

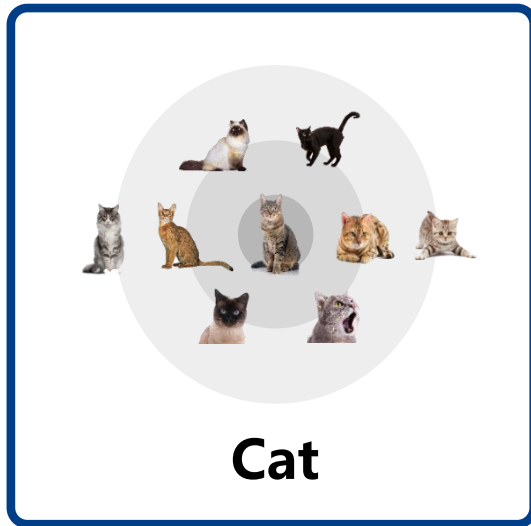
Knowledge-Driven Perception

Wenguan Wang

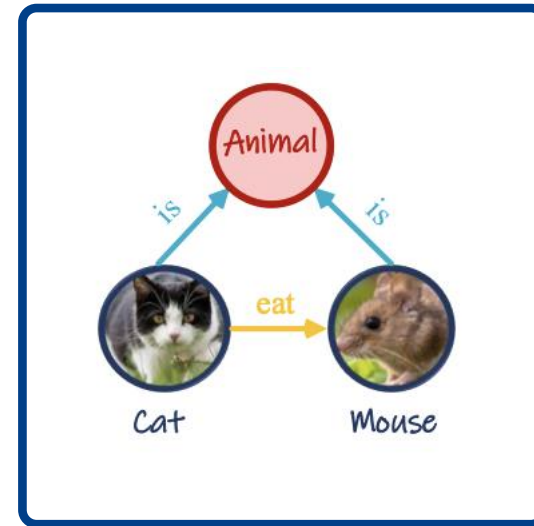
2023.04.19

<https://sites.google.com/view/wenguanwang>

The expression of knowledge



Visual Knowledge



Knowledge graph



ICLR2023 Spotlight

Visual Recognition with Deep Nearest Centroids



CVPR2022

Deep Hierarchical Semantic Segmentation

Visual Recognition with Deep Nearest Centroids

Wenguan Wang, Cheng Han, Tianfei Zhou, Dongfang Liu

Part 1

**Visual Recognition
(ICLR23 Spotlight)**

Image Classification

Dog? Pig?
... or Cat?



Image Classification

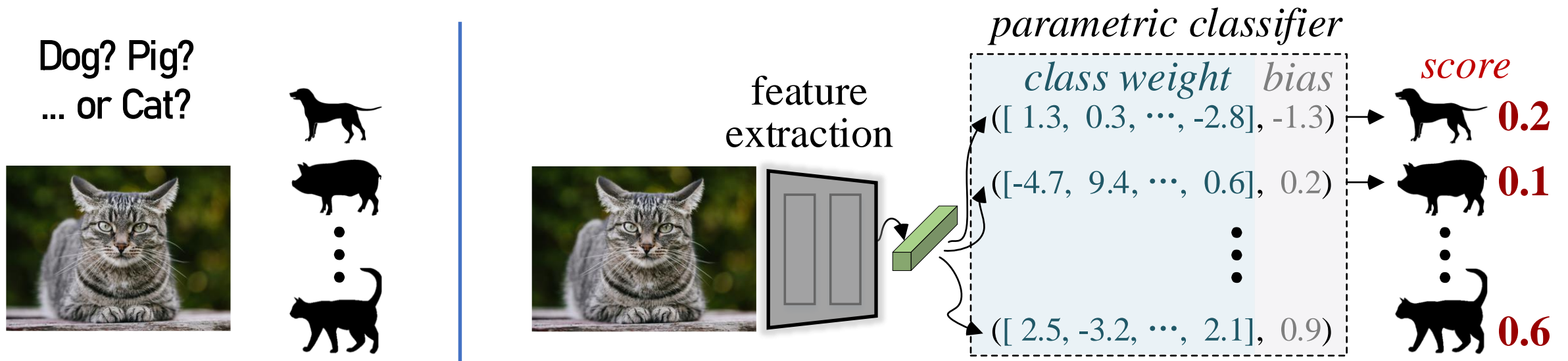


Image Classification:

Feature extractor + **Parametric softmax classifier**

Image Classification

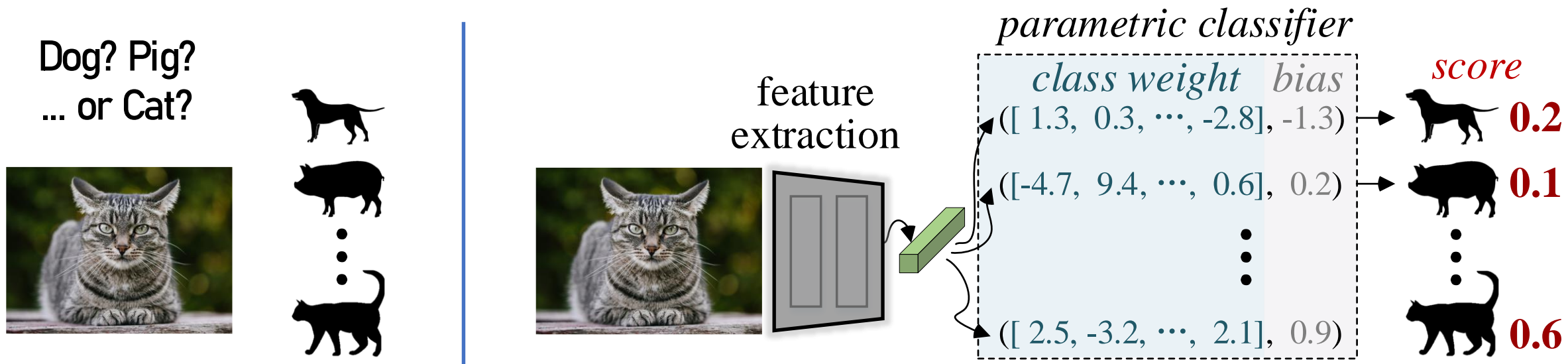


Image Classification:

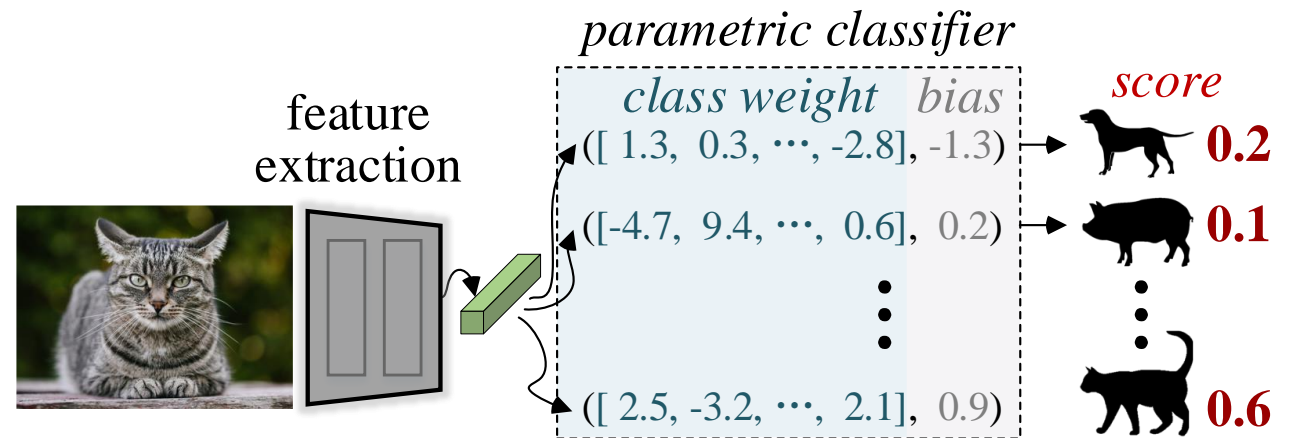
Feature extractor + **Parametric softmax classifier**

$$p(c|\mathbf{x}; \boldsymbol{\omega}, \boldsymbol{\theta}) = \frac{\exp(y_c)}{\sum_{c'} \exp(y_{c'})} = \frac{\exp(\mathbf{w}_c^\top \mathbf{x} + b_c)}{\sum_{c'} \exp(\mathbf{w}_{c'}^\top \mathbf{x} + b_{c'})} = \frac{\exp(\mathbf{w}_c^\top f_{\boldsymbol{\theta}}(\mathbf{x}) + b_c)}{\sum_{c'} \exp(\mathbf{w}_{c'}^\top f_{\boldsymbol{\theta}}(\mathbf{x}) + b_{c'})}$$

Deficiency of Parametric Softmax Classifier

Parametric softmax classifier:

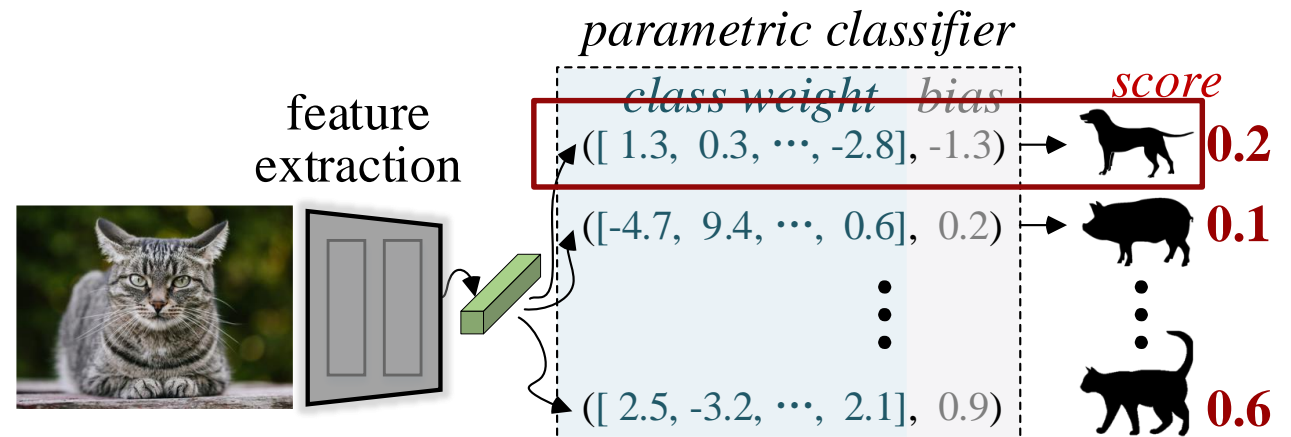
$$p(c|\mathbf{x}; \boldsymbol{\omega}, \boldsymbol{\theta}) = \frac{\exp(y_c)}{\sum_{c'} \exp(y_{c'})} = \frac{\exp(\mathbf{w}_c^\top \mathbf{x} + b_c)}{\sum_{c'} \exp(\mathbf{w}_{c'}^\top \mathbf{x} + b_{c'})}$$



Deficiency of Parametric Softmax Classifier

Parametric softmax classifier:

$$p(c|\mathbf{x}; \boldsymbol{\omega}, \boldsymbol{\theta}) = \frac{\exp(y_c)}{\sum_{c'} \exp(y_{c'})} = \frac{\exp(\mathbf{w}_c^\top \mathbf{x} + b_c)}{\sum_{c'} \exp(\mathbf{w}_{c'}^\top \mathbf{x} + b_{c'})}$$

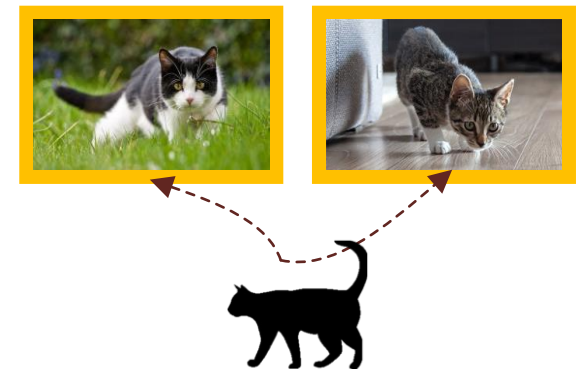


- Implicit **unimodality** assumption; Bearing **no within-class variation**.

The unimodality assumption is rarely the case in real-world scenarios.



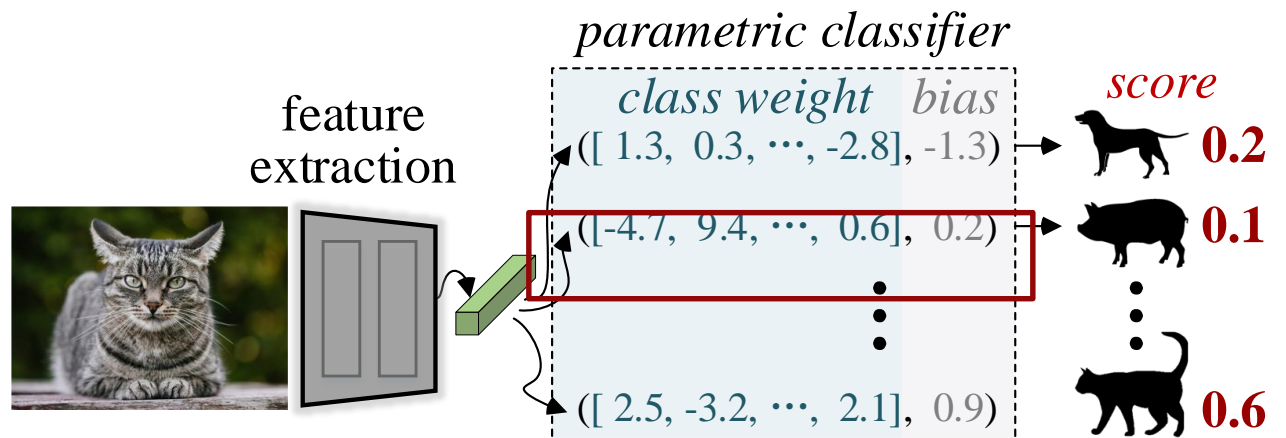
The model **less tolerant of intra-class variances**.



Deficiency of Parametric Softmax Classifier

Parametric softmax classifier:

$$p(c|\mathbf{x}; \boldsymbol{\omega}, \boldsymbol{\theta}) = \frac{\exp(y_c)}{\sum_{c'} \exp(y_{c'})} = \frac{\exp(\mathbf{w}_c^\top \mathbf{x} + b_c)}{\sum_{c'} \exp(\mathbf{w}_{c'}^\top \mathbf{x} + b_{c'})}$$



- Poor **explainability**.



Black-box gradient updated parameters

class weight	bias
([1.3, 0.3, ..., -2.8], -1.3)	
([-4.7, 9.4, ..., 0.6], 0.2)	
⋮	
([2.5, -3.2, ..., 2.1], 0.9)	



It should be Cat, because...umm...

cannot provide information about **internal decision-making**.

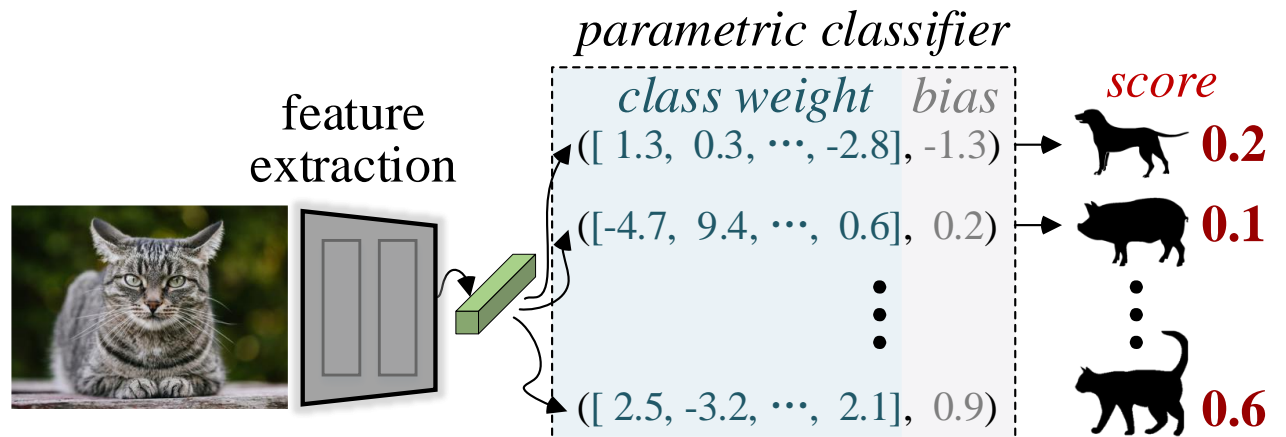


Hard to lend to a **human-readable explanation**.

Deficiency of Parametric Softmax Classifier

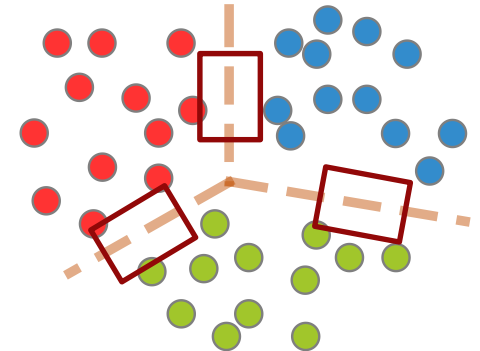
Parametric softmax classifier:

$$p(c|\mathbf{x}; \boldsymbol{\omega}, \boldsymbol{\theta}) = \frac{\exp(y_c)}{\sum_{c'} \exp(y_{c'})} = \frac{\exp(\mathbf{w}_c^\top \mathbf{x} + b_c)}{\sum_{c'} \exp(\mathbf{w}_{c'}^\top \mathbf{x} + b_{c'})}$$



- **Indirect supervision** on the representation.

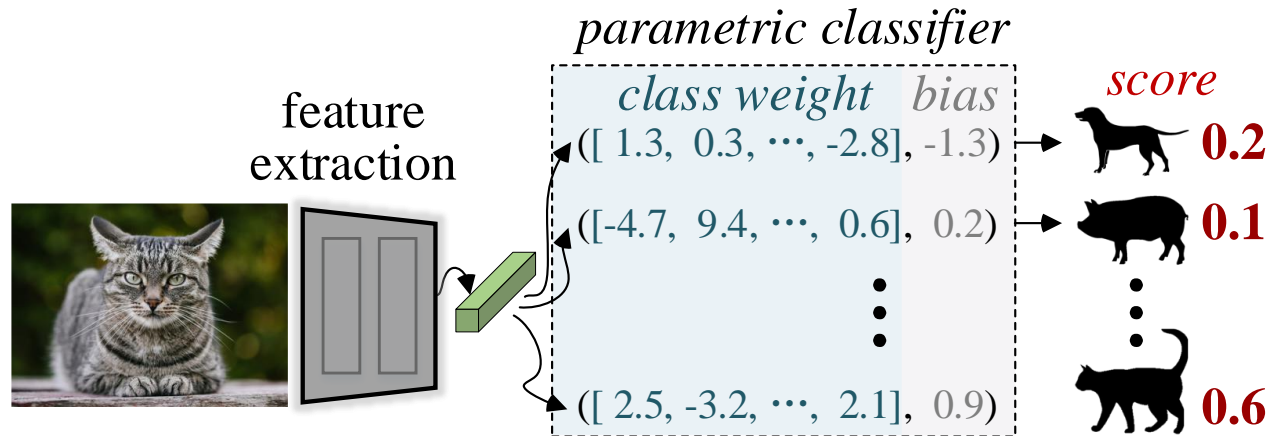
🗨️ The loss only depend on the **relative relation among logits**; cannot directly supervise on the representation.



Deficiency of Parametric Softmax Classifier

Parametric softmax classifier:

$$p(c|\mathbf{x}; \boldsymbol{\omega}, \boldsymbol{\theta}) = \frac{\exp(y_c)}{\sum_{c'} \exp(y_{c'})} = \frac{\exp(\mathbf{w}_c^\top \mathbf{x} + b_c)}{\sum_{c'} \exp(\mathbf{w}_{c'}^\top \mathbf{x} + b_{c'})}$$

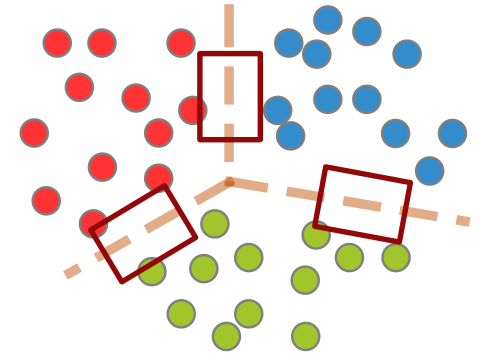


- **Indirect supervision** on the representation.

🗨️ The loss only depend on the **relative relation among logits**; cannot directly supervise on the representation.

- **Incontiguous knowledge transfer** to downstream applications.

🗨️ The classifier from a pretrained model has to be abandoned, even though the learnt parameters contain valuable knowledge from the source task.



Rethinking de facto Paradigm

- Implicit **unimodality** assumption; Bearing **no within-class variation**.
- Poor **explainability**.
- **Indirect supervision** on the representation.
- **Incontiguous knowledge transfer** to downstream applications.

Rethinking de facto Paradigm

- Implicit **unimodality** assumption; Bearing **no within-class variation**.
- Poor **explainability**.
- **Indirect supervision** on the representation.
- **Incontiguous knowledge transfer** to downstream applications.

Is there any way to address the **limitations** of *de facto* visual recognition regime?

Rethinking de facto Paradigm

- Implicit **unimodality** assumption; Bearing **no within-class variation**.
- Poor **explainability**.
- **Indirect supervision** on the representation.
- **Incontiguous knowledge transfer** to downstream applications.

Is there any way to address the **limitations** of *de facto* visual recognition regime?

Our answer:

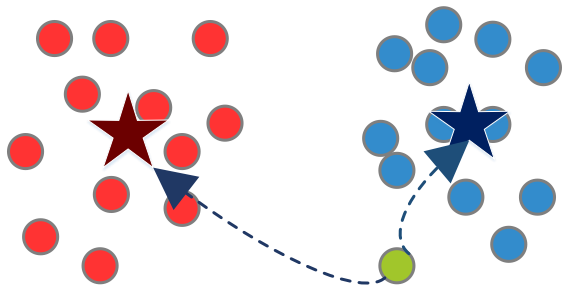


Deep Nearest Centroids (DNC)

Deep Nearest Centroids

- Nearest Centroids¹

Core principle: Given a data sample, it is directly classified to the class of training examples whose **mean (centroid) is closest to it.**



[1] Discriminatory analysis - nonparametric discrimination: Small sample performance. Tech. rep., UCB, 1952;

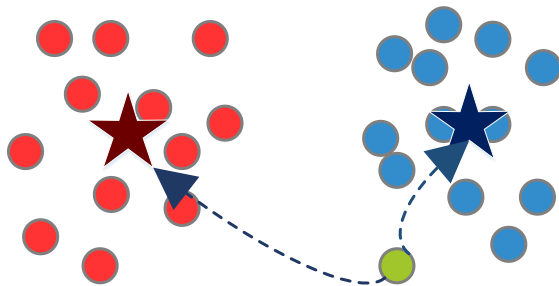
[2] An introduction to case-based reasoning. Artificial Intelligence Review, 1992.

Deep Nearest Centroids

- Nearest Centroids¹

Core principle: Given a data sample, it is directly classified to the class of training examples whose **mean (centroid) is closest to it**.

👍 One of the simplest classifier; **Internal transparency (ad-hoc interpretability)**.



[1] Discriminatory analysis - nonparametric discrimination: Small sample performance. Tech. rep., UCB, 1952;

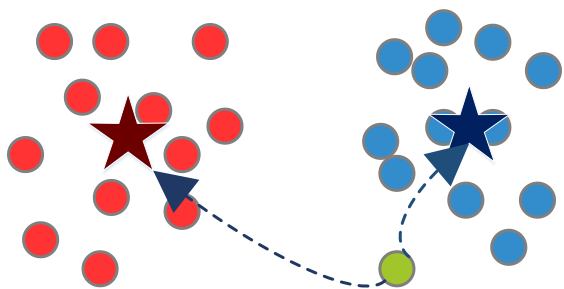
[2] An introduction to case-based reasoning. Artificial Intelligence Review, 1992.

Deep Nearest Centroids

- Nearest Centroids¹

Core principle: Given a data sample, it is directly classified to the class of training examples whose **mean (centroid) is closest to it**.

- 👍 One of the simplest classifier; **Internal transparency (ad-hoc interpretability)**.
- 👍 A classical form of exemplar-based reasoning, which is fundamental to our most effective strategies for **tactical decision-making**².



Human



Cat...it looks like



I've ever seen,

but not like



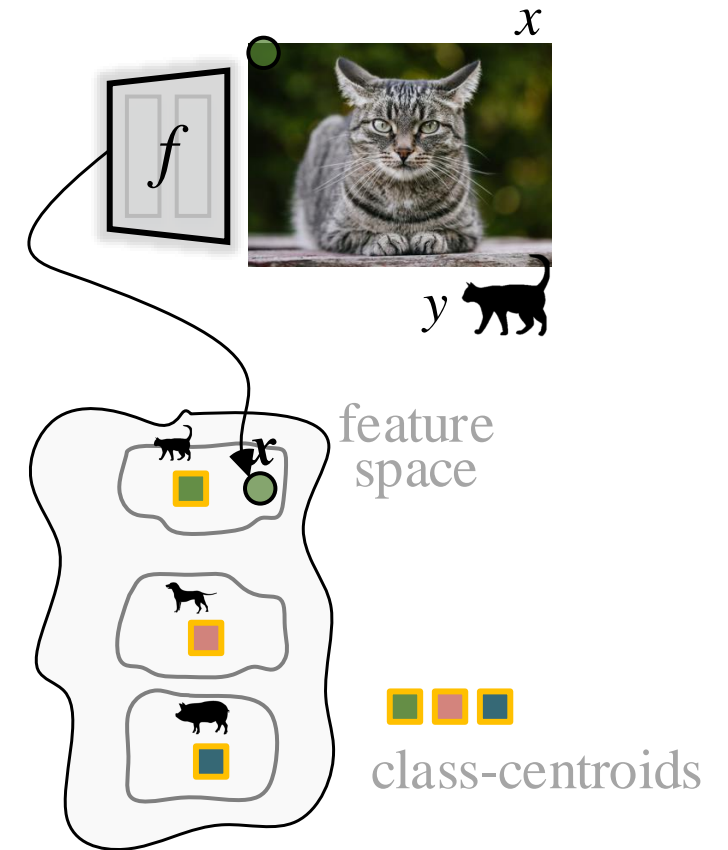
...

[1] Discriminatory analysis - nonparametric discrimination: Small sample performance. Tech. rep., UCB, 1952;

[2] An introduction to case-based reasoning. Artificial Intelligence Review, 1992.

Deep Nearest Centroids: Inference

- Feature extractor $f : \mathcal{X} \mapsto \mathcal{F}$; $\mathbf{x} = f(x) \in \mathcal{F}$

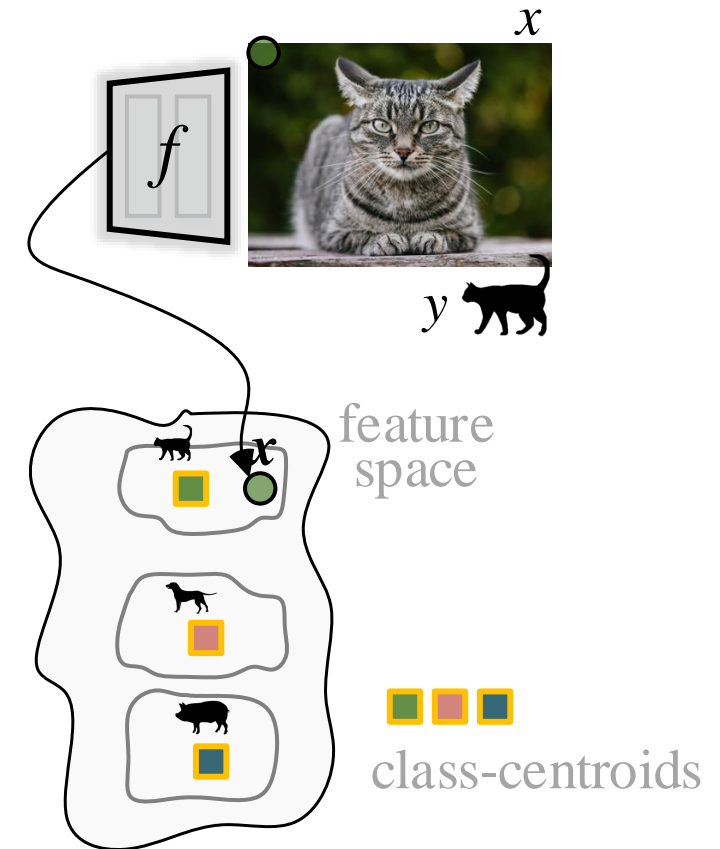


Deep Nearest Centroids: Inference

- Feature extractor $f : \mathcal{X} \mapsto \mathcal{F}$; $\mathbf{x} = f(x) \in \mathcal{F}$
- Basic **Nearest Centroids Classification**:

$$\hat{y} = \arg \min_{c \in \mathcal{Y}} \langle \mathbf{x}, \bar{\mathbf{x}}^c \rangle$$

where $\bar{\mathbf{x}}^c = \frac{1}{N^c} \sum_{x_n^c: y_n^c=c} \mathbf{x}_n^c$ is the mean vector of class c .

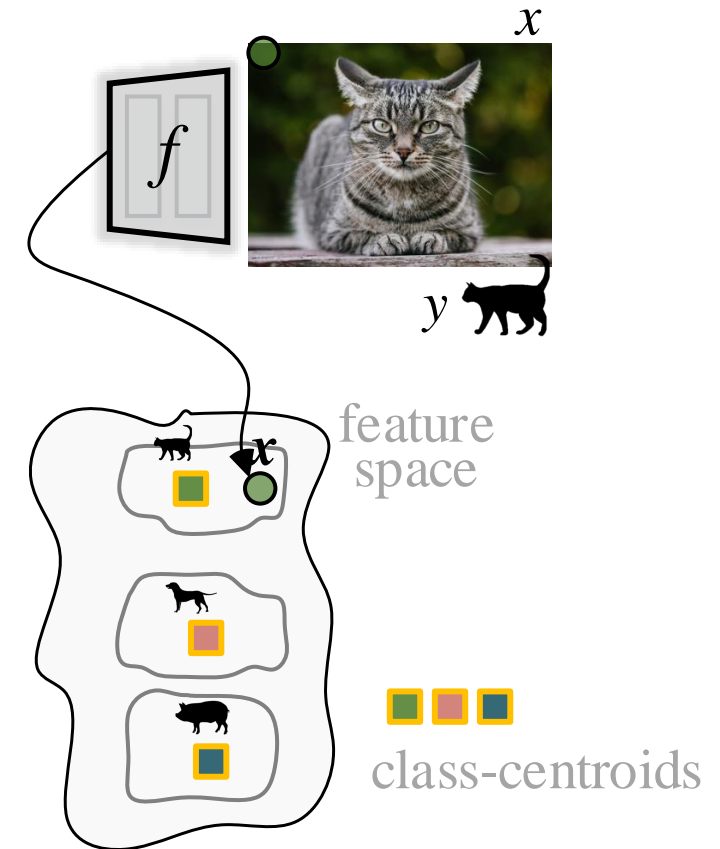
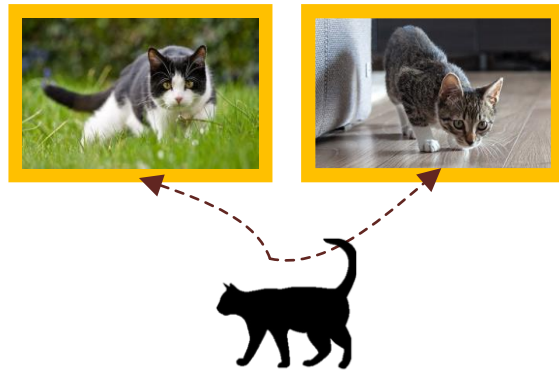


Deep Nearest Centroids: Inference

- Feature extractor $f : \mathcal{X} \mapsto \mathcal{F}$; $\mathbf{x} = f(x) \in \mathcal{F}$
- Basic **Nearest Centroids Classification**:

$$\hat{y} = \arg \min_{c \in \mathcal{Y}} \langle \mathbf{x}, \bar{\mathbf{x}}^c \rangle$$

where $\bar{\mathbf{x}}^c = \frac{1}{N^c} \sum_{x_n^c: y_n^c = c} \mathbf{x}_n^c$ is the mean vector of class c .

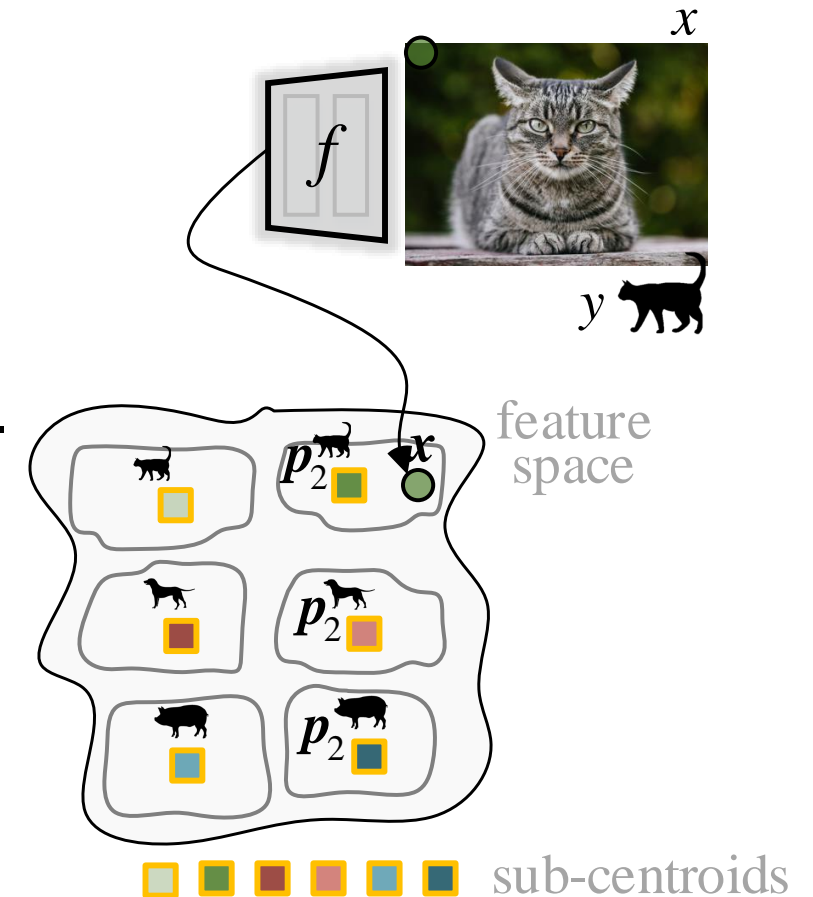


Deep Nearest Centroids: Inference

- Feature extractor $f : \mathcal{X} \mapsto \mathcal{F}$; $\mathbf{x} = f(x) \in \mathcal{F}$
- Basic **Nearest Centroids Classification**:

$$\hat{y} = \arg \min_{c \in \mathcal{Y}} \langle \mathbf{x}, \bar{\mathbf{x}}^c \rangle$$

where $\bar{\mathbf{x}}^c = \frac{1}{N^c} \sum_{x_n^c: y_n^c=c} \mathbf{x}_n^c$ is the mean vector of class c .



Deep Nearest Centroids: Inference

- Feature extractor $f : \mathcal{X} \mapsto \mathcal{F}$; $\mathbf{x} = f(x) \in \mathcal{F}$
- Basic **Nearest Centroids Classification**:

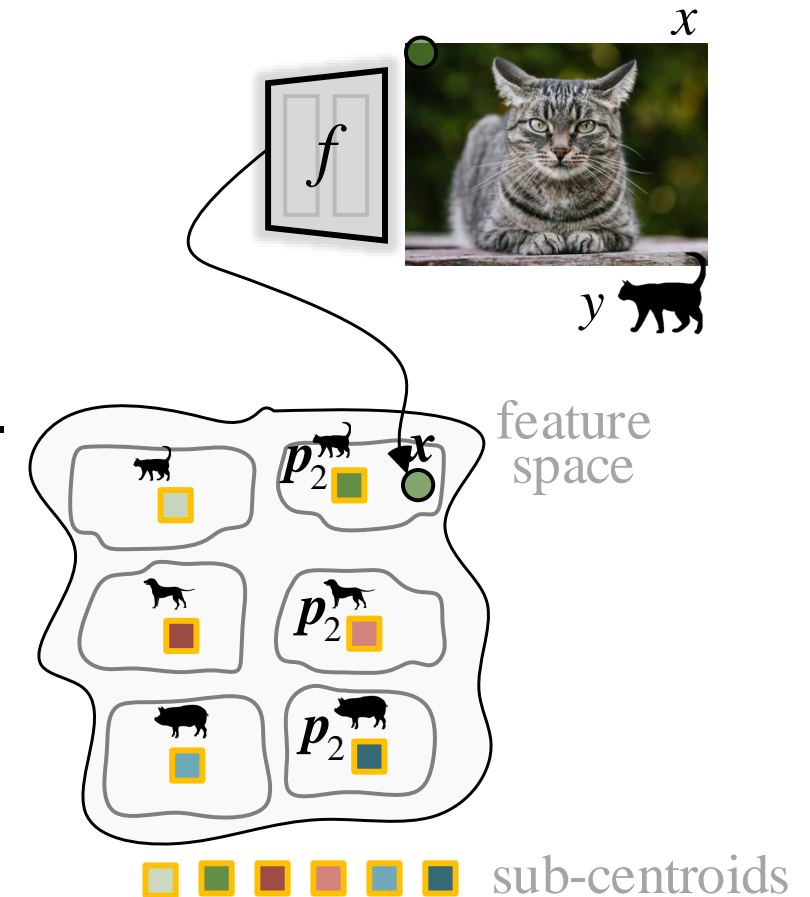
$$\hat{y} = \arg \min_{c \in \mathcal{Y}} \langle \mathbf{x}, \bar{\mathbf{x}}^c \rangle$$

where $\bar{\mathbf{x}}^c = \frac{1}{N^c} \sum_{x_n^c: y_n^c=c} \mathbf{x}_n^c$ is the mean vector of class c .

- *Winner-takes-all*, **multiple sub-centroids classification**:

$$\hat{y} = c^*, \quad (c^*, k^*) = \arg \min_{c \in \mathcal{Y}, k \in \{1, \dots, K\}} \langle \mathbf{x}, \mathbf{p}_k^c \rangle.$$

Here $\{\mathbf{p}_k^c \in \mathbb{R}^d\}_{k=1}^K$ represents K sub-centroids in each class c .



Deep Nearest Centroids: Sub-pattern Discovery



- Informative sub-centroids discovery; **Binary integer program:**

$$\max_{Q^c \in \mathcal{Q}^c} \text{Tr}((Q^c)^\top (P^c)^\top X^c), \quad \mathcal{Q}^c = \{Q^c \in \{0, 1\}^{K \times N^c} \mid (Q^c)^\top \mathbf{1}_K = \mathbf{1}_{N^c}\}$$

$X^c = [\mathbf{x}_1^c, \dots, \mathbf{x}_{N^c}^c]$: data representations of c -th class

$P^c = [p_1^c, \dots, p_K^c]$: sub-centroids of c -th class

$Q^c = [q_1^c, \dots, q_{N^c}^c]$: data-centroid assignments

Deep Nearest Centroids: Sub-pattern Discovery



- Informative sub-centroids discovery; Casting into **Optimal Transport** problem:

$$\max_{Q^c \in \mathcal{Q}^c} \text{Tr}((Q^c)^\top (P^c)^\top X^c), \quad \mathcal{Q}^c = \{Q^c \in \mathbb{R}_+^{K \times N^c} \mid (Q^c)^\top \mathbf{1}_K = \mathbf{1}_{N^c}, Q^c \mathbf{1}_{N^c} = \frac{N^c}{K} \mathbf{1}_K\}$$

$X^c = [\mathbf{x}_1^c, \dots, \mathbf{x}_{N^c}^c]$: data representations of c -th class

$P^c = [p_1^c, \dots, p_K^c]$: sub-centroids of c -th class

$Q^c = [q_1^c, \dots, q_{N^c}^c]$: data-centroid assignments

Deep Nearest Centroids: Sub-pattern Discovery



- Informative sub-centroids discovery; Casting into **Optimal Transport** problem:

$$\max_{Q^c \in \mathcal{Q}^c} \text{Tr}((Q^c)^\top (P^c)^\top X^c), \quad \mathcal{Q}^c = \{Q^c \in \mathbb{R}_+^{K \times N^c} \mid (Q^c)^\top \mathbf{1}_K = \mathbf{1}_{N^c}, Q^c \mathbf{1}_{N^c} = \frac{N^c}{K} \mathbf{1}_K\}$$

Equipartition

Constraint

$X^c = [\mathbf{x}_1^c, \dots, \mathbf{x}_{N^c}^c]$: data representations of c -th class

$P^c = [p_1^c, \dots, p_K^c]$: sub-centroids of c -th class

$Q^c = [q_1^c, \dots, q_{N^c}^c]$: data-centroid assignments

Deep Nearest Centroids: Sub-pattern Discovery



- Informative sub-centroids discovery:

$$\max_{Q^c \in \mathcal{Q}^c} \text{Tr}((Q^c)^\top (P^c)^\top X^c), \quad \mathcal{Q}^c = \{Q^c \in \mathbb{R}_+^{K \times N^c} \mid (Q^c)^\top \mathbf{1}_K = \mathbf{1}_{N^c}, Q^c \mathbf{1}_{N^c} = \frac{N^c}{K} \mathbf{1}_K\}$$

$X^c = [x_1^c, \dots, x_{N^c}^c]$: data representations of c -th class

$P^c = [p_1^c, \dots, p_K^c]$: sub-centroids of c -th class

$Q^c = [q_1^c, \dots, q_{N^c}^c]$: data-centroid assignments

Solved by **Sinkhorn-Knopp** algorithm¹:

$$Q^{c*} = \text{diag}(\alpha) \exp\left(\frac{(P^c)^\top X^c}{\varepsilon}\right) \text{diag}(\beta)$$

[1] Sinkhorn distances: Lightspeed computation of optimal transport. NeurIPS, 2013.

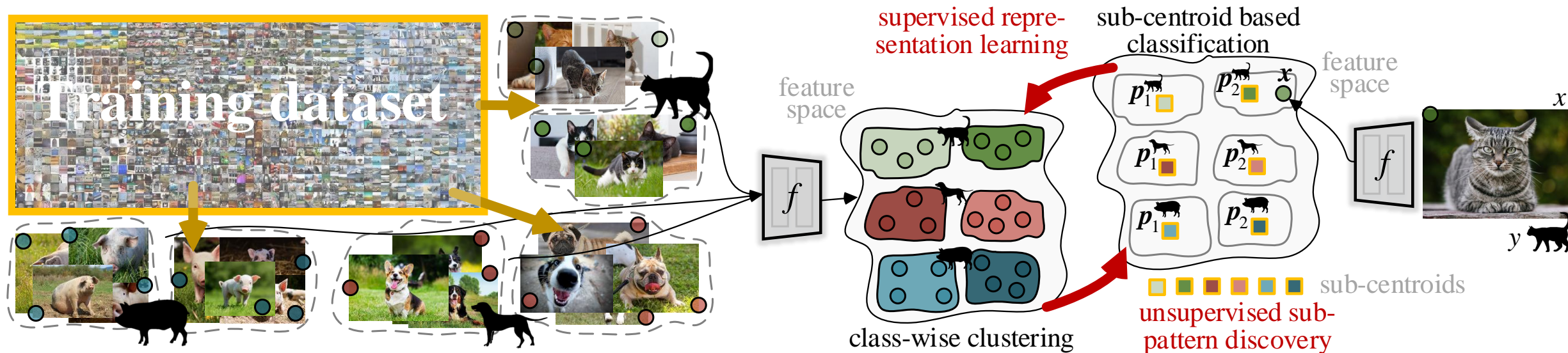
Deep Nearest Centroids: Overall Framework



- Training of DNC

Unsupervised Sub-class Pattern Mining.

Deep Nearest Centroids: Overall Framework

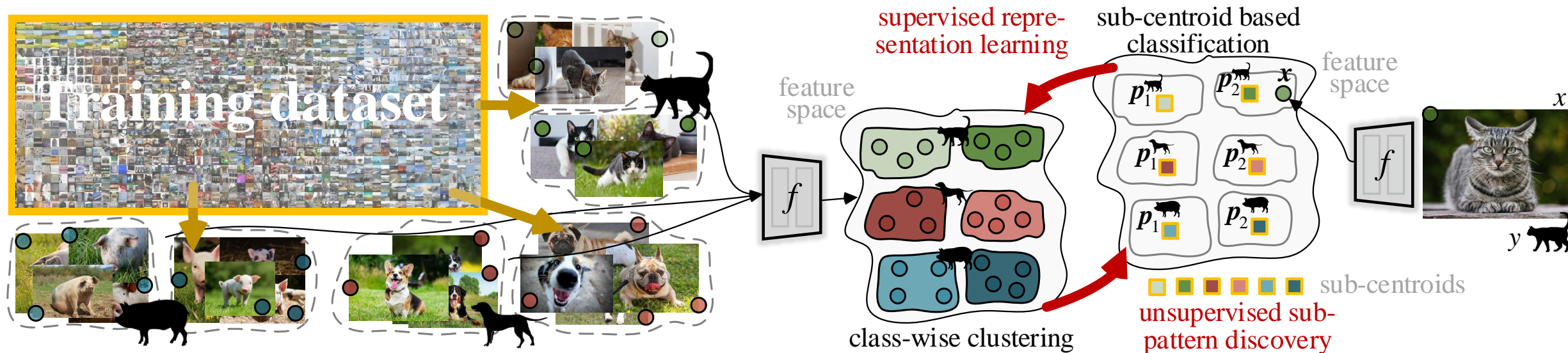


- Training of DNC

= **Supervised Representation Learning** + Unsupervised Sub-class Pattern Mining.

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^N -\log p(y_n | x_n), \quad p(y|x) = \frac{\exp(-\min(\{\langle \mathbf{x}, \mathbf{p}_k^y \rangle\}_{k=1}^K))}{\sum_{c \in \mathcal{Y}} \exp(-\min(\{\langle \mathbf{x}, \mathbf{p}_k^c \rangle\}_{k=1}^K))}.$$

Deep Nearest Centroids: Overall Framework



- Training of DNC
 - = **Supervised Representation Learning** + **Unsupervised Sub-class Pattern Mining**.
- DNC alternates **two steps iteratively**:
 - I. class-wise clustering for automatic sub-centroid discovery
 - II. sub-centroid based classification for supervised representation learning

Deep Nearest Centroids: Overall Framework

- **Problem:** Contiguously evolved latent space; Class sub-centroids should be synchronized.

Adopt **momentum update** and **online clustering** to avoid expensive offline sub-centroid estimation after each batch update.

$$\mathbf{p}_k^c \leftarrow \mu \mathbf{p}_k^c + (1 - \mu) \bar{\mathbf{x}}_k^c$$

Deep Nearest Centroids: Overall Framework

- **Problem:** **Contiguously evolved latent space**; Class sub-centroids should be synchronized.

Adopt **momentum update** and **online clustering** to avoid expensive offline sub-centroid estimation after each batch update.

$$\mathbf{p}_k^c \leftarrow \mu \mathbf{p}_k^c + (1 - \mu) \bar{\mathbf{x}}_k^c$$

-
- **Problem:** **Batch-wise clustering** could not extend to **a large number of classes**.

Clustering on both batch and stored features from several prior batches in a **memory**.

Deep Nearest Centroids: Overall Framework

- **Problem:** **Contiguously evolved latent space**; Class sub-centroids should be synchronized.

Adopt **momentum update** and **online clustering** to avoid expensive offline sub-centroid estimation after each batch update.

$$\mathbf{p}_k^c \leftarrow \mu \mathbf{p}_k^c + (1 - \mu) \bar{\mathbf{x}}_k^c$$

-
- **Problem:** **Batch-wise clustering** could not extend to **a large number of classes**.

Clustering on both batch and stored features from several prior batches in a **memory**.

Efficiently trained in **small-batch setting**,
with **negligible lagging** (**~5%** training delay on ImageNet).

Summary: DNC

- **Versatility:**

 DNC can be integrated into any parametric classifier based DNNs **effortlessly**.

Experiments: Image Classification

CIFAR-10

Method	Backbone	#Params	top-1
ResNet [2]	ResNet50	23.52M	95.55%
DNC-ResNet		23.50M	95.78%
ResNet [2]	ResNet101	42.51M	95.58%
DNC-ResNet		42.49M	95.82%

CIFAR-100

Method	Backbone	#Params	top-1
ResNet [2]	ResNet50	23.71M	79.81%
DNC-ResNet		23.50M	79.91%
ResNet [2]	ResNet101	42.70M	79.83%
DNC-ResNet		42.49M	79.99%

ImageNet

Method	Backbone	#Params	top-1	top-5
ResNet [2]	ResNet50	25.56M	76.20%	93.01%
DNC-ResNet		23.51M	76.49%	93.08%
ResNet [2]	ResNet101	44.55M	77.52%	93.06%
DNC-ResNet		42.50M	77.80%	93.85%
Swin [3]	Swin-S	49.61M	83.02%	96.29%
DNC-Swin		48.84M	83.26%	96.40%
Swin [3]	Swin-B	87.77M	83.36%	96.44%
DNC-Swin		86.75M	83.68%	97.02%

CUB-200-2011

Model (ImageNet-trained)	Backbone	top-1	top-5
ResNet [2]	ResNet50	84.48%	96.31%
DNC-ResNet		85.21%	96.70%

Experiments: Image Classification

ImageNetV2


test set	Method	Backbone	top-1	top-5
MatchedFrequency	ResNet [2]	ResNet50	63.30%	84.70%
	DNC-ResNet		63.96%	85.17%
Threshold0.7	ResNet [2]	ResNet50	72.70%	92.00%
	DNC-ResNet		73.59%	92.26%
TopImages	ResNet [2]	ResNet50	78.10%	94.70%
	DNC-ResNet		78.62%	94.96%

Summary: DNC

- **Versatility:**

 DNC can be integrated into any parametric classifier based DNNs effortlessly.

- **Transferability:**

 DNC can store all the **knowledge learnt on a source task** in the backbone feature extractor, which can be completely **transferred for a new task**.

Experiments: Image Segmentation

ADE20K and Cityscape

Method	Backbone	ImageNet top-1 acc.	#Params	ADE20K mIoU	Cityscapes mIoU
FCN [34]	ResNet101 [2]	77.52%	68.6M	39.9 ± (0.11)%	75.6 ± (0.13)%
	DNC-ResNet101	77.80%	68.6M	40.4 ± (0.11)%	76.3 ± (0.12)%
DNC-FCN	ResNet101 [2]	77.52%	68.5M	41.1 ± (0.10)%	76.7 ± (0.11)%
	DNC-ResNet101	77.80%	68.5M	42.3 ± (0.10)%	77.5 ± (0.11)%
DeepLab _{v3} [35]	ResNet101 [2]	77.52%	62.7M	44.1 ± (0.13)%	78.1 ± (0.12)%
	DNC-ResNet101	77.80%	62.7M	44.6 ± (0.13)%	78.7 ± (0.13)%
DNC-DeepLab_{v3}	ResNet101 [2]	77.52%	62.6M	45.0 ± (0.10)%	79.1 ± (0.13)%
	DNC-ResNet101	77.80%	62.6M	45.7 ± (0.09)%	79.8 ± (0.12)%
UperNet [36]	Swin-B [3]	83.36%	90.6M	48.0 ± (0.11)%	79.8 ± (0.13)%
	DNC-Swin-B	83.68%	90.6M	48.4 ± (0.10) %	80.1 ± (0.12)%
DNC-UperNet	Swin-B [3]	83.36%	90.5M	48.6 ± (0.09) %	80.5 ± (0.10)%
	DNC-Swin-B	83.68%	90.5M	50.5 ± (0.09)%	80.9 ± (0.10)%

Experiments: Image Segmentation

COCO-Stuff


Method	Backbone	COCO-Stuff test mIoU
FCN [34]	ResNet101 [2]	32.63%
	DNC-ResNet101	32.89% \uparrow 0.26
DNC-FCN	ResNet101 [2]	33.04% \uparrow 0.41
	DNC-ResNet101	33.49% \uparrow 0.86
DeepLab _{v3} [35]	ResNet101 [2]	36.01%
	DNC-ResNet101	36.28% \uparrow 0.27
DNC-DeepLab_{v3}	ResNet101 [2]	36.51% \uparrow 0.50
	DNC-ResNet101	36.79% \uparrow 0.78
UperNet [36]	Swin-B [3]	42.77%
	DNC-Swin-B	42.84% \uparrow 0.07
DNC-UperNet	Swin-B [3]	43.13% \uparrow 0.36
	DNC-Swin-B	43.29% \uparrow 0.52

Summary: DNC

- **Versatility:**

 DNC can be integrated into any parametric classifier based DNNs effortlessly.

- **Transferability:**

 DNC can store all the knowledge learnt on a source task in the backbone feature extractor, which can be completely transferred for a new task.

- **Ad-hoc Explainability:**

 DNC is a **transparent classifier** that has a built-in **case-based reasoning process**

 Sub-centroids can be summarized from or anchoring **to real observations**.

Experiments: Ad-hoc Explainability

- **Ad-hoc Explainability:**



DNC is a **transparent classifier** that has a built-in **case-based reasoning process**



Sub-centroids can be summarized from or anchoring **to real observations.**

- **Discovered sub-centroid images:**

goose



castle

*desk
computer*



convertible



Experiments: Ad-hoc Explainability

- **Explainable Inner Decision-Making Mode based on IF . . . -Then Rules¹:**

Experiments: Ad-hoc Explainability

- **Explainable Inner Decision-Making Mode based on IF . . . Then Rules¹:**

For query image I

$$\begin{aligned} &IF ([I, \hat{I}_1] > [I, \check{I}_1] \text{ AND } \dots \text{ AND } [I, \hat{I}_1] > [I, \check{I}_T]) \\ &OR ([I, \hat{I}_2] > [I, \check{I}_1] \text{ AND } \dots \text{ AND } [I, \hat{I}_2] > [I, \check{I}_T]) \\ &OR \dots OR ([I, \hat{I}_K] > [I, \check{I}_1] \text{ AND } \dots \text{ AND } [I, \hat{I}_K] > [I, \check{I}_T]) \text{ THEN (class } c). \end{aligned}$$

$\hat{I}_{1:K}$: sub-centroid images for class c ; $\check{I}_{1:T}$: representative images for all the other classes.

IF ($[I, \img alt="goose" data-bbox="128 653 181 708"/>] > [I, \img alt="night building" data-bbox="244 653 297 708"/>] \text{ AND } \dots \text{ AND } [I, \img alt="goose" data-bbox="481 653 534 708"/>] > [I, \img alt="car" data-bbox="598 653 651 708"/>]$)
OR ($[I, \img alt="clouds" data-bbox="166 714 219 769"/>] > [I, \img alt="night building" data-bbox="283 714 336 769"/>] \text{ AND } \dots \text{ AND } [I, \img alt="clouds" data-bbox="519 714 572 769"/>] > [I, \img alt="car" data-bbox="636 714 689 769"/>]$)
OR ($[I, \img alt="goose head" data-bbox="198 775 251 830"/>] > [I, \img alt="night building" data-bbox="311 775 364 830"/>] \text{ AND } \dots \text{ AND } [I, \img alt="goose head" data-bbox="551 775 604 830"/>] > [I, \img alt="car" data-bbox="668 775 721 830"/>]$)
OR ($[I, \img alt="goose in pond" data-bbox="234 836 287 891"/>] > [I, \img alt="night building" data-bbox="351 836 404 891"/>] \text{ AND } \dots \text{ AND } [I, \img alt="goose in pond" data-bbox="589 836 642 891"/>] > [I, \img alt="car" data-bbox="706 836 759 891"/>]$) *THEN (goose)*

Experiments: Ad-hoc Explainability

Using **cluster center** vs **resembling real observation**
as class sub-centroids

Sub-centroid	Architecture	top-1	top-5
<i>cluster center</i>	DNC-ResNet50	76.49%	93.08%
<i>real observation</i>		76.37%	93.04%
-	ResNet50[2]	76.20%	93.01%

Experiments: Sub-category Discovery

- **Sub-category Discovery** on CIFAR100
- CIFAR100:
 - 20 coarse-grained classes, e.g., food containers
 - 100 fine-grained classes, e.g., bottles, bowls, cans, cups, plates

Method	Backbone	20 classes	100 classes
ResNet [2]	ResNet50	86.21%	53.22%
DNC-ResNet		86.33%	67.46%
ResNet [2]	ResNet101	86.48%	54.31%
DNC-ResNet		86.60%	68.13%

Experiments: Sub-category Discovery

- **Sub-category Discovery** on CIFAR100
- CIFAR100:
 - 20 coarse-grained classes, e.g., food containers
 - 100 fine-grained classes, e.g., bottles, bowls, cans, cups, plates

Method	Backbone	<i>k</i> -NN	
		20 classes	100 classes
ResNet [2]	ResNet50	86.21%	53.22%
DNC-ResNet		86.33%	67.46%
ResNet [2]	ResNet101	86.48%	54.31%
DNC-ResNet		86.60%	68.13%

DNC

Experiments: Sub-category Discovery

- **Sub-category Discovery** on ImageNet
- ImageNet :
 - 127 coarse-grained classes, e.g., food containers
 - 1000 fine-grained classes, e.g., dog, bowls, cans, cups, plates

Method	Backbone	127 classes	1000 classes
ResNet [2]	ResNet50	84.29%	43.23%
DNC-ResNet		84.39%	52.21%
ResNet [2]	ResNet101	85.88%	45.31%
DNC-ResNet		85.91%	54.60%

Experiments: Training Time

- **Impact of clustering algorithm** on CIFAR-10 and CIFAR-100

Dataset	Method	Backbone	top-1	Training time (hours)
CIFAR10 [33]	DNC-k-means	ResNet50	93.88%	4.5
	DNC-Sinkhorn		95.78%	2.5
CIFAR100 [33]	DNC-k-means	ResNet50	77.86%	16.3
	DNC-Sinkhorn		79.91%	3.7

Experiments: ablative study

Output dimensionality

Output dimensionality	ImageNet	
	top-1	top-5
640	76.23%	92.83%
1024	76.28%	92.90%
1280	76.61%	93.12%
2048	76.49%	93.08%

Temperature parameter

ϵ	ImageNet	
	top-1	top-5
0.01	76.34%	92.97%
0.05	76.49%	93.08%
0.1	76.40%	93.02%

Experiments: ablative study

Statistics of **GPU memory cost** with respect to the **number of class centroids** and **external memory size**, where 1000 batches = 256000 image samples.

K (Fixed memory size: 1000 batches)	GPU memory cost (GB per GPU)
1	16.07
2	19.04
3	22.03
4	26.31

memory size (Fixed $K = 4$)	GPU memory cost (GB per GPU)
400 batches	11.88
600 batches	16.35
800 batches	21.65
1000 batches	26.35

Deep Hierarchical Semantic Segmentation

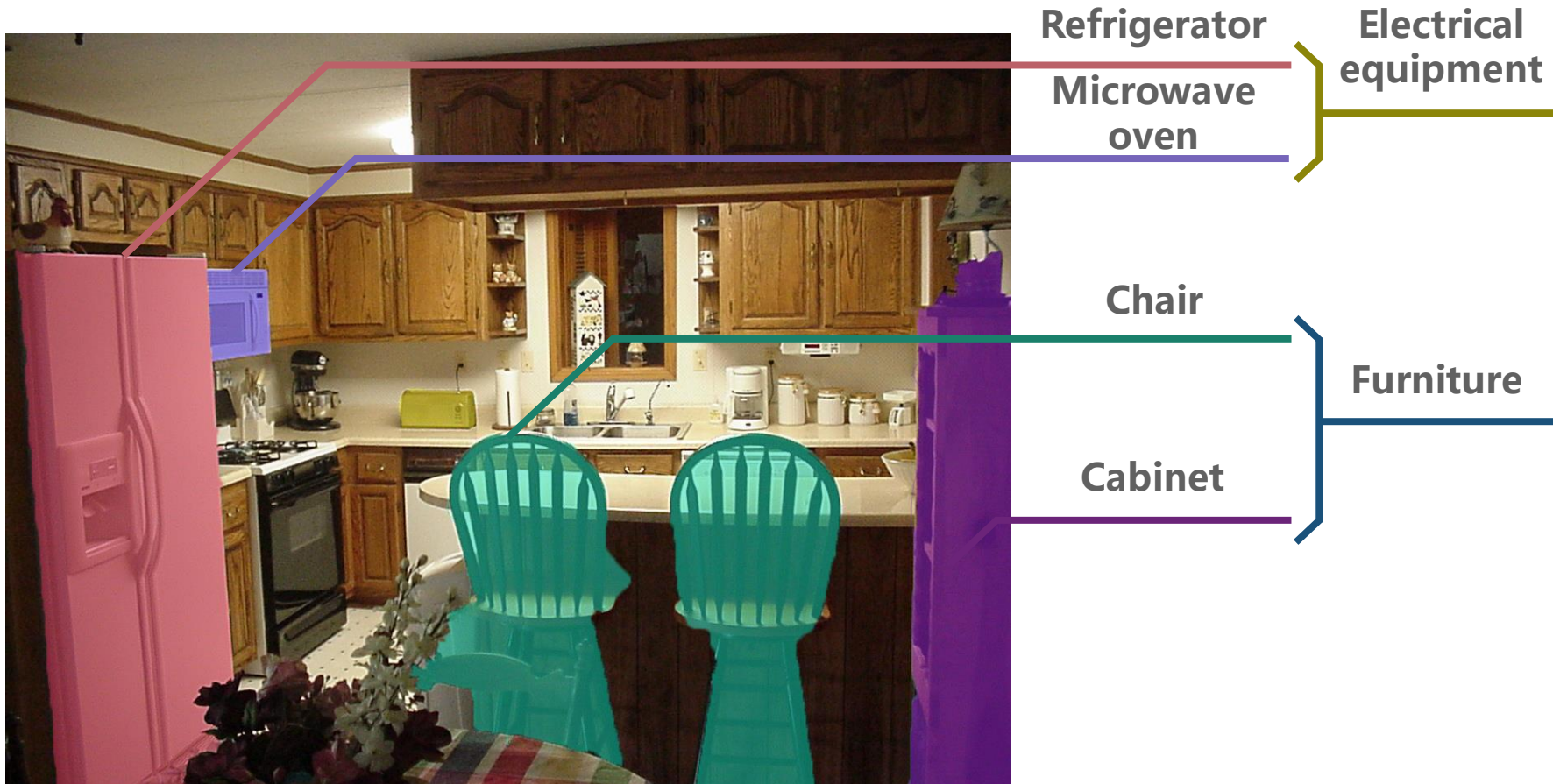
Liulei Li, Tianfei Zhou, Wenguan Wang, Jianwu Li, Yi Yang

Part2

**Semantic Segmentation
(CVPR22)**

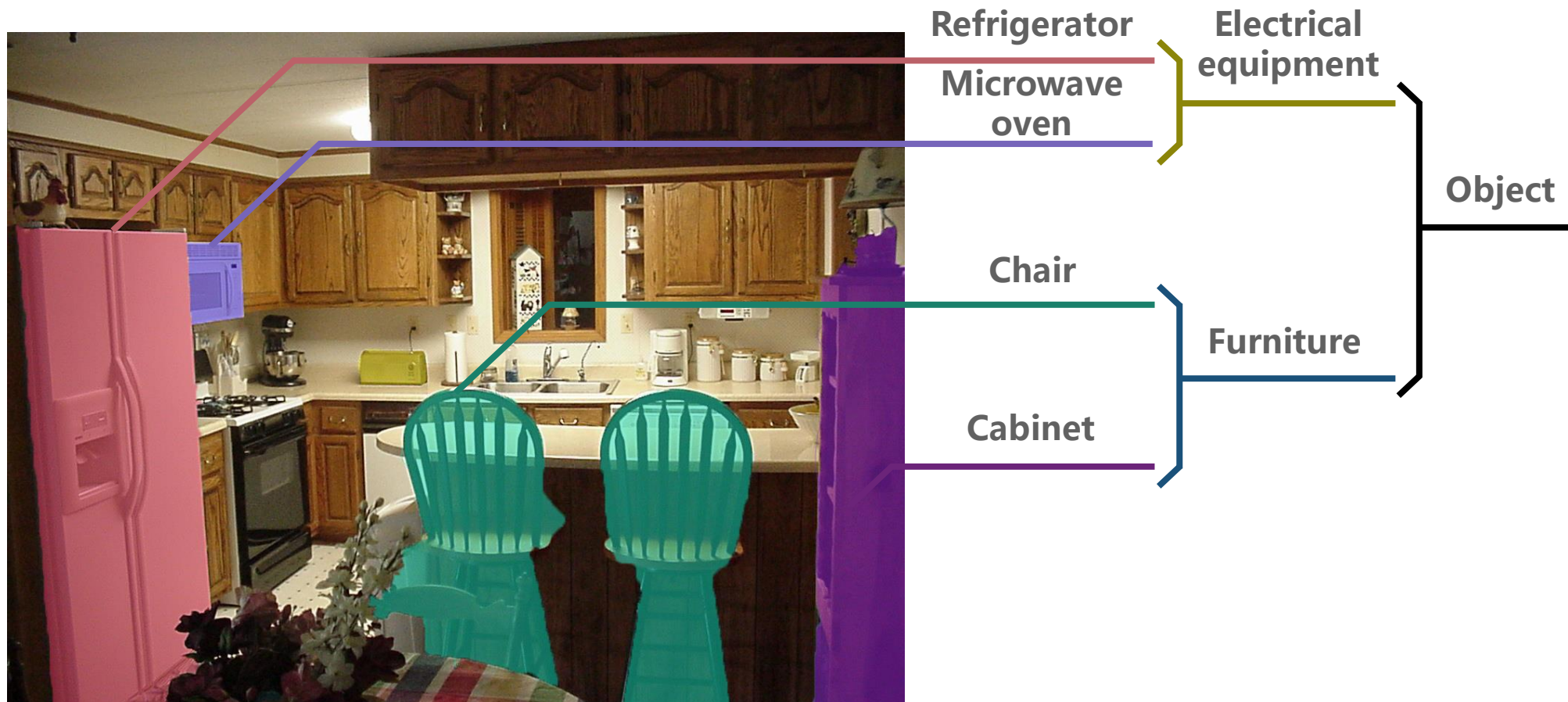
Introduction: Knowledge Graph

- **Structured** visual world: complex scenes arise from the composition of simpler entities.
- Humans abstract visual concepts **hierarchically**.



Introduction: Knowledge Graph

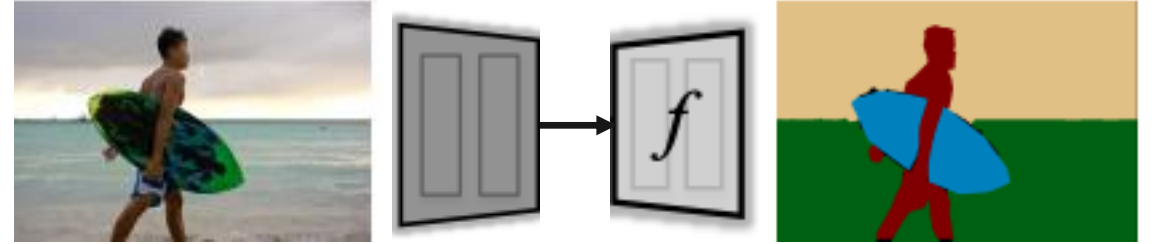
- **Structured** visual world: complex scenes arise from the composition of simpler entities.
- Humans abstract visual concepts **hierarchically**.



HieraSeg: Problem Setup

(Hierarchy-Agnostic) Semantic Segmentation:

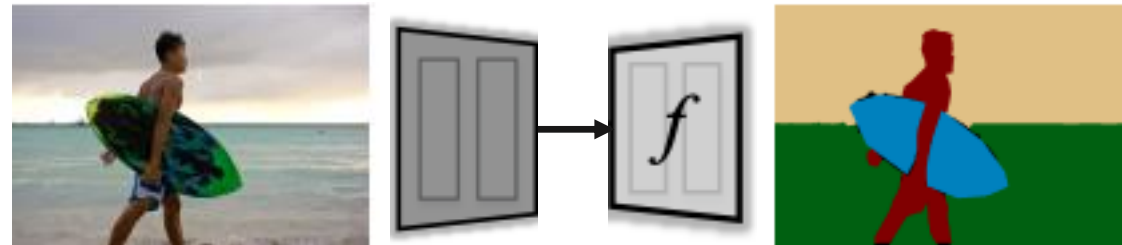
$$\mathcal{L}^{\text{CCE}}(\mathbf{y}) = -\log(\mathbf{y}\hat{\mathbf{v}}_x).$$



HieraSeg: Problem Setup

(Hierarchy-Agnostic) Semantic Segmentation:

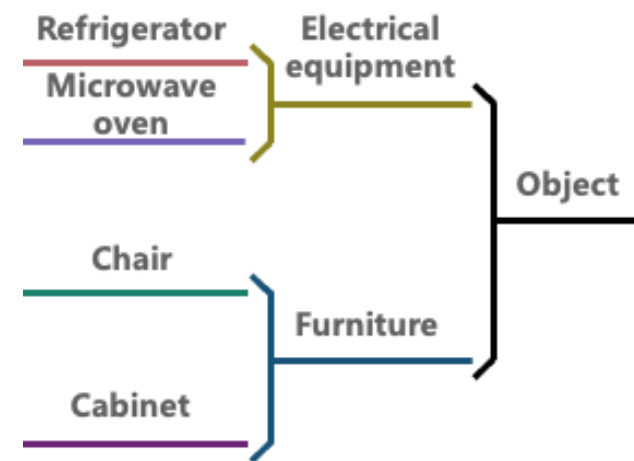
$$\mathcal{L}^{\text{CCE}}(\mathbf{y}) = -\log(\mathbf{y}\hat{v}_x).$$



- **Ignoring** tree-shaped label structure

🗑️ existing solutions rarely explore the **structures between semantic concepts**.

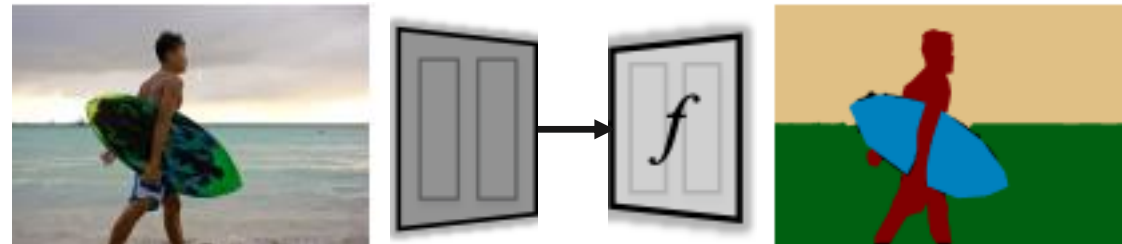
predications cannot reflect the structured nature of our visual world.



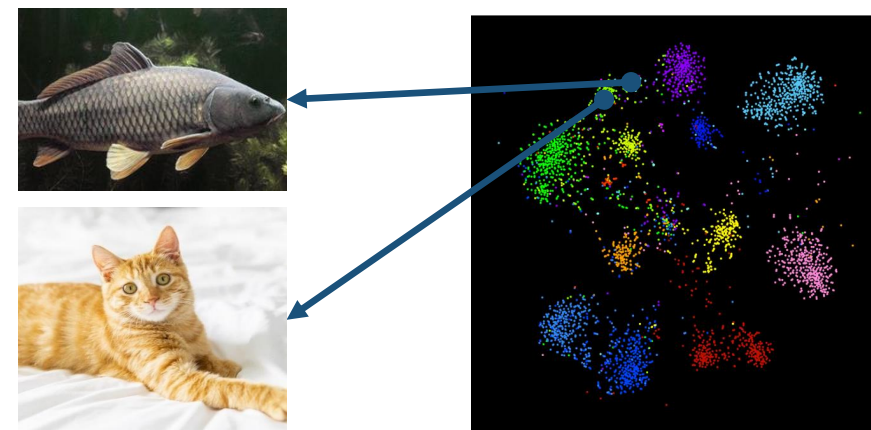
HieraSeg: Problem Setup

(Hierarchy-Agnostic) Semantic Segmentation:

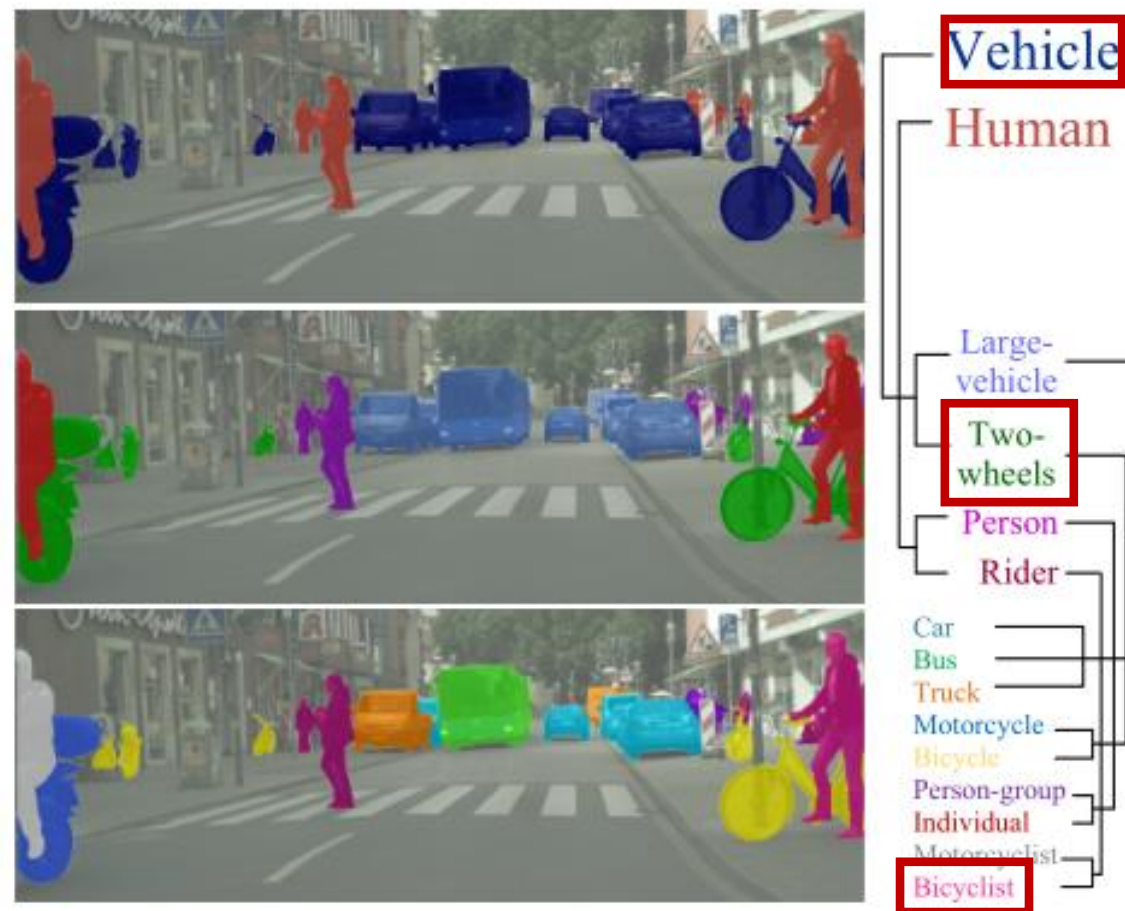
$$\mathcal{L}^{\text{CCE}}(\mathbf{y}) = -\log(\mathbf{y}\hat{v}_x).$$



- distance cannot reflect **semantic similarity**
- ignoring integrate **different levels** of concept abstraction into **representation embedding**.

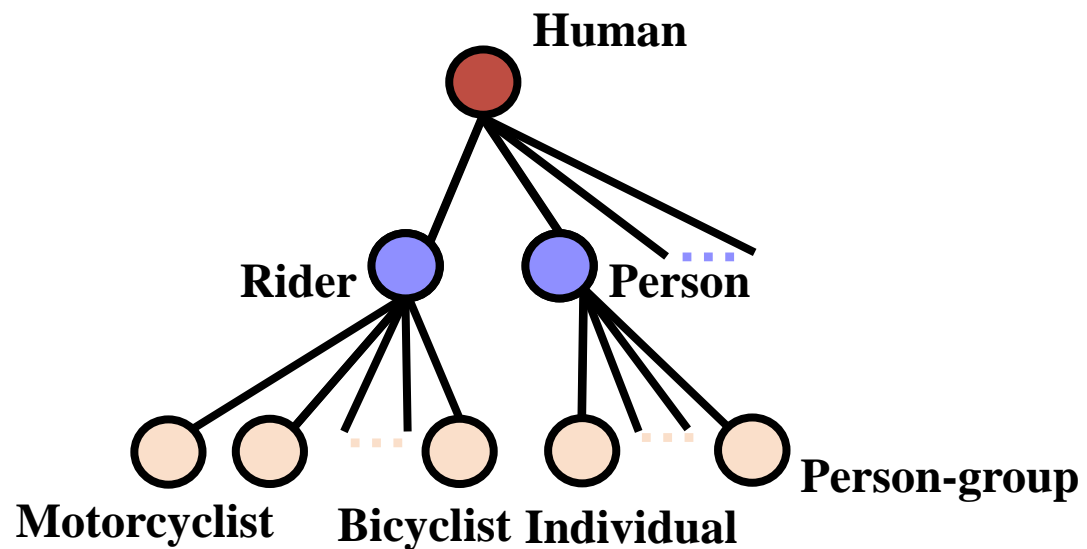


HieraSeg: Problem Setup

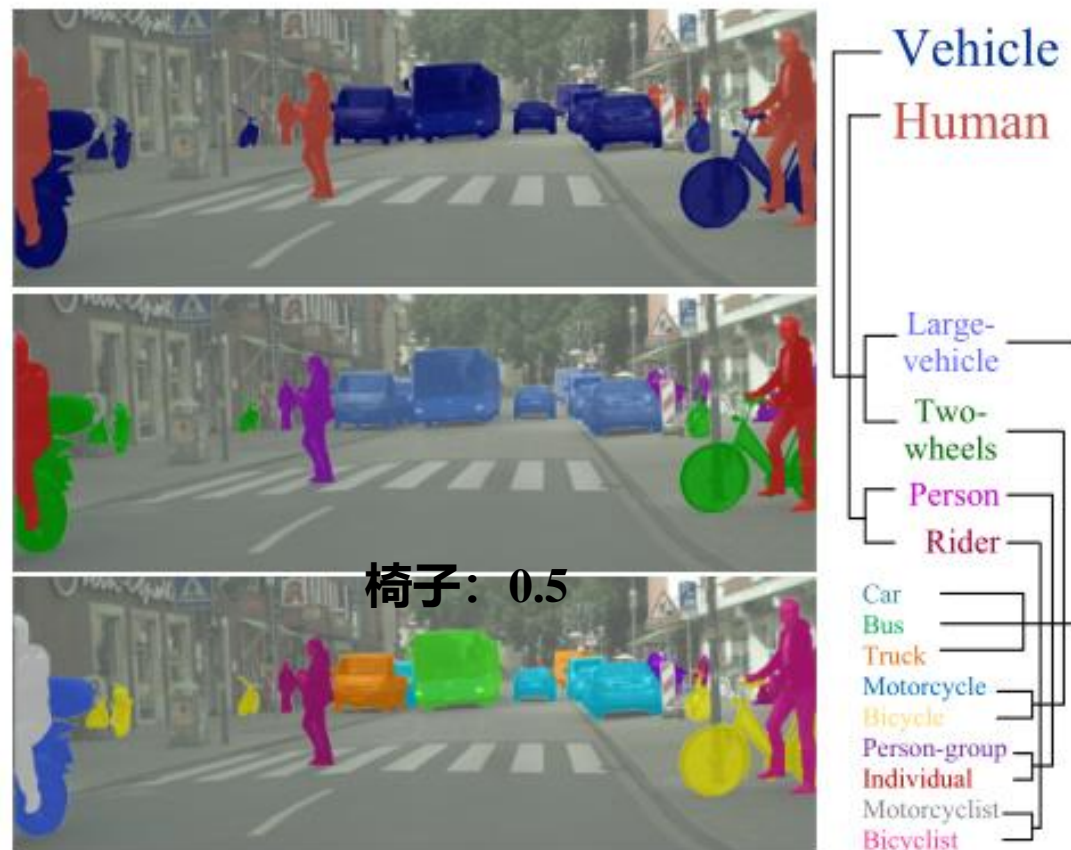


In **hierarchical semantic segmentation (HSS)**, classes are not arranged in a “flat” structure but organized as a tree-shaped hierarchy.

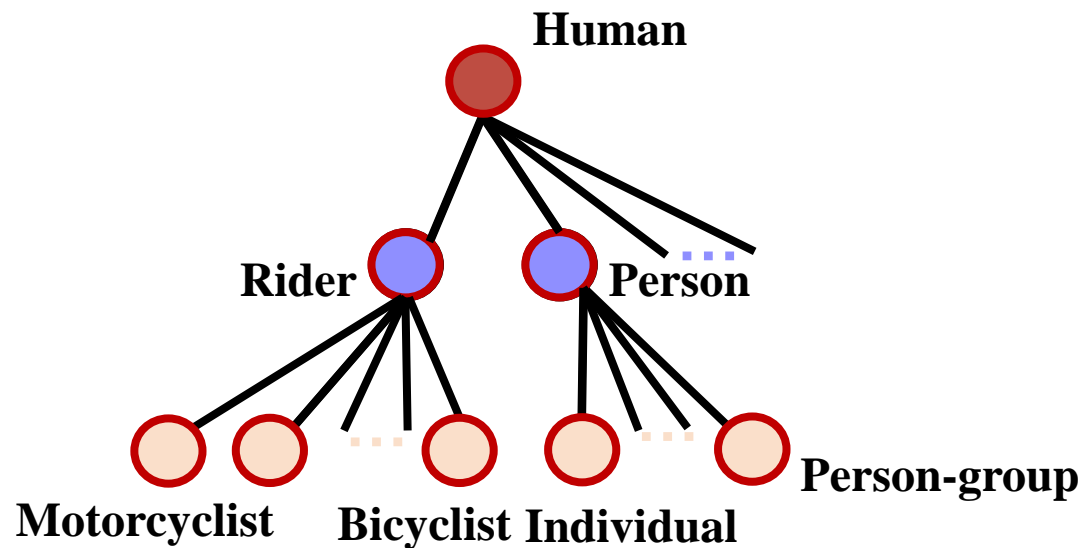
HieraSeg: Problem Setup



$\mathcal{T} = (\mathcal{V}, \mathcal{E})$: tree-structured hierarchy

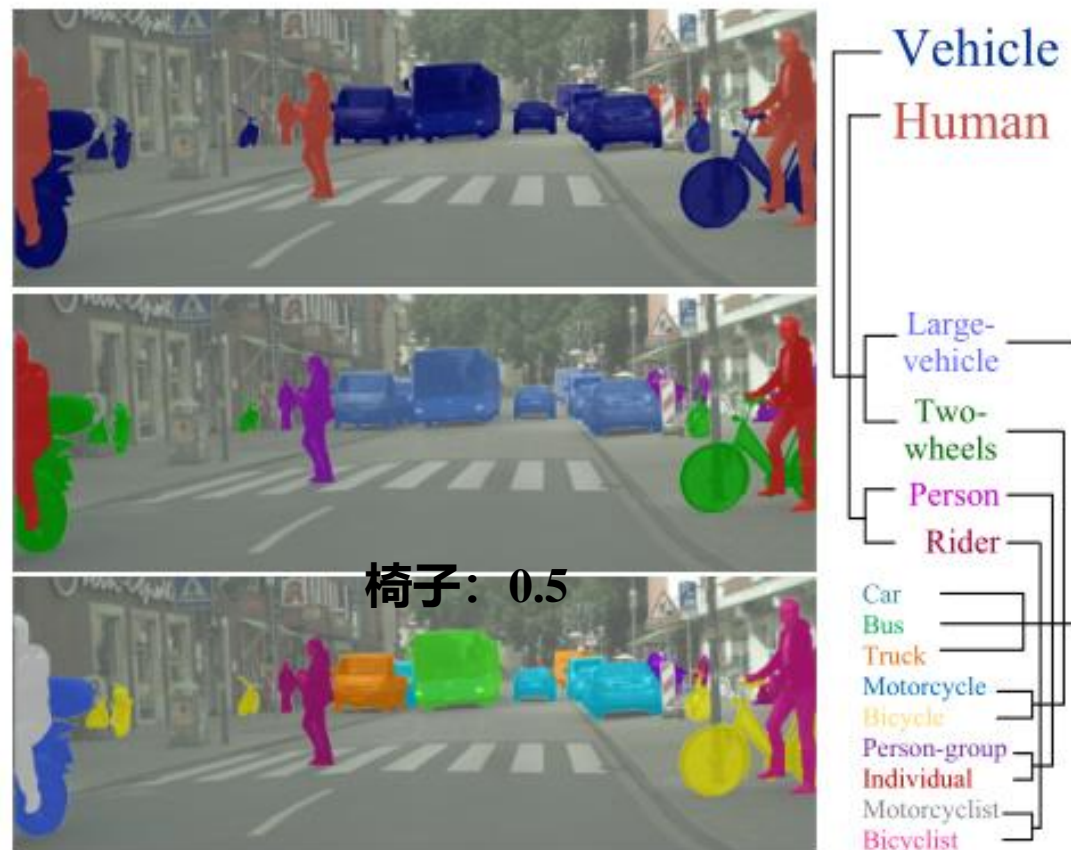


HieraSeg: Problem Setup

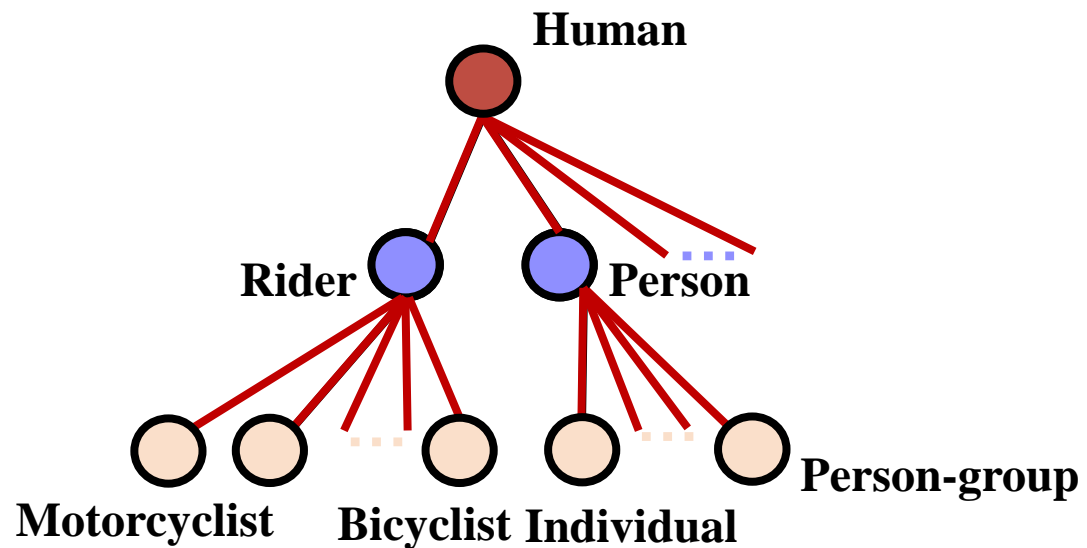


$\mathcal{T} = (\mathcal{V}, \mathcal{E})$: tree-structured hierarchy

$v \in \mathcal{V}$: graph node, semantic concepts



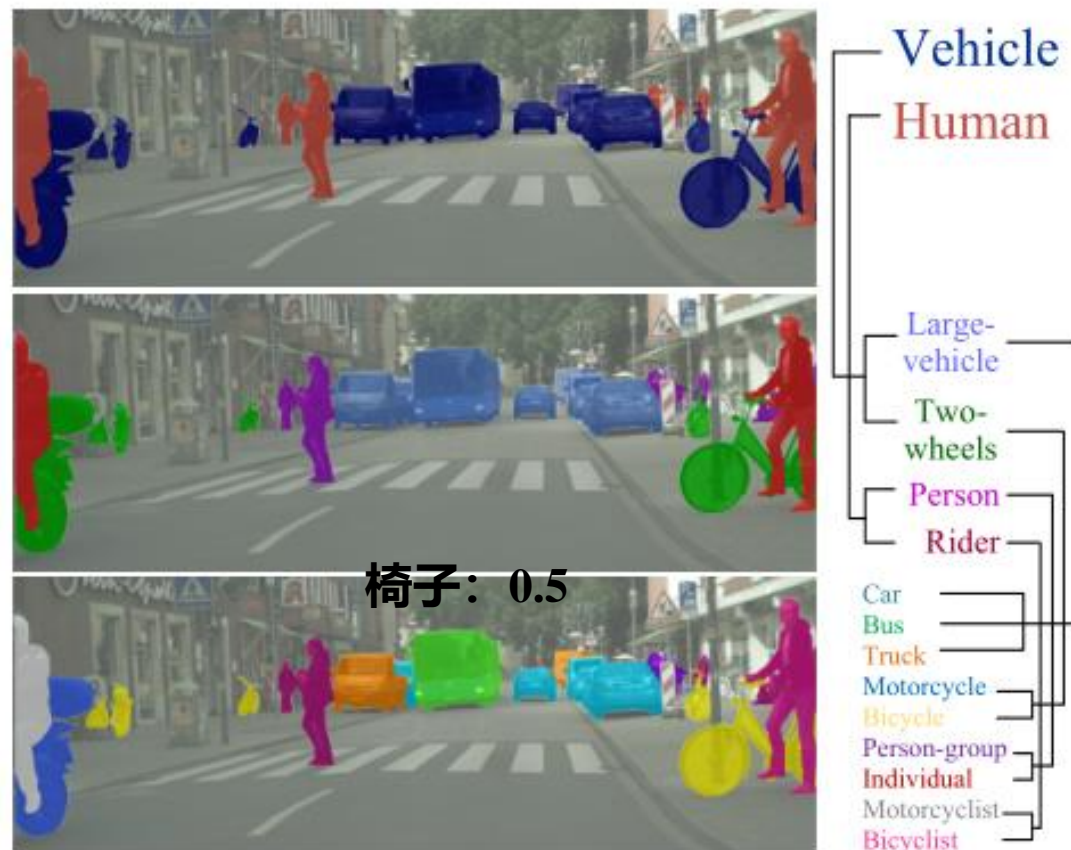
HieraSeg: Problem Setup



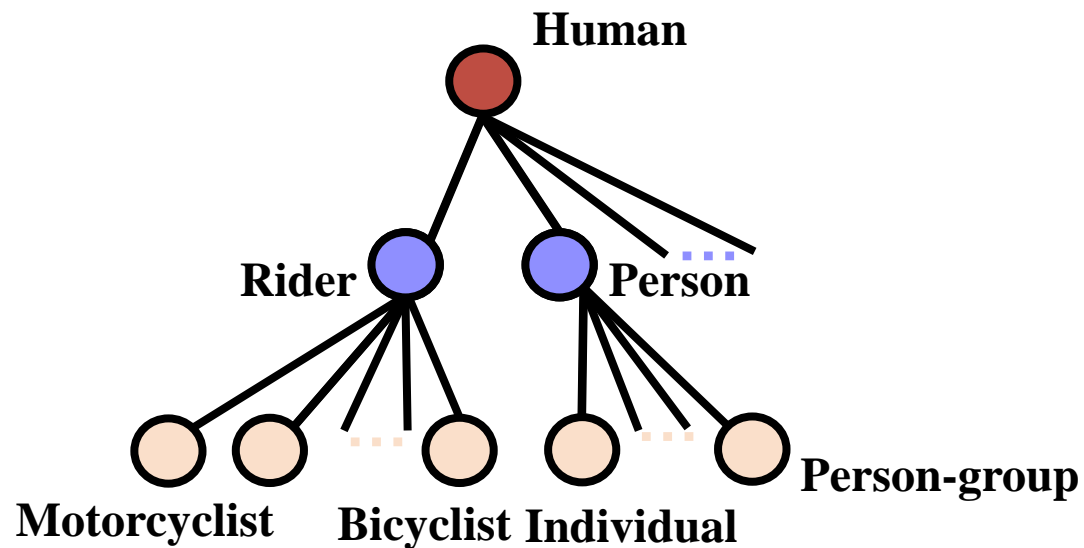
$\mathcal{T} = (\mathcal{V}, \mathcal{E})$: tree-structured hierarchy

$v \in \mathcal{V}$: graph node, semantic concepts

$(u, v) \in \mathcal{E}$: graph edge, relation between concepts



HieraSeg: Problem Setup



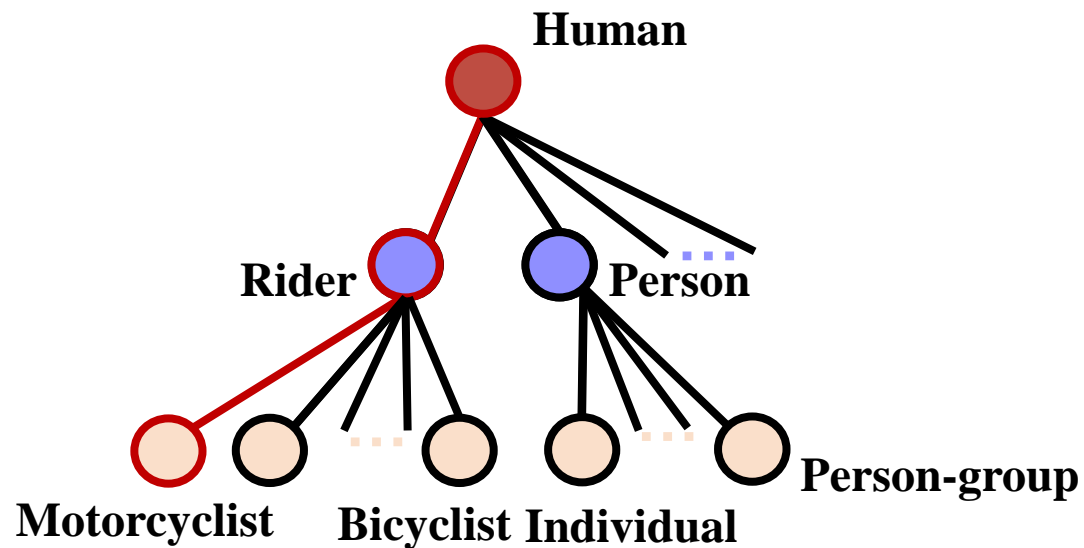
- First formulates **HSS** as a pixel-wise multi-label classification task:

$$\mathcal{L}^{\text{BCE}}(\mathbf{s}) = \sum_{v \in \mathcal{V}} -\hat{l}_v \log(s_v) - (1 - \hat{l}_v) \log(1 - s_v)$$

$\mathbf{s} = [s_v]_{v \in \mathcal{V}} \in [0, 1]^{|\mathcal{V}|}$: score vector

$\hat{\mathbf{l}} = [\hat{l}_v]_{v \in \mathcal{V}} \in \{0, 1\}^{|\mathcal{V}|}$: ground-truth binary label set

HieraSeg: Problem Setup



- First formulates **HSS** as a pixel-wise multi-label classification task:

$$\mathcal{L}^{\text{BCE}}(\mathbf{s}) = \sum_{v \in \mathcal{V}} -\hat{l}_v \log(s_v) - (1 - \hat{l}_v) \log(1 - s_v)$$

- During inference, each pixel i is associated with the top-scoring root-to-leaf path in the class hierarchy :

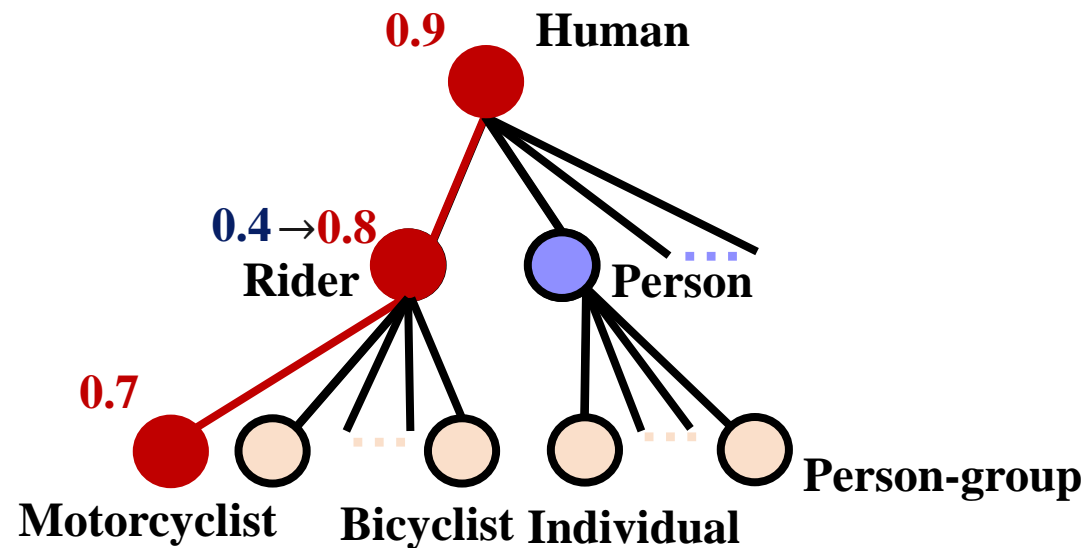
$$\{v_1^*, \dots, v_{|\mathcal{P}|}^*\} = \arg \max_{\mathcal{P} \subseteq \mathcal{T}} \sum_{v_p \in \mathcal{P}} s_{v_p}$$

$\mathbf{s} = [s_v]_{v \in \mathcal{V}} \in [0, 1]^{|\mathcal{V}|}$: score vector

$\hat{\mathbf{l}} = [\hat{l}_v]_{v \in \mathcal{V}} \in \{0, 1\}^{|\mathcal{V}|}$: ground-truth binary label set

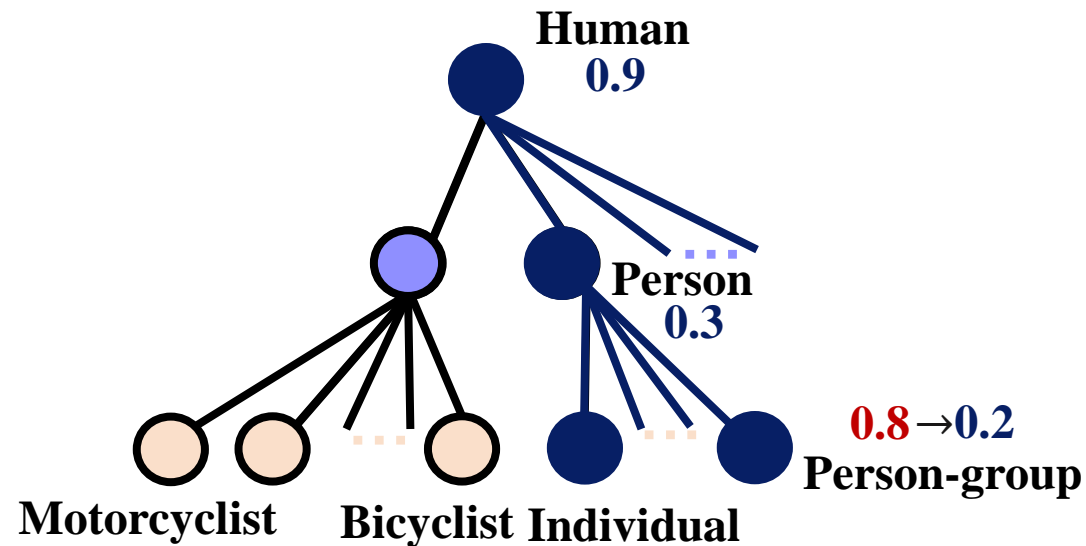
Hierarchy-Aware Segmentation Learning

- **Definition 1 (Positive T -Property):** For each pixel, if a class is labeled positive, all its ancestor nodes (i.e., superclasses) in \mathcal{T} should be labeled positive.
- **Definition 2 (Positive T -Constraint):** For each pixel, if v class is labeled positive, and u is an ancestor node (i.e., superclass) of v , it should hold that $s_v \leq s_u$.

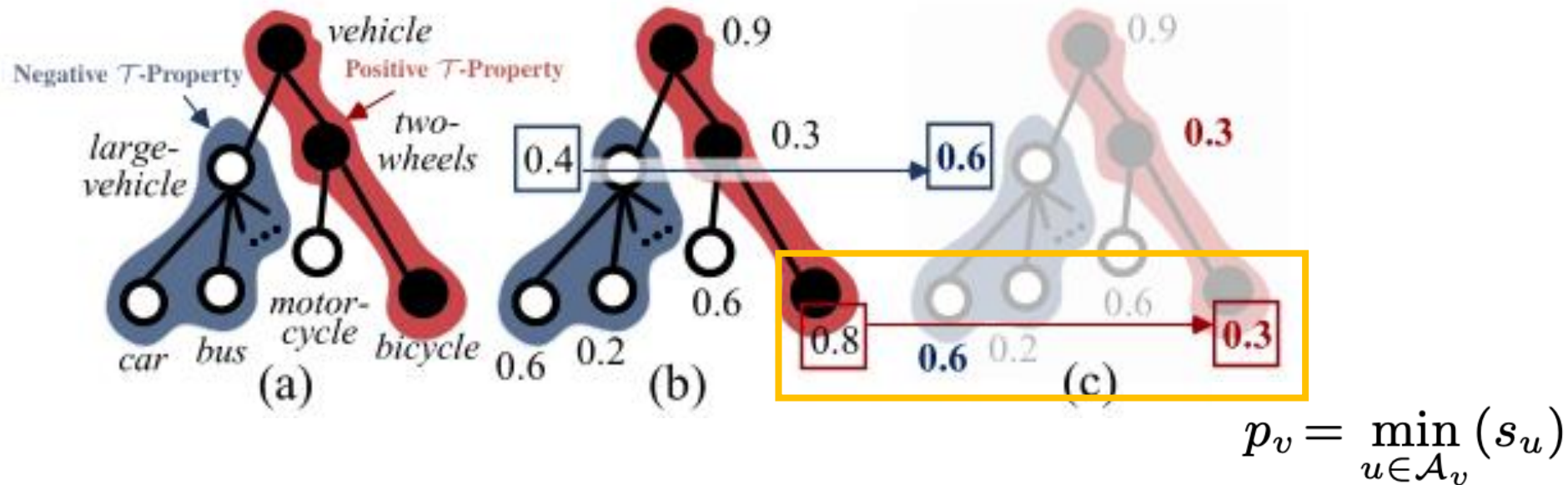


Hierarchy-Aware Segmentation Learning

- **Definition 3 (Positive T -Property):** For each pixel, if a class is labeled positive, all its ancestor nodes (i.e., superclasses) in \mathcal{T} should be labeled positive.
- **Definition 4 (Negative T -Constraint):** For each pixel, if v class is labeled negative, and u is a child node (i.e., subclass) of v , it should hold that $1 - s_v \leq 1 - s_u$.

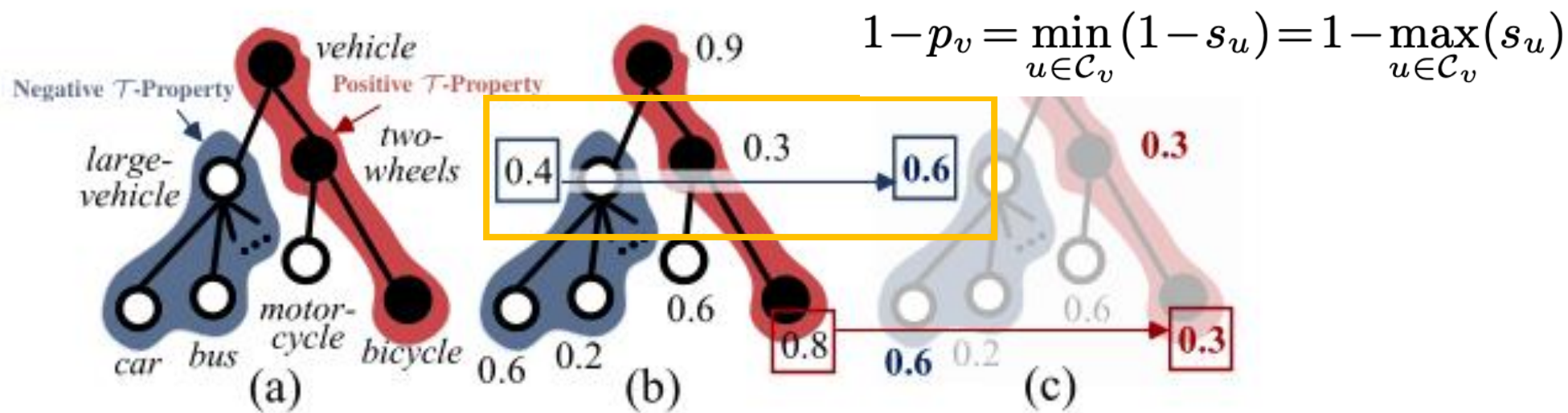


Hierarchy-Aware Segmentation Learning



\mathcal{A}_v : superclass sets of nodes v in \mathcal{T}

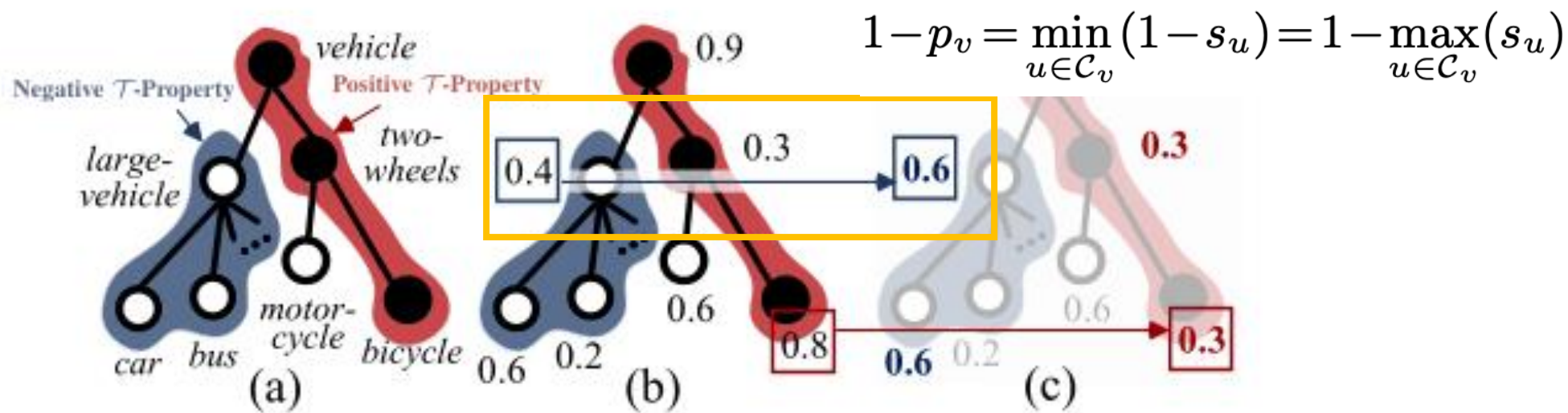
Hierarchy-Aware Segmentation Learning



\mathcal{A}_v : superclass sets of nodes v in \mathcal{T}

\mathcal{C}_v : subclass sets of nodes v in \mathcal{T}

Hierarchy-Aware Segmentation Learning



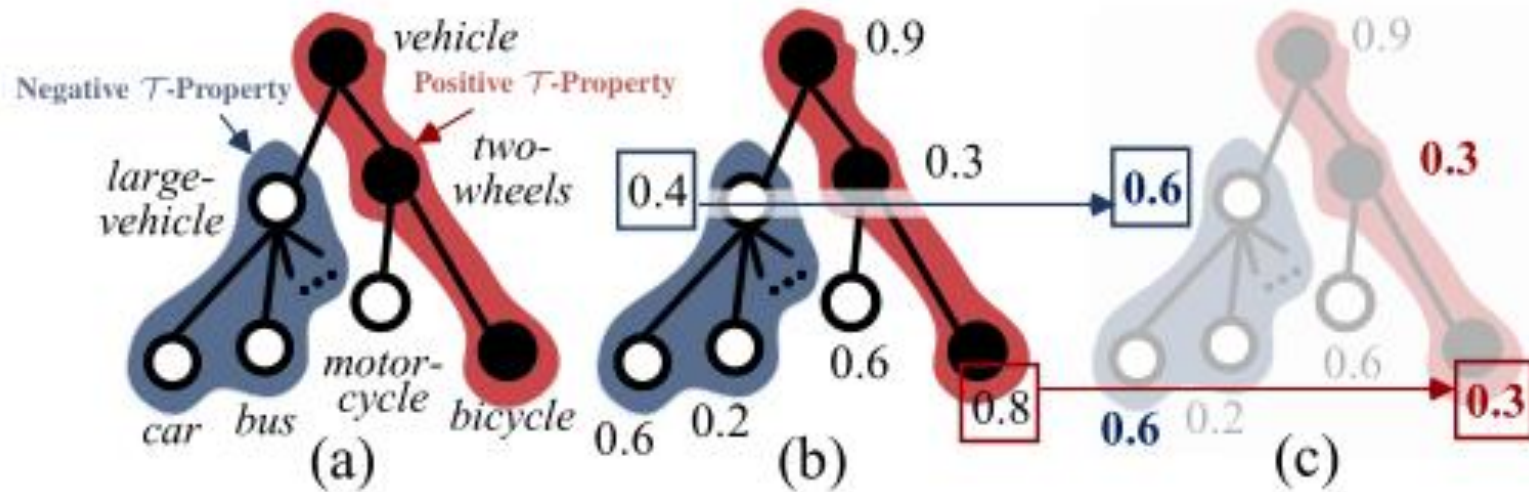
- Hierarchy-coherent score:

$$\begin{cases} p_v = \min_{u \in \mathcal{A}_v} (s_u) & \text{if } \hat{l}_v = 1, \\ 1 - p_v = \min_{u \in \mathcal{C}_v} (1 - s_u) = 1 - \max_{u \in \mathcal{C}_v} (s_u) & \text{if } \hat{l}_v = 0, \end{cases}$$

\mathcal{A}_v : superclass sets of nodes v in \mathcal{T} $\mathbf{s} = [s_v]_{v \in \mathcal{V}} \in [0, 1]^{|\mathcal{V}|}$: score vector

\mathcal{C}_v : subclass sets of nodes v in \mathcal{T}

Hierarchy-Aware Segmentation Learning



- **Tree-Min Loss:**

$$\begin{aligned}\mathcal{L}^{\text{TM}}(\mathbf{p}) &= \sum_{v \in \mathcal{V}} -\hat{l}_v \log(p_v) - (1 - \hat{l}_v) \log(1 - p_v), \\ &= \sum_{v \in \mathcal{V}} -\hat{l}_v \log(\min_{u \in \mathcal{A}_v}(s_u)) - \\ &\quad (1 - \hat{l}_v) \log(1 - \max_{u \in \mathcal{C}_v}(s_u)).\end{aligned}$$

structured score distribution p is constructed by strictly following the hierarchy constraints, and hence the violation of the hierarchy properties can be explicitly penalized.

Hierarchy-Aware Segmentation Learning

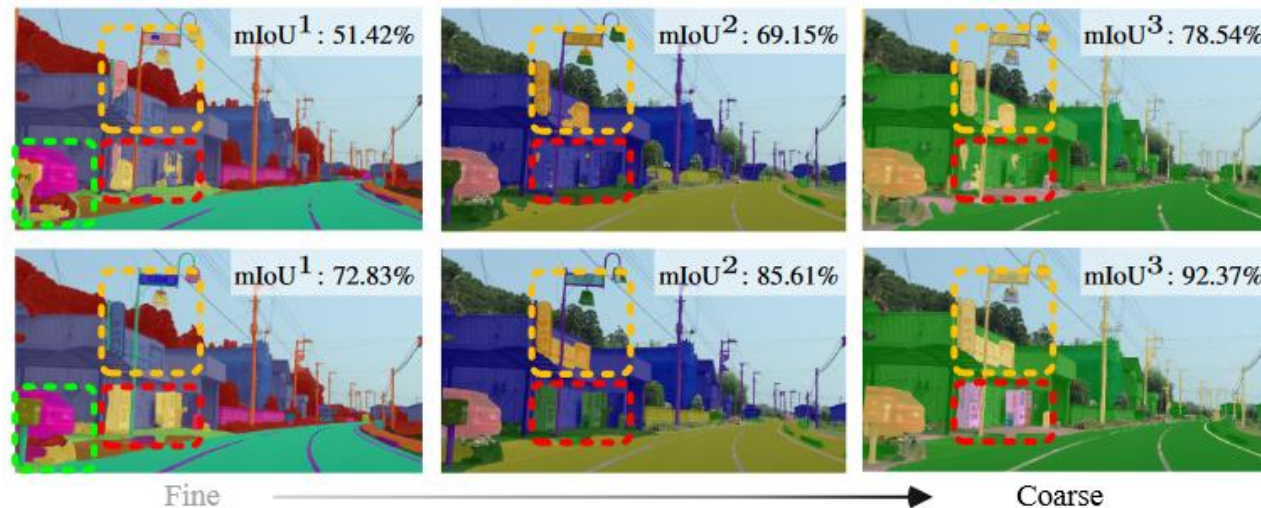


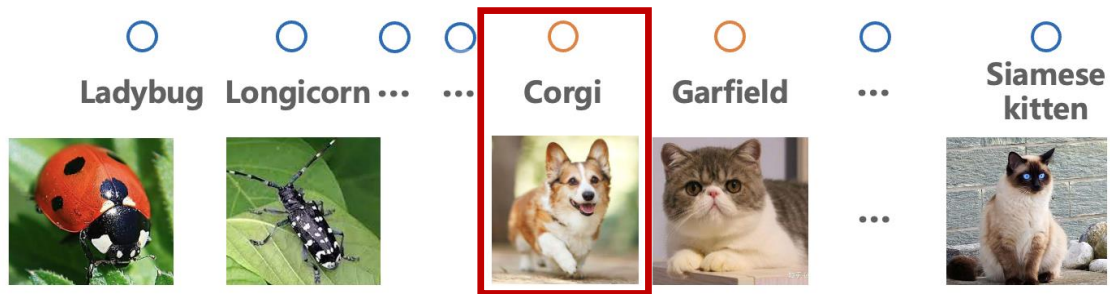
Figure 3. Effect of \mathcal{L}^{BCE} in Eq. 2 (top) vs \mathcal{L}^{FTM} in Eq. 6 (bottom).

- **Focal Tree-Min Loss:**

$$\begin{aligned}\mathcal{L}^{\text{FTM}}(p) &= \sum_{v \in \mathcal{V}} -\hat{l}_v (1-p_v)^\gamma \log(p_v) - (1-\hat{l}_v) (p_v)^\gamma \log(1-p_v), \\ &= \sum_{v \in \mathcal{V}} -\hat{l}_v (1 - \min_{u \in \mathcal{A}_v} (s_u))^\gamma \log(\min_{u \in \mathcal{A}_v} (s_u)) - \\ &\quad (1-\hat{l}_v) (\max_{u \in \mathcal{C}_v} (s_u))^\gamma \log(1 - \max_{u \in \mathcal{C}_v} (s_u)),\end{aligned}$$

we add a modulating factor to the tree-min loss, so as to reduce the relative loss for well-classified pixel samples and focus on those difficult ones.

Pixel-Wise Hierarchical Representation Learning



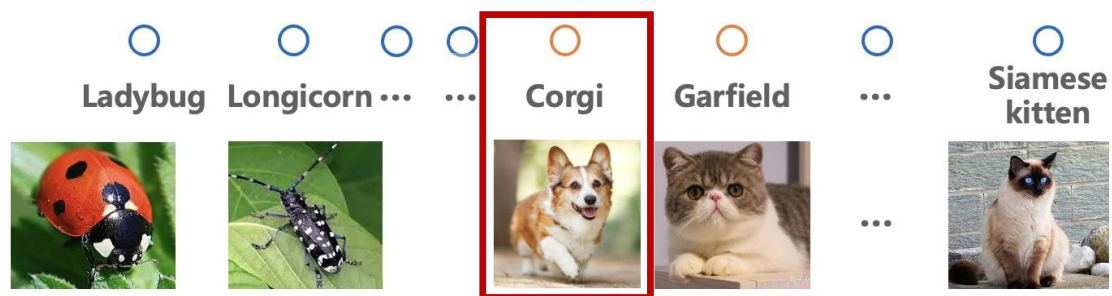
- **Triplet loss:**

Organizing data samples into a high-dimensional space where their distance reflects their semantic similarity:

$$\mathcal{L}(i, i^+, i^-) = \max\{\langle i, i^+ \rangle - \langle i, i^- \rangle + m, 0\} \quad \langle \mathbf{x}, \mathbf{y} \rangle = \frac{1}{2} \left(1 - \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} \right) \in [0, 1]$$

the anchor and positive samples are from the same class, while the anchor and negative samples are from different classes.

Pixel-Wise Hierarchical Representation Learning



- **Triplet loss:**

$$\mathcal{L}(i, i^+, i^-) = \max\{\langle i, i^+ \rangle - \langle i, i^- \rangle + m, 0\} \quad \langle \mathbf{x}, \mathbf{y} \rangle = \frac{1}{2} \left(1 - \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} \right) \in [0, 1]$$

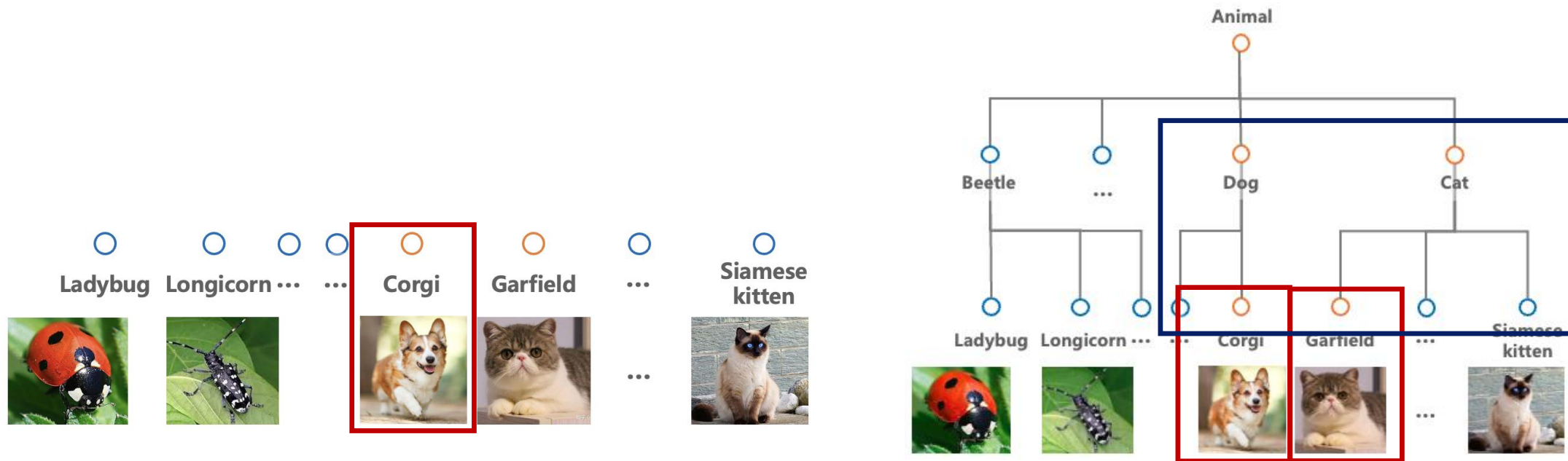
i : Corgi

m : margin, determined

i^+ : must Corgi

i^- : any class expecting Corgi

Pixel-Wise Hierarchical Representation Learning

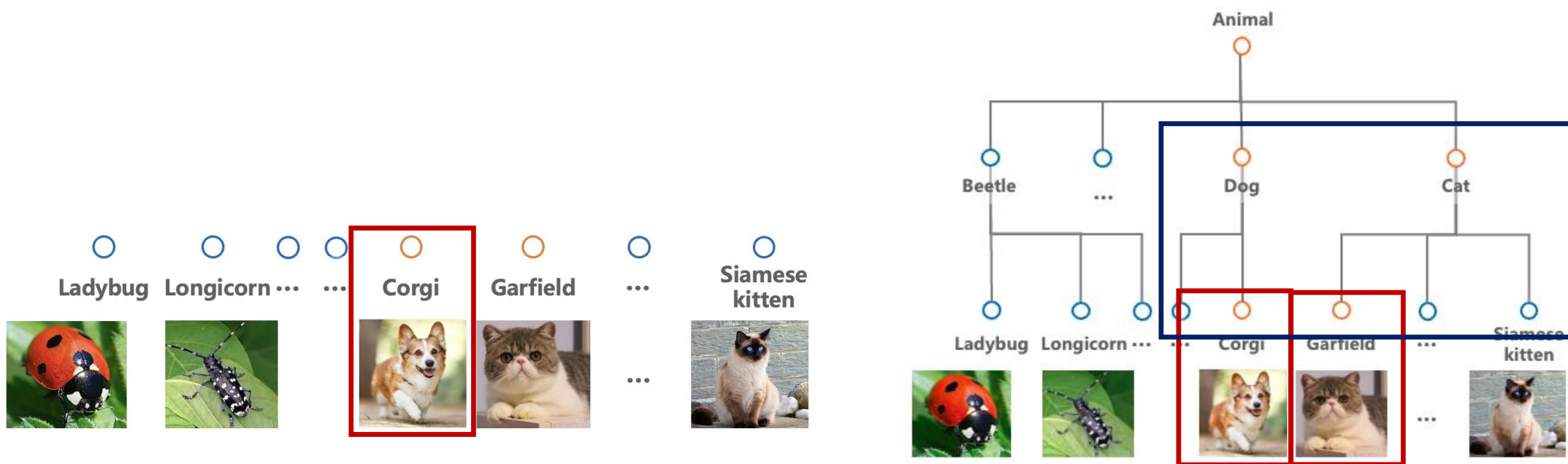


- **Tree-triplet loss:**

We exploit structured knowledge to reshape the pixel embedding space according to class hierarchy \mathcal{T}

the positive samples are more semantically similar to the anchor pixels (i.e., closer in \mathcal{T}), compared with the negative pixels.

Pixel-Wise Hierarchical Representation Learning



- **Tree-triplet loss:**

$$\mathcal{L}(i, i^+, i^-) = \max\{\langle i, i^+ \rangle - \langle i, i^- \rangle + m, 0\} \quad \langle x, y \rangle = \frac{1}{2} \left(1 - \frac{x \cdot y}{\|x\| \|y\|} \right) \in [0, 1]$$

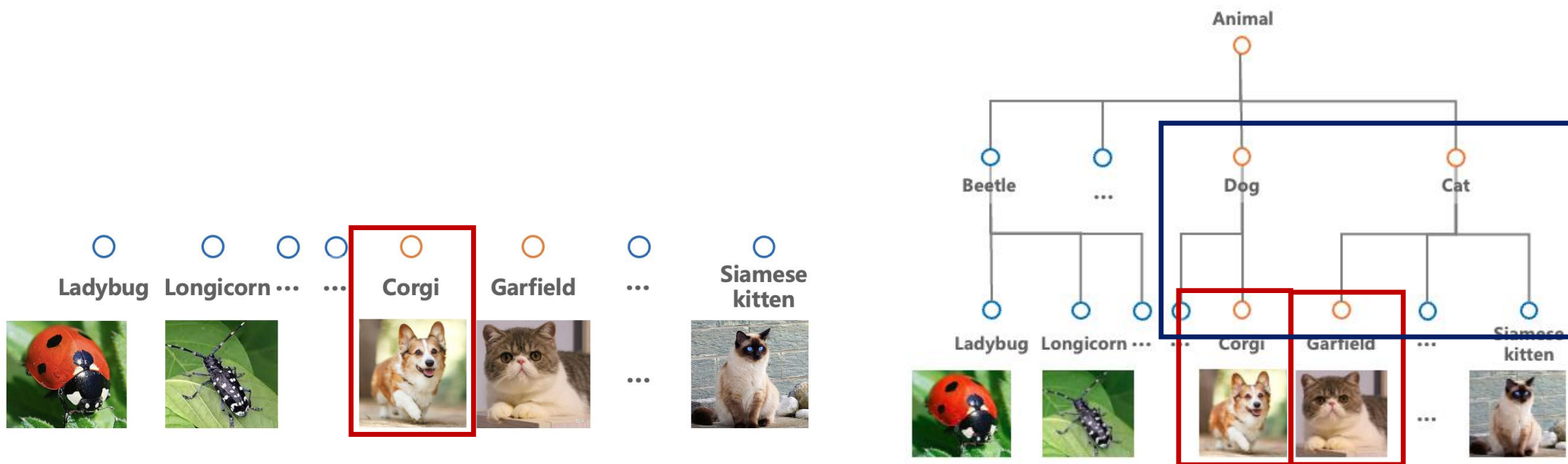
i : Corgi

m : margin, determined \rightarrow hierarchy-aware

i^+ : can be Garfield

i^- : any class expecting Dog and Cat

Pixel-Wise Hierarchical Representation Learning

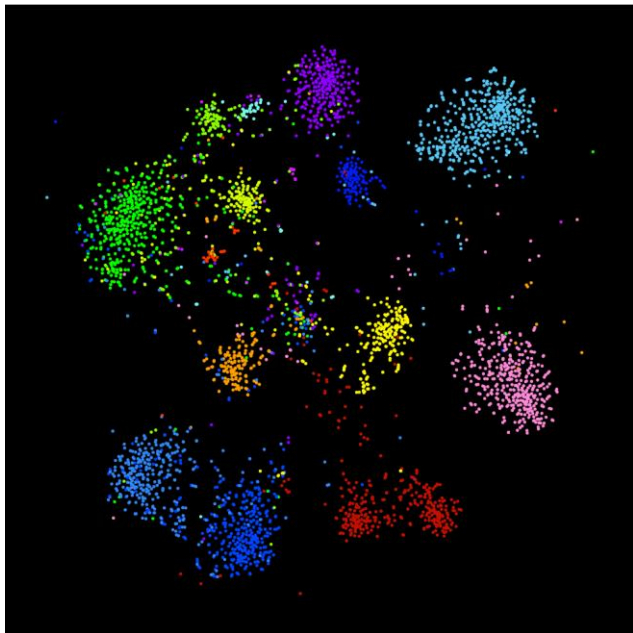


- The separation margin m is determined as:

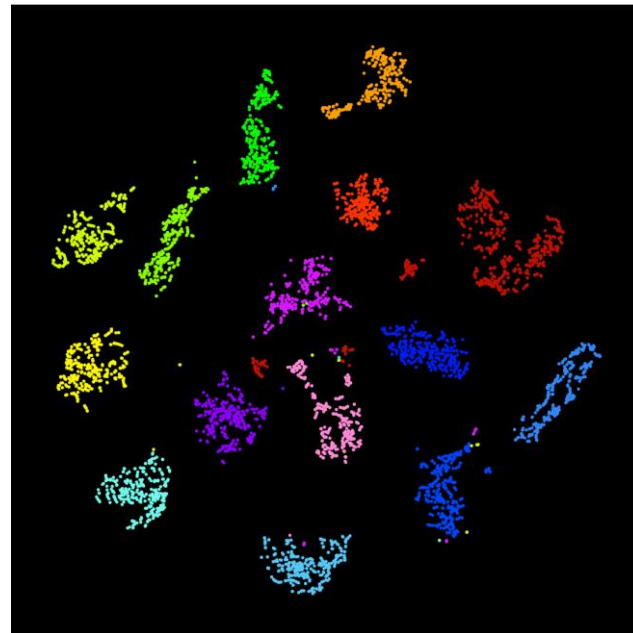
$$m = m_\varepsilon + 0.5m_\tau$$
$$m_\tau = (\psi(\hat{v}_\chi, \hat{v}_\chi^-) - \psi(\hat{v}_\chi, \hat{v}_\chi^+))/2D,$$

where $m_\varepsilon = 0.1$ is set as a constant for the tolerance of the intra-class variance, $m_\tau \in [0, 1]$ is a dynamic violate margin, and D refers to the height of \mathcal{T} .

Pixel-Wise Hierarchical Representation Learning

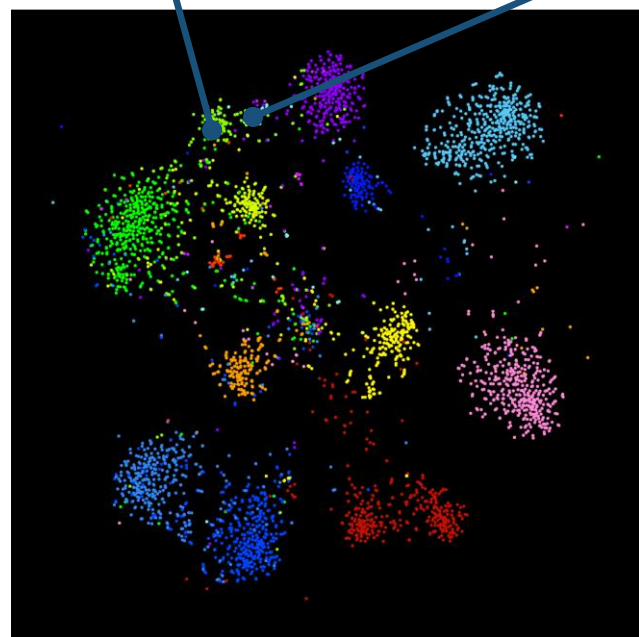


DeepLabV3+

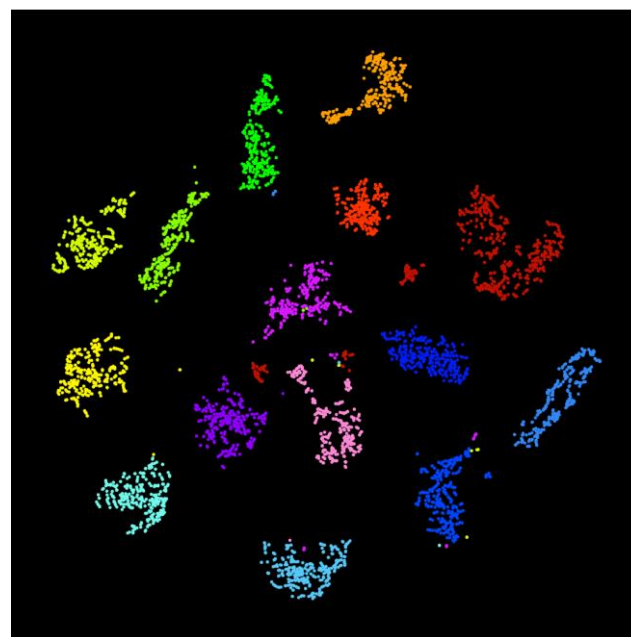


HieraSeg

Pixel-Wise Hierarchical Representation Learning

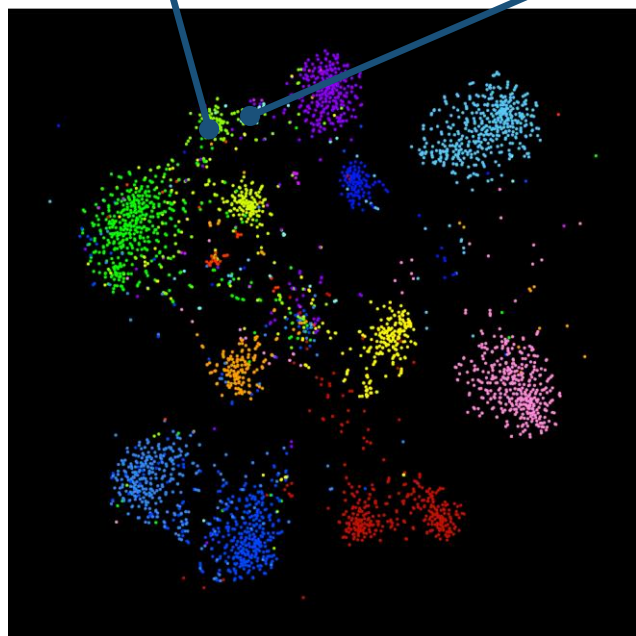


DeepLabV3+

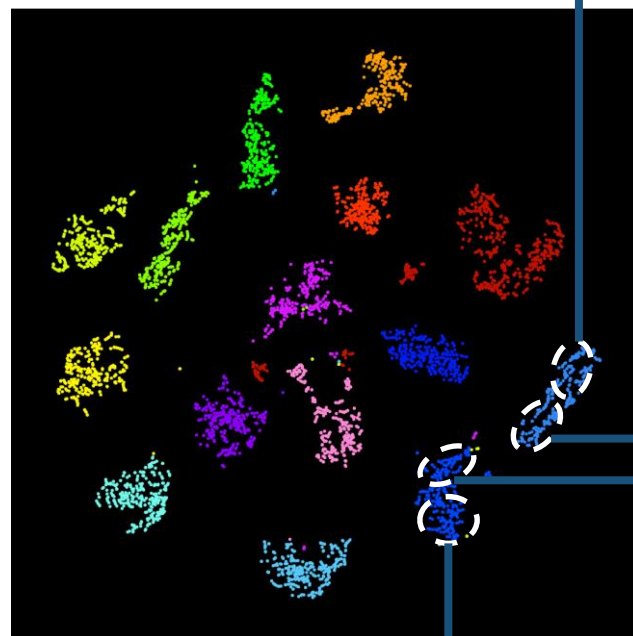


HieraSeg

Pixel-Wise Hierarchical Representation Learning



DeepLabV3+



HieraSeg

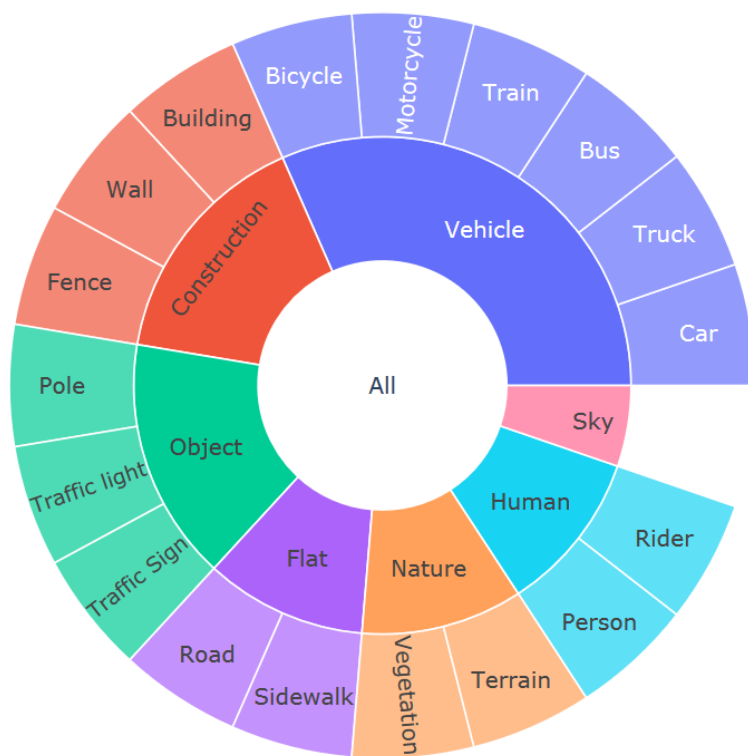


Leopard
subfamily

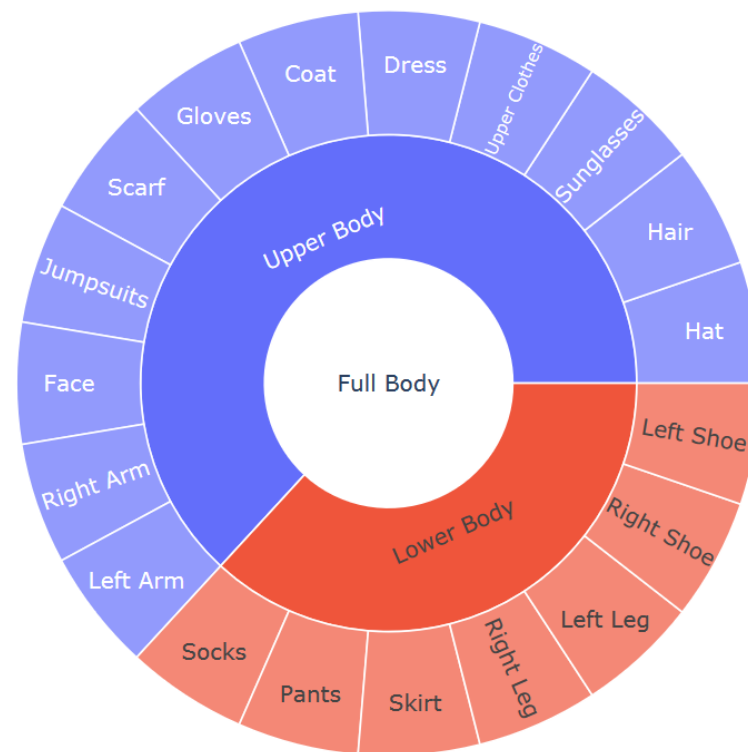
Felidae

Felinae

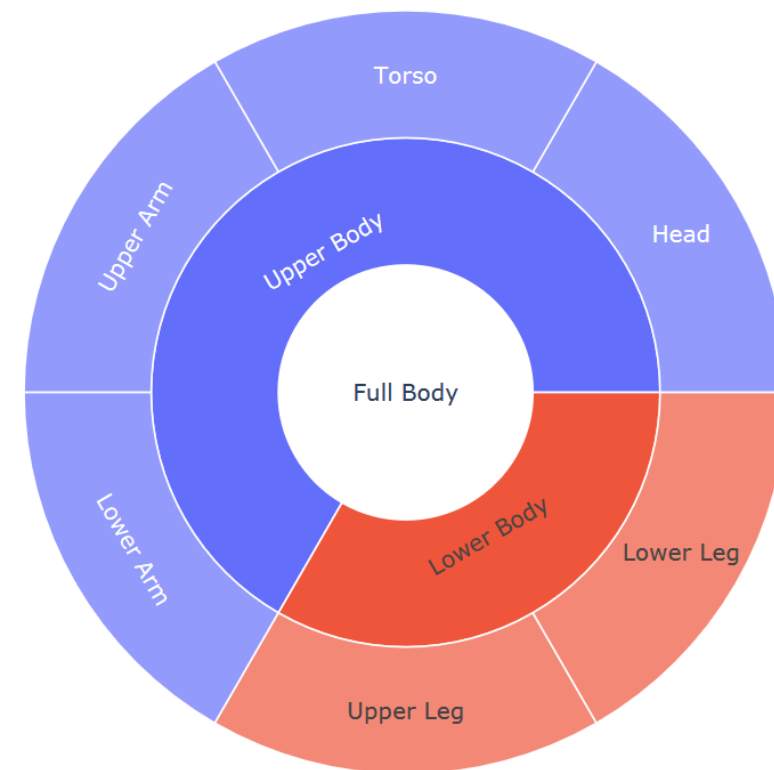
Experiments: Hierarchical architecture



Cityscapes



LIP



PASCAL-Person-Part

Experiments: Analysis of Focal Tree-Min Loss

Analysis of **focal tree-min loss**, we compare it with **four different losses**

Loss	Mapillary Vistas 2.0			Pascal-Person-Part		
	mIoU ³ ↑	mIoU ² ↑	mIoU ¹ ↑	mIoU ³ ↑	mIoU ² ↑	mIoU ¹ ↑
CCE	81.86	68.17	37.43	93.58	83.04	67.84
BCE	81.56	67.61	37.26	93.12	82.55	67.38
Focal	82.63	68.48	38.09	94.07	83.66	68.42
TM	83.48	69.13	38.69	95.32	85.99	72.17
FTM	84.17	69.62	39.17	96.33	86.72	72.89
Full	85.27	71.40	40.16	97.69	88.20	75.44

Experiments: Analysis of Focal Tree-Min Loss

Analysis of **focal tree-min loss**, we compare it with **four different losses**

Loss	Mapillary Vistas 2.0			Pascal-Person-Part		
	mIoU ³ ↑	mIoU ² ↑	mIoU ¹ ↑	mIoU ³ ↑	mIoU ² ↑	mIoU ¹ ↑
CCE	81.86	68.17	37.43	93.58	83.04	67.84
BCE	81.56	67.61	37.26	93.12	82.55	67.38
Focal	82.63	68.48	38.09	94.07	83.66	68.42
TM	83.48	69.13	38.69	95.32	85.99	72.17
FTM	84.17	69.62	39.17	96.33	86.72	72.89
Full	85.27	71.40	40.16	97.69	88.20	75.44

Experiments: Analysis of Focal Tree-Min Loss

Investigating the design of our **tree-triplet loss**:

- Vanilla: vanilla triplet loss with a **constant margin**
- Tree-triplet loss + Constant: adopting **hierarchy-aware triplet sampling strategy**
- Tree-triplet loss + Hierarchy: adopting **hierarchy-aware triplet sampling strategy + dynamic margin**

Triplet Loss	Margin m	Mapillary Vistas 2.0			Pascal-Person-Part		
		mIoU ³ ↑	mIoU ² ↑	mIoU ¹ ↑	mIoU ³ ↑	mIoU ² ↑	mIoU ¹ ↑
Vanilla	Constant	84.25	70.13	39.41	96.58	87.03	74.10
\mathcal{L}^{TT}	Constant	84.66	70.42	39.67	97.30	87.86	74.83
\mathcal{L}^{TT}	Hierarchy	85.27	71.40	40.16	97.69	88.20	75.44

Experiments: Analysis of Focal Tree-Min Loss

Analysis of **distance measure** for **tree-triplet loss**

Distance Measurement	Mapillary Vistas 2.0			Pascal-Person-Part		
	mIoU ³ ↑	mIoU ² ↑	mIoU ¹ ↑	mIoU ³ ↑	mIoU ² ↑	mIoU ¹ ↑
Euclidean	84.23	70.02	39.33	96.28	86.73	73.88
Cosine	85.27	71.40	40.16	97.69	88.20	75.44

Experiments: Quantitative Results

Mapillary Vistas 2.0

Method		Backbone	mIoU ³ ↑	mIoU ² ↑	mIoU ¹ ↑
DeepLabV3+ [13] [ECCV18]		ResNet-101	81.86	68.17	37.43
Seamless [61] [CVPR19]		ResNet-101	-	-	38.17
OCRNet [98] [ECCV20]		HRNet-W48	83.19	69.32	38.26
HMSANet [83] [ArXiv19]		HRNet-W48	84.63	70.71	39.53
MaskFormer [16] [NeurIPS21]		ResNet-101	84.56	70.82	39.60
MaskFormer [16] [NeurIPS21]		Swin-Small	87.93	73.88	42.16
HSSN	DeepLabV3+	ResNet-101	85.27	71.40	40.16
	OCRNet	HRNet-W48	86.46	72.34	41.13
	MaskFormer	Swin-Small	90.02	75.81	43.97

Cityscapes

Method		Backbone	mIoU ² ↑	mIoU ¹ ↑
DeepLabV2 [10] [CVPR17]		ResNet-101	-	70.22
PSPNet [105] [CVPR17]		ResNet-101	-	80.91
PSANet [106] [ECCV18]		ResNet-101	-	80.96
PAN [40] [ArXiv18]		ResNet-101	-	81.12
DeepLabV3+ [13] [ECCV18]		ResNet-101	92.16	82.08
DANet [25] [CVPR19]		ResNet-101	-	81.52
Acfnet [100] [ICCV19]		ResNet-101	-	81.60
CCNet [35] [ICCV19]		ResNet-101	-	81.08
HANet [17] [CVPR20]		ResNet-101	-	81.82
HRNet [79] [TPAMI20]		HRNet-W48	92.12	81.96
OCRNet [98] [ECCV20]		HRNet-W48	92.57	82.33
MaskFormer [16] [NeurIPS21]		Swin-Small	92.96	82.57
HSSN	DeepLabV3+	ResNet-101	93.31	83.02
	OCRNet	HRNet-W48	93.92	83.37
	MaskFormer	Swin-Small	94.39	83.74

LIP

Method		Backbone	mIoU ³ ↑	mIoU ² ↑	mIoU ¹ ↑
SegNet [3] [TPAMI17]		ResNet-101	-	-	18.17
FCN-8s [50] [CVPR15]		ResNet-101	-	-	28.29
DeepLabV2 [10] [CVPR17]		ResNet-101	-	-	41.64
Attention [12] [CVPR16]		ResNet-101	-	-	42.92
MMAN [54] [ECCV18]		ResNet-101	-	-	46.93
DeepLabV3+ [13] [ECCV18]		ResNet-101	88.13	83.97	52.28
CE2P [64] [AAAI19]		ResNet-101	-	-	53.10
BraidNet [48] [ACMMM19]		ResNet-101	-	-	54.42
SemaTree [36] [ECCV20]		ResNet-101	90.78	87.12	54.73
BGNet [102] [ECCV20]		ResNet-101	-	-	56.82
PCNet [101] [CVPR20]		ResNet-101	-	-	57.03
CNIF [80] [ICCV19]		ResNet-101	95.92	91.83	57.74
HRNet [79] [TPAMI20]		HRNet-W48	95.53	91.21	57.23
OCRNet [98] [ECCV20]		HRNet-W48	96.78	92.56	58.47
HHP [83] [CVPR20]		ResNet-101	97.41	93.43	59.25
HSSN	DeepLabV3+	ResNet-101	98.86	94.75	60.37

PASCAL-Person-Part

Method		Head	Torso	U-Arm	L-Arm	U-Leg	L-Leg	U-Body	L-Body	F-Body	B.G.	mIoU ³ ↑	mIoU ² ↑	mIoU ¹ ↑
DeepLabV3+ [13] [ECCV18]		87.02	72.02	60.37	57.36	53.54	48.52	90.07	65.88	93.02	96.07	94.55	84.01	67.84
SPGNet [15] [ICCV19]		87.67	71.41	61.69	60.35	52.62	48.80	-	-	-	95.98	-	-	68.36
PGN [30] [CVPR19]		90.89	75.12	55.83	64.61	55.42	41.57	-	-	-	95.33	-	-	68.40
CNIF [80] [ICCV19]		88.02	72.91	64.31	63.52	55.61	54.96	91.82	66.56	94.33	96.02	95.18	84.80	70.76
SemaTree [36] [ECCV20]		89.15	74.76	63.90	63.95	57.53	54.62	92.36	67.13	95.11	96.84	95.98	85.44	71.59
HHP [83] [CVPR20]		89.73	75.22	66.87	66.21	58.69	58.17	93.44	68.02	96.77	96.94	96.86	86.13	73.12
BGNet [102] [ECCV20]		90.18	77.44	68.93	67.15	60.79	59.27	-	-	-	97.12	-	-	74.42
PCNet [101] [CVPR20]		90.04	76.89	69.11	68.40	60.78	60.14	-	-	-	96.78	-	-	74.59
HSSN	DeepLabV3+	90.19	78.72	70.67	69.71	61.15	60.44	95.86	71.56	98.20	97.18	97.69	88.20	75.44

Experiments: Qualitative Results

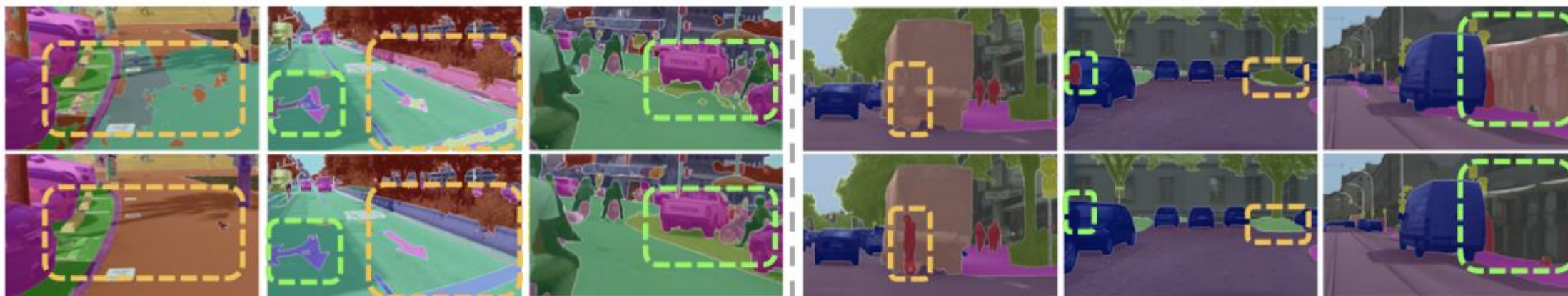
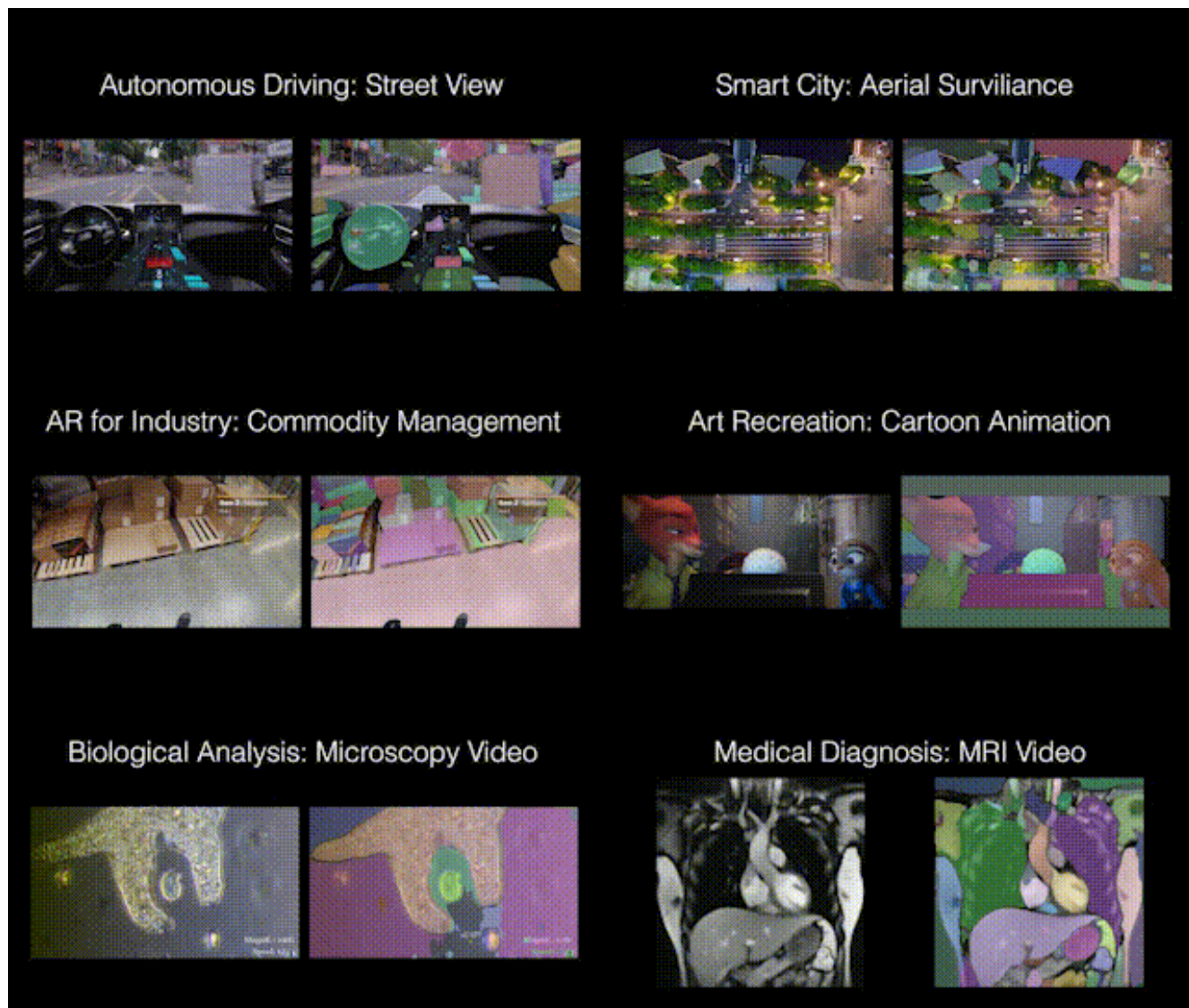


Figure 5. **Visual results** (§4.3) on Mapillary Vistas 2.0 [58] val (left) and Cityscapes [18] val (right). Top: MaskFormer, Bottom: HSSN.



Figure 6. **Visual results** (§4.3) on LIP [44] val (left) and PASCAL-Person-Part [87] test (right). Top: DeepLabV3+, Bottom: HSSN.

Segment and Track Anything (SAM-Track)



Our open-source project, **Segment-and-Track Anything (SAM-Track)**, extends SAM to videos, and supports both automatic and interactive video segmentation modes.





Paper



Code

DNC



Paper



Code

HieraSeg

Thanks!

Wenguan Wang

<https://sites.google.com/view/wenguanwang>
wenguanwang.ai@gmail.com