APPENDIX

A MORE DISCUSSIONS

A.1 CLASS LABELS

Our method uses class labels, but it does not introduce any extra data collection cost compared to one-for-one AD models. Because our method only explicitly uses class labels, while they implicitly use class labels. The existing AD datasets are collected for one-for-one anomaly detection (*i.e.*, we need to train a model for each class). Thus, the existing AD datasets need to be separated according to classes, with each class as a subdataset. Therefore, one-for-one AD methods also need class labels, as they require normal samples from the same class to train. If the classes are not separated (in other words, without class labels), these one-for-one AD methods cannot be used either. Our method actually has the same supervision as these methods. The difference is that we explicitly use the class labels but they don't explicitly use the class labels. So, our method still follows the same data organization format as the one-for-one AD models, we don't introduce any extra data collection cost. But the advantage of our unified AD method is that we can train one model for all classes, greatly reducing the resource costs of training and deploying. Moreover, our method only uses class labels and does not require pixel-level annotations. For real-world industrial applications, this doesn't incur extra data collection costs, as we usually consciously collect data according to different classes. So, class labels are bonuses for us.

We summarize the training samples and the supervision information required by our method and other methods as follows:

Table 4: Training samples and supervision information summarization.

PaDiM	MKD	DRAEM	PMAD	UniAD	OmniAL	FastFlow	CFLOW	HGAD (Ours)
N	N	N+P	N	N	N+P	N	N	N
C	C	C	C	w/o C	w/o C	C	C	C

where N means only using normal samples during training, P means also using pseudo (or synthetic) anomalies during training, C means requiring class labels and w/o C means not using class labels.

Our method can be easily extended to completely unsupervised, as industrial images often have significant differences between different classes. For instance, after extracting global features, we can use a simple unsupervised clustering algorithm to divide each image into a specific class. Or we can only require few-shot samples for each class as a reference, and then compute the feature distances between each input sample to these reference samples. In this way, we can also conveniently divide each sample into the most relevant class.

A.2 MORE DISCUSSIONS WITH UNIAD

Here, we provide more discussions between our HGAD and UniAD (You et al., 2022) to further clarify that we does not simply replace the reconstruction model of UniAD with normalizing flows and accordingly introduce the "homogeneous mapping" issue.

Our method and UniAD (You et al., 2022) both aim to tackle the multi-class AD task, but we adopt a different research line from UniAD. Unlike UniAD, our method doesn't face the "identical shortcut" issue, as it doesn't have the abnormal information leakage risk in principle (see the second paragraph in sec. 1). However, the NF-based models have their own problems when used for multi-class anomaly detection. We first empirically observe a significant performance degradation when directly using the previous NF-based AD methods for multi-class anomaly detection (see Fig. 2(a) and Tab. 1). The "homogeneous mapping" issue is a possible explanation we provide for this phenomenon, rather than casually introduced. Moreover, as analyzed in sec. 3.2, we provide a reasonable explanation from the perspective of the formula in normalizing flow. Finally, the key designs proposed in our method are completely different from those in UniAD. Based on these designs, we can achieve much better unified AD performance than UniAD on all four real-world AD datasets. Therefore, compared to UniAD, our method should be reasonably seen as an effective exploration for multi-class anomaly detection in the direction of normalizing flow based anomaly detection.

Table 5: **Complexity comparison** between our HGAD and other baseline methods.

	PaDiM	MKD	DRAEM	PMAD	UniAD	FastFlow	CFLOW	HGAD (Ours)
FLOPs	14.9G	24.1G	198.7G	52G	9.7G	36.2G	30.7G	32.8G
Learnable Parameters	/	24.9M	97.4M	163.4M	9.4M	69.8M	24.7M	30.8M
Inference Speed	12.8fps	23fps	22fps	10.8fps	29fps	42.7fps	24.6fps	24.3fps
Training Epochs	/	50	700	300	1000	400	200	100

A.3 The Way to Guarantee Anomalies Out-of-Distribution.

Here, we further explain how to guarantee that anomalies are out-of-distribution. In our method, increasing inter-class distances is to ensure that the latent space has sufficient capacity to accommodate the features of multiple classes. In addition, we also model the intra-class Gaussian mixture distribution for each class to ensure that the normal distribution of each class still remains compact. Therefore, even if anomalies fall into the inter-class Gaussian mixture distribution, they are usually in the low-density regions among the inter-class class centers. So, we can still ensure that anomalies are out-of-distribution through intra-class Gaussian mixture distributions. As described in Anomaly Scoring section (sec. 3.4), we can guarantee that anomalies are recognized as out-of-distribution by combining intra-class log-likelihood and inter-class entropy to measure anomalies. Because only if the anomaly is out-of-distribution, the anomaly score based on the association of log-likelihood and entropy will be high, and the detection metrics can be better. The visualization results (decision-level results based on log-likelihood) in Fig. 2 and 5 also intuitively show that our method has fewer normal-abnormal overlaps and the normal boundary is more compact.

A.4 Limitations

In this paper, we propose a novel HGAD to accomplish the multi-class anomaly detection task. Even if our method manifests good unified AD performance, there are still some limitations of our work. Here, we discuss two main limitations as follows:

One limitation is that our method mainly targets NF-based AD methods to improve their unified AD abilities. To this end, our method cannot be directly utilized to the other types of anomaly detection methods, such as reconstruction-based, OCC-based, embedding-based, and distillation-based approaches (see Related Work, Sec. 2). However, we believe that the other types of anomaly detection methods can also be improved into unified AD methods, but we need to find and solve the corresponding issues in the improvement processes, such as the ”identical shortcut” issue (You et al., 2022) in reconstruction-based AD methods. How to upgrade the other types of anomaly detection methods to unified AD methods and how to find a general approach for unified anomaly detection modeling will be the future works.

In this work, our method is mainly aimed at solving multi-class anomaly detection, it doesn’t have the ability to directly generalize to unseen classes. Because, in our method, the new class features usually do not match the learned known multi-class feature distribution, which can lead to normal samples being misrecognized as anomalies. Generalization to unseen classes can be defined as cross-class anomaly detection (Yao et al., 2023b), where the model is trained with normal instances from multiple known classes with the objective to detect anomalies from unseen classes. In the practical industrial scenarios, models with cross-class anomaly detection capabilities are very valuable and necessary, because new products will continuously appear and it’s cost-ineffective and inconvenient to retrain models for new products. We think our method should achieve better performance on unseen classes than previous NF-based methods due to the ability to learn more complex multi-class distribution, but it’s far from solving the cross-class problem. How to design a general approach for cross-class anomaly detection modeling will be the future works.

A.5 Model Complexity

With the image size fixed as 256×256, we compare the FLOPs and learnable parameters with all competitors. In Tab. 5, we can conclude that the advantage of HGAD does not come from a larger model capacity. Compared to UniAD, our method requires fewer epochs (100 vs. 1000) and has a shorter training time.

A.6 Real-world Applications

In industrial inspection scenarios, the class actually means a type of product on the production line. Multi-class anomaly detection can be applied to train one model to detect defects in all products, without the need to train one model for each type of product. This can greatly reduce the resource costs of training and deploying. In video surveillance scenarios, we can use one model to simultaneously detect anomalies in multiple camera scenes.

B Social Impacts and Ethics

As a unified model for multi-class anomaly detection, the proposed method does not suffer from particular ethical concerns or negative social impacts. All datasets used are public. All qualitative visualizations are based on industrial product images, which doesn't infringe personal privacy.

C Implementation Details

Optimization Strategy. In the initial a few epochs, we only optimize with $\mathcal{L}_g$ and $\mathcal{L}_{mi}$ to form distinguishable inter-class main class centers. And then we simultaneously optimize the intra-class delta vectors and the main class centers with the overall loss $\mathcal{L}$ in Eq. 11. In this way, we can better decouple the inter-class and intra-class learning processes. This strategy can make the intra-class learning become much easier, as optimizing after forming distinguishable inter-class main centers will not have the problem that many centers initially overlap with each other.

Model Architecture. The normalizing flow model in our method is mainly based on Real-NVP (Dinh et al., 2017) architecture, but the convolutional subnetwork in Real-NVP is replaced with a two-layer MLP network. As in Real-NVP, the normalizing flow in our model is composed of the so-called coupling layers. All coupling layers have the same architecture, and each coupling layer is designed to tractably achieve the forward or reverse affine coupling transformation (Dinh et al., 2017) (see Eq. 4). Then each coupling layer is followed by a random and fixed soft permutation of channels Ardizzone et al. (2019) and a fixed scaling by a constant, similar to ActNorm layers introduced by (Kingma & Dhariwal, 2019). For the coupling coefficients (*i.e.*, $\exp(s(x_1))$ and $t(x_1)$ in Eq. 4), each subnetwork predicts multiplicative and additive components simultaneously, as done by (Dinh et al., 2017). Furthermore, we adopt the soft clamping of multiplication coefficients used by (Dinh et al., 2017). The layer numbers of the normalizing flow models are all 12. We add positional embeddings to each coupling layer, which are concatenated with the first half of the input features (*i.e.*, x_1 in Eq. 4). Then, the concatenated embeddings are sent into the subnetwork for predicting couping coefficients. The dimension of all positional embeddings is set to 256. The implementation of the normalizing flows in our model is based on the FrEIA library `https://github.com/VLLHD/FrEIA`.

D Datasets

MVTecAD. The MVTecAD (Bergmann et al., 2019) dataset is widely used as a standard benchmark for evaluating unsupervised image anomaly detection methods. This dataset contains 5354 high-resolution images (3629 images for training and 1725 images for testing) of 15 different product categories. 5 classes consist of textures and the other 10 classes contain objects. A total of 73 different defect types are presented and almost 1900 defective regions are manually annotated in this dataset.

BTAD. The BeanTech Anomaly Detection dataset (Mishra et al., 2021) is an another popular benchmark, which contains 2830 real-world images of 3 industrial products. Product 1, 2, and 3 of this dataset contain 400, 1000, and 399 training images respectively.

MVTecAD-3D. The MVTecAD-3D (Bergmann et al., 2021) dataset is recently proposed for 3D anomaly detection, which contains 4147 high-resolution 3D point cloud scans paired with 2D RGB images from 10 real-world categories. In this dataset, most anomalies can also be detected only through RGB images. Since we focus on image anomaly detection, we only use RGB images of the MVTecAD-3D dataset. We refer to this subset as MVTec3D-RGB.

VisA. The Visual Anomaly dataset (zou et al., 2022) is a recently proposed larger anomaly detection dataset compared to MVTecAD (Bergmann et al., 2019). This dataset contains 10821 images with

9621 normal and 1200 anomalous samples. In addition to images that only contain single instance, the VisA dataset also have images that contain multiple instances. Moreover, some product categories of the VisA dataset, such as Cashew, Chewing gum, Fryum and Pipe fryum, have objects that are roughly aligned. These characteristics make the VisA dataset more challenging than the MVTecAD dataset, whose images only have single instance and are better aligned.

E Detailed Loss Function Derivation

In this section, we provide the detailed derivation of the loss functions proposed in the main text, including $\mathcal{L}_g$ (Eq. 6), $\mathcal{L}_{mi}$ (Eq. 8), and $\mathcal{L}_{in}$ (Eq. 10).

Derivation of $\mathcal{L}_g$. We use a Gaussian mixture model with class-dependent means μ_y and unit covariance $\mathbb{I}$ as the inter-class Gaussian mixture prior, which is defined as follows:

$$p_Z(z) = \sum\nolimits_y p(y)\mathcal{N}(z; \mu_y, \mathbb{I}) \tag{12}$$

Below, we use c_y as a shorthand of $\log p(y)$. Then, we can calculate the log-likelihood as follows:

$$\begin{aligned}\log p_Z(z) &= \log\big[\sum\nolimits_y p(y)\mathcal{N}(z; \mu_y, \mathbb{I})\big] \\ &= \log\big[\sum\nolimits_y p(y)(2\pi)^{-\frac{d}{2}}\mathrm{e}^{-\frac{1}{2}(z-\mu_y)^T(z-\mu_y)}\big] \\ &= -\frac{d}{2}\log(2\pi) + \log\big(\sum\nolimits_y \mathrm{e}^{c_y}\cdot \mathrm{e}^{-\frac{||z-\mu_y||_2^2}{2}}\big) \\ &= -\frac{d}{2}\log(2\pi) + \log\big(\sum\nolimits_y \mathrm{e}^{-\frac{||z-\mu_y||_2^2}{2}+c_y}\big) \\ &= -\frac{d}{2}\log(2\pi) + \underset{y}{\mathrm{logsumexp}}\left(-\frac{||z-\mu_y||_2^2}{2} + c_y\right)\end{aligned} \tag{13}$$

Then, we bring the $\log p_Z(z)$ into Eq. 1 to obtain the log-likelihood $\log p_\theta(x)$ as:

$$\log p_\theta(x) = -\frac{d}{2}\log(2\pi) + \underset{y}{\mathrm{logsumexp}}\left(-\frac{||\varphi_\theta(x)-\mu_y||_2^2}{2} + c_y\right) + \log|\det J| \tag{14}$$

Further, the maximum likelihood loss in Eq. 2 can be written as:

$$\begin{aligned}\mathcal{L}_m &= \mathbb{E}_{x\sim p(X)}[-\log p_\theta(x)] \\ &= \mathbb{E}_{x\sim p(X)}\left[-\underset{y}{\mathrm{logsumexp}}\left(-\frac{||\varphi_\theta(x)-\mu_y||_2^2}{2} + c_y\right) - \log|\det J| + \frac{d}{2}\log(2\pi)\right]\end{aligned} \tag{15}$$

The loss function $\mathcal{L}_g$ is actually defined as the above maximum likelihood loss $\mathcal{L}_m$ with inter-class Gaussian mixture prior.

Extending $\mathcal{L}_g$ for Learning Intra-Class Mixed Class Centers. When we extend the Gaussian prior $p(Z|y) = \mathcal{N}(\mu_y, \mathbb{I})$ to mixture Gaussian prior $p(Z|y) = \sum_{i=1}^M p_i(y)\mathcal{N}(\mu_i^y, \mathbb{I})$, where M is the number of intra-class latent centers, the likelihood of latent feature z can be calculated as follows:

$$p_Z(z) = \sum\nolimits_y p(y)\Big(\sum\nolimits_{i=1}^M p_i(y)\mathcal{N}(\mu_i^y, \mathbb{I})\Big) \tag{16}$$

Then, following the derivation in Eq. 13, we have:

$$\log p_Z(z) = \log\Big(\sum\nolimits_y p(y)\,\underset{i}{\mathrm{sumexp}}\left[\frac{-||z-\mu_i^y||_2^2}{2} + c_i^y - \frac{d}{2}\log(2\pi)\right]\Big) \tag{17}$$

where c_i^y is the shorthand of $\log p_i(y)$. The $\mathcal{L}_g$ for learning intra-class mixed class centers can be defined as:

$$\mathcal{L}_g = \mathbb{E}_{x\sim p(X)}\left[-\log\Big(\sum\nolimits_y p(y)\,\underset{i}{\mathrm{sumexp}}\left[\frac{-||\varphi_\theta(x)-\mu_i^y||_2^2}{2} + c_i^y - \frac{d}{2}\log(2\pi)\right]\Big) - \log|\det J|\right] \tag{18}$$

However, as the initial latent features Z usually have large distances with the intra-class centers $\{\mu_i^y\}_{i=1}^M$, this will cause the value after sumexp operation close to 0. After calculating the logarithm function, it's easy to cause the loss to be numerically ill-defined (NaN). Besides, we find that directly employing Eq. 18 for learning intra-class mixed class centers will lead to much worse results, as we need to simultaneously optimize all intra-class centers of all classes to fit the inter-class Gaussian mixture prior. To this end, we propose to decouple the inter-class Gaussian mixture prior fitting and the intra-class latent centers learning. The loss function of learning intra-class mixed class centers is defined in Eq. 22.

Derivation of $\mathcal{L}_{mi}$. We first derive the general format of the mutual information loss in Eq. 7 as follows:

$$\begin{aligned}\mathcal{L}_{mi} &= -I(Y,Z) = -H(Y) + H(Y|Z) = -H(Y) - H(Z) + H(Y,Z)\\ &= -H(Y) - \mathbb{E}_{x\sim p(X)}\Big[-\log\big(\sum\nolimits_y p(y)p(\varphi_\theta(x)|y)\big)\Big]\\ &\quad + \mathbb{E}_{(x,y)\sim p(X,Y)}[-\log(p(y)p(\varphi_\theta(x)|y))]\\ &= -H(Y) - \mathbb{E}_{(x,y)\sim p(X,Y)}\left[\log\frac{p(y)p(\varphi_\theta(x)|y)}{\sum_{y'}p(y')p(\varphi_\theta(x)|y')}\right]\\ &= -\mathbb{E}_{y\sim p(Y)}[-\log p(y)] - \mathbb{E}_{(x,y)\sim p(X,Y)}\left[\log\frac{p(y)p(\varphi_\theta(x)|y)}{\sum_{y'}p(y')p(\varphi_\theta(x)|y')}\right]\end{aligned} \tag{19}$$

Then, by replacing $p(\varphi_\theta(x)|y)$ with $\mathcal{N}(\varphi_\theta(x);\mu_y,\mathbb{I})$ in the mutual information loss, we can derive the following practical loss format for the second part of Eq. 19. We also use c_y as a shorthand of $\log p(y)$.

$$\begin{aligned}&-\mathbb{E}_{(x,y)\sim p(X,Y)}\left[\log\frac{p(y)p(\varphi_\theta(x)|y)}{\sum_{y'}p(y')p(\varphi_\theta(x)|y')}\right]\\ &= -\mathbb{E}_{(x,y)\sim p(X,Y)}\left[\log\frac{p(y)\mathcal{N}(\varphi_\theta(x);\mu_y,\mathbb{I})}{\sum_{y'}p(y')\mathcal{N}(\varphi_\theta(x);\mu_{y'},\mathbb{I})}\right]\\ &= -\mathbb{E}_{(x,y)\sim p(X,Y)}\left[\log\frac{(2\pi)^{-\frac{d}{2}}\mathrm{e}^{-\frac{1}{2}(\varphi_\theta(x)-\mu_y)^T(\varphi_\theta(x)-\mu_y)}\cdot \mathrm{e}^{c_y}}{\sum_{y'}(2\pi)^{-\frac{d}{2}}\mathrm{e}^{-\frac{1}{2}(\varphi_\theta(x)-\mu_{y'})^T(\varphi_\theta(x)-\mu_{y'})}\cdot \mathrm{e}^{c_{y'}}}\right]\\ &= -\mathbb{E}_{(x,y)\sim p(X,Y)}\left[\log\frac{\mathrm{e}^{-\frac{||\varphi_\theta(x)-\mu_y||_2^2}{2}+c_y}}{\sum_{y'}\mathrm{e}^{-\frac{||\varphi_\theta(x)-\mu_{y'}||_2^2}{2}+c_{y'}}}\right]\\ &= -\mathbb{E}_{(x,y)\sim p(X,Y)}\left[\underset{y}{\mathrm{logsoftmax}}\left(-\frac{||\varphi_\theta(x)-\mu_{y'}||_2^2}{2}+c_{y'}\right)\right]\end{aligned} \tag{20}$$

By replacing Eq. 20 back to the Eq. 19, we can obtain the following practical loss format of the mutual information loss.

$$\begin{aligned}\mathcal{L}_{mi} &= -\mathbb{E}_{y\sim p(Y)}[-\log p(y)] - \mathbb{E}_{(x,y)\sim p(X,Y)}\left[\underset{y}{\mathrm{logsoftmax}}\left(-\frac{||\varphi_\theta(x)-\mu_{y'}||_2^2}{2}+c_{y'}\right)\right]\\ &= -\mathbb{E}_{y\sim p(Y)}[-c_y] - \mathbb{E}_{(x,y)\sim p(X,Y)}\left[\underset{y}{\mathrm{logsoftmax}}\left(-\frac{||\varphi_\theta(x)-\mu_{y'}||_2^2}{2}+c_{y'}\right)\right]\\ &= -\mathbb{E}_{(x,y)\sim p(X,Y)}\left[\underset{y}{\mathrm{logsoftmax}}\left(-\frac{||\varphi_\theta(x)-\mu_{y'}||_2^2}{2}+c_{y'}\right)-c_y\right]\end{aligned} \tag{21}$$

Intra-Class Mixed Class Centers Learning Loss. The loss function for learning the intra-class class centers is actually the same as the $\mathcal{L}_g$ in Eq. 6. But we note that we need to replace the class centers with the intra-class centers: $\mu_i^y = \{\mu_1^y + \Delta\mu_i^y\}_{i=1}^M$, and the sum operation is performed on all intra-class centers μ_i^y within the corresponding class y. Another difference is that we need to detach the main center μ_1^y from the gradient graph and only optimize the delta vectors. The loss function can be written as:

$$\mathcal{L}_{in} = \mathbb{E}_{(x,y)\sim p(X,Y)}\left[-\underset{i}{\mathrm{logsumexp}}\left(-\frac{||\varphi_\theta(x)-(SG[\mu_1^y]+\Delta\mu_i^y)||_2^2}{2}+c_i^y\right)-\log|\det J|\right] \tag{22}$$

Finally, we note that the use of $\mathrm{logsumexp}$ and $\mathrm{logsoftmax}$ pytorch operations above is quite important. As the initial $||\varphi_\theta(x) - \mu_y||_2^2/2$ distance values are usually large, if we explicitly perform the $\exp$ and then $\log$ operations, the values will become too large and the loss will be numerically ill-defined (NaN).

F An Information-Theoretic View

Information theory (Shannon, 1948) is an important theoretical foundation for explaining deep learning methods. The well-known *Information Bottleneck principle* (Tishby et al., 1999; Tishby & Zaslavsky, 2015; Alemi et al., 2017; Saxe et al., 2018) is also rooted from the information theory, which provides an explanation for representation learning as the trade-off between information compression and informativeness retention. Below, we denote the input variable as X, the latent variable as Z, and the class variable as Y. Formally, in this theory, supervised deep learning attempts to minimize the mutual information $I(X, Z)$ between the input X and the latent variable Z while maximizing the mutual information $I(Z, Y)$ between Z and the class Y:

$$\min I(X, Z) - \alpha I(Z, Y) \tag{23}$$

where the hyperparameter $\alpha > 0$ controls the trade-off between compression (*i.e.*, redundant information) and retention (*i.e.*, classification accuracy).

In this section, we will show that our method can be explained by the *Information Bottleneck principle* with the learning objective $\min I(X, Z_\mathcal{E}) - \alpha I(Z, Y)$, where $Z_\mathcal{E} = \varphi_\theta(X + \mathcal{E})$ and $p(\mathcal{E}) = \mathcal{N}(0, \sigma^2\mathbb{I})$ is Gaussian with mean zero and covariance $\sigma^2\mathbb{I}$. First, we derive $I(X, Z_\mathcal{E})$ as follows:

$$\begin{aligned} I(X, Z_\mathcal{E}) &= I(Z_\mathcal{E}, X) = H(Z_\mathcal{E}) - H(Z_\mathcal{E}|X) \\ &= \underbrace{\mathbb{E}_{x\sim p(X),\epsilon\sim p(\mathcal{E})}[-\log p(\varphi_\theta(x+\epsilon))]}_{:=A} + \underbrace{\mathbb{E}_{x\sim p(X),\epsilon\sim p(\mathcal{E})}[\log p(\varphi_\theta(x+\epsilon)|x)]}_{:=B} \end{aligned} \tag{24}$$

To approximate the second item (B), we can replace the condition x with $\varphi_\theta(x)$, because φ_θ is bijective and both conditions convey the same information (Ardizzone et al., 2020).

$$B = \mathbb{E}_{x\sim p(X),\epsilon\sim p(\mathcal{E})}[\log p(\varphi_\theta(x+\epsilon)|x)] = \mathbb{E}_{x\sim p(X),\epsilon\sim p(\mathcal{E})}[\log p(\varphi_\theta(x+\epsilon)|\varphi_\theta(x))] \tag{25}$$

We can linearize $\varphi_\theta(x+\epsilon)$ by its first order Taylor expansion: $\varphi_\theta(x+\epsilon) = \varphi_\theta(x) + J\epsilon + \mathcal{O}(\epsilon^2)$, where the matrix $J = \triangledown_x \varphi_\theta(x)$ is the Jacobian matrix of the bijective transformation ($z = \varphi_\theta(x)$ and $x = \varphi_\theta^{-1}(z)$). Then, we have:

$$\begin{aligned} B &= \mathbb{E}_{x\sim p(X),\epsilon\sim p(\mathcal{E})}[\log p(\varphi_\theta(x) + J\epsilon + \mathcal{O}(\epsilon^2)|\varphi_\theta(x))] \\ &= \mathbb{E}_{x\sim p(X),\epsilon\sim p(\mathcal{E})}[\log p(\varphi_\theta(x) + J\epsilon|\varphi_\theta(x))] + \mathbb{E}_{\epsilon\sim p(\mathcal{E})}[\mathcal{O}(\epsilon^2)] \\ &= \mathbb{E}_{x\sim p(X),\epsilon\sim p(\mathcal{E})}[\log p(\varphi_\theta(x) + J\epsilon|\varphi_\theta(x))] + \mathcal{O}(\sigma^2) \end{aligned} \tag{26}$$

where the $\mathbb{E}_{\epsilon\sim p(\mathcal{E})}[\mathcal{O}(\epsilon^2)]$ is actually the covariance of $p(\mathcal{E}) = \mathcal{N}(0, \sigma^2\mathbb{I})$, thus can be replaced with $\mathcal{O}(\sigma^2)$. Since $p(\mathcal{E})$ is Gaussian with mean zero and covariance $\sigma^2\mathbb{I}$, the conditional distribution is Gaussian with mean $\varphi_\theta(x)$ and covariance $\sigma^2 JJ^T$. Then, we have:

$$\begin{aligned} B &= \mathbb{E}_{x\sim p(X),\epsilon\sim p(\mathcal{E})}[\log \mathcal{N}(\varphi_\theta(x) + J\epsilon; \varphi_\theta(x), \sigma^2 JJ^T)] + \mathcal{O}(\sigma^2) \\ &= \mathbb{E}_{x\sim p(X),\epsilon\sim p(\mathcal{E})}[\log((2\pi)^{-\frac{d}{2}} \cdot (|\sigma^2 JJ^T|)^{-\frac{1}{2}} \cdot \mathrm{e}^{-\frac{1}{2}\frac{1}{\sigma^2}\epsilon^T\epsilon})] + \mathcal{O}(\sigma^2) \\ &= \mathbb{E}_{x\sim p(X)}[-\frac{1}{2}\log(|\sigma^2 JJ^T|)] - \frac{d}{2}\log(2\pi) - \frac{1}{2\sigma^2}\mathbb{E}_{\epsilon\sim p(\mathcal{E})}[\epsilon^T\epsilon] + \mathcal{O}(\sigma^2) \\ &= \mathbb{E}_{x\sim p(X)}[-\frac{1}{2}\log(|JJ^T|)] - d\log(\sigma) - \frac{d}{2}\log(2\pi) - \frac{1}{2\sigma^2}\mathcal{O}(\sigma^2) + \mathcal{O}(\sigma^2) \\ &= \mathbb{E}_{x\sim p(X)}[-\log|\det J|] - d\log(\sigma) - \frac{d}{2}\log(2\pi) - \frac{1}{2\sigma^2}\mathcal{O}(\sigma^2) + \mathcal{O}(\sigma^2) \end{aligned} \tag{27}$$

For the first item (A), we can use the derivation in Eq. 13.

$$\begin{aligned} A &= \mathbb{E}_{x\sim p(X),\epsilon\sim p(\mathcal{E})}[-\log p(\varphi_\theta(x+\epsilon))] \\ &= \mathbb{E}_{x\sim p(X),\epsilon\sim p(\mathcal{E})}\Big[\frac{d}{2}\log(2\pi) - \underset{y}{\text{logsumexp}}\Big(-\frac{||\varphi_\theta(x+\epsilon)-\mu_y||_2^2}{2}+c_y\Big)\Big] \\ &= \mathbb{E}_{x\sim p(X),\epsilon\sim p(\mathcal{E})}\Big[-\underset{y}{\text{logsumexp}}\Big(-\frac{||\varphi_\theta(x+\epsilon)-\mu_y||_2^2}{2}+c_y\Big)\Big] + \frac{d}{2}\log(2\pi) \end{aligned} \tag{28}$$

Finally, we put the above derivations together and drop the constant items and the items that vanish with rate $\mathcal{O}(\sigma^2)$ as $\sigma \to 0$. The $I(X, Z_{\mathcal{E}})$ becomes:

$$I(X, Z_{\mathcal{E}}) = \mathbb{E}_{x\sim p(X),\epsilon\sim p(\mathcal{E})}\Big[-\underset{y}{\text{logsumexp}}\Big(-\frac{||\varphi_\theta(x+\epsilon)-\mu_y||_2^2}{2}+c_y\Big) - \log|\det J|\Big] \tag{29}$$

We can find that the $I(X, Z_{\mathcal{E}})$ has the same formula as the loss $\mathcal{L}_g$ except the constant item $\frac{d}{2}\log(2\pi)$, and $I(Z, Y) = I(Y, Z) = -\mathcal{L}_{mi}$ (see Eq. 19). Thus, the learning objective $\min I(X, Z_{\mathcal{E}}) - \alpha I(Z, Y)$ in *Information Bottleneck principle* can be converted to $\mathcal{L}_g + \alpha\mathcal{L}_{mi}$, which is the first half part of the training loss in Eq. 11.

From the *Information Bottleneck principle* perspective, we can explain our method: it attempts to minimize the mutual information $I(X, Z_{\mathcal{E}})$ between X and $Z_{\mathcal{E}}$, forcing the model to ignore the irrelevant aspects of $X + \mathcal{E}$ which do not contribute to fit the latent distribution and only increase the potential for overfitting. Therefore, the $\mathcal{L}_g$ loss function actually endows the normalizing flow model with the compression ability for establishing correct invertible mappings between input X and the latent Gaussian mixture prior Z, which is effective to prevent the model from learning the "homogeneous mapping". Simultaneously, it encourages to maximize the mutual information $I(Y, Z)$ between Y and Z, forcing the model to map different class features to their corresponding class centers which can contribute to class discriminative ability.

G Additional Results

Quantitative Results Under the Single-Class Setting. In Tab. 6, we report the detailed results of anomaly detection and localization on MVTecAD (Bergmann et al., 2019) under the single-class setting. We can find that all baselines achieve excellent results under the single-class setting, but their performances drop dramatically under the unified case (see Tab. 1 in the main text). For instance, the strong baseline, DRAEM, suffers from a drop of 9.9% and 10.1%. The performance of the previous SOTA NF-based AD method, FastFlow, drops by 7.6% and 2.5%. This demonstrates that the unified anomaly detection is quite more challenging than the conventional single-class anomaly detection task, and current SOTA AD methods cannot be directly applied to the multi-class AD task well. Thus, how to improve the unified AD ability for AD methods should be further studied. On the other hand, compared with reconstruction-based AD methods (*e.g*, DRAEM (Zavrtanik et al., 2021)), NF-based AD methods have less performance degradation when directly applied to the unified case, indicating that NF-based approaches may be a more suitable way for the multi-class AD modeling than the reconstruction-based approaches.

Log-likelihood Histograms. In Fig. 5, we show log-likelihoods generated by the single-class NF-based AD method and our method. All categories are from the MVTecAD dataset. The visualization results can empirically verify our speculation that the single-class NF-based AD methods may fall into the "homogeneous mapping" issue, where the normal and abnormal log-likelihoods are highly overlapped.

Qualitative Results. We present in Fig. 6 additional anomaly localization results of categories with different anomalies in the MVTecAD dataset. It can be found that our approach can generate much better anomaly score maps that the single-class NF-based baseline CFLOW (Gudovskiy et al., 2022) even for different categories from the MVTecAD dataset.

Table 6: **Anomaly detection and localization results on MVTecAD**. All methods are evaluated under the single-class setting. $\cdot/\cdot$ means the image-level and pixel-level AUROCs.

Category	**Baseline Methods**			**Unified Methods**		**NF Based Methods**	
	PaDiM	MKD	DRAEM	PMAD	UniAD	FastFlow	CFLOW
Carpet	99.8/99.0	79.3/95.6	97.0/95.5	99.7/98.8	99.9/98.0	100/99.4	100/99.3
Grid	96.7/97.1	78.0/91.8	99.9/99.7	97.7/96.3	98.5/94.6	99.7/98.3	97.6/99.0
Leather	100/99.0	95.1/98.1	100/98.6	100/99.2	100/98.3	100/99.5	97.7/99.7
Tile	98.1/94.1	91.6/82.8	99.6/99.2	100/94.4	99.0/91.8	100/96.3	98.7/98.0
Wood	99.2/94.1	94.3/84.8	99.1/96.4	98.0/93.3	97.9/93.4	100/97.0	99.6/96.7
Bottle	99.9/98.2	99.4/96.3	99.2/99.1	100/98.4	100/98.1	100/97.7	100/99.0
Cable	92.7/96.7	89.2/82.4	91.8/94.7	98.0/97.5	97.6/96.8	100/98.4	100/97.6
Capsule	91.3/98.6	80.5/95.9	98.5/94.3	89.8/98.6	85.3/97.9	100/99.1	99.3/99.0
Hazelnut	92.0/98.1	98.4/94.6	100/99.7	100/98.8	99.9/98.8	100/99.1	96.8/98.9
Metal nut	98.7/97.3	73.6/86.4	98.7/99.5	99.2/97.5	99.0/95.7	100/98.5	91.9/98.6
Pill	93.3/95.7	82.7/89.6	98.9/97.6	94.3/95.5	88.3/95.1	99.4/99.2	99.9/99.0
Screw	85.8/98.4	83.3/96.0	93.9/97.6	73.9/91.4	91.9/97.4	97.8/99.4	99.7/98.9
Toothbrush	96.1/98.8	92.2/96.1	100/98.1	91.4/98.2	95.0/97.8	94.4/98.9	95.2/99.0
Transistor	97.4/97.6	85.6/76.5	93.1/90.9	99.8/97.8	100/98.7	99.8/97.3	99.1/98.0
Zipper	90.3/98.4	93.2/93.9	100/98.8	99.5/96.7	96.7/96.0	99.5/98.7	98.5/99.1
Mean	95.5/97.4	87.8/90.7	98.0/97.3	96.1/96.8	96.6/96.6	99.4/98.5	98.3/98.6

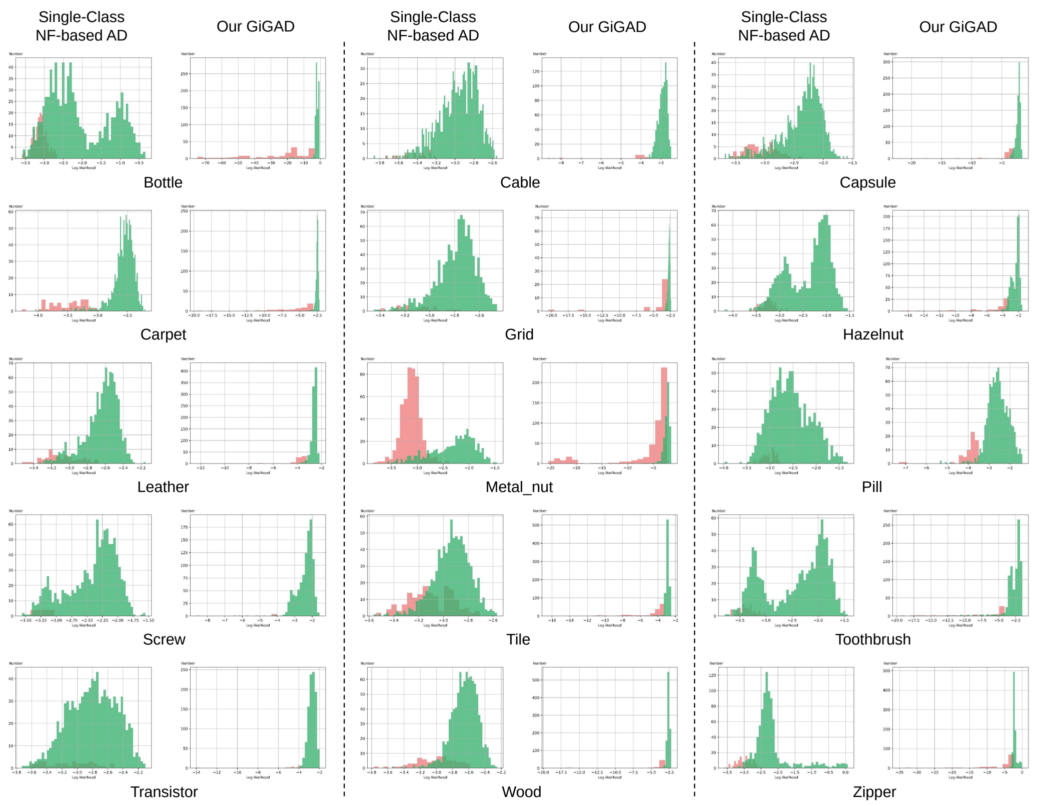

Figure 5: **Log-likelihood histograms on MVTecAD**. All categories are from the MVTecAD dataset.

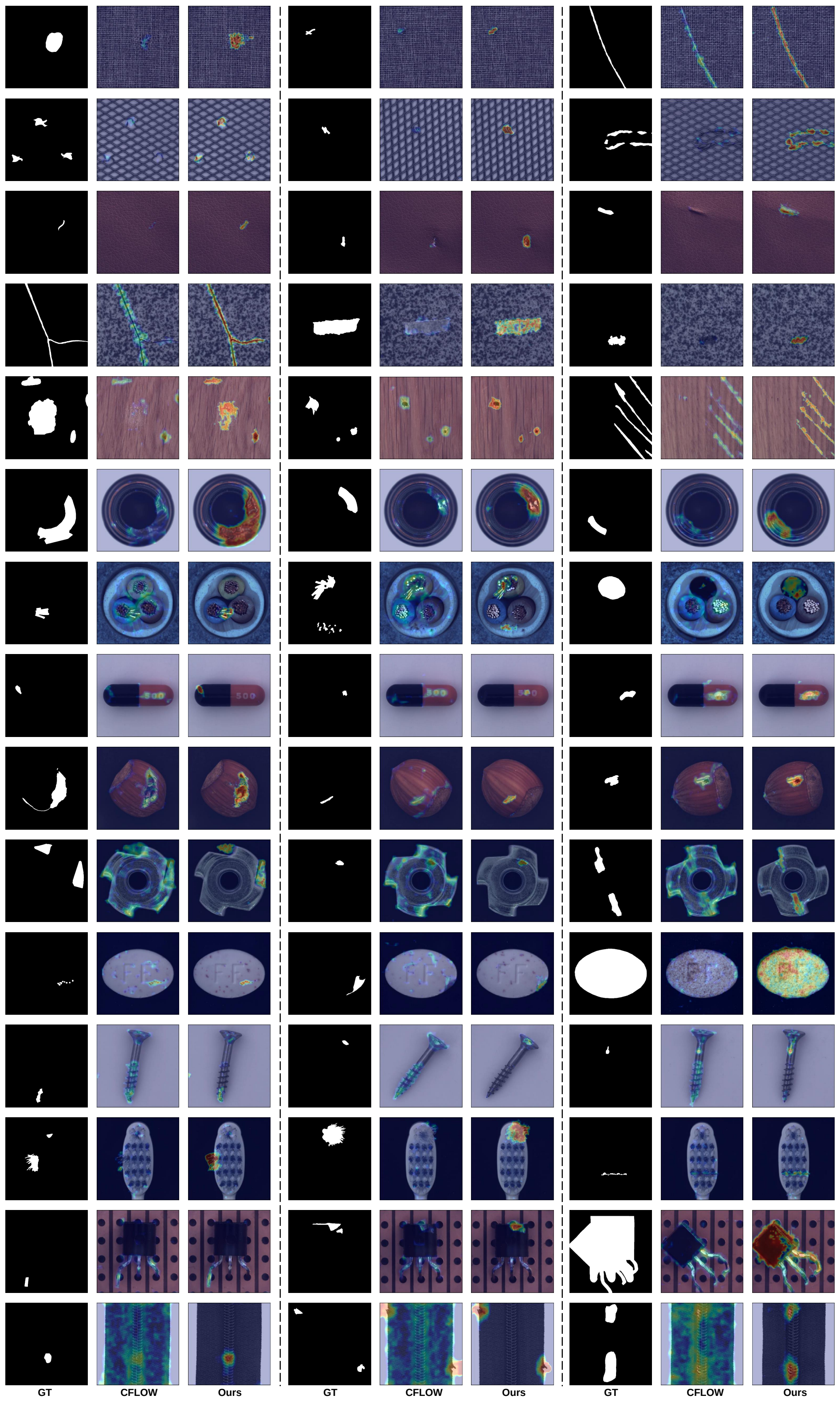

Figure 6: **Qualitative results on MVTecAD**. More visualization of anomaly localization maps generated by our method on industrial inspection data. All examples are from the MVTecAD dataset.