A. Implementation Details

For ViT[14], we adopt checkpoints from OpenAI's official repository[34], and for WideResNet[53], we utilize official checkpoint provided by PyTorch[33]. The default size of the memory bank is set to 196×768 for ViT-B[14], 256×1024 for ViT-L[14], and 196×512 for WideResNet[53]. The pixel feature was taken from outputs of the 6-th layer for ViT-B[14], the 10-th layer for ViT-L[14], and the 3rd layer for WideResNet-50[53]. Further model architecture and size details are in Tab. 11.

For Adapters and modulate layers, we use linear layers by default with their parameters set to $D \times D$, where D represents the feature dimension of the inserted layer. For instance, the size of the linear layer is 1024×1024 for ViT-L[14]. We use SGD[36] with a learning rate of 1×10^{-3} for adapters and 5×10^{-5} for FMN. The training is conducted over 20 epochs for the first and 10 for the second stages. The overall training takes $1 \sim 2$ hours in a single NVIDIA RTX-3090 in MVTec AD[7] dataset.

B. Further Explanation of Experiment Settings

Continual anomaly detection defines a multi-class training set $I_{train}^{total} = \{I_{train}^1, I_{train}^2, \ldots, I_{train}^n\}$ and corresponding test set $I_{test}^{total} = \{I_{test}^1, I_{test}^2, \ldots, I_{test}^n\}$. While unsupervised anomaly detection trains a separate model with each I_{train}^i, continual anomaly detection requires training a unified model for all classes in a sequential manner. During inference, the unified model is tested on all past test sub-datasets in a random order.

Few-shot anomaly detection aims to detect and localize anomalies using only a limited number of normal samples. In our experiments, we specifically consider 1, 2, 4, and 8 normal samples.

Supervised anomaly detection involves training with both normal and anomalous samples, simulating real-world scenarios where a limited number of anomalous examples might be available. We conduct experiments solely on the VisA dataset[55], as it is the only dataset among those considered that provides an official split for this setting.

C. Experiments on More Datasets

To further evaluate the performance of our model, we conducted experiments on continual and few-shot anomaly detection using more datasets including VisA[55], BTAD[29], MVTec LOCO AD[8]. The experimental results for the continual and few-shot settings are presented in Tab. 8 and Tab. 10 respectively. For the VisA dataset [55], the results of UCAD[1] and WinCLIP[18] were obtained from their original papers, while for the other datasets, we utilized the publicly available open-source implementations [44, 56]. As demonstrated in Tab. 8, our method consistently outperforms the current state-of-the-art method, UCAD[1], in the continual setting, with the exception of the pixel-AP on the MVTec LOCO AD dataset[8]. In the few-shot setting, our method achieves comparable or superior results compared to WinCLIP[18] and consistently improves the performance of the original PatchCore[37]. Notably, however, all evaluated methods exhibit inferior performance on the MVTec LOCO AD dataset[8] in both continual and few-shot settings, indicating that the detection of logical anomalies requires further investigation.

Dataset	I-AUROC UCAD, DFM	P-AP UCAD, DFM
VisA[55]	87.4, 94.1	30.0,31.8
BTAD[29]	93.2, 94.5	34.3, 53.2
MVTec LOCO AD [8]	74.4, 76.3	3.2, 2.6

Table 8. Results of continual anomaly detection on more datasets

D. Additional Ablations

Ablation of backbones on continual setting. We implement our framework in different backbones, including ViT[14] with various size and pre-training methods and WideResNet[53]. The experiment is conducted in continual setting using the MVTec AD[7] dataset. ViT-B(MAE) and ViT-B(DINO) denote backbone pre-trained with MAE[15] and DINO[9]. ViT-B(ImageNet) denotes the backbone trained with classification tasks on ImageNet[13] only. As shown in Tab. 9, the backbone pre-trained with CLIP[34] shows significantly better performance than ViT pre-trained by MAE[15] and classification task in ImageNet[13]. The results further prove that better feature representation matters in anomaly detection, which is our motivation to adapt feature extractors.

backbone	I-AUROC	P-AP
ViT-B (ImageNet)	91.54	43.53
ViT-B (MAE)	93.16	45.50
ViT-B (DINO)	95.90	48.40
ViT-B (CLIP)	96.44	49.91
ViT-L (CLIP)	**96.94**	**51.07**
WideResNet-50	94.87	49.78

Table 9. Ablation of different backbones on continual anomaly detection setting using MVTec AD[7] dataset .

Time and memory efficiency analysis on more models. We provide further statistical results of time and memory consumption on Tab. 12 and model size on Tab. 11. As shown in Tab. 11, our method adds negligible parameters to the original PatchCore[37]. Additionally, as discussed in Sec. 4.5, our methods show advantage in inference time

Instance-AUROC									
K	VisA[55]			BTAD[29]			MVTec LOCO AD[8]		
	WinCLIP	PatchCore	DFM	WinCLIP	PatchCore	DFM	WinCLIP	PatchCore	DFM
1	83.8	79.9	84.0	90.0	59.6	87.4	61.3	54.5	60.5
2	84.6	81.6	86.0	89.6	85.3	88.8	63.9	59.7	64.6
4	87.3	85.3	89.8	90.4	87.4	89.8	67.6	62.0	64.8
Pixel-AUROC									
K	VisA[55]			BTAD[29]			MVTec LOCO AD[8]		
	WinCLIP	PatchCore	DFM	WinCLIP	PatchCore	DFM	WinCLIP	PatchCore	DFM
1	96.4	95.4	96.4	94.7	96.0	96.9	64.0	83.4	85.5
2	96.8	96.1	96.8	94.7	97.0	97.0	64.4	82.0	83.7
4	97.2	96.8	97.1	94.9	97.2	97.1	65.2	82.3	83.4

Table 10. Results of few shot anomaly detection on more datasets

Backbone	Layers	Feature Layer	Feature Dimension	Parameters	
				PatchCore	DFM
ViT-B	10	6	768	47.3 M	49.9 M
ViT-L	20	10	1024	152 M	164 M
WideResNet-50	4	3	512	27.0 M	29.1 M

Table 11. Detailed information about model architecture and size. We insert adapters into all layers for DFM here

over PatchCore[37] but requires more memory as shown in Tab. 12

Backbone	Method	FPS	Memory
		Train / Infer	Train / Infer
ViT-B	PatchCore	- , 20.32	- , 2713
ViT-B	DFM	24.09, 49.90	4923, 3671
ViT-L	PatchCore	- , 17.99	- , 3383
ViT-L	DFM	20.9, 42.74	7527, 5255

Table 12. Training and Inference efficiency comparison between our method and PatchCore[37]

Ablation of adapter layers on supervised setting. We conduct experiments in a supervised setting to further investigate the influence of adapter layers. Using ViT-L as the backbone and output of the 10-th layer as patch features, we divide accessible layers into three sections based on varying depths. As shown in Tab.13, our method consistently outperforms the frozen backbone, validating the effectiveness of utilizing real anomalous samples for improved anomaly detection. Additionally, the results demonstrate that, in general, deeper adapters result in more significant performance improvements, which aligns with the findings in the few-shot setting. As the number of adapters increases, the model's performance improves, contrasting with the results observed in the few-shot setting. We hypothesize that in the few-shot scenario, the scarcity of samples leads to over-fitting when the number of trainable parameters is too large.

1,2,3	4,5,6	7,8,9	Instace-AUROC	Pixel-AP
frozen backbone			91.5	15.0
✓			94.5	18.0
	✓		94.7	17.5
		✓	95.0	18.1
	✓	✓	95.4	19.7
✓	✓	✓	**95.9**	**20.0**

Table 13. Ablation of different adapter layers under supervised setting on VisA dataset [55]

	Instance-AUROC		Pixel-AUROC	
Backbone	PatchCore	DFM	PatchCore	DFM
ViT-B	97.93	98.35	97.26	97.62
ViT-L	98.52	99.06	97.63	97.78

Table 14. Results on standard unsupervised anomaly detection using MVTec AD[7] dataset.

Results on unsupervised anomaly detection. We re-implement PatchCore[37] on unsupervised setting. The memory bank is set to 1024 x 1024 for ViT-L and 980 x 768 for ViT-B for both PatchCore[37] and our method here. Experimental results are presented in the Tab.14. Note that our implementation here didn't use multi-layer

feature aggregation and re-weighting, utilized in the original PatchCore[37]. It turned out our method steadily shows better performance of PatchCore[37] in the unsupervised anomaly detection. The improvement is limited since the performance of original PatchCore[37] already approaches saturation. However, in a few-shot and continual settings where PatchCore[37] shows inferior performance, our methods can have significant improvement.