# PatchGuard: Adversarially Robust Anomaly Detection and Localization through Vision Transformers and Pseudo Anomalies

Supplementary Material

## A. Additional Related Work

In this section, we shed more light on some of the previous pioneering works in AD and AL, and their pipeline.

**Recontrast** [31] innovates anomaly detection through contrastive reconstruction by adapting encoder and decoder networks specifically to the target domain. Unlike traditional approaches relying on frozen pre-trained encoders, it embeds contrastive learning elements into feature reconstruction to stabilize training, avoid pattern collapse, and improve domain relevance. This ensures precise anomaly detection in industrial and medical imaging tasks.

**Transformaly** [16] focuses on anomaly detection using a dual-feature approach. It leverages a pre-trained ViT to extract agnostic feature vectors and employs teacher-student training to fine-tune a student network on normal samples. This complementary representation enhances anomaly detection, achieving high AUROC results in unimodal and multimodal settings.

**GeneralAD** [72] utilizes a Vision Transformer-based framework for anomaly detection across diverse domains. It introduces a self-supervised anomaly feature generation module to create pseudo-abnormal samples by applying operations like noise addition and patch shuffling. These are fed into an attention-based discriminator to detect and localize anomalies while producing interpretable anomaly maps.

**GLASS** [13] uses gradient ascent for anomaly synthesis. This unified approach combines global anomaly synthesis to manipulate feature manifolds and local strategies to refine weak anomalies. Together, it improves the precision and breadth of industrial anomaly detection and localization.

## B. Augmentation Details

In this section, we clarify the soft and hard augmentations, $t_i^s, t_i^h$, which are used in the foreground estimation part of our method. Here, we explain in more detail what these augmentations are and what are the rationales behind choosing them.

**Soft Augmentations.** Soft augmentations refer to transformations that do not alter the semantic content of the image, preserving the original context and interpretability of the visual information. Examples include color jitter (which modifies brightness, contrast, saturation, or hue slightly), color tint (adding a consistent color overlay), grayscale conversion (removing color information but maintaining structure), and minor Gaussian noise (introducing slight variations that mimic sensor noise). These transformations ensure that the augmented images remain perceptually similar to the originals, focusing on maintaining semantic integrity while introducing subtle variability. Such augmentations are critical in our method for refining the estimation of the foreground without distorting the regions of interest.

**Hard Augmentations.** Hard augmentations, in contrast, involve transformations that can significantly alter the semantic meaning or structure of the image. Examples include large rotations (which may distort spatial relationships), extreme cropping (removing substantial portions of the image, potentially excluding key objects), elastic transformations (which deform image structures in ways that can obscure original semantics), and heavy noise injection. These transformations challenge the robustness of the foreground estimation by introducing substantial changes, effectively creating conditions where the boundaries of semantic preservation are tested. In our method, hard augmentations are designed to evaluate the resilience of the anomaly generation process and its ability to adapt under challenging conditions.

## C. Additional Ablation Studies

### C.1. Clean Training

In this section, we chose to omit adversarial training and instead trained our method using standard training while keeping all other components unchanged. The results reveal an improvement in clean performance, highlighting PatchGuard's effectiveness across various training and evaluation scenarios. These results are detailed in Table 7. Additionally, we evaluated the clean-trained model under an adversarial setup, demonstrating that our pipeline benefits significantly from adversarial training. This underscores the impact of our regularization technique in adversarial training, which enhances the robustness of attention-based mechanisms against adversarial examples.

Table 7. Performance of the model trained without adversarial training under clean and adversarial setups.

| Method | Task | Dataset | | | |
|---|---|---|---|---|---|
| | | MVTec AD | VisA | BTAD | BraTS2021 |
| Clean | AD | 94.7 / 12.9 | 93.9 / 16.3 | 91.3 / 11.6 | 97.1 / 10.7 |
| | AL | 97.4 / 12.5 | 98.0 / 8.1 | 95.7 / 11.6 | 98.5 / 12.3 |

## C.2. Ablation on $\delta$

To determine the attention degree for an output token in an attention head, you need to identify how many of the total input tokens it attends to more than the others. In our case, the ViT model has 256 input tokens. Intuitively, we set $\delta$ to $\frac{1}{255}$, drawn from a uniform distribution. In Table 8, we present an ablation study on the value of $\delta$.

Table 8. Ablation study on the value of $\delta$ and its effect on the attention degree in the ViT model.

| $\delta$ | Task | Dataset | | | |
|---|---|---|---|---|---|
| | | **MVTec AD** | **VisA** | **BTAD** | **BraTS2021** |
| $\frac{1}{2\times256}$ | AD | 89.0 / 69.1 | 88.8 / 72.1 | 86.7 / 79.4 | 94.4 / 79.4 |
| | AL | 93.1 / 70.9 | 96.8 / 82.0 | 94.0 / 70.8 | 97.8 / 90.1 |
| $\frac{1}{256}$ | AD | 88.1 / 71.1 | 88.5 / 74.3 | 85.3 / 82.1 | 94.3 / 81.0 |
| | AL | 92.7 / 73.8 | 96.9 /85.2 | 93.2 /73.0 | 97.7 /94.5 |
| $\frac{2}{256}$ | AD | 85.1 / 71.9 | 84.7 / 75.8 | 82.4 / 82.8 | 91.7 / 81.3 |
| | AL | 88.6 / 74.3 | 93.4 / 85.6 | 90.3 / 74.1 | 92.7 / 81.9 |

## C.3. Diffenet ViT Backbone

As mentioned in the implementation details, we used a ViT small model with a patch size of 14, initialized with random weights. In this section, we evaluate our method by replacing the backbone with larger variations of ViT models (note that these models use random weights and are not pre-trained). All other components are kept fixed. As shown in Table X, the results demonstrate that our method achieves high performance and consistency across different backbones.

Table 9. Evaluation of our method with different ViT backbones initialized with random weights. The results demonstrate high performance and consistency across various backbone configurations.

| ViT | Task | Dataset | | | |
|---|---|---|---|---|---|
| | | **MVTec AD** | **VisA** | **BTAD** | **BraTS2021** |
| Small(*Ours*) | AD | 88.1 / 71.1 | 88.5 / 74.3 | 85.3 / 82.1 | 94.3 / 81.0 |
| | AL | 92.7 / 73.8 | 96.9 /85.2 | 93.2 /73.0 | 97.7 /94.5 |
| Base | AD | 89.1 / 71.0 | 87.9 / 73.1 | 84.5 / 81.7 | 93.0 / 82.3 |
| | AL | 91.0 / 72.6 | 95.8 / 82.1 | 94.7 / 74.8 | 98.4 / 94.9 |
| Large | AD | 90.0 / 70.6 | 87.5 / 74.7 | 85.5 / 81.9 | 95.1 / 81.6 |
| | AL | 92.9 / 73.4 | 95.8 / 86.0 | 94.1 / 73.5 | 96.7 / 93.2 |

## C.4. Backbone

In Sections 4 and 5, we provided intuitions and theoretical insights on why vision transformers achieve better adversarial robustness than convolution-based methods. In this section, we use convolution-based backbones in our pipeline instead on the ViT, while preserving all other components as they are. To support this claim, we provide the detection and localization results in Table 10.

Adapting convolution-based backbones like ResNet [32] to our patch-based pipeline poses certain challenges. To integrate ResNet, we incorporate a binary classification layer at the model's final stage. Each image is divided into patches manually, following the same patching approach used by the vision transformer. Anomaly scores are then computed for each patch independently, and the final anomaly detection decision is based on the top-$k$ patches with the highest scores. For a fair comparison, we apply the same hyperparameters used in our original method.

To adapt U-Net [64], we shift from a patch-wise approach to pixel-wise localization, given the architectural constraints of U-Net. Notably, the top-$k$ selection used previously is incompatible in this context. Instead, we employ a top-$p$ percent pixel selection for the anomaly detection decision, where $p = \frac{k}{N}$, and $N$ represents the total number of patches in an image.

## C.5. Integrating Sparse Attention Mechanism into Our Methodology

We evaluate the performance of our method after incorporating BigBird [87], a sparse attention mechanism, as shown in Table 11. The results reveal two key findings. First, applying the sparse attention mechanism generally reduces the robustness of our method. This aligns with our intuition and theory, which suggest that a higher attention degree enhances model robustness, while sparse attention decreases the attention degree. Second, our regularization term remains effective even in the sparse attention setup—when applied, it still improves the model's robustness.

## C.6. Impact of Regularization Layer on Model Performance

We investigated the effect of applying regularization to different layers of the network. The results, shown in Table 12, indicate that regularization in inner layers generally improves model robustness. However, the last layer performs slightly better according to our experiments.

Table 10. A study on the performance of various backbone networks, as alternatives to the ViT, within our architecture.

| Backbone | Task | Dataset | | | |
|---|---|---|---|---|---|
| | | **MVTec-AD** | **VisA** | **BTAD** | **BraTS2021** |
| **U-Net** | AD | 84.7 / 15.1 | 80.0 / 15.7 | 76.9 / 14.0 | 70.3 / 12.9 |
| | AL | 85.4 / 17.8 | 81.8 / 13.9 | 79.6 / 18.2 | 76.3 / 16.0 |
| **Resnet50** | AD | 88.1 / 27.6 | 84.9 / 23.3 | 85.7 / 23.9 | 86.0 / 23.5 |
| | AL | 87.1 / 28.3 | 83.7 / 24.6 | 86.0 / 24.7 | 85.1 / 23.9 |

Table 11. Performance comparison of our method with and without the BigBird sparse attention mechanism.

| Backbone | Task | MVTec AD | VisA | BTAD | BraTS2021 |
|---|---|---|---|---|---|
| BigBird | AD | 86.7 / 51.7 | 90.3 / 49.7 | 86.0 / 58.9 | 95.9 / 61.8 |
| | AL | 91.4 / 53.1 | 93.9 / 59.8 | 92.5 / 51.4 | 96.4 / 68.5 |
| BigBird + Our Regularization | AD | 85.6 / 63.0 | 88.5 / 61.4 | 86.2 / 69.8 | 92.2 / 73.1 |
| | AL | 90.0 / 64.7 | 92.1 / 73.6 | 90.5 / 62.2 | 94.6 / 78.9 |
| Ours | AD | 88.1 / 71.1 | 88.5 / 74.3 | 85.3 / 82.1 | 94.3 / 81.0 |
| | AL | 92.7 / 73.8 | 96.9 / 85.2 | 93.2 / 73.0 | 97.7 / 94.5 |

Table 12. Performance of Regularization at Different Layers

| Regularization Layer | Task | MVTec AD | VisA | BTAD | BraTS2021 |
|---|---|---|---|---|---|
| $(N-2)^{\text{th}}$ | AD | 87.3 / 68.2 | 89.7 / 72.9 | 86.3 / 79.1 | 93.5 / 75.4 |
| | AL | 91.5 / 70.6 | 96.0 / 83.1 | 91.7 / 68.5 | 96.8 / 90.6 |
| $(N-1)^{\text{th}}$ | AD | 89.0 / 69.4 | 87.5 / 73.1 | 84.3 / 82.3 | 92.6 / 79.9 |
| | AL | 93.2 / 70.6 | 95.7 / 86.1 | 93.1 / 71.7 | 97.0 / 93.2 |
| $N^{\text{th}}$ (Last Layer) | AD | 88.1 / 71.1 | 88.5 / 74.3 | 85.3 / 82.1 | 94.3 / 81.0 |
| | AL | 92.7 / 73.8 | 96.9 / 85.2 | 93.2 / 73.0 | 97.7 / 94.5 |

## D. Dataset Details

We conducted our experiments on eight datasets covering a diverse range of domains, from industrial to medical applications. The medical datasets include BraTS2021 and Head-CT, while the remaining six datasets focus on industrial and synthetic anomaly detection and localization tasks. Below, we provide detailed descriptions of each dataset.

- MVTec AD: MVTec AD is a dataset for benchmarking anomaly detection methods in industrial inspection. It includes over 5,000 high-resolution images across fifteen object and texture categories. Each category contains defect-free training images and a test set with both normal and defective samples, featuring defects like scratches, dents, and misalignments.
- VisA: The VisA dataset contains 12 object subsets with 10,821 images, comprising 9,621 normal samples and 1,200 anomalous samples. The subsets include printed circuit boards, multi-instance objects like Capsules and Macaroni, and roughly aligned objects such as Cashew and Chewing Gum. Anomalies include surface defects like scratches and dents, and structural issues such as missing components.
- BTAD: The BTAD [60] consists of 2,830 images of three industrial products. It provides samples with body and surface defects, intended for evaluating visual anomaly detection methods in industrial settings.
- MPDD: MPDD [40] is a dataset for defect detection in metal parts manufacturing, consisting of over 1,000 images with pixel-level defect annotations. The dataset is divided into six distinct classes and includes anomaly-free training samples and test samples with normal and defective parts, covering a variety of surface and structural defects.
- WFDD: WFDD [13] is a dataset for anomaly detection in textile inspection, comprising 4,101 woven fabric images across four categories: grey cloth, grid cloth, yellow cloth, and pink flower. Defects are categorized as block-shaped, point-like, or line-type, with pixel-level annotations.
- DTD-Synthetic: The DTD-Synthetic [3] is designed for anomaly detection and segmentation tasks, containing synthetic texture images generated from predefined texture patterns. It includes twelve classes of normal texture samples and those with artificially introduced anomalies such as structural distortions or irregular patterns.
- BraTS2021: BraTS2021 is a medical dataset for anomaly segmentation, containing 1,251 MRI cases with voxel-level annotations for tumor regions. Each case includes multiple imaging modalities (T1, T1ce, T2, and FLAIR). In this paper, only the FLAIR modality is used due to its sensitivity to tumor regions.
- Head-CT: The Head-CT [42] contains 200 head CT slices, evenly split between normal slices and those with hemorrhages, without distinguishing between hemorrhage types.

## E. Details of Adaptation of State-of-the-Art Methods for Adversarial Robustness

Before proposing a novel approach for adversarial AD and AL setups, our idea was to adapt existing state-of-the-art methods in the field and enhance their robustness through adversarial training [13, 31, 72, 82]. Adversarial training involves feeding adversarial examples to the model during training [54]. In the following section, we explain how we create adversarial examples for each method and the details of our best approach to make them robust, which are reported in Table 3.

Anomaly-free methods like PatchCore [65] and ReContrast [31] have anomaly-free training. For adapting PatchCore, we used an adversarially trained ResNet-50 [32] as the feature extractor. Adding adversarially generated normal samples to the memory bank was tested but did not improve performance. For adapting ReContrast, we replaced both the teacher and student networks with adversarially trained ResNet-50 models, and using adversarial samples generated by the PGD-100 attack [54] on the final anomaly map, we trained the network to improve robustness.

Embedding-space synthesis methods, such as GeneralAD [72] and SimpleNet [49], operate in the embedding space. For adapting SimpleNet, we employed an adversarially robust WideResNet-50 [86] as the feature extractor. Two strategies were tested: input-space adversarial training, which was ineffective due to the absence of anomaly samples, and embedding-space adversarial training, where the discriminator was adversarially trained using both normal and anomaly features. The latter approach performed better, as reported in Table 3, but the overall performance was still insufficient. For GeneralAD, due to the model's dependency on DINO [9] pre-trained weights, we could not find a proper robust pre-trained ViT backbone for adaptation that maintained

Table 13. Class-wise Clean and Adversarial AUROC (%) Results for Image-level and Pixel-level Evaluations on the MVTec-AD Dataset.

| Class Name | Image-level AUROC (%) | | Pixel-level AUROC (%) | |
|---|---|---|---|---|
| | Clean | Adversarial | Clean | Adversarial |
| Bottle | 97.6 | 84.7 | 96.7 | 84.6 |
| Cable | 87.3 | 74.0 | 97.2 | 77.8 |
| Capsule | 71.8 | 79.6 | 93.6 | 85.2 |
| Carpet | 83.9 | 43.2 | 95.8 | 53.7 |
| Grid | 95.7 | 74.9 | 93.1 | 55.5 |
| Hazelnut | 99.5 | 80.5 | 97.2 | 91.3 |
| Leather | 91.0 | 80.2 | 97.6 | 67.6 |
| Metal Nut | 88.8 | 54.4 | 87.9 | 75.3 |
| Pill | 81.5 | 59.1 | 86.7 | 76.3 |
| Screw | 55.4 | 56.6 | 93.8 | 86.5 |
| Tile | 95.3 | 71.7 | 86.2 | 64.0 |
| Toothbrush | 100 | 90.6 | 93.9 | 81.9 |
| Transistor | 94.1 | 84.3 | 95.4 | 84.8 |
| Wood | 93.1 | 56.0 | 89.6 | 57.4 |
| Zipper | 87.6 | 76.9 | 86.2 | 65.7 |

clean performance. We tested multiple approaches to make it robust while maintaining clean performance, and the best one was embedding-space adversarial training of the discriminator with access to both normal and anomaly features, and replacing the ViT with a robust pre-trained model on ImageNet [19].

Input-space synthesis methods like DRAEM [88] and GLASS [13] generate synthetic anomalies in the input space. In adapting DRAEM, adversarial samples were created using PGD-100 on the focal loss [46] of the anomaly map, and these samples were used to train both the reconstructive and discriminative sub-networks adversarially. For adapting GLASS, the best results were achieved by combining an adversarially trained feature extractor with adversarial samples (PGD-100) applied to both $L_n$ and $L_{las}$.

According to Table 3, state-of-the-art methods in AD and AL, even after adapting to adversarial training scenarios, still suffer from vulnerability to adversarial attacks and perform weakly.

## F. Per-Class Results

In this section, we present the per-class AUROC results for anomaly detection and localization using PatchGuard across the reported datasets, as detailed in Tables 13, 14, 15, 16, 17, and 18.

## G. Implementation Details

The optimizer used is `AdamW` [51], with a learning rate of 0.0008 and a weight decay of 0.00001. For learning rate scheduling, we utilize a `CosineAnnealingLR` scheduler with a decay factor of 0.0125, where the minimum learning rate ($\eta_{\text{min}}$) is calculated as `lr` $\times$ `lr_decay_factor`, and `T_max` is set to the number of epochs, which is set to 300 but we observe empirically that convergence usually happens much faster. The batch size for both training and testing is set to 16. The input image size is $224 \times 224$. We use a `ViT`

Table 14. Class-wise Clean and Adversarial AUROC (%) Results for Image-level and Pixel-level Evaluations on the VisA Dataset.

| Class Name | Image-level AUROC (%) | | Pixel-level AUROC (%) | |
|---|---|---|---|---|
| | Clean | Adversarial | Clean | Adversarial |
| Candle | 83.6 | 82.5 | 94.9 | 70.7 |
| Capsules | 77.7 | 66.2 | 97.1 | 57.0 |
| Cashew | 88.7 | 83 | 97.3 | 89.9 |
| Chewing gum | 92.2 | 73.5 | 97.7 | 92.2 |
| Fryum | 85.1 | 74.5 | 95.9 | 88.0 |
| Macaroni 1 | 86.5 | 66.2 | 97.0 | 84.8 |
| Macaroni 2 | 68.3 | 42.3 | 95.3 | 85.0 |
| Pcb 1 | 95.4 | 85.3 | 99.0 | 95.4 |
| Pcb 2 | 97.3 | 91.2 | 96.8 | 87.1 |
| Pcb 3 | 94.8 | 73.5 | 98.7 | 93.0 |
| Pcb 4 | 98.5 | 92.1 | 95.9 | 83.7 |
| Pipe fryum | 94.3 | 61.5 | 98.1 | 96.2 |

Table 15. Class-wise Clean and Adversarial AUROC (%) Results for Image-level and Pixel-level Evaluations on the BTAD Dataset.

| Class Name | Image-level AUROC (%) | | Pixel-level AUROC (%) | |
|---|---|---|---|---|
| | Clean | Adversarial | Clean | Adversarial |
| 01 | 98.6 | 96.1 | 91.7 | 77.1 |
| 02 | 65.3 | 65.6 | 92.1 | 64.2 |
| 03 | 92.0 | 84.6 | 95.8 | 77.8 |

Table 16. Class-wise Clean and Adversarial AUROC (%) Results for Image-level and Pixel-level Evaluations on the MPDD Dataset.

| Class Name | Image-level AUROC (%) | | Pixel-level AUROC (%) | |
|---|---|---|---|---|
| | Clean | Adversarial | Clean | Adversarial |
| Bracket Black | 83.4 | 60.1 | 92.1 | 88.4 |
| Bracket Brown | 84.5 | 77.4 | 91.0 | 84.7 |
| Bracket White | 79.8 | 52.9 | 92.1 | 85.9 |
| Connector | 94.3 | 92.5 | 93.8 | 76.8 |
| Metal Plate | 100 | 86.2 | 98.2 | 95.2 |
| Tubes | 70.4 | 42.8 | 95.9 | 88.9 |

Table 17. Class-wise Clean and Adversarial AUROC (%) Results for Image-level and Pixel-level Evaluations on the WFDD Dataset.

| Class Name | Image-level AUROC (%) | | Pixel-level AUROC (%) | |
|---|---|---|---|---|
| | Clean | Adversarial | Clean | Adversarial |
| Gray Cloth | 88.8 | 57.3 | 93.1 | 75.9 |
| Grid Cloth | 99.6 | 94.7 | 97.8 | 85.1 |
| Pink Flower | 52.3 | 33.0 | 94.4 | 63.0 |
| Yellow Cloth | 96.3 | 75.0 | 93.3 | 62.5 |

Table 18. Class-wise Clean and Adversarial AUROC (%) Results for Image-level and Pixel-level Evaluations on the DTD-Synthetic Dataset.

| Class Name | Image-level AUROC (%) | | Pixel-level AUROC (%) | |
|---|---|---|---|---|
| | Clean | Adversarial | Clean | Adversarial |
| Blotchy 099 | 86.1 | 73.2 | 94.7 | 87.5 |
| Fibrous 183 | 100 | 55.8 | 99.0 | 80.9 |
| Marbled 078 | 89.3 | 63.8 | 97.6 | 88.4 |
| Matted 069 | 93.1 | 53.8 | 97.1 | 78.1 |
| Mesh 114 | 94.4 | 74.9 | 97.6 | 73.6 |
| Perforated 037 | 99.9 | 81.1 | 94.4 | 55.5 |
| Stratified 154 | 91.6 | 42.5 | 98.2 | 76.2 |
| Woven 001 | 83.6 | 56.0 | 96.2 | 71.6 |
| Woven 068 | 88.5 | 55.5 | 93.6 | 71.4 |
| Woven 104 | 79.6 | 55.4 | 87.9 | 73.0 |
| Woven 125 | 95.3 | 56.9 | 98.2 | 71.6 |
| Woven 127 | 96.8 | 57.9 | 97.2 | 70.5 |

(Vision Transformer) small model as the feature extractor, which is not pre-trained. Finally, we perform a top-$k$ selection of the localization map achieved, to obtain a final AD decision, with $k$ being set to 5.

## H. Attack Adaptation Details

**Adaptation Classification Attack.** We evaluated PatchGuard's resilience against several advanced adapted attacks from the classification domain, including CAA, AutoAttack, A3, and PGD-1000. Originally designed to compromise classification tasks by exploiting the cross-entropy loss, these attacks were adapted for anomaly localization (AL) and anomaly detection (AD) tasks. The focus was on altering the sum of cross-entropy for all patches in detector models, aiming to increase the loss values for normal regions of test samples while decreasing them for anomalous regions. Adapting AutoAttack (AA) for AD and AL tasks posed significant challenges. AutoAttack comprises a suite of different attack methods, such as FAB, multi-targeted FAB, Square Attack, APGDT, APGD with cross-entropy loss, and APGD with DLR loss. The primary difficulty in adaptation arises because attacks using the DLR loss assume the model's output contains at least three elements, an assumption valid for classification tasks with three or more classes but not applicable to AD and AL tasks. Consequently, we replaced the DLR loss component in AutoAttack with a PGD attack. However, for the other attacks under consideration, no modifications were necessary.

**Adaptation Segmentation Attack.** We evaluate our method against advanced adapted semantic segmentation attacks, specifically SegPGD and SEA, with a key modification: instead of operating at the pixel level, our approach applies these attacks patch-wise. SegPGD is a segmentation-specific adaptation of the Projected Gradient Descent (PGD) attack that dynamically balances focus between misclassified and correctly classified pixels. It starts by prioritizing correctly classified pixels, progressively shifting its emphasis to achieve an effective balance as the attack unfolds. On the other hand, SEA integrates multiple complementary loss functions, such as Jensen-Shannon divergence and Masked Cross-Entropy, to exploit various weaknesses in model robustness. Through progressive radius reduction and adaptive optimization, SEA generates potent adversarial perturbations, selecting the worst-case attack outcome to ensure a thorough robustness evaluation.

Table 19. Comparison of our model's performance with and without the attention discriminator.

| Method | Task | Dataset | | | |
|---|---|---|---|---|---|
| | | MVTec AD | VisA | BTAD | BraTS2021 |
| w/o discriminator | AD | 85.9 / 69.5 | 87.7 / 73.0 | 83.8 / 80.6 | 93.4 / 80.6 |
| | AL | 91.1 / 71.5 | 95.4 / 83.9 | 91.7 /72.2 | 96.4 / 93.5 |
| w/ discriminator (*Ours*) | AD | 88.1 / 71.1 | 88.5 / 74.3 | 85.3 / 82.1 | 94.3 / 81.0 |
| | AL | 92.7 / 73.8 | 96.9 / 85.2 | 93.2 /73.0 | 97.7 / 94.5 |

## I. Attention Discriminator

The attention discriminator in our method plays a pivotal role in the anomaly detection and localization pipeline. Conceptually, this component operates similarly to a single layer of a Vision Transformer, where the input embeddings undergo self-attention operations. The resulting embeddings are subsequently passed through a Multi-Layer Perceptron (MLP) to compute a set of anomaly scores for each embedding. Furthermore, the "Attention Degrees," introduced in previous sections, are derived directly from this attention discriminator, reinforcing its critical position in our framework.

To substantiate the necessity and efficacy of the attention discriminator, we performed an ablation study, the results of which are presented in Table 19. In the alternative setup without the discriminator, the attention degrees and MLP components are instead placed on top of the ViT's final layer. This bypass eliminates the intermediate role played by the attention discriminator. However, the comparative results demonstrate that the inclusion of the attention discriminator provides marginally superior performance. This advantage highlights its significance not only in enhancing the model's performance but also in improving its interpretability by providing more precise and structured attention degree calculations.

## J. Evaluating Our Model Under Various Attacks with Diverse Epsilon

To demonstrate our model's robustness, we conducted an experiment in which we trained it under varying $\epsilon$ values of PGD with $l_\infty$ norm and evaluated it using the same $\epsilon$ (ensur-

ing that the training and evaluation $\epsilon$ were identical). The results, as presented in Table 20, indicate that PatchGuard performs effectively across different $\epsilon$ values.

Table 20. Performance of PatchGuard under varying $\epsilon$ values, demonstrating consistent robustness and effectiveness across different settings.

| Epsilon | Task | Dataset | | | |
|---|---|---|---|---|---|
| | | MVTec AD | VisA | BTAD | BraTS2021 |
| $\frac{2}{255}$ | AD | 90.1 / 80.3 | 89.6 / 77.8 | 88.3 / 85.6 | 95.7 / 86.3 |
| | AL | 94.0 / 81.6 | 97.1 / 88.7 | 93.7 / 79.1 | 98.2 / 95.3 |
| $\frac{4}{255}$ | AD | 88.9 / 74.2 | 88.8 / 75.4 | 86.0 / 83.5 | 94.8 / 83.4 |
| | AL | 93.2 / 76.4 | 97.1 / 77.5 | 93.1 / 75.4 | 97.9 / 94.6 |
| $\frac{8}{255}(Ours)$ | AD | 88.1 / 71.1 | 88.5 / 74.3 | 85.3 / 82.1 | 94.3 / 81.0 |
| | AL | 92.7 / 73.8 | 96.9 /85.2 | 93.2 /73.0 | 97.7 /94.5 |

In this section, we evaluate PatchGuard trained on PGD-10 with $l_\infty$ norm under $\epsilon = \frac{8}{255}$ using PGD-1000 with $l_2$ norms with various $\epsilon$. As shown in Table 20, our model remains robust against these types of attacks.

Table 21. Evaluation of PatchGuard's robustness when trained on PGD-10 with $l_\infty$ norm under $\epsilon = \frac{8}{255}$, assessed using PGD-1000 with $l_2$ norms with various $\epsilon$. The results demonstrate the model's sustained robustness against different types of attacks.

| $\epsilon$ | Task | Dataset | | | |
|---|---|---|---|---|---|
| | | MVTec AD | VisA | BTAD | BraTS2021 |
| Clean | AD | 88.1 | 88.5 | 85.3 | 94.3 |
| | AL | 92.7 | 96.9 | 93.2 | 97.7 |
| $\frac{16}{255}$ | AD | 84.3 | 82.7 | 81.7 | 89.2 |
| | AL | 85.7 | 91.7 | 84.3 | 96.7 |
| $\frac{32}{255}$ | AD | 82.1 | 81.0 | 79.5 | 87.4 |
| | AL | 83.6 | 90.3 | 82.9 | 96.1 |
| $\frac{64}{255}$ | AD | 78.2 | 79.8 | 78.5 | 86.4 |
| | AL | 81.0 | 88.7 | 79.6 | 95.7 |
| $\frac{128}{255}$ | AD | 77.2 | 78.6 | 76.7 | 84.7 |
| | AL | 78.3 | 86.7 | 77.9 | 95.0 |

## K. Limitations

Our proposed PatchGuard method includes a "foreground-aware anomaly generation" component that leverages Grad-CAM, which inherently ties our approach to a pretrained model. While this dependency enables our method to focus on relevant regions, it also introduces reliance on the quality and biases of the pretrained model. Furthermore, although we employ soft augmentations to encourage this component to identify accurate regions, there is no theoretical guarantee that it consistently achieves this objective. Nonetheless, as our empirical results demonstrate, the component performs well in practice, effectively highlighting anomalies in diverse scenarios.

## L. Trade-Off Between Anomaly Detection and Localization

The anomaly score and localization map of a method play a crucial role in shaping the design of attacks, enabling attackers to target either anomaly localization or detection with greater precision. In this study, however, we design our attacks on other methods to simultaneously target both localization and detection. In our proposed method, PatchGuard, the anomaly score is derived as the average of the top-k values in the anomaly map. This mechanism ensures that the optimal attack strategy for anomaly detection inherently aligns with the strategy for anomaly localization.

A particularly noteworthy aspect of our study is the approach we use to attack anomaly localization. Specifically, we flip anomaly patches to appear normal and normal patches to appear anomalous. An alternative logical attack could involve manipulating normal images to make all pixels anomalous, while for anomalous samples, the attack would preserve the normal pixels as they are and convert the anomalous pixels to appear normal. This approach would ultimately make the anomaly map of an anomalous sample indistinguishable from that of a normal sample.

Although this alternative attack is specifically designed for anomaly detection, it is far less effective for anomaly localization. Existing methods, even without explicitly addressing this type of targeted attack, are already highly vulnerable to detection-based attacks. In contrast, our method has been experimentally shown to be robust against such attacks. This robustness arises from our use of stronger adversarial training strategies, where all pixels are flipped to create more challenging adversarial examples during the training process.

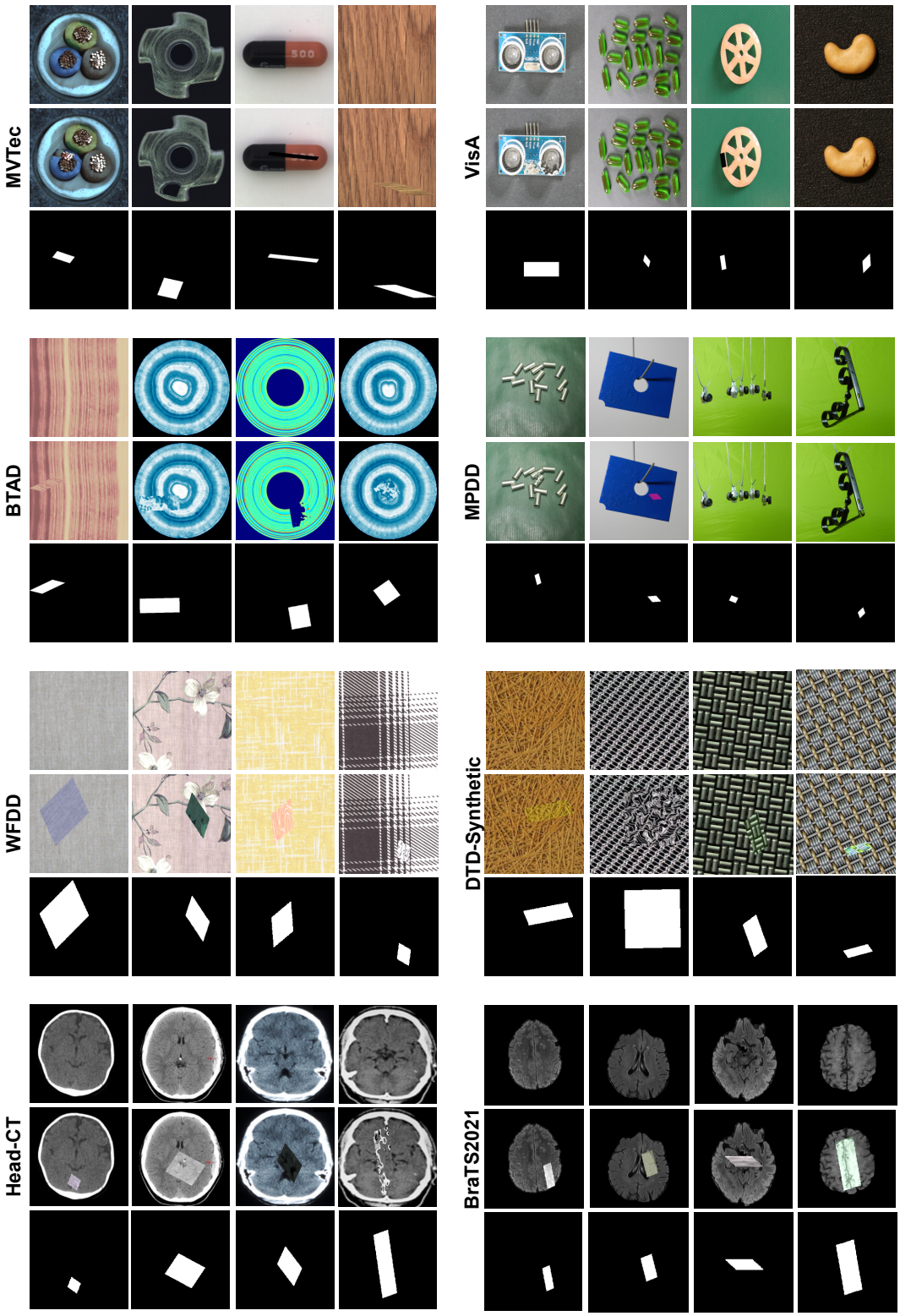


Figure 4. Visualization of Pseudo-Anomaly Generated for Each Dataset. Each group corresponds to one dataset: MVTec AD, VisA, BTAD, MPDD, WFDD, DTD-Synthetic, BraTS2021, and Head-CT. Within each group, columns represent randomly selected samples from the respective dataset. The first row shows a normal image, the second row depicts the corresponding pseudo-anomaly generated image, and the third row illustrates the associated anomaly mask.