A Code Source

The code for LaT-IB is publicly available at: https://github.com/RingBDStack/LaT-IB.

B Algorithms and Complexity Analysis

B.1 The Overall Process of LaT-IB

The overall training pipeline of LaT-IB is outlined in Algorithm B.1.

Algorithm B.1: The Overall Process of LaT-IB

Input: Input data $\mathcal{D}$ with label Y, epoch number of each period $E_{Warmup}, E_{Injection}, E_{Robust}$, overall epoch number E
Output: Predicted label $\hat{Y}$

```
1:  Parameter initialization
2:  for epoch $i = 1, \ldots, E$ do
3:      Encode inputs to obtain $(\mu_S, \sigma_S), (\mu_T, \sigma_T)$
4:      Decode the reparameterized embedding to generate predictions $(\hat{y}_S, \hat{y}_T)$
5:      if $i \le E_{Warmup}$ then
6:          Optimize the encoder$_S$ with Eq. (13)
7:      end if
8:      if $E_{Warmup} < i \le E_{Injection}$ then
9:          Obtain Clean/Noise/Uncertain Set using InfoJS selector
10:         Optimize the LaT-IB model with Eq. (15) and (16)
11:     end if
12:     if $i > E_{Injection}$ then
13:         Optimize the LaT-IB model with Eq. (18)
14:         Optimize the discriminator with Eq. (19)
15:     end if
16: end for
```

B.2 InfoJS Selector Algorithms Details

Algorithm B.2 implements a sample selection scheme based on MI and JS divergence.

Algorithm B.2: InfoJS Selector

Input: Trained model f_θ, data $(\mathcal{D}, Y)$, selection ratio δ
Output: Binary selection mask: mask$_S$, mask$_T$

```
1:  Run $f_\theta$ to obtain $(\mu_S, \sigma_S), (\mu_T, \sigma_T)$ and predictions $(\hat{y}_S, \hat{y}_T)$
2:  Compute $I(S;Y)$: $\ell_S \leftarrow \mathcal{L}_{CE}(\hat{y}_S, y)$ {Eq. (10)}
3:  Compute JS divergence: $JS \leftarrow D_{JS}(\mu_S, \sigma_S, \mu_T, \sigma_T)$
4:  Initialize mask$_S$, mask$_T$
5:  for each class $j = 1, \ldots, C$ do
6:      $\mathcal{I}_j \leftarrow$ indices of class-$j$ samples in $\mathcal{M}$
7:      $k_j \leftarrow \max(1, \min(\lceil \delta \cdot |\mathcal{I}_j| \rceil, |\mathcal{D}|/C))$
8:      Select top-$k_j$ and bottom-$k_j$ samples by $\ell_S$ and update mask$_S$, mask$_T$
9:      Select top-$k_j$ and bottom-$k_j$ samples by $JS$ and update mask$_S$, mask$_T$
10: end for
11: return mask$_S$, mask$_T$ {mask$_S$ for clean set; mask$_T$ for noise set}
```

B.3 $\mathcal{L}_{ConCE}$ Algorithms Details

Proposition 4.1 states:

$$-I(Y;S,T) \le -\max(I(Y;S), I(Y;T)). \tag{B.1}$$

Further reformulated as a cross-entropy loss, this provides an implementation method for $I(Y;S,T)$:

$$-I(Y;S,T) \le \sum_i \min(\mathcal{L}(\hat{y}_{S,i}, y_i), \mathcal{L}(\hat{y}_{T,i}, y_i)). \tag{B.2}$$

In the loss function design, we tailor the loss for two scenarios: image classification (many samples) and node classification (few samples).

Graph Tasks. First, we present the loss function design for the node classification task under the few-sample condition, as it is closest to Equation (B.2). Since the min function is not continuous and may hinder model learning, we instead use the smooth approximation:

$$f(a,b) = \frac{1}{\beta} \cdot \log\left(\exp(-\beta \cdot a) + \exp(-\beta \cdot b)\right). \tag{B.3}$$

Notably, as $b \to +\infty$, this expression becomes equivalent to $\min(a,b)$.

Algorithm B.3: $\mathcal{L}_{ConCE}$ for node classification

Input: Predictions $prob_1, prob_2$; Labels y; Threshold β
Output: Final loss value $\mathcal{L}_{ConCE}$

```
1: $a \leftarrow \mathcal{L}_{CE}(prob_1, y)$ {Per-sample $\mathcal{L}_{CE}$}
2: $b \leftarrow \mathcal{L}_{CE}(prob_2, y)$
3: $L \leftarrow \frac{1}{\beta} \cdot \log(\exp(-\beta \cdot a) + \exp(-\beta \cdot b))$ {Eq. (B.3)}
4: return Mean$(L)$
```

Image Tasks. We then directly apply Algorithm B.3 to the image classification task and observe that it causes lazy training, where the model tends to produce identical outputs. We hypothesize that this behavior arises because: as the number of training samples increases, the gradient contributions from noisy samples become diluted. Meanwhile, Algorithm B.3 imposes a strict separation, causing each encoder to receive only a limited portion of the input data. As a result, when the predictions from the two encoders become overly similar, the model lacks sufficient supervision signals, potentially leading to training collapse.

To address this issue, we train both encoders on the samples where the two prediction heads produce consistent outputs, rather than assigning them to a single encoder. This significantly increases the amount of usable information for each encoder. Moreover, we relax the condition in Equation B.2 to:

$$\begin{aligned} -I(Y;S,T) &\le \sum_i \min(\mathcal{L}(\hat{y}_{S,i}, y_i), \mathcal{L}(\hat{y}_{T,i}, y_i)) \\ &\le \sum_i \max(\mathcal{L}(\hat{y}_{S,i}, y_i), \mathcal{L}(\hat{y}_{T,i}, y_i)) \end{aligned} \tag{B.4}$$

The final loss function is shown in Algorithm B.4.

Algorithm B.4: $\mathcal{L}_{ConCE}$ for image classification

```
Input: Predictions $prob_1, prob_2$; Labels $y$; Threshold $t$
Output: Final loss value $\mathcal{L}_{ConCE}$
1: Compute $a \leftarrow \mathcal{L}_{CE}(prob_1.\text{copy}(), y)$
2: Compute $b \leftarrow \mathcal{L}_{CE}(prob_2.\text{copy}(), y)$
3: $pred_1 \leftarrow \arg\max(prob_1.\text{copy}())$ {Per-sample $\mathcal{L}_{CE}$}
4: $pred_2 \leftarrow \arg\max(prob_2.\text{copy}())$
5: $mask_a \leftarrow (a < b)$
6: $mask_b \leftarrow (a > b) \vee (pred_1 = pred_2)$ {The second part prevents $T$ from learning nothing.}
7: if $\sum(pred_1 = pred2)/|y| > t$ then
8:     $mask_a \leftarrow (a < b) \vee (pred_1 = pred2)$
9:     $mask_b \leftarrow (a > b)$
10: end if
11: Initialize $y_1 \leftarrow y, y_2 \leftarrow y$
12: Replace $y_1[mask_a] \leftarrow pred_1[mask_a]$ {Focus on clean samples}
13: Replace $y_2[mask_b] \leftarrow pred_2[mask_b]$ {Focus on noise samples}
14: $loss_1 \leftarrow \mathcal{L}_{CE}(prob_1, y_1)$
15: $loss_2 \leftarrow \mathcal{L}_{CE}(prob_2, y_2)$
16: return $(loss_1 + loss_2)/2$
```

B.4 Time Consumption

We adopt the constant definitions from Section 3. For Algorithm B.2, the $\mathcal{L}_{CE}$ has a time complexity of $\mathcal{O}(NC)$. The JS divergence, computed between all sample pairs, requires $\mathcal{O}(N^2k)$ operations, where k is the dimensionality of the Gaussian embeddings. Assuming class-balanced data, the per-class selection of top-k and bottom-k samples involves sorting subsets of size approximately N/C, giving a total sorting complexity of $\mathcal{O}(N\log(N/C))$. Overall, the dominant term is $\mathcal{O}(N^2k)$, since typically $C \ll N$, making the pairwise divergence computation the main computational bottleneck.

Algorithms B.3 and B.4 perform a finite number of cross-entropy loss computations. Therefore, their overall time complexity is equivalent to that of the cross-entropy loss, which is $\mathcal{O}(NC)$.

To further assess the efficiency of different methods, we report the training time consumption of several top-performing approaches.

Image Classification. We selected three baselines including VIB, (ELR+)+VIB and (Promix[1])+VIB as well as compared their time consumption under the condition of achieving a performance like Table 2 on CIFAR-10N dataset, as shown in the Table B.1. The results demonstrate that although our model incurs higher computation than VIB, it achieves better performance within fewer epochs, and overall outperforms two-stage models in terms of efficiency.

[1]For the Promix method, strictly following the original experimental setup results in a runtime exceeding 24 hours, leading to out-of-time (OOT) termination. To address this, we reduced the number of training epochs, with only a minor performance drop, to ensure timely completion.

Method	VIB	(ELR+)+VIB	(Promix)+VIB	LaT-IB
Time (h)	1.6	4.9+1.6	12.9+1.6	5.18
Epoch	100	200+100	300+100	200
Per Epoch (min)	0.94	1.30	2.18	1.55

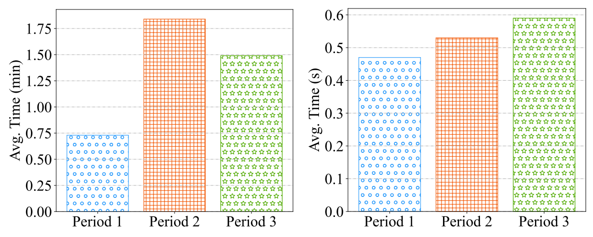
Table B.1: Comparison of training time consumption (IC)

Node Classification. We selected GIB, GIB ($\mathcal{L}_{GCE}$) and RNCGLN+GIB as baselines. Table B.2 presents the results on the DBLP dataset, supporting similar conclusions as in the image classification task. Notably, although RNCGLN achieves strong performance, it incurs significant time overhead, as also reported in NoisyGL (Wang et al. 2024).

Method	GIB	GIB ($\mathcal{L}_{GCE}$)	RNCGLN+GIB	LaT-IB
Time (s)	37.8	37.8	3445.4+37.8	55.9
Epoch	100	100	500+100	100
Per Epoch (s)	0.38	0.38	5.80	0.56

Table B.2: Comparison of training time consumption (NC)

We evaluate the time cost at each period of the pipeline. The results are shown in Figure B.1. The results indicate that for image tasks with a **large** number of samples, Knowledge Injection consumes the most time, which aligns with our theoretical time complexity analysis. In contrast, for graph tasks with **fewer** samples, the time cost of Knowledge Injection is slightly lower than that of Robust Training. We speculate that this is due to lower actual time complexity than the theoretical value, possibly resulting from low-level computational optimizations, and the relatively small sample size helps offset part of the time overhead.



(a) CIFAR-10N (worst) (b) DBLP

Figure B.1: Time Consumption Analysis

C Proofs

In this section, we present the relevant proofs from the paper. We first restate the statements to be proved, followed by detailed proofs.

C.1 The proof of Theorem 1.1

Theorem C.1 (Cumulative Degradation)**.** *In the two-stage approach, f_1 is used to modify the labels $Y' = f_1(\mathcal{D})$, and f_2 is responsible for extracting valid information from $\mathcal{D}$ while approximating the prediction result to $f_1(\mathcal{D})$. For one-stage model $g(\mathcal{D})$, it extracts the relevant information* *while removing noise. If the denoising abilities of f_1 and g are the same, the following inequality holds:*

$$P(f_2(\mathcal{D}) \neq g(\mathcal{D})) \geq \frac{H(Y'|\mathcal{D}) - 1}{\log(|\mathcal{Y}| - 1)}, \tag{C.1}$$

where $\mathcal{Y}$ denotes the support of Y, and $|\mathcal{Y}|$ denotes the number of elements in $\mathcal{Y}$. The two models perform identically iff f_2 achieves the error lower bound and $H(Y'|\mathcal{D}) = 0$.

Proof. Without loss of generality, we illustrate this using the image classification task, where $\mathcal{D} = X$.

We begin by introducing **Fano's inequality** (Fano 1952): Let the discrete random variables X and Y denote the input and output messages, respectively, with joint distribution $P(x, y)$. Let e represent the event of an error, i.e., $X \neq \tilde{X}$, where $\tilde{X} = f(Y)$ is an estimate of X. Then **Fano's inequality** states:

$$H(X|Y) \leq H_b(e) + P(e)\log(|\mathcal{X}| - 1), \tag{C.2}$$

where $\mathcal{X}$ denotes the support set of the random variable X, and $|\mathcal{X}|$ is its cardinality (i.e., the number of elements in $\mathcal{X}$). Here, $H(X|Y) = -\sum_{i,j} P(x_i, y_j)\log P(x_i|y_j)$ is the conditional entropy, $P(e) = P(X \neq \hat{X})$ is the probability of a communication error, and $H_b(e) = -P(e)\log P(e) - (1 - P(e))\log(1 - P(e))$ is the binary entropy.

Part 1: Consider the second stage f_2 of the two-stage model, whose input Y' serves as a reference for the final model output $f_2(x)$. According to the inequality above, we have:

$$H(Y'|X) \leq H_b(e) + P(e)\log(|\mathcal{Y}| - 1), \tag{C.3}$$

where $P(e) := P(f_2(x) \neq f_1(x))$. Since the binary entropy has an upper bound of 1, the lower bound of the error probability is:

$$\begin{aligned} H(Y'|X) &\leq H_b(e) + P(e)\log(|\mathcal{Y}| - 1) \\ &\leq 1 + P(e)\log(|\mathcal{Y}| - 1), \end{aligned} \tag{C.4}$$

Given the assumption that f_1 and g have the same denoising capability, we can, without loss of generality, assume $f_1(x) = g(x)$, thus:

$$P(f_2(x) \neq g(x)) = P(e) \geq \frac{H(Y'|X) - 1}{\log(|\mathcal{Y}| - 1)}. \tag{C.5}$$

Part 2: When the two models behave identically, i.e., $P(f_2(x) \neq g(x)) = 0$, from inequality C.3 we know that $H(Y'|X) \leq 0$. Since entropy is non-negative, it follows that $H(Y'|X) = 0$. At this point, f_2 has optimized its classification error to the theoretical lower bound.

When $H(Y'|X) = 0$, it is easy to see that $P(f_2(x) \neq g(x)) = 0$ satisfies inequality C.3. Since $P(f_2(x) \neq g(x)) \geq 0$, and 0 is the lower bound of the error rate, if f_2 reaches this bound, the performance of the two models is exactly the same. □

C.2 The proof of Lemma 4.1 and 4.2

Lemma C.1 (Nuisance Invariance)**.** *Taking the part of $\mathcal{D}$ that does not contribute to Y as $\mathcal{D}_n$ (D_n is independent of Y), and considering the Markov chain $(Y, \mathcal{D}_n) \to \mathcal{D} \to (S, T)$, the following inequality holds:*

$$I(\mathcal{D}_n; S, T) \leq -I(Y; S, T) + I(\mathcal{D}; S, T). \tag{C.6}$$

Proof. We proof this lemma in three steps.

Step 1: According to the property of the Markov chain, we have:

$$I(S, T; \mathcal{D}_n, Y|\mathcal{D}) = 0. \tag{C.7}$$

And because:

$$\begin{aligned} &I(S, T; \mathcal{D}_n, Y|\mathcal{D}) \\ &= \underbrace{I(S, T; \mathcal{D}_n|\mathcal{D})}_{\geq 0} + \underbrace{I(S, T; Y|\mathcal{D}_n, \mathcal{D})}_{\geq 0} \\ &= 0. \end{aligned} \tag{C.8}$$

So we have:

$$I(S, T; \mathcal{D}_n|\mathcal{D}) = I(S, T; Y|\mathcal{D}_n, \mathcal{D}) = 0 \tag{C.9}$$

By expanding $I(S, T; \mathcal{D}, \mathcal{D}_n)$, we obtain:

$$\begin{aligned} I(S, T; \mathcal{D}, \mathcal{D}_n) &= I(S, T; \mathcal{D}) + I(S, T; \mathcal{D}_n|\mathcal{D}) \\ &= I(S, T; \mathcal{D}). \end{aligned} \tag{C.10}$$

By an alternative expansion, we obtain:

$$I(S, T; \mathcal{D}, \mathcal{D}_n) = I(S, T; \mathcal{D}_n) + I(S, T; \mathcal{D}|\mathcal{D}_n). \tag{C.11}$$

Combining Eq. (C.10) and (C.11), we obtain the following equality:

$$I(S, T; \mathcal{D}) = I(S, T; \mathcal{D}_n) + I(S, T; \mathcal{D}|\mathcal{D}_n). \tag{C.12}$$

Step 2: By expanding $I(S, T; Y, \mathcal{D}|\mathcal{D}_n)$, we obtain:

$$I(S, T; Y, \mathcal{D}|\mathcal{D}_n) = I(S, T; \mathcal{D}|\mathcal{D}_n) + I(S, T; Y|\mathcal{D}, \mathcal{D}_n). \tag{C.13}$$

Since we have $I(S, T; Y|\mathcal{D}, \mathcal{D}_n) = 0$:

$$I(S, T; Y, \mathcal{D}|\mathcal{D}_n) = I(S, T; \mathcal{D}|\mathcal{D}_n). \tag{C.14}$$

Similarly, by expanding in another way:

$$I(S, T; Y, \mathcal{D}|\mathcal{D}_n) = I(S, T; Y|\mathcal{D}_n) + I(S, T; \mathcal{D}|Y, \mathcal{D}_n). \tag{C.15}$$

Combining Eq. (C.14) and (C.15), we obtain

$$\begin{aligned} I(S, T; \mathcal{D}|\mathcal{D}_n) &= I(S, T; Y|\mathcal{D}_n) + I(S, T; \mathcal{D}|Y, \mathcal{D}_n) \\ &\Rightarrow I(S, T; \mathcal{D}|\mathcal{D}_n) \geq I(S, T; Y|\mathcal{D}_n). \end{aligned} \tag{C.16}$$

Step 3: Substituting Eq. (C.16) into Eq. (C.12), we obtain:

$$\begin{aligned} I(S, T; \mathcal{D}) &\geq I(S, T; \mathcal{D}_n) + I(S, T; Y|\mathcal{D}_n) \\ &= I(S, T; \mathcal{D}_n) + I(S, T; Y). \end{aligned} \tag{C.17}$$

Therefore, we obtain the conclusion:

$$I(\mathcal{D}_n; S, T) \leq -I(Y; S, T) + I(\mathcal{D}; S, T). \tag{C.18}$$

□

Lemma C.2 (Feature Convergence)**.** *Assuming that Y can potentially contain all information about Y_c and Y_n, the following inequality holds when* $\max(I(Y_n;S), I(Y_c;T)) \leq \max(I(S;T), \varepsilon)/2 = K, \varepsilon \in \mathbb{R}$ *is satisfied:*

$$-I(Y_c;S) - I(Y_n;T) - \varepsilon \leq -I(Y;S,T) + I(S;T|Y). \tag{C.19}$$

Proof. By expanding the mutual information and combining the assumptions, we have:

$$\begin{aligned}
&I(Y;S,T) \\
=&I(Y;S) + I(Y;T) - I(Y;S;T) \\
=&I(Y;S) + I(Y;T) - (I(S;T) - I(S;T|Y)) \\
=&I(Y;S) + I(Y;T) - I(S;T) + I(S;T|Y) \\
=&I(Y_c,Y_n;S) + I(Y_c,Y_n;T) - I(S;T) + I(S;T|Y) \\
\leq&I(Y_c;S) + I(Y_n;S) + I(Y_c;T) + I(Y_n;T) \\
&- I(S;T) + I(S;T|Y).
\end{aligned} \tag{C.20}$$

Rearranging the terms, we obtain:

$$\begin{aligned}
&- I(Y;S,T) + I(S;T|Y) \\
\geq& - I(Y_c;S) - I(Y_n;T) + (I(S;T) - I(Y_n;S) - I(Y_c;T)).
\end{aligned} \tag{C.21}$$

Now, we consider two cases:

Case 1: When $I(S,T) \geq \varepsilon$:

$$\begin{aligned}
&- I(Y;S,T) + I(S;T|Y) \\
\geq& - I(Y_c;S) - I(Y_n;T) + (I(S;T) - I(Y_n;S) - I(Y_c;T)) \\
\geq& - I(Y_c;S) - I(Y_n;T).
\end{aligned} \tag{C.22}$$

Case 2: When $I(S,T) < \varepsilon$:

$$\begin{aligned}
&- I(Y;S,T) + I(S;T|Y) \\
\geq& - I(Y_c;S) - I(Y_n;T) + (I(S;T) - I(Y_n;S) - I(Y_c;T)) \\
\geq& - I(Y_c;S) - I(Y_n;T) - I(Y_n;S) - I(Y_c;T) \\
\geq& - I(Y_c;S) - I(Y_n;T) - \varepsilon.
\end{aligned} \tag{C.23}$$

Thus, the conclusion is proven. □

C.3 The proof of Proposition 4.1 $\sim$ 4.4

Proposition C.1 (The upper bound of $-I(Y;S,T)$)**.** *Given the label Y and the variable S, T that learns the characteristics of the real label space and the noise label space respectively, we have:*

$$-I(Y;S,T) \leq -\max\left(I(Y;S), I(Y;T)\right). \tag{C.24}$$

Proof. Expand directly using the definition of mutual information:

$$\begin{aligned}
I(Y;S,T) &= I(Y;S) + I(Y;T|S) \\
&= I(Y;T) + I(Y;S|T) \\
&\geq \max(I(Y;S), I(Y;T))
\end{aligned} \tag{C.25}$$

So that $-I(Y;S,T) \leq -\max\left(I(Y;S), I(Y;T)\right)$. □

Proposition C.2 (The upper bound of $I(\mathcal{D};S,T)$)**.** *Let $\mathcal{D}$, S, T be random variables. Assume the probabilistic mapping $p(\mathcal{D},S,T)$ follows the Markov chain $S \leftrightarrow \mathcal{D} \leftrightarrow T$. Then:*

$$I(\mathcal{D};S,T) \leq I(\mathcal{D};S) + I(\mathcal{D};T). \tag{C.26}$$

Proof. Without loss of generality, here we take $\mathcal{D} = X$ as an example to prove it.

Expanded by definition and processed over the probability distributions, we can obtain:

$$\begin{aligned}
&I(\mathcal{D};S,T) \\
=&\mathbb{E}_{p(x,s,t)} \log \frac{p(s,t,x)}{p(x)p(s,t)} \\
=&\mathbb{E}_{p(x,s,t)} \log \frac{p(x)p(s|x)p(t|x)}{p(x)p(s,t)} \\
=&\mathbb{E}_{p(x,s,t)} \log \left[\frac{p(s|x)p(x)}{p(x)p(s)} \cdot \frac{p(t|x)p(x)}{p(x)p(t)} \cdot \frac{p(s)p(t)}{p(s,t)}\right] \\
=&I(\mathcal{D};S) + I(\mathcal{D};T) - I(S;T) \\
\leq&I(\mathcal{D};S) + I(\mathcal{D};T)
\end{aligned} \tag{C.27}$$

□

Proposition C.3 (Reformulation of $I(S,T|Y)$)**.** *Given the label Y and the variable S, T, minimizing $I(S;T|Y)$ is equivalent to minimize $I(S,Y;T,Y)$.*

Proof.

$$\begin{aligned}
&I(S;T|Y) \\
=&\int_y p(y) \iint_{s,t} p(s,t|y) \log \frac{p(s,t|y)}{p(s|y)p(t|y)}\, ds\, dt\, dy \\
=&\iiint_{(s,t,y)} p(s,t,y) \log \frac{p(s,t,y)}{p(s,y)p(t,y)}\, p(y)\, ds\, dt\, dy \\
=&\mathbb{E}_{p(s,t,y)} \log \frac{p(s,t,y)}{p(s,y)p(t,y)} + \mathbb{E}_{p(y)} \log p(y) \\
=&I(S,Y;T,Y) - H(Y)
\end{aligned} \tag{C.28}$$

Since $H(Y)$ is a constant, it follows that $I(S;T|Y) \propto I(S,Y;T,Y)$, thus proved.

□

Proposition C.4 (Reformulation of the condition in Eq. (8): $\max(I(Y_n;S), I(Y_c;T)) \leq K$)**.** *Minimizing $I(Y_c;T)$ and $I(Y_n;S)$ is equivalent to increase $I(Y_n;T)$ and $I(Y_c;S)$.*

Proof. For S, with limited capacity ($I(Y;S) \leq A$), we have:

$$I(Y_c;S) + I(Y_n;S) \leq I(Y;S) + I(Y_c;Y_n) \leq A + \mathit{const} \tag{C.29}$$

Similarly, for T, enlarging one part constrains the other under limited capacity. □

C.4 The proof of Eq. (10): $\max I(Y;Z)$ is equivalent to $\min \mathcal{L}_{CE}$

Proof.

$$\begin{aligned}
I(Y;Z) &= \iint_{(y,z)} p(y,z) \log \frac{p(y,z)}{p(y)p(z)} \, dy \, dz \\
&= \iint_{(y,z)} p(y,z) \log \frac{p(y|z)}{p(y)} \, dy \, dz \\
&= \mathbb{E}_{p(y,z)} \left(\log(q_\theta(y|z))\right) + \\
&\quad \mathbb{E}_{p(z)} (D_{KL}(p(y|z) \| q_\theta(y|z))) + H(Y) \\
&\geq \mathbb{E}_{p(y,z)} \left(\log(q_\theta(y|z))\right) \; = -\mathcal{L}_{CE}(Z,Y),
\end{aligned} \tag{C.30}$$

where $q_{\theta_i}(\cdot)$ is variational approximation of $p(\cdot)$. □

D Implement Details

D.1 Implement Details of $\mathcal{L}_{Minimal}$

As established in Proposition 4.2, minimizing the $I(\mathcal{D}; S, T)$ objective reduces to minimizing $I(\mathcal{D}; Z)$, where $Z \in \{S, T\}$. We now discuss the methodology for estimating and optimizing $I(\mathcal{D}; Z)$.

The input is an image. In this case, $\mathcal{D} = X$, i.e., the input to the model consists solely of image features. Therefore, minimizing $I(\mathcal{D}; Z)$ reduces to minimizing the divergence between the approximate posterior and a fixed prior.

Proof. Expanded via the definition of mutual information:

$$\begin{aligned}
& I(X;Z) \\
&= \iint_{(x,z)} p(x,z) \log \frac{p(x,z)}{p(x)p(z)} \, dx \, dz \\
&= \int_x p(x) \left[\int_z p(z|x) \log \frac{p(z|x)}{q(z)} \, dz \right] dx - \int_z p(z) \log \frac{p(z)}{q(z)} \, dz \\
&= \int_x p(x) \left[D_{KL}[p(z|x) \parallel q(z)] \right] dx - D_{KL}[p(z) \parallel q(z)] \\
&\leq \mathbb{E}_{p(x)} \left[D_{KL}[p(z|x) \parallel q(z)] \right].
\end{aligned} \tag{D.1}$$

□

Since $q(z)$ represents the marginal distribution of the latent variable and is not constrained during training, we can assume without loss of generality that $q(z) \sim \mathcal{N}(0, I_n)$. The encoder maps input features to a Gaussian distribution, i.e., $p(z|x) \sim \mathcal{N}(\mu_n, \sigma_n)$. In this case, the KL divergence between two Gaussian distributions admits a closed-form solution and can be computed as:

$$D_{\mathrm{KL}}[p(z|x)|q(z)] = \frac{1}{2} \sum_{i=1}^{n} \left(\mu_i^2 + \sigma_i^2 - \log \sigma_i^2 - 1 \right). \tag{D.2}$$

The input is a graph. In this case, the input data is the graph $\mathcal{D} = \mathcal{G} = (X, A)$, where X denotes node features and A denotes the adjacency matrix. As a result, the estimation of the mutual information $I(\mathcal{D}; Z)$ becomes more intricate compared to scenarios with only feature inputs.

Following the framework of Graph Information Bottleneck (GIB), we consider two groups of indices $S_X, S_A \subseteq [L]$ that satisfy the Markovian dependence. Specifically, we assume $\mathcal{D} \perp Z_X^{(L)} \setminus \{Z_X^{(l)}\}_{l \in S_X} \cup \{Z_A^{(l)}\}_{l \in S_A}$, where $Z_X^{(l)}$ denotes the node feature representation at layer l, and $Z_A^{(l)}$ denotes the structural representation at layer l. Based on this condition, for any set of distributions $\mathbb{Q}(Z_X^{(l)})$ with $l \in S_X$, and $\mathbb{Q}(Z_A^{(l)})$ with $l \in S_A$, the following upper bound holds:

$$I(\mathcal{D}; Z_X^{(L)}) \leq \sum_{l \in S_A} \mathrm{AIB}^{(l)} + \sum_{l \in S_X} \mathrm{XIB}^{(l)}, \tag{D.3}$$

where $\mathrm{AIB}^{(l)}$ and $\mathrm{XIB}^{(l)}$ denote the information contributions from the adjacency and feature paths, respectively, and are defined as:

$$\begin{aligned}
\mathrm{AIB}^{(l)} &= \mathbb{E}\left[\log \frac{\mathbb{P}(Z_A^{(l)} | A, Z_X^{(l-1)})}{\mathbb{Q}(Z_A^{(l)})} \right], \\
\mathrm{XIB}^{(l)} &= \mathbb{E}\left[\log \frac{\mathbb{P}(Z_X^{(l)} | Z_X^{(l-1)}, Z_A^{(l)})}{\mathbb{Q}(Z_X^{(l)})} \right].
\end{aligned} \tag{D.4}$$

For the structural branch, we adopt a Bernoulli-based KL divergence estimator:

$$\widehat{\mathrm{AIB}}^{(l)} = \sum_{v \in V, t \in [\mathcal{T}]} D_{KL} \left(\mathrm{Bernoulli}(\phi_{v,t}^{(l)}) || \mathrm{Bernoulli}(\alpha) \right), \tag{D.5}$$

where $\phi^{(l)} v, t$ denotes the probability of an edge between node v and its t-hop neighbors, and α is the prior class probability.

To estimate $\mathrm{XIB}^{(l)}$, we model $\mathbb{Q}(Z_X^{(l)})$ as a mixture of Gaussians with learnable parameters. For any node v, we assume $Z_{X,v}^{(l)} \sim \sum_{i=1}^m w_i \mathcal{N}(\mu_{0,i}, \sigma_{0,i}^2)$, where w_i, $\mu_{0,i}$ and $\sigma_{0,i}$ are shared parameters across all nodes, and $Z_{X,v} \perp Z_{X,u}$ if $v \neq u$. We compute:

$$\begin{aligned}
\widehat{\mathrm{XIB}}^{(l)} &= \log \frac{\mathbb{P}(Z_X^{(l)} \mid Z_X^{(l-1)}, Z_A^{(l)})}{\mathbb{Q}(Z_X^{(l)})} \\
&= \sum_{v \in V} \log \Phi(Z_{X,v}^{(l)}; \mu_v, \sigma_v^2) \\
&\quad - \log \left(\sum_{i=1}^{m} w_i \Phi(Z_{X,v}^{(l)}; \mu_i, \sigma_{0,i}^2) \right),
\end{aligned} \tag{D.6}$$

where $\Phi(\cdot)$ denotes the probability density function of a Gaussian distribution.

In conclusion, we select proper sets of indices S_X, S_A and use substitution:

$$I(\mathcal{D}, Z) \leftarrow \sum_{l \in S_A} \widehat{\mathrm{AIB}}^{(l)} + \sum_{l \in S_X} \widehat{\mathrm{XIB}}^{(l)}. \tag{D.7}$$

More detailed content and proof can be found in GIB (Wu et al. 2020).

D.2 Implement Details of Knowledge Injection

Building on the proven success of mixup-based techniques in computer vision, we apply FMix (Harris et al. 2020) for image data augmentation across both clean and noisy subsets. To counteract potential class bias during training, we incorporate Debiased Margin-based Loss (Xiao et al. 2023) and Debiased Pseudo Labeling (Menon et al. 2020), fostering unbiased model predictions.

D.3 Implement Details of Robust Training

To further improve model performance, we perform certain modifications to the label Y during robust training period.

For image classification, every k epochs, the model predictions are combined with the original labels using exponential moving average to replace the original labels.

For node classification, at each epoch, the labels of samples with prediction confidence higher than a threshold τ are replaced with the model's predicted results.

E Experiments Details and Results

E.1 Data

Image Classification. We select CIFAR-based datasets to simulate both synthetic and real-world label noise. To mitigate the risk of overfitting to a specific dataset, we also include the Animal-10N dataset. Specifically:

- **CIFAR-10/100 (Krizhevsky, Hinton et al. 2009)[2]:** These are classic image classification datasets with 10 and 100 categories, respectively. By constructing a noise transition matrix, we introduce symmetric noise (Sym Noise, where each noisy label is uniformly sampled from all classes) and asymmetric noise (Asym Noise, where each class is flipped to a specific incorrect class based on semantic similarity). In our experiments, we consider symmetric noise with noise rates of 20% and 50%, which are **independent of the input features**. Additionally, based on class-wise correlations, we design a 40% asymmetric noise setting on CIFAR-10.
- **CIFAR-10N/100N(Wei et al. 2022)[3]:** These datasets introduce real-world label noise based on standard CIFAR-10 and CIFAR-100. Each image is annotated by multiple human workers, and the final label is obtained via majority voting, thereby reflecting more natural and realistic label noise. Previous studies have shown that **the noise is not independent of the input features**. CIFAR-10N includes five noise levels (Aggregate: 9.03%, Random 1: 17.23%, Random 2: 18.12%, Random 3: 17.64%, Worst: 40.21%), while CIFAR-100N includes one noise level (40.20%).
- **Animal-10N(Song, Kim, and Lee 2019)[4]:** This dataset is constructed based on animal categories from ImageNet. It consists of images from 10 common animal classes collected and labeled by non-expert annotators. The label noise mainly stems from **confusion between fine-grained categories**, such as dog vs. wolf or cow vs. horse. The estimated noise rate is around 8%.

A more intuitive comparison of the selected datasets is presented in Table E.1.

Dataset	# Class	Noise Type	Noise Ratio
CIFAR	10 / 100	Sym/Asym	20%, 40%, 50%
CIFARN	10 / 100	Real-world	9.03% $\sim$ 40.21%
Animal-10N	10	Real-world	$\approx$8%

Table E.1: Comparison of image datasets

Node Classification. We select three classic citation network datasets: Cora, Citeseer, and Pubmed. In addition, we include the one author collaboration network DBLP for evaluation. To ensure consistency across methods, we randomly sample 20 nodes per class for training. For validation and testing, 500 and 1000 nodes are randomly selected from the graph, respectively. Specifically:

- **Cora, Citeseer and Pubmed (Sen et al. 2008)[5]:** These citation network datasets are widely adopted in graph learning research involving label noise. In each dataset, nodes correspond to academic papers, and edges represent citation connections among them. The node features consist of binary word vectors indicating whether particular words from a vocabulary are present or absent. Each node is labeled according to the research topic category of the corresponding paper.
- **DBLP (Pan et al. 2016)[6]:** This dataset represents an author collaboration network within the field of computer science. Nodes represent documents, while edges correspond to citation relationships between these documents. Node features are derived from word vectors extracted from the text, and labels reflect the category of the research topic.

A more intuitive comparison of the selected datasets is presented in Table E.2.

Dataset	# Class	# Node	# Edge	# Feat.
Cora	7	2,708	5,278	1,433
Citeseer	6	3,327	4,552	3,703
Pubmed	3	19,717	44,324	500
DBLP	4	17,716	52,867	1,639

Table E.2: Comparison of graph datasets

E.2 Baselines

We compare with four categories, 16 baselines in two scenarios: ① Classic IB methods; ② IB with robust loss functions; ③ Improved IB variants; ④ Two-stage denoising + IB methods. They comprehensively evaluate our LaT-IB's performance from multiple perspectives. Specifically:

[2]https://www.cs.toronto.edu/~kriz/cifar.html

[3]https://github.com/UCSC-REAL/cifar-10-100n/tree/main

[4]https://dm.kaist.ac.kr/datasets/animal-10n/

[5]https://github.com/kimiyoung/planetoid/tree/master/data

[6]https://github.com/abojchevski/graph2gauss/raw/master/data

Image Classification.

- Classical IB Models: This paper selects two IB methods, **VIB** (Alemi et al. 2017) and **NIB** (Kolchinsky, Tracey, and Wolpert 2019), as baseline approaches. The core idea of VIB is to approximate the optimization of the IB objective using variational inference. NIB further extends the original IB principle to address the limitations of KL divergence estimation in VIB, which is restricted by the assumptions and simplicity of the prior distribution. Instead of relying on an analytical KL computation, NIB adopts kernel density estimation (KDE) for learning.
- IB with robust loss functions: In the IB framework, the mutual information $I(Z;Y)$ is typically optimized via cross-entropy. The **Generalized Cross-Entropy (GCE)** loss (Zhang and Sabuncu 2018) combines cross-entropy and mean absolute error to enhance robustness. The **Symmetric Cross-Entropy (SCE)** loss (Wang et al. 2019) integrates standard cross-entropy with reverse cross-entropy, improving resistance to label noise. By replacing the original loss with these robust alternatives, new IB variants are constructed under robust loss functions.
- Improved IB Methods: Many subsequent studies have enhanced the feature extraction capability and robustness of IB. The **SIB** (Yang et al. 2025) framework leverages an auxiliary encoder to capture missing structural information, improving learning performance. **DT-JSCC** (Xie et al. 2023) proposes a robust encoding architecture based on the IB principle to improve transmission resilience under varying communication channels. These two methods represent improved IB frameworks and are included in the baseline comparisons.
- Two-Stage Denoise + IB Methods: Although Theorem 3.1 indicates that denoising followed by IB may not yield optimal results, this study further explores this approach. Three representative denoising methods are adopted: **JoCoR** (Wei et al. 2020), a robust learning method based on co-training; **ELR+** (Liu et al. 2020), which uses noise-robust regularization for label correction; and **ProMix** (Xiao et al. 2023), which integrates Mixup data augmentation and dynamic confidence modeling, representing a state-of-the-art denoising approach. These denoised datasets are subsequently used for IB training.

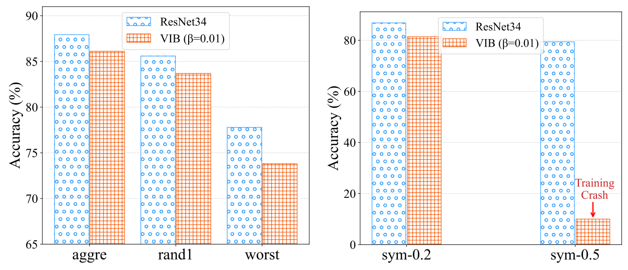
Node Classification.

- Classical IB Models: This paper selects the **GIB** (Wu et al. 2020) method as the classical IB baseline. GIB leverages the IB principle by learning graph representations that compress input feature and structure while preserving label-relevant information.
- IB with robust loss functions: Consistent with the robust loss settings used in the image classification task.
- Improved IB Methods: Two representative improvements are included. **CurvGIB** (Fu et al. 2025) introduces discrete Ricci curvature into the IB framework to better capture topological structures in graphs, enabling the model to discard spurious connectivity information and preserve label-relevant substructures. **IS-GIB** (Yang et al. 2023) enhances generalization under distribution shifts by jointly modeling individual and structural information bottlenecks, improving the robustness and transferability of graph representations.
- Two-Stage Denoise + IB Methods: For denoise model, **RNCGLN** (Zhu et al. 2024) first applies graph contrastive learning and multi-head self-attention to learn local-global representations, followed by pseudo-labeling strategies to address graph and label noise. **CGNN** (Yuan et al. 2023) performs neighborhood-based label correction and contrastive learning to smooth representations across graph views. It iteratively corrects noisy labels using neighbors' predictions before applying the IB objective for final node classification.

E.3 Preliminary Experiment

Vulnerability of IB Methods to Noisy Labels. To investigate the sensitivity of IB methods to label noise, we first conduct preliminary experiments on image classification and node classification to examine the relationship between IB performance and label corruption.

For image classification, Figure E.1a demonstrates that the VIB (Alemi et al. 2017) framework suffers performance degradation on the CIFAR-10N (Wei et al. 2022) dataset as label noise increases. Moreover, the decline becomes more pronounced with higher noise levels. In extreme cases, excessive noise can even lead to training collapse under the IB framework as shown in Figure E.1b.

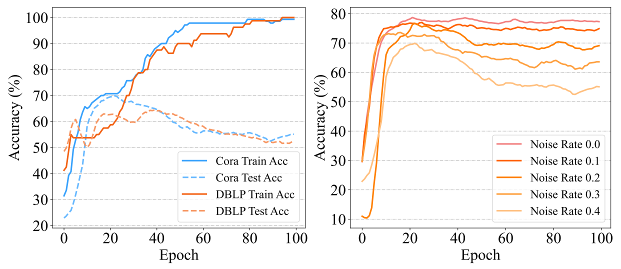


(a) With feature-dependent noise (b) With symmetric noise

Figure E.1: Preliminary Experiments on Image Classification.

For node classification, Figure E.2a shows that GIB (Wu et al. 2020) gradually fits the noisy labels during training, leading to a steady decline in performance on the testing set. Furthermore, Figure E.2b, using the Cora dataset as an example, illustrates that this trend becomes increasingly severe as the level of label noise increases.

Performance Degradation in Cascaded Models. Theorem 3.1 shows that cascading models weakens the denoising effect of the first stage. This phenomenon is further illustrated in Figure E.3, where the data is first denoised using the ELR+ model and then used to train with the IB method. The resulting performance often falls short of the accuracy achieved after denoising alone.



(a) With 40% uniform noise (b) Cora (varying noise levels)

Figure E.2: Preliminary Experiments on Node Classification.

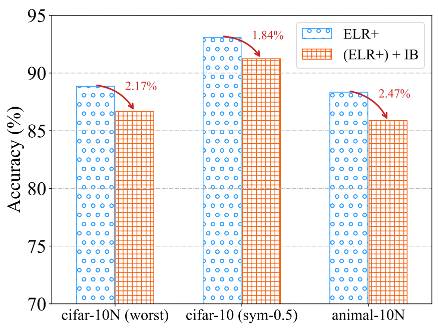


Figure E.3: Performance degradation.

E.4 Additional Results

Table E.3, E.4 and E.5 presents a broader comparison of various LaT-IB methods under different types and levels of noise. We use a dash (–) to denote cases where the model fails or produces invalid results. Classification accuracy (%) is used as the evaluation metric, where a higher value indicates better model performance.

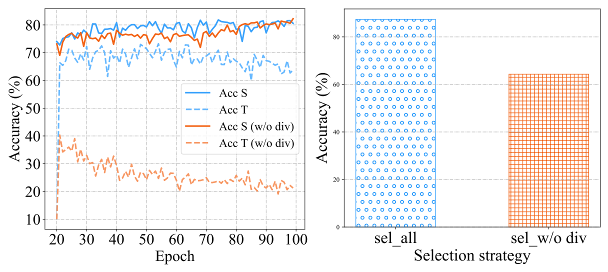
For image classification, the model does not consistently outperform denoise + IB approaches in some cases. A possible explanation is that the denoising models are particularly effective on the CIFAR dataset, thereby significantly enhancing the IB performance. To further validate this, we also evaluated on the Animal-10N dataset, as shown in Table 2, where LaT-IB achieves the best performance. Moreover, Table 4 demonstrates that our method substantially outperforms two-stage approaches under adversarial attacks, further confirming the superiority of LaT-IB.

For the node classification task, it is evident that under high noise conditions, our model significantly outperforms other competitive baselines as shown in Table E.4 and E.5, indicating the strong robustness of LaT-IB to label noise.

E.5 Analysis of Sample Selection Strategy

Our proposed sample selection strategy is based on two key components: mutual information and divergence. Mutual information is indirectly measured through the cross-entropy loss, whose effectiveness has been validated in numerous prior works (Arpit et al. 2017; Song et al. 2019). As for divergence, following Observation 4.2, we adopt Jensen-Shannon (JS) divergence as a criterion to filter samples. This section primarily investigates how divergence affects the learning behavior of the encoder.

Figure E.4a shows the training process during the knowledge injection stage on the CIFAR-10N (worst) dataset, comparing the original method (sel_all) with a variant that excludes divergence-based selection (sel_w/o div). Figure E.4b presents the final performance of both methods. Experimental results demonstrate that removing divergence leads to a larger accuracy gap between the two encoders but ultimately worse performance. We attribute this to encoder T failing to learn meaningful representations of noisy samples, rendering its predictions less informative. As a result, it is unable to provide effective guidance for feature separation in the third stage. These findings confirm that divergence-based sample selection plays a critical role in training the encoder effectively.



(a) Training Process of Knowledge Injection (b) Training Accuracy under Different Select Strategy.

Figure E.4: The influence of D_{JS}

E.6 Hyperparameter Sensitivity Analysis of δ in Algorithm B.2

In this section, we investigate the effect of the hyperparameter δ, which controls the amount of knowledge the encoder (S, T) acquires during the Knowledge Injection phase. Figure E.5 illustrates the model's performance on the Cora and Citeseer datasets under various noise types and levels.

The results show that neither excessively high nor low values of δ yield optimal performance. We hypothesize that a too-small δ limits the encoder's exposure to training data, impairing its ability to learn useful representations. Conversely, a too-large δ causes the two encoders to converge in their learning, reducing their ability to distinguish between clean and noisy information effectively.

Furthermore, the influence of δ varies across datasets, indicating that the model's capacity to separate clean and noisy information is dataset-dependent.

E.7 Analysis of Model Learning Behavior

To further investigate the learning behavior of the encoder, we perform a t-SNE dimensionality reduction analysis on the embeddings obtained from models trained on the CIFAR-10N (worst) dataset. For a comprehensive comparison, we analyze the embeddings under the following four

Method	Model	CIFAR-10			CIFAR-100	
		20%(sym)	50%(sym)	40%(asym)	20%(sym)	50%(sym)
Classic IB	VIB	$81.49_{\pm 1.00}$	$72.58_{\pm 1.63}$	$79.27_{\pm 1.13}$	$53.86_{\pm 0.47}$	$44.25_{\pm 0.98}$
	NIB	$83.44_{\pm 0.83}$	$76.16_{\pm 1.27}$	$78.16_{\pm 1.32}$	$55.99_{\pm 0.79}$	$46.20_{\pm 0.77}$
Robust Loss	VIB (L_{GCE})	$88.43_{\pm 0.17}$	$84.82_{\pm 0.33}$	$81.32_{\pm 1.61}$	—	—
	VIB (L_{SCE})	$84.36_{\pm 0.13}$	$77.67_{\pm 1.40}$	$76.59_{\pm 0.73}$	$53.06_{\pm 1.83}$	—
Improved IB	SIB	$86.40_{\pm 0.55}$	$65.52_{\pm 1.53}$	$80.06_{\pm 1.52}$	$57.64_{\pm 1.92}$	$35.01_{\pm 1.23}$
	DT-JSCC	$84.51_{\pm 0.41}$	$72.49_{\pm 0.77}$	$80.95_{\pm 0.41}$	$49.58_{\pm 0.13}$	$35.80_{\pm 0.96}$
Deniose + IB	JoCoR+VIB	$88.71_{\pm 0.18}$	$81.71_{\pm 0.21}$	$60.66_{\pm 0.08}$	$62.61_{\pm 0.27}$	$53.69_{\pm 0.11}$
	(ELR+)+VIB	$\underline{93.16_{\pm 0.23}}$	$\underline{91.28_{\pm 0.06}}$	$84.75_{\pm 0.11}$	$71.44_{\pm 0.93}$	$54.12_{\pm 0.20}$
	Promix+VIB	$92.98_{\pm 0.14}$	$\mathbf{92.40_{\pm 0.10}}$	$\mathbf{91.87_{\pm 0.05}}$	$71.89_{\pm 0.16}$	$\mathbf{69.77_{\pm 0.56}}$
Ours	LaT-IB	$\mathbf{94.56_{\pm 0.12}}$	$91.13_{\pm 0.16}$	$\underline{88.89_{\pm 0.73}}$	$\mathbf{75.79_{\pm 0.19}}$	$\underline{67.28_{\pm 0.55}}$

Table E.3: Classification accuracy (%) of CIFAR under Symmetric/Asymmetric Noise. All the best results are highlighted in **bold**, and the second-best results are underlined.

Method	Model	Clean	Uniform Noise				Pair Noise			
			10%	20%	30%	40%	10%	20%	30%	40%
ClassicIB	GIB	$\mathbf{75.33_{\pm 3.19}}$	$\underline{75.10_{\pm 2.48}}$	$\mathbf{73.03_{\pm 5.67}}$	$\underline{72.77_{\pm 2.36}}$	$57.17_{\pm 7.83}$	$74.73_{\pm 4.39}$	$69.80_{\pm 6.18}$	$\underline{66.30_{\pm 6.60}}$	$\underline{46.60_{\pm 5.45}}$
Robust Loss	GIB ($\mathcal{L}_{GCE}$)	$75.03_{\pm 2.79}$	$74.43_{\pm 3.01}$	$72.03_{\pm 6.91}$	$72.27_{\pm 4.07}$	$57.57_{\pm 6.42}$	$73.67_{\pm 3.85}$	$71.03_{\pm 4.62}$	$62.60_{\pm 6.99}$	$43.63_{\pm 6.69}$
	GIB ($\mathcal{L}_{SCE}$)	$74.23_{\pm 3.56}$	$72.67_{\pm 2.55}$	$71.10_{\pm 6.45}$	$70.63_{\pm 4.03}$	$\underline{58.47_{\pm 5.17}}$	$73.33_{\pm 4.64}$	$70.07_{\pm 4.93}$	$59.87_{\pm 6.40}$	$43.10_{\pm 6.80}$
Improved IB	CurvGIB	$70.67_{\pm 3.23}$	$67.67_{\pm 2.71}$	$64.63_{\pm 6.52}$	$61.97_{\pm 4.46}$	$54.47_{\pm 6.13}$	$66.93_{\pm 1.79}$	$64.03_{\pm 4.55}$	$56.27_{\pm 7.96}$	$45.03_{\pm 1.68}$
	IS-GIB	$54.73_{\pm 0.15}$	$53.17_{\pm 1.48}$	$42.93_{\pm 2.46}$	$45.53_{\pm 0.79}$	$38.77_{\pm 6.34}$	$49.73_{\pm 0.82}$	$46.97_{\pm 1.19}$	$40.33_{\pm 3.64}$	$38.80_{\pm 1.22}$
Denoise + IB	RNCGLN+GIB	$\underline{74.70_{\pm 2.65}}$	$73.37_{\pm 0.65}$	$\underline{72.97_{\pm 5.09}}$	$70.80_{\pm 2.57}$	$52.27_{\pm 6.65}$	$\underline{75.00_{\pm 3.45}}$	$\underline{69.93_{\pm 3.40}}$	$64.67_{\pm 6.10}$	$39.83_{\pm 7.13}$
	CGNN+GIB	$73.17_{\pm 1.86}$	$69.37_{\pm 6.01}$	$67.67_{\pm 5.28}$	$67.70_{\pm 1.85}$	$54.70_{\pm 5.06}$	$66.53_{\pm 5.74}$	$64.80_{\pm 6.16}$	$52.00_{\pm 5.40}$	$44.33_{\pm 4.74}$
Ours	LaT-IB	$74.57_{\pm 0.99}$	$\mathbf{75.87_{\pm 0.33}}$	$71.13_{\pm 2.22}$	$\mathbf{75.60_{\pm 0.71}}$	$\mathbf{62.53_{\pm 7.56}}$	$\mathbf{75.43_{\pm 1.04}}$	$\mathbf{73.70_{\pm 3.10}}$	$\mathbf{68.57_{\pm 3.02}}$	$\mathbf{53.00_{\pm 7.26}}$

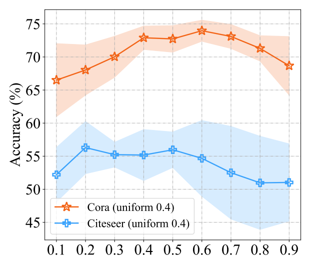
Table E.4: Classification accuracy (%) on the DBLP dataset under different noise types and noise rates. All the best results are highlighted in **bold**, and the second-best results are underlined.

Method	Model	Cora				Citeseer			
		Uniform		Pair		Uniform		Pair	
		20%	40%	20%	40%	20%	40%	20%	40%
ClassicIB	GIB	$\mathbf{75.37_{\pm 2.47}}$	$71.60_{\pm 1.51}$	$73.73_{\pm 0.39}$	$65.27_{\pm 3.84}$	$60.00_{\pm 3.37}$	$48.20_{\pm 2.20}$	$57.50_{\pm 2.35}$	$\underline{43.67_{\pm 2.90}}$
Robust Loss	GIB ($\mathcal{L}_{GCE}$)	$\underline{74.77_{\pm 0.76}}$	$70.47_{\pm 2.25}$	$75.00_{\pm 1.77}$	$65.27_{\pm 2.52}$	$60.20_{\pm 3.93}$	$50.80_{\pm 2.14}$	$58.73_{\pm 2.31}$	$41.03_{\pm 1.25}$
	GIB ($\mathcal{L}_{SCE}$)	$74.40_{\pm 0.45}$	$69.17_{\pm 1.09}$	$\underline{75.97_{\pm 2.02}}$	$\underline{66.73_{\pm 4.08}}$	$58.83_{\pm 5.19}$	$50.93_{\pm 0.84}$	$\mathbf{59.10_{\pm 3.61}}$	$41.40_{\pm 1.35}$
Improved IB	CurvGIB	$65.90_{\pm 3.69}$	$52.63_{\pm 3.23}$	$67.17_{\pm 2.11}$	$52.00_{\pm 3.19}$	$49.80_{\pm 4.19}$	$46.20_{\pm 1.69}$	$48.77_{\pm 2.09}$	$38.70_{\pm 3.70}$
	IS-GIB	$69.20_{\pm 0.62}$	$54.73_{\pm 1.28}$	$70.30_{\pm 0.99}$	$56.47_{\pm 4.54}$	$55.73_{\pm 3.44}$	$39.03_{\pm 1.10}$	$54.90_{\pm 4.28}$	$40.13_{\pm 2.36}$
Denoise + IB	RNCGLN+GIB	$74.53_{\pm 1.58}$	$\underline{71.67_{\pm 1.49}}$	$73.57_{\pm 1.59}$	$63.60_{\pm 3.40}$	$\underline{60.90_{\pm 2.95}}$	$\underline{52.83_{\pm 4.82}}$	$55.77_{\pm 3.17}$	$\mathbf{46.00_{\pm 3.01}}$
	CGNN+GIB	$70.53_{\pm 4.69}$	$64.73_{\pm 6.75}$	$73.57_{\pm 1.37}$	$59.00_{\pm 3.29}$	$57.53_{\pm 3.73}$	$45.73_{\pm 4.29}$	$54.43_{\pm 3.30}$	$41.57_{\pm 1.90}$
Ours	LaT-IB	$74.30_{\pm 2.01}$	$\mathbf{74.07_{\pm 1.46}}$	$\mathbf{76.87_{\pm 1.06}}$	$\mathbf{66.80_{\pm 3.14}}$	$\mathbf{61.63_{\pm 2.24}}$	$\mathbf{55.17_{\pm 3.86}}$	$\underline{58.93_{\pm 2.77}}$	$\mathbf{46.00_{\pm 0.71}}$

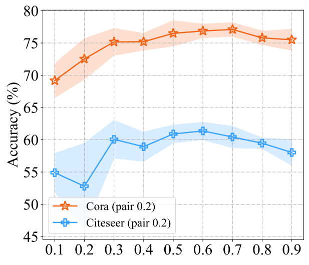
Table E.5: Classification accuracy (%) on the Cora and Citeseer dataset under different noise types and noise rates. All the best results are highlighted in **bold**, and the second-best results are underlined.

settings: ① VIB without the $I(X;Z)$ constraint, which is approximately equivalent to a standard ResNet34 model; ② Standard VIB model; ③ LaT-IB model at the end of Knowledge Injection, referred to as LaT-IB KI; ④ Full LaT-IB model.

Figure E.6a and E.6b show that IB methods produce more compact embeddings by minimizing $I(X;Z)$, reducing the encoding space and slightly lowering performance. Figure



(a) Under 40% Uniform Noise



(b) Under 20% Pair Noise

Figure E.5: The influence of δ

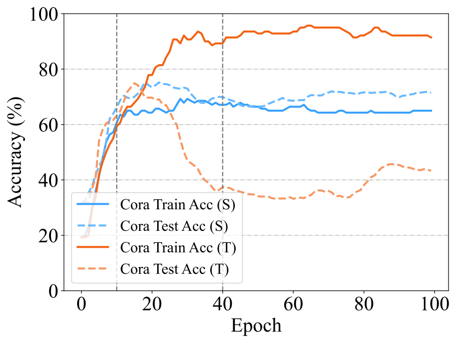
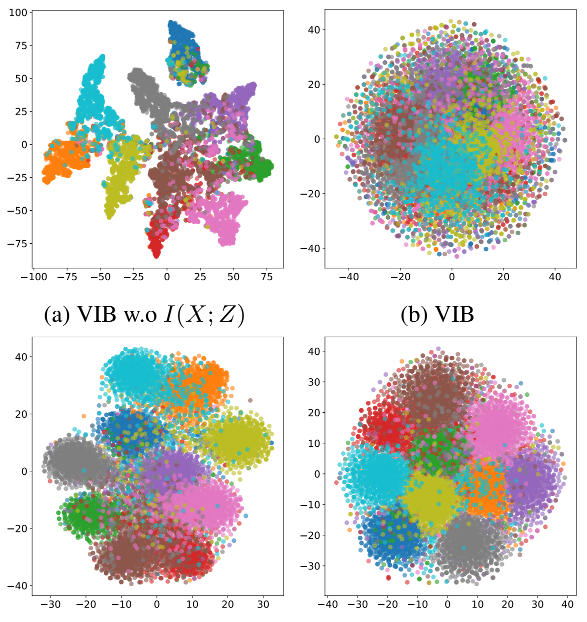


Figure E.7: The learning behavior of (S, T)

E.6c and E.6d illustrate that the third robust training stage of LaT-IB further restricts $I(\mathcal{D}; S, T)$ and improves noise robustness. In particular, Figure E.6d shows embeddings **similar to IB's minimal sufficient property** (Figure E.6b), while clearer class boundaries demonstrate LaT-IB's ability to **learn cleaner representations**.



(a) VIB w.o $I(X; Z)$ (b) VIB

(c) LaT-IB KI (d) LaT-IB

Figure E.6: The embedding distributions of different models

To investigate the learning process of the two encoders S and T, Figure E.7 shows their prediction accuracy on training and test sets with 40% uniform noise on the Cora dataset, where vertical dashed lines divide the process into three periods. Encoder T gradually fits all (noisy) data, while S, influenced by T, achieves about 65% accuracy on the training set by fitting **mostly clean data**.

E.8 Hyperparameter settings

Image Classification. For CIFAR-related datasets, we set the batch size to 256, and each image is reshaped to a size of (32, 32). For the Animal-10N dataset, the batch size is set to 128, and each image is reshaped to (68, 68). We use SGD as the optimizer with a learning rate of 0.005, momentum of 0.9, and a weight decay of 5e-4. A cosine learning rate scheduler is applied.

The dimension of the encoder's latent space is set to 128. Each experiment is repeated three times, and we report the mean and standard deviation of the results.

Node Classification. For node classification tasks, we use the Adam optimizer with a learning rate of 0.001. To enable the computation of structural KL divergence, the model backbone is configured as a two-layer GAT. In our hyperparameter settings, the KL divergence weight for the features is set to 0.001, and the weight for the structural KL divergence is set to 0.01. The dimension of the encoder's latent space is set to 16 or 20. Each experiment is repeated three times, and we report the mean and standard deviation of the results.

In the experiments, the Algorithm B.2 uses $\delta \in \{0.1, 0.2, \cdots, 0.9\}$. Besides, we set $\beta \in \{1e^{-1}, \cdots, 1e^{-5}\}$ and $\gamma = \in \{1e^{0}, \cdots, 1e^{-4}\}$ in our hyperparameter configuration. The predicted confidence scores are set based on the learning behavior of Warmup samples for each dataset, ensuring that the upper confidence bound is greater than 0.5, while the lower confidence bound is less than 0.5.

E.9 Hardware and Software Configurations

We conduct the experiments with:

- Operating System: Ubuntu 22.04.4 LTS.
- CPU: Intel(R) Xeon(R) Silver 4110 CPU @ 2.10GHz.
- GPU: NVIDIA Tesla V100 SMX2 with 32GB of Memory.
- Software: CUDA 12.8, Python 3.10.12, PyTorch[7] 2.2.0, PyTorch Geometric[8] 2.6.1.

[7]https://github.com/pytorch/pytorch
[8]https://github.com/pyg-team/pytorch_geometric

References

Alemi, A. A.; Fischer, I.; Dillon, J. V.; and Murphy, K. 2017. Deep Variational Information Bottleneck. In *International Conference on Learning Representations*.

Arpit, D.; Jastrzebski, S.; Ballas, N.; Krueger, D.; Bengio, E.; Kanwal, M. S.; Maharaj, T.; Fischer, A.; Courville, A.; Bengio, Y.; et al. 2017. A closer look at memorization in deep networks. In *International conference on machine learning*, 233–242. PMLR.

Fano, R. 1952. Class notes for course 6.574: Transmission of information. *Lecture Notes*.

Fu, X.; Wang, J.; Gao, Y.; Sun, Q.; Yuan, H.; Li, J.; and Li, X. 2025. Discrete curvature graph information bottleneck. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 16666–16673.

Harris, E.; Marcu, A.; Painter, M.; Niranjan, M.; Prügel-Bennett, A.; and Hare, J. 2020. Fmix: Enhancing mixed sample data augmentation. *arXiv preprint arXiv:2002.12047*.

Kolchinsky, A.; Tracey, B. D.; and Wolpert, D. H. 2019. Nonlinear information bottleneck. *Entropy*, 21(12): 1181.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Liu, S.; Niles-Weed, J.; Razavian, N.; and Fernandez-Granda, C. 2020. Early-learning regularization prevents memorization of noisy labels. *Advances in neural information processing systems*, 33: 20331–20342.

Menon, A. K.; Jayasumana, S.; Rawat, A. S.; Jain, H.; Veit, A.; and Kumar, S. 2020. Long-tail learning via logit adjustment. *arXiv preprint arXiv:2007.07314*.

Pan, S.; Wu, J.; Zhu, X.; Zhang, C.; and Wang, Y. 2016. Tri-party deep network representation. In *International Joint Conference on Artificial Intelligence 2016*, 1895–1901. Association for the Advancement of Artificial Intelligence (AAAI).

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine*, 29(3): 93–93.

Song, H.; Kim, M.; and Lee, J.-G. 2019. Selfie: Refurbishing unclean samples for robust deep learning. In *International conference on machine learning*, 5907–5915. PMLR.

Song, H.; Kim, M.; Park, D.; and Lee, J.-G. 2019. How does early stopping help generalization against label noise? *arXiv preprint arXiv:1911.08059*.

Wang, Y.; Ma, X.; Chen, Z.; Luo, Y.; Yi, J.; and Bailey, J. 2019. Symmetric cross entropy for robust learning with noisy labels. In *Proceedings of the IEEE/CVF international conference on computer vision*, 322–330.

Wang, Z.; Sun, D.; Zhou, S.; Wang, H.; Fan, J.; Huang, L.; and Bu, J. 2024. NoisyGL: A Comprehensive Benchmark for Graph Neural Networks under Label Noise. *arXiv preprint arXiv:2406.04299*.

Wei, H.; Feng, L.; Chen, X.; and An, B. 2020. Combating noisy labels by agreement: A joint training method with co-regularization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 13726–13735.

Wei, J.; Zhu, Z.; Cheng, H.; Liu, T.; Niu, G.; and Liu, Y. 2022. Learning with Noisy Labels Revisited: A Study Using Real-World Human Annotations. In *International Conference on Learning Representations*.

Wu, T.; Ren, H.; Li, P.; and Leskovec, J. 2020. Graph information bottleneck. *Advances in Neural Information Processing Systems*, 33: 20437–20448.

Xiao, R.; Dong, Y.; Wang, H.; Feng, L.; Wu, R.; Chen, G.; and Zhao, J. 2023. ProMix: combating label noise via maximizing clean sample utility. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, 4442–4450.

Xie, S.; Ma, S.; Ding, M.; Shi, Y.; Tang, M.; and Wu, Y. 2023. Robust information bottleneck for task-oriented communication with digital modulation. *IEEE Journal on Selected Areas in Communications*, 41(8): 2577–2591.

Yang, H.; Wu, Y.; Wen, D.; Zhou, Y.; and Shi, Y. 2025. Structured IB: Improving Information Bottleneck with Structured Feature Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 21922–21928.

Yang, L.; Zheng, J.; Wang, H.; Liu, Z.; Huang, Z.; Hong, S.; Zhang, W.; and Cui, B. 2023. Individual and structural graph information bottlenecks for out-of-distribution generalization. *IEEE Transactions on Knowledge and Data Engineering*, 36(2): 682–693.

Yuan, J.; Luo, X.; Qin, Y.; Zhao, Y.; Ju, W.; and Zhang, M. 2023. Learning on graphs under label noise. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5. IEEE.

Zhang, Z.; and Sabuncu, M. 2018. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, 31.

Zhu, Y.; Feng, L.; Deng, Z.; Chen, Y.; Amor, R.; and Witbrock, M. 2024. Robust node classification on graph data with graph and label noise. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, 17220–17227.