

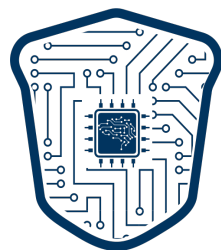
Course I: Trustworthy Data-centric Learning

Prof. Bo Han

HKBU TMLR Group / RIKEN AIP Team

Associate Professor / BAIHO Visiting Scientist

<https://bhanml.github.io>



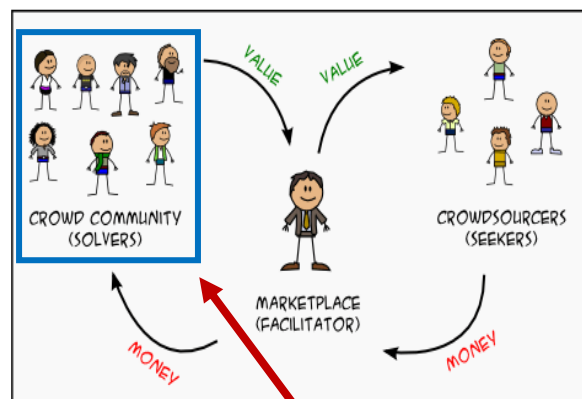
TRUSTWORTHY MACHINE LEARNING AND REASONING

TMLR

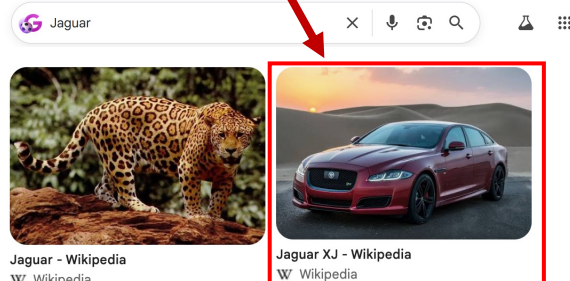


Course I: Trustworthy Data-centric Learning

Part 1. Data-noise Learning

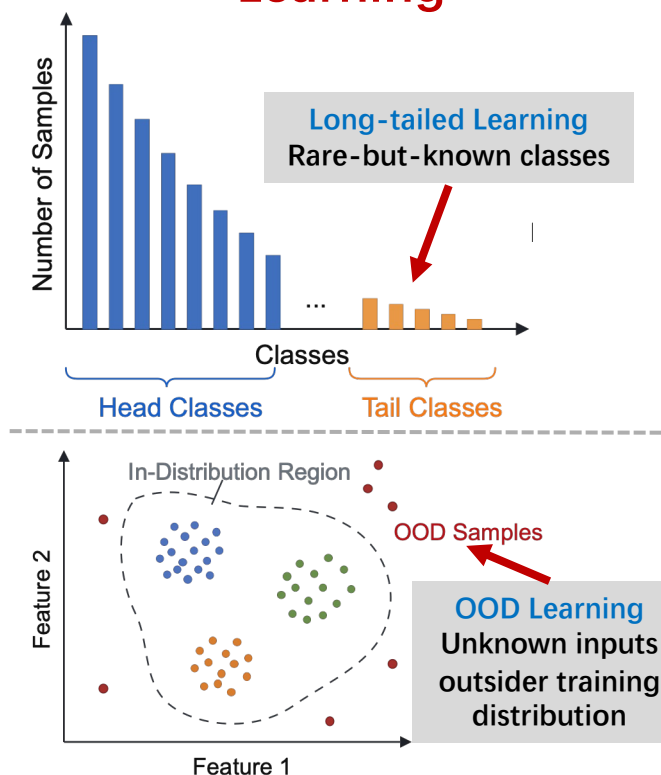


Data Noise



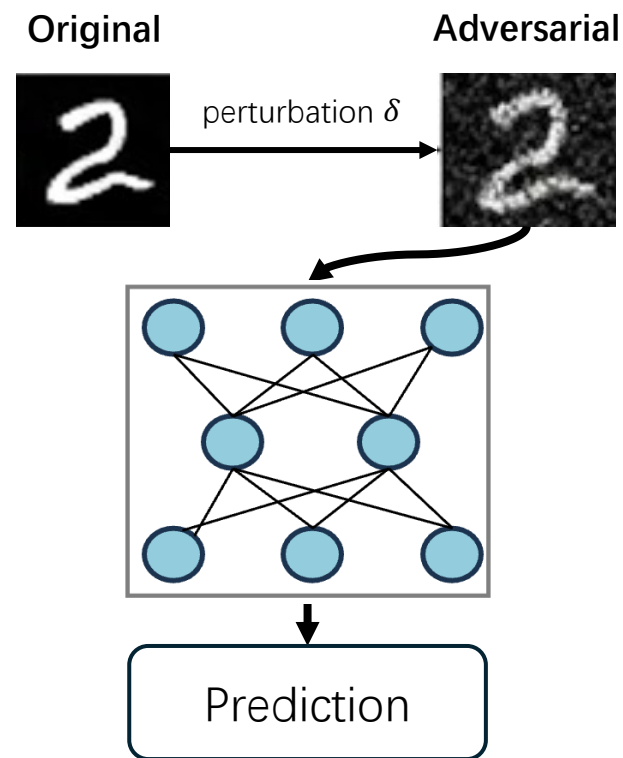
Learn from data with noisy information

Part 2. Long-tailed and Out-of-distribution Learning



Handle imbalanced / OOD data

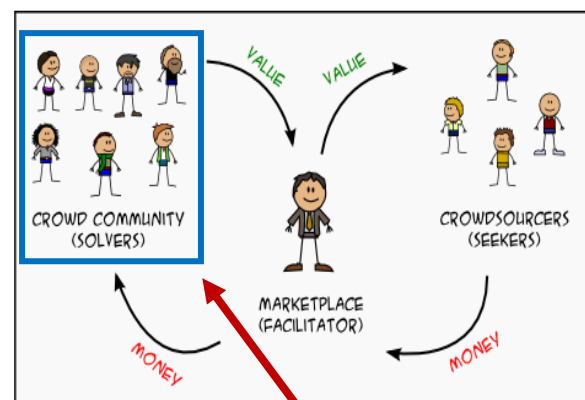
Part 3. Adversarial Attacks and Defenses



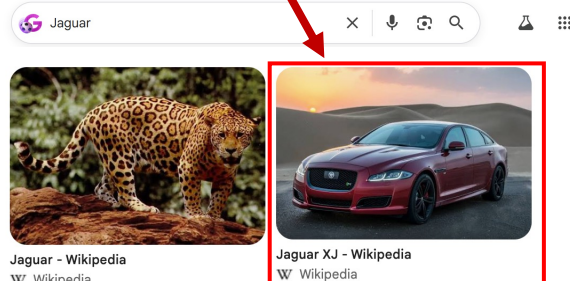
Defend adversarial attacks

Course I: Trustworthy Data-centric Learning

Part 1. Data-noise Learning

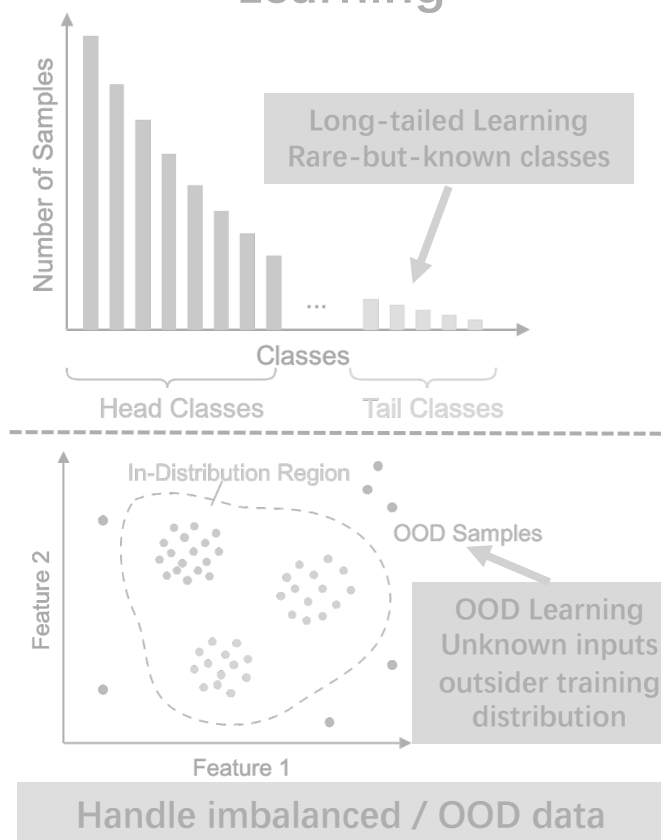


Data Noise

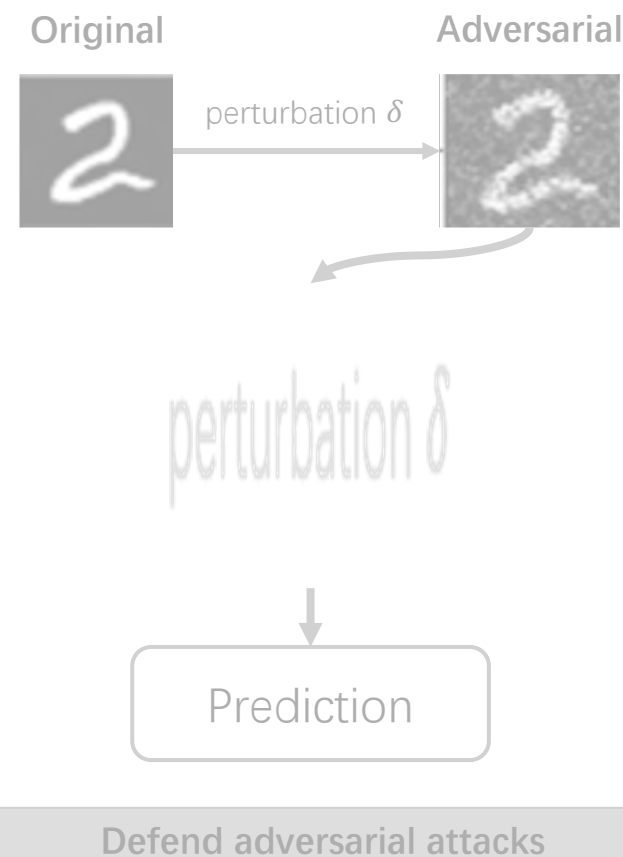


Learn from data with noisy information

Part 2. Long-tailed and Out-of-distribution Learning

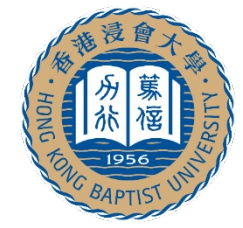


Part 3. Adversarial Attacks and Defenses



Part 1 Data-noise Learning

- Part 1.1: Overview of Data-noise Learning
- Part 1.2: Correspondence Noise Learning
- Part 1.3: Graph-noise Learning
- Part 1.4: Group-structured Noise Learning
- Part 1.5: Takeaways



Part 1 Data-noise Learning

- Part 1.1: Overview of Data-noise Learning
- Part 1.2: Correspondence Noise Learning
- Part 1.3: Graph-noise Learning
- Part 1.4: Group-structured Noise Learning
- Part 1.5: Takeaways

Overview of Data-noise Learning

Typically, **noisy supervision signals** tend to mislead the **minimization of the empirical risk objective** towards wrong directions, causing models to fit incorrect patterns (e.g., noise).

Credit to [1]

Clean training examples:

$$(x_i, y_i)_{i=1}^n \sim p(\mathcal{X}, \mathcal{Y})$$

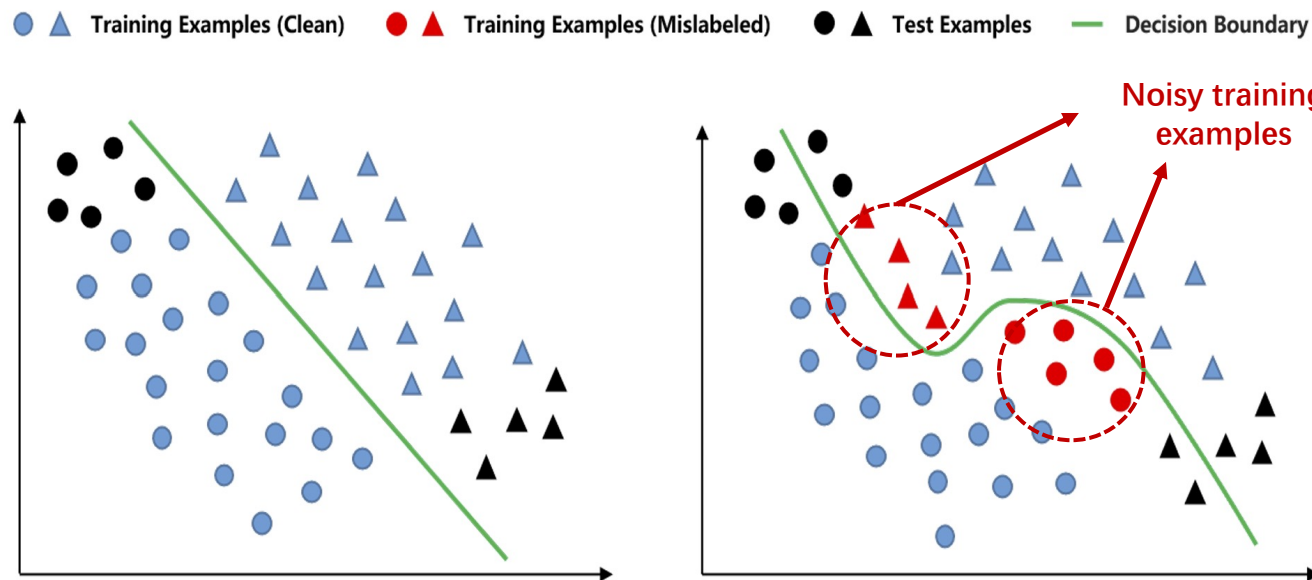
$$R(f) = \frac{1}{n} \sum_{i=1}^n \ell(f(x_i), y_i)$$

Noisy training examples:

$$(x_i, \bar{y}_i)_{i=1}^n \sim p(\mathcal{X}, \bar{\mathcal{Y}})$$

$$\bar{R}_n(f) = \frac{1}{n} \sum_{i=1}^n \ell(f(x_i), \bar{y}_i)$$

Fit on data with **noisy labels**



Observation: Model training is sensitive to noisy supervision signals. Mislabeled training examples may significantly influence the learned decision boundary of classifiers.

AAAI'24 Tutorial:

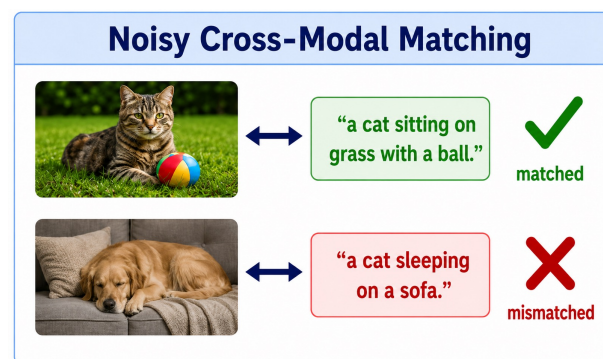
<https://tmlr-group.github.io/static/tutorials/aaai24/bo.pdf>

[1] Selective-supervised contrastive learning with noisy labels. In *CVPR*, 2022.

<https://bhanml.github.io/> & <https://github.com/tmlr-group>

Scenarios in Data-noise Learning

Data-noise learning considers scenarios in which the supervision signals may be incorrect, including **correspondence noise**, **graph noise**, and **group-structured noise**.



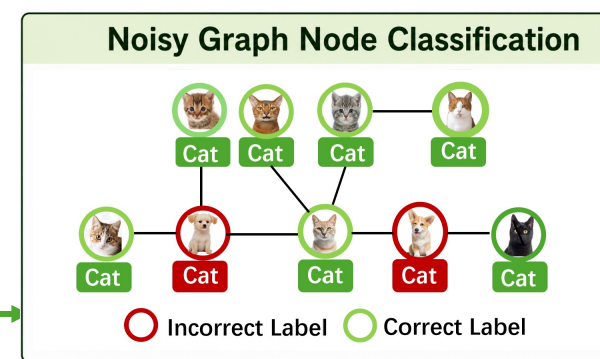
Mismatched image-text pairs are treated as matched ones due to web crawling.

Correspondence noise

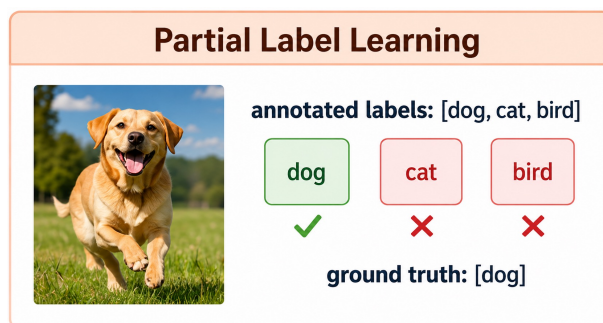
There exist **mismatch** relations between paired image-text data.

Graph noise

Some nodes are annotated with **incorrect node labels**.



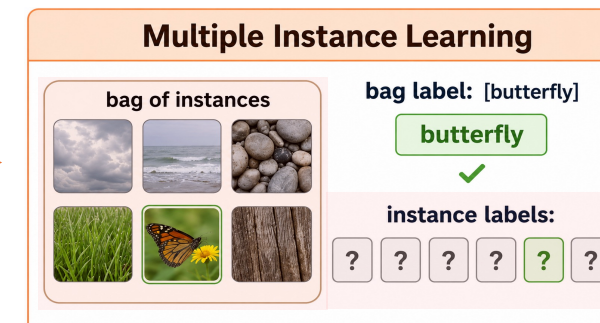
Pets may be assigned **incorrect labels** due to imperfect human annotation.



Ground-truth label is hidden in **multiple candidate labels**.

Group-structured noise

The true instance-label pairs are submerged by **multiple noisy candidates in a group** due to coarse annotation.



A label is provided for a **bag of instances** rather than for individual instances.

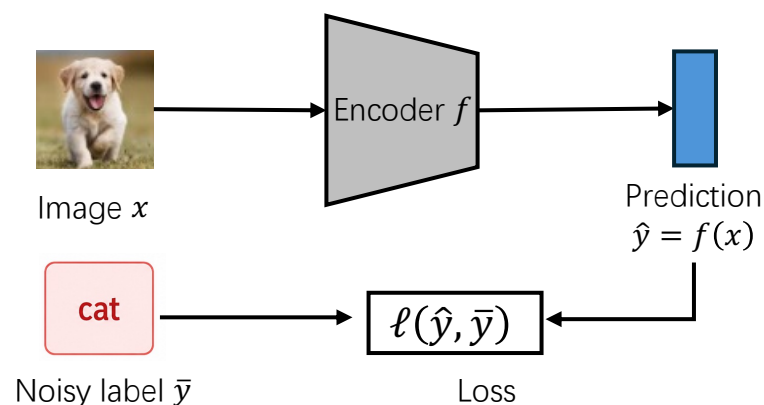
Part 1 Data-noise Learning

- Part 1.1: Overview of Data-noise Learning
- Part 1.2: Correspondence Noise Learning
- Part 1.3: Graph-noise Learning
- Part 1.4: Group-structured Noise Learning
- Part 1.5: Takeaways

Correspondence Noise Learning

Learning from **noisy labels**

Problem: Learning robust models with mislabeled instances.



Noisy training examples:

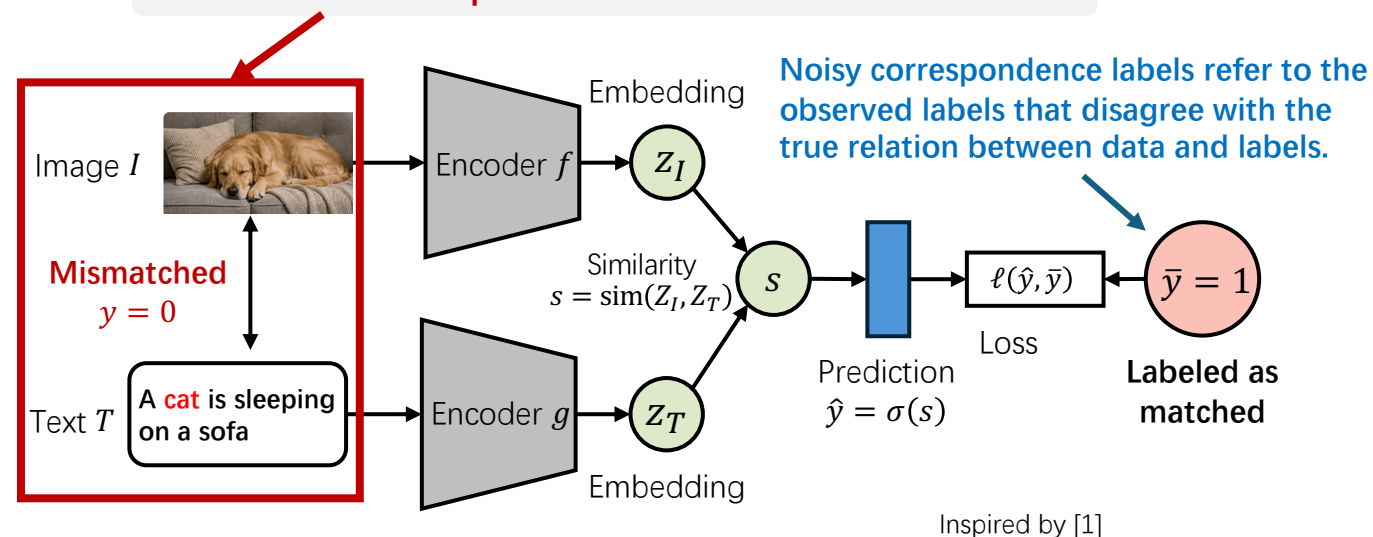
$$(x_i, \bar{y}_i)_{i=1}^n \sim p(\mathcal{X}, \bar{\mathcal{Y}})$$

$$\bar{R}_n(f) = \frac{1}{n} \sum_{i=1}^n \ell(f(x_i), \bar{y}_i)$$

Challenge: Noisy **single-modal** labels mislead the model learning.

Learning from **noisy correspondence**

Problem: Learning to **align data-label** representations with **mismatched pairs**.



Noisy image-text pairs with wrong label signals:

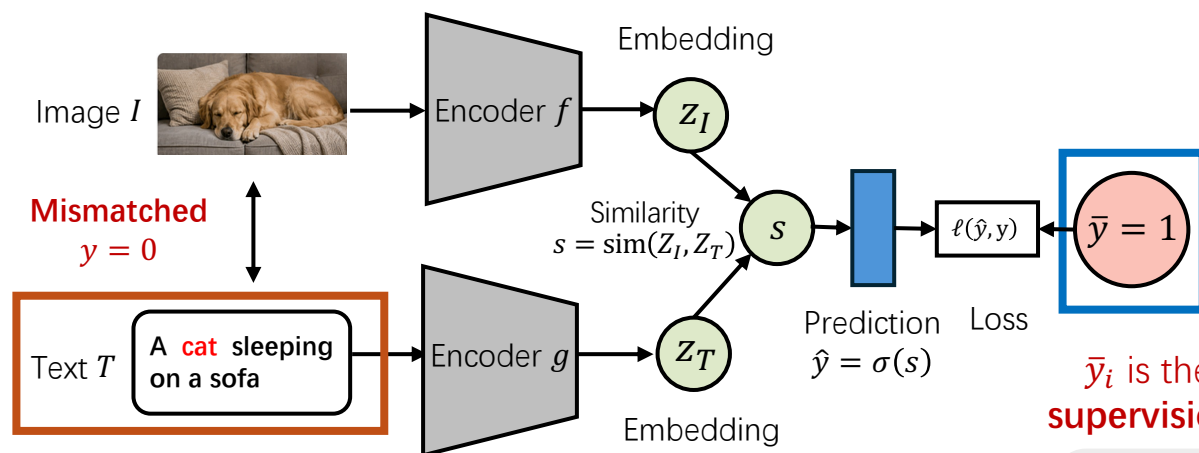
$$(I_i, T_i, \bar{y}_i)_{i=1}^n \sim p(\mathcal{I}, \mathcal{T}, \bar{\mathcal{Y}})$$

$$\bar{R}_n(f, g) = \frac{1}{n} \sum_{i=1}^n \ell(\hat{y}_i, \bar{y}_i), \quad \hat{y}_i = \sigma(\text{sim}(f(I_i), g(T_i)))$$

Challenge: Noisy correspondence labels corrupt the **cross-modal** alignment.

Correspondence Noise Learning

Correspondence noise learning aims at mitigating **the mismatch data issue** (e.g., noisy supervision signals for unmatched image-text pairs) in **cross-modal** learning.



- **From the label perspective:** Modifications can be performed on the labels to instruct models to align the correct information.
- **From the data perspective:** Textual data can be refined for correct alignment.

Correspondence Noise Learning

$$(I_i, T_i, \bar{y}_i)_{i=1}^n \sim p(\mathcal{I}, \mathcal{T}, \bar{\mathcal{Y}})$$

$$\bar{R}_n(f, g) = \frac{1}{n} \sum_{i=1}^n \ell(\hat{y}_i, \bar{y}_i), \quad \hat{y}_i = \sigma(\text{sim}(f(I_i), g(T_i)))$$

Challenge: Noisy correspondence labels corrupt the cross-modal alignment.

Label Perspective

$\bar{y}_i$ is the **noisy supervision signal**

$$(I_i, T_i, \bar{y}_i)_{i=1}^n \sim p(\mathcal{I}, \mathcal{T}, \bar{\mathcal{Y}})$$

$$\bar{R}_n(f, g) = \frac{1}{n} \sum_{i=1}^n \ell(\hat{y}_i, \bar{y}_i)$$

$$\Rightarrow \bar{R}_n(f, g) = \frac{1}{n} \sum_{i=1}^n \ell(\hat{y}_i, \tilde{y}_i)$$

Solution: Modify noisy **labels** via knowledge from early training phase.

Data Perspective

T_i is the **mismatched text**

$$(I_i, T_i, \bar{y}_i)_{i=1}^n \sim p(\mathcal{I}, \mathcal{T}, \bar{\mathcal{Y}})$$

$$\bar{R}_n(f, g) = \frac{1}{n} \sum_{i=1}^n \ell(\hat{y}_i, \bar{y}_i)$$

$$\Rightarrow \bar{R}_n(f, g) = \frac{1}{n} \sum_{i=1}^n \ell(\hat{y}_i', \bar{y}_i),$$

where $\hat{y}_i' = \sigma(\text{sim}(f(I_i), g(\tilde{T}_i)))$

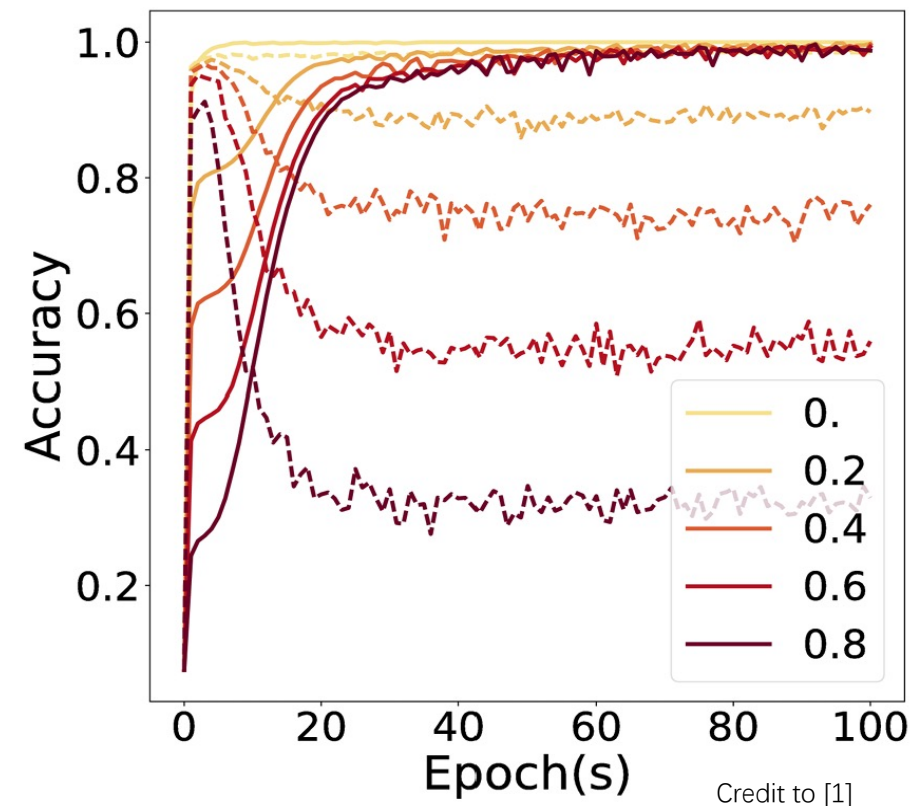
Solution: Modify **textual data** in image-text pairs by generating cleaner data.

Observation: Memorization Effect

Memorization effect refers to that DNNs initially learn the clean patterns within the dataset and then over-fit the noise.

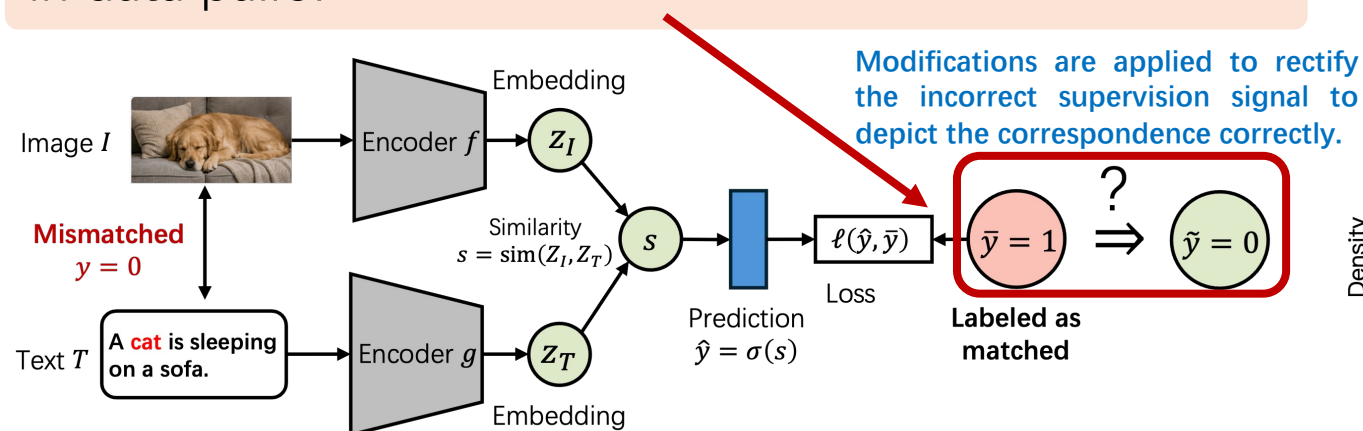
- **Training acc. curves (solid)** increase to fit noisy training data.
- **Test acc. curves (dashed)** first increase to learn patterns of clean data, then decrease to overfit the patterns of noisy data.

Accuracy of classifiers under label noise



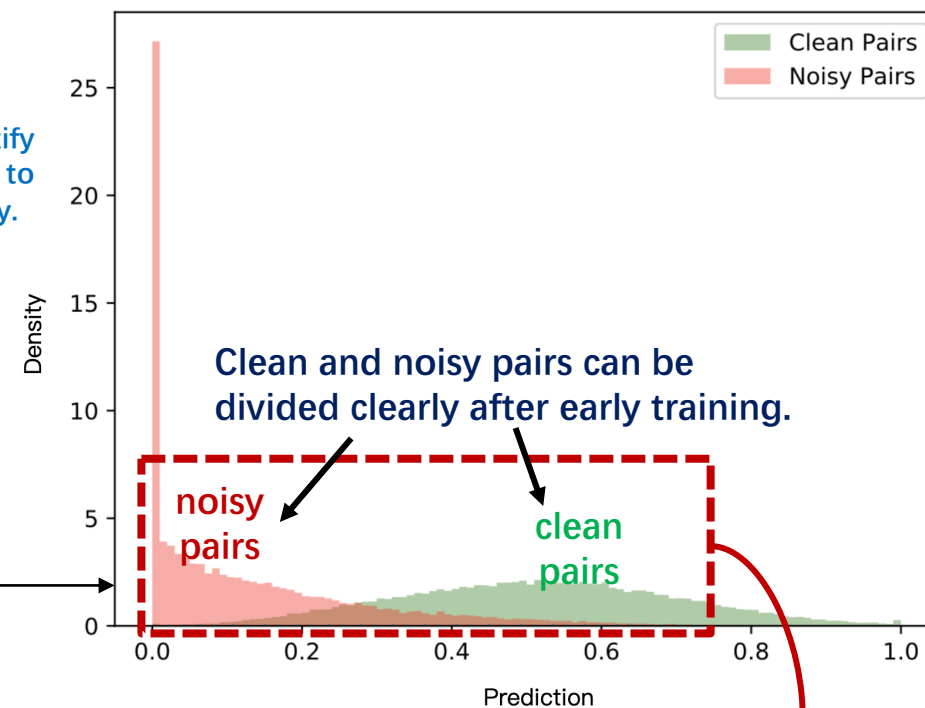
Label Perspective

Problem: How to rectify **noisy correspondence labels** in data pairs?



Observation: Due to the memorization effect, **models fit clean pairs before memorizing noisy pairs**. Thus, per-sample losses are **well-separated** in the early training.

Method: Rectify noisy labels via the knowledge obtained from **early training phase** due to the memorization effect.



$$\bar{R}_n(f, g) = \frac{1}{n} \sum_{i=1}^n \ell(\hat{y}_i, \bar{y}_i)$$

$$\Rightarrow \bar{R}_n(f, g) = \frac{1}{n} \sum_{i=1}^n \ell(\hat{y}_i, \tilde{y}_i)$$

NCR Method

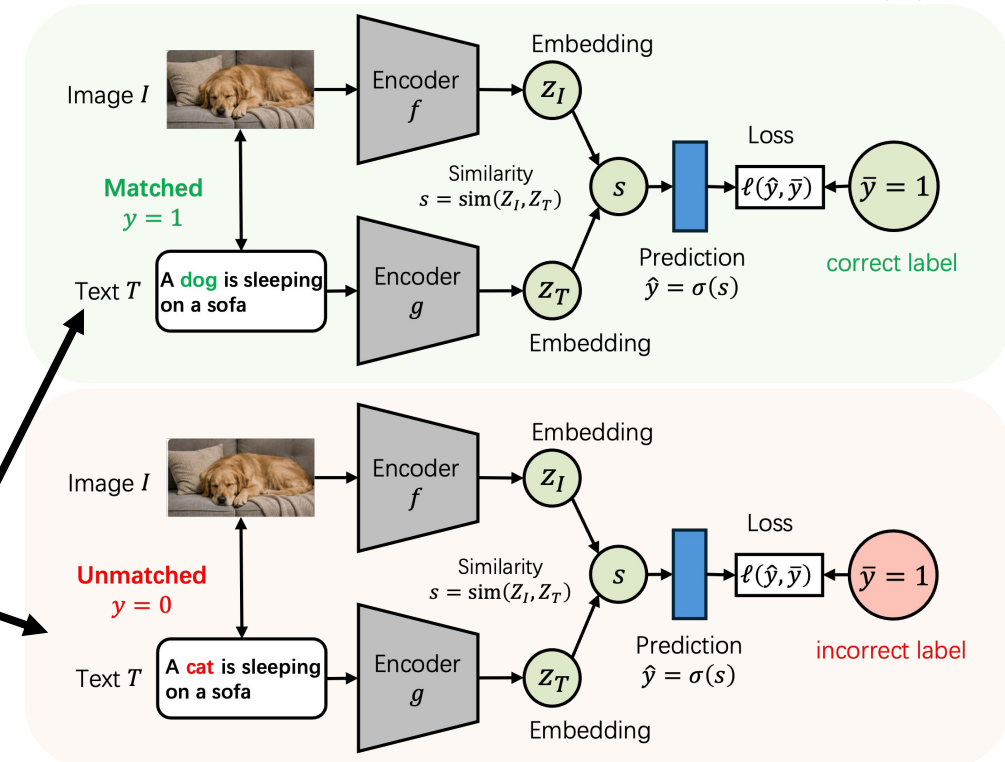
Step 1: Divide data by fitting a Gaussian mixed model on the **per-sample loss** during the *early training phase*.

- **Lower loss** $\Rightarrow$ **clean pairs** (with a prob. $\omega \rightarrow 1$, e.g., 0.9)
- **Larger loss** $\Rightarrow$ **noisy pairs** (with a prob. $\omega \rightarrow 0$, e.g., 0.1)

Step 2: Rectify correspondence labels by a **soft combination** of the **observed label** $\bar{y}_i$ and the **predicted similarity** $S(I, T)$.

$$\hat{y} = \omega \bar{y} + (1 - \omega) S(I, T)$$

- **Clean Pair:** keep the label $\hat{y} \approx \bar{y}$ (e.g., $\omega = 0.9$ or $\omega = 1.0$)
- **Noisy Pair:** Use the prediction $\hat{y} \approx S(I, T)$ (e.g., $\omega = 0.01$)



Step 3: Train with rectified data

NCR uses a triplet loss that pulls the matched similarity $S(I, T)$ above and the hardest negative by a **soft margin** $\hat{\alpha}$.

$$L_{\text{soft}} = [\hat{\alpha} - \underbrace{S(I, T)}_{\text{matched pair similarity}} + \underbrace{S(I, T_h)}_{\text{hardest unmatched text}}]_+ + [\hat{\alpha} - S(I, T) + \underbrace{S(I_h, T)}_{\text{hardest unmatched image}}]_+$$

where I_h and T_h denote the hardest unmatched image and text, $[x]_+ = \max(0, x)$.

Experimental Results

Comparison with CLIP on MS-COCO 5K

Noise Ratio	Methods	Image → Text			Text → Image		
		R@1	R@5	R@10	R@1	R@5	R@10
0%, Zero-Shot	CLIP (ViT-L/14 [†])	58.4	81.5	88.1	37.8	62.4	72.2
	CLIP (ViT-B/32)	50.2	74.6	83.6	30.4	56.0	66.8
	NCR	58.2	84.2	91.5	41.7	71.0	81.3
20%, Fine-tune	CLIP (ViT-B/32)	21.4	49.6	63.3	14.8	37.6	49.6
	NCR	56.9	83.6	91.0	40.6	69.8	80.1
50%, Fine-tune	CLIP (ViT-B/32)	10.9	27.8	38.3	7.8	19.5	26.8
	NCR	53.1	80.7	88.5	37.9	66.6	77.8

- The performance of **CLIP** inevitably degrade after fine-tuning with noisy correspondence.
- NCR** achieve excellent matching performance under noisy correspondence.

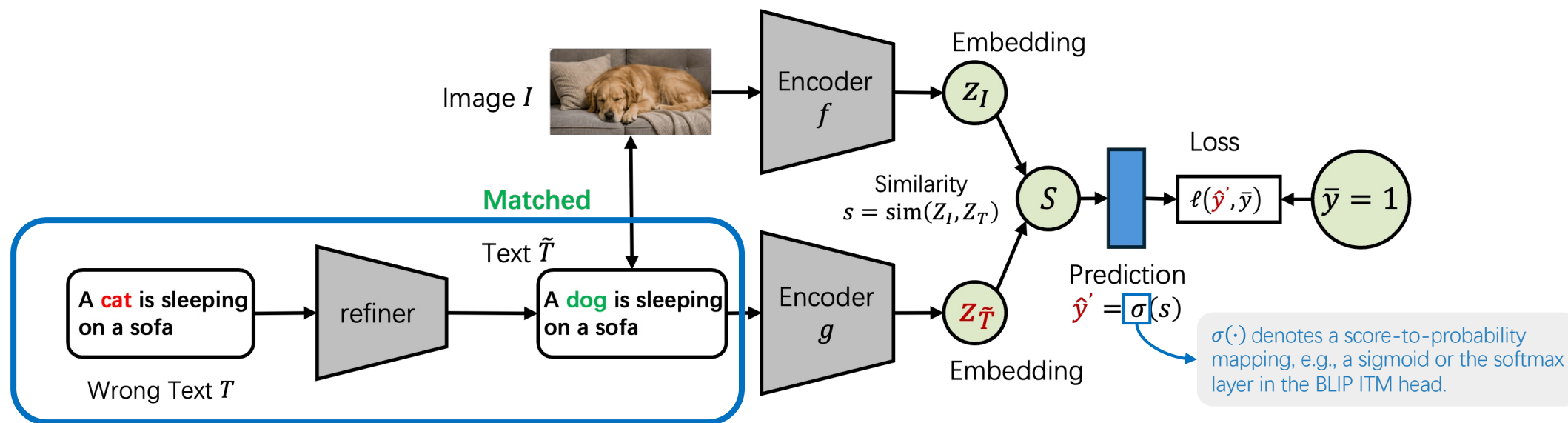
Noisy examples identified by NCR

				
digital art selected for the #	look at him, like it 's no work at all	take a look at this !	share some with your friends !	family walking on a beach

- The images and captions of **the first four pairs** are **completely unrelated**, which can be easily detected by NCR.
- Although the image and caption of **the last pair** are **partially related**, NCR can still detect.

Data Perspective

Motivation: Instead of rectifying the supervision signal, the other way is to refine the **noisy captions** in the image-text pairs.



Method: Refine the noisy captions of the images to form a matched correspondence.

$$\bar{R}_n(f, g) = \frac{1}{n} \sum_{i=1}^n \ell(\hat{y}_i, \bar{y}_i)$$

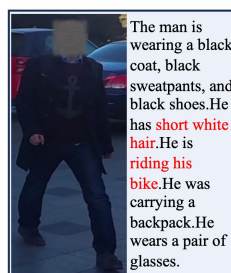
$$\Rightarrow \bar{R}_n(f, g) = \frac{1}{n} \sum_{i=1}^n \ell(\hat{y}'_i, \bar{y}_i), \text{ where } \hat{y}'_i = \sigma(\text{sim}(f(I_i), g(\tilde{T}_i)))$$

BLIP: Bootstrapping Language-Image Pretraining

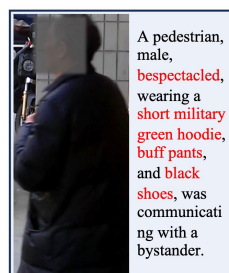
Problem: In practical scenarios, image-text pairs, which are **collected from websites**, are **noisy**. Pretraining vision-language models with such data pairs may achieve suboptimal results.



(a)



(b)



(c)



(d)

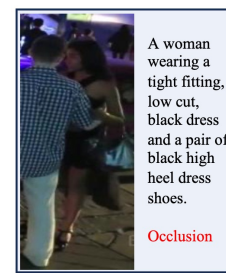


(e)



(f)

(Credits for [1])



(a)



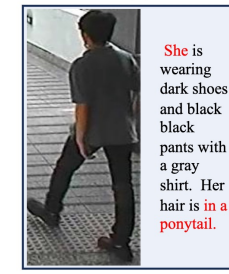
(b)



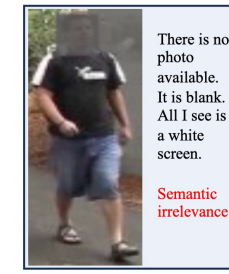
(c)



(d)



(e)



(f)

BLIP addresses noisy web data by **generating cleaner image-text captions** and **training a unified vision-language model**.

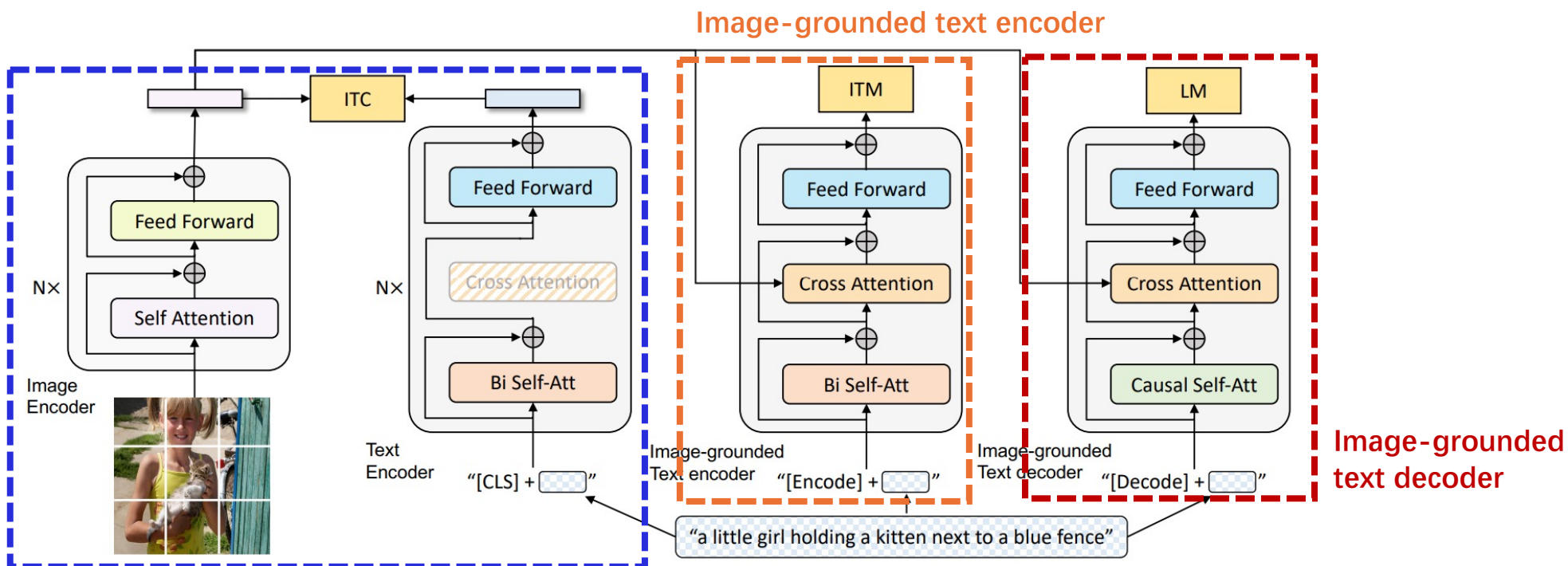
[1] Noisy-correspondence learning for text-to-image person re-identification. In *CVPR*, 2024.

[2] BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, 2022.

Multi-modal Mixture of Encoder-decoder

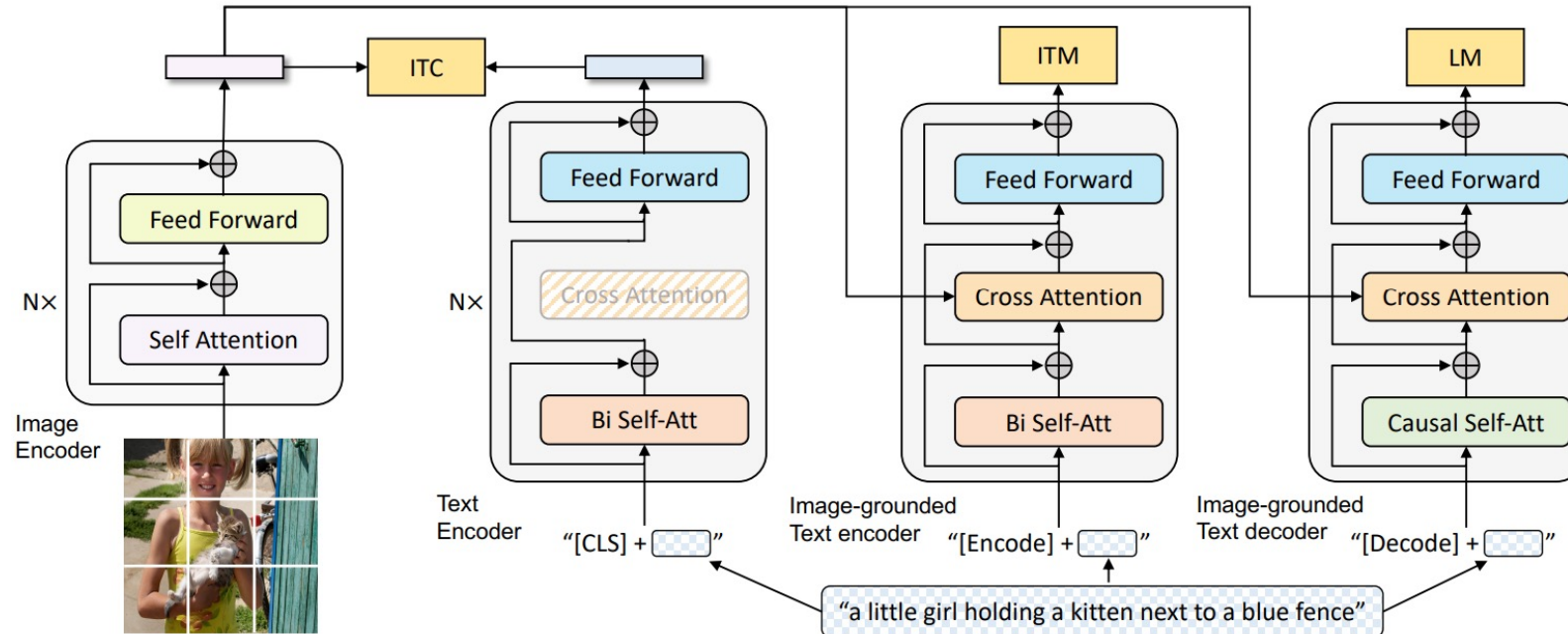
Unimodal encoder:

- Visual encoder
- Textual encoder



- **Unimodal encoder** is trained with an **image-text contrastive (ITC) loss** to align the vision and language representation.
- **Image-grounded text encoder** uses additional cross-attention layers to model vision-language interactions, and is trained with an **image-text matching (ITM) loss** to distinguish between positive and negative image-text pairs.
- **Image-grounded text decoder** is trained with a **language modeling (LM) loss** to generate captions given images.

Pretraining Loss Functions in BLIP

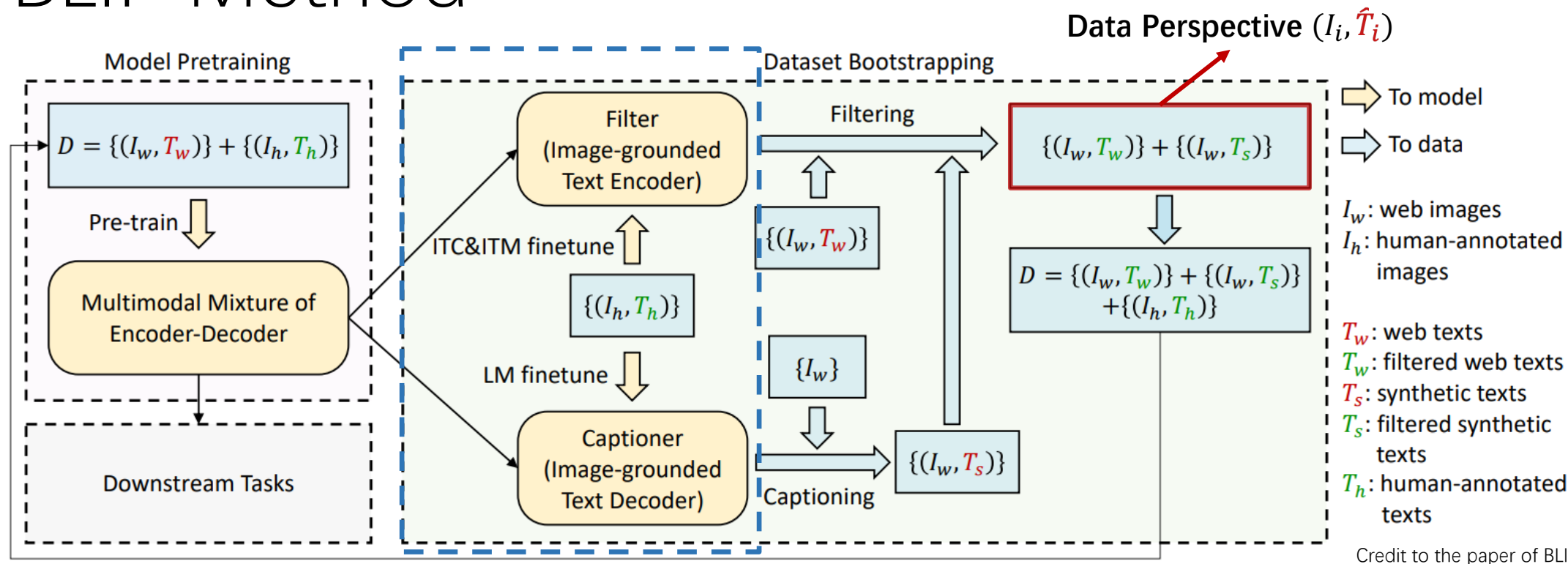


- **Image-Text Contrastive Loss (ITC)** activates the **unimodal encoder**. It aims to **align the feature space** of the visual transformer and the text transformer by **encouraging positive image-text pairs to have similar representations in contrast to the negative pairs**.

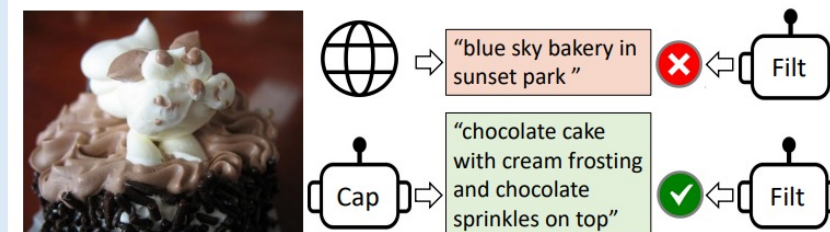
- **Image-Text Matching Loss (ITM)** activates the **image-grounded text encoder**. ITM is a binary classification loss to evaluate **whether the semantic information in images and texts is mutually matched** and aims to learn **fine-grained image-text multimodal representation**.

- **Language Modeling Loss (LM)** activates the **image-grounded text decoder**. LM loss aims to generate textual descriptions given an image by training models to maximize the likelihood of the text in an autoregressive manner.

BLIP Method



- **Captioner** is used to produce synthetic captions for web images and **filter** is used to remove noisy image-text pairs.
- The captioner and filter are initialized from the same pre-trained model and finetuned individually on a small-scale human-annotated dataset.
- The bootstrapped dataset is used to pre-train a new model.

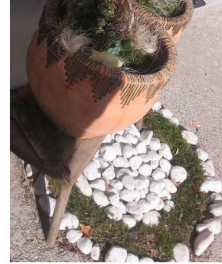


Experimental Results



T_w : "from bridge near my house"

T_s : "a flock of birds flying over a lake at sunset"



T_w : "in front of a house door in Reichenfels, Austria"

T_s : "a potted plant sitting on top of a pile of rocks"



T_w : "the current castle was built in 1180, replacing a 9th century wooden castle"

T_s : "a large building with a lot of windows on it"



T_w : "a week spent at our rented beach house in Sandbridge"

T_s : "an outdoor walkway on a grass covered hill"



T_w : "that's what a sign says over the door"

T_s : "the car is driving past a small old building"



T_w : "hand held through the glass in my front bedroom window"

T_s : "a moon against the night sky with a black background"

- T_w : Web text
- T_s : Synthetic text

- **Green**: Accepted by filter
- **Red**: Rejected by filter

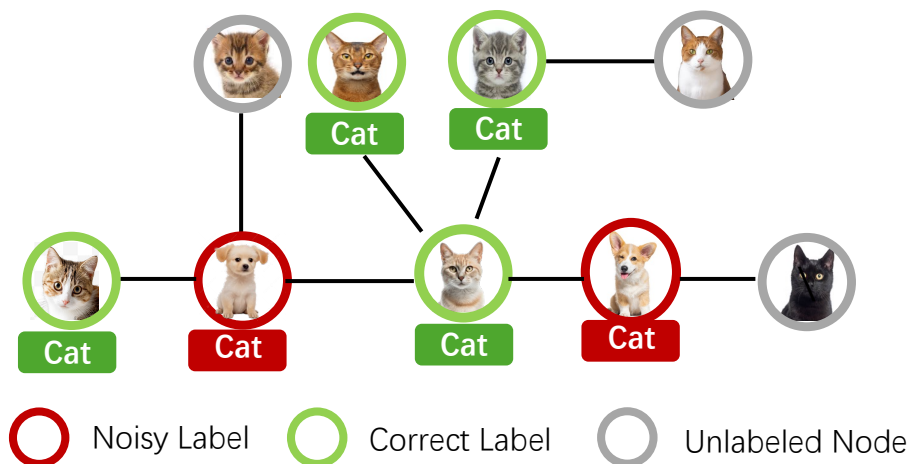
These examples indicate the capabilities of the captioner in generating new textual descriptions and the filter in removing noisy captions from both the original web texts and the synthetic texts.

Part 1 Data-noise Learning

- Part 1.1: Overview of Data-noise Learning
- Part 1.2: Correspondence Noise Learning
- Part 1.3: Graph-noise Learning
- Part 1.4: Group-structured Noise Learning
- Part 1.5: Takeaways

Graph-noise Learning

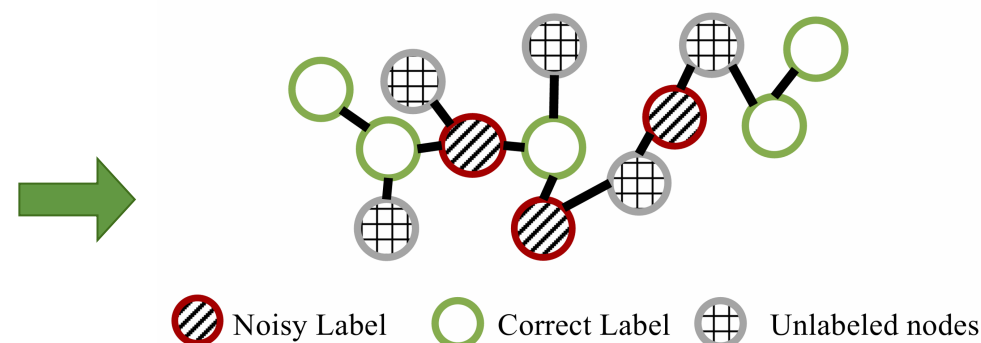
Example:



Inspired by [1]

Observation: In real-world cases, graph nodes (i.e., individuals) may have **incorrect labels** due to **noisy annotations from human annotators** or **weak/ambiguous rules**.

Graph node classification under label noise



Inspired by [1]

Noisy graph node training examples:

$$(G, X, \bar{Y}) \sim p(G, X, \bar{Y}),$$

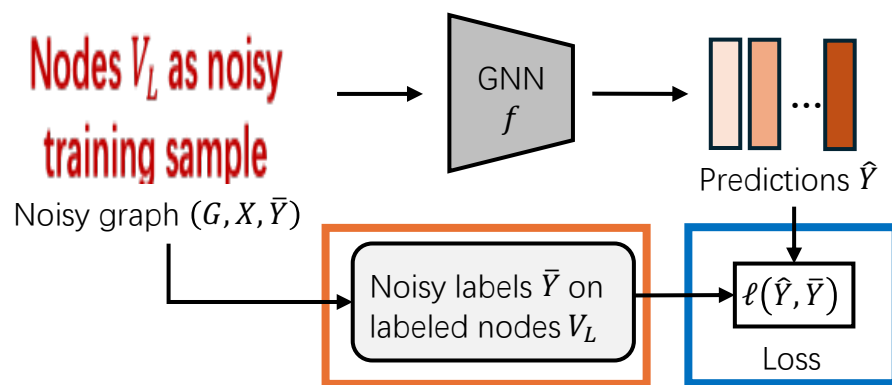
$$\bar{R}_n(f) = \frac{1}{|V_L|} \sum_{i \in V_L} \ell(f(G, X)_i, \bar{y}_i),$$

where $G = (\mathcal{V}, \mathcal{E})$, $\mathcal{V} = \{V_L, V_U\}$ is the union set of labeled nodes V_L and unlabeled nodes V_U , $\mathcal{E}$ is the set of edges, $X = \{x_i\}_{i \in \mathcal{V}}$, and $\bar{Y} = \{\bar{y}_i\}_{i \in V_L}$.

Challenge: Noisy node labels mislead the neighborhood aggregation on graph.

Graph-noise Learning

Graph-noise learning aims at mitigating the negative influence of **noisy node labels** in graph node classification.



Graph-noise Learning

$$(G, X, \bar{Y}) \sim p(G, X, \bar{Y})$$

$$\bar{R}_n(f) = \frac{1}{|V_L|} \sum_{i \in V_L} \ell(f(G, X)_i, \bar{y}_i)$$

Challenge: Noisy node labels mislead the neighborhood aggregation on graph.

Node Perspective
Nodes V_L as noisy training sample

Loss Perspective
Loss $\ell(f(G, X)_i, \bar{y}_i)$ as noisy optimization signal

- From the **node perspective**: **Graph nodes** with clean labels can be identified and prioritized for noise-robust learning.
- From the **loss perspective**: **Loss function** can be modified to regularize graph node embeddings without relying on clean labels.

$$(G, X, \bar{Y}) \sim p(G, X, \bar{Y})$$

$$\bar{R}_n(f) = \frac{1}{|V_L|} \sum_{i \in V_L} \ell(f(G, X)_i, \bar{y}_i)$$

$$\Rightarrow \bar{R}_n(f) = \frac{1}{|V_L|} \sum_{i \in V_L'} \ell(f(G, X)_i, \bar{y}_i)$$

Solution: Modify labeled **nodes** via training with easy nodes first.

$$(G, X, \bar{Y}) \sim p(G, X, \bar{Y})$$

$$\bar{R}_n(f) = \frac{1}{|V_L|} \sum_{i \in V_L} \ell(f(G, X)_i, \bar{y}_i)$$

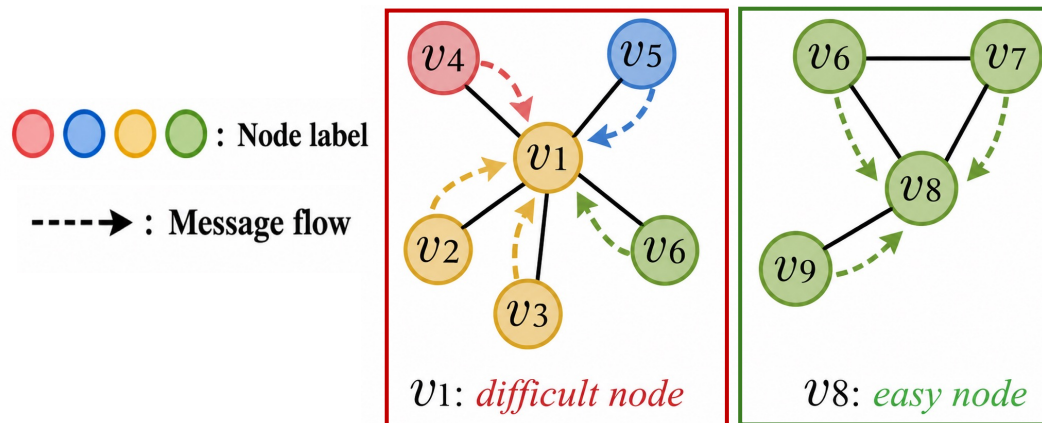
$$\Rightarrow \bar{R}_n(f) = \frac{1}{|V_L|} \sum_{i \in V_L} \ell(f(G, X)_i, \bar{y}_i) + \ell_{\text{reg}}(G, X)$$

Solution: Modify **loss function** by employing a graph contrastive regularization.

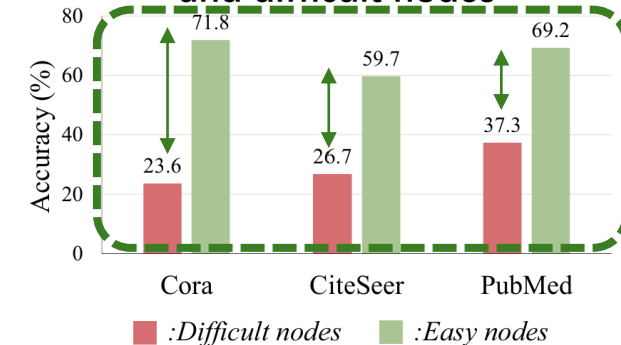
Node Perspective

Problem: Noisy nodes can corrupt neighborhood aggregation in noisy graphs. How can we identify the nodes with clean labels for noise-robust training?

- Observations**
- Nodes with **high neighborhood diversity** are treated as **difficult nodes**, whereas those with **lower diversity** are treated as **easy nodes**.
 - Training with **easy nodes** achieves better performance than **difficult nodes**, indicating **the cleanness of easy nodes**.



Performance gap between training with easy and difficult nodes



Credit to the paper of CLNode

Method: Measure node difficulty based on **neighborhood diversity and feature similarity** and achieve noise-robust learning via training with **easy nodes** first.

$$\bar{R}_n(f) = \frac{1}{|V_L|} \sum_{i \in V_L} \ell(f(G, X)_i, \bar{y}_i)$$

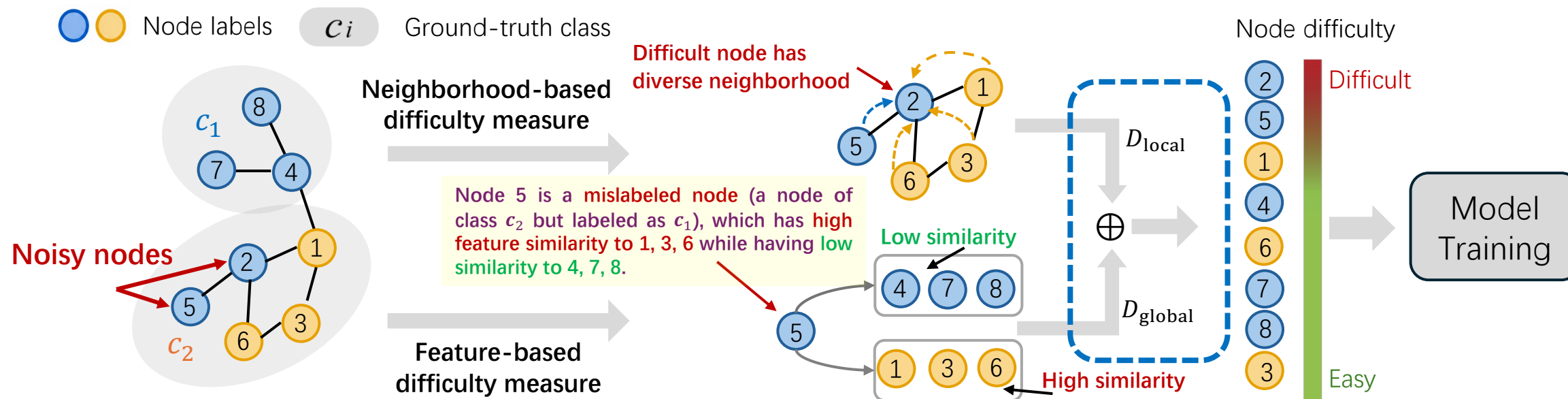
noisy nodes

$$\Rightarrow \bar{R}_n(f) = \frac{1}{|V'_L|} \sum_{i \in V'_L} \ell(f(G, X)_i, \bar{y}_i)$$

identified nodes

CLNode Method

Evaluate nodes with **neighborhood and feature-based difficult measures simultaneously**.



Neighborhood-based difficulty measure

- Leverage an entropy-based metric D_{local} to evaluate how diverse the local neighborhood of the node is.
- Higher entropy means **more diverse neighborhood**, while low entropy indicates **less diverse neighborhood**.

Feature-based difficulty measure

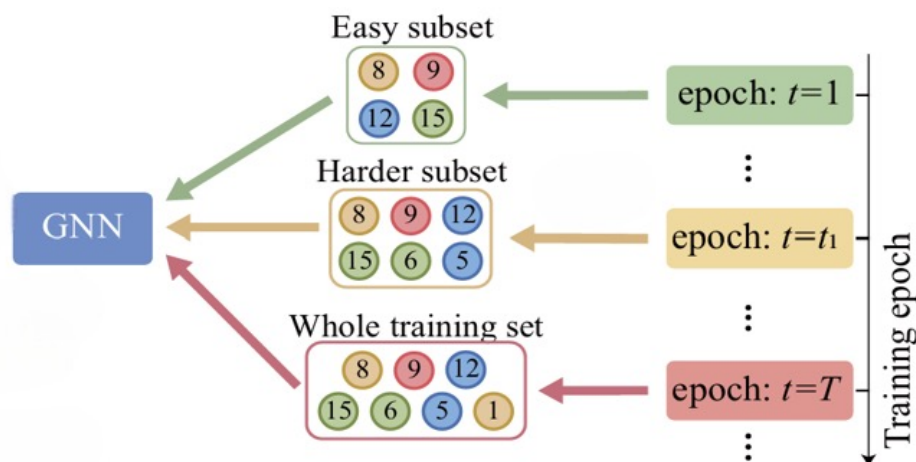
- Leverage a similarity-based metric D_{global} to evaluate how similar the node to the same-class nodes is.
- **Mislabeled nodes** are **dissimilar to the nodes with the same label**, while are **similar to the nodes in the class where they are located**.

CLNode Method

CLNode trains graph neural network (GNN) by employing an **easy-to-hard curriculum strategy**.

Sort and select training nodes V'_L based on difficulty

- **Easy-to-hard curriculum strategy**
- Use **pacing function** $g(t)$ to control the learning pace. Specifically, **linear pacing** for **uniform** learning progress; **root pacing** for **learning hard nodes early**; **geometric pacing** for **more time to learn easy nodes**.



Credit to the paper of CLNode

Pacing function $g(t)$ controls the learning process.

Linear pacing:

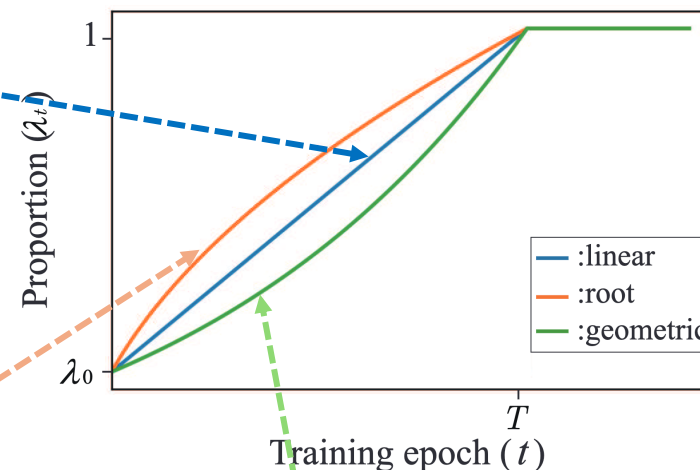
$$g(t) = \min\left(1, \lambda_0 + (1 - \lambda_0) * \frac{t}{T}\right)$$

- Increase the proportion of nodes **uniformly**.
- Add harder nodes at a constant rate

Root pacing:

$$g(t) = \min\left(1, \sqrt{\lambda_0^2 + (1 - \lambda_0^2) * \frac{t}{T}}\right)$$

- Introduce **harder nodes earlier**
- Fast increase at the begging, then slow down



Geometric pacing:

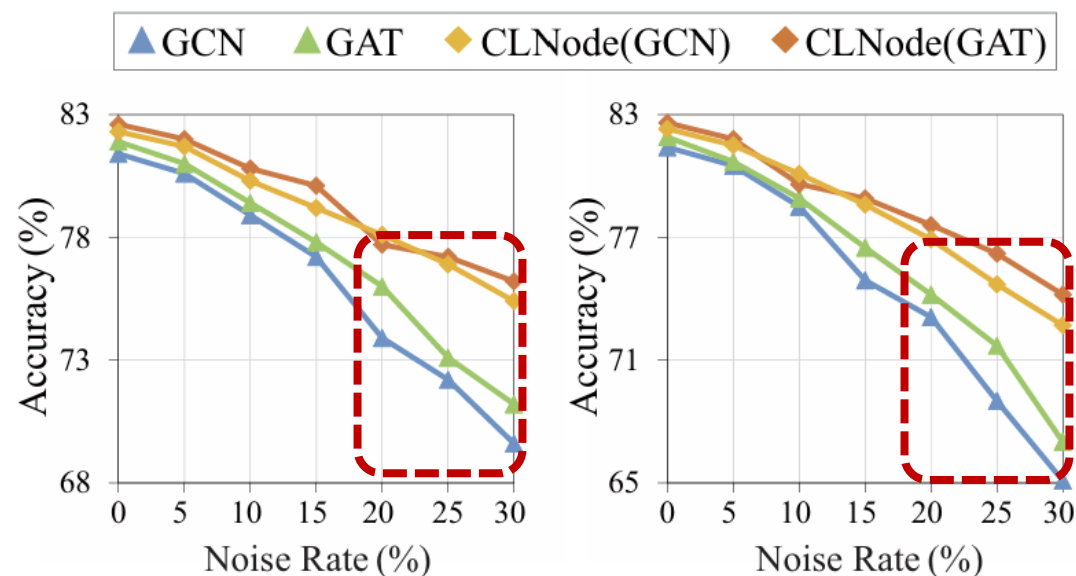
$$g(t) = \min\left(1, 2^{\log_2 \lambda_0 - \log_2 \lambda_0 * \frac{t}{T}}\right)$$

- Spend **more epochs on easy nodes**
- Slow start, then rapid increase near the end

Experimental Results

Noise robustness experiments

Credit to the paper of CLNode



As the noise rate in graph nodes increases, **the performance gap** between CLNode and its corresponding baseline becomes **larger**.

Ablation experiments

Credit to the paper of CLNode

Method		Cora	CiteSeer	PubMed
GCN	original	69.6	55.3	69.4
	+CLNode(local)	74.8	61.8	74.2
	+CLNode(global)	72.3	62.5	73.2
	+CLNode	75.4	63.1	74.4
Pacing Function		Cora	CiteSeer	PubMed
CLNode	linear	74.8	62.7	74.2
	root	74.5	62.5	73.9
	geometric	75.4	63.1	74.4

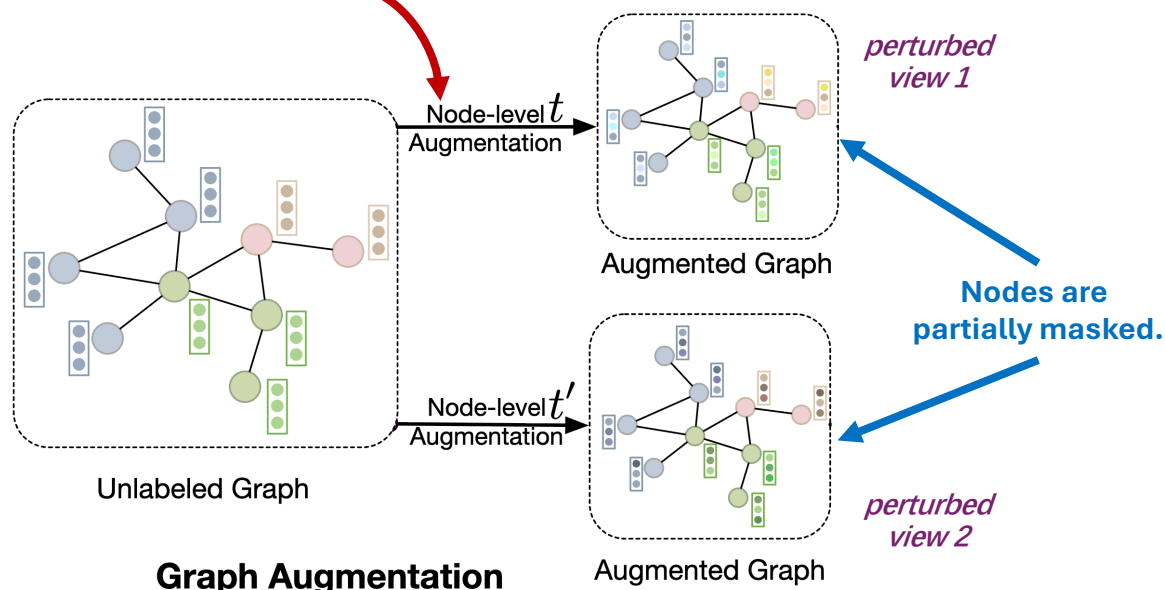
CLNode achieves the best results by combining both **local and global difficulty** and utilizing **geometric pace function**.

Loss Perspective

Problem: The main challenge of graph-noise learning is **lacking the correct supervision signals**. If supervision signals are not reliable, are there any other effective signals?

Motivation:

- **Graph augmentation** changes the graph but **preserves the underlying semantics**.
- The representations of two perturbed views of one node should be consistent, which provide **a label-free contrastive signal** to regularize the node representations.

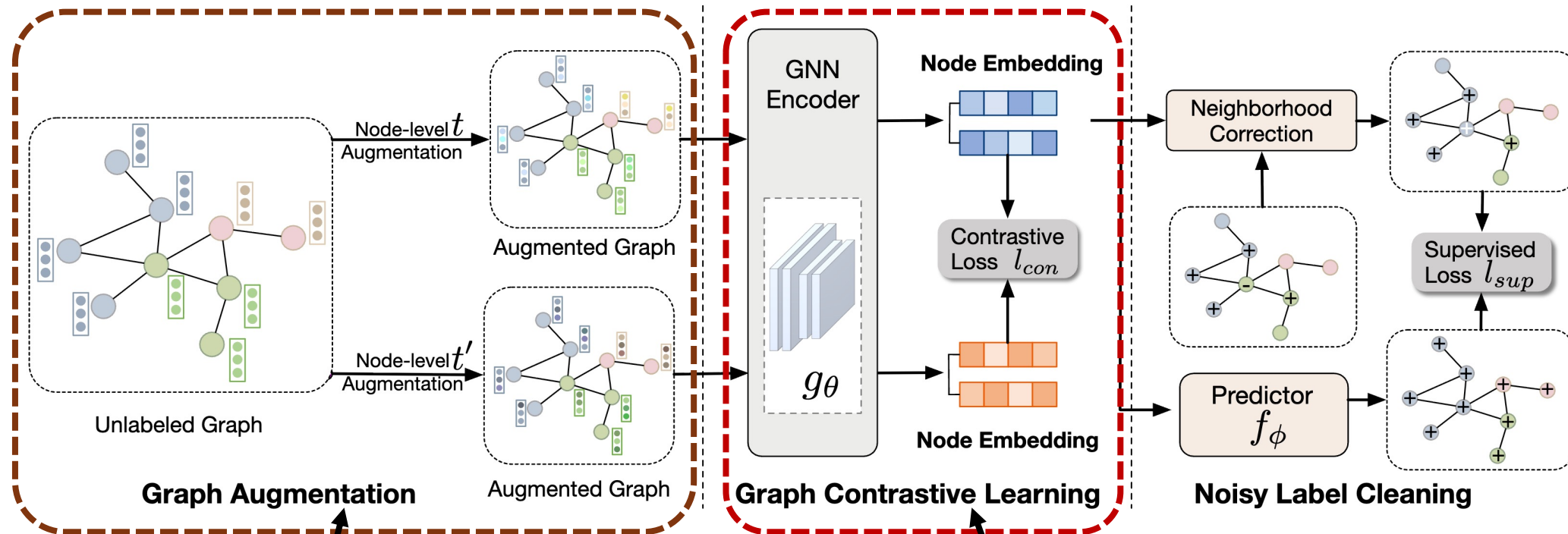


Method: Enhance the robustness to noise by employing **graph contrastive learning** with a label-free regularization term.

$$\bar{R}_n(f) = \frac{1}{|V_L|} \sum_{i \in V_L} \ell(f(G, X)_i, \bar{y}_i)$$

$$\Rightarrow \bar{R}_n(f) = \frac{1}{|V_L|} \sum_{i \in V_L} \ell(f(G, X)_i, \bar{y}_i) + \ell_{\text{reg}}(G, X)$$

CGNN Method



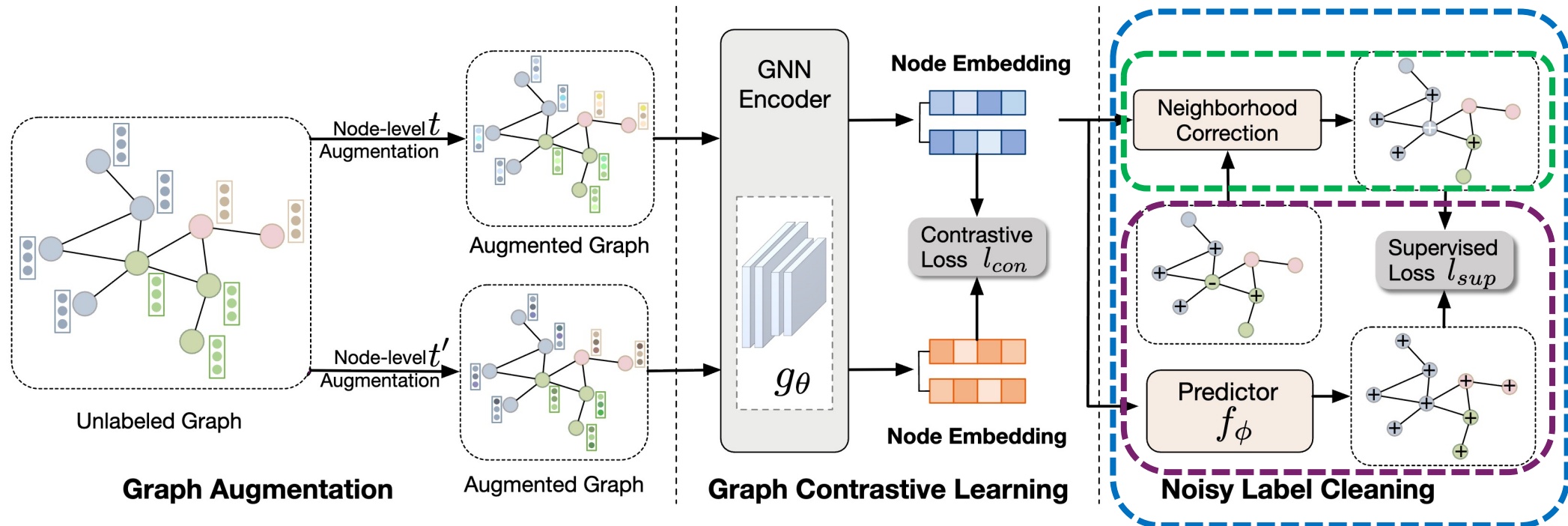
Step 1: Graph augmentation is applied for two perturbed views:

- Randomly drop edges
- Randomly mask attributes

Step 2: Contrastive loss ℓ_{con}

- Learning perturbation-invariant node embeddings via comparing two perturbed views.

CGNN Method



Step 3: Supervised loss ℓ_{sup}

- Step 3.1 Neighborhood Correction:** (1) Predict pseudo labels for all nodes and majority vote the most common labels; (2) Compute the proportion whose embeddings are truly similar to the node to evaluate the trustworthiness of the prediction; (3) Change the label with the majority label only if it differs from the original one and with high trust region.
- Step 3.2 Supervised Loss:** Calculate the supervised loss with the **rectified labels** and the **predictions**.

Update the model: $\mathcal{L} = \ell_{con} + \ell_{sup}$

Experimental Results

Noise robustness experiments

Credit to the paper of CGNN

Dataset	Noise	GCN	GIN	Forward	Coteaching+	D-GNN	CP	Ours
Co-author CS	Uniform Pair	80.3 $\pm$ 1.4	80.5 $\pm$ 1.2	81.3 $\pm$ 0.4	80.7 $\pm$ 1.4	81.9 $\pm$ 1.1	82.5 $\pm$ 1.2	84.1 $\pm$0.4
		79.5 $\pm$ 1.1	79.3 $\pm$ 1.7	80.5 $\pm$ 0.8	77.6 $\pm$ 3.3	79.9 $\pm$ 0.7	80.2 $\pm$ 1.0	81.0 $\pm$1.1
Amazon Photo	Uniform Pair	82.2 $\pm$ 0.9	82.1 $\pm$ 0.4	79.8 $\pm$ 0.6	78.5 $\pm$ 0.6	84.5 $\pm$ 3.4	83.7 $\pm$ 0.7	85.3 $\pm$0.9
		80.9 $\pm$ 1.2	81.2 $\pm$ 1.5	78.7 $\pm$ 0.3	75.5 $\pm$ 1.8	83.2 $\pm$ 2.2	83.1 $\pm$ 1.1	85.1 $\pm$0.7
Amazon Computers	Uniform Pair	76.9 $\pm$ 1.2	76.4 $\pm$ 0.7	78.4 $\pm$ 0.4	70.3 $\pm$ 3.6	81.2 $\pm$ 0.6	80.6 $\pm$ 1.3	81.8 $\pm$0.3
		76.3 $\pm$ 1.6	76.8 $\pm$ 1.2	77.3 $\pm$ 0.6	60.9 $\pm$ 3.9	81.3 $\pm$ 1.0	80.4 $\pm$ 2.2	81.6 $\pm$0.2

Achieve better performance with small deviation

Ablation study on Amazon Photo dataset

Credit to the paper of CGNN

Noise	CGNN w/o contr	CGNN w/o corr	Full Model
Uniform Pair	84.4 $\pm$ 0.6	85.1 $\pm$ 0.9	85.3 $\pm$0.9
	84.3 $\pm$ 0.7	85.0 $\pm$ 0.9	85.1 $\pm$0.7

The performance of CGNN w/o contrastive learning loss l_{con} will drop a lot, **indicating the importance of its regularization role.**

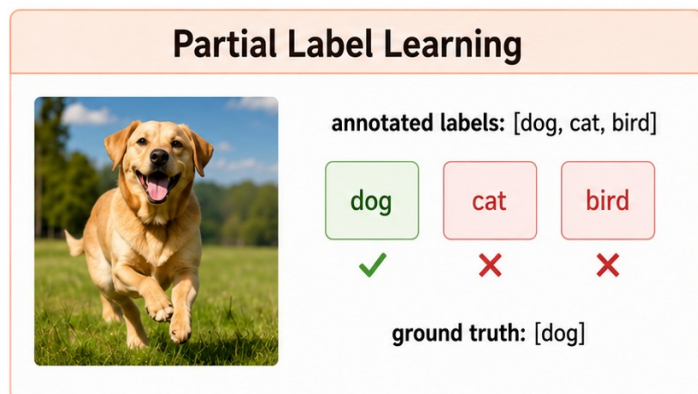
Part 1 Data-noise Learning

- Part 1.1: Overview of Data-noise Learning
- Part 1.2: Correspondence Noise Learning
- Part 1.3: Graph-noise Learning
- Part 1.4: Group-structured Noise Learning
- Part 1.5: Takeaways

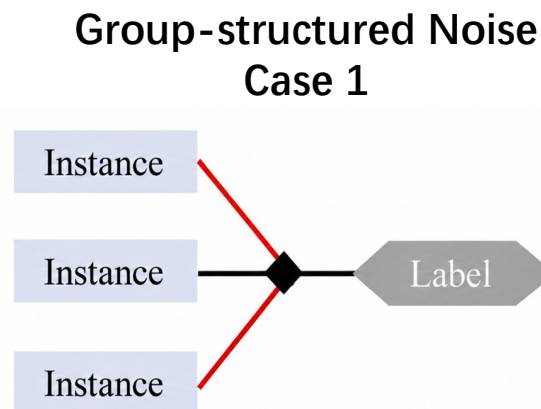
Group-structured Noise Learning



Generated by ChatGPT



Generated by ChatGPT



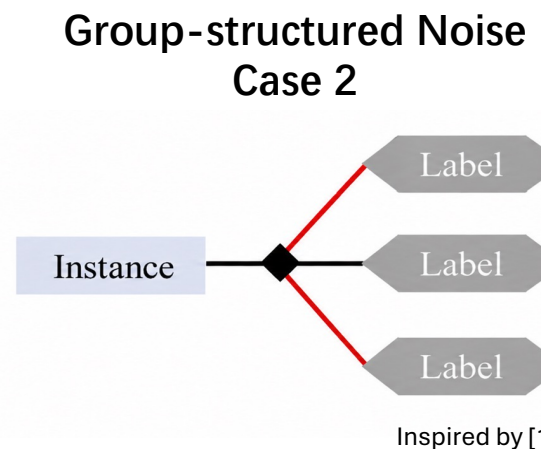
Multiple Instance Learning (MIL):

$$(\bar{X}_i, y_i)_{i=1}^n \sim p(\bar{X}, y)$$

$$\bar{R}_n(f) = \frac{1}{n} \sum_{i=1}^n \ell(f(\bar{X}_i), y_i),$$

where $f(\bar{X}_i) = \{f(\bar{x}_{i1}), \dots, f(\bar{x}_{iK})\}$

Challenge: Models can **hardly identify** positive instances from **bag-level supervision**.



Inspired by [1]

Partial Label Learning (PLL):

$$(x_i, \bar{Y}_i)_{i=1}^n \sim p(X, \bar{Y})$$

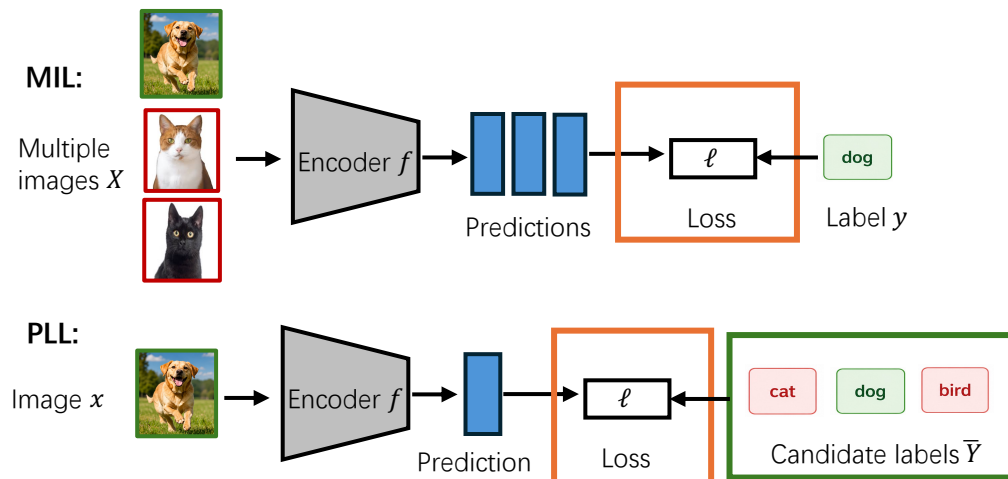
$$\bar{R}_n(f) = \frac{1}{n} \sum_{i=1}^n \ell(f(x_i), \bar{Y}_i),$$

where $\bar{Y}_i = \{\bar{y}_{i1}, \dots, \bar{y}_{iK}\}$

Challenge: Models can **hardly learn** true labels from **multiple candidates**.

Group-structured Noise Learning

Group-structured noise learning aims at learning a proper model that predicts accurate instance-level labels under **coarse-grained** supervision.



Group-structured Noise Learning

MIL: $(\bar{X}_i, y_i)_{i=1}^n \sim p(\bar{X}, y)$
 $\bar{R}_n(f) = \frac{1}{n} \sum_{i=1}^n \ell(f(\bar{X}_i), y_i)$

PLL: $(x_i, \bar{Y}_i)_{i=1}^n \sim p(x, \bar{Y})$
 $\bar{R}_n(f) = \frac{1}{n} \sum_{i=1}^n \ell(f(x_i), \bar{Y}_i)$

Challenge: Models can hardly identify true instance-label pairs with coarse labels.

Loss Perspective

Loss $\ell(f(\bar{X}_i), g(y_i))$ as
noisy optimization signal

Label Perspective

Labels $\bar{Y}_i$ as **noisy**
supervision target

- From the **loss perspective (MIL)**: **Loss functions** are modified to guide the classifier to converge to optimal one without noise.
- From the **label perspective (PLL)**: **True labels** can be identified from all candidate labels to achieve noise-robust learning.

$$(\bar{X}_i, y_i)_{i=1}^n \sim p(\bar{X}, y)$$

$$\bar{R}_n(f, g) = \frac{1}{n} \sum_{i=1}^n \ell(f(\bar{X}_i), g(y_i))$$

$$\Rightarrow \bar{R}_n(f, g) = \frac{1}{n} \sum_{i=1}^n \ell_{\text{robust}}(f(\bar{X}_i), g(y_i))$$

Solution: Modify **loss function** by selecting the best-matched object in the group

$$(x_i, \bar{Y}_i)_{i=1}^n \sim p(x, \bar{Y})$$

$$\bar{R}_n(f) = \frac{1}{n} \sum_{i=1}^n \ell(f(x_i), \bar{Y}_i)$$

$$\Rightarrow \bar{R}_n(f) = \frac{1}{n} \sum_{i=1}^n \ell(f(x_i), \hat{Y}_i)$$

Solution: Modify noisy **labels** via smoothness in the learned representation space

Loss Perspective

Problem: How to design an optimal **loss** to guide models to bridge a **connection between the label and the correct sample**?

Motivation:

- **Maximizing the log-likelihood** of $(\bar{X}_i, y_i)$ guides the models f, g to converge to the correct sample regarding the label.
- The log-likelihood of $(\bar{X}_i, y_i)$ is lower bounded by two components: **pair matching** term and **group weighting** term.

$$\log \hat{P}(\bar{X}_i, y_i | f, g) \geq \underbrace{\max_{\bar{x}_{ik} \in \bar{X}_i} \log \hat{P}(y_i | \bar{x}_{ik}; f, g)}_{\text{Select the best matched case}} + \underbrace{\log \hat{P}(\bar{x}_{ik} | \bar{X}_i; f, g)}_{\text{Pair Matching}} + \underbrace{\log \hat{P}(\bar{x}_{ik} | \bar{X}_i; f, g)}_{\text{Group Weighting}}$$

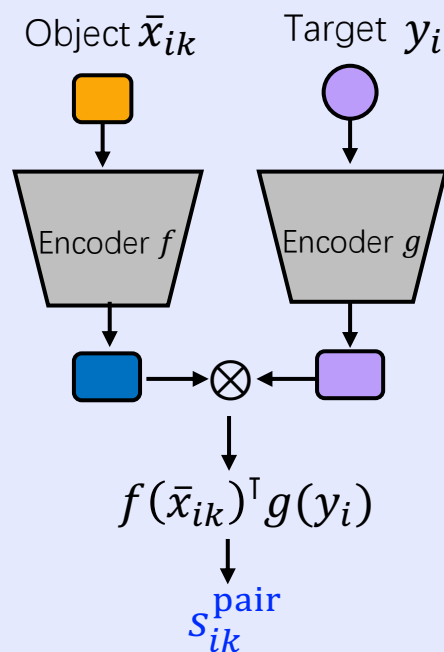
Method: Obtain a noise-robust loss function by selecting the best-matched object in the group.

- **Pair Matching:** score how well each candidate object matches the observed target
- **Group Weighting:** score how representative each object is within the group
- **Selection:** keep the best-matched object based on scores

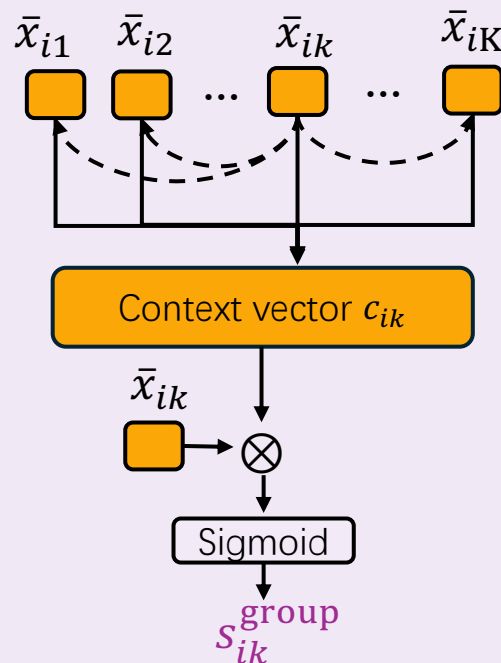
$$\begin{aligned} & (\bar{X}_i, y_i)_{i=1}^n \sim p(\bar{X}, y) \\ & \bar{R}_n(f, g) = \frac{1}{n} \sum_{i=1}^n \ell(f(\bar{X}_i), g(y_i)) \quad \text{original loss} \\ \Rightarrow & \bar{R}_n(f, g) = \frac{1}{n} \sum_{i=1}^n \ell_{\text{robust}}(f(\bar{X}_i), g(y_i)) \quad \text{noise-robust loss} \end{aligned}$$

Max-Matching Method

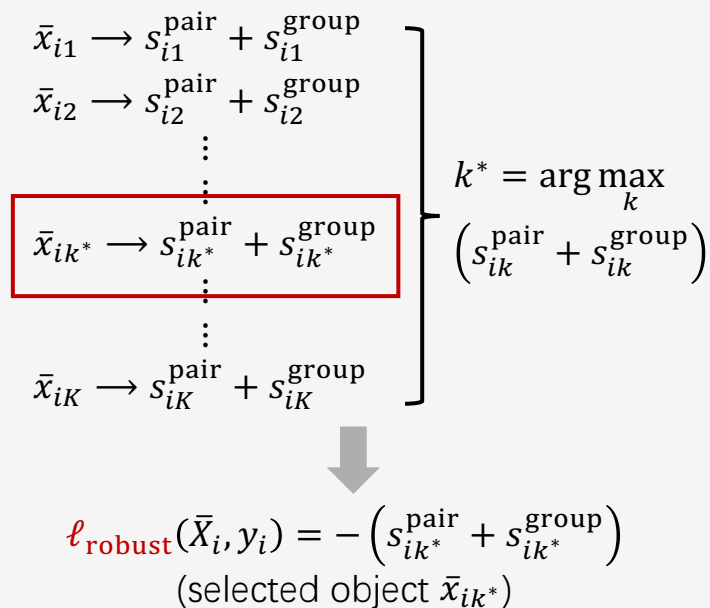
Pair matching



Group weighting



Selection and loss



Step 1: Pair matching

Compute **pair-level score** s_{ik}^{pair} by **the normalized similarity** between **the object** $\bar{x}_{ik}$ and **the target** y_i .

Step 2: Group weighting

Estimate **group-level score** s_{ik}^{group} by a **transformed similarity** between **the object** $\bar{x}_{ik}$ and **its context vector** c_{ik} .

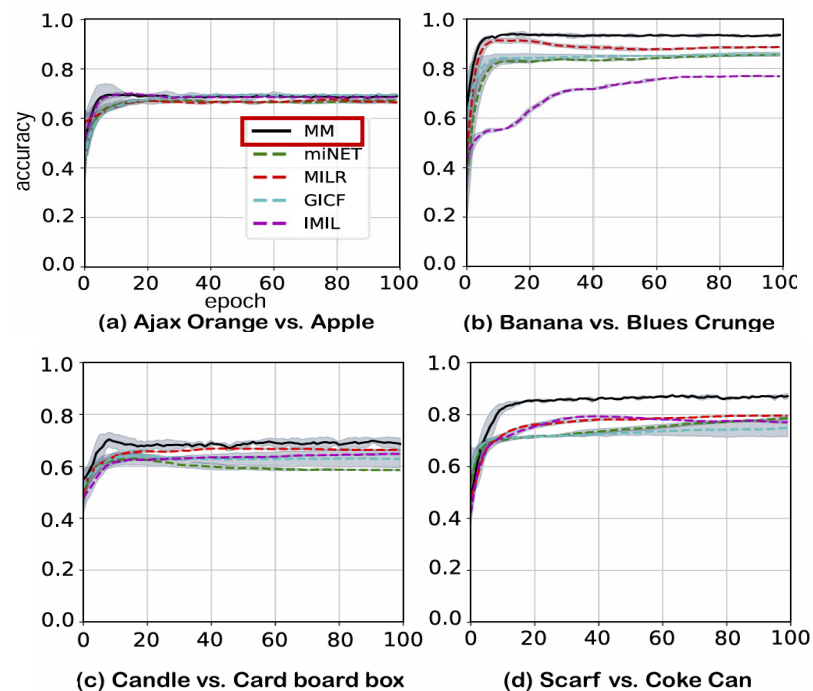
Step 3: Selection and loss

Select the **best-matched object** $\bar{x}_{ik^*}$ based on scores, and then optimize with **the robust loss** ℓ_{robust} .

Experimental Results

Experiments on multiple instance learning (MIL)

Credit to the paper of Max-Matching



Max-Matching (MM) achieves superior performance over the **MIL baselines**.

Experiments on partial label learning (PLL)

Credit to the paper of Max-Matching

	<i>FG-NET</i>	<i>MSRC v2</i>	<i>Bird Song</i>	<i>Yahoo! News</i>	<i>Lost</i>
PLKNN	0.059 ± 0.005	0.446 ± 0.007	0.613 ± 0.004	0.426 ± 0.004	0.428 ± 0.003
PLSVM	0.064 ± 0.013	0.475 ± 0.008	0.625 ± 0.019	0.629 ± 0.012	0.801 ± 0.025
CLPL	0.065 ± 0.029	0.480 ± 0.015	0.628 ± 0.012	0.537 ± 0.017	0.793 ± 0.022
PALOC	0.054 ± 0.005	0.463 ± 0.011	0.598 ± 0.020	0.434 ± 0.0013	0.546 ± 0.007
Max-Matching	0.110 ± 0.021	0.517 ± 0.007	0.642 ± 0.010	0.647 ± 0.005	0.823 ± 0.025

Max-Matching has the best performance among **PLL baselines**.

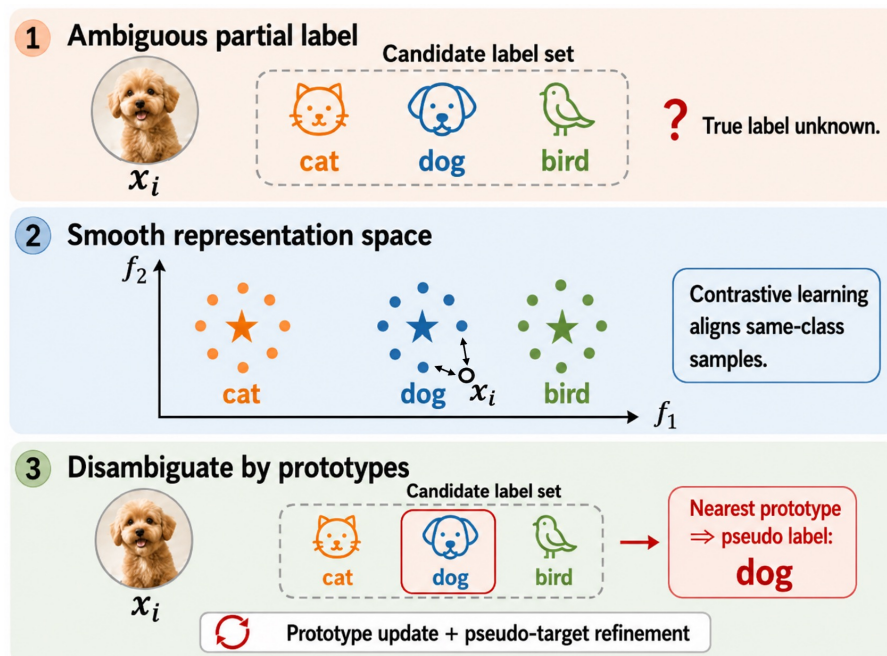
Label Perspective

Problem: How to identify the ground truth label from candidate labels in partial label learning?

Motivation: **Contrastive learning** can **improve the smoothness in the learned representation space**, where similar samples are close to each other.

Method:

- **Contrastive learning:** align representations of positive pairs from same classes to **enhance smoothness**.
- **Prototype-based label disambiguation:** leverage the smoothness in the learned representation space to update prototype and pseudo target.



$$\mathcal{L}_{\text{cont}} = -\log \frac{\exp(q^\top k_+ / \tau)}{\sum_{k' \in A(x)} \exp(q^\top k' / \tau)},$$

where q is the embedding of x , and k_+ denotes the positive sample.

- **Maximize** the similarity between **positive pairs**
- **Minimize** the similarity between **negative pairs**

$$\begin{aligned} (x_i, \bar{y}_i)_{i=1}^n &\sim p(\mathcal{X}, \bar{\mathcal{Y}}) \\ \bar{R}_n(f) &= \frac{1}{n} \sum_{i=1}^n \ell(f(x_i), \bar{y}_i), \text{ where } \bar{y}_i = \{\bar{y}_{i1}, \dots, \bar{y}_{ik}\} \\ \Rightarrow \bar{R}_n(f) &= \frac{1}{n} \sum_{i=1}^n \ell(f(x_i), \hat{y}_i), \text{ where } \hat{y}_i = \{\hat{y}_{i1}, \dots, \hat{y}_{ik}\} \end{aligned}$$

PiCO Method

Data Augmentation

Perform data augmentations on images for positive and negative data pairs.

Learning with disambiguated labels

- Use **classifier** for positive sample selection;
- Use **contrastive learned prototypes** for label update.

- The classifier perform classification on the inputs: $\mathbf{q} \in \mathbf{c}$.

- Updating **uniformly initialized** class prototype μ_c :

$$\mu_c = \text{Normalize}(\gamma \mu_c + (1 - \gamma) \mathbf{q})$$

example predicted as class c

- Updating pseudo target $\mathbf{s}_i$:

$$\mathbf{s}_i = \phi \mathbf{s}_i + (1 - \phi) \mathbf{z}$$

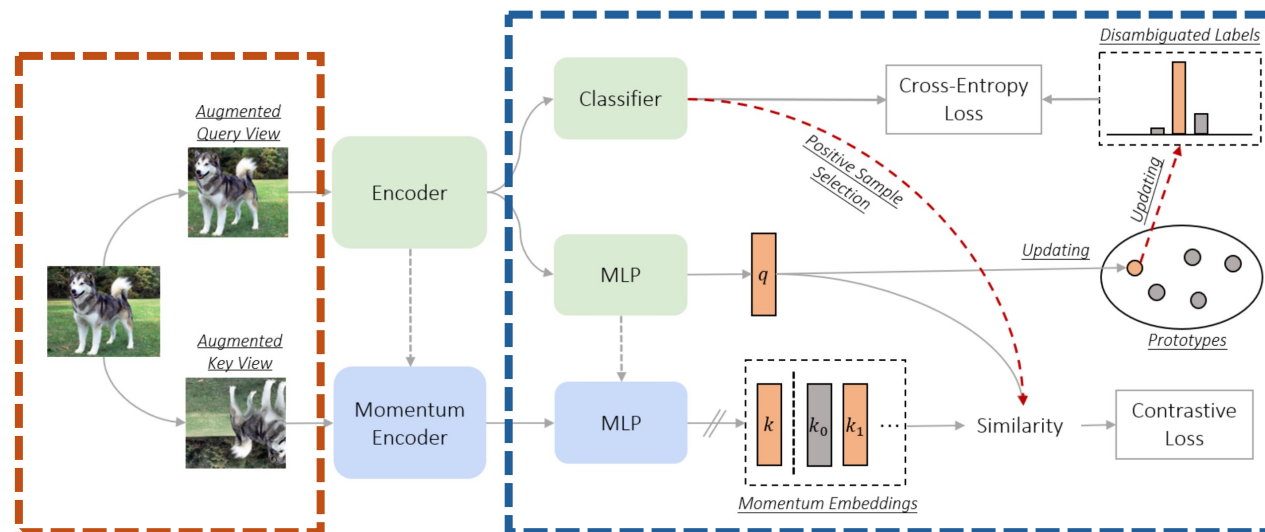
where $\mathbf{z} = \text{onehot}(\arg \max_{c \in \hat{\mathcal{Y}}_i} \mu_c^\top \mathbf{q})$

label assigned by the nearest prototype

- Train the model with a surrogate cross-entropy loss and contrastive loss:

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \lambda \mathcal{L}_{\text{cont}} = \ell(f(x_i), \mathbf{s}_i) + \lambda \mathcal{L}_{\text{cont}}$$

surrogate CE loss



- Contrastive learning** is applied to learn clustered representations without the noisy information of labels
- Prototypes** are leveraged to **cluster samples shared the same core representations**, indicating the **class clusters**.
- Use the similarity between samples and prototypes to **re-assign pseudo labels** to samples for supervised learning.

Theoretical justification: from Expectation-Maximization view
 contrastive learning (E step)  label disambiguation (M step)

Experimental Results

Comparison on CIFAR-10, where $q = P(\bar{y} \in \bar{Y} \mid \bar{y} \neq y)$, y is the ground truth label

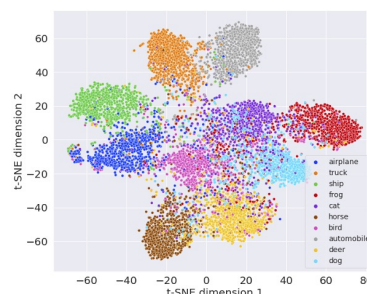
Credit to the paper of PiCO

Dataset	Method	$q = 0.1$	$q = 0.3$	$q = 0.5$
CIFAR-10	PiCO (ours)	$94.39 \pm 0.18\%$	$94.18 \pm 0.12\%$	$93.58 \pm 0.06\%$
	LWS	$90.30 \pm 0.60\%$	$88.99 \pm 1.43\%$	$86.16 \pm 0.85\%$
	PRODEN	$90.24 \pm 0.32\%$	$89.38 \pm 0.31\%$	$87.78 \pm 0.07\%$
	CC	$82.30 \pm 0.21\%$	$79.08 \pm 0.07\%$	$74.05 \pm 0.35\%$
	MSE	$79.97 \pm 0.45\%$	$75.64 \pm 0.28\%$	$67.09 \pm 0.66\%$
	EXP	$79.23 \pm 0.10\%$	$75.79 \pm 0.21\%$	$70.34 \pm 1.32\%$
	Fully Supervised	$94.91 \pm 0.07\%$		

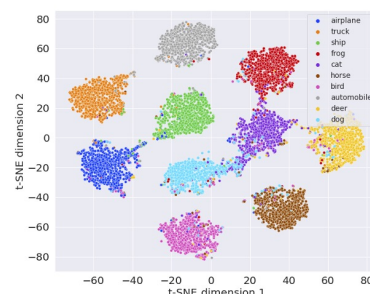
PiCO achieves results that are **comparable to fully supervised learning model**, showing that the label disambiguation from candidate labels is sufficiently accomplished.

T-SNE visualization of the image features on CIFAR-10 with $q = 0.5$

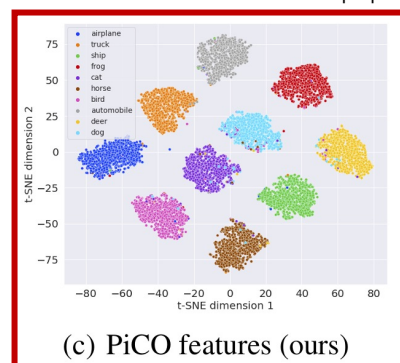
Credit to the paper of PiCO



(a) Uniform features



(b) PRODEN features



(c) PiCO features (ours)

PiCO can produce **well-separated** clusters and significantly enhance the **smoothness** of feature space.

Part 1 Data-noise Learning

- Part 1.1: Overview of Data-noise Learning
- Part 1.2: Correspondence Noise Learning
- Part 1.3: Graph-noise Learning
- Part 1.4: Group-structured Noise Learning
- Part 1.5: Takeaways

Takeaways

Correspondence noise learning aims at addressing the mismatch data issue in cross-modal learning.

- Label perspective: rectify noisy labels via knowledge obtained from early training phase.
- Data perspective: modify textual data in image-text pairs by generating cleaner data.

Graph-noise learning aims at mitigating the negative influence of noisy node labels in graph node classification.

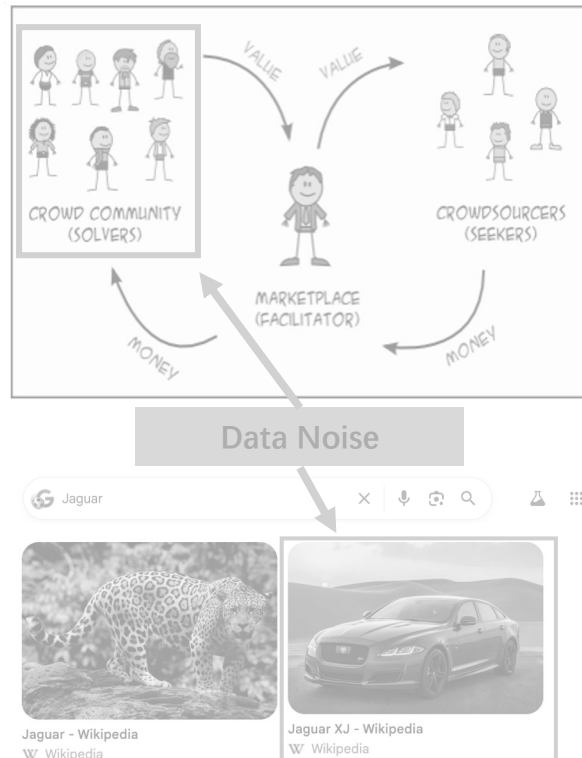
- Node perspective: identify clean nodes via training with easy nodes first.
- Loss perspective: modify loss function by employing a graph contrastive regularization.

Group-structured noise learning aims at predicting accurate instance-level labels under coarse-grained supervision.

- Loss perspective: design robust loss function by selecting the best-matched object.
- Label perspective: disambiguate candidate labels via smoothness in the feature space.

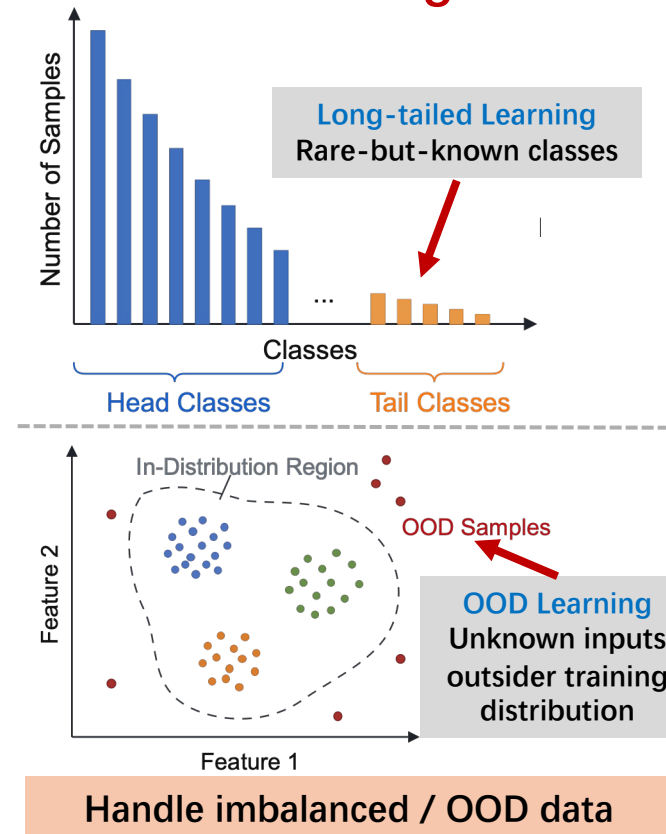
Course I: Trustworthy Data-centric Learning

Part 1. Data-noise Learning

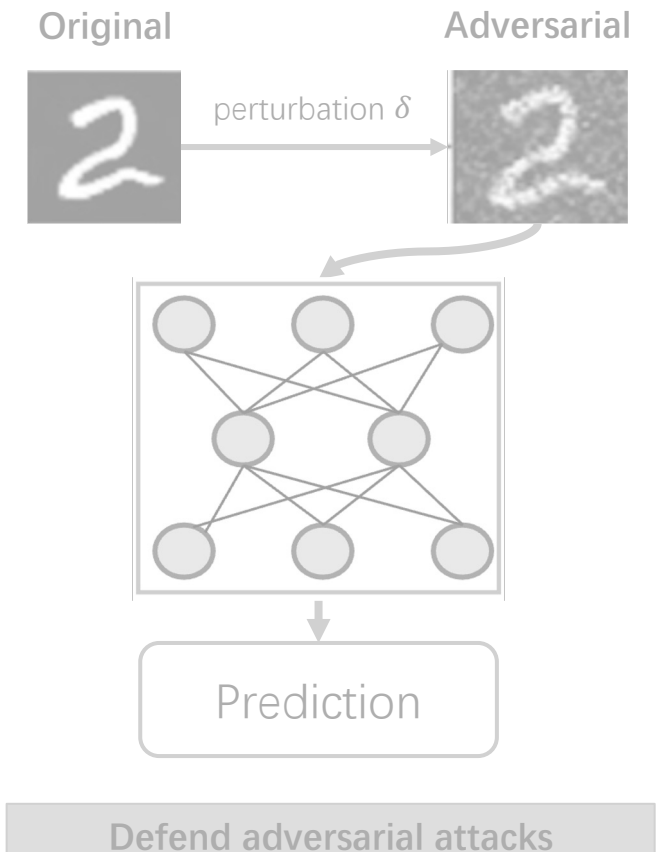


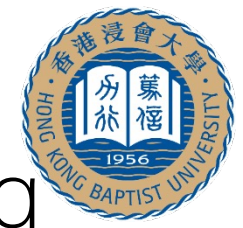
Learn from data with noisy information

Part 2. Long-tailed and Out-of-distribution Learning



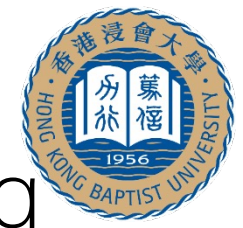
Part 3. Adversarial Attacks and Defenses





Part 2 Long-tailed and Out-of-distribution Learning

- Part 2.1: Long-tailed Learning
- Part 2.2: Out-of-Distribution Detection
- Part 2.3: Out-of-Distribution Generalization
- Part 2.4: Takeaways



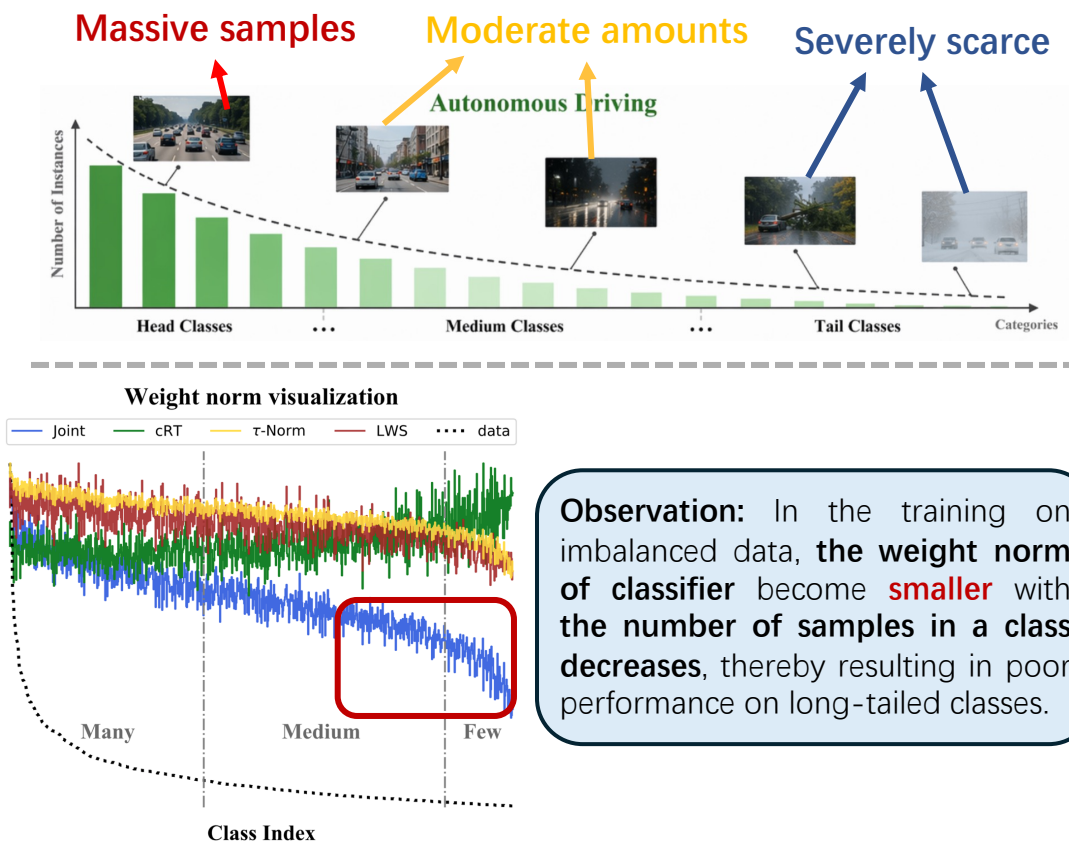
Part 2 Long-tailed and Out-of-distribution Learning

- Part 2.1: Long-tailed Learning
- Part 2.2: Out-of-Distribution Detection
- Part 2.3: Out-of-Distribution Generalization
- Part 2.4: Takeaways

Long-tailed Learning

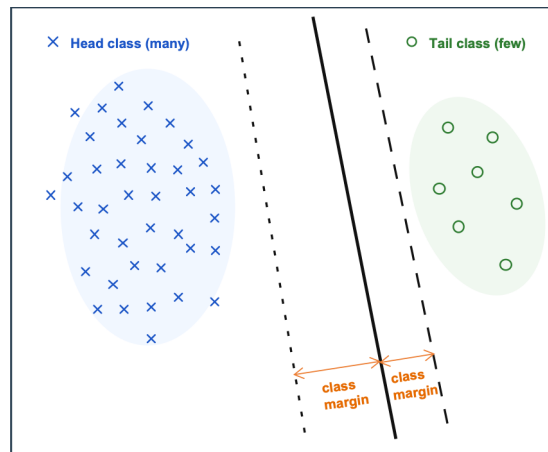
Background: Real-world data naturally exhibits a **long-tailed distribution**, where sample quantities across categories are highly imbalanced. Consequently, models become **biased** toward head classes and **unreliable** on rare but critical cases.

In the context of deep learning, two feasible directions to handle long-tailed learning problem are learning (1) **better decision boundaries**, and (2) **better representations**.



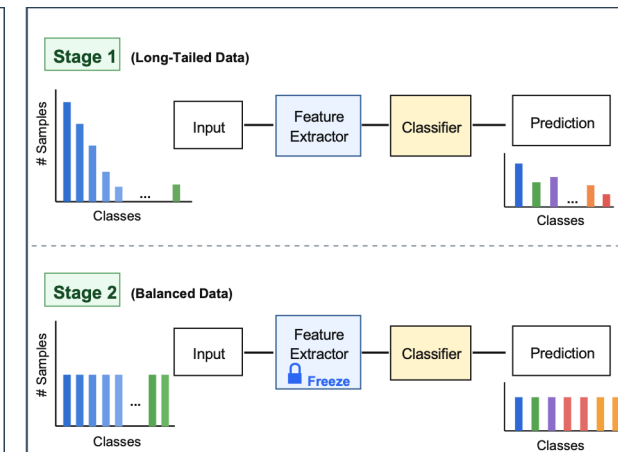
Observation: In the training on imbalanced data, the **weight norm of classifier** become **smaller** with the number of samples in a class **decreases**, thereby resulting in poor performance on long-tailed classes.

Boundary Perspective: Logit Adjustment



Aims to adjust logits/margins to **reduce bias against tail classes** and **enlarge decision boundaries**.

Representation Perspective: Representation / Classifier Geometry

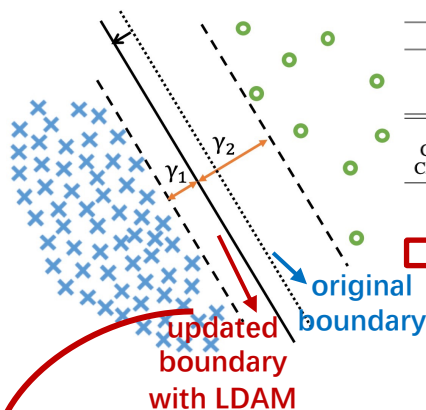


Aims to learn **better representations**, and then refining classifier geometry with a compound objective.

Logit Adjustment

Existing works on **logit adjustment** mainly focus on **correcting biased predictions** by modifying decision margins or aligning predictions with class priors.

Fix decision boundary



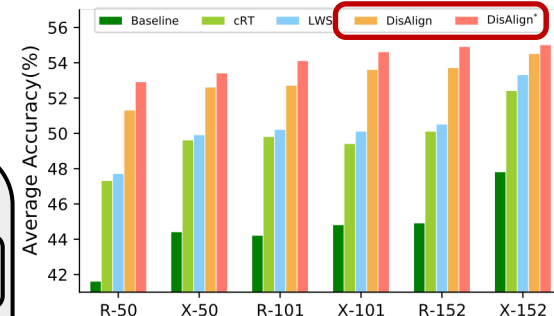
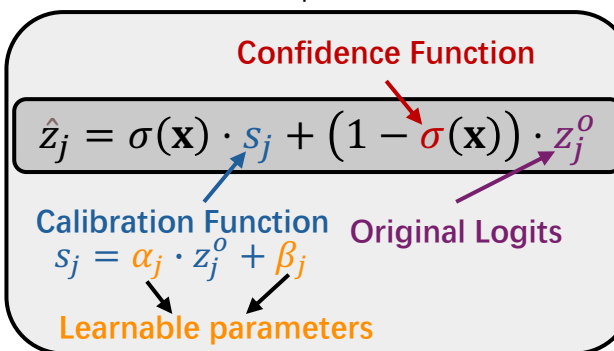
Dataset	Imbalanced CIFAR-10				Imbalanced CIFAR-100			
	long-tailed		step		long-tailed		step	
	100	10	100	10	100	10	100	10
ERM	29.64	13.61	36.70	17.50	61.68	44.30	61.45	45.37
Focal [Lin et al., 2017]	29.62	13.34	36.09	16.36	61.59	44.22	61.43	46.54
LDAM	26.65	13.04	33.42	15.00	60.40	43.09	60.42	43.73
CB RS	29.45	13.21	38.14	15.41	66.56	44.94	66.23	46.92
CB RW [Cui et al., 2019]	27.63	13.46	38.06	16.20	66.01	42.88	78.69	47.52
CB Focal [Cui et al., 2019]	25.43	12.90	39.73	16.54	63.98	42.01	80.24	49.98
HG-DRS	27.16	14.03	29.93	14.85	-	-	-	-
LDAM-HG-DRS	24.42	12.72	24.53	12.82	-	-	-	-
M-DRW	24.94	13.57	27.67	13.17	59.49	43.78	58.91	44.72
LDAM-DRW	22.97	11.84	23.08	12.19	57.96	41.29	54.64	40.54

Superior performance to pure cross-entropy loss and other baselines

Label-Distribution-Aware Margin (LDAM)

- **Problem:** Imbalanced training **skews decision boundaries**, hurting minority-class accuracy under balanced metrics.
- **Method:** Label-Distribution-Aware Margin uses label frequency information to reweight the loss for **enlarging margins γ_2 for rarer classes**.
- **Results:** **Stronger** minority-class and overall **performance** on long-tailed benchmarks.

Optimal logit adjustment should be adaptive, depending on the model's **confidence** in its prediction.



Raw logits are biased to the head classes due to long-tail training. Thus, a **calibration function** is required to **adaptively** correct per-class bias when the model is uncertain.

Distribution Alignment (DisAlign)

- **Problem:** Two-stage long-tail pipelines often keep **biased decision boundaries** and rely on heuristic score fixes.
- **Method:** Freeze features and learn class-specific α_j and β_j to adaptively **rescales logits** towards **balanced predictions**.
- **Results:** **Significant gains** across tasks and datasets with consistent improvements in semantic segmentation.

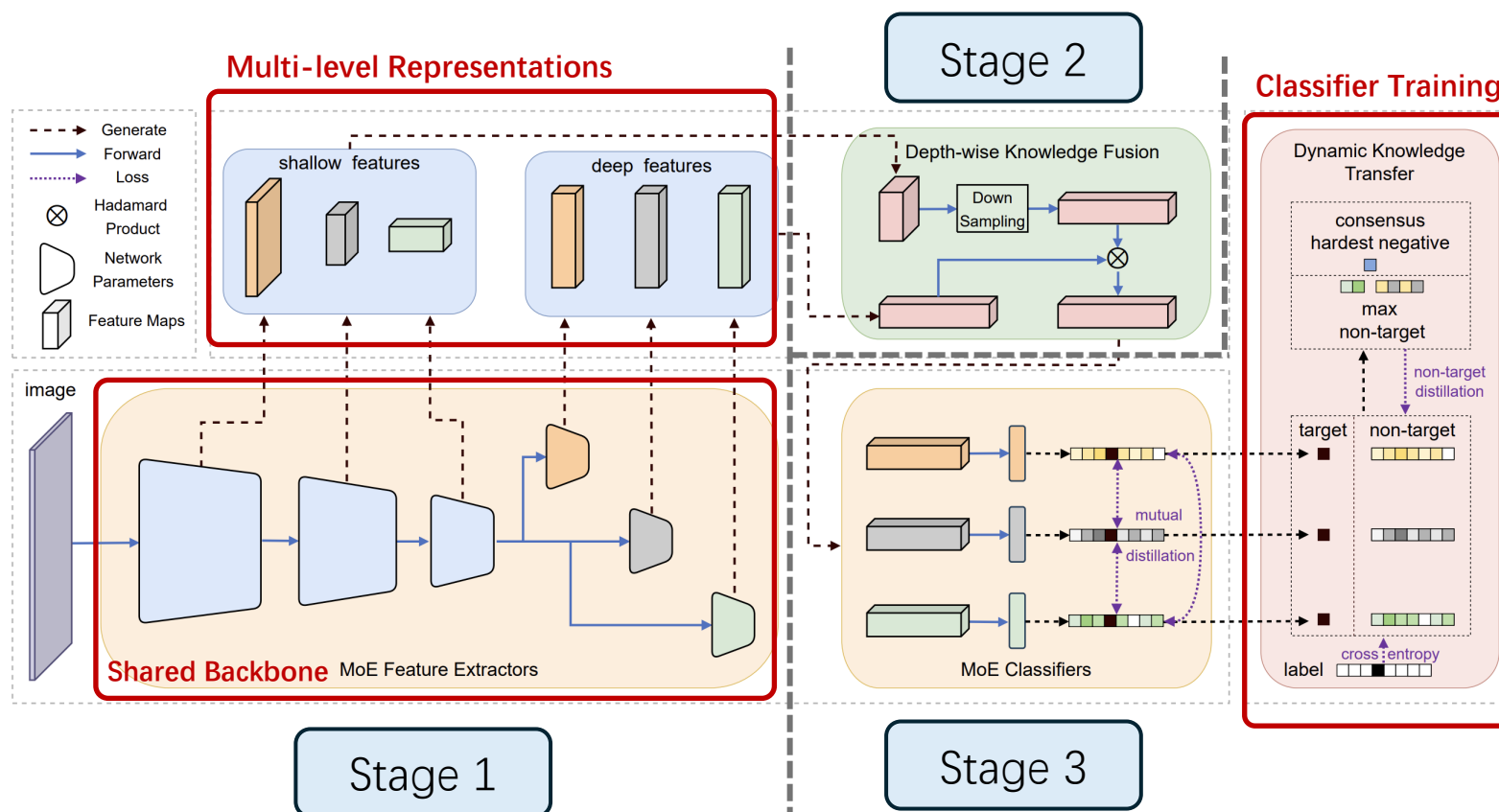
Representation and Classifier Geometry

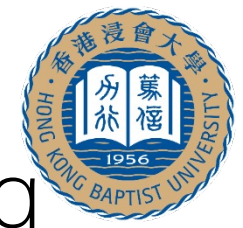
Existing works on **representation and classifier geometry** mainly focus on **re-balance the feature space** by integrating multi-depth representations and transferring debiased knowledge.

Problem: Homogeneous expert architectures **mismatch diverse class-depth preferences** and suffer from severe negative interference, thereby distorting the **classifier geometry** of tail classes.

Method: SHIKE

- **Stage 1: Shared backbone routing**
Extracts multi-level representations from different backbone depths.
- **Stage 2: Depth-wise knowledge fusion**
Fuses unique shallow features with deep features to calibrate representation geometry.
- **Stage 3: Dynamic knowledge transfer**
Transfers debiased knowledge among experts to suppress negative classes and reshape boundaries.





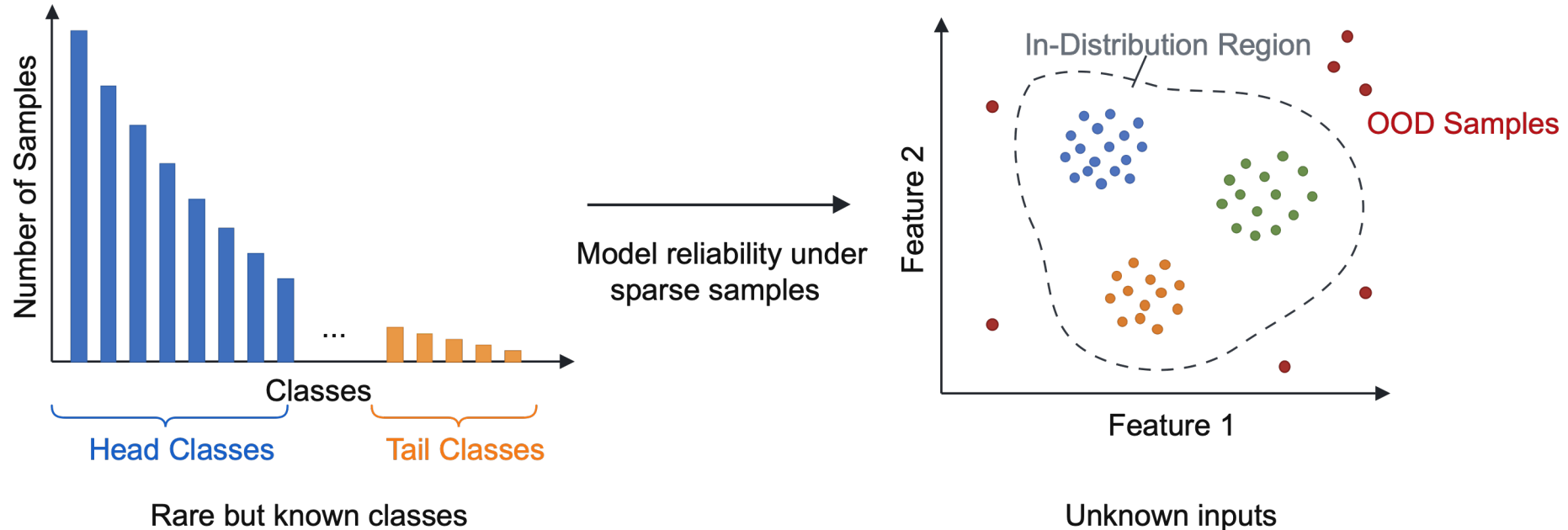
Part 2 Long-tailed and Out-of-distribution Learning

- Part 2.1: Long-tailed Learning
- Part 2.2: Out-of-Distribution Detection
- Part 2.3: Out-of-Distribution Generalization
- Part 2.4: Takeaways

From Rare Knowns to Unknowns

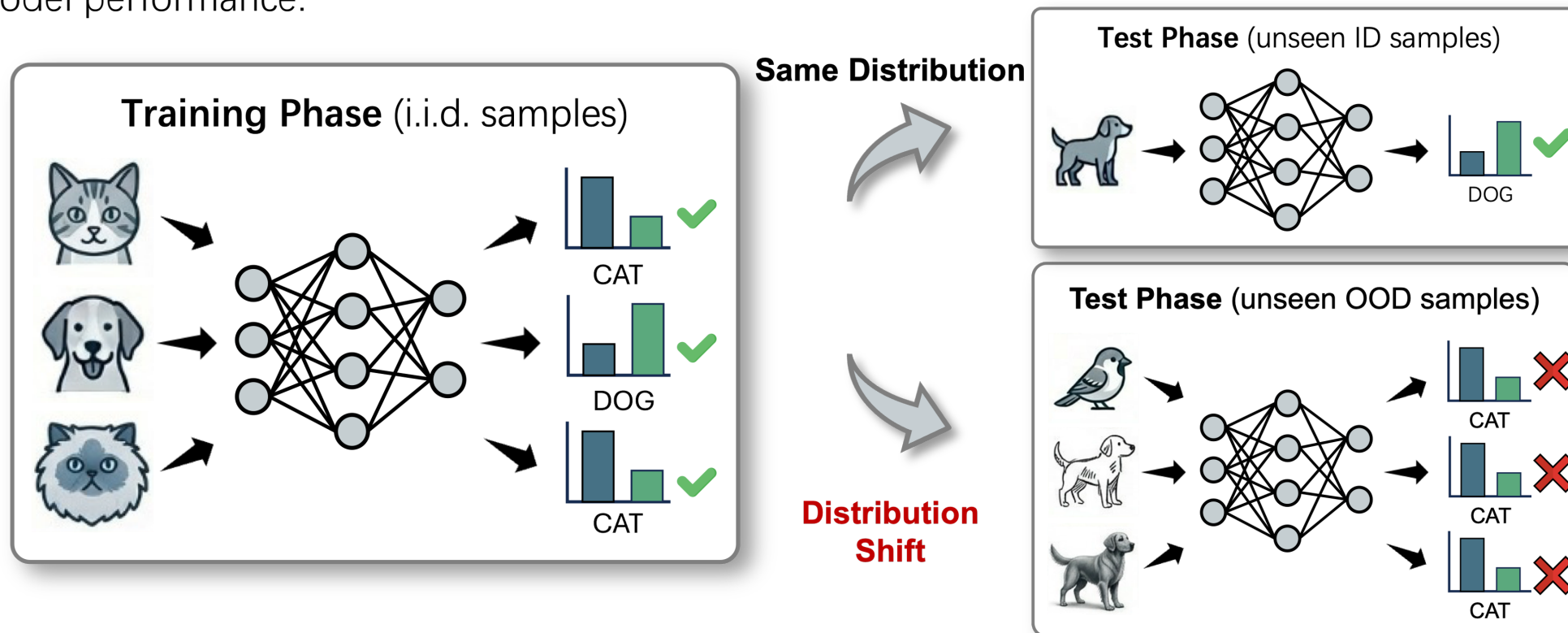
As **training samples** becomes sparse (i.e., long-tail) or invisible (i.e., out-of-distribution), model reliability becomes a **vital concern**.

- **Tail Class Recognition:** Data scarcity causes high predictive uncertainty, leading to unstable decision boundaries for minority classes.
- **Out-of-Distribution Learning:** The inability to effectively delineate between the learned feature manifold and novel, unseen inputs poses significant risk to model robustness.



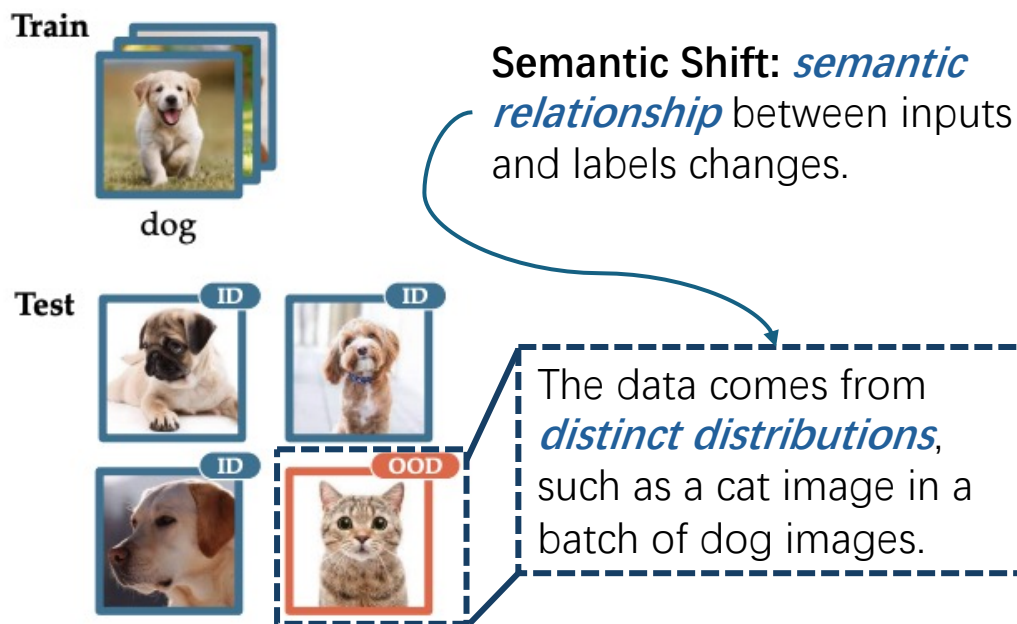
Out-of-distribution Learning

- Conventional machine learning models are trained to generalize from a finite dataset to new, unseen **i.i.d.** samples drawn from **the same distribution**, i.e., **in-distribution (ID) data**.
- However, when test data distribution shifts, i.e., **out-of-distribution (OOD) data**, we can no longer guarantee the model performance.



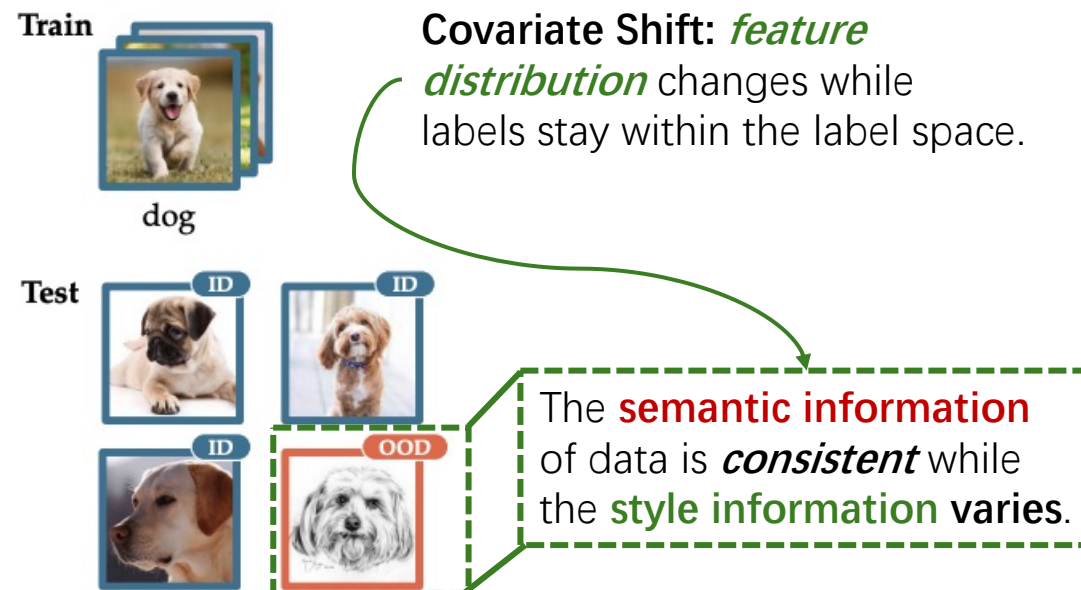
Two Scenarios of Out-of-distribution Learning

OOD Detection: Detect OOD samples among mixed unseen test samples and reject them to reduce risks.



Ultimate Goal: **Detect** and **reject** OOD samples.

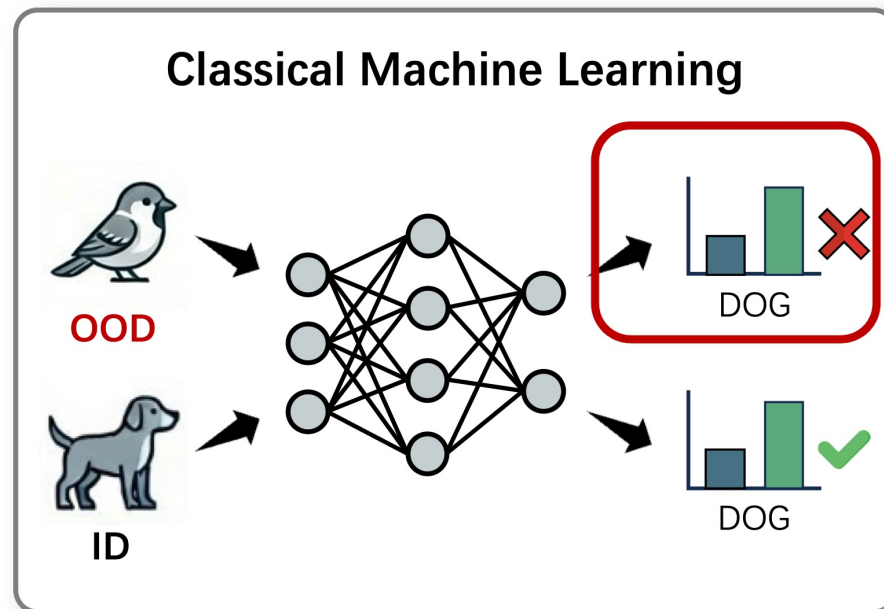
OOD Generalization: Learn models that maintain good performance on unseen distributions with shifted features or environments.



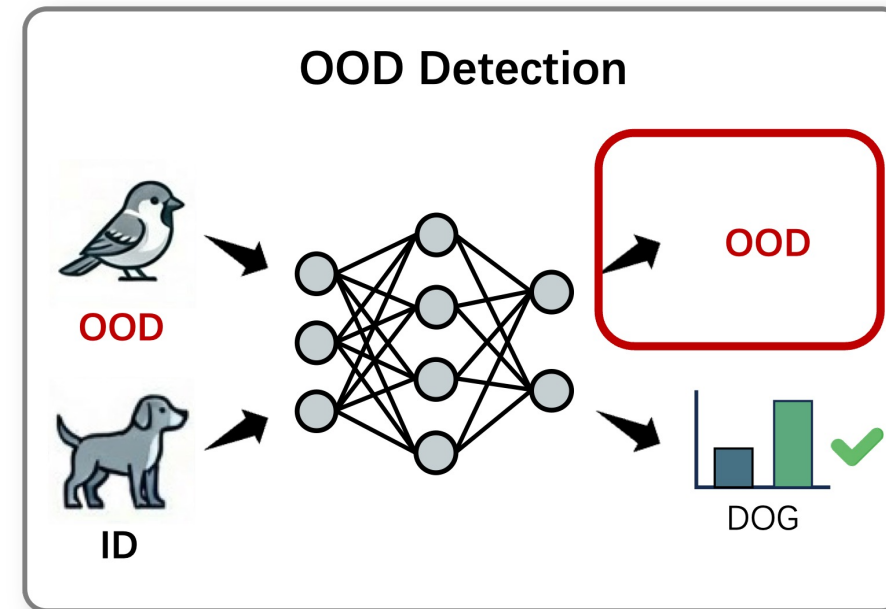
Ultimate Goal: **Generalize** to OOD samples.

OOD Detection

In the context of OOD detection, machine learning models are expected to **detect semantic distribution shifts**, instead of making label predictions on OOD examples.



Regardless of models, **conventional** machine learning models will make **incorrect predictions** on OOD examples.



In the context of **OOD detection**, the models are expected to **detect OOD data without having to perform label predictions**.

A Practical Scenario of OOD Detection

Road safety in autonomous driving: *Identifying unknown scenarios or objects* to prevent traffic accidents and ensure robustness.

Standard Vehicle
(Known)



Horse-Drawn Carriage
(OOD Objects)



Fallen Tree on Road
(OOD Scenarios)

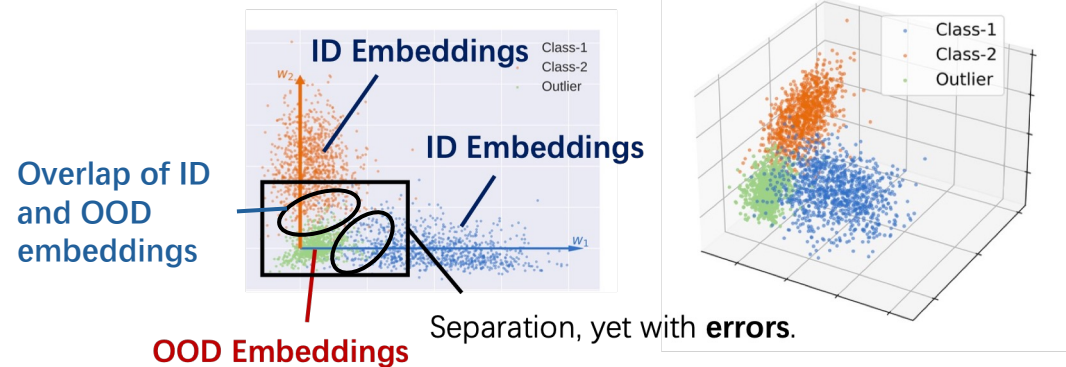


- Models must identify **previously unseen objects** on the road to maintain safety boundaries.
- The primary goal is to reliably **flag OOD data**, not to force **the vehicle prediction**.

Preliminary of OOD Detection

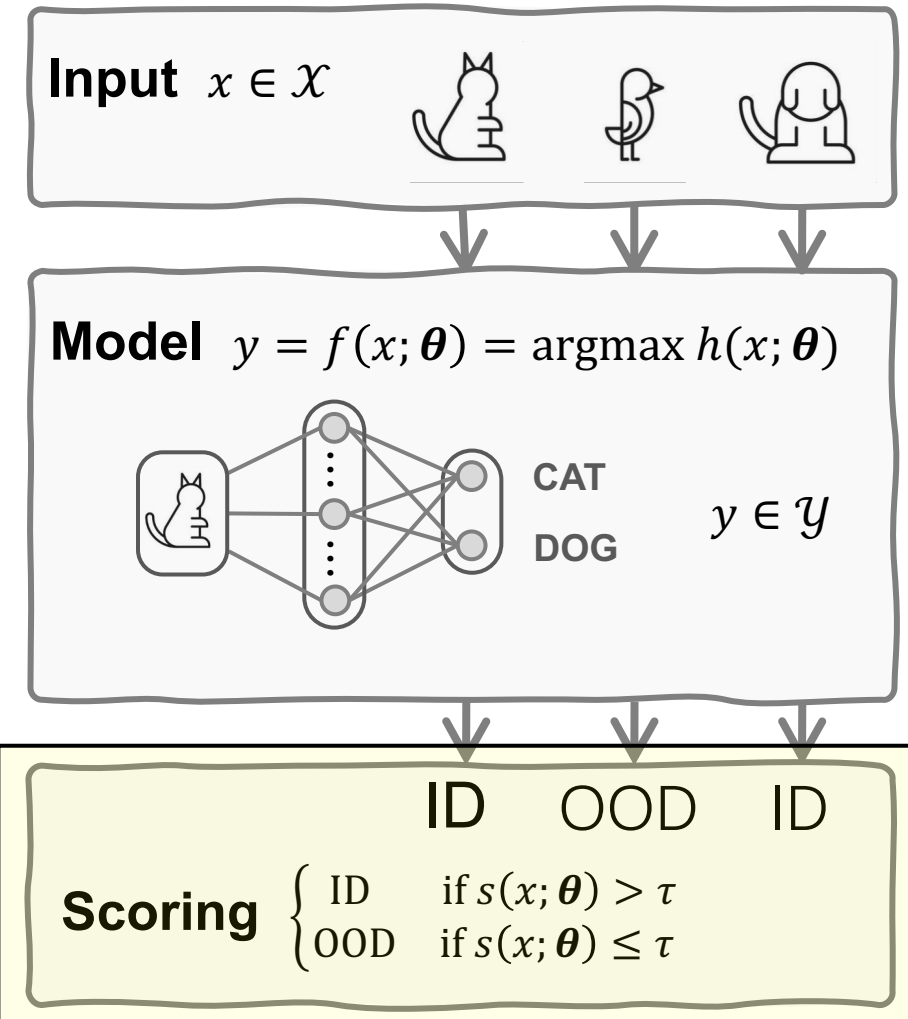
Observations

- Pre-trained models can separate ID and OOD data in embedding space **to some extent**.
- However, the inherent manifold **overlap** remains a **major obstacle** to reliable detection.



Solutions

- Technically formulate OOD detection as a **binary classification problem**.
- Design and optimize the **score function** for OOD example identification.



Post-hoc OOD Detection in DNNs

Post-hoc OOD detection for deep neural networks.

- Perform detection at inference time with pre-trained models.
- No modification to model weights and no re-training operations are required.
- Can be applied to most existing DNNs.

Post-hoc OOD detection aims to design **OOD scores** from the perspective of model outputs or features.

Output-level OOD Score

- Use **model outputs** to distinguish OOD samples.

MSP: using the maximum softmax probability as an OOD score.

$$s_{\text{MSP}}(x; \theta) = \max_k \text{softmax}_k h(x; \theta)$$

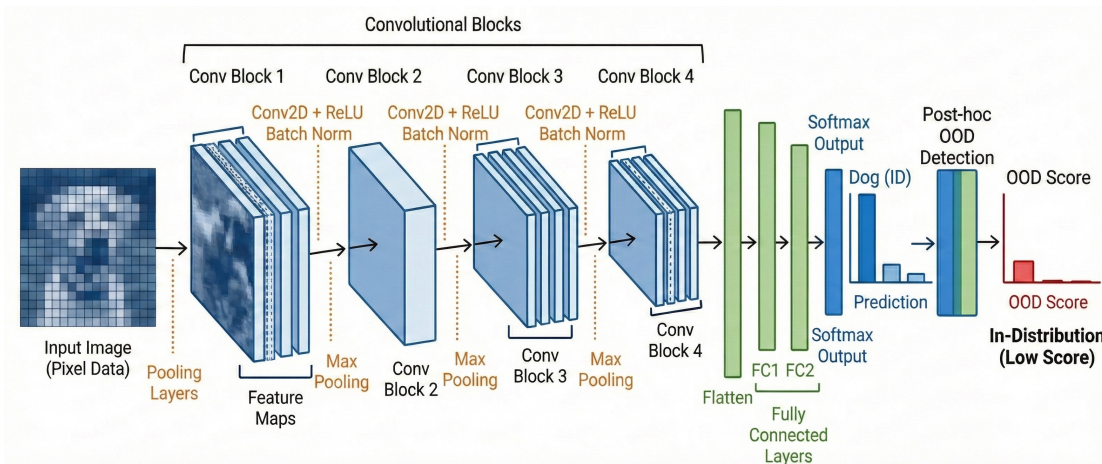
Feature-level OOD Score

- Use **model features** to distinguish OOD samples.

k-NN: using distances between features and k -nearest neighbors in the feature space as an OOD score.

$$s_{k\text{-NN}}(x; \theta) = \|h(x; \theta) - \boxed{z(k)}\|_2$$

$\downarrow$
k-th nearest neighbor



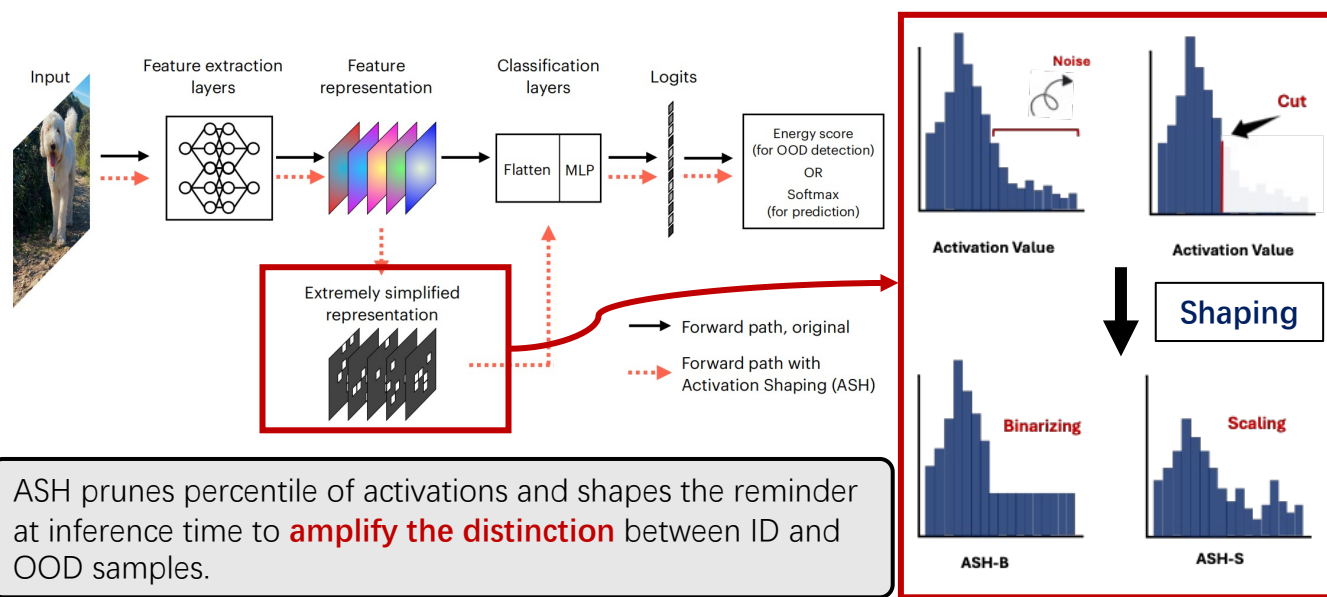
ASH Method

Problem:

- Raw post-hoc logits provide **weak ID/OOD separation**.
- Need a training-free, architecture-agnostic plug-in to **sharpen separation at inference**.
- Apply on only **layer activations** without training.

Method: Activation Shaping (ASH)

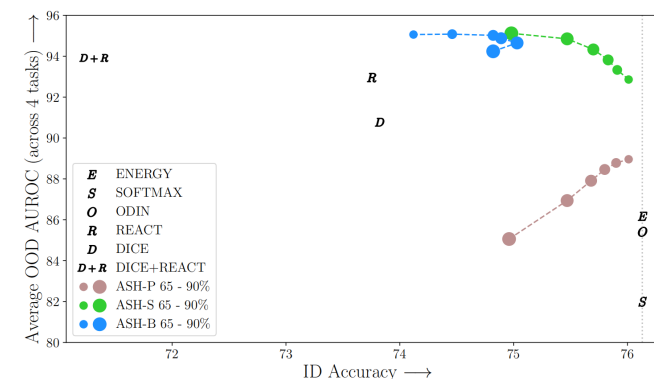
- Prune** percentile of activations
- Shape** the remainder at inference time
- Amplify** the distinction between ID and OOD samples



Results:

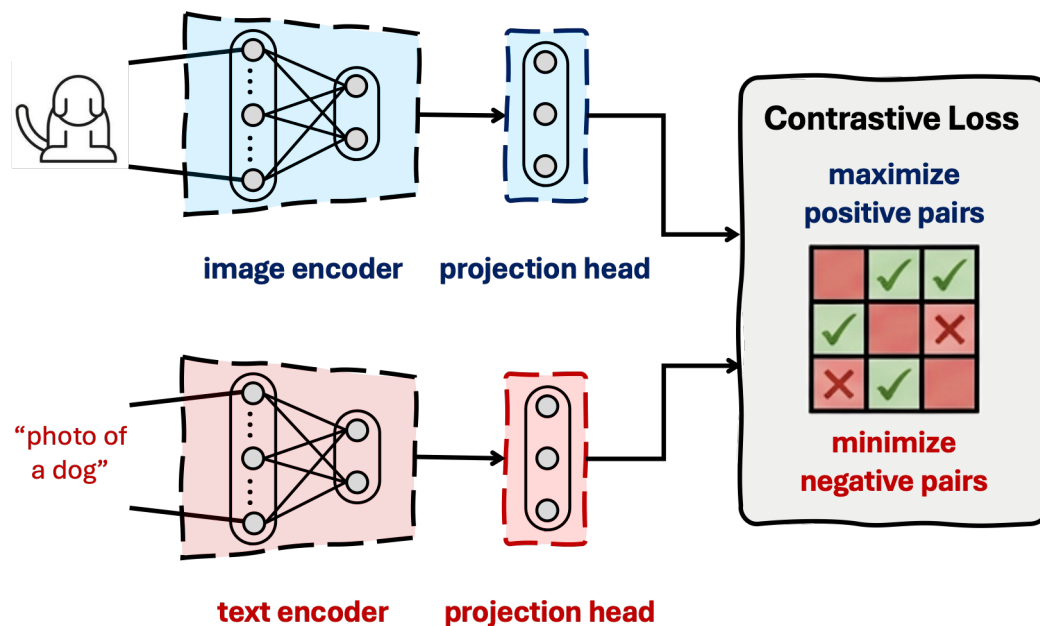
- Works well** with different OOD scores.
- Lower affect** on ID Accuracy.
- Consistently improves** OOD detection performance

Model	Methods	OOD Datasets									
		iNaturalist		SUN		Places		Textures		Average	
		FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
ResNet	Softmax score	54.99	87.74	70.83	80.86	73.99	79.76	68.00	79.61	66.95	81.99
	ODIN	47.66	89.66	60.15	84.59	67.89	81.78	50.23	85.62	56.48	85.41
	Mahalanobis	97.00	52.65	98.50	42.41	98.40	41.79	55.80	85.01	87.43	55.47
	Energy score	55.72	89.95	59.26	85.89	64.92	82.86	53.72	85.99	58.41	86.17
	ReAct	20.38	96.22	24.20	94.20	33.85	91.58	47.30	89.80	31.43	92.95
	DICE	25.63	94.49	35.15	90.83	46.49	87.48	31.72	90.30	34.75	90.77
	DICE + ReAct	18.64	96.24	25.45	93.94	36.86	90.67	28.07	92.74	27.25	93.40
	ASH-P (Ours)	44.57	92.51	52.88	88.35	61.79	85.58	42.06	89.70	50.32	89.04
	ASH-B (Ours)	14.21	97.32	22.08	95.10	33.45	92.31	21.17	95.50	22.73	95.06
	ASH-S (Ours)	11.49	97.87	27.98	94.02	39.78	90.98	11.93	97.60	22.80	95.12
MobileNet	Softmax score	64.29	85.32	77.02	77.10	79.23	76.27	73.51	77.30	73.51	79.00
	ODIN	55.39	87.62	54.07	85.88	57.36	84.71	49.96	85.03	54.20	85.81
	Mahalanobis	62.11	81.00	47.82	86.33	52.09	83.63	92.38	33.06	63.60	71.01
	Energy score	59.50	88.91	62.65	84.50	69.37	81.19	58.05	85.03	62.39	84.91
	ReAct	42.40	91.53	47.69	88.16	51.56	86.64	38.42	91.53	45.02	89.47
	DICE	43.09	90.83	38.69	90.46	53.11	85.81	32.80	91.30	41.92	89.60
	DICE + ReAct	32.30	93.57	31.22	92.86	46.78	88.02	16.28	96.25	31.64	92.68
	ASH-P (Ours)	54.92	90.46	58.61	86.72	66.59	83.47	48.48	88.72	57.15	87.34
	ASH-B (Ours)	31.46	94.28	38.45	91.61	51.80	87.56	20.92	95.07	35.66	92.13
	ASH-S (Ours)	39.10	91.94	43.62	90.02	58.84	84.73	13.12	97.10	38.67	90.95



Multi-Modal CLIP

Cross-modal alignment (e.g., CLIP) reduces reliance on the **shortcut** to learn classification, encouraging more general, semantically meaningful features.



Contrastive Language-Image Pre-training
(CLIP) framework

CLIP aligns images and their corresponding captions in the representation space.

Core Insight

- maximizing similarity of positive pairs
- minimizing similarity of negative pairs

 ↔ a photo of **dog**

 ↔ **X** a photo of **cat**

Why does CLIP work?

CLIP is pre-trained to align images and text representations on a dataset with **400M** image-text pairs.

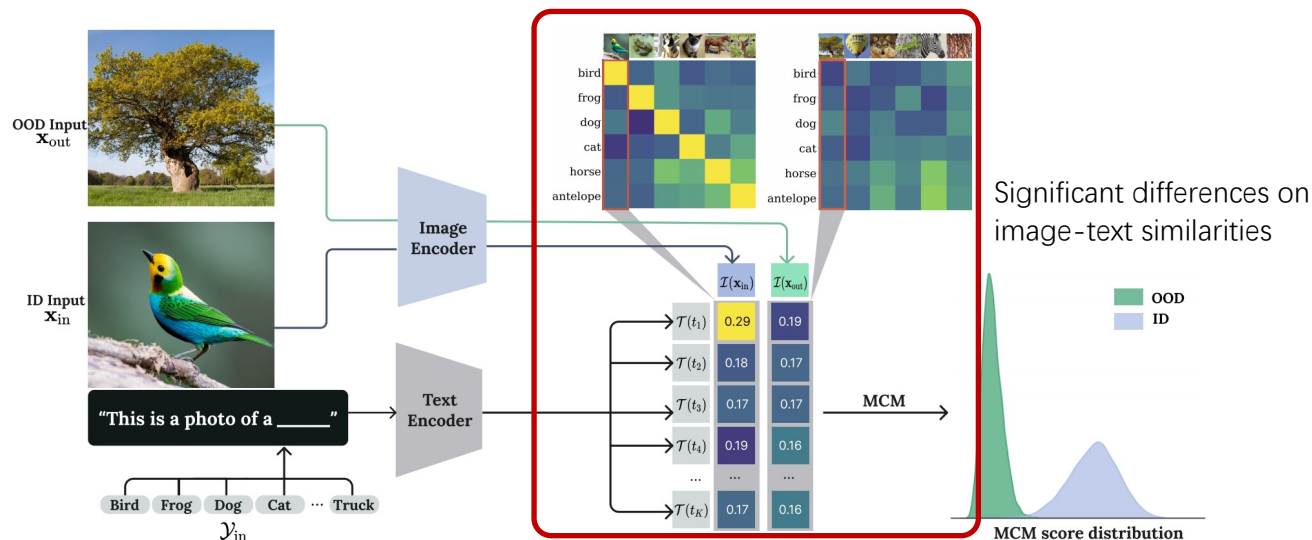
MCM Method

Problem:

- **ID-training dependency:** Most existing OOD detection methods require **in-distribution training** and are tied to **specific model architectures**.
- **Zero-shot scoring gap:** It is unclear how to turn CLIP's image-text similarity into a reliable OOD detection score using only text prompts.

Method: Maximum Concept Matching (MCM)

- **Align** visual features with textual concept prototypes
- **Employ softmax scaling** to amplify the distinction between ID and OOD in a zero-shot setting.



Results:

- **Formulate** a CLIP-based zero-shot OOD detection setting.
- **Delivers strong OOD detection performance** across several standard benchmarks.

Superior performance across different ID datasets in zero-shot setting.

Table 1: Zero-shot OOD detection with MCM score based on CLIP-B/16 with various ID datasets.

ID Dataset	OOD Dataset									
	iNaturalist		SUN		Places		Texture		Average	
	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑
CUB-200 [80]	9.83	98.24	4.93	99.10	6.65	98.57	6.97	98.75	7.09	98.66
Stanford-Cars [39]	0.05	99.77	0.02	99.95	0.24	99.89	0.02	99.96	0.08	99.89
Food-101 [6]	0.64	99.78	0.90	99.75	1.86	99.58	4.04	98.62	1.86	99.43
Oxford-Pet [57]	2.85	99.38	1.06	99.73	2.11	99.56	0.80	99.81	1.70	99.62
ImageNet-10	0.12	99.80	0.29	99.79	0.88	99.62	0.04	99.90	0.33	99.78
ImageNet-20	1.02	99.66	2.55	99.50	4.40	99.11	2.43	99.03	2.60	99.32
ImageNet-100	18.13	96.77	36.45	94.54	34.52	94.36	41.22	92.25	32.58	94.48

Table 2: OOD detection performance for ImageNet-1k [11] as ID.

Method	OOD Dataset									
	iNaturalist		SUN		Places		Texture		Average	
	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑
Requires training (or w. fine-tuning)										
MOS [32] (BiT)	9.28	98.15	40.63	92.01	49.54	89.06	60.43	81.23	39.97	90.11
Fort et al. [19] (ViT-B)	15.07	96.64	54.12	86.37	57.99	85.24	53.32	84.77	45.12	88.25
Fort et al. [19] (ViT-L)	15.74	96.51	52.34	87.32	55.14	86.48	51.38	85.54	43.65	88.96
Energy [48] (CLIP-B)	21.59	95.99	34.28	93.15	36.64	91.82	51.18	88.09	35.92	92.26
Energy [48] (CLIP-L)	10.62	97.52	30.46	93.83	32.25	93.01	44.35	89.64	29.42	93.50
MSP [25] (CLIP-B)	40.89	88.63	65.81	81.24	67.90	80.14	64.96	78.16	59.89	82.04
MSP [25] (CLIP-L)	34.54	92.62	61.18	83.68	59.86	84.10	59.27	82.31	53.71	85.68
Zero-shot (no training required)										
MCM (CLIP-B)	30.91	94.61	37.59	92.57	44.69	89.77	57.77	86.11	42.74	90.77
MCM (CLIP-L)	28.38	94.95	29.00	94.14	35.42	92.00	59.88	84.88	38.17	91.49

Superior performance in zero-shot setting than other finetuning-based baselines.

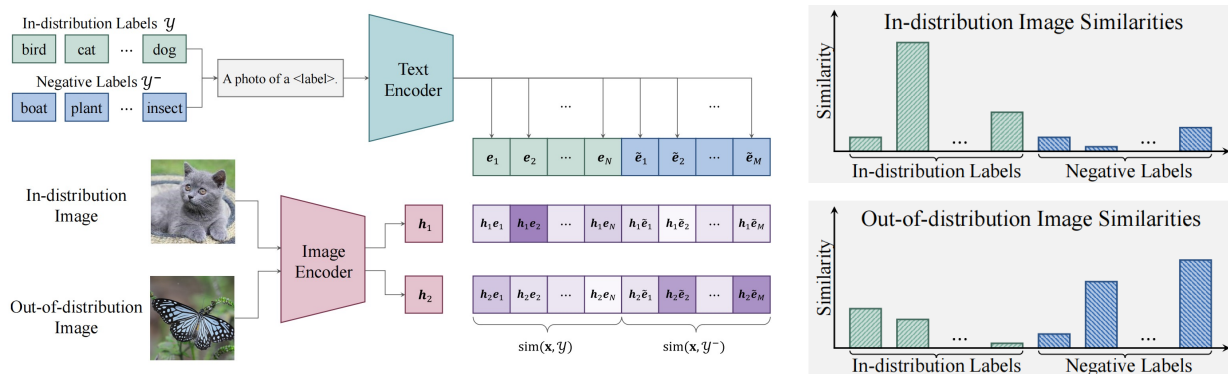
NegLabel Method

Problem:

- **Underused semantics:** Ignore the vast semantic knowledge of non-ID concepts available in pretrained vision-language models.
- **Sensitivity to domain shift:** Relies on precise classification confidence.
- **Weak in hard OOD tasks:** Struggles when OOD classes are semantically similar to ID classes.

Method: NegLabel

- **Select** effective negative labels from a large corpus;
- **Integrate** similarities to both ID labels and a large number of negative labels using a sum-softmax design.

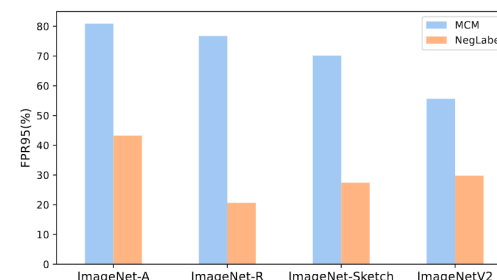


$$S(\mathbf{x}) = S^*(\blacksquare, \blacksquare) = \frac{\sum \exp(\blacksquare)}{\sum \exp(\blacksquare) + \sum \exp(\blacksquare)}$$

Results:

- **Higher performance for large-scale** OOD benchmarks.
- **Robustness** on multiple ID shifted datasets.
- **Better adaptation** on different CLIP-like architectures.

Methods	iNaturalist		SUN		Places		Textures		Average	
	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓
Requires training (or w. fine-tuning)										
MSP (Hendrycks & Gimpel, 2017)	87.44	58.36	79.73	73.72	79.67	74.41	79.69	71.93	81.63	69.61
ODIN (Liang et al., 2018)	94.65	30.22	87.17	54.04	85.54	55.06	87.85	51.67	88.80	47.75
Energy (Liu et al., 2020)	95.33	26.12	92.66	35.97	91.41	39.87	86.76	57.61	91.54	39.89
GradNorm (Huang et al., 2021)	72.56	81.50	72.86	82.00	73.70	80.41	70.26	79.36	72.35	80.82
ViM (Wang et al., 2022)	93.16	32.19	87.19	54.01	83.75	60.67	87.18	53.94	87.82	50.20
KNN (Sun et al., 2022)	94.52	29.17	92.67	35.62	91.02	39.61	85.67	64.35	90.97	42.19
VOS (Du et al., 2022)	94.62	28.99	92.57	36.88	91.23	38.39	86.33	61.02	91.19	41.32
NPOS (Tao et al., 2023)	96.19	16.58	90.44	43.77	89.44	45.27	88.80	46.12	91.22	37.93
Zero-shot (no training required)										
Mahalanobis (Lee et al., 2018)	55.89	99.33	59.94	99.41	65.96	98.54	64.23	98.46	61.50	98.94
Energy (Liu et al., 2020)	85.09	81.08	84.24	79.02	83.38	75.08	65.56	93.65	79.57	82.21
ZOC (Esmailpour et al., 2022)	86.09	87.30	81.20	81.51	83.39	73.06	76.46	98.90	81.79	85.19
MCM (Ming et al., 2022a)	94.59	32.20	92.25	38.80	90.31	46.20	86.12	58.50	90.82	43.93
CLIPN (Wong et al., 2023)	95.27	23.04	92.02	26.17	92.28	22.45	90.03	40.83	92.10	21.10
NegLabel	99.49	1.91	95.49	20.53	91.64	35.59	90.22	43.56	94.21	25.40



Architecture	Backbone	AUROC↑	FPR95↓
CLIP	ResNet50	92.97	30.70
	ResNet50x16	93.43	29.50
	ViT-B/32	93.67	27.92
	ViT-B/16	94.21	25.40
	ViT-L/14	94.47	24.81
ALIGN	EfficientNet-B7	90.76	39.66
AltCLIP	ViT-L/14	94.60	23.63
GroupViT	GroupViT	91.72	33.10

Training-time OOD Detection

Training-time OOD detection

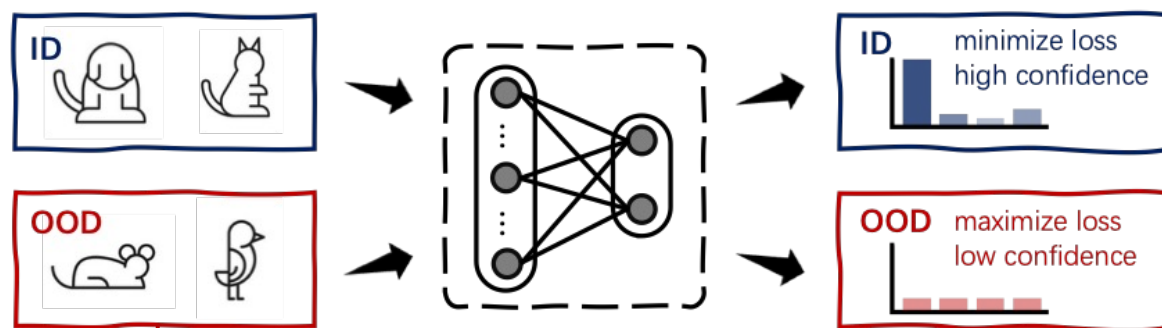
Different from post-hoc OOD detection, training-time OOD detection takes OOD detection as **an additional binary classification task for training**, enabling models to directly learn to distinguish ID from OOD patterns.

Outlier Exposure

Improves OOD detection by training models with auxiliary outlier data

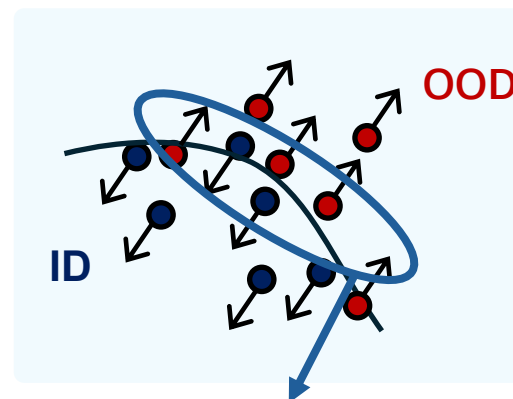
Outlier Exposure regularization

$$\mathcal{L}_{OE} = \mathbb{E}_{(x,y) \in \mathcal{D}_{in}} [\ell_{CE}(f(x), y)] + \lambda \mathbb{E}_{x' \in \mathcal{D}_{out}^{aux}} [\ell_{OE}(f(x'))]$$



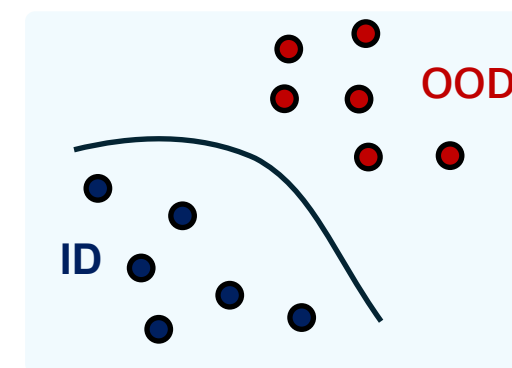
make the model **learn from extra OOD data**

before Outlier Exposure



The decision boundary **cannot** well separate ID and OOD data.

after Outlier Exposure



ID and OOD data are **clearly separated**.

DAL Method

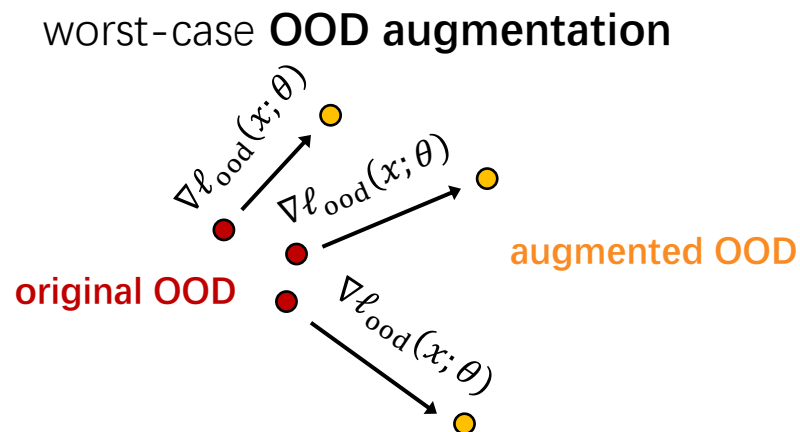
Data sampling learns from fewer examples, while data augmentation from more.

Problem:

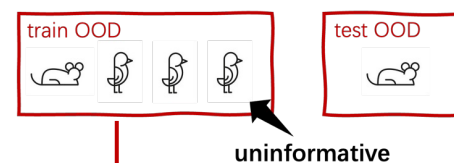
- **Redundant data:** Training data include test data, but many samples are uninformative.
- **Scarce data:** Training data contain no test data.

Method: DAL

- **Sample** original OOD data.
- Gradient ascent to increase their respective loss values.
- **Augment** OOD data for training.

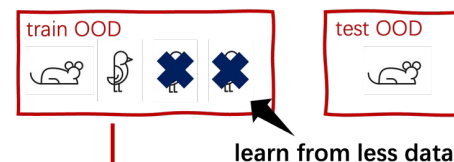


Redundant Data



Data sampling / reweighting
removes uninformative samples.

Redundant Data → Data Sampling

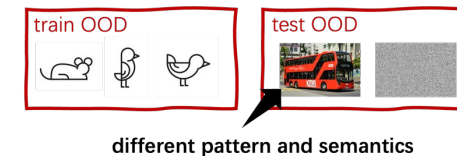


$$\mathbb{E}_{x \sim \mathcal{D}_{ood}} [w_x \ell_{ood}(x; \theta)]$$

reweighting

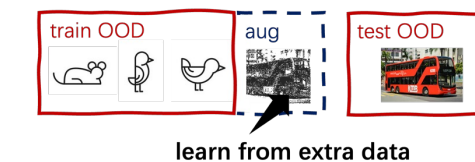
learn from resampled data

Scarce Data



Data augmentation enlarges the
OOD coverage during training.

Scarce Data → Data Augmentation



$$\mathbb{E}_{x \sim \mathcal{D}_{ood}} [\ell_{ood}(x'; \theta)], x' = \Phi(x)$$

augmentation

learn from extra data

VOS Method

Generating **pseudo OOD samples** for training OOD loss is another good choice.

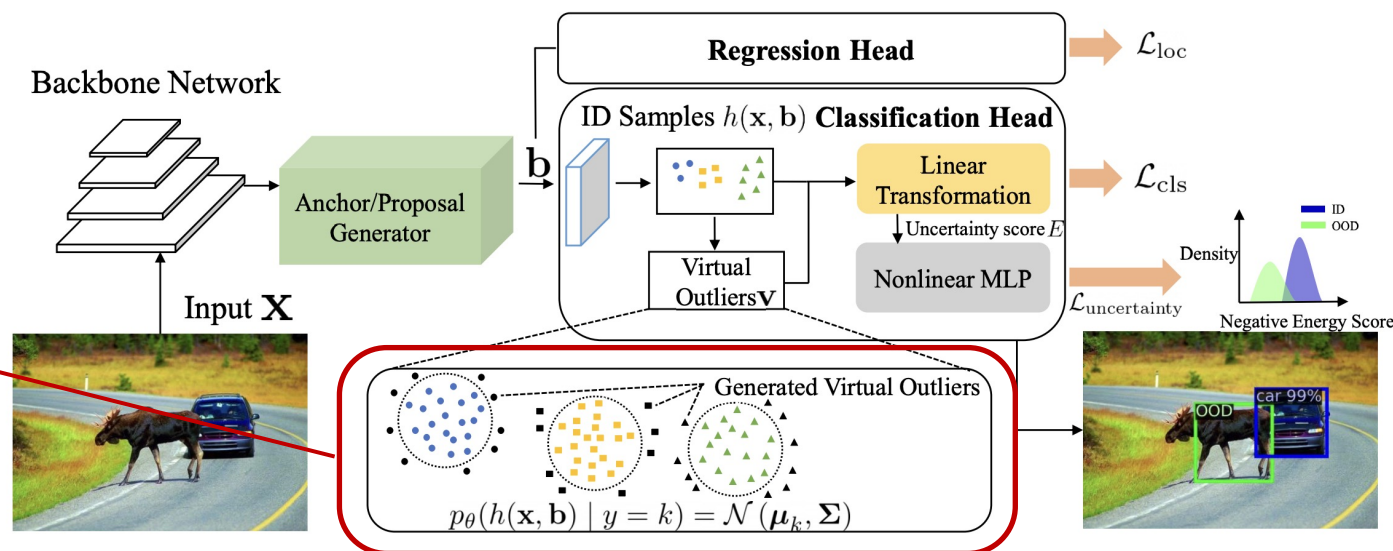
Problem

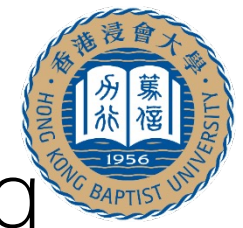
- Previous methods depend on curated OOD sets with limited and unreliable coverage.
- How to train a classifier that recognizes unknowns at test time **without any auxiliary OOD data**?

VOS Method

- **Estimate ID support** by fitting class-conditional Gaussians with a shared covariance to penultimate-layer features;
- **Generate virtual outliers** beyond the Mahalanobis boundary and instruct the classifier to be uncertain there.

Near-boundary “unknown” supervision are generated near the ID data without a reference of external OOD data, thereby improving OOD detection while preserving ID accuracy.



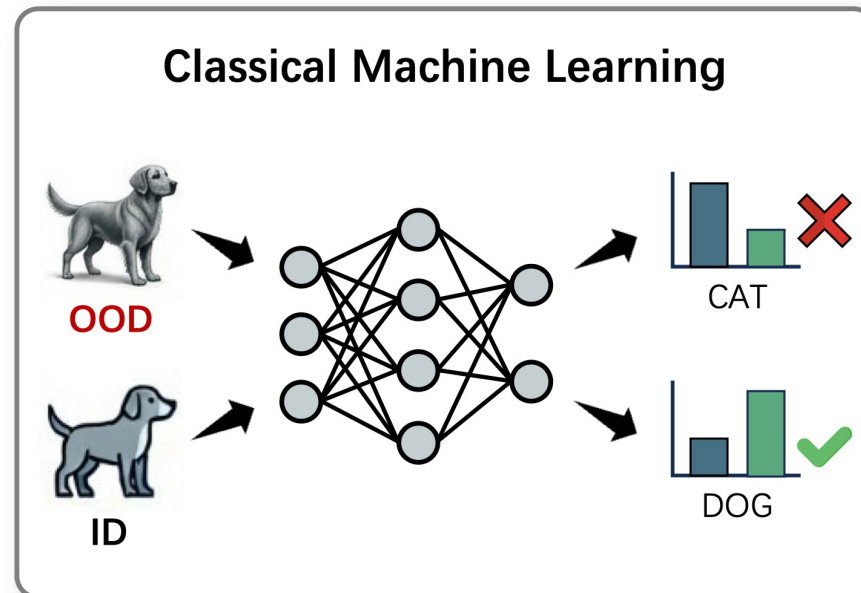


Part 2 Long-tailed and Out-of-distribution Learning

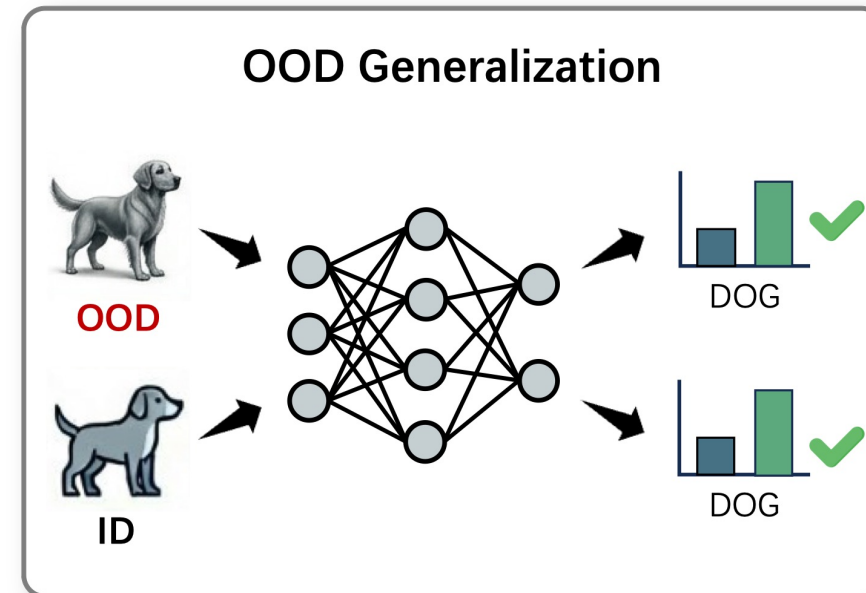
- Part 2.1: Long-tailed Learning
- Part 2.2: Out-of-Distribution Detection
- Part 2.3: Out-of-Distribution Generalization
- Part 2.4: Takeaways

OOD Generalization

In the context of OOD generalization, machine learning models are expected to **generalize well to data with covariate distribution shifts** and **still perform correct label predictions**.



If the model encounters images with different styles compared with training data, it would make **wrong predictions**.



The model should **generalize** to unseen OOD data with right label predictions.

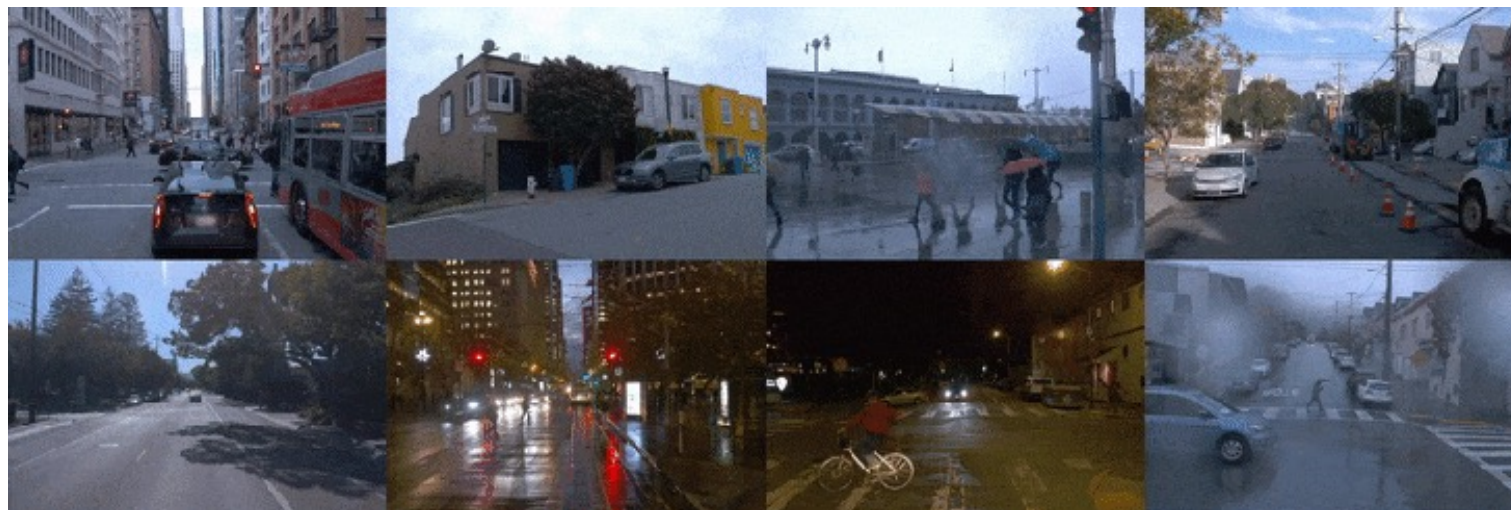
Practical Scenarios of OOD Generalization

Beyond OOD detection that merely flags unseen objects, autopilot needs to **generalize to unseen scenarios** to avoid accidents and ensure safety under different conditions.

Training Environment



OOD Test Environments



Waymo Open Challenge

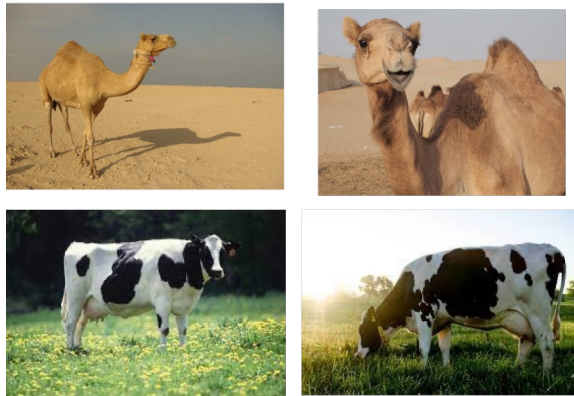
The model is **trained** on **clear, daytime urban scenes** with stable sensor inputs.

At **test** time, it must **generalize safely** to unseen scenarios such as rain, fog, night, and other sensor shifts.

How to Handle Distribution Shift?

- If we relate to how we humans classify the animals, we indeed leverage the **invariance principle** of the independent causal mechanisms.
- **Causal Invariance Principle**: The causal mechanism generating the target variable from its direct parents is independent from the changes.

Train Environment



Test Environment



Intuition: The labels of the animals are determined by its **invariant** physical features (e.g., shapes, furs and so on)

Using the **causal invariance** to classify the animals is able to **get rid of** the influence of distribution shifts.

$P(\text{Label} \mid \text{Animal etc.})$ is invariant!

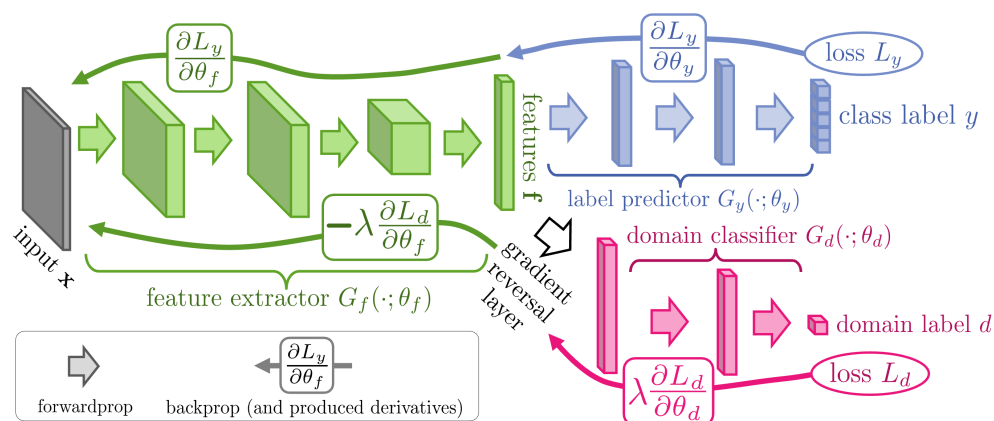
Many Implementations of Invariance

The **invariance** can be implemented in various ways.

- regularize the learned features to be domain-invariant;
- regularize the models to be robust to mild distributional perturbations of the training distributions.

Learning Domain-Invariant Features

- **Mechanism:** Use Gradient Reversal to suppress domain-specific information.
- **Objective:** Extract features that are predictive of labels but invariant to domain shifts.

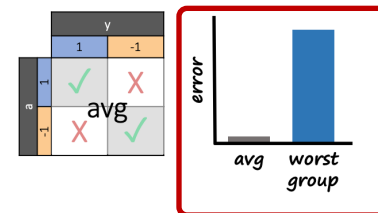


Group DRO

- **Mechanism:** Directly optimize the loss of the worst-performing group rather than the average.
- **Objective:** Ensure consistent accuracy across all groups, preventing bias against minorities.

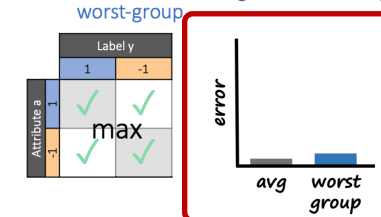
Standard (ERM): average loss

$$\mathcal{R}_{\text{ERM}}(w) = \mathbb{E}_{(x,y,g)} [\ell(w; (x, y))]$$



Group DRO: worst-group loss

$$\mathcal{R}_{\text{gDRO}}(w) = \max_{g' \in \mathcal{G}} \mathbb{E}_{(x,y,g)} [\ell(w; (x, y)) \mid g = g']$$

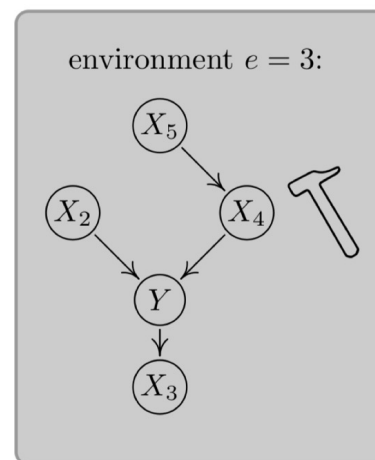
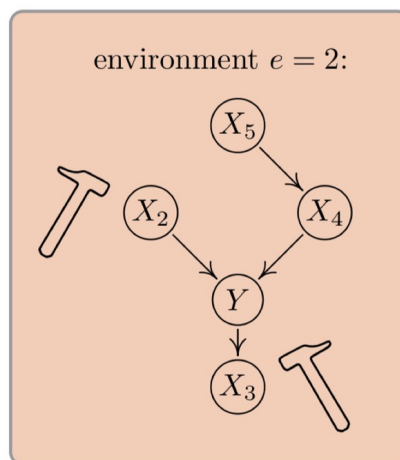
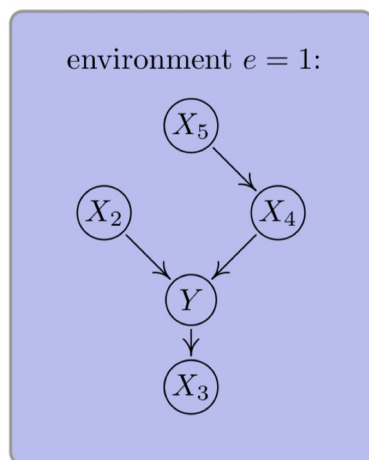


Train: known groups for each example
Test: unknown groups

Standard ERM performs poorly on minority groups (**high worst-group error**), whereas **Group DRO ensures low error across all groups**.

The Causal Invariance Principle

- Force a single classifier to work in all environments $\rightarrow$ the model relies on causal, invariant signals and generalizes OOD.



Intuition: learn an invariant representation φ so that one classifier ω remains optimal across all environments in the environment set $\mathcal{E}_{\text{tr}}$.

Invariant risk minimization (IRM):

$$\min_{f=\omega \circ \varphi} \sum_{e \in \mathcal{E}_{\text{tr}}} \mathcal{L}_e(\omega \circ \varphi)$$

← Optimization for invariant representations

$$\text{s. t. } \omega \in \arg \min_{\bar{\omega}} \mathcal{L}_e(\bar{\omega} \circ \varphi), \forall e \in \mathcal{E}_{\text{tr}}$$

← The classifier optimal for all environments

Formulation and Challenges

IRM aims at training an invariant predictor that is **simultaneously optimal** across different training domains:

$$\begin{aligned} \min_{f=\omega \circ \varphi} \sum_{e \in \mathcal{E}_{\text{tr}}} \mathcal{L}_e(\omega \circ \varphi), \\ \text{s. t. } \omega \in \arg \min_{\bar{\omega}} \mathcal{L}_e(\bar{\omega} \circ \varphi), \forall e \in \mathcal{E}_{\text{tr}} \end{aligned}$$

However, the original formulation of IRM is **challenging** to implement:

- **Bi-level Optimization Loop:** Requires solving an inner optimization problem (argmin) to **convergence for every single update of the representation φ** .
- **Gradient Flow Obstruction:** Standard backpropagation **cannot directly compute gradients through the nested argmin operator**.
- **Intractable Computation:** Scaling this "inner-loop-within-outer-loop" optimization to large deep neural networks and multiple environments is **computationally prohibitive**.

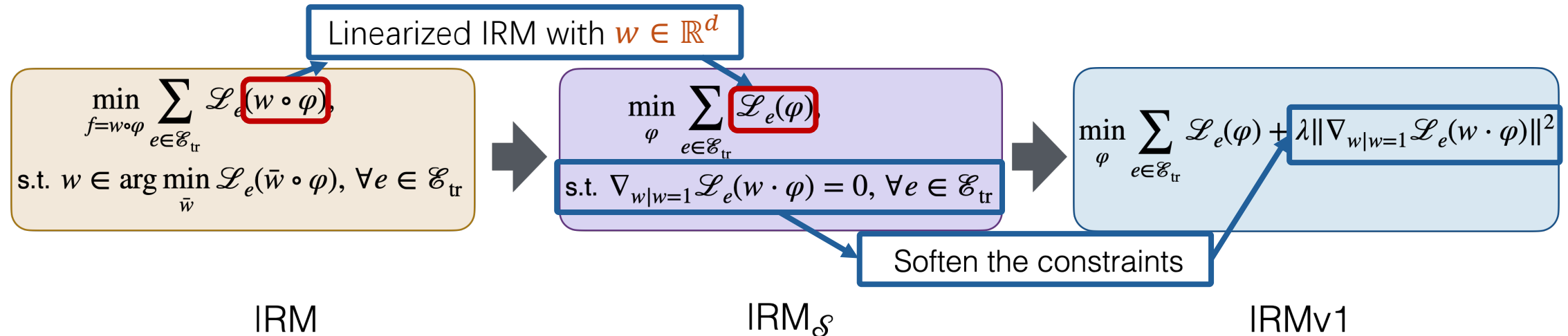
Relaxation of IRM

- **Step 1: Linearization**

- **Dummy Classifier** ($w = 1$): Without loss of generality, we fix w to a constant scalar 1.0, allowing the highly expressive network φ to absorb any classifier scaling. **→ Do not have to optimize classifier.**
- **First-Order Condition:** We replace the **argmin** constraint with **Fermat's theorem**: if $w = 1$ is optimal, the gradient of the loss at $w = 1$ must be zero $\nabla_{w|w=1} \mathcal{L}_e(w \cdot \varphi) = 0$. **→ Actually, this is a strict constraint.**

- **Step 2: Softening**

- Enforcing strict zero-gradient constraints during stochastic gradient descent (SGD) is **numerically difficult**. We convert the equality constraint into **a squared L_2 norm penalty**: $\lambda \|\nabla_{w|w=1} \mathcal{L}_e(w \cdot \varphi)\|^2$.
- **Intuition:** The penalty **suppresses** domain-specific shortcuts, penalizing the model if the optimal decision boundary attempts to shift across different environments.



PAIR Method

Does IRM truly explore the optimal solution?

Core Insight 1: Reframe the loss of IRM for the optimal solution.

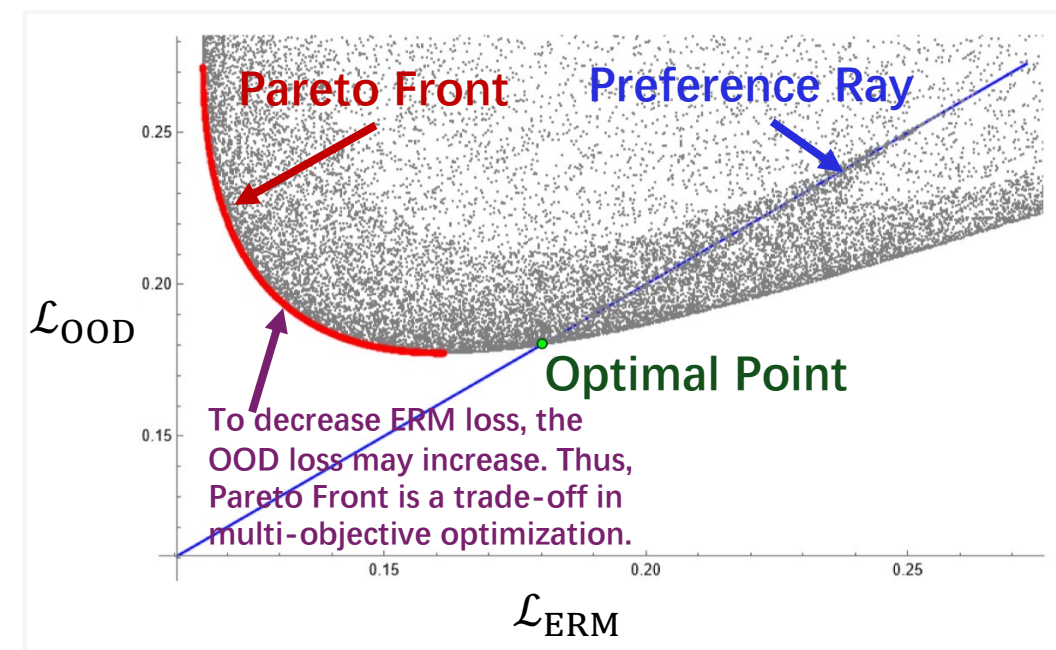
- **Main Limitation:** Traditional **linear** weighting (using λ) only finds **a single, blind trade-off** and can only find the optimal solution in the **convex regions**. **→ Shows preference but hard to tune.**
- **Solution of PAIR:** In contrast, PAIR maps the entire Pareto front and directly guides the optimization to the exact **optimal solution (Green Dot)** matching your **preference (Blue Line)**.

$$\min_f \mathcal{L}_{\text{ERM}} + \lambda \cdot \mathcal{L}_{\text{OOD}} \longrightarrow \min_f \{\mathcal{L}_{\text{ERM}}, \mathcal{L}_{\text{OOD}}\}^T$$

Framing IRM as a **Multi-Objective Problem**:

- Simultaneously minimize two objectives;
- **Pareto Front (Red Curve)** indicates the optimal trade-offs between the two objectives. Any point on this curve is **Pareto optimal**, meaning one objective cannot be improved without worsening the other.

Target Solution (Green Dot): The exact optimal point found along your chosen preference path.



PAIR Method

Preference Vector p : Encoding “what we care about”.

- $p = \{p_{\text{ERM}}, p_{\text{OOD}}\}^T$ is a user-given weight vector, **set before training — not learned**.
- It encodes the relative importance of each objective. e.g.
 - $\{1, 1\}$ means treating the objectives equally;
 - $\{1, 4\}$ means prioritizing the OOD part.

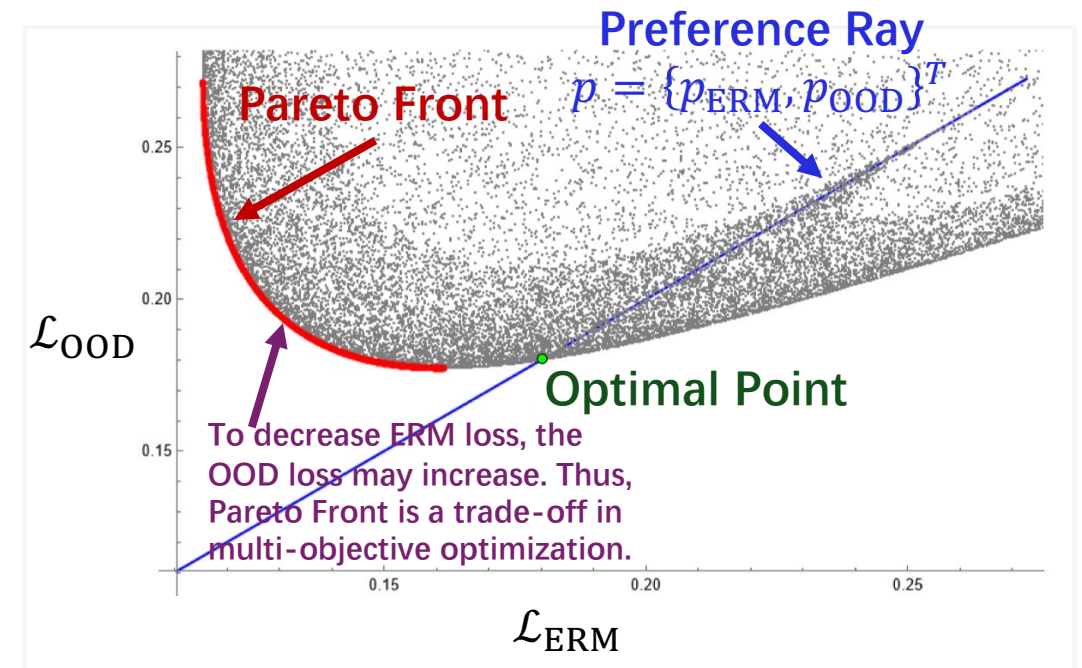
Given a preference vector $p = \{p_{\text{ERM}}, p_{\text{OOD}}\}^T$ and losses $\{\mathcal{L}_{\text{ERM}}, \mathcal{L}_{\text{OOD}}\}^T$ Exact Pareto Optimality is achieved iff.:

$$p_{\text{ERM}} \cdot \mathcal{L}_{\text{ERM}} = p_{\text{OOD}} \cdot \mathcal{L}_{\text{OOD}}$$

From preference to a single optimal point

- p draws a **preference ray** in the loss space (the **blue line**).
- The optimization process aims to achieve a case where every objective contributes **equally** given the preference $\rightarrow$ the ray hits the Pareto front at **one well-defined point (Green Dot)**.

p is our intent; the equation turns that intent into one exact point on the Pareto front.



PAIR Method

Core Insight 2: Improve the generalization via the loss variance among different environments.

Considering the biases across **different training environments**, PAIR simultaneously optimizes three objectives:
(1) $\mathcal{L}_{\text{ERM}}$: empirical risk minimization; **(2) $\mathcal{L}_{\text{OOD}}$: OOD loss**; and **(3) $\mathcal{L}_{\text{REx}}$: loss variance among environments**.

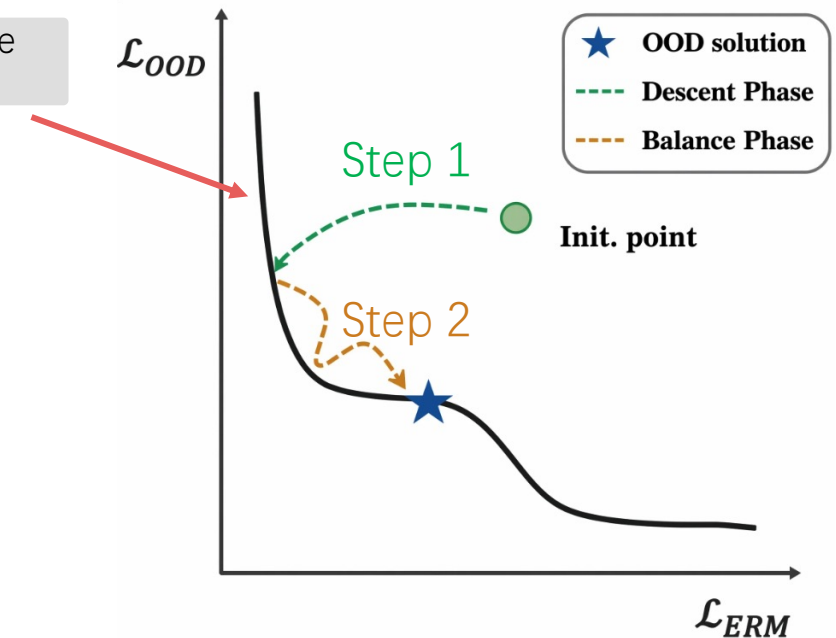
$$\min_f \{\mathcal{L}_{\text{ERM}}, \mathcal{L}_{\text{OOD}}, \mathcal{L}_{\text{REx}}\}^T$$

Minimize the variance of losses across different training environments

The three objectives form the Pareto Front together

PAIR formulates OOD generalization problem as a 3-objective optimization problem. PAIR contains two steps:

- **Step1 Descent**: Find a **common** descent direction that jointly reduces $\mathcal{L}_{\text{ERM}}$, $\mathcal{L}_{\text{OOD}}$, and $\mathcal{L}_{\text{REx}}$, quickly moving the solution toward the Pareto front.
- **Step 2 Balance**: Move along the Pareto front with a **preference-aware solver** to obtain the desired trade-off among **accuracy** ($\mathcal{L}_{\text{ERM}}$), **invariance** ($\mathcal{L}_{\text{OOD}}$), and **loss variance** ($\mathcal{L}_{\text{REx}}$).



This is a 2-D slice of a 3-D search: $\mathcal{L}_{\text{REx}}$ acts along the depth axis, so the optimal point is where all **three** losses align.

Experimental Results

	CAMELYON17	CIVILCOMMENTS	FMOw	IWILDCAM	POVERTYMAP	RxRx1	AVG. RANK(L) [†]
	Avg. acc. (%)	Worst acc. (%)	Worst acc. (%)	Macro F1	Worst Pearson r	Avg. acc. (%)	
ERM	70.3 (± 6.4)	56.0 (± 3.6)	32.3 (± 1.25)	30.8 (± 1.3)	0.45 (± 0.06)	29.9 (± 0.4)	4.50
CORAL	59.5 (± 7.7)	65.6 (± 1.3)	31.7 (± 1.24)	32.7 (± 0.2)	0.44 (± 0.07)	28.4 (± 0.3)	5.50
GroupDRO	68.4 (± 7.3)	70.0 (± 2.0)	30.8 (± 0.81)	23.8 (± 2.0)	0.39 (± 0.06)	23.0 (± 0.3)	6.83
IRMv1	64.2 (± 8.1)	66.3 (± 2.1)	30.0 (± 1.37)	15.1 (± 4.9)	0.43 (± 0.07)	8.2 (± 0.8)	7.67
V-REx	71.5 (± 8.3)	64.9 (± 1.2)	27.2 (± 0.78)	27.6 (± 0.7)	0.40 (± 0.06)	7.5 (± 0.8)	7.00
Fish	74.3 (± 7.7)	73.9 (± 0.2)	34.6 (± 0.51)	24.8 (± 0.7)	0.43 (± 0.05)	10.1 (± 1.5)	4.33
LISA	74.7 (± 6.1)	70.8 (± 1.0)	33.5 (± 0.70)	24.0 (± 0.5)	0.48 (± 0.07)	31.9 (± 0.8)	2.67
IRMx	67.0 (± 6.6)	74.3 (± 0.8)	33.7 (± 0.78)	26.6 (± 0.9)	0.45 (± 0.04)	28.7 (± 0.2)	4.00
PAIR-o	74.0 (± 7.0)	75.2 (± 0.7)	35.5 (± 1.13)	27.9 (± 0.7)	0.47 (± 0.06)	28.8 (± 0.1)	2.17

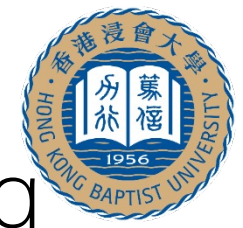
[†] Averaged rank is reported because of the dataset heterogeneity. A lower rank is better.

PAIR re-empowers IRMv1 and achieves new state-of-the-arts across **6 challenging realistic datasets**.

PAIR substantially improves the worst environment performance of all representative OOD methods up to **10%**.

		COLOREDMNIST [†]				PACS [‡]					TERRAINCOGNITA [†]				
	PAIR-s	+90%	+80%	10%	Δ wr.	A	C	P	S	Δ wr.	L100	L38	L43	L46	Δ wr.
ERM		71.0	73.4	10.0		87.2	79.5	95.5	76.9		46.7	41.8	57.4	39.7	
DANN		71.0	73.4	10.0		86.5	79.9	97.1	75.3		46.1	41.2	56.7	35.6	
DANN	✓	71.6	73.3	10.9	+0.9	87.0	81.4	96.8	77.5	+2.2	43.1	41.1	55.2	38.7	+3.1
GroupDRO		72.6	73.1	9.9		87.7	82.1	98.0	79.6		48.4	40.3	57.9	40.0	
GroupDRO	✓	72.7	73.2	13.0	+3.1	86.7	83.2	97.8	81.4	+1.8	48.4	40.3	57.9	40.0	+0.0
IRMv1		72.3	72.6	9.9		82.3	80.8	95.8	78.9		48.4	35.6	55.4	40.1	
IRMv1	✓	67.4	64.8	24.2	+14.3	85.3	81.7	97.4	79.7	+0.8	40.4	38.3	48.8	37.0	+1.4
Fishr		72.2	73.1	9.9		88.4	82.2	97.7	81.6		49.2	40.6	57.9	40.4	
Fishr	✓	69.1	70.9	22.6	+12.7	87.4	82.6	97.5	82.2	+0.6	51.0	40.7	58.2	40.8	+0.3

[†]Using the training domain validation accuracy. [‡]Using the test domain validation accuracy.



Part 2 Long-tailed and Out-of-distribution Learning

- Part 2.1: Long-tailed Learning
- Part 2.2: Out-of-Distribution Detection
- Part 2.3: Out-of-Distribution Generalization
- Part 2.4: Takeaways

Takeaways

Structural Decoupling of Feature Geometry and Classifier Calibration

- Mitigating long-tail bias has two paradigms : establishing balanced, high-dimensional representation manifolds while dynamically rescaling low-dimensional decision boundaries.

Paradigm Shift: From Empirical Data Fitting to Semantic Prior Exploitation

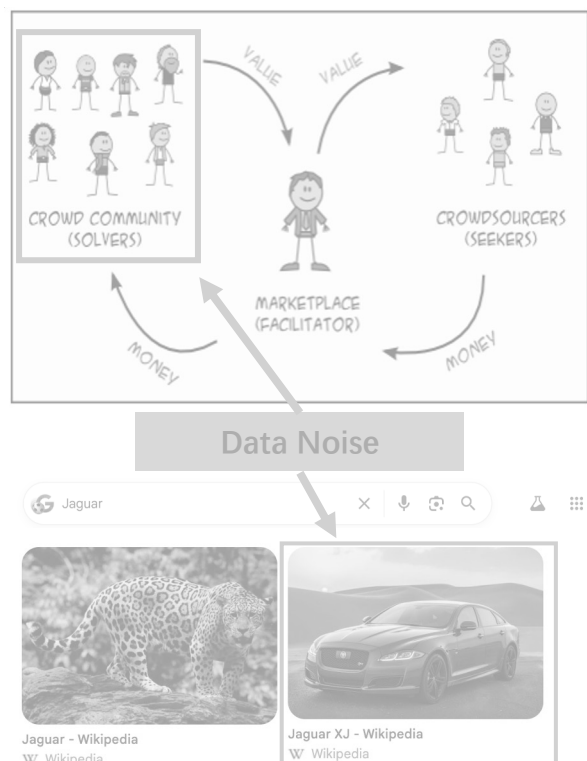
- Robust OOD detection shifts the focus from training-time empirical boundary fitting to inference-time feature purification and zero-shot semantic mapping.

Pareto Reconciliation of Empirical Utility and Causal Invariance

- The fundamental limit of OOD generalization lies in the inherent conflict between empirical utility (ERM) and causal invariance (IRM). Resolving this trade-off requires traversing the multi-objective Pareto front rather than relying on static scalar regularization.

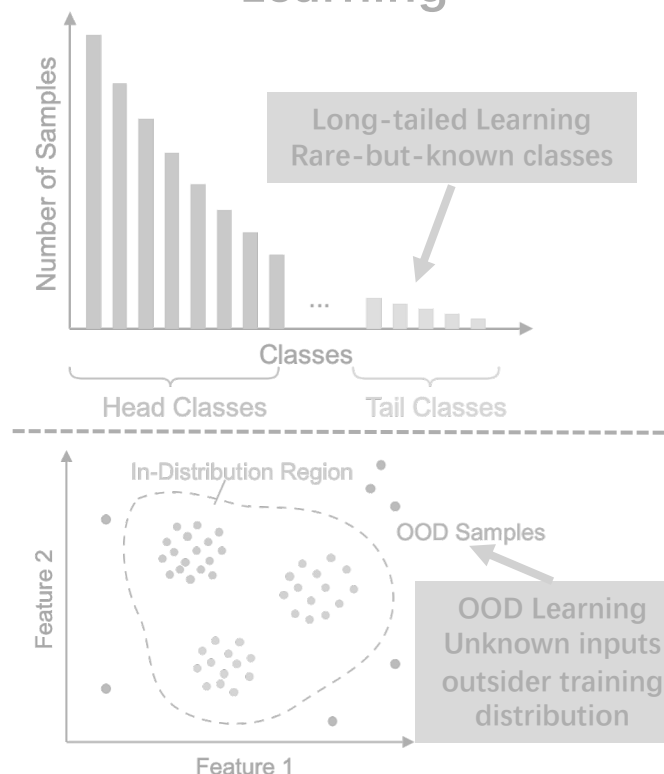
Course I: Trustworthy Data-centric Learning

Part 1. Data-noise Learning



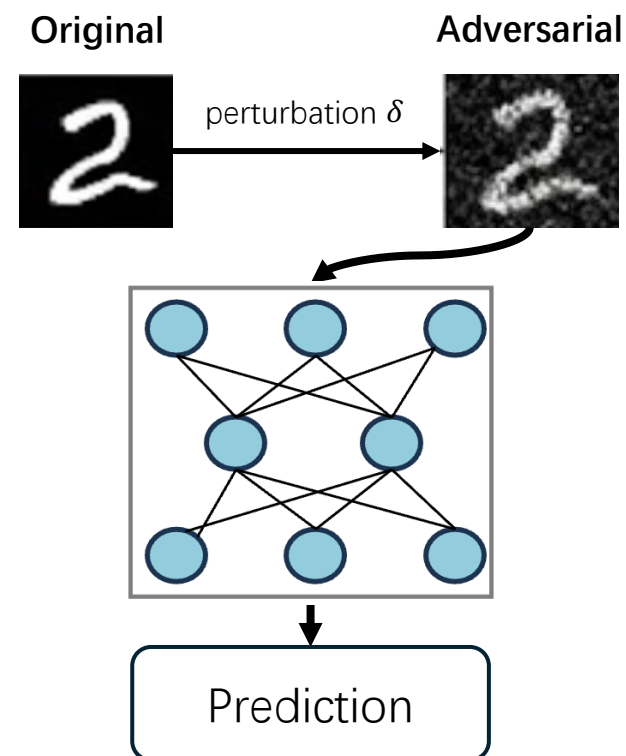
Learn from data with noisy information

Part 2. Long-tailed and Out-of-distribution Learning

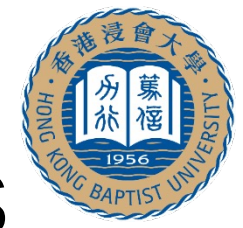


Handle imbalanced / OOD data

Part 3. Adversarial Attacks and Defenses

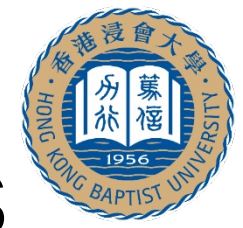


Defend adversarial attacks



Part 3 Adversarial Attacks and Defenses

- Part 3.1: Adversarial Examples and Attacks
- Part 3.2: Adversarial Training
- Part 3.3: Adversarial Purification
- Part 3.4: Adversarial Detection
- Part 3.5: Takeaways



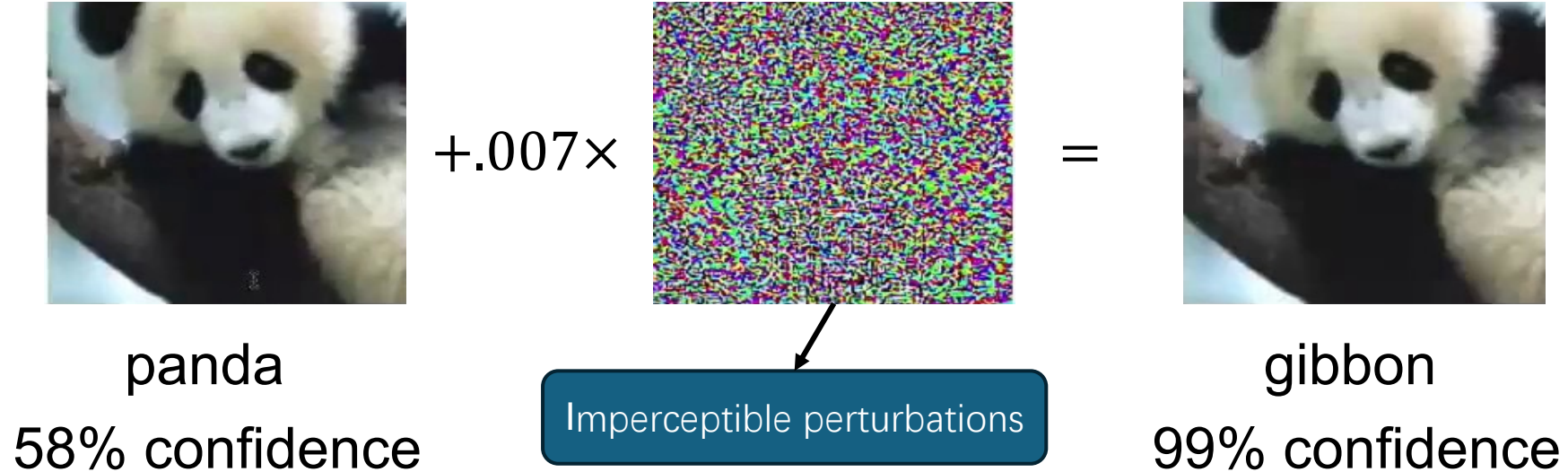
Part 3 Adversarial Attacks and Defenses

- Part 3.1: Adversarial Examples and Attacks
- Part 3.2: Adversarial Training
- Part 3.3: Adversarial Purification
- Part 3.4: Adversarial Detection
- Part 3.5: Takeaways

Adversarial Examples

Adversarial examples are generated by adding **imperceptible perturbations** to the input but may **mislead** the model to make unexpected predictions.

Panda or Gibbon?



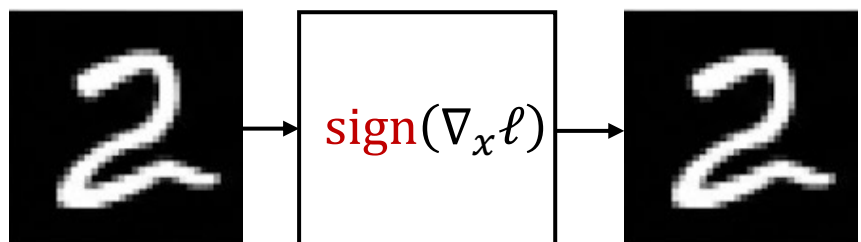
Adversarial Attacks

FGSM Attack

- Fast **Gradient Sign** Method
- **One-time** attack

$$x_{\text{adv}} = x + \epsilon \cdot \text{sign}(\nabla_x \ell(\theta, x, y))$$

(x : Original Data, x_{adv} : Adversarial Data, ϵ : Perturbation)



x (Original)

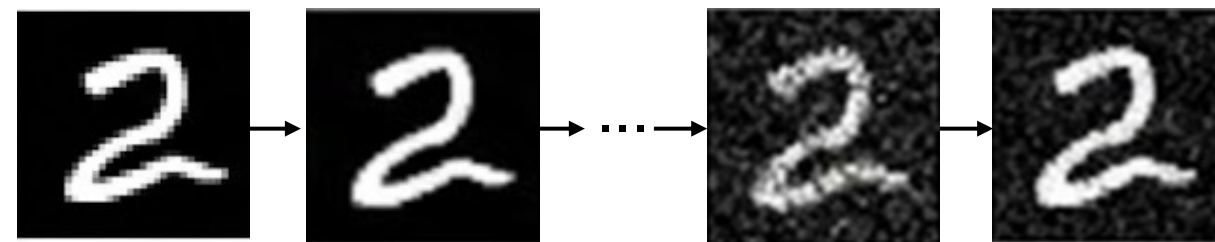
Classification: 8

PGD Attack

- Projected **Gradient Descent**
- **Iterative** attack:

$$x_t = x_{t-1} + \alpha \cdot \text{sign}(\nabla_x \ell(\theta, x_{t-1}, y))$$

- Projection: $x_t \leftarrow \text{proj}_{L_\infty}(x_t, x, \epsilon)$

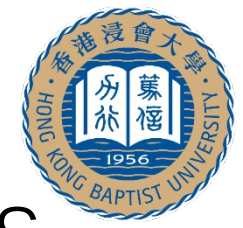


x (Original)

x_1

x_{t-1}

Classification: 8



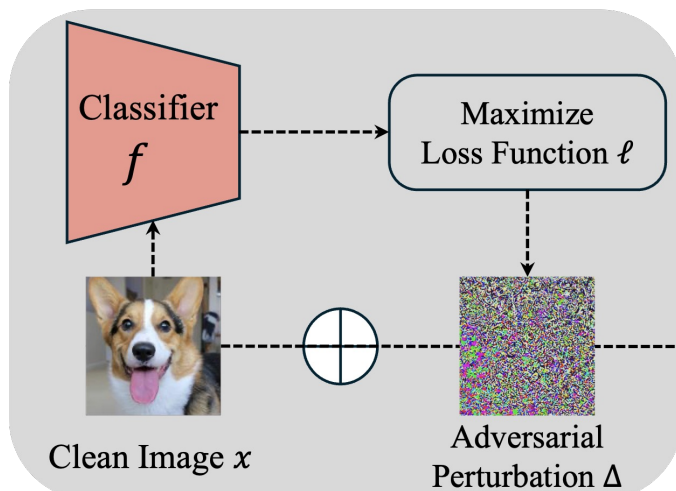
Part 3 Adversarial Attacks and Defenses

- Part 3.1: Adversarial Examples and Attacks
- Part 3.2: Adversarial Training
- Part 3.3: Adversarial Purification
- Part 3.4: Adversarial Detection
- Part 3.5: Takeaways

Adversarial Training

Adversarial training enhances the **robustness** of DNNs through introducing **adversarial data** into training and solving a **min-max optimization** problem.

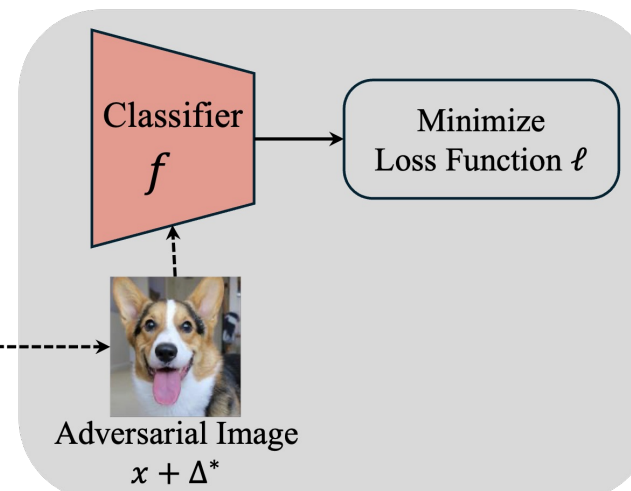
Generation of Adversarial Examples



Find the **strongest** perturbation Δ that **maximizes** ℓ .

$$\max_{\Delta} \ell(f(x + \Delta), y), \|\Delta\|_p \leq \epsilon$$

Optimization of Model Weights



Optimize model f to **resist attacks** by **minimizing** ℓ .

$$\min_{f \in F} \ell(f(x + \Delta^*), y)$$

Recent Advances in Adversarial Training

TRADES

- **Core Question:**
How to balance clean accuracy and adversarial robustness?
- **Key Idea:**
Decompose robust error into: natural error + boundary error.
- **Method:**
Optimizing a classification learning objective with robustness regularization.

FAT

- **Challenge:**
Strong adversarial attacks may over-regularize training.
- **Key Idea:**
Use more “friendly” adversarial examples to balance the training.
- **Method:**
Early-stopped PGD once the sample becomes misclassified.

CausalAdv

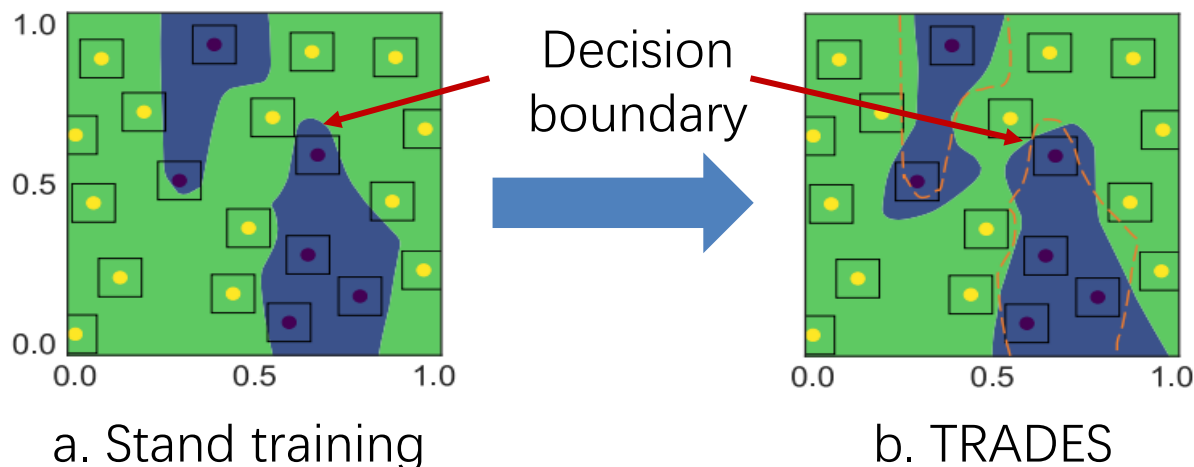
- **Insight:**
Models mainly rely on spurious correlations.
- **Key Idea:**
Learn invariant causal features and reduces reliance on spurious correlations.
- **Method:**
Align feature distributions with causal interventions.

How does **adversarial example generation** affect the adversarial training?

What makes models **vulnerable** to adversarial perturbations?

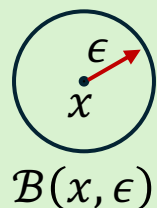
TRADES: Robustness or Accuracy?

Problem: How to achieve **robustness** to attacks without significantly **sacrificing** clean **accuracy** ?



Observation: The decision boundary shifts away, sacrificing tight fit for a **robust margin** to attacks.

Denote $\mathcal{B}(x, \epsilon) = \{x' | \|x - x'\|_p \leq \epsilon\}$ a ball of radius ϵ centered at sample x , TRADES performs



$$\min_f \mathbb{E} \left\{ \underbrace{\mathcal{L}(f(x), y)}_{\text{Accuracy}} + \underbrace{\max_{x' \in \mathcal{B}(x, \epsilon)} \mathcal{L}(f(x), f(x'))}_{\text{Robustness}} / \underbrace{\lambda}_{\text{trade-off}} \right\}$$

TRADES: Robustness or Accuracy?

Quantitative results

Table 4: Sensitivity of regularization hyperparameter λ on MNIST and CIFAR10 datasets.

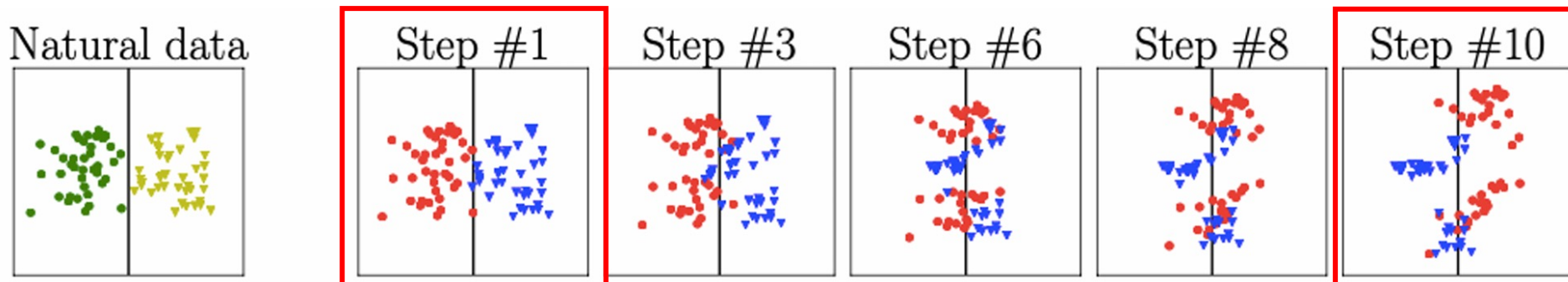
trade-off ← $1/\lambda$	MNIST		Robustness	CIFAR10		Accuracy
	$\mathcal{A}_{\text{rob}}(f)$ (%)	$\mathcal{A}_{\text{nat}}(f)$ (%)		$\mathcal{A}_{\text{rob}}(f)$ (%)	$\mathcal{A}_{\text{nat}}(f)$ (%)	
0.1	91.09 ± 0.0385	99.41 ± 0.0235	↓	26.53 ± 1.1698	91.31 ± 0.0579	↑
0.2	92.18 ± 0.0450	99.38 ± 0.0094		37.71 ± 0.6743	89.56 ± 0.2154	
0.4	93.21 ± 0.0660	99.35 ± 0.0082		41.50 ± 0.3376	87.91 ± 0.2944	
0.6	93.87 ± 0.0464	99.33 ± 0.0141		43.37 ± 0.2706	87.50 ± 0.1621	
0.8	94.32 ± 0.0492	99.31 ± 0.0205		44.17 ± 0.2834	87.11 ± 0.2123	
1.0	94.75 ± 0.0712	99.28 ± 0.0125		44.68 ± 0.3088	87.01 ± 0.2819	
2.0	95.45 ± 0.0883	99.29 ± 0.0262		48.22 ± 0.0740	85.22 ± 0.0543	
3.0	95.57 ± 0.0262	99.24 ± 0.0216		49.67 ± 0.3179	83.82 ± 0.4050	
4.0	95.65 ± 0.0340	99.16 ± 0.0205		50.25 ± 0.1883	82.90 ± 0.2217	
5.0	95.65 ± 0.1851	99.16 ± 0.0403		50.64 ± 0.3336	81.72 ± 0.0286	

Higher values of $1/\lambda$ lead to **better robust accuracy** but **lower natural accuracy**.

FAT: Friendly Adversarial Training

Question: Is the **min-max optimization** paradigm **sufficient to the adversarial training (AT)**?

Observation: Many studies find AT tends to result in **severe degradation of natural accuracy**.



In **PGD adversarial training**, adversarial variants of natural data **cross over the decision boundary** and are **located at their peer's area**.

Finding: Standard min-max AT **pushes points apart excessively**. Since models are required to fit natural and adversarial data simultaneously, a **severe accuracy degradation** happens.

Intuition: Replace **inner max** with **inner min** to keep samples **close to** the original data distribution.

$$\tilde{x}_i = \arg \max_{x \in B(x_i)} \ell(f(\tilde{x}), y_i)$$

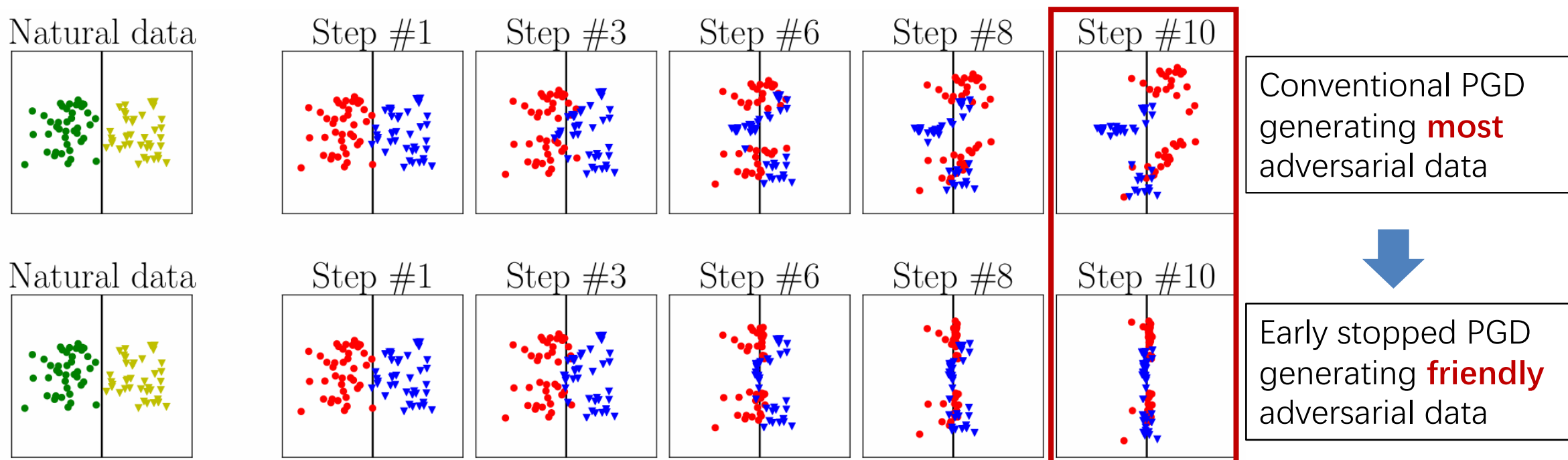


$$\begin{aligned} \tilde{x}_i &= \arg \min_{\tilde{x} \in B(x_i)} \ell(f(\tilde{x}), y_i) \\ \text{s.t. } \ell(f(\tilde{x}), y_i) - \min_{y \in \mathcal{Y}} \ell(f(\tilde{x}), y_i) &\geq \rho \end{aligned}$$

ensure $\tilde{x}$ is **misclassified** with **margin ρ**

FAT: Friendly Adversarial Training

Implementation: FAT employs the **friendly adversarial data** generated by **early stopped PGD** to avoid the excessive cross-over problem and update the model accordingly.



Finding: FAT mitigates the severe mixing of data points along the decision boundary.

FAT: Friendly Adversarial Training

Quantitative results

Table 1. Evaluations (test accuracy) of deep models (WRN-32-10) on CIFAR-10 dataset

Defense	Natural	FGSM	PGD-20	C&W $_{\infty}$	PGD-100
Madry	87.30	56.10	45.80	46.80	-
CAT	77.43	57.17	46.06	42.28	-
DAT	85.03	63.53	48.70	47.27	-
FAT ($\epsilon_{train} = 8/255$)	89.34 $\pm$ 0.221	65.52 $\pm$ 0.355	46.13 $\pm$ 0.409	46.82 $\pm$ 0.517	45.31 $\pm$ 0.531
FAT ($\epsilon_{train} = 16/255$)	87.00 $\pm$ 0.203	65.94 $\pm$ 0.244	49.86 $\pm$ 0.328	48.65 $\pm$ 0.176	49.56 $\pm$ 0.255

Table 2. Evaluations (test accuracy) of deep models (WRN-34-10) on CIFAR-10 dataset

Defense	Natural	FGSM	PGD-20	C&W $_{\infty}$	PGD-100
TRADES ($\beta = 1.0$)	88.64	56.38	49.14	-	-
FAT for TRADES ($\epsilon_{train} = 8/255$)	89.94 $\pm$ 0.303	61.00 $\pm$ 0.418	49.70 $\pm$ 0.653	49.35 $\pm$ 0.363	48.35 $\pm$ 0.240
TRADES ($\beta = 6.0$)	84.92	61.06	56.61	54.47	55.47
FAT for TRADES ($\epsilon_{train} = 8/255$)	86.60 $\pm$ 0.548	61.97 $\pm$ 0.570	55.98 $\pm$ 0.209	54.29 $\pm$ 0.173	55.34 $\pm$ 0.291
FAT for TRADES ($\epsilon_{train} = 16/255$)	84.39 $\pm$ 0.030	61.73 $\pm$ 0.131	57.12 $\pm$ 0.233	54.36 $\pm$ 0.177	56.07 $\pm$ 0.155

FAT can **improve** standard **accuracy** while **maintaining** the superior adversarial **robustness**.

CausalAdv: Causal Adversarial Robustness

□ Causal view on adversarial data

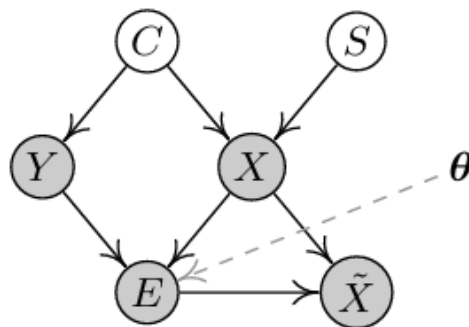


Figure 1: Causal graph of the perturbed data generation process. $C, S, X, Y, E, \tilde{X}, \theta$ are content variable, **style** variable, natural data, **label**, perturbation, perturbed data and parameters of a neural network, respectively.

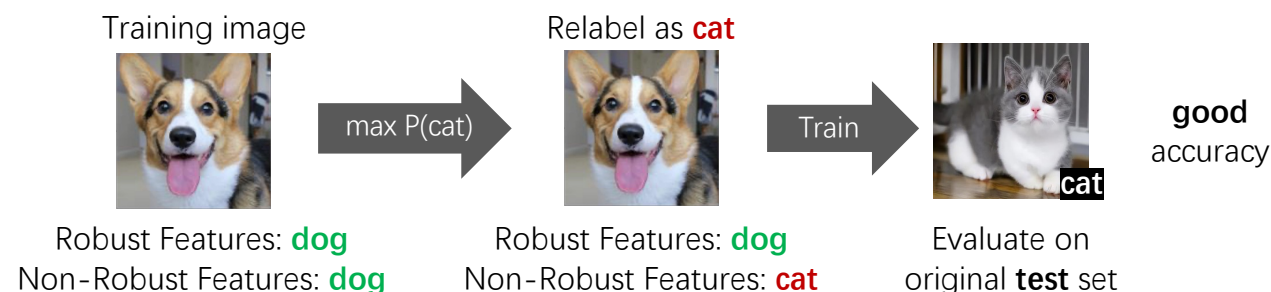
□ Soft intervention:

$$E_{\text{adv}} = \arg \max_{E' \in \mathcal{B}} \ell(h(X + E'; \theta), Y)$$

□ Correlation between **styles** and **labels**



Given X (dog), the path S (cat) $\rightarrow Y$ (cat) leads to the **spurious** correlation:



Given $P(X, Y)$, $P_{\theta}(\tilde{X}, Y)$, difference comes from the **conditional association**:

Modifying style is not allowed

$$P(Y | X) = \sum_{s \in \mathcal{S}} \underbrace{P(s | X)}_{\text{Making the difference}} \underbrace{P(Y | X, s)}_{\text{Making the difference}} \quad P_{\theta}(Y | \tilde{X}) = \sum_{s \in \mathcal{S}} \underbrace{P_{\theta}(s | \tilde{X})}_{\text{Making the difference}} \underbrace{P_{\theta}(Y | \tilde{X}, s)}_{\text{Making the difference}}$$

Adversarial Distribution Alignment

- Aligning the adversarial distribution:

$$\min_{\theta} d\left(P(Y | X), \underbrace{P_{\theta}(Y | \tilde{X})}_{\text{non-differentiable}}\right) + \lambda \mathbb{E}_s d\left(P(Y | X, s), \underbrace{P_{\theta}(Y | \tilde{X}, s)}_{\text{non-differentiable}}\right)$$

- Introducing relation and approximation (by triangle inequality):

$$\begin{aligned} \min_{\theta, W_g} \mathbb{E}_{(X, Y) \sim P(X, Y)} \text{CE}(h(X + E_{\text{adv}}; \theta), Y) + \gamma \text{CE}(h(X; \theta), Y) & \leftarrow h \text{ is output distribution of networks} \\ + \lambda \left(\mathbb{E}_s \text{CE}(g(s(X + E_{\text{adv}}); W_g), Y) + \beta \text{CE}(g(s(X); W_g), Y) \right) & \leftarrow g \text{ shares the features of networks with } h \end{aligned}$$

- Replacing with an upper bound:

$$\mathbb{E}_s \text{CE}(g(s(X); W_g), Y) \approx \mathbb{E}_{\hat{s}(X) \sim \mathcal{N}(\mu(X), \sigma^2 I)} \text{CE}(g(\hat{s}(X); W_g), Y)$$

Adversarial Distribution Alignment

- The realization of causal-inspired adversarial distribution alignment

$$\min_{\theta, W_g} \mathbb{E}_{(X,Y) \sim P(X,Y)} \text{CE}(h(X + E_{\text{adv}}; \theta), Y) + \gamma \text{CE}(h(X; \theta), Y) \\ + \lambda \left(\text{CE}(\bar{g}(\mu(X + E_{\text{adv}}); W_g), Y) + \beta \text{CE}(\bar{g}(\mu(X); W_g), Y) \right)$$

Without considering the spurious correlation of s and y , we achieve Madry's AT



$$\min_{\theta} \rho(\theta), \text{ where } \rho(\theta) = \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[\max_{\delta \in \mathcal{S}} L(\theta, x + \delta, y) \right]$$



With considering the soft label due to model capacity, we achieve TRADES

$$\min_f \mathbb{E} \left\{ \mathcal{L}(f(\mathbf{X}), \mathbf{Y}) + \max_{\mathbf{X}' \in \mathbb{B}(\mathbf{X}, \epsilon)} \mathcal{L}(f(\mathbf{X}), f(\mathbf{X}')) / \lambda \right\}$$

Adversarial Distribution Alignment

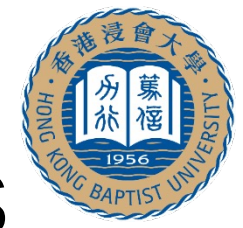
Table 2: Classification accuracy (%) of ResNet-18 on CIFAR-10 under the white-box threat model with $\epsilon = 8/255$. The best-performance model and the corresponding accuracy are highlighted.

Method	Best checkpoint				Last checkpoint			
	Natural	FGSM	PGD-20	CW-20	Natural	FGSM	PGD-20	CW-20
Madry	83.56	56.69	51.92	51.00	84.65	54.37	46.38	46.73
CausalAdv-M	80.42 $\pm$ 0.39	57.98 $\pm$ 0.21	54.44 $\pm$ 0.18	52.51 $\pm$ 0.25	83.72 $\pm$ 0.41	59.17 $\pm$ 0.24	51.82 $\pm$ 0.19	50.93 $\pm$ 0.27
TRADES	81.39	57.25	53.64	51.39	82.91	57.95	52.80	51.27
CausalAdv-T	81.22 $\pm$ 0.27	58.97 $\pm$ 0.17	54.55 $\pm$ 0.16	52.95 $\pm$ 0.26	81.62 $\pm$ 0.30	58.90 $\pm$ 0.16	53.64 $\pm$ 0.14	52.70 $\pm$ 0.37

Table 3: Classification accuracy (%) of ResNet-18 on CIFAR-100 under the white-box threat model with $\epsilon = 8/255$. The best-performance model and the corresponding accuracy are highlighted.

Method	Best checkpoint				Last checkpoint			
	Natural	FGSM	PGD-20	CW-20	Natural	FGSM	PGD-20	CW-20
Madry	55.98	28.39	25.15	24.04	55.08	25.35	21.63	21.42
CausalAdv-M	54.07 $\pm$ 0.17	29.76 $\pm$ 0.16	27.62 $\pm$ 0.13	25.44 $\pm$ 0.15	54.81 $\pm$ 0.23	26.83 $\pm$ 0.19	23.34 $\pm$ 0.15	22.93 $\pm$ 0.13
TRADES	53.85	29.04	27.91	24.09	53.54	29.29	26.80	23.79
CausalAdv-T	53.17 $\pm$ 0.39	30.66 $\pm$ 0.20	28.57 $\pm$ 0.18	25.74 $\pm$ 0.18	54.79 $\pm$ 0.41	30.81 $\pm$ 0.40	28.51 $\pm$ 0.35	25.32 $\pm$ 0.27

Addressing spurious correlations yields SOTA robustness across all attacks!



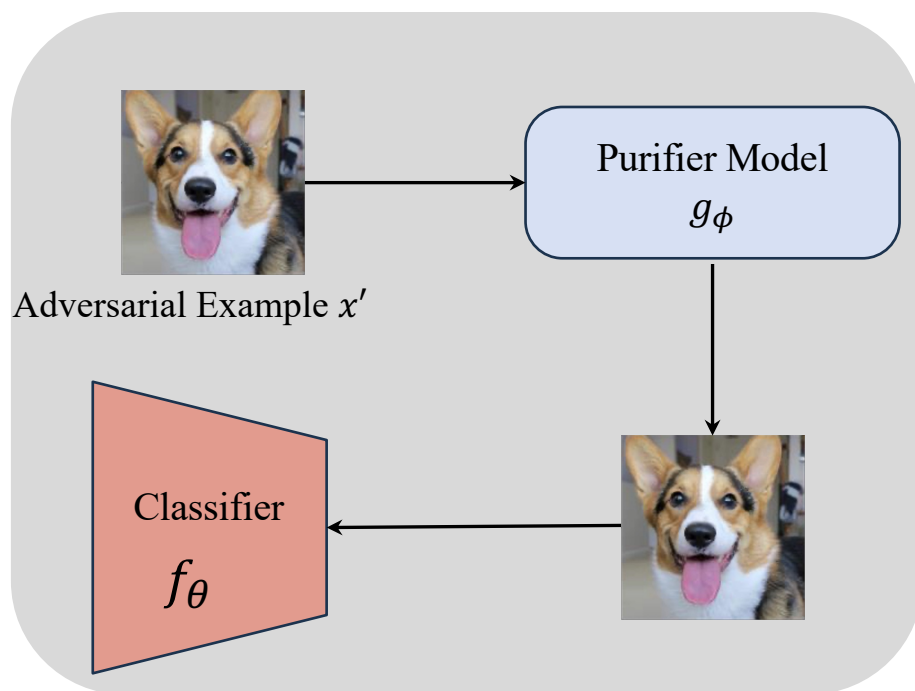
Part 3 Adversarial Attacks and Defenses

- Part 3.1: Adversarial Examples and Attacks
- Part 3.2: Adversarial Training
- Part 3.3: Adversarial Purification
- Part 3.4: Adversarial Detection
- Part 3.5: Takeaways

Adversarial Purification

Adversarial purification aims at achieving **input level robustness** by adapting **denoising** operations to **recover** clean data from the adversarial examples.

Purification Process



- The input is preprocessed by a purifier module g_ϕ before classification

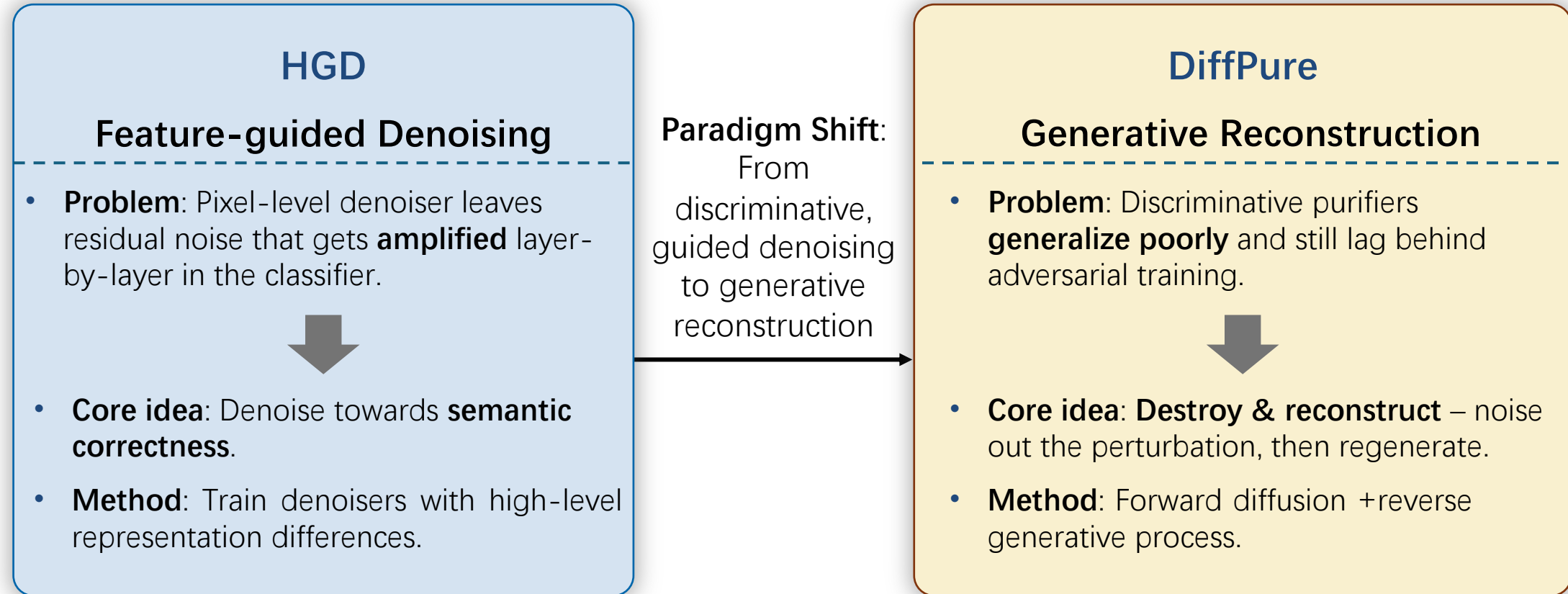
$$\min_{\phi, \theta} \mathbb{E}_{p_{\text{data}}(x, y)} \left[\max_{x' \in \mathcal{B}(x, \epsilon)} \mathcal{L}(f_\theta(g_\phi(x')), y) \right]$$

($\mathcal{B}(x, \epsilon)$ is the perturbation ball)

- Compared to adversarial training:

- Input-level Denoising
- Plug-and-play Purifier
- Attack-agnostic Defense Strategy

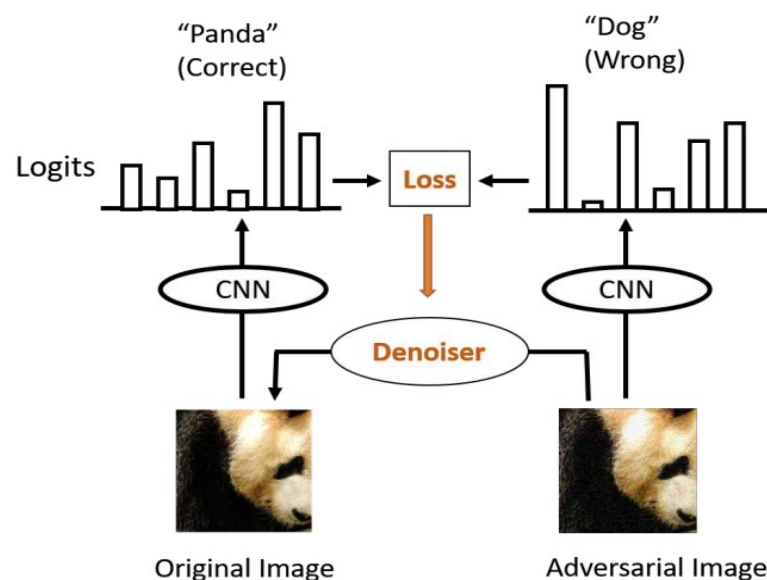
Adversarial Purification



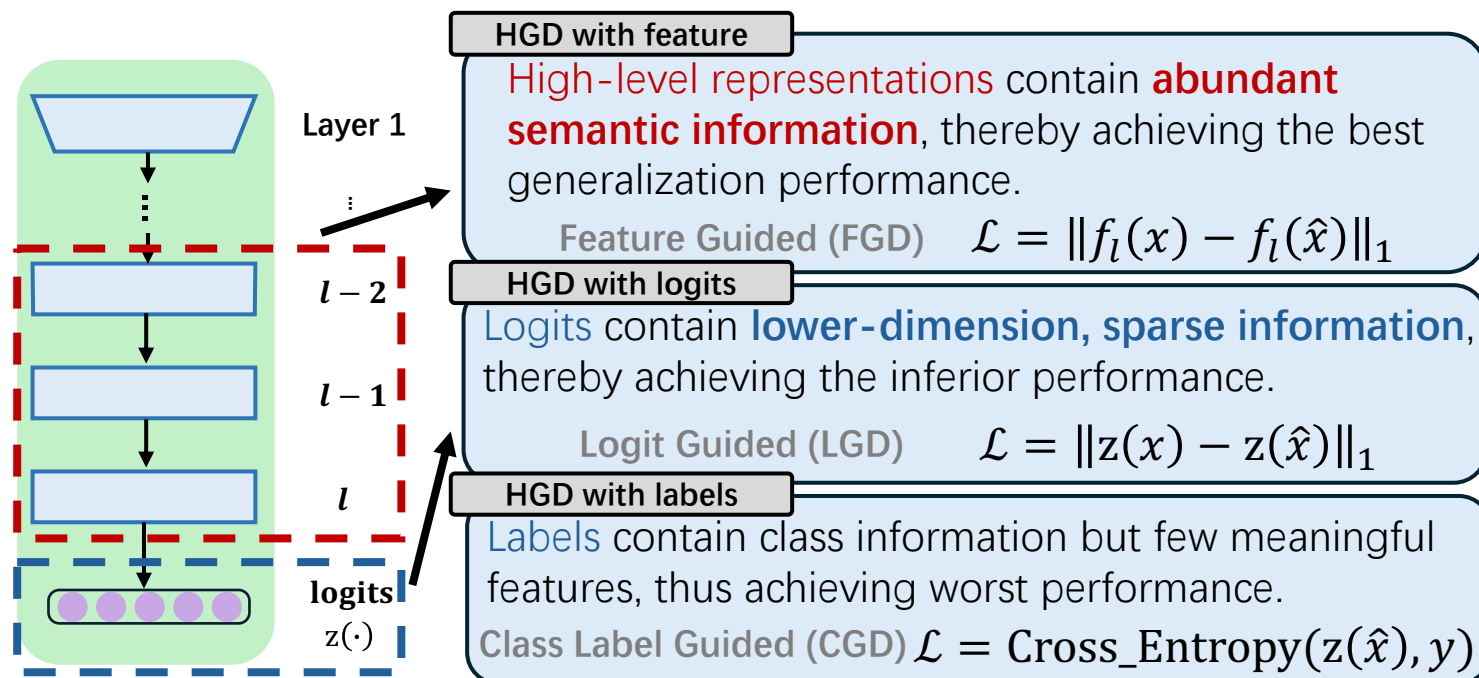
HGD: Representation-level Purification

■ **Perturbation amplification effect:** Adversarial perturbation tends to be **amplified** in **high-level representations** by CNNs, although it is imperceptible.

■ **Key idea: HGD** overcomes this with a loss defined as **the difference** between the **high-level representations** of the clean and denoised data from target models.



High-level representation guided denoiser



HGD: Representation-level Purification

Defense	Clean	WhiteTestSet		BlackTestSet	
		$\epsilon = 4$	$\epsilon = 16$	$\epsilon = 4$	$\epsilon = 16$
NA	76.7%	14.5%	14.4%	61.2%	41.0%
PGD	75.3%	20.0%	13.8%	67.5%	55.7%
ensV3 [31]	76.9%	69.8%	58.0%	72.4%	62.0%
FGD	76.1%	73.7%	67.4%	74.3%	71.8%
LGD	76.2%	75.2%	69.2%	75.1%	72.2%
CGD	74.9%	75.8%	73.2%	74.5%	71.1%

Denoiser for Resnet	Clean	WhiteTestSet		BlackTestSet	
		$\epsilon = 4$	$\epsilon = 16$	$\epsilon = 4$	$\epsilon = 16$
NA	78.5%	63.3%	38.4%	67.8%	48.6%
IncV3 guided LGD	77.4%	75.8%	71.7%	76.1%	72.7%
Resnet guided LGD	78.4%	76.1%	72.9%	76.5%	74.6%

Defense	Clean	WhiteTestSet		BlackTestSet	
		$\epsilon = 4$	$\epsilon = 16$	$\epsilon = 4$	$\epsilon = 16$
NA	76.6%	15.4%	15.3%	61.5%	41.7%
LGD	76.3%	73.9%	65.7%	74.8%	72.2%

High Robust

All 3 HGD methods **outperform** PGD and ensV3 for all attacks with slightly decreased clean accuracy on ImageNet.

High Transferability

- **Cross-model:** HGD achieves very close defending performance as the Resnet guided LGD.
- **Cross-dataset:** Although the 250 classes in the test set are **never** trained, LGD still learns to defend against the attacks targeting at them.

DiffPure: Diffusion-based Purification

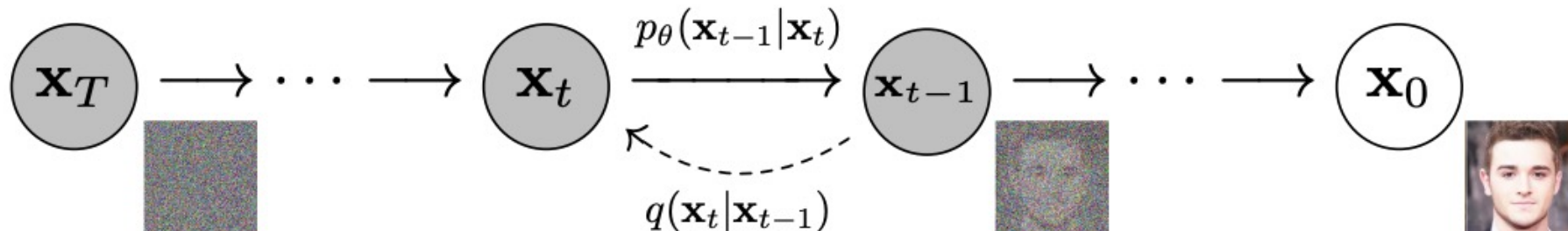
- ❑ **Problem:** Existing purification methods often falls behind adversarial training methods due to the **shortcomings of generative models**.

mode collapse: lacks diversity, failing to cover all modes of clean data distribution.

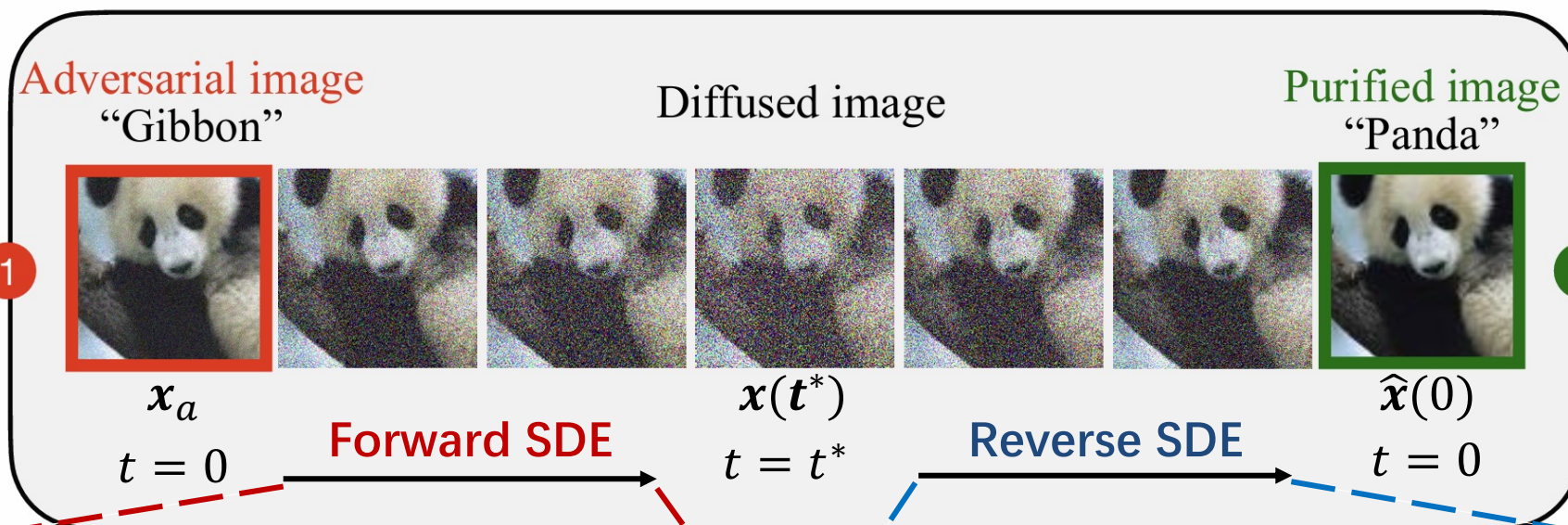
lack proper stochasticity: vulnerable to adaptive attacks.

- ❑ **Diffusion model is an ideal purification model**

- The high generation quality and diversity ensure the purified images comprehensively **cover all modes** of the **original distribution of clean data**.
- The **stochasticity** in diffusion process can make a **powerful stochastic defense**.



DiffPure: Diffusion-based Purification



Forward SDE

Add noise to adversarial images by following the forward process with a small diffusion timestep t^*

$$x(t^*) = \sqrt{\alpha(t^*)}x_a + \sqrt{1 - \alpha(t^*)}\epsilon$$

Reverse SDE

Recover clean images through the reverse denoising process before classification

$$\hat{x}(0) = \text{SDE_Solver}(x(t^*), f_{\text{rev}}, g_{\text{rev}}, \bar{w}, t^*, 0)$$

The adversarial attacks are removed together with noise during the reverse SDE phase.

DiffPure: Diffusion-based Purification

Method	Extra Data	Standard Acc	Robust Acc
WideResNet-28-10			
(Zhang et al., 2020)	✓	89.36	59.96
(Wu et al., 2020)	✓	88.25	62.11
(Gowal et al., 2020)	✓	89.48	62.70
(Wu et al., 2020)	✗	85.36	59.18
(Rebuffi et al., 2021)	✗	87.33	61.72
(Gowal et al., 2021)	✗	87.50	65.24
Ours	✗	89.02±0.21	70.64±0.39
WideResNet-70-16			
(Gowal et al., 2020)	✓	91.10	66.02
(Rebuffi et al., 2021)	✓	92.23	68.56
(Gowal et al., 2020)	✗	85.29	59.57
(Rebuffi et al., 2021)	✗	88.54	64.46
(Gowal et al., 2021)	✗	88.74	66.60
Ours	✗	90.07±0.97	71.29±0.55

Experiments on CIFAR-10
AutoAttack ℓ_∞ ($\epsilon = 8/255$)

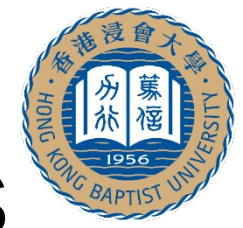
Method	Extra Data	Standard Acc	Robust Acc
WideResNet-28-10			
(Augustin et al., 2020)*	✓	92.23	77.93
(Rony et al., 2019)	✗	89.05	66.41
(Ding et al., 2020)	✗	88.02	67.77
(Wu et al., 2020)*	✗	88.51	72.85
(Sehwag et al., 2021)*	✗	90.31	75.39
(Rebuffi et al., 2021)	✗	91.79	78.32
Ours	✗	91.03±0.35	78.58±0.40
WideResNet-70-16			
(Gowal et al., 2020)	✓	94.74	79.88
(Rebuffi et al., 2021)	✓	95.74	81.44
(Gowal et al., 2020)	✗	90.90	74.03
(Rebuffi et al., 2021)	✗	92.41	80.86
Ours	✗	92.68±0.56	80.60±0.57

Experiments on CIFAR-10
AutoAttack ℓ_2 ($\epsilon = 0.5$)

Method	Extra Data	Standard Acc	Robust Acc
ResNet-50			
(Engstrom et al., 2019)	✗	62.56	31.06
(Wong et al., 2020)	✗	55.62	26.95
(Salman et al., 2020)	✗	64.02	37.89
(Bai et al., 2021) [†]	✗	67.38	35.51
Ours	✗	67.79±0.43	40.93±1.96
WideResNet-50-2			
(Salman et al., 2020)	✗	68.46	39.25
Ours	✗	71.16±0.75	44.39±0.95
DeiT-S			
(Bai et al., 2021) [†]	✗	66.50	35.50
Ours	✗	73.63±0.62	43.18±1.27

Experiments on ImageNet
AutoAttack ℓ_∞ ($\epsilon = 4/255$)

Observations: In defense of various strong adaptive attacks on different classifier architectures, DiffPure **largely outperforms** previous approaches.



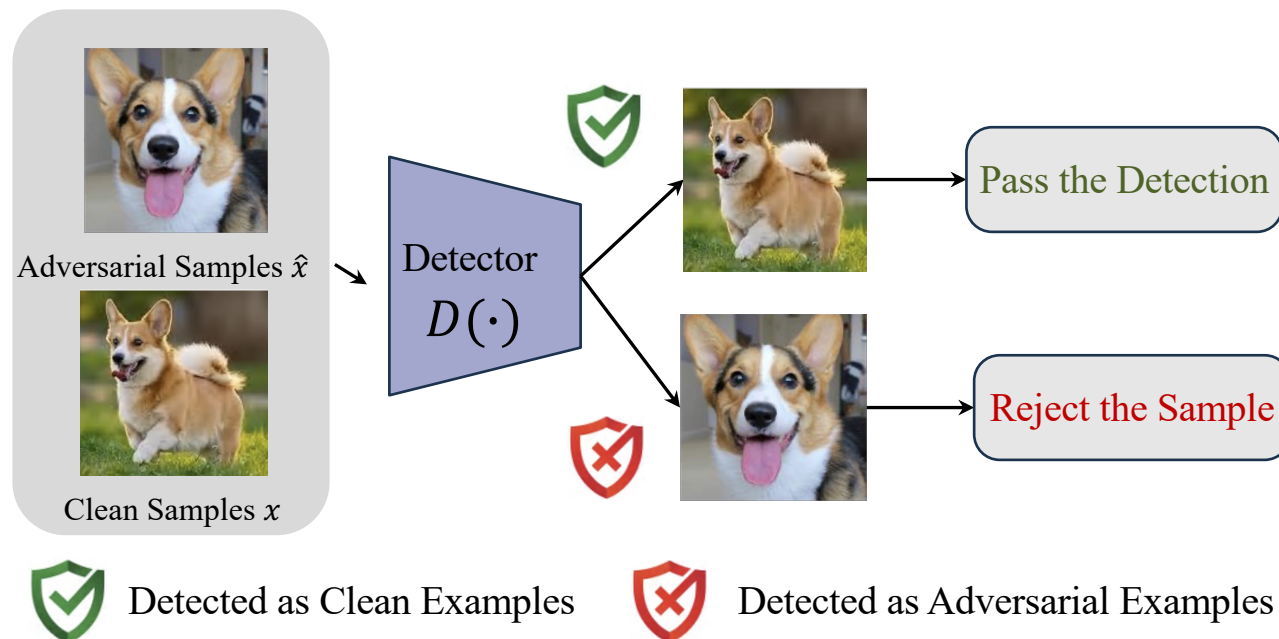
Part 3 Adversarial Attacks and Defenses

- Part 3.1: Adversarial Examples and Attacks
- Part 3.2: Adversarial Training
- Part 3.3: Adversarial Purification
- Part 3.4: Adversarial Detection
- Part 3.5: Takeaways

Adversarial Detection

Adversarial detection **identifies** whether a test sample comes from a **clean** or **adversarial** data distribution.

Detection of Adversarial Examples (AEs)



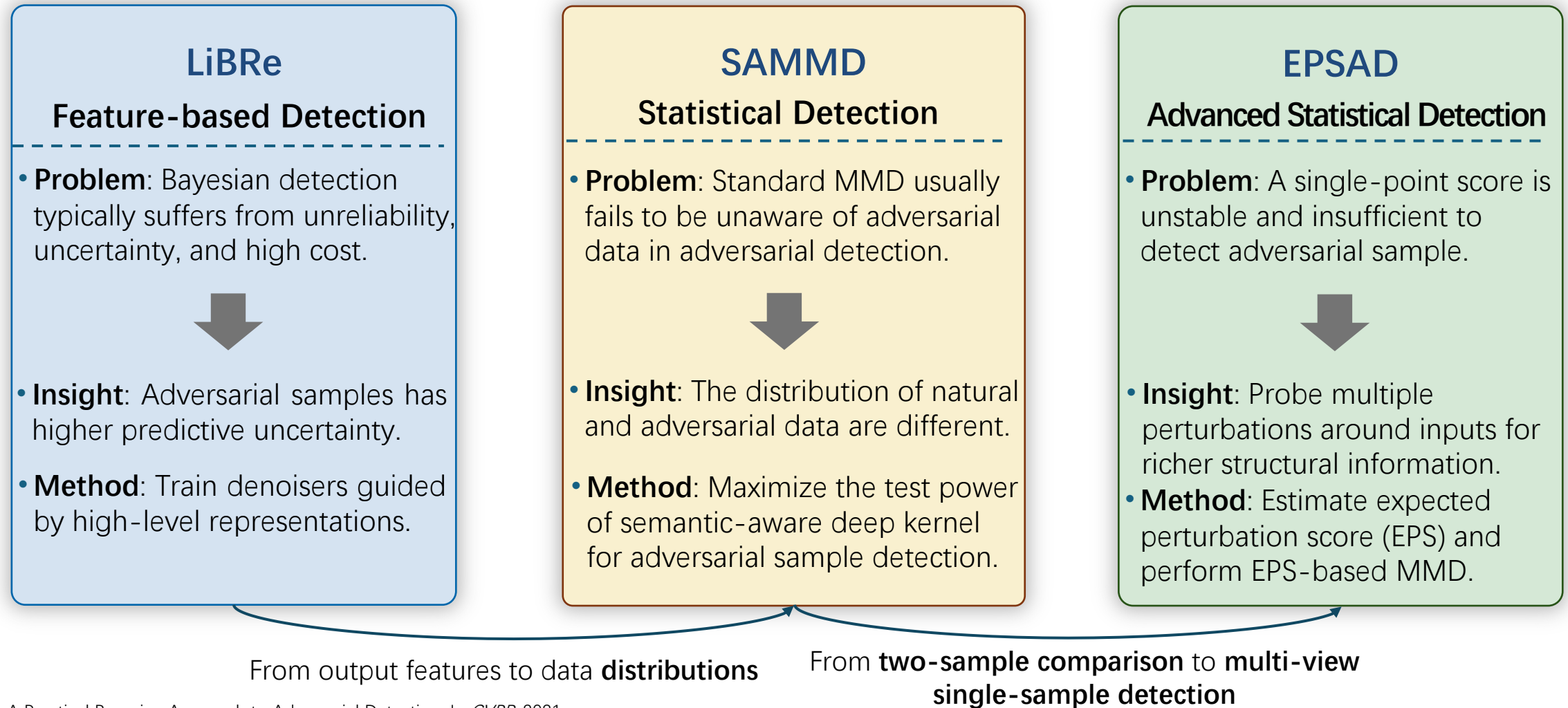
□ Detect & Reject

$$D(x) = \begin{cases} 1, & \text{if } x \text{ is an AE} \\ 0, & \text{if } x \text{ is a clean sample} \end{cases}$$

□ Advantages

- Preserves Clean Accuracy
- Model-Agnostic Add-On
- Plug-and-Play Defense

Adversarial Detection



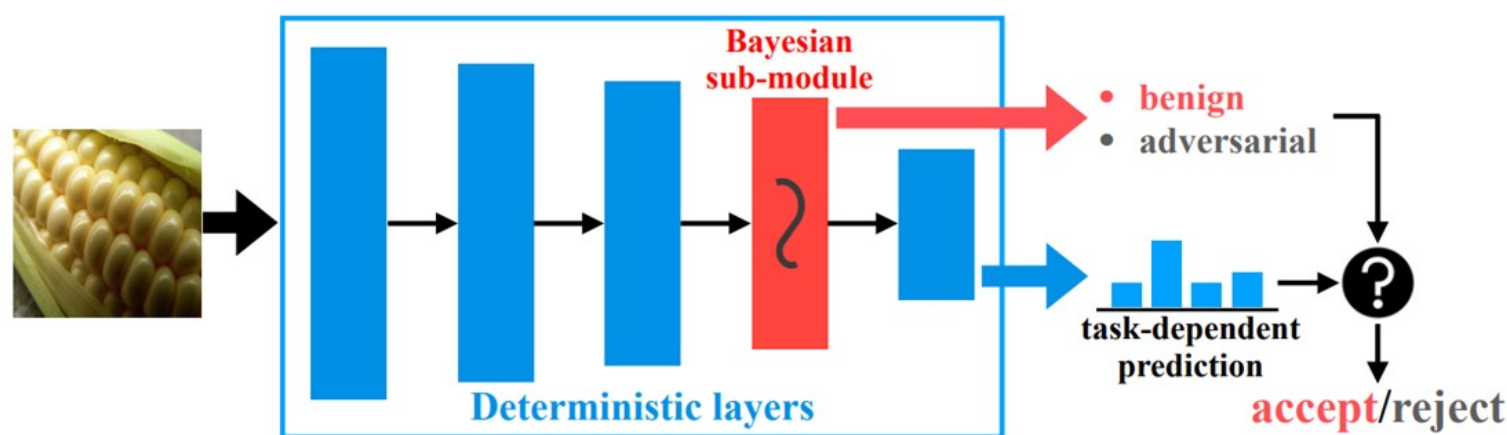
LiBRe: A Practical Bayesian Approach to Adversarial Detection. In *CVPR*, 2021.

Maximum mean discrepancy test is aware of adversarial attacks. In *ICML*, 2021.

Detecting Adversarial Data by Probing Multiple Perturbations Using Expected Perturbation Score. In *ICML*, 2023.

<https://bhanml.github.io/> & <https://github.com/tmlr-group> 105

LiBRe: Feature-based Detection



Deterministic layers: extract general features from the input.

Bayesian sub-module: last few layers are converted to Bayesian (e.g., BNN with MC Dropout) and trained on a mixture of benign and adversarial examples.

Given a pre-trained DNN, LiBRe converts its last few layers to be **Bayesian** and uses the **uncertainty**, i.e., **an unbiased variance estimator**, as a detection metric:

$$U(\mathbf{x}) = \frac{1}{T-1} \left[\sum_{t=1}^T \|z^{(t)}\|_2^2 - T \left(\left\| \frac{1}{T} \sum_{t=1}^T z^{(t)} \right\|_2^2 \right) \right]$$

Empirical Expectation

variance

where $z^{(t)}$ is the t -th logits of the BNN.

Feature-based Detection: LiBRe

Method	Prediction accuracy $\uparrow$		AUROC of adversarial detection under <i>model transfer</i> $\uparrow$			
	TOP1	TOP5	PGD	MIM	TIM	DIM
<i>MAP</i>	76.13%	92.86%	-	-	-	-
<i>MC dropout</i> [17]	74.86%	92.33%	0.660	0.723	0.695	0.605
<i>LMFVI</i>	76.06%	92.92%	0.125	0.200	0.510	0.018
<i>MFVI</i>	75.24%	92.58%	0.241	0.205	0.504	0.150
<i>LiBRe</i>	76.19%	92.98%	1.000	1.000	0.982	1.000

Table 1: Left: comparison on accuracy. Right: comparison on AUROC of adversarial detection under *model transfer*. (ImageNet)

Method	FGSM	BIM	C&W	PGD	MIM	TIM	DIM	FGSM- ℓ_2	BIM- ℓ_2	PGD- ℓ_2
<i>KD</i> [14]	0.639	<u>1.000</u>	<u>0.999</u>	<u>1.000</u>	<u>1.000</u>	<u>0.999</u>	0.624	0.633	<u>1.000</u>	<u>1.000</u>
<i>LID</i> [39]	0.846	<u>0.999</u>	<u>0.999</u>	<u>0.999</u>	<u>0.997</u>	<u>0.999</u>	0.762	0.846	<u>0.999</u>	<u>0.999</u>
<i>MC dropout</i> [17]	0.607	<u>1.000</u>	<u>0.980</u>	<u>1.000</u>	<u>1.000</u>	<u>0.999</u>	0.628	0.577	<u>0.999</u>	<u>0.999</u>
<i>LMFVI</i>	0.029	<u>0.992</u>	0.738	0.943	<u>0.996</u>	<u>0.997</u>	0.021	0.251	<u>0.993</u>	0.946
<i>MFVI</i>	0.102	1.000	0.780	0.992	1.000	0.999	0.298	0.358	0.952	0.935
<i>LiBRe</i>	1.000	0.984	0.985	0.994	0.996	0.994	1.000	0.995	0.983	0.993

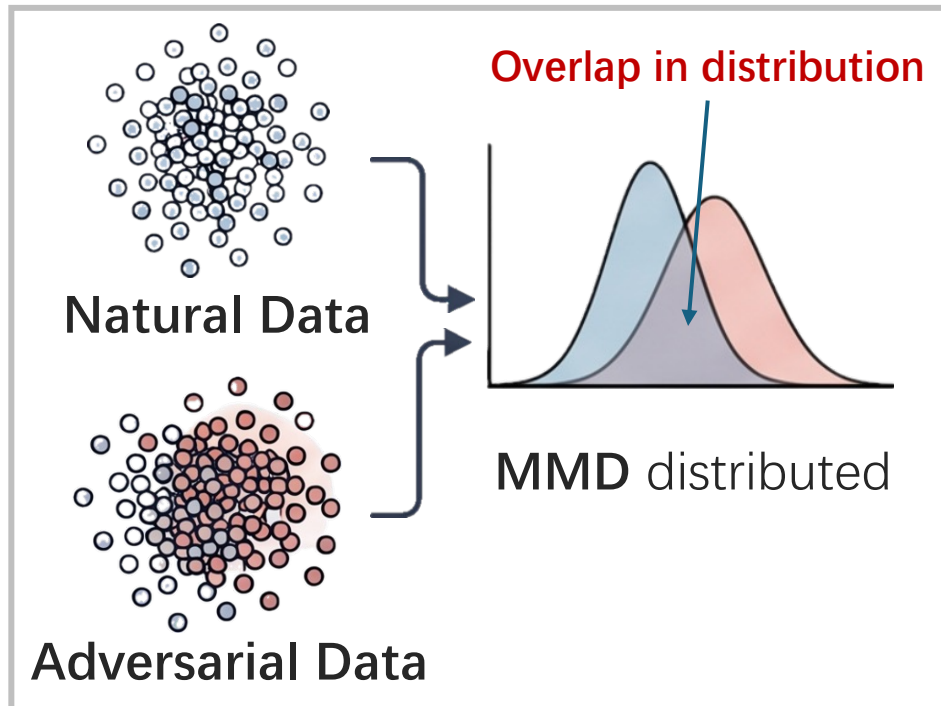
Table 2: Comparison on AUROC of adversarial detection for *regular attacks* $\uparrow$. (ImageNet)

Unlike other Bayesian methods, LiBRe maintains the **highest prediction accuracy** and **AUROC scores**, showing **superior generalization** to unseen transfer-based attacks.

SAMMD: Statistical Detection Strategy

$$\text{MMD}(\mathbb{P}, \mathbb{Q}; \mathcal{F}) := \sup_{f \in \mathcal{F}} |\mathbb{E}[f(X)] - \mathbb{E}[f(Y)]|$$

Maximum mean discrepancy (MMD) can theoretically detect any distribution shift, but empirically fails in **adversarial attacks**.



Problem 1: Limited representation extraction

- Gaussian kernels fail in **similarity measure** on complex *high-dimensional* data (e.g., images).

Problem 2: Weak detection capability

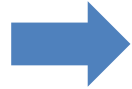
- Overlooked **optimization** of kernels for better adversarial detection capability (i.e., test power).

Problem 3: Violating i.i.d. assumption

- Natural & adversarial data are often **dependent**, and traditional permutation tests fail completely.

SAMMD: Statistical Detection Strategy

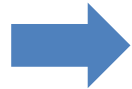
Representation



Solution 1: Semantic Kernel (Get Feature)

- **Action:** Extract **deep semantic features** instead of raw pixels.
- **Impact:** Amplify natural and adversarial data discrepancies.

Capability



Solution 2: Maximizing Test Power (Train)

- **Action:** **Adaptively optimize** kernel parameters.
- **Impact:** Maximize approximate test power (enhanced capability).

Assumption



Solution 3: Utilizing Wild Bootstrap (Test)

- **Action:** Replace Permutation with **Wild Bootstrap** resampling.
- **Impact:** Controls Type I error for Non-i.i.d. data.

SAMMD: Statistical Detection Strategy

Algorithm 1 The SAMMD Test

Input: $S_X, S_Y, \hat{f}$, various hyperparameters used below;
 $\omega \leftarrow \omega_0; \lambda \leftarrow 10^{-8}$;
 Split the data as $S_X = S_X^{tr} \cup S_X^{te}$ and $S_Y = S_Y^{tr} \cup S_Y^{te}$;
Phase 1: train the kernel parameters ω and β ; on S_X^{tr} and S_Y^{tr}
for $T = 1, 2, \dots, T_{max}$ **do**
 $S'_X \leftarrow$ minibatch from S_X^{tr} ;
 $S'_Y \leftarrow$ minibatch from S_Y^{tr} ;
 $k_\omega \leftarrow$ kernel function with parameters ω using Eq. (7);
 $M(\omega) \leftarrow \widehat{\text{SAMMD}}_u^2(S'_X, S'_Y; k_\omega)$ using Eq. (8);
 $V_\lambda(\omega) \leftarrow \hat{\sigma}_{H_1, \lambda}^2(S'_X, S'_Y; k_\omega)$ using Eq. (11);
 $\hat{J}_\lambda(\omega) \leftarrow M(\omega) / \sqrt{V_\lambda(\omega)}$ using Eq. (10);
 $\omega \leftarrow \omega + \eta \nabla_{\text{Adam}} \hat{J}_\lambda(\omega)$; *# maximize $\hat{J}_\lambda(\omega)$*
end for
Phase 2: testing with k_ω on S_X^{te} and S_Y^{te}
 $est \leftarrow \widehat{\text{SAMMD}}_b^2(S_X^{te}, S_Y^{te}; k_\omega)$
for $i = 1, 2, \dots, n_{perm}$ **do**
 Generate $\{W_i^X\}_{i=1}^n$ and $\{W_i^Y\}_{i=1}^m$ using Eq. (6);
 $\{\tilde{W}_i^X\}_{i=1}^n \leftarrow \{W_i^X\}_{i=1}^n - \frac{1}{n} \sum_{i=1}^n W_i^X$;
 $\{\tilde{W}_i^Y\}_{i=1}^m \leftarrow \{W_i^Y\}_{i=1}^m - \frac{1}{m} \sum_{i=1}^m W_i^Y$;
 $perm_i \leftarrow \frac{1}{n(n-1)} \sum_{i,j} H_{ij} \tilde{W}_i^X \tilde{W}_j^Y$; *# resample*
end for
Output: $k_\omega, est, p\text{-value: } \frac{1}{n_{perm}} \sum_{i=1}^{n_{perm}} \mathbf{1}(perm_i \geq est)$

Extract deep features from pre-trained classifiers and use these semantic features to calculate **MMD-kernel**:

$$k_\omega(\mathbf{x}, \mathbf{y}) = [(1 - \epsilon_0)s_{\hat{f}}(\mathbf{x}, \mathbf{y}) + \epsilon_0]q(\mathbf{x}, \mathbf{y}), \quad (7)$$

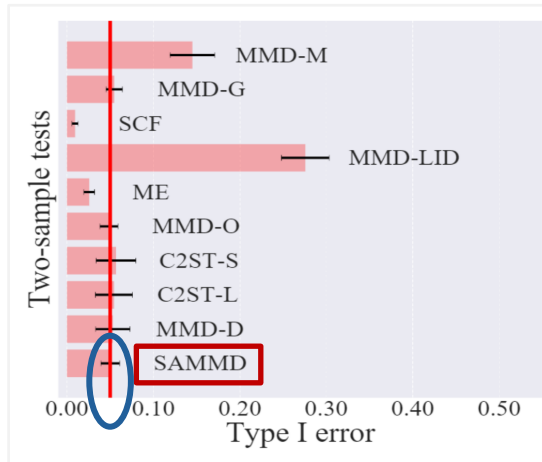
Maximize the **approximate test power** criterion under the alternative hypothesis:

$$\hat{J}_\lambda(S_X, S_Y; k_\omega) := \frac{\widehat{\text{SAMMD}}_u^2(S_X, S_Y; k_\omega)}{\hat{\sigma}_{H_1, \lambda}^2(S_X, S_Y; k_\omega)}, \quad (10)$$

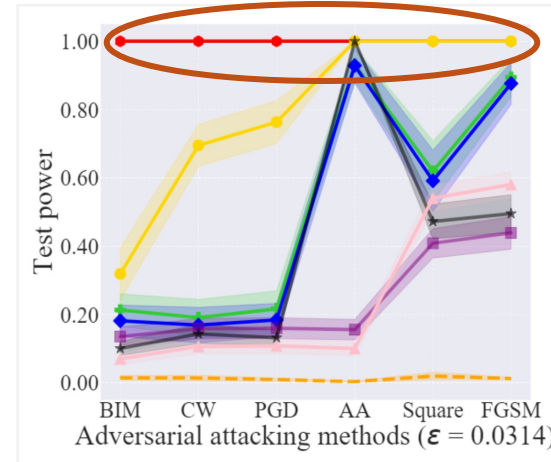
Use **wild bootstrap resampling** to simulate the correct null distribution when facing **non-I.I.D.** adversarial data:

$$W_t = e^{-1/l} W_{t-1} + \sqrt{1 - e^{-2/l}} \epsilon_t, \quad (6)$$

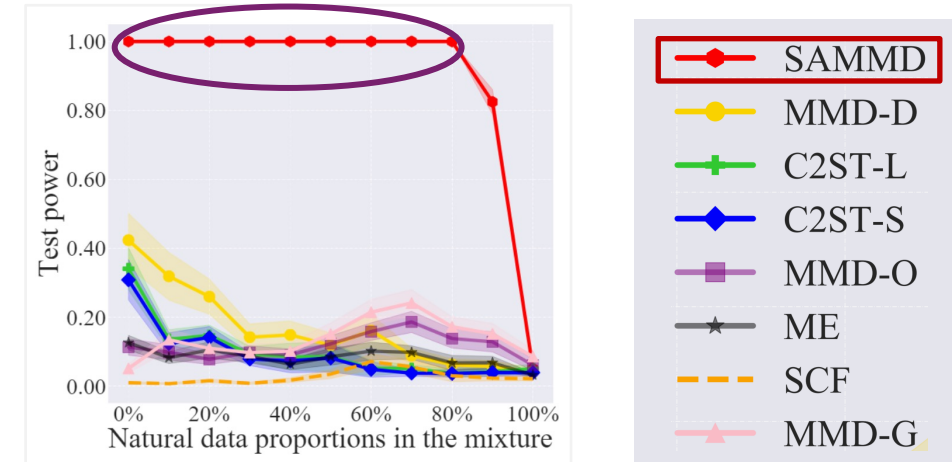
SAMMD: Statistical Detection Strategy



(a) Type I Error



(b) Test Power



(c) Robustness to mixture data

Valid & Reliable (Type I Error)

- Strictly controls False Alarms at the ideal threshold ($\alpha = 0.05$)

SOTA Detection Rate (Test Power)

- Comprehensively outperforms baselines (MMD-D, C2ST) across six different attacks

High Robustness

- Highly stable across various mixture adv and natural

EPSAD: Statistical Detection Strategy

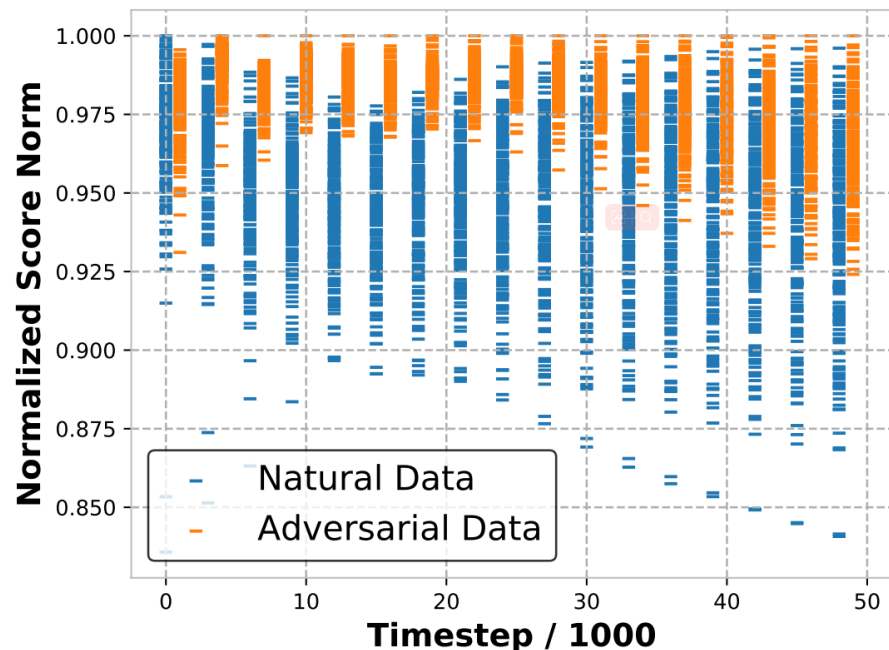


Figure. Score norm of samples across a diffusion process

Motivation: *Score* $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ represents the momentum of the sample towards high density of natural data



Lower $\|\nabla_{\mathbf{x}} \log p(\mathbf{x})\|$ means more likely **high-density** regions

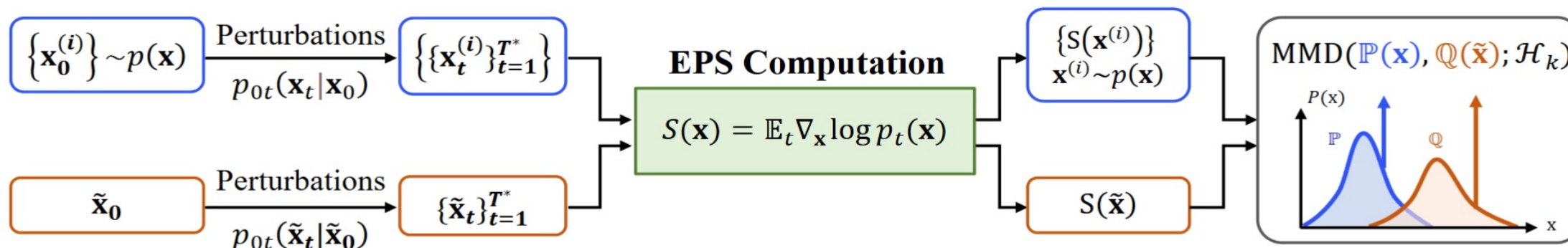
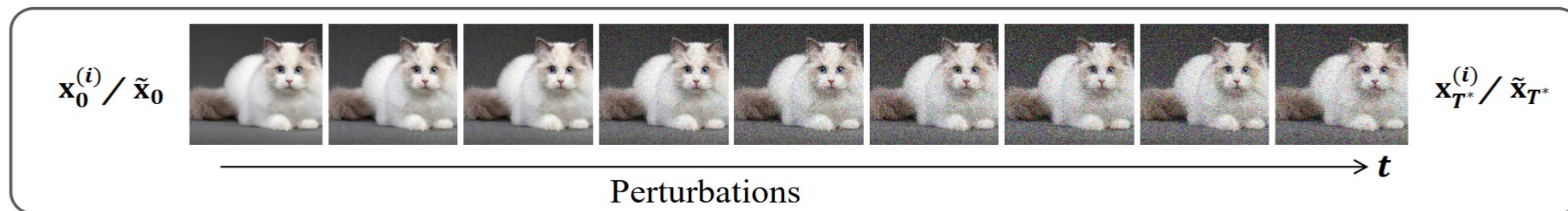
□ **Observation:** Although **individual** $\|\nabla_{\mathbf{x}} \log p(\mathbf{x})\|$ is discriminative, it is **not effective** enough.



□ **New statistic: Expected Perturbation Score (EPS)**

$$S(\mathbf{x}) = \mathbb{E}_{t \sim U(0, T)} \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$$

EPSAD: Statistical Detection Strategy

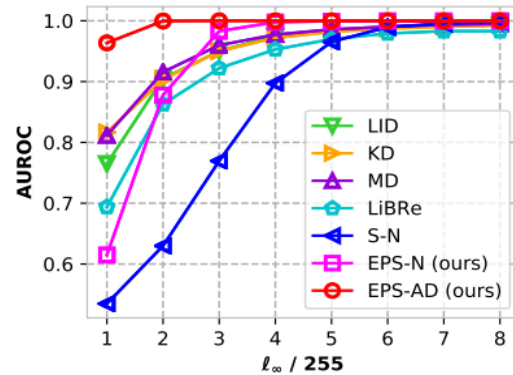


Denote *natural* samples $\mathbb{P}_X = \{\mathbf{x}^{(i)}\}_{i=1}^n$ and a *test* sample $\mathbb{Q}_Y = \{\tilde{\mathbf{x}}\}$, **EPS-MMD** between $\mathbb{P}_X$ and $\mathbb{Q}_Y$ is

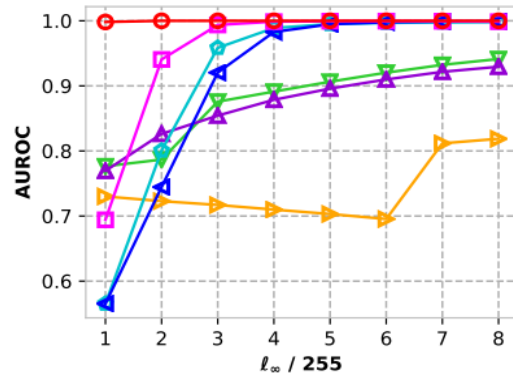
$$\widehat{\text{MMD}}^2[\mathbb{P}_X, \mathbb{Q}_Y; \mathcal{H}_k] = \frac{1}{n^2} \sum_{i,j=1}^n \underbrace{k(S(\mathbf{x}^{(i)}), S(\mathbf{x}^{(j)}))}_{\text{Similarity (kernel) between the EPSs of any two natural samples}} - \frac{2}{n} \sum_{i=1}^n \underbrace{k(S(\mathbf{x}^{(i)}), S(\tilde{\mathbf{x}}))}_{\text{Similarity (kernel) between the EPSs of natural and test samples}} + \underbrace{k(S(\tilde{\mathbf{x}}), S(\tilde{\mathbf{x}}))}_{\text{Similarity (kernel) between the EPS of test samples itself}}$$

where $\mathbf{x}^{(i)}, \mathbf{x}^{(j)} \in \mathbb{P}_X$, $\tilde{\mathbf{x}} \in \mathbb{Q}_Y$

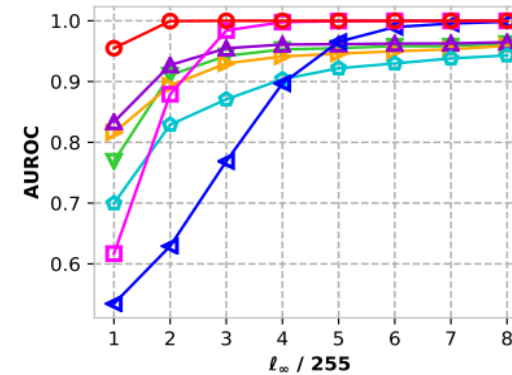
EPSAD: Statistical Detection Strategy



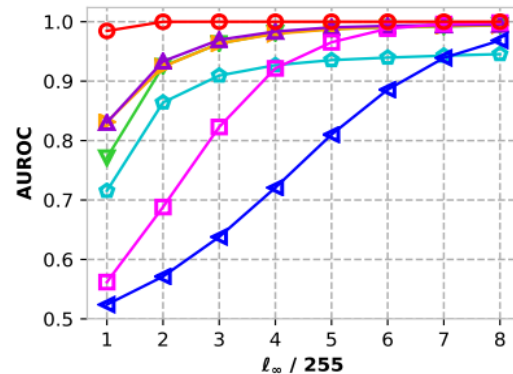
(a) Attack Method: PGD



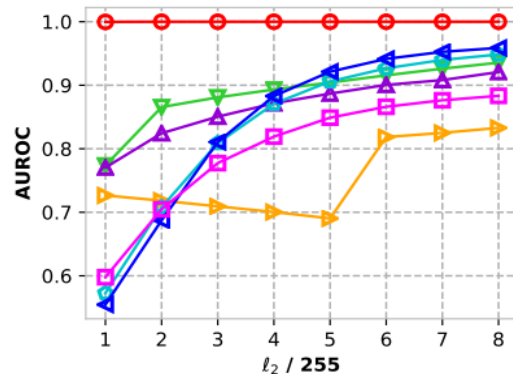
(b) Attack Method: FGSM



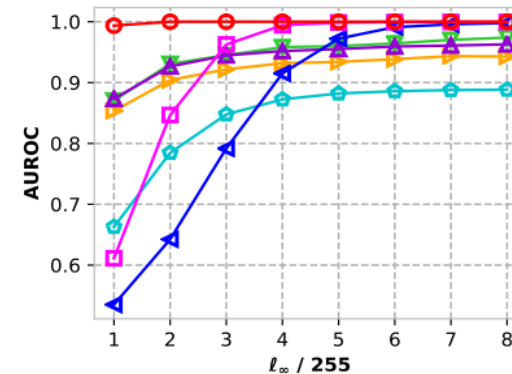
(c) Attack Method: CW



(d) Attack Method: BIM

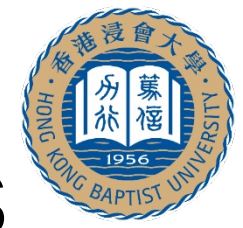


(e) Attack Method: FGSM- l_2



(f) Attack Method: AA Attack

EPS-AD achieves the best detection performance over ImageNet, especially for low attack ϵ !



Part 3 Adversarial Attacks and Defenses

- Part 3.1: Adversarial Examples and Attacks
- Part 3.2: Adversarial Training
- Part 3.3: Adversarial Purification
- Part 3.4: Adversarial Detection
- Part 3.5: Takeaways

Takeaways

Adversarial attacks expose the vulnerability of models.

- Imperceptible perturbations may can cause model predictions to collapse.
- Gradient-based attacks exploit these vulnerabilities by disturbing inputs along loss-increasing directions.

Various defense strategies aim to improve model robustness.

- Adversarial training enhance robustness by **exploring high-quality adversarial data pairs**.
- Adversarial purification explores **powerful purifier** to **restore clean data** from adversarial data.
- Adversarial detection **identifies adversarial examples** through **a well-designed detector**.

Two perspectives are central to adversarial defenses.

- **Data** (training) perspective, adversarial example generation and feature correlations matter.
- **Model** (purification/detection) perspective, task-specific modeling and strategy design matter.