Appendix

Scenarios	Defect Types
Electrical Insulator	crack
Metal Welding	weld beading, weld pit
Photovoltaic Module	broken, miss, foreign body
Wind Turbine	crack
Catenary Dropper	looseness, broken, miss
Nut and Bolt	looseness, bolt miss, nut miss
Witness Mark	looseness

Table A.1. Defect types for each scenario in MIAD.

A.1. Defect Types in MIAD

MIAD contains 14 reasonable and common defect types in total, detailed defect types for each scenario are listed in Table A.1.

A.2. Implementation Details on Experiments

For each method, we give a detailed description of the code repository, image input size, data augmentation, neural net architecture and training strategy (*e.g*. learning rate, batch size).

A.2.1. Reverse Distillation

We use the publicly available code in Anomalib [1] as the implementation of Reverse Distillation [3]. Both training and testing images are zoomed to the input size of 256 × 256 pixels. No data augmentation is applied, as this requires prior knowledge about class-retaining augmentations. We adopt WideResNet50 as the backbone in the T-S model. During training, we utilize Adam optimizer [4] with the learning rate of 0.005, β_1 of 0.5 and β_2 of 0.999. All experiments on the MIAD dataset are run with a batch size of 32 on 1 GPU (NVIDIA Tesla V100 32GB). The other parameters not mentioned are consistent with the default configuration in Anomalib.

A.2.2. PatchCore

We use the publicly available code in Anomalib [1] as the implementation of PatchCore [8]. Both training and testing images are zoomed to the input size of 224 × 224 pixels and no data augmentation is applied. We adopt WideResNet50 as the backbone. All experiments on the MIAD dataset are run with a batch size of 32 on 1 GPU (NVIDIA Tesla V100 32GB). Specifically, due to the limitation of GPU memory, PatchCore failed to run when training data size is larger than 2000. Therefore, we randomly sample 2000 images to train the model. The other parameters not mentioned are consistent with the default configuration in Anomalib.

A.2.3. FastFlow

We implement the code of FastFlow following the setting of the original paper [12]. Both training and testing images are resized to the input size of 256 × 256 pixels and no data augmentation is applied. We adopt Wide-ResNet50-2 as the backbone and directly use the embedding of the last layer in the first three blocks, and then put these features into the 2D flow model to obtain their respective anomaly heatmaps, then we average the outputs of each heatmap. All the backbones used in the network are initialized from Imagenet pretrained weights. We use 8-step flows for Wide-ResNet50-2. We train this model using Adam optimizer with the learning rate of 1e-3 and weight decay of 1e-5. We evaluate the final results after 500 epochs training with a batch size of 32 on 1 GPU (NVIDIA 1080Ti 12GB).

A.2.4. DRAEM

For the implementation of DRAEM [13], we use the publicly available code at https://github.com/vitjanz/draem. Both training and testing images are zoomed to the input size of 256 × 256 pixels. Random augmentation sampling is applied during training by a set of 3 random augmentation functions sampled from the set: {posterize, sharpness,solarize, equalize, brightness change, color change, auto-contrast}. Additional image rotation in the range of $(-45, 45)$ degrees is used as a data augmentation method on non-defective images during training. The neural network, which consists of a reconstructive sub-network and a discriminative sub-network following DRAEM [13], is trained for 700 epochs with a batch size of 8 on 1 GPU (NVIDIA Tesla V100 32GB). The learning rate is set to 10^{-4} and is multiplied by 0.2 after 560 and 630 epochs with the Adam [4] optimizer. The l_2 and SSIM loss [10] are applied on the reconstructive sub-network, and the Focal Loss [5] is applied on the discriminative sub-network output to increase robustness towards accurate segmentation of

hard examples.

A.2.5. Auto-Encoder

We implement the code of the Auto-Encoder following the setting of MVTecAD [2]. A modified U-Net [7] is used for the backbone instead of a simple stack of convolution layers. All input RGB image is normalized by a mean of 0.5 and an standard deviation of 0.5. Both training and testing images are zoomed to the input size of 256×256 pixels and no other data augmentation is used. All experiments on the MIAD dataset are run for 100 epochs with a batch size of 16 on 1 GPU (NVIDIA Tesla V100 32GB). All models are trained using the Adam [4] optimizer with a learning rate of 0.0001 and a weight decay of 0.0005.

A.2.6. UniAD

For the implementation of UniAD [11], we use the publicly code at https://github.com/zhiyuanyou/UniAD. A pretrained EfficientNet-b4 [9] model is used for feature extraction, and the feature maps from stage-1 to stage-4 are resized and concatenated together to form a 272-channel feature map. Both training and testing images are zoomed to the input size of 224×224 pixels and no data augmentation is applied. The model is trained for 500 epochs on 8 GPUs (NVIDIA Tesla V100 32GB) with batch size 64. AdamW [6] optimizer with weight decay 0.0001 is used and the learning rate is 0.0001 initially, and dropped by 0.1 after 400 epochs. Other settings are consistent with the original code.

A.3. More Anomaly Localization Visualizations

More qualitative results of each method are shown in Figure A.1. The photovoltaic module scenario is only affected by uncontrolled viewpoints, and is similar to indoor texture anomaly in MVTec AD. Therefore, many methods including Reverse Distillation, FastFlow and DRAEM perform well. The wind turbine is relatively easy for Reverse Distillation, PatchCore, FastFlow and DRAEM. However, the other scenarios are very difficult. For example, methods fail on the electrical insulator, the nut and bolt mainly due to uncontrolled viewpoints. Methods fail on the metal welding mainly result from uncontrolled surface. Methods fail on the catenary droper mainly because of uncontrolled background. And the combination of uncontrolled viewpoints, background and surface result in the failure cases of all methods on the witness mark.

Moreover, DRAEM is more preferable for surface anomaly because it is intended for surface anomaly detection and simulates surface anomalies by augmented images. However logical anomalies can not be simulated well by simple data augmentation, which coincides with the imperfect performance of DRAEM.

References

[1] Samet Akcay, Dick Ameln, Ashwin Vaidya, Barath Lakshmanan, Nilesh Ahuja, and Utku Genc. Anomalib: A deep learning library for anomaly detection, 2022. 1

[2] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. MVTec AD – A comprehensive real-world dataset for unsupervised anomaly detection. In *CVPR*, 2019. 2

[3] Hanqiu Deng and Xingyu Li. Anomaly detection via reverse distillation from one-class embedding. In *CVPR*, 2022. 1

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1, 2

[5] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 1

[6] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 2

[7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 2

[8] Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler. Towards total recall in industrial anomaly detection. In *CVPR*, 2022. 1

[9] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 2

[10] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on image processing*, 2004. 1

[11] Zhiyuan You, Lei Cui, Yujun Shen, Kai Yang, Xin Lu, Yu Zheng, and Xinyi Le. A unified model for multi-class anomaly detection. In *NeurIPS*, 2022. 2

[12] Jiawei Yu, Ye Zheng, Xiang Wang, Wei Li, Yushuang Wu, Rui Zhao, and Liwei Wu. FastFlow: Unsupervised anomaly detection and localization via 2D normalizing flows. *arXiv preprint arXiv:2111.07677*, 2021. 1

[13] Vitjan Zavrtanik, Matej Kristan, and Danijel Skočaj. DRAEM – A discriminatively trained reconstruction embedding for surface anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8330–8339, 2021. 1

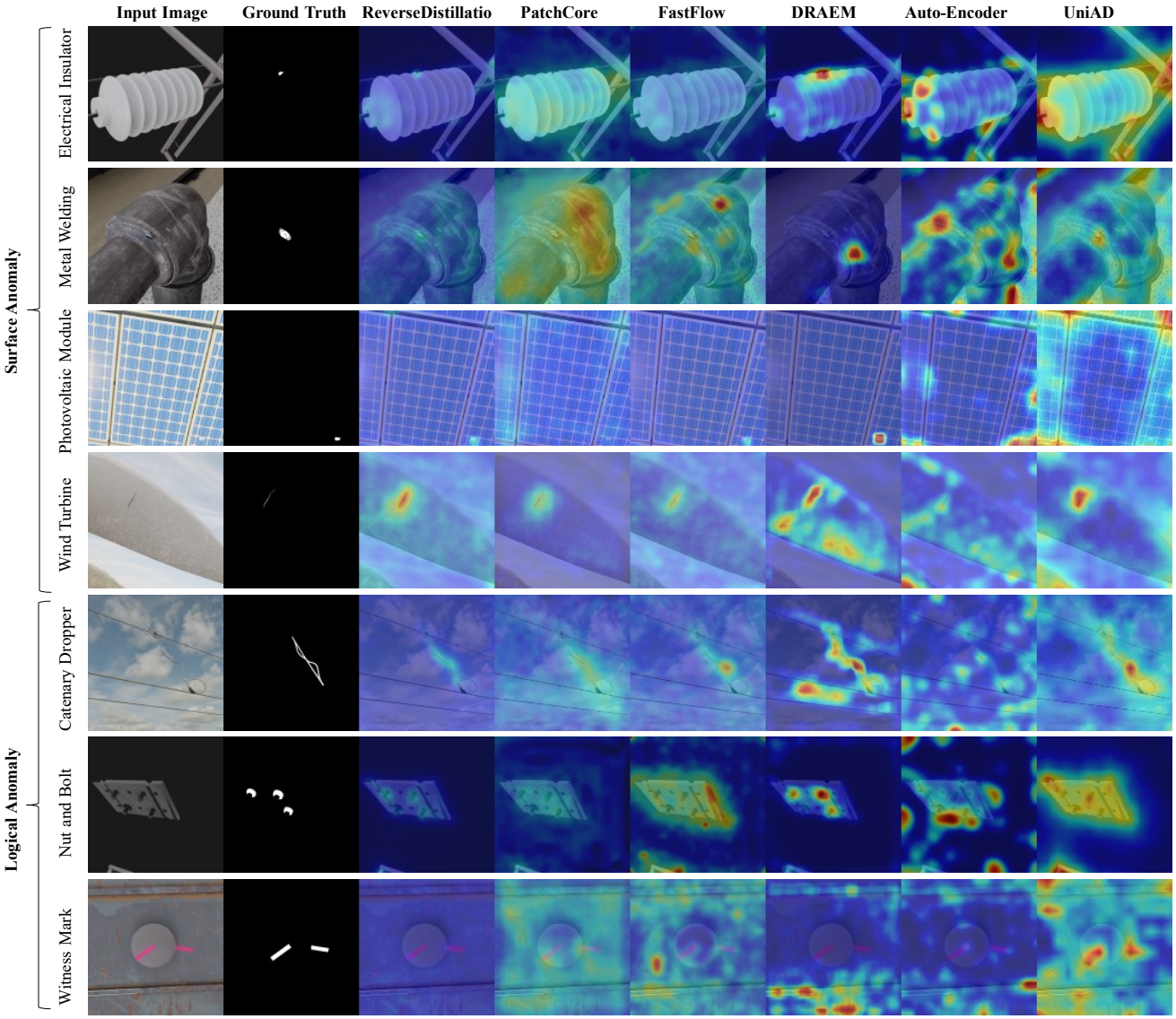

Figure A.1. Qualitative anomaly segmentation results for each evaluated method on the MIAD dataset.