Learning from little data

Subhransu Maji
University of Massachusetts

May 10, 2022

UMass Amherst | College of Information & Computer Sciences

Deep learning — reality vs. practice

source: reddit

Datasets + Models + Hyperparams

Issues with learning from little data

Not just computational!

- Overfitting
- Bias
- Calibration
- Label noise
- …

Little data

solutions

Unlabeled examples

- Self-/Semi-supervised learning
- Active learning

Related datasets

- Transfer learning
- Multi-tasking
- Meta learning

Pre-trained models

- Robust finetuning, adaptors

Today

Learning to represent tasks [ICCV’19, ECCV’20, CVPR’21]

- Build vector representations of tasks & learn their relations
- Goal: amortize solution search across tasks & visualization

Learning with diverse labeling styles [AAAI’19, BMVC’21, arXiv’22]

- learn from diverse (coarse) labels
- Goal: use related datasets to improve performance

Task embedding (TASK2VEC)

If we have a universal **vectorial representation** of **tasks,** we can frame all sorts of interesting computer vision application engineering problems as **machine-learning** problems.

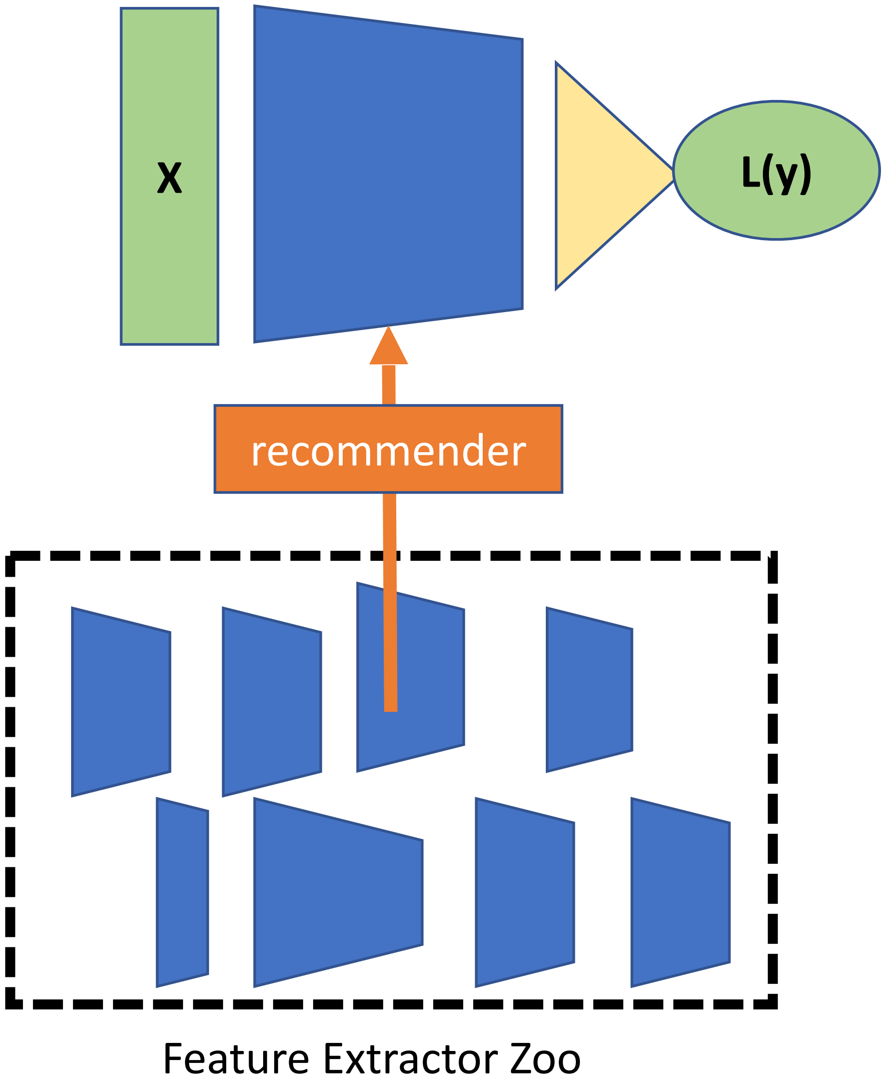
Task = {images, labels, loss}

$$x = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

What are similar tasks?
What architecture should I use?
What pre-training dataset?
What hyper parameters?
Do I need more training data?
How difficult is this task?
.
.
.

Achille at al., ICCV'19 @ AWS AI

Application: Model recommendation



Brute Force:

Input: Task = (**dataset, loss**)

For each feature extractor architecture **F**:
1. Train **classifier** on **F(dataset)**
2. Compute validation performance

Output: best performing model

Task Embedding:

Input: Task = (**dataset, loss**)

1. Compute task embedding **t = E(**Task**)**
2. Predict best extractor **F = M(t)**
2. Train **classifier** on **F(dataset)**
3. Compute validation performance

Output: best performing model

Similarity measures on the space of tasks

Domain similarity

Unbiased look at dataset bias, Torralba and Efros, CVPR 11

Similarity measures on the space of tasks

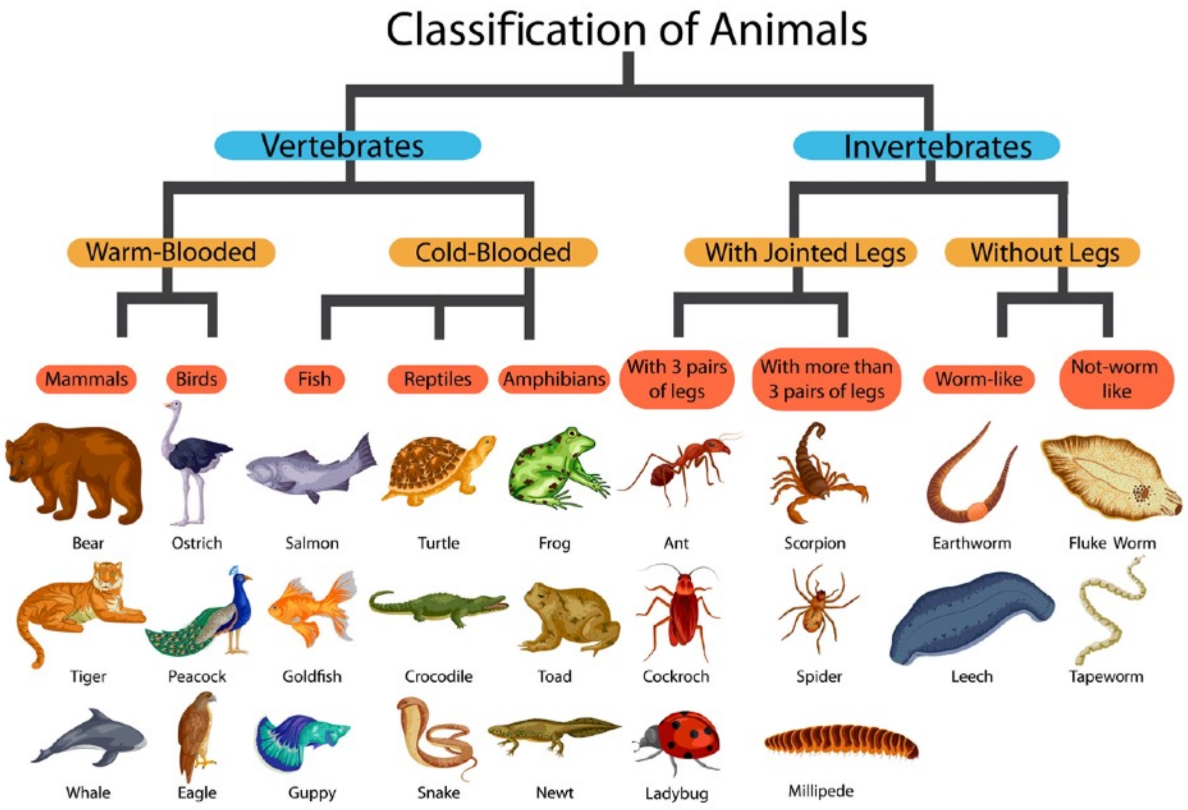
Domain similarity

Range / label similarity

- e.g., Taxonomic distance

$$D_{\text{tax}}(t_a, t_b) = \min_{i \in S_a, j \in S_b} d(i, j),$$

D(***bird*** task, ***mammal*** task) < D(***bird*** task, ***worm*** task)



Similarity measures on the space of tasks

Domain similarity

Range / label similarity

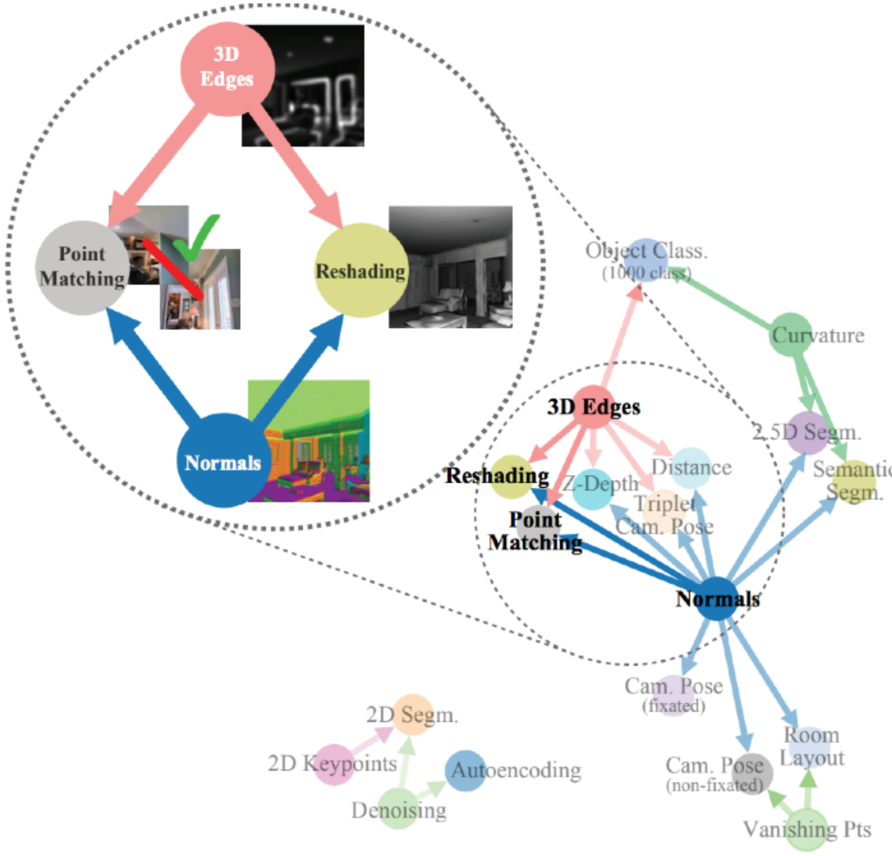
- e.g., Taxonomic distance

$$D_{\text{tax}}(t_a, t_b) = \min_{i \in S_a, j \in S_b} d(i, j),$$

Transfer "distance"

- Train on task *a* followed by *b*

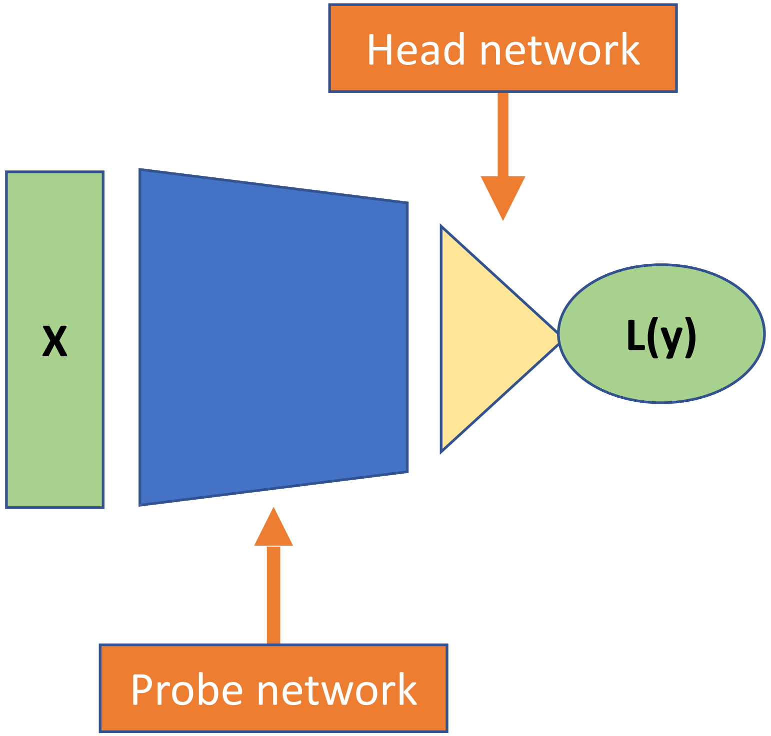
$$D_{\text{ft}}(t_a \to t_b) = \frac{\mathbb{E}[\ell_{a \to b}] - \mathbb{E}[\ell_b]}{\mathbb{E}[\ell_b]}$$



Taskonomy: Disentangling Task Transfer Learning, Amir Zamir, Alexander Sax, William Shen, Leonidas Guibas, Jitendra Malik, Silvio Savarese, CVPR 18

Task embedding using a probe network

1. Given a **task**, train a classifier with the **task loss** on features from a generic “probe network”
2. Compute gradients of **probe network** parameters (θ) w.r.t. task loss (e.g., log-likelihood)
3. Use statistics of the probe parameter **gradients** as the fixed dimensional **task embedding**



Task embedding as the Fisher Information

1. Given a **task**, train a classifier with the **task loss** on features from a generic "probe network"
2. Compute gradients of **probe network** parameters (θ) w.r.t. task loss (e.g., log-likelihood)
3. Use statistics of the probe parameter **gradients** as the fixed dimensional **task embedding**

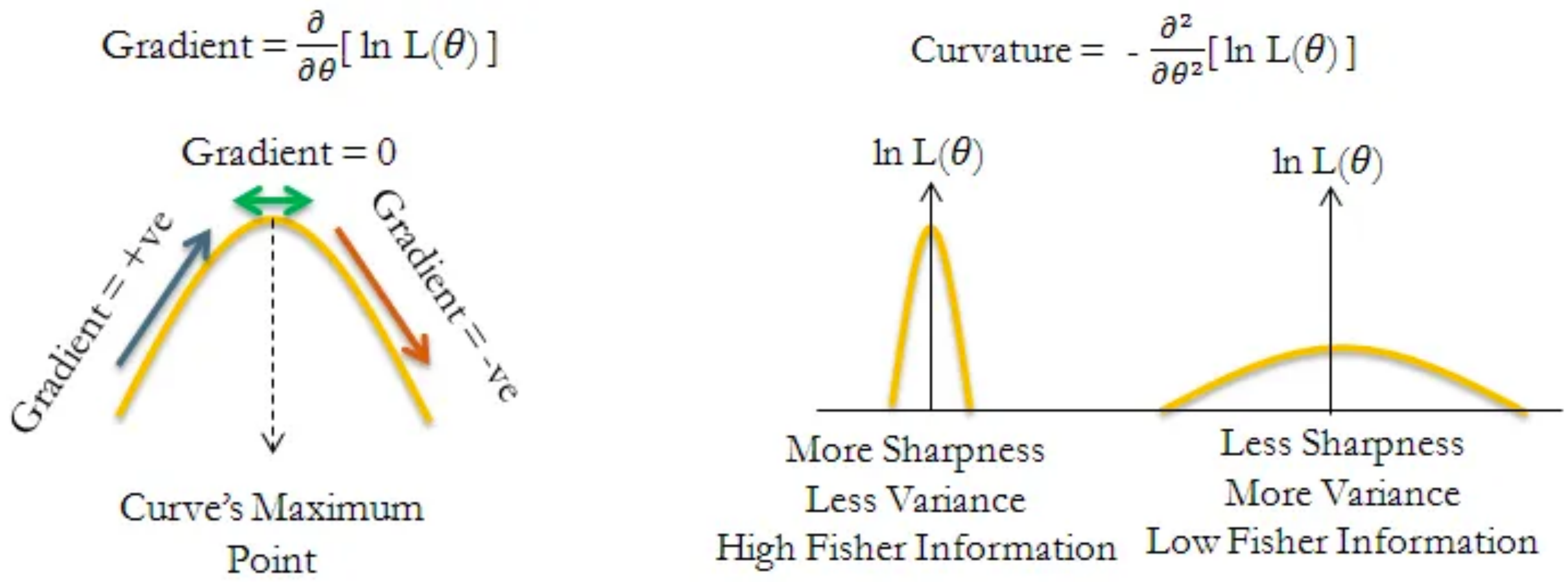
$$\tilde{F} = \sum_n \left[\nabla_\theta \log p_\theta(y_n|x_n) \nabla_\theta \log p_\theta(y_n|x_n)^\top\right]$$

Intuition: F provides information about the **sensitivity** of the task performance to small perturbations of **parameters** in the probe network

$$\theta' = \theta + \delta\theta$$

$$\mathbb{E}_{x\sim\hat{p}}\, KL\, p_{\theta'}(y|x) p_\theta(y|x) = \delta\theta \cdot F \cdot \delta\theta + o(\delta\theta^2),$$

Curvature and Fisher Information



https://www.gaussianwaves.com/2012/10/score-and-fisher-information-estimator-sensitivity

Practical issues and properties of TASK2VEC

1. For realistic CV tasks we want to use deep CNNs (e.g., **ResNet30**) and estimate FIM for all the parameters
2. **Challenge:** FIM can be hard to estimate (noisy loss landscape; high dimensions; small training set)
3. **Approximate FIM**
    1. Restrict it to a diagonal
    2. Restrict it a single value per filter in a CNN layer
    3. Robust estimation via perturbation

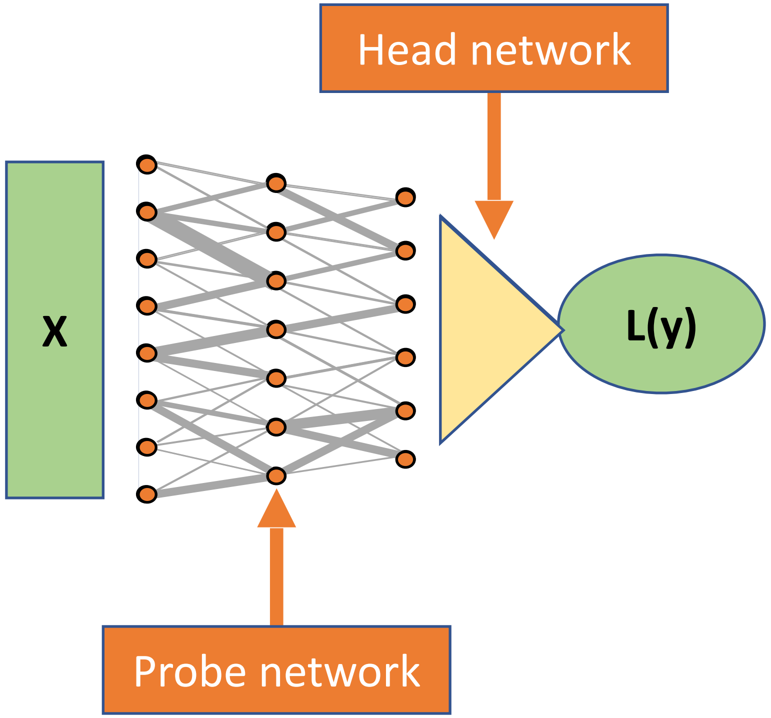
1. **Invariance** to label space
2. Encodes task **difficulty**
3. Encodes task **domain**
4. Encodes **useful features** for the task

$$\tilde{F} = \sum_n \left[\nabla_\theta \log p_\theta(y_n|x_n) \nabla_\theta \log p_\theta(y_n|x_n)^\top \right]$$

Task embedding illustration

t_1

$\phi(t_1)$



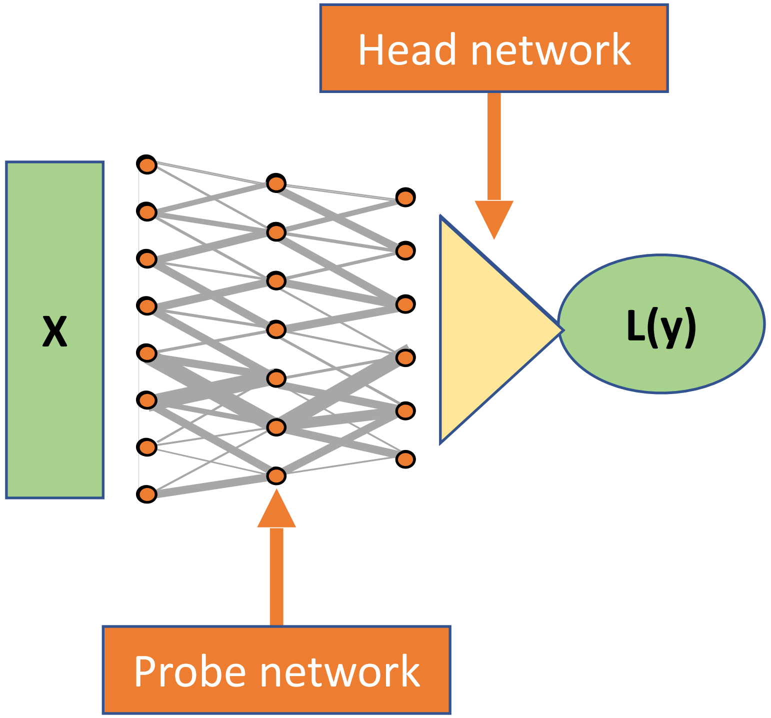
Input: Task = (**dataset, loss**)

1. Initialize the **probe network** and the **head network** (e.g., **linear classifier**)
2. Train the **head network** by minimizing the loss
3. Compute the (approximate) FIM of the **probe network**

Task embedding illustration

t_2

$\phi(t_2)$



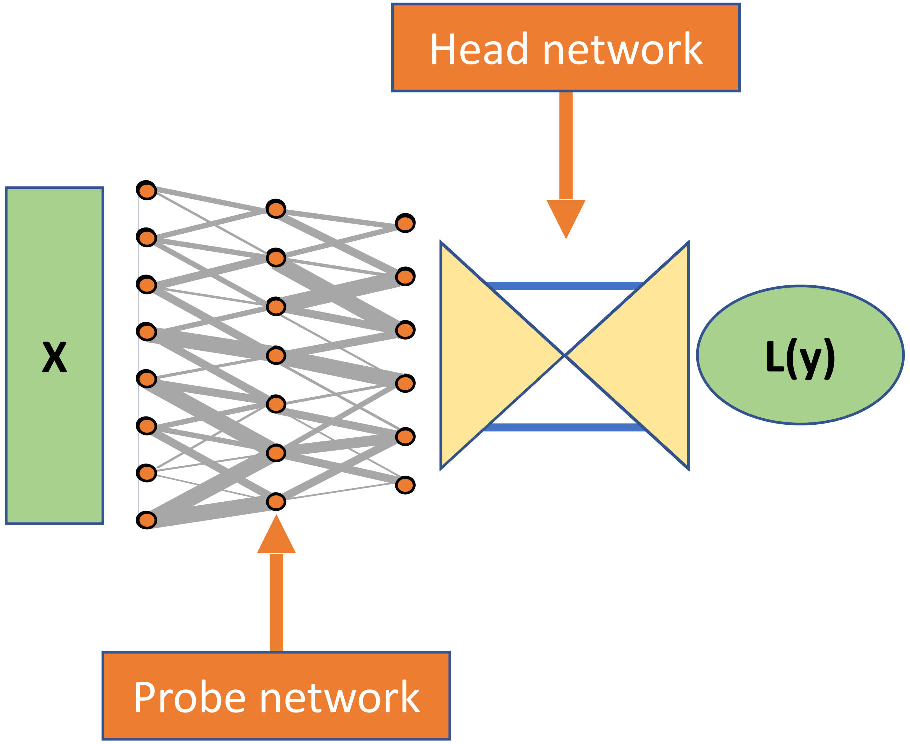
Input: Task = (**dataset, loss**)

1. Initialize the **probe network** and the **head network** (e.g., **linear classifier**)
2. Train the **head network** by minimizing the loss
3. Compute the (approximate) FIM of the **probe network**

Task embedding illustration

t_3

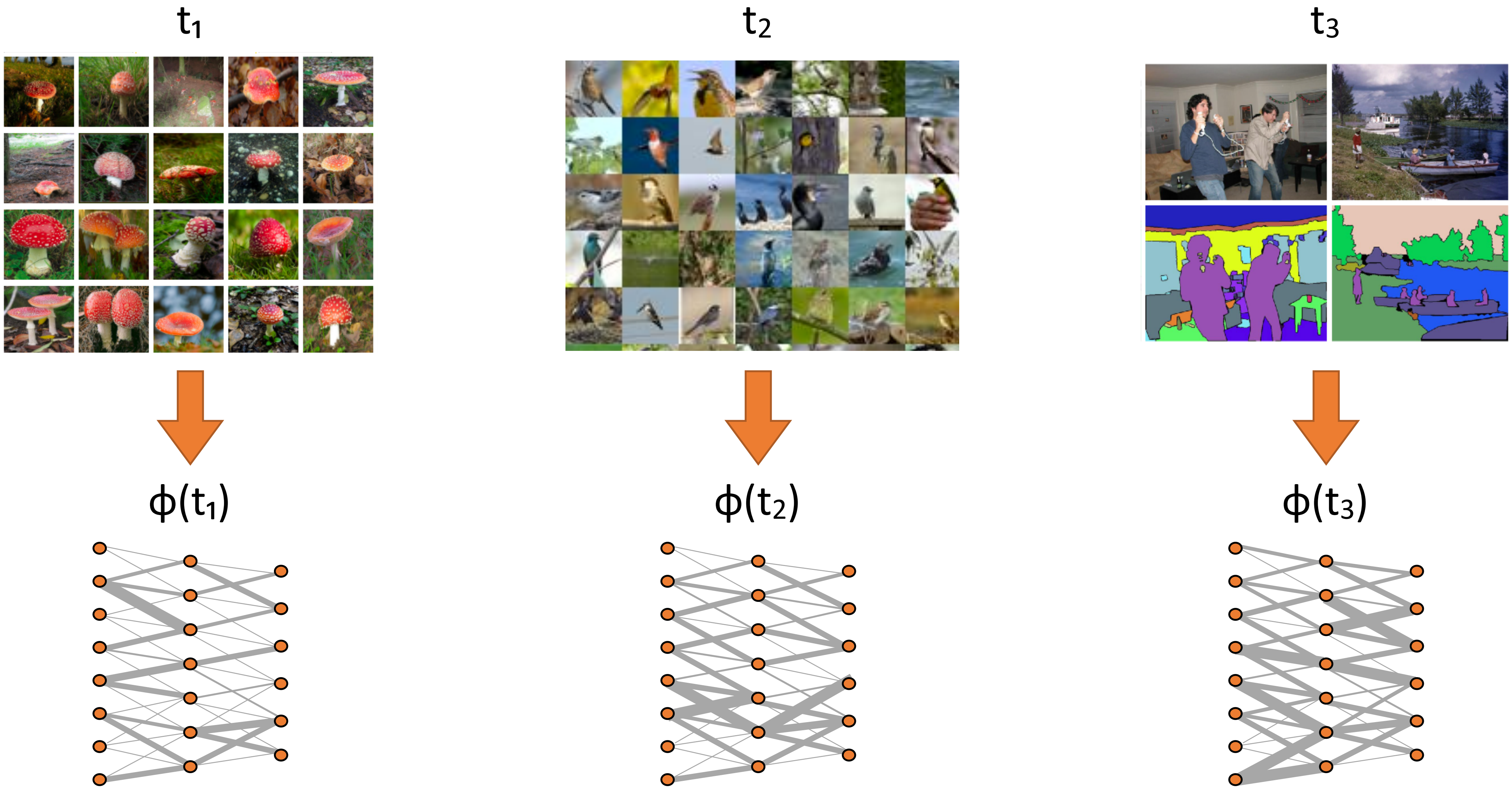
$\phi(t_3)$



Input: Task = (**dataset, loss**)

1. Initialize the **probe network** and the **head network** (e.g., **UNet**)
2. Train the **head network** by minimizing the loss
3. Compute the (approximate) FIM of the **probe network**

Task embedding illustration



Distance measures on TASK2VEC embedding

Symmetric distance

$$d_{\text{sym}}(F_a, F_b) = d_{\cos}\left(\frac{F_a}{F_a + F_b}, \frac{F_b}{F_a + F_b}\right)$$

Asymmetric "distance"

$$d_{\text{asym}}(t_a \to t_b) = d_{\text{sym}}(t_a, t_b) - \alpha d_{\text{sym}}(t_a, t_0)$$

task embedding for the "trivial" task

Task Zoo

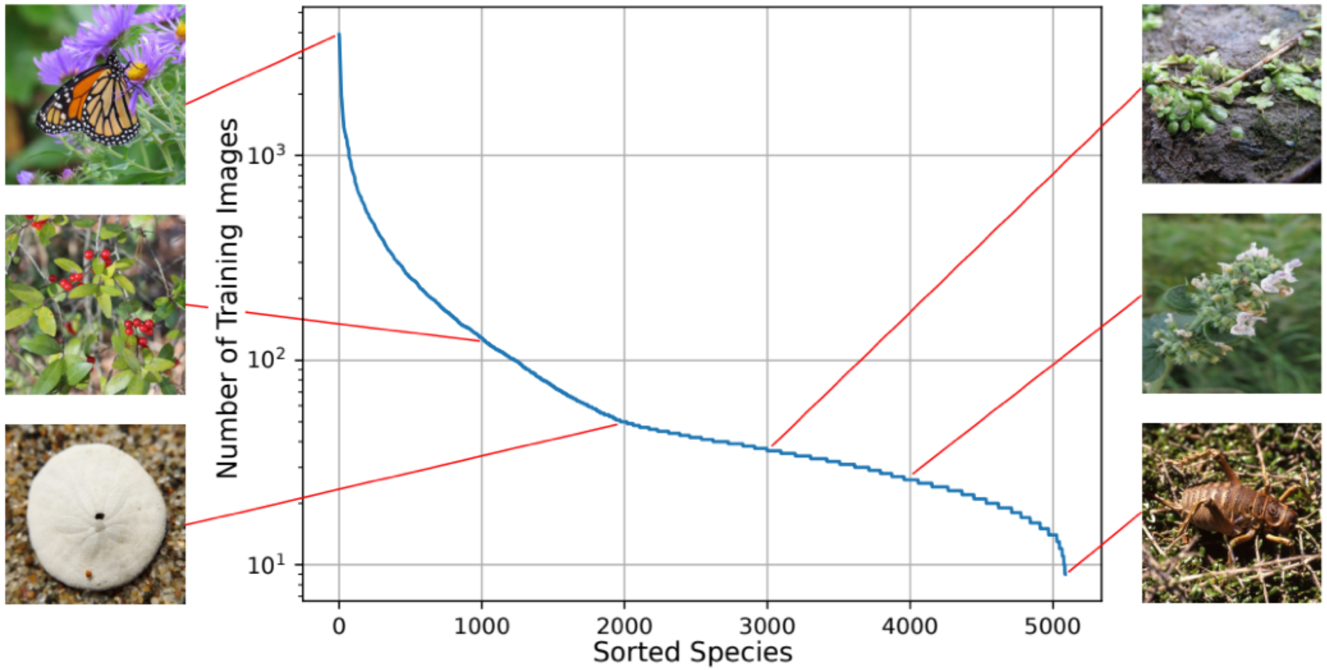
Tasks [1460]

- iNaturalist [207]
- CUB 200 [25]
- iMaterialist [228]
- DeepFashion [1000]

Task Zoo

Tasks [1460]

- **iNaturalist [207]**
- CUB 200 [25]
- iMaterialist [228]
- DeepFashion [1000]



Super-Class	Class
Plantae	2,101
Insecta	1,021
Aves	964
Reptilia	289
Mammalia	186
Fungi	121
Amphibia	115
Mollusca	93
Animalia	77
Arachnida	56
Actinopterygii	53
Chromista	9
Protozoa	4

Task Zoo

Tasks [1460]

- iNaturalist [207]
- CUB 200 [25]
- iMaterialist [228]
- **DeepFashion [1000]**

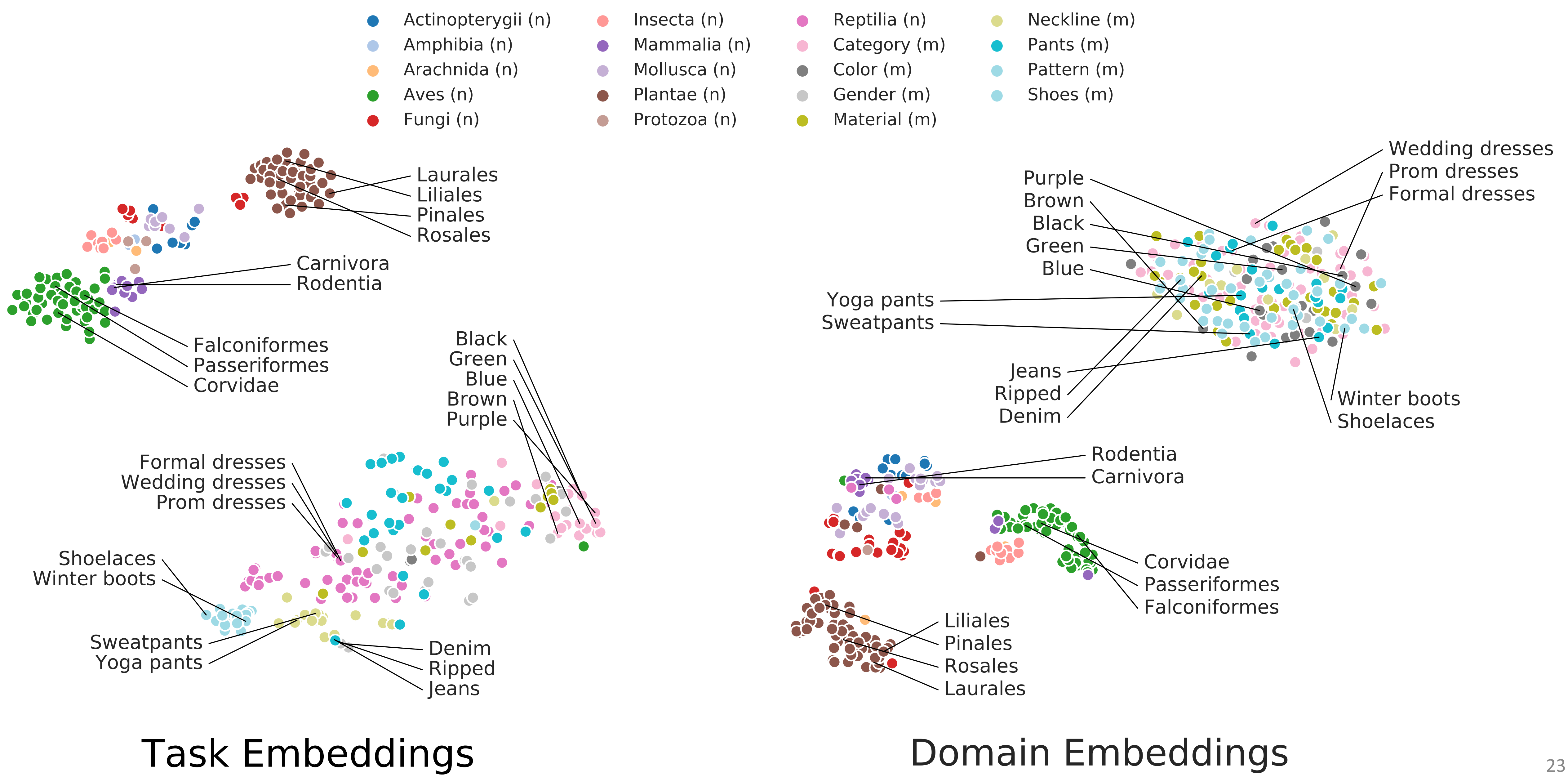
Task Zoo

Tasks [1460]

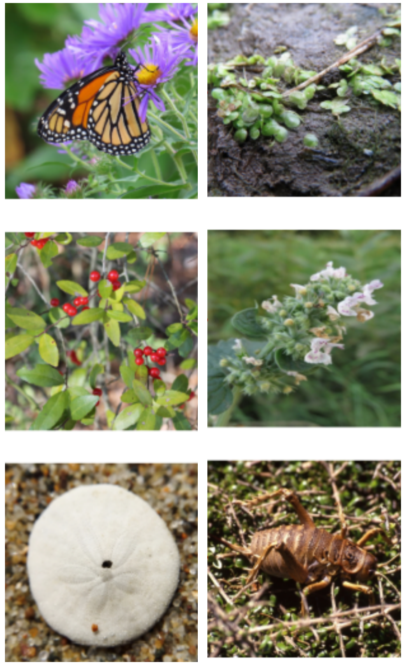
- iNaturalist [207]
- CUB 200 [25]
- iMaterialist [228]
- **DeepFashion [1000]**

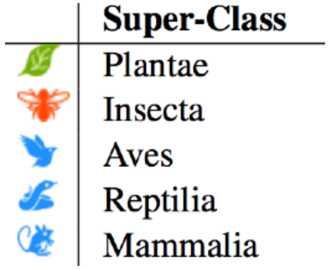
- Few tasks > 10K training samples but most have 100-1000 samples

Experiment: TASK2VEC vs DOMAIN2VEC

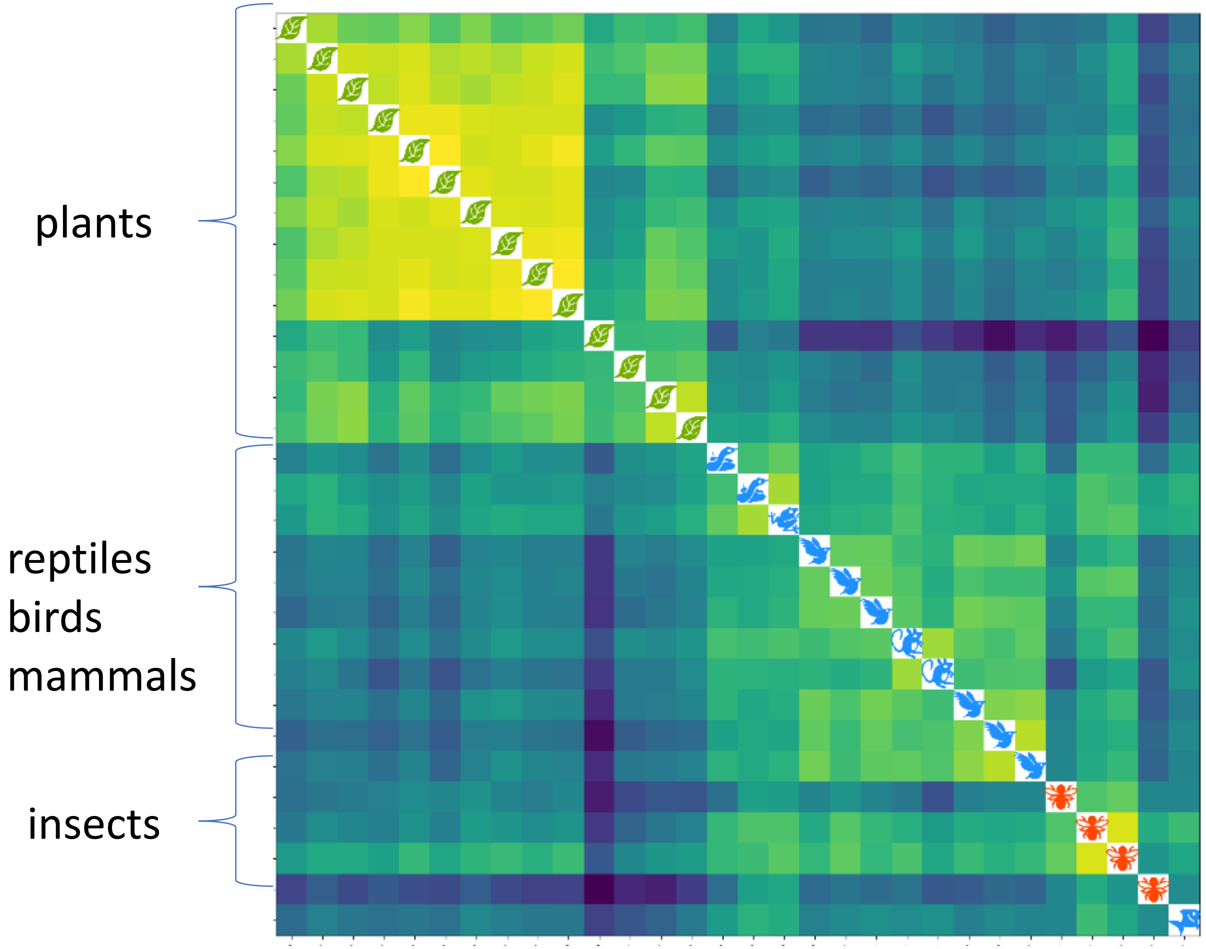


Experiment: TASK2VEC recapitulates iNaturalist taxonomy





Task embedding cosine similarity

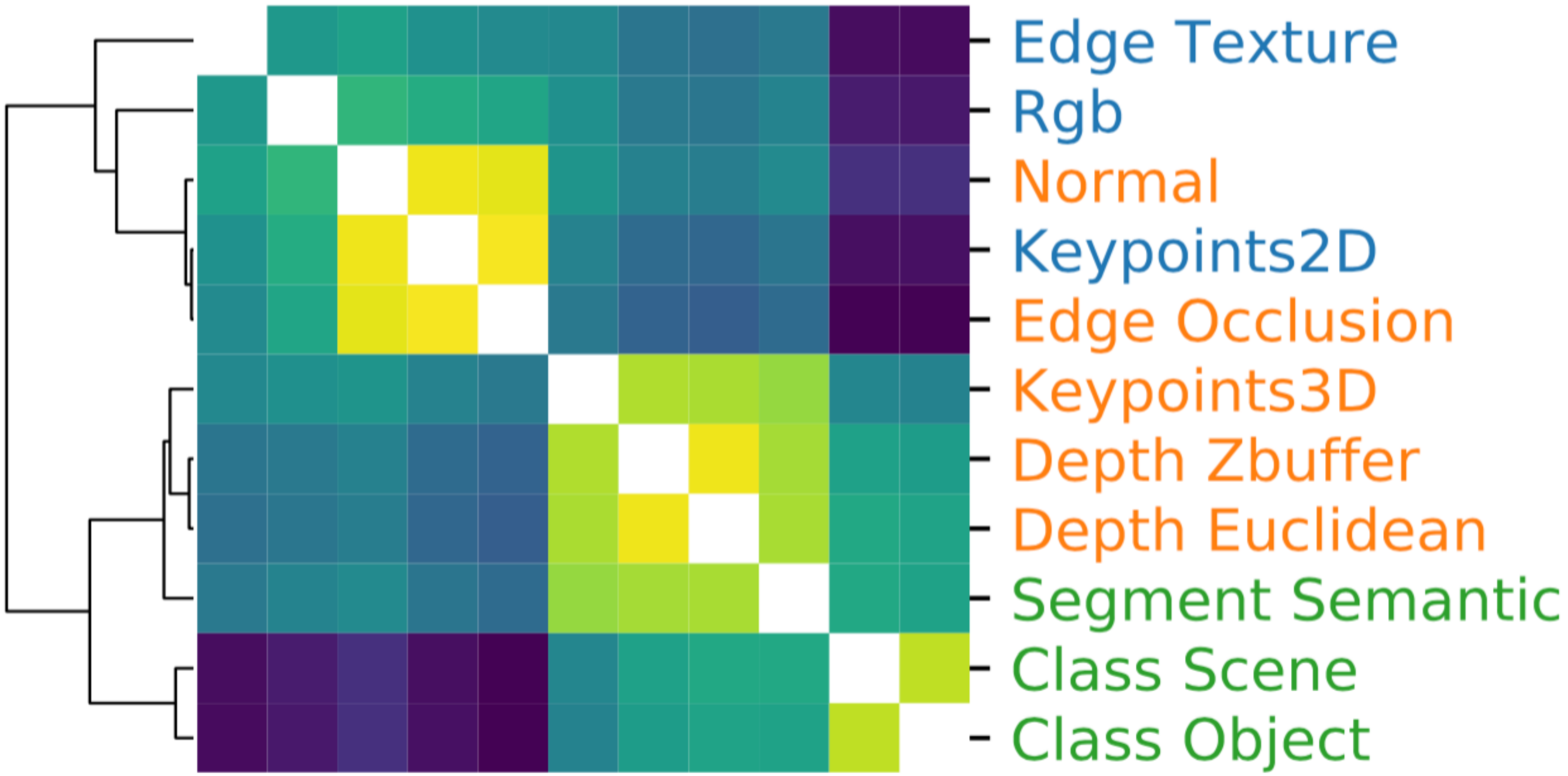


ResNet trained on ImageNet as probe network

Experiment: TASK2VEC recovers "Taskonomy"

Taskonomy: Disentangling Task Transfer Learning, Amir Zamir, Alexander Sax, William Shen, Leonidas Guibas, Jitendra Malik, Silvio Savarese, CVPR 18

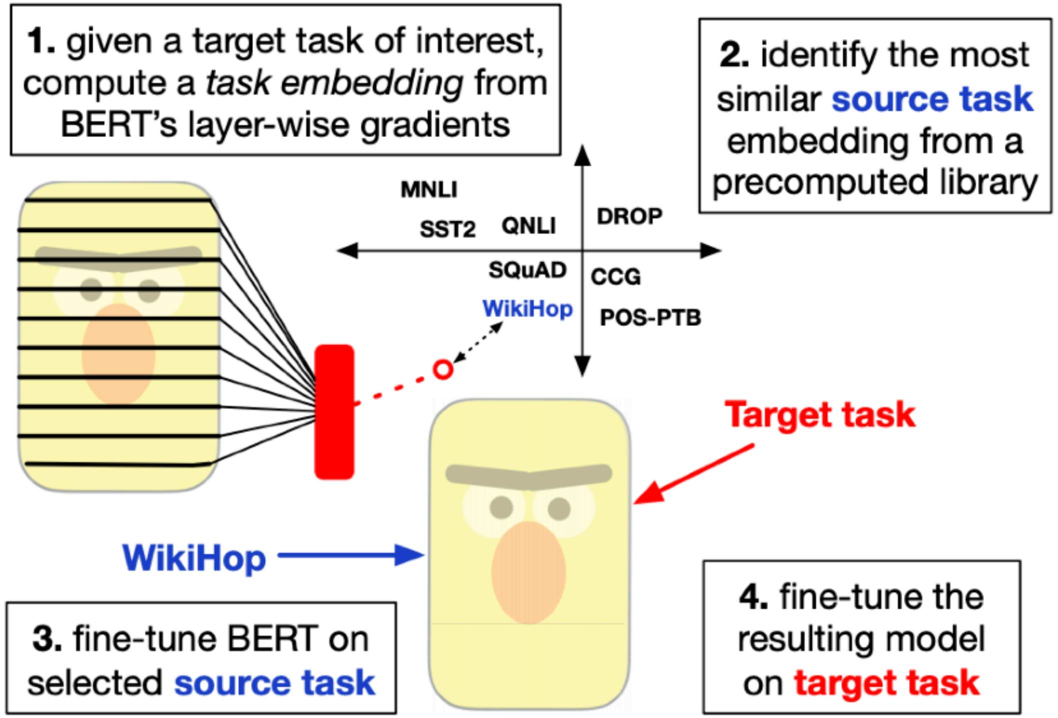
Task embedding cosine similarity

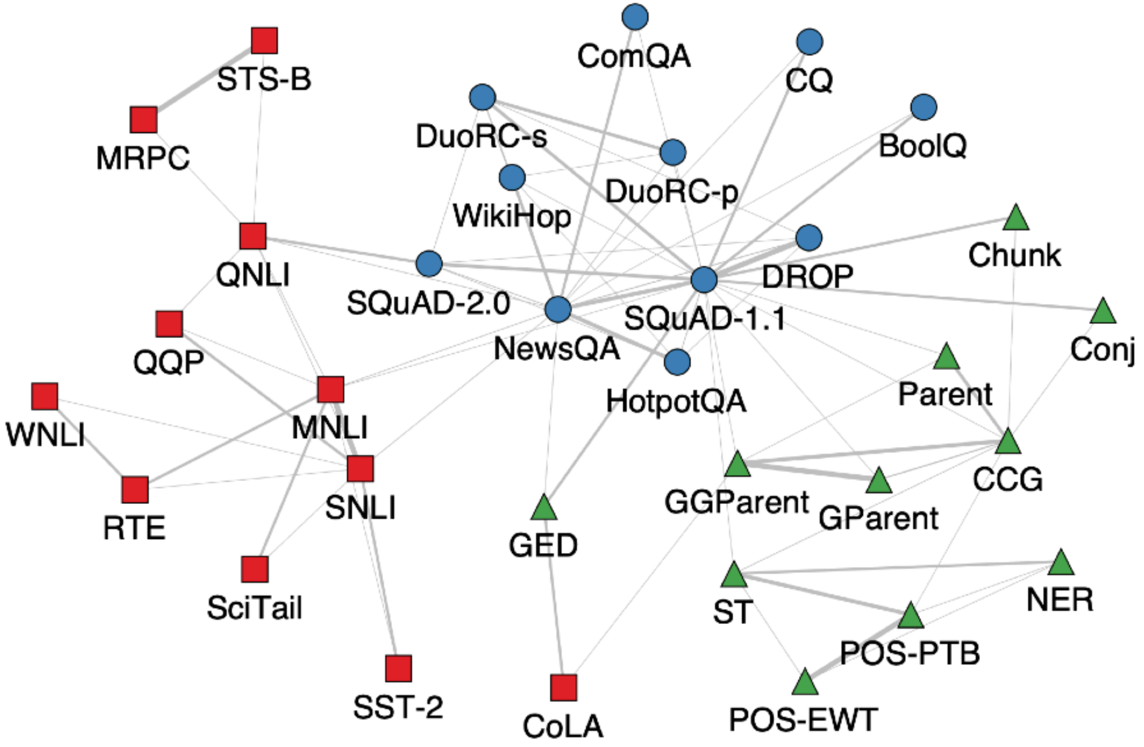


Classifier "head" replaced by a fully-convolutional layer.

Requires far less compute (5 GPU hours for the whole matrix).

Also works for natural language tasks





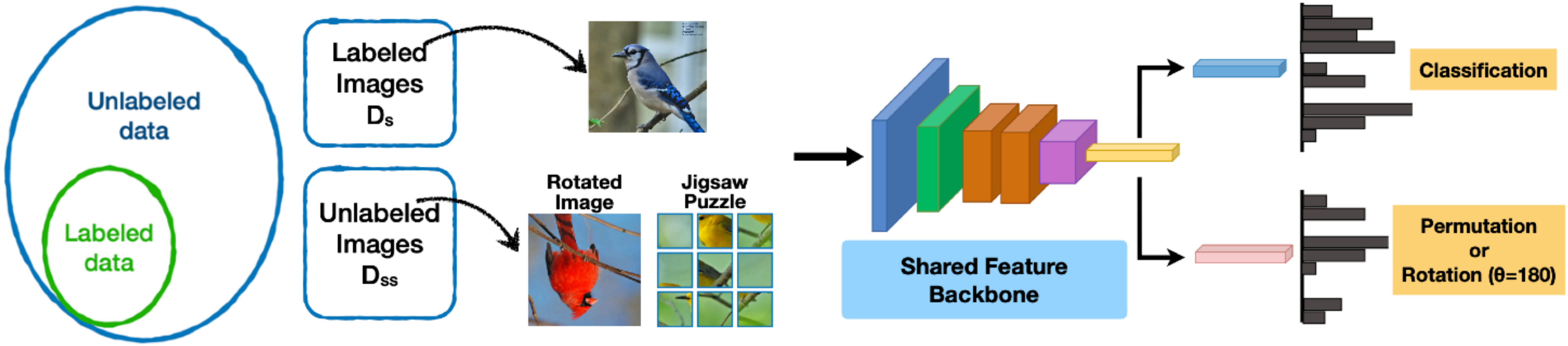
Task Embedding

Exploring and Predicting Transferability across NLP Tasks, Vu et al., EMNLP 2020

Modeling domains can be useful

Does unlabeled data improve few-shot learning?

- Yes, as long as unlabeled data domain **(D_{ss})** ≈ task domain **(D_s)**



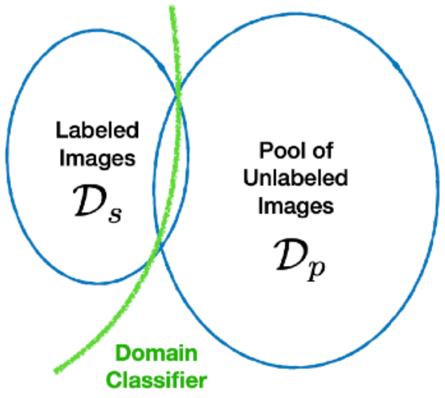
References:

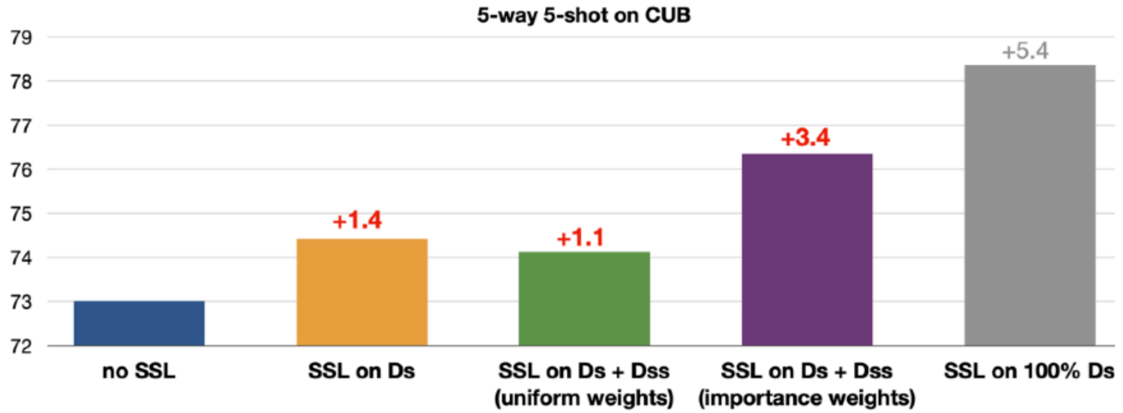
- Shot in the Dark: Few-shot Learning with No Base Class Labels, L2ID Workshop, CVPR’21
- **When does Self-Supervision improve Few-Shot Learning? ECCV’20**
- A Realistic Evaluation of Semi-Supervised Learning for Fine-Grained Classification, CVPR’21

Modeling domains can be useful

Does unlabeled data improve few-shot learning?

- Yes, as long as unlabeled data domain **(D_{ss})** ≈ task domain **(D_s)**





ProtoNet
(baseline)

References:

- Shot in the Dark: Few-shot Learning with No Base Class Labels, L2ID Workshop, CVPR'21
- **When does Self-Supervision improve Few-Shot Learning? ECCV'20**
- A Realistic Evaluation of Semi-Supervised Learning for Fine-Grained Classification, CVPR'21

Today

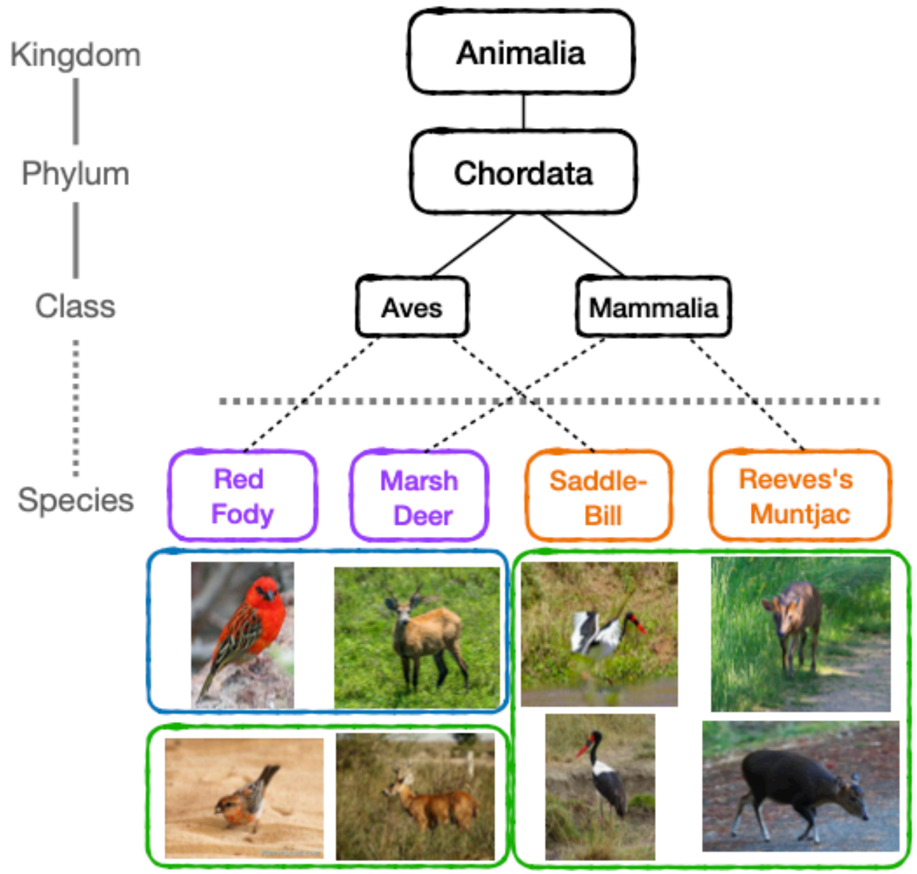
Learning to represent tasks [ICCV’19, ECCV’20, CVPR’21]

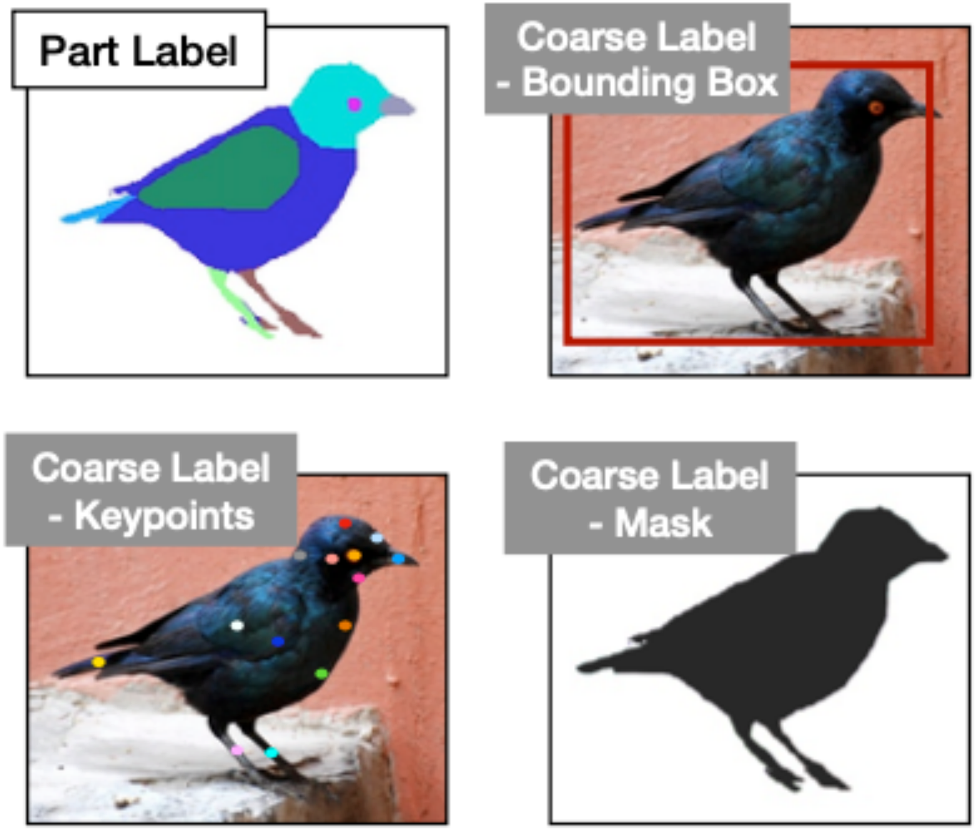
- Build vector representations of tasks & learn their relations
- Goal: amortize solution search across tasks & visualization

Learning with diverse labeling styles [AAAI’19, BMVC’21, arXiv’22]

- learn from diverse (coarse) labels
- Goal: use related datasets to improve performance

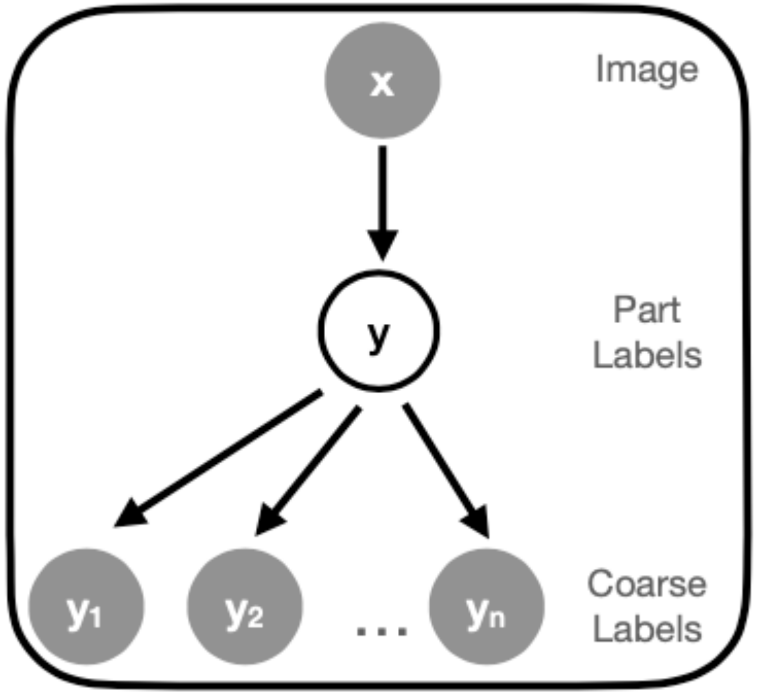
Learning from coarsely labeled datasets



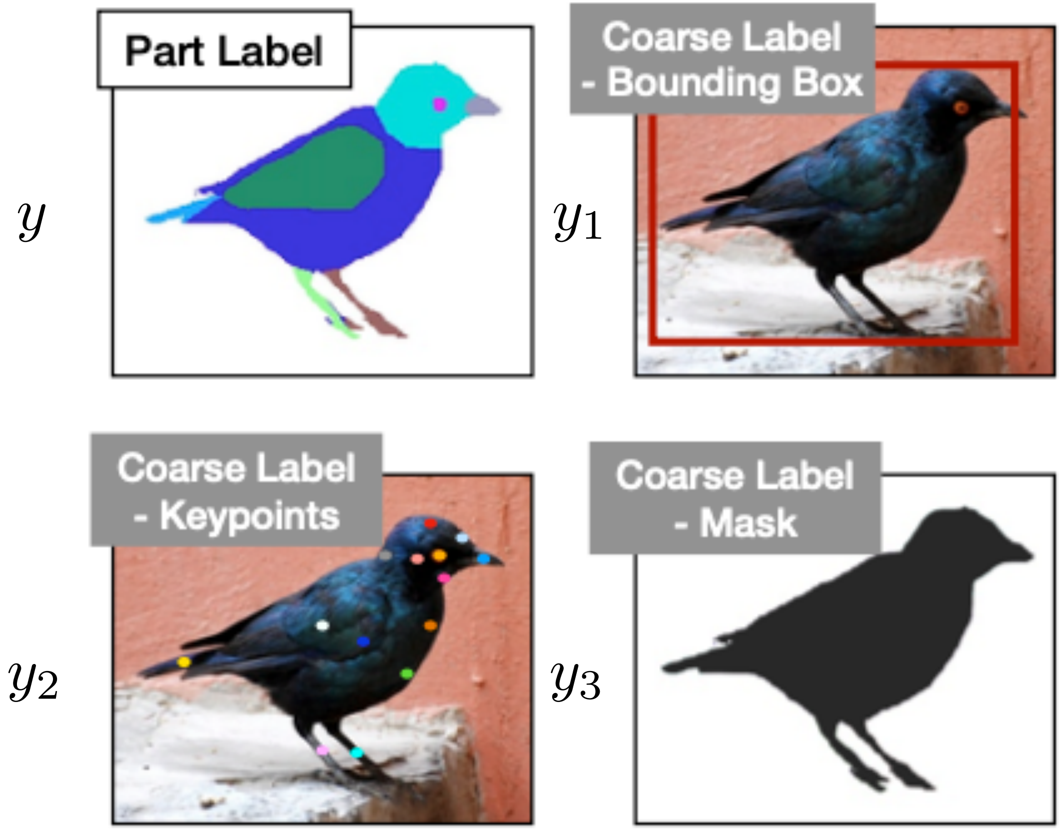


Coarsely labeled datasets are easier to find

A probabilistic model



$$p(y, y_1, \ldots, y_n|x) = p(y|x) \prod_{i=1}^{n} p(y_1|y)$$



Assumption — coarse labels are independent given the part labels

Learning

Maximum likelihood estimation:

$$\max_\theta \mathcal{L}(\theta) = \log p(y_1, y_2, \ldots, y_n | x, \theta).$$

$$\geq \sum_y q(y) \left[\log p(y|x) \prod_{i=1}^{n} p(y_i|y, \theta) \right] + H(q) := \mathcal{F}(q, \theta). \quad \text{(ELBO)}$$

EM algorithm:

– **E step:** maximize $\mathcal{F}(q, \theta)$ wrt distribution over y given the parameters:

$$q^{(k)}(y) = \arg\max_{q(y)} \mathcal{F}(q(y), \theta^{(k-1)}).$$

– **M step:** maximize $\mathcal{F}(q, \theta)$ wrt parameters given the distribution $q(y)$:

$$\theta^{(k)} = \arg\max_\theta \mathcal{F}(q^{(k)}(y), \theta) = \arg\max_\theta \sum_y q^{(k)}(y) \log p(y, y_1, y_2, \ldots y_n | x, \theta)$$

Example: Keypoints and Mask Supervision

Parameterization

- $p(y|x) \propto \exp(-\alpha|y-\mu(x)|)$, $\mu(x)$ is distribution over parts
- $p(y_{kp}|y) \propto \exp(-\lambda|y_{kp}-\mu_{kp}(y)|)$, $\mu_{kp}(y)$ is the keypoints given parts
- $p(y_{mask}|y) \propto B(y_{mask}, \mu_{mask}(y))$, $\mu_{mask}(y)$ is the mask given parts

E Step: maximize q(y)

$$\sum_y q(y) \exp\big(-\big|y-\mu(x)\big|\big) \exp\big(-\big|y_{kp}-\mu_{\mathrm{kp}}(y)\big|\big) B\big(y_{\mathrm{mask}}, \mu_{\mathrm{mask}}(y)\big).$$

Agrees w/ parts   Agrees w/ keypoints   Agrees w/ mask

Amortized Variational Inference

E Step: maximize q(y) for each x

$$\sum_y q(y) \exp\left(-\left|y-\underset{\substack{\uparrow \\ \text{Agrees w/ parts}}}{\mu(x)}\right|\right) \exp\left(-\left|y_{kp}\underset{\substack{\uparrow \\ \text{Agrees w/ keypoints}}}{-}\mu_{\mathrm{kp}}(y)\right|\right) B\left(y_{\mathrm{mask}}, \underset{\substack{\uparrow \\ \text{Agrees w/ mask}}}{\mu_{\mathrm{mask}}(y)}\right).$$

Generally intractable!

- Hard EM: Solve for argmax via SGD (each term is differentiable!)
- Langevin dynamics [SGLD, Welling & Teh'11]
- Amortized VI: approximate via $q(y|x, y_{mask}, y_{kp}) \propto q_x(y)$ (our approach)

Ours — Improving few-shot part segmentation using coarse supervision, Saha et al. arXiv'22

Results: Bird part segmentation

Training data

- 450 w/ 10 parts (CUB+PASCAL)
- 5,500 w/ keypoints & masks (CUB)

Model

- FCN w/ ResNet34 on 256x256 image
- Random or ImageNet initialization

Evaluation

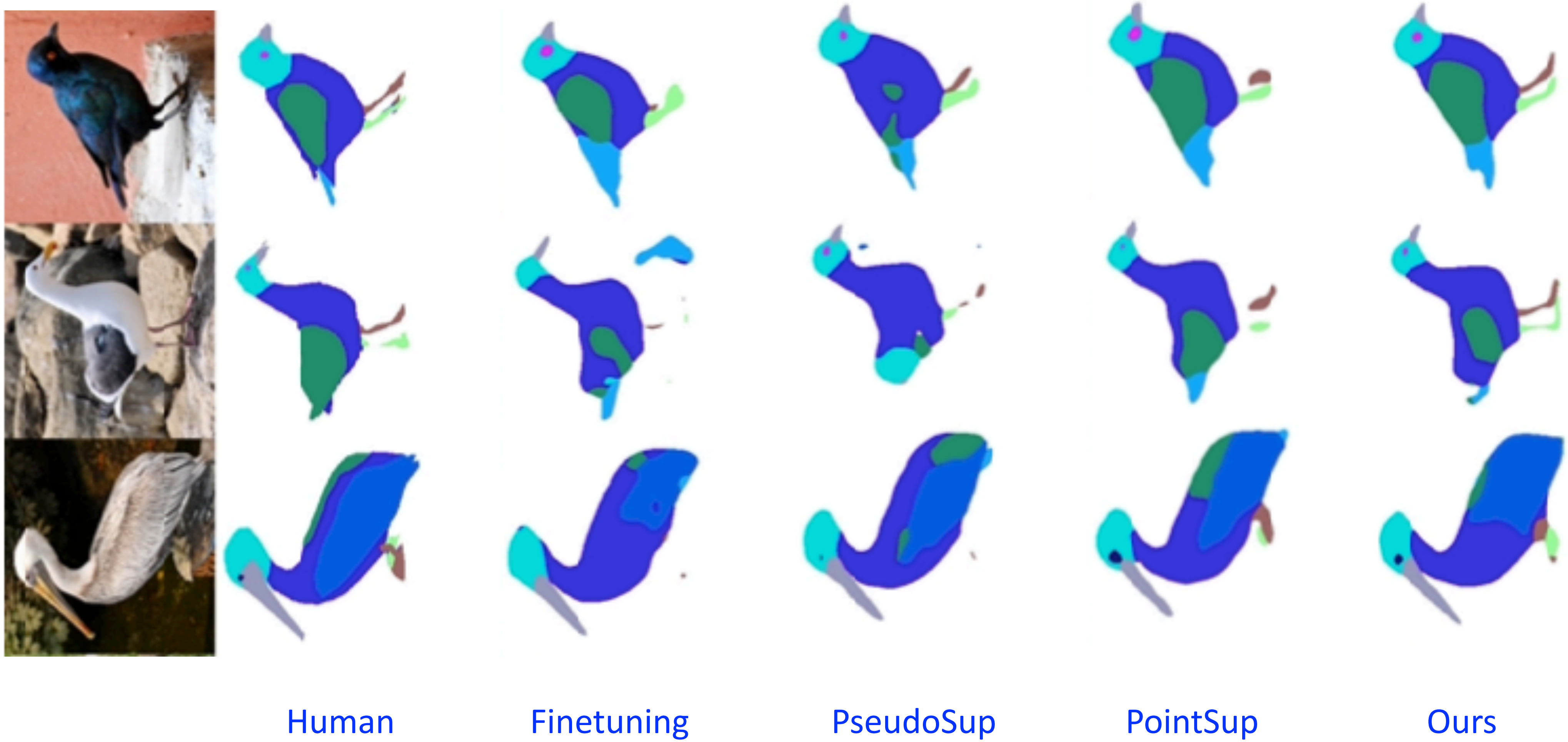
- mean IOU over 10 parts
- 150 images on CUB

metric: mean IOU over parts

	Random	ImageNet
Fine-tuning	28.9	45.4
Multi-tasking	36.9	41.3
PseudoSup [1]	30.8	46.0
PointSup [2]	35.2	46.8
Ours (EM)	**37.9**	**49.0**

[1] — PseudoSup, Chen et al., CVPR'21 (semi-supervised)
[2] — PointSup, Cheng et al., CVPR’22 (point supervision)

Results: Bird part segmentation



Summary & Conclusion

Two ways to learn with little data

- Modeling tasks and their relations — Task2Vec [ICCV’19], ECCV’20, CVPR’21
- Learning from coarse and diverse labels — classification [BMVC’21], segmentation [arXiv’22], detection [AAAI’19]

Challenges

- **Engineering:** compute, memory, energy, software infrastructure
- **Statistical:** bias-variance tradeoffs, noisy evaluation
- **Science:** how is information represented in deep networks? Are foundation models better probes?