Appendix

A. Blockwise Masking

Blockwise Masking is utilized to generate continuous masked regions during training, so as to better simulate the anomaly occurrence. In this strategy, a block of image patches is masked each step, we repeat the masking step until obtaining enough masked patches. For each masking step, we set the minimum number of patches to 4 and the maximum number of patches to mN (total number of masked patches), and we randomly choose an aspect ratio for the masking block. Then, we randomly choose the coordinates of the upper left corner of the masking block. A block of image patches can be masked with the coordinates and the block size. The procedure of Blockwise Masking is summarized in Algorithm 1.

Algorithm 1: Blockwise Masking

Input: $N(= H \cdot W)$ image patches, mask ratio: m
Output: $\mathcal{M}$ (M Masking patches)
Initialize $\mathcal{M} \leftarrow \{\}$
repeat
  $sz \leftarrow \mathrm{Rand}(4, mN - |\mathcal{M}|)$ ▷ Block size
  $ar \leftarrow \mathrm{Rand}(0.3, \frac{1}{0.3})$ ▷ Aspect ratio
  $bh \leftarrow \sqrt{sz \cdot ar}$ ▷ Block height
  $bw \leftarrow \sqrt{sz/ar}$ ▷ Block width
  $t \leftarrow \mathrm{Rand}(0, h - bh)$ ▷ Top coordinate
  $l \leftarrow \mathrm{Rand}(0, w - bw)$ ▷ Left coordinate
  $\mathcal{M} \leftarrow \mathcal{M} \cup \{p_{i \cdot W + j} : i \in [t, t + bh], j \in [l, l + bw]\}$
until $|\mathcal{M}| \geq mN$ ▷ Obtaining enough masked patches
return $\mathcal{M}$

B. Combinational Masking

Considering that generating more masked patches in harder-to-reconstruct regions is conducive to train more generalizable models with stronger reconstruction ability, we thus propose combinational masking strategy. This strategy is combined with region-limited masking, frequency-based masking, dynamic masking, and basic random masking and blockwise masking.

Region-limited Masking. Considering that the proportion of the foreground object in the image is small, it is easy to generate masked patches in the background regions of the image. However, the background is usually simple and easy to be reconstructed, so we need to pay more attention to the reconstruction effect in the foreground regions. Thus, we propose a region-limited masking strategy to only generate masked patches in the foreground regions. By region-limited masking strategy, we can generate more masked patches in the foreground regions, it is conducive for the model to learn how to effectively reconstruct the foreground regions.

Frequency-based Masking. The high-frequency regions of the image generally represent more details and are more difficult to be reconstructed. Thus, the reconstruction effect in the high-frequency regions should receive more attention. We can convert the image to the frequency domain, and then calculate the frequency of each image patch. All frequency values are then normalized into range [0,1] by max-min normalization, and these normalized values are utilized as sampling coefficients. With these sampling weights, we can perform weighted sampling to obtain the masked patches. For the high-frequency patches, we can generate larger sampling weights, thus more masked patches can be generated from these regions.

Dynamic Masking. The model would have different reconstruction effects for different regions, and the reconstruction error for each patch can represent the learning status of the model for this patch. We propose dynamic masking strategy to make the model pay more attention to the image patches with worse reconstruction status. We can utilize the model's reconstruction error for each patch as the model's current reconstruction state in this patch. By recording the reconstruction errors of each patch, we can obtain the model's reconstruction status of different patches. When generating masked patches, we can normalize all recorded reconstruction errors into range [0,1] by max-min normalization, and these normalized values are utilized as sampling coefficients. With these sampling weights, we can perform weighted sampling to obtain the masked patches. For image patches with larger reconstruction errors, we can generate larger sampling weights, thus more masked patches can be generated from these regions.

Combinational Masking. As shown in Figure 4, we can combine these above mentioned masking strategies with the basic random masking and blockwise masking strategies to formulate a combinational masking strategy. When generating mask for one image, we first select whether to limit the masking region, and then select whether to use frequency-based masking or dynamic masking, and finally select random masking or a blockwise masking to generate the specific masked patches. Note that non region-limited masking means that masked patches can be generated from all image patches, normal masking means that all image patches have the same sampling weights to be selected as masked patches.

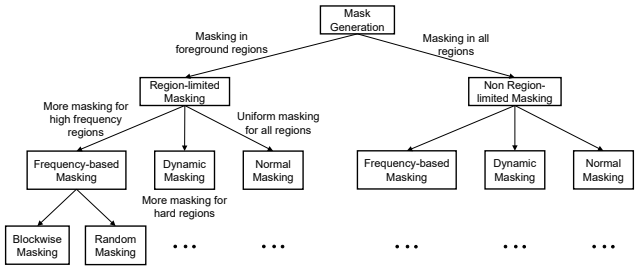

Figure 4: Combinational masking strategy.

C. Nominal Feature Prototype-guided Proposal Masking Module

During inference, to reduce the leakage of anomaly information in the masked image sequence, we propose a nominal feature prototype-guided flow model (ProtoFlow) to generate masked patches (suspicious anomalies). The ProtoFlow model is composed of two key designs: 1. Generating nominal feature prototypes and matching test patch

features. 2. Modeling the distribution of nominal residual features by normalizing flow. First, we indicate that our prototype-guided proposal masking module also should be class-adaptive (*i.e.*, it can mask suspicious anomalous regions as much as possible even in unseen classes rather than generating almost random masked patches). However, learning a class-adaptive model has two key issues: 1. *Normal data distribution of multiple classes is far more complex*. This may lead to learning a unified feature representation model more difficult. 2. *The normal patterns from different classes are significantly different*. This may result in many "mis-masking" of normal patches. (*i.e.*, one normal patch from unseen classes may be mistaken as abnormal due to it's quite different from the seen normal patches). Thus, we shouldn't model the distribution of normal features directly, because the normal patterns vary significantly across classes, causing models to be poorly class-adaptive. In this paper, we propose that we can address the two issues by modeling the distribution of normal residual features. The normal residual features can be obtained by subtracting the closet nominal prototype features from the normal features. As shown in Figure 5, normal residual features can address the two issues from two aspects: 1. Even if the normal feature distribution of multiple classes is far more complex, the distribution of normal residual features can be significantly simplified (see Figure 5(a)). 2. Even in unseen classes, the distribution of normal residual features would not remarkably shift from the learned distribution (see Figure 5(b)). We then describe the details of our prototype-guided proposal masking model in the following paragraphs.

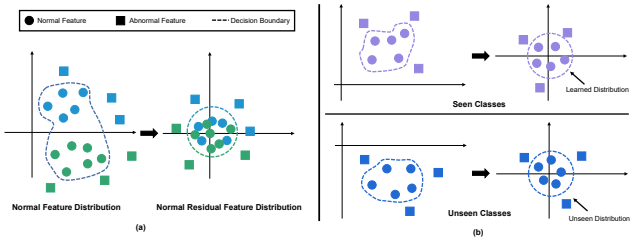

Figure 5: Feature distribution. (a) The feature distribution can be significantly simplified by converting to residual features. (b) Even in unseen classes, the unseen residual feature distribution won't shift remarkably from the learned distribution.

Generating Nominal Feature Prototypes. For normal images, we can utilize an ImageNet pre-trained encoder to extract normal multi-scale feature maps, then similar to (Defard et al. 2021), we aggregate these feature maps to one single scale. All features in the aggregated feature maps are denoted as the original normal features pool $\mathcal{P}_N$, and then we follow (Roth et al. 2022) to establish a nominal feature prototype pool $\mathcal{P}$ by the coreset subsampling mechanisms. The *minimax facility location coreset selection* algorithm is utilized (Sener and Savarese 2018; Sinha et al. 2020), the procedure to generate $\mathcal{P}$ can be defined as follows:

$$\mathcal{P}^* = \underset{\mathcal{P}\in\mathcal{P}_N}{\operatorname{argmin}} \max_{f_1\in\mathcal{P}_N} \min_{f_2\in\mathcal{P}} ||f_1 - f_2||_2 \tag{5}$$

Normalizing Flow. We employ normalizing flow (NF) (Dinh, Sohl-Dickstein, and Bengio 2016) to model the normal feature distribution because of its ability to estimate the exact likelihoods. We denote φ_θ as our normalizing flow. It is built as a composition of coupling layers (Dinh, Krueger, and Bengio 2015) such that $\varphi_\theta = \varphi_L \circ \cdots \circ \varphi_2 \circ \varphi_1$, where L is the total number of layers. The input distribution $p_\theta(x)$ can be estimated by model according to the change of variables formula as follows (Dinh, Sohl-Dickstein, and Bengio 2016):

$$p(x) = p_{\mathcal{Z}}(\varphi_\theta(x))\left|\det J\right| \tag{6}$$

where $J = \frac{\partial \varphi_\theta(x)}{\partial x}$ is the Jacobian matrix. $p_{\mathcal{Z}}$ is the latent distribution and generally implemented by the multivariate Gaussian distribution.

Training ProtoFlow. With nominal feature prototypes, we propose to train a normalizing flow model to distinguish the normal and abnormal with the guidance of prototypes. As shown in Figure 6, for each patch feature x_p, we can find a nearest nominal prototype $x_n = \underset{x\in\mathcal{P}}{\operatorname{argmin}} ||x - x_p||_2$ from the prototype pool $\mathcal{P}$, and then the residual patch feature $x_r = x_p - x_n$ is fed into the normalizing flow model. The input residual patch features will be transformed into latent variables $z = \varphi_\theta(x_r)$ by the NF model. When training the NF model, we can assume the latent variables to obey $\mathcal{N}(0, \mathbf{I})$. The optimization objective can be defined as follows (see (Dinh, Sohl-Dickstein, and Bengio 2016; Gudovskiy et al. 2022) for more details):

$$\mathcal{L} = \frac{1}{2}\varphi_\theta(x_r)^T\varphi_\theta(x_r) - \log\left|\det J\right| + \frac{d}{2}\log(2\pi) \tag{7}$$

where the d is the feature dimension.

Because abnormal features have a larger disparity than the normal prototypes, residual abnormal features in latent space can't obey the $\mathcal{N}(0, \mathbf{I})$ distribution. Therefore, we can measure the abnormality of each image patch by the distance of each residual feature to the $\mathcal{N}(0, \mathbf{I})$ distribution center. With the pre-defined mask ratio m, we can sort the abnormality of the image patches, and then select the top m percent patches in the abnormality ranking of the image patches as the masked patches.

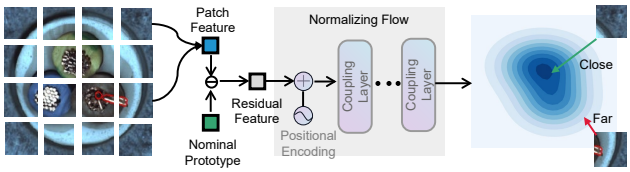

Figure 6: Architecture of nominal feature prototype-guided flow model (ProtoFlow). In the latent space, the normal residual features are closer to the distribution center, and the abnormal residual features are farther away from the distribution center.

D. Training Hyperparameters

Training Hyperparameters. The training hyperparameters for Vision Transformer and ProtoFlow are listed in Table 5.

Table 5: Training hyperparameters.

Models	Hyperparameters	ViT Base	ViT Large
Vision Transformer	Layers	12	24
	Hidden size	768	1024
	FFN hidden size	3072	4096
	Attention heads	12	16
	Attention head size	64	
	Patch size	16×16	
	optimizer	AdamW (Loshchilov and Hutter 2019)	
	Training epochs	300	
	Batch size	4	
	optimizer momentum	$\beta_1, \beta_2 = 0.9, 0.999$	
	Peak learning rate	1.5e-3	
	Minimal learning rate	1e-5	
	Learning rate schedule	Cosine	
	Warmup epochs	10	
	Gradient clipping	3.0	
	Weight decay	0.05	
	Input resolution	224×224	
ProtoFlow	Layers	4	
	Hidden size	272	
	Position embedding size	128	
	Clamp alpha	1.9	
	Pre-trained Encoder	Efficientnet-b6	
	Training epochs	200	
	Warmup epochs	2	
	Batch size	32	
	Learning rate	2e-4	
	Learning rate schedule	Cosine	
	Learning rate decay	0.1	
	Learning rate decay epochs	[96,144,176]	
	Input resolution	224×224	

Table 6: Mask ratios for different classes.

MVTecAD				
Carpet	Grid	Leather	Tile	Wood
0.16	0.16	0.12	0.38	0.46
Bottle	Cable	Capsule	Hazelnut	Metal nut
0.42	0.28	0.16	0.36	0.48
Pill	Screw	Toothbrush	Transistor	Zipper
0.48	0.04	0.22	0.48	0.2

Mask Ratios. We select a suitable mask ratio for each class through extensive experiments, all mask ratios are listed in Table 6.

E. More Experimental Results

Quantitative Results. More experimental results under the cross-class setting are shown in Table 7. In table 7, we divide the dataset by each class. For each class, we train the model with all remaining classes and test to detect anomalies in this class. Our approach can outperform the best unsupervised SOTA method by a significantly large margin with 13.2% and 6.7% on image-level and pixel-level AUROC, respectively. Compared with the class-adaptive AD method (RegAD), our approach can also obtain 4.3% image-level AUROC improvement.

Number of Normal Samples. We further show the results of our method by providing different numbers of unseen normal samples under the cross-class setting. The results are shown in Table 8. It can be found that even if only 50 normal samples are provided, our method can already achieve the performance of these SOTA methods (see Table 2) for cross-class anomaly detection.

Qualitative Results. More qualitative results under the cross-class setting are shown in Figure 7. In Figure 7, we show localization anomaly maps for all classes of the MVTecAD dataset. The experimental setting is the same as in Table 3.

Table 7: AUROC results on the MVTecAD dataset under the cross-class settings. For each class, we train the models with all remaining classes and test to detect anomalies in this class. The best results are in red and the second-best are in blue. $\cdot/\cdot$ means image-level AUROC and pixel-level AUROC, respectively.

Datasets	Cross-Class Setting							
	DFR	PaDiM	PatchSVDD	DRAEM	MSFD	CFLOW	RegAD	PMAD (ours)
Carpet	0.467/0.221	0.675/0.841	0.587/0.704	0.644/0.609	0.976/0.985	0.965/0.980	0.985/0.989	0.994/0.985
Grid	0.675/0.296	0.714/0.377	0.755/0.667	0.627/0.616	1.000/0.969	0.822/0.845	0.915/0.887	0.985/0.950
Leather	0.628/0.624	0.968/0.954	0.735/0.751	0.904/0.950	0.999/0.989	0.996/0.988	1.000/0.989	1.000/0.991
Tile	0.604/0.304	0.825/0.651	0.966/0.791	0.861/0.640	0.955/0.919	0.810/0.874	0.974/0.952	0.997/0.948
Wood	0.989/0.837	0.975/0.743	0.953/0.772	0.966/0.870	0.890/0.919	0.901/0.905	0.994/0.946	0.982/0.933
Bottle	0.360/0.585	0.831/0.837	0.810/0.844	0.687/0.690	0.973/0.946	0.702/0.867	0.998/0.975	1.000/0.983
Cable	0.515/0.673	0.500/0.746	0.616/0.809	0.534/0.611	0.605/0.785	0.513/0.729	0.806/0.949	0.983/0.975
Capsule	0.234/0.906	0.439/0.905	0.611/0.879	0.605/0.843	0.660/0.918	0.618/0.896	0.763/0.982	0.885/0.985
Hazelnut	0.949/0.960	0.814/0.917	0.861/0.939	0.883/0.964	0.739/0.918	0.965/0.985	0.771/0.896	0.997/0.983
Metal nut	0.571/0.674	0.512/0.738	0.466/0.782	0.707/0.728	0.733/0.787	0.983/0.969	0.755/0.621	0.998/0.967
Pill	0.547/0.784	0.515/0.758	0.759/0.876	0.705/0.924	0.789/0.810	0.472/0.733	0.806/0.978	0.922/0.961
Screw	0.540/0.925	0.522/0.920	0.542/0.824	0.917/0.948	0.598/0.931	0.504/0.918	0.634/0.971	0.713/0.928
Toothbrush	0.606/0.896	0.569/0.893	0.708/0.841	0.686/0.927	0.783/0.937	0.611/0.915	0.985/0.987	0.892/0.981
Transistor	0.353/0.571	0.450/0.428	0.718/0.907	0.637/0.561	0.661/0.719	0.643/0.597	0.934/0.968	0.987/0.976
Zipper	0.503/0.735	0.424/0.706	0.748/0.632	0.777/0.802	0.978/0.963	0.601/0.918	0.940/0.974	0.989/0.958
MVTecAD Mean	0.569/0.667	0.649/0.761	0.722/0.801	0.743/0.779	0.823/0.900	0.712/0.845	0.912/0.967	0.955/0.967

Table 8: AUROC results on two real-world AD datasets under the cross-class setting with different numbers of normal samples. Seen classes (textures or objects) mean all the other classes apart from the unseen classes.

Datasets	Seen Classes (train)	Unseen Classes (test)	Shots of Normal Samples				
			1	5	10	20	50
MVTecAD	Seen Textures	Grid	0.531/0.586	0.600/0.724	0.593/0.666	0.558/0.760	0.728/0.802
		Tile	0.981/0.900	0.991/0.871	0.994/0.903	0.986/0.891	0.984/0.923
		Wood	0.914/0.887	0.965/0.886	0.980/0.900	0.966/0.900	0.969/0.903
		Mean	0.809/0.791	0.852/0.827	0.856/0.823	0.837/0.850	0.894/0.876
	Seen Objects	Hazelnut	0.685/0.895	0.745/0.930	0.778/0.946	0.788/0.946	0.851/0.955
		Metal nut	0.429/0.713	0.626/0.810	0.625/0.862	0.600/0.861	0.717/0.874
		Pill	0.643/0.857	0.637/0.871	0.697/0.872	0.577/0.902	0.658/0.906
		Toothbrush	0.736/0.927	0.692/0.943	0.700/0.941	0.656/0.939	0.736/0.948
		Zipper	0.746/0.861	0.625/0.909	0.760/0.915	0.801/0.916	0.795/0.916
		Mean	0.648/0.851	0.665/0.892	0.712/0.907	0.684/0.913	0.751/0.920
MVTecAD	Seen Textures	Carpet	0.957/0.954	0.985/0.975	0.976/0.978	0.984/0.985	0.995/0.985
		Leather	0.980/0.974	1.000/0992	1.000/0.991	1.000/0.991	1.000/0.990
		Mean	0.968/0.964	0.992/0.984	0.988/0.984	0.992/0.988	0.998/0.987
	Seen Objects	Bottle	0.886/0.935	0.971/0.954	0.964/0.953	0.923/0.955	0.934/0.958
		Cable	0.705/0.787	0.638/0.859	0.642/0.864	0.737/0.871	0.741/0.880
		Capsule	0.493/0.915	0.583/0.939	0.599/0.924	0.621/0.933	0.747/0.948
		Screw	0.435/0.882	0.575/0.918	0.571/0.922	0.520/0.931	0.567/0.931
		Transistor	0.503/0.608	0.473/0.774	0.546/0.767	0.532/0.727	0.632/0.730
		Mean	0.604/0.825	0.648/0.889	0.664/0.886	0.667/0.883	0.724/0.890

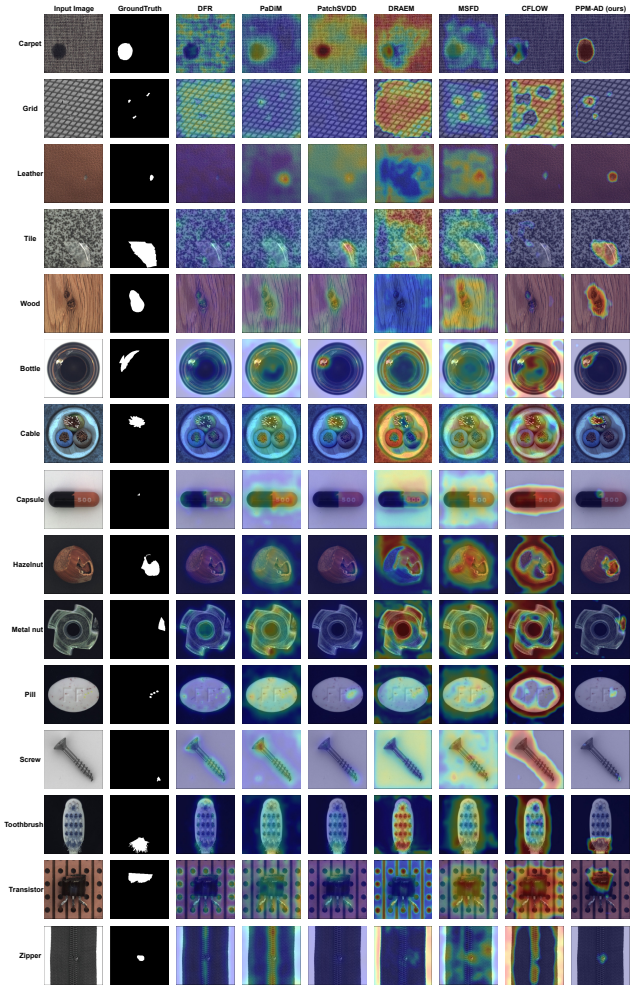

Figure 7: Qualitative results. The anomaly score maps are generated under the cross-class setting, where the training set doesn't contain the shown classes.