offs between black-box, optimization-based, and non-parametric meta-learning
- Familiarity with applied formulations of meta-learning

Course Logistics

Lecture Topics

Done with meta-learning algorithms!

Next week: unsupervised pre-training

Coursework

Homework 1 due tonight.

Homework 2 released, due Mon 10/24.

Project mentors to be assigned this week.

Project proposal due next Weds 10/19.

(graded lightly, for your benefit)