**Optimizing Anomaly Detection on VisA Dataset: Enhanced Noise Handling and Feature Extraction via Conditioned Denoising Diffusion Models**

HENG HUN CHOU

National Yang Ming Chiao Tung University

# **Abstract**

This study introduces a paradigm shift in anomaly detection by adopting the diverse VisA dataset over the traditional MVTec, bringing in a variety of visual anomalies for exploration.

The research focused on enhancing the model's noise handling and refining the scoring of features and textures, thereby boosting detection accuracy. Architectural improvements in the UNET's residual structure and better upsampling and downsampling techniques have improved feature extraction efficiency and precision.

The model's complexity was balanced with generalization capabilities, incorporating L1 and L2 regularizations. Broader evaluation metrics now include the F1 score, latency, and FPS, leading to comprehensive performance assessments.

The improved model showcases competitive performance with an AUROC close to 98%, leading in F1 score, FPS, and latency, and setting new benchmarks for anomaly detection. The study paves the way for further enhancements in the field. Releasing the code at https://github.com/henrychou1233/newddad-on-visa

# **1 Introduction**

This study transitions an anomaly detection model from the MVTec to the VisA dataset, enhancing its scope and performance. Key developments include refining the model's noise handling and recalibrating feature and texture scoring weights, enriching the evaluation process. Architectural improvements in feature extraction and network functions have also been made, boosting efficiency in processing diverse anomalies.

Key contributions are:

* Adapting the VisA dataset for seamless integration with MVTec-specific codes.
* Establishing comprehensive metrics, with this model achieving top rankings in AUROC, F1 scores, FPS and latency.
* Advancing the literature on VisA's evaluation metrics, notably in FPS and latency.
* Exploring UNET-based diffusion processes for superior anomaly detection.

# **2 Related Work**

Self-supervised learning leverages auxiliary tasks to capture image features crucial for anomaly detection (e.g., [13, 31, 33]). Notably, DN2 [2] utilizes ResNets [17] pretrained on ImageNet [38] for feature extraction, while SPADE [6], PaDiM [7], and PatchCore [36] employ memory banks and specialized features for the task. However, these methods often bypass domain-specific feature adaptation. This study approach enhances this by incorporating domain-adapted patch features from PatchCore [36], which improves anomaly detection performance.

Regarding reconstruction-based methods, initial models like VAEs [26] often produced subpar reconstructions of anomalies. Contemporary strategies employ SSIM-based perceptual loss [3] and adversarial autoencoders [34] for better fidelity in anomaly representation but lack anomaly localization. Ganomaly [1] uses conditional GANs [15, 32], advancing beyond prior models. Denoising diffusion models, new to the scene, have shown promise in anomaly detection, particularly in medical applications like brain tumor detection [45], with AnoDDPM [46] outperforming GANs.

This study aims to optimize an anomaly detection model and make significant improvements tailored to the specific requirements of the VisA dataset. Initially, the research adapts the VisA dataset to align with the MVTec format, enhancing compatibility.

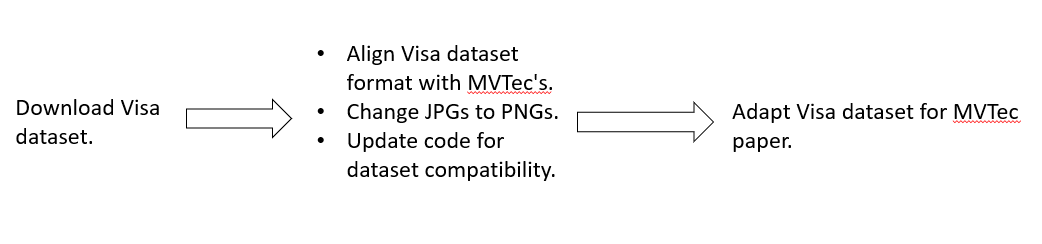


Figure 1. Adapt the Visa dataset for compatibility with the MVTec format: convert JPGs to PNGs and modify the code for dataset alignment, facilitating its integration into MVTec-based research.

# **3 Optimizing**

This study explores the optimization of feature extraction and network architecture while introducing new evaluation metrics such as F1 score, FPS, and latency to comprehensively assess performance.

Future work includes further optimizing the feature extractor, modifying the denoising U-Net, and continuing to improve anomaly detection and scoring methods. The goal of this research is to continually enhance the performance of anomaly detection and achieve further breakthroughs in this field.

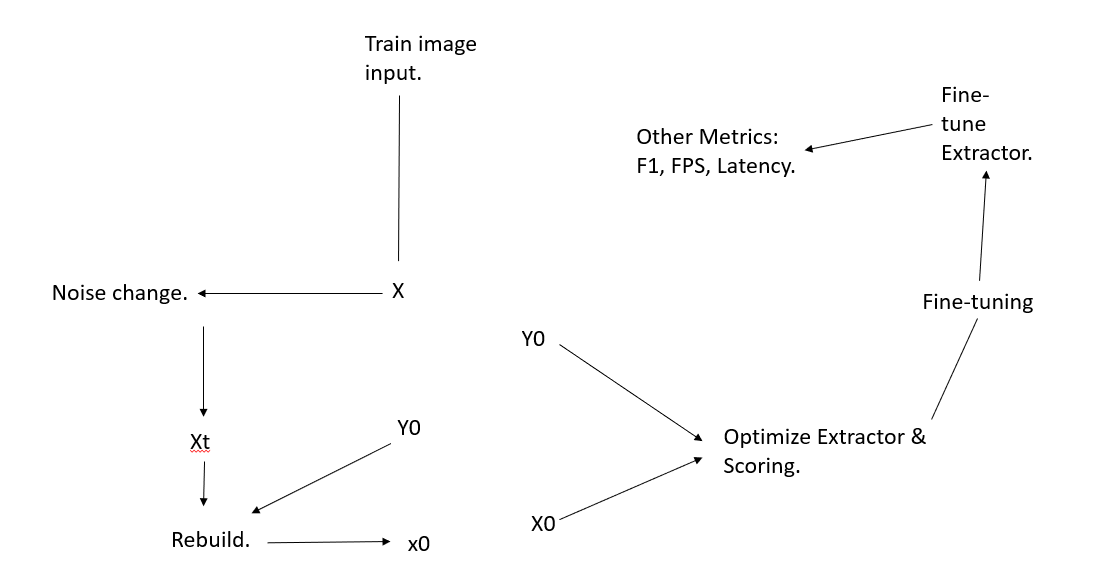


Figure 2. Training images are diffused to simulate anomalies, then reconstructed. A feature extractor, optimized by scoring adjustments and network changes (dropout, n\_heads), is evaluated for F1 Score, FPS, and latency.

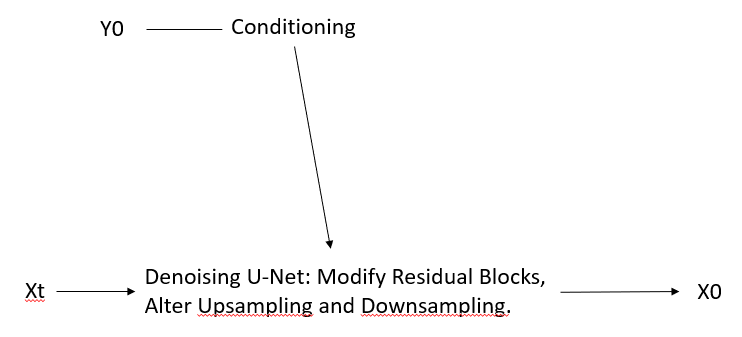


Figure 3. Modify a Denoising U-Net's residual blocks, activation function, and consider transpose convolution for upsampling, max pooling for downsampling.

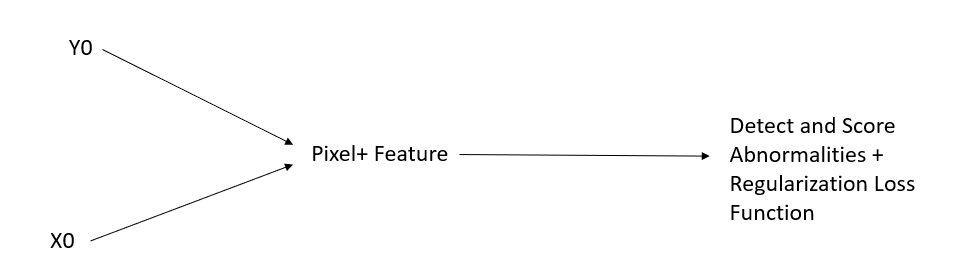
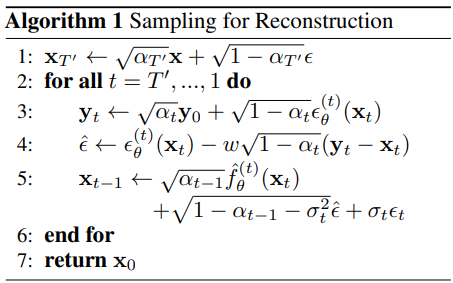
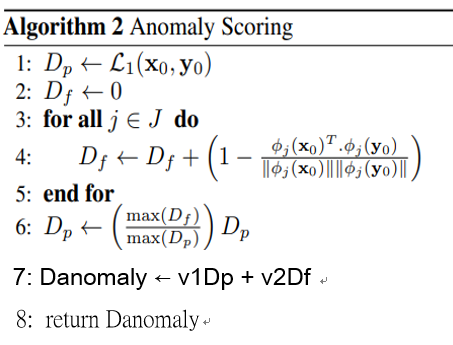


Figure 4. Utilize both pixel and feature-based methods for anomaly detection and scoring, incorporating a regularization loss function.

# **4 Denoising Diffusion Anomaly Detection**





4.1. Conditioned Denoising Process for Reconstruction

This study conditions the denoising of a perturbed imageo resemble the target image (y), addressing signal-to-noise ratio discrepancies. Adding predicted noise to y (creating ) guides towards

in each denoising step. The deviation between and informs an adjusted noise term, essential for creating a less noisy image (-1).

4.2. Reconstruction for Anomaly Detection

For anomaly detection, the input image (x) is used as y, with the model trained on normal data. Anomalies, less probable under the model's distribution, are reconstructed less accurately. The model, DDAD-n, is tailored for precise anomaly detection through optimized denoising iterations.

4.3. Anomaly Scoring

Anomalies are detected by comparing the input with its reconstruction using pixel-wise () and feature-wise () distances. is computed using the L1 norm, and employs cosine similarity across selected network layers. The final anomaly score combines these distances, weighted by two factors (v1 for pixel-wise and v2 for feature-wise importance):

,

balancing the significance of pixel and feature comparisons.

4.4. Domain Adaptation

A domain adaptation technique fine-tunes a pretrained feature extractor for specific anomaly detection tasks. This includes minimizing feature distance between reconstructed and target images, and a distillation loss to maintain network generality.

# **5 Experiments**

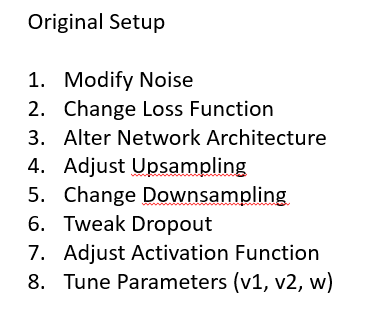
In this study, fine-tuning involves various adjustments to boost the model's performance, such as noise level modification for better input accuracy, loss function changes for precise predictions, and network architecture alterations for improved efficiency. It also focuses on refining details through upsampling adjustments and enhancing pattern recognition via downsampling changes. To prevent overfitting and improve learning, techniques like dropout tweaking and activation function adjustment are utilized. Additionally, tuning parameters (v1, v2, w) are optimized to enhance overall model performance. Initial parameter settings are v1 = 1, v2 = 1, w = 6, and nhead = 4 to optimize performance. Specific experimental data should be placed at the end in the 'Implementation Details' section. 

Figure 5. There are 8 steps in this study

## 

Table 1. origin model

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| pcb series | auroc | pixel | pro | f1 | fps | latency |
| mean | 0.949798286 | 0.924526155 | 0.765917591 | 0.206895485 | 3.23438408 | 66916.26114 |

the results in Table 1 were obtained using the original settings. To save time and resources, the study primarily conducted experiments on the critical PCB series. The insights gained from these experiments were later used to adjust parameters for other categories.

Table 2. change noise

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| pcb series | auroc | pixel | pro | f1 | fps | latency |
| origin (Gaussian) | 0.949798286 | 0.924526155 | 0.765917591 | 0.206895485 | 3.23438408 | 66916.26114 |
| Rayleigh | 0.949927524 | 0.925340086 | 0.76324375 | **0.2069515** | 3.578279596 | 59666.58491 |
| pepper | 0.949448273 | 0.923277497 | 0.738716596 | 0.204807032 | 3.632738903 | 58819.57108 |
| chebyshev | **0.9512015** | **0.9270945** | **0.7662243** | 0.20581123 | 3.158490604 | 67475.89851 |
| Uniform | 0.94818069 | 0.92267856 | 0.749360061 | 0.202002925 | **3.7568364** | **57192.138** |

In short, Table 2 shows that Chebyshev noise outperformed other noise types due to its unique ability to alter data distribution, ultimately improving the model's capacity to detect anomalies in various conditions. Chebyshev noise's distribution exhibits a long-tail property, which implies that it has more extreme values in the tail of the distribution. These extreme values can significantly alter the data distribution, making it easier for the model to capture unusual patterns or anomalies.

Table 3. regularization of the loss function

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Pcb series | auroc | pixel | pro | f1 | fps | latency |
| origin | **0.9512** | **0.9271** | **0.7662** | **0.2058** | 3.1585 | 67475.9 |
| L1 | 0.9487 | 0.9224 | 0.7529 | 0.2036 | **3.5241** | **61262.5** |
| L2 | 0.9503 | 0.9239 | 0.7483 | 0.2002 | 3.4067 | 64699.3 |

Table 4. altering the network backbone of the feature extractor

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Pcb series | auroc | pixel | pro | f1 | fps | latency |
| origin(wide\_resnet101\_2) | **0.9512** | **0.9271** | **0.7662** | 0.2058 | 3.1585 | 67475.8985 |
| wide\_resnet50\_2 | 0.9331 | 0.9067 | 0.7383 | **0.2313** | 3.4507 | **62214.9246** |
| resnet50 | 0.9265 | 0.8972 | 0.7481 | 0.2308 | **3.4687** | 64701.8011 |

Table 5. modifying the upsampling techniques

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Pcb series | auroc | pixel | pro | f1 | fps | latency |
| origin(nearest-neighbor interpolation method) | **0.9512** | **0.9271** | **0.7662** | **0.2058** | 3.1585 | 67475.8985 |
| transpose convolution | 0.9296 | 0.8962 | 0.7372 | 0.1979 | **3.5901** | **60076.3767** |

Table 6. modifying the downsampling techniques

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Pcb series | auroc | pixel | pro | f1 | fps | latency |
| origin(average pooling) | **0.9512** | **0.9271** | **0.7662** | **0.2058** | 3.1585 | 67475.8985 |
| max pooling | 0.9324 | 0.9066 | 0.7211 | 0.2021 | **3.6538** | **58770.9564** |

From table 3-6, Modifications including switching to max pooling, using transpose convolution for upsampling, and applying L1 and L2 losses in the feature extractor's network backbone did not yield optimal results. These changes likely hindered the model's ability to capture and reconstruct intricate details crucial for effective anomaly detection, prompting a return to the original methodology.

Table 7. Change dropout

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Pcb series | auroc | pixel | pro | f1 | fps | latency |
| origin( 0 ) | 0.9512 | **0.9271** | **0.7662** | **0.2058** | 3.1585 | 67475.9 |
| 0.1 | **0.9515** | 0.9250 | 0.7523 | 0.2019 | **3.6026** | **61465.6** |

Based on the results, it was observed that a dropout rate of 0.1 yields the best performance. This setting does not lead to significant loss of information, yet effectively prevents overfitting and enhances accuracy.

Table 8 details a network architecture incorporating the Chebyshev polynomial and the original loss function, alongside the original network backbone. It maintains the original upsampling and downsampling techniques, includes a dropout rate of 0.1, and employs a modified residual activation function.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Pcb series | auroc | pixel | pro | f1 | fps | latency |
| origin(silu) | 0.9515 | 0.9250 | 0.7523 | 0.2019 | 3.6026 | 61465.6258 |
| Mish | **0.9523** | **0.9291** | **0.7639** | **0.2124** | **3.7127** | **58782.1059** |

In comparing different activation functions for the residual blocks, experimental results indicated that the MILU activation function outperforms SiLU. The superiority of MILU could be attributed to its ability to better manage the gradient flow and provide more dynamic range in activation, which is crucial for handling complex patterns in anomaly detection tasks.

Table 9. describes a network configuration that utilizes the Chebyshev polynomial, original loss function, and original network backbone. Additionally, it retains the original upsampling and downsampling methods, uses the Mish activation function with a dropout set at 0.1, and involves adjustments to parameters v1, v2, w, and n\_heads.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| visa | auroc | pixel | pro | f1 | fps | latency |
| pcb1 | 0.9754999876 | 0.8386648893 | 0.5715666429 | 0.1245437327 | 4.593978171 | 43535.25257 |
| pcb2 | 0.9927999973 | 0.9299597144 | 0.7583173855 | 0.2211874914 | 1.659051487 | 120550.8096 |
| pcb3 | 0.9853465557 | 0.9333935976 | 0.7351767782 | 0.1656654098 | 5.019471562 | 40044.05594 |
| pcb4 | 0.9892079234 | 0.9453049898 | 0.7340442309 | 0.3155216032 | 2.267844553 | 88630.41329 |
| pipe\_fryum | 0.9153999686 | 0.9502823353 | 0.7146063581 | 0.3206277692 | 7.257187922 | 20669.16299 |
| candle | 0.9901000261 | 0.9756087065 | 0.933150949 | 0.3976588377 | 7.723328585 | 25895.57052 |
| capsules | 0.9898333549 | 0.9780046344 | 0.8915693296 | 0.5987913655 | 3.514089346 | 45530.99942 |
| cashew | 0.9747999907 | 0.9313732386 | 0.6254780248 | 0.5235213167 | 0.5235213167 | 20234.16686 |
| chewinggum | 0.9754000902 | 0.7638657689 | 0.4060399769 | 0.07809419781 | 4.093335814 | 36644.92893 |
| fryum | 1 | 0.9066184163 | 0.8328368344 | 0.4490907669 | 7.01730247 | 21375.73528 |
| macaroni1 | 0.9775999784 | 0.9685521722 | 0.9296662499 | 0.2296142763 | 9.267431894 | 21580.95169 |
| macaroni2 | 0.9872000813 | 0.9765995741 | 0.9776408202 | 0.3562729779 | 9.326440436 | 21444.40866 |
| mean | 0.9794323295 | 0.9248523364 | 0.759174465 | 0.3150491454 | 5.188581963 | 42178.03798 |
| rank | 5 | 4 | 14 | 1 | 1 | 1 |
| total | 29 | 29 | 29 | 29 | 29 | 29 |

This study applies PCB insights to other areas, optimizing parameters like v1, v2, w, and n\_heads. Enhancing n-heads in attention mechanisms aids in anomaly detection, but too high values risk overfitting and complexity. Feature prominence varies across categories; it's crucial for detailed PCB images but less so for objects with curved shapes like capsules and cashews.

# Experimental Results

Table 10. This table presents the impact of each step on six major metrics for the PCB series, including AUROC, Pixel, PRO, F1, FPS, and Latency.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| pcb series | auroc | pixel | pro | f1 | fps | latency |
| origin | 0.9497982859 | 0.924526155 | 0.7659175911 | 0.206895485 | 3.23438408 | 66916.26114 |
| noise | 0.9512014687 | 0.9270945191 | 0.7662243236 | 0.2058112296 | 3.158490604 | 67475.89851 |
| loss upsampling downsampling backbone | decrease |  |  |  |  |  |
| noise+dropout | 0.9514511973 | 0.9249589294 | 0.752309907 | 0.201938393 | 3.602601894 | 61465.62577 |
| noise+dropout+activation function | 0.9522868544 | 0.9291268736 | 0.7639495605 | 0.212427783 | 3.712699309 | 58782.10592 |
| noise+dropout+activation function+adjusting parameters\*4  (v1 v2 w nhead) | **0.985713616** | 0.9118307978 | 0.6997762594 | 0.2067295593 | 3.385086443 | 73190.13286 |

In summary:

* Chebyshev noise was the most effective for highlighting subtle image anomalies by altering data distribution.
* Original methods for the network backbone, upsampling, and downsampling performed better, likely due to their alignment with dataset nuances and effective preservation of vital image details.
* The Mish activation function outperformed SiLU, offering better gradient handling for complex pattern learning.
* Generally, a higher n\_head in the model architecture led to better performance in capturing various features and patterns within images. However, the optimal number of heads may vary based on dataset characteristics and computational resources.
* Model weighting adjustments for features and pixels were customized to dataset characteristics, optimizing anomaly detection performance for different dataset categories

# **6. Conclusion**

## 6.1 Study Limitations

One limitation is the need to tune parameters for each category, making it impractical for real-world applications. Ideally, demonstrating robustness would involve using the same parameters for all categories. Many top-ranking papers also employ different parameters for each category.

Theoretical steps involve exploring various options, but time constraints allow only the best option to proceed.

Currently, improvements focus on minor model adjustments, limiting significant enhancement potential. To achieve further advancement, substantial revisions or starting a new in certain areas may be necessary.

## 6.2 Future Works

In summary, the study could benefit from testing different methods to find the optimal strategy. Expanding the testing scope to include other 2D datasets would enhance its applicability. While the current focus is on improving AUROC and PRO metrics, it is essential to prioritize increasing F1 scores, reducing latency, and improving FPS metrics in the future, rather than merely calculating them. Finally, considering the application of 2D methods in 3D contexts may be worth exploring.

**7. References**

[1] Samet Akcay, Amir Atapour-Abarghouei, and Toby P Breckon. "GANomaly: Semi-supervised anomaly detection via adversarial training." In Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III, pages 622–637. Springer, 2019.

[2] Liron Bergman, Niv Cohen, and Yedid Hoshen. "Deep nearest neighbor anomaly detection." arXiv preprint arXiv:2002.10445, 2020.

[3] Paul Bergmann, Sindy Lowe, Michael Fauser, David Sattlegger, and Carsten Steger. "Improving unsupervised defect segmentation by applying structural similarity to autoencoders." arXiv preprint arXiv:1807.02011, 2018.

[4] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. "Mvtec ad–a comprehensive real-world dataset for unsupervised anomaly detection." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 9592–9600, 2019.

[5] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. "Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 4183–4192, 2020.

[6] Niv Cohen and Yedid Hoshen. "Sub-image anomaly detection with deep pyramid correspondences." arXiv preprint arXiv:2005.02357, 2020.

[7] Thomas Defard, Aleksandr Setkov, Angelique Loesch, and Romaric Audigier. "Padim: a patch distribution modeling framework for anomaly detection and localization." In Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part IV, pages 475–489. Springer, 2021.

[8] David Dehaene and Pierre Eline. "Anomaly localization by modeling perceptual features." arXiv preprint arXiv:2008.05369, 2020.

[9] Hanqiu Deng and Xingyu Li. "Anomaly detection via reverse distillation from one-class embedding." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9737–9746, 2022.

[10] Prafulla Dhariwal and Alexander Nichol. "Diffusion models beat GANs on image synthesis." Advances in Neural Information Processing Systems, 34:8780–8794, 2021.

[11] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using real NVP." In International Conference on Learning Representations, 2017.

[12] Alexey Dosovitskiy and Thomas Brox. "Generating images with perceptual similarity metrics based on deep networks." Advances in neural information processing systems, 29, 2016.

[13] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. "Unsupervised representation learning by predicting image rotations." arXiv preprint arXiv:1803.07728, 2018.

[14] Izhak Golan and Ran El-Yaniv. "Deep anomaly detection using geometric transformations." In Advances in Neural Information Processing Systems. Curran Associates, Inc., 2018.

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial networks." Communications of the ACM, 63(11):139–144, 2020.

[16] Denis Gudovskiy, Shun Ishizaka, and Kazuki Kozuka. "Cflow-ad: Real-time unsupervised anomaly detection with localization via conditional normalizing flows." In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pages 98–107, 2022.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[18] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. "Using self-supervised learning can improve model robustness and uncertainty." Advances in neural information processing systems, 32, 2019.

[19] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network." arXiv preprint arXiv:1503.02531, 2015.

[20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." In Advances in Neural Information Processing Systems, pages 6840–6851. Curran Associates, Inc., 2020.

[21] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." Advances in Neural Information Processing Systems, 33:6840–6851, 2020.

[22] Yibin Huang, Congying Qiu, and Kui Yuan. "Surface defect saliency of magnetic tile." The Visual Computer, 36:85–96, 2020.

[23] Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn Ball, Katie Shpanskaya, et al. "Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison." In Proceedings of the AAAI conference on artificial intelligence, pages 590–597, 2019.

[24] Jongheon Jeong, Yang Zou, Taewan Kim, Dongqing Zhang, Avinash Ravichandran, and Onkar Dabeer. "Winclip: Zero-/few-shot anomaly classification and segmentation." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 19606–19616, 2023.

[25] Durk P Kingma and Prafulla Dhariwal. "Glow: Generative flow with invertible 1x1 convolutions." Advances in neural information processing systems, 31, 2018.

[26] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes." arXiv preprint arXiv:1312.6114, 2013.

[27] Wen Liu, Weixin Luo, Dongze Lian, and Shenghua Gao. "Future frame prediction for anomaly detection–a new baseline." In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 6536–6545, 2018.

[28] Zhikang Liu, Yiming Zhou, Yuansheng Xu, and Zilei Wang. "SimpleNet: A simple network for image anomaly detection and localization." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 20402–20411, 2023.

[29] Fanbin Lu, Xufeng Yao, Chi-Wing Fu, and Jiaya Jia. "Removing anomalies as noises for industrial defect localization." In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 16166–16175, 2023.

[30] Yuchen Lu and Peng Xu. "Anomaly detection for skin disease images using variational autoencoder." arXiv preprint arXiv:1807.01349, 2018.

[31] Michael Mathieu, Camille Couprie, and Yann LeCun. "Deep multi-scale video prediction beyond mean square error." arXiv preprint arXiv:1511.05440, 2015.

[32] Mehdi Mirza and Simon Osindero. "Conditional generative adversarial nets." arXiv preprint arXiv:1411.1784, 2014.

[33] Mehdi Noroozi and Paolo Favaro. "Unsupervised learning of visual representations by solving jigsaw puzzles." In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI, pages 69–84. Springer, 2016.

[34] Stanislav Pidhorskyi, Ranya Almohsen, and Gianfranco Doretto. "Generative probabilistic novelty detection with adversarial autoencoders." Advances in neural information processing systems, 31, 2018.

[35] Nicolae-Cat˘ alin Ristea, Neelu Madan, Radu Tudor Ionescu, ˘ Kamal Nasrollahi, Fahad Shahbaz Khan, Thomas B Moeslund, and Mubarak Shah. "Self-supervised predictive convolutional attentive block for anomaly detection." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 13576–13586, 2022.

[36] Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Scholkopf, Thomas Brox, and Peter Gehler. "Towards total recall in industrial anomaly detection." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 14318–14328, 2022.

[37] Marco Rudolph, Bastian Wandt, and Bodo Rosenhahn. "Same same but different: Semi-supervised defect detection with normalizing flows." In Proceedings of the IEEE/CVF winter conference on applications of computer vision, pages 1907–1916, 2021.

[38] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. "ImageNet large scale visual recognition challenge." International journal of computer vision, 115:211–252, 2015.

[39] Mohammad Sabokrou, Mohammad Khalooei, Mahmood Fathy, and Ehsan Adeli. "Adversarially learned one-class classifier for novelty detection." In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3379–3388, 2018.

[40] Woosang Shin, Jonghyeon Lee, Taehan Lee, Sangmoon Lee, and Jong Pil Yun. "Anomaly detection using score-based perturbation resilience." In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 23372–23382, 2023.

[41] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. "Deep unsupervised learning using nonequilibrium thermodynamics." In International Conference on Machine Learning, pages 2256–2265. PMLR, 2015.

[42] Jiaming Song, Chenlin Meng, and Stefano Ermon. "Denoising diffusion implicit models." In International Conference on Learning Representations, 2021.

[43] Yang Song and Stefano Ermon. "Generative modeling by estimating gradients of the data distribution." Advances in neural information processing systems, 32, 2019.

[44] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. "Score-based generative modeling through stochastic differential equations." In International Conference on Learning Representations, 2021.

[45] Julia Wolleb, Florentin Bieder, Robin Sandkuhler, and Philippe C Cattin. "Diffusion models for medical anomaly detection." In Medical Image Computing and Computer Assisted Intervention–MICCAI 2022: 25th International Conference, Singapore, September 18–22, 2022, Proceedings, Part VIII, pages 35–45. Springer, 2022.

[46] Julian Wyatt, Adam Leach, Sebastian M Schmon, and Chris G Willcocks. "Anoddpm: Anomaly detection with denoising diffusion probabilistic models using simplex noise." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 650–656, 2022.

[47] 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.

[48] Sergey Zagoruyko and Nikos Komodakis. "Wide residual networks." arXiv preprint arXiv:1605.07146, 2016.

[49] Vitjan Zavrtanik, Matej Kristan, and Danijel Skocaj. "Reconstruction by inpainting for visual anomaly detection." Pattern Recognition, 112:107706, 2021.

[50] Vitjan Zavrtanik, Matej Kristan, and Danijel Skocaj. "Draema discriminatively trained reconstruction embedding for surface anomaly detection." In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 8330–8339, 2021.

[51] Hui Zhang, Zheng Wang, Zuxuan Wu, and Yu-Gang Jiang. "Diffusionad: Denoising diffusion for anomaly detection." arXiv preprint arXiv:2303.08730, 2023.

[52] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. "The unreasonable effectiveness of deep features as a perceptual metric." In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 586–595, 2018.

[53] Ying Zhao. "Omnial: A unified CNN framework for unsupervised anomaly localization." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 3924–3933, 2023.

[54] David Zimmerer, Jens Petersen, Gregor Kohler, Paul Jäger, Peter Full, Klaus Maier-Hein, Tobias Roß, Tim Adler, Annika Reinke, and Lena Maier-Hein. "Medical out-of-distribution analysis challenge 2022, 2022."

[55] Yang Zou, Jongheon Jeong, Latha Pemula, Dongqing Zhang, and Onkar Dabeer. "Spot-the-difference self-supervised pretraining for anomaly detection and segmentation." In Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXX, pages 392–408. Springer, 2022.

[56] Mousakhan, A., Brox, T., & Tayyub, J. (2023). Anomaly Detection with Conditioned Denoising Diffusion Models. ArXiv, abs/2305.15956.

**8. Implementation Details**

Table 11.origin model

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | auroc | pixel | pro | f1 | fps | latency |
| pcb1 | 0.8921000361 | 0.8535388708 | 0.7464483204 | 0.1282527881 | 3.842100809 | 52054.85487 |
| pcb2 | 0.9564000368 | 0.9596131444 | 0.8683160105 | 0.233044595 | 3.000815728 | 66648.54431 |
| pcb3 | 0.9638613462 | 0.9486213923 | 0.7807121802 | 0.1897548173 | 4.078027841 | 49288.53059 |
| pcb4 | 0.9868317246 | 0.9363312125 | 0.6681938533 | 0.2765297396 | 2.016591941 | 99673.11478 |

Table12. change noise

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| pcb1 | auroc | pixel | pro | f1 | fps | latency |
| origin(Gaussian) | 0.8921000361 | 0.8535388708 | 0.7464483204 | 0.1282527881 | 3.842100809 | 52054.85487 |
| Rayleigh | 0.8953000307 | 0.8543448448 | 0.7416151852 | 0.1278376485 | 4.01241593 | 49845.28112 |
| pepper | 0.8920000196 | 0.8521294594 | 0.6861836117 | 0.1321316805 | 4.124218333 | 48494.03787 |
| chebyshev | **0.8988999128** | **0.860730648** | **0.743587683** | **0.1286459339** | **4.155907735** | **48124.26376** |
| Uniform | 0.8885000348 | 0.8505647182 | 0.7315568527 | 0.1241350109 | 4.532930964 | 44121.5632 |
|  |  |  |  |  |  |  |
| pcb2 | auroc | pixel | pro | f1 | fps | latency |
| origin(Gaussian) | 0.9564000368 | 0.9596131444 | 0.8683160105 | 0.233044595 | 3.000815728 | 66648.54431 |
| Rayleigh | 0.9554001093 | 0.9598103762 | 0.8711151955 | 0.2318240164 | 3.343294977 | 59821.22469 |
| pepper | 0.9551000595 | 0.9598298073 | 0.8674227745 | 0.2303237577 | 3.324497053 | 60159.4758 |
| chebyshev | **0.9564999938** | **0.9604209065** | **0.8746431025** | **0.2346019114** | **2.384485081** | **83875.55099** |
| Uniform | 0.9565000534 | 0.9600646496 | 0.8705039388 | 0.233174932 | 3.278321679 | 61006.82592 |
|  |  |  |  |  |  |  |
| pcb3 | auroc | pixel | pro | f1 | fps | latency |
| origin(Gaussian) | 0.9638613462 | 0.9486213923 | 0.7807121802 | 0.1897548173 | 4.078027841 | 49288.53059 |
| Rayleigh | 0.9622772336 | 0.9468628168 | 0.759876497 | 0.1874187465 | 4.601536802 | 43681.05888 |
| pepper | 0.9641583562 | 0.9473272562 | 0.7592624242 | 0.1895675854 | 4.680161924 | 42947.23201 |
| chebyshev | **0.9622772336** | **0.9476521015** | **0.775882073** | **0.186149801** | **3.680258012** | **54615.73601** |
| Uniform | 0.9623761773 | 0.9470049739 | 0.7679349466 | 0.1891558982 | 4.742463722 | 42383.03375 |
|  |  |  |  |  |  |  |
| pcb4 | auroc | pixel | pro | f1 | fps | latency |
| origin(Gaussian) | 0.9868317246 | 0.9363312125 | 0.6681938533 | 0.2765297396 | 2.016591941 | 99673.11478 |
| Rayleigh | 0.9867327213 | 0.9403423071 | 0.680368123 | 0.2807257337 | 2.355870676 | 85318.77494 |
| pepper | 0.9865346551 | 0.9338234663 | 0.6419975718 | 0.2672051054 | 2.402078303 | 83677.53863 |
| chebyshev | **0.9871287346** | **0.9395744205** | **0.670784436** | **0.2738472721** | **2.413311589** | **83288.04326** |
| Uniform | 0.9853464961 | 0.9330798984 | 0.6274445072 | 0.2615458588 | 2.473629101 | 81257.12943 |

Table 13. regularization of the loss function

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| pcb1 | auroc | pixel | pro | f1 | fps | latency |
| origin | **0.8988999128** | **0.860730648** | **0.743587683** | **0.1286459339** | **4.155907735** | **48124.26376** |
| L1 | 0.8928000331 | 0.8540740609 | 0.7494523284 | 0.1274277974 | 4.105482502 | 48715.34586 |
| L2 | 0.8962999582 | 0.8562864065 | 0.7224112559 | 0.1189278753 | 4.413983361 | 45310.54688 |
|  |  |  |  |  |  |  |
| pcb2 | auroc | pixel | pro | f1 | fps | latency |
| origin | **0.9564999938** | **0.9604209065** | **0.8746431025** | **0.2346019114** | **2.384485081** | **83875.55099** |
| L1 | 0.9552000165 | 0.9606360793 | 0.8733155811 | 0.231679881 | 3.282318873 | 60932.53207 |
| L2 | 0.9572000504 | 0.958835125 | 0.8684779716 | 0.2288890955 | 2.68789149 | 74407.76563 |
|  |  |  |  |  |  |  |
| pcb3 | auroc | pixel | pro | f1 | fps | latency |
| origin | **0.9622772336** | **0.9476521015** | **0.775882073** | **0.186149801** | **3.680258012** | **54615.73601** |
| L1 | 0.9627722502 | 0.9483230114 | 0.7755221001 | 0.1887751146 | 4.491282958 | 44753.35932 |
| L2 | 0.9611880779 | 0.946526587 | 0.7596489056 | 0.1858984405 | 4.364050225 | 46058.13169 |
|  |  |  |  |  |  |  |
| pcb4 | auroc | pixel | pro | f1 | fps | latency |
| origin | **0.9871287346** | **0.9395744205** | **0.670784436** | **0.2738472721** | **2.413311589** | **83288.04326** |
| L1 | 0.9841584563 | 0.9266471267 | 0.6132802309 | 0.26647381 | 2.217346639 | 90648.88477 |
| L2 | 0.9864356518 | 0.9338340759 | 0.6425866829 | 0.2671572326 | 2.160804966 | 93020.88952 |

Table 14. altering the network backbone of the feature extractor

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| pcb1 | auroc | pixel | pro | f1 | fps | latency |
| origin(wide\_resnet101\_2) | **0.8988999128** | **0.860730648** | **0.743587683** | **0.1286459339** | **4.155907735** | **48124.26376** |
| wide\_resnet50\_2 | 0.8940000534 | 0.8039560318 | 0.6830109775 | 0.09051869466 | 4.513110534 | 44315.33384 |
| resnet50 | 0.8522000313 | 0.7476089001 | 0.7203276535 | 0.1120264073 | 4.456011859 | 44883.18396 |
|  |  |  |  |  |  |  |
| pcb2 | auroc | pixel | pro | f1 | fps | latency |
| origin(wide\_resnet101\_2) | **0.9564999938** | **0.9604209065** | **0.8746431025** | **0.2346019114** | **2.384485081** | **83875.55099** |
| wide\_resnet50\_2 | 0.9714999795 | 0.9466593266 | 0.8116691255 | 0.2745419097 | 2.86554023 | 69794.86728 |
| resnet50 | 0.9180999994 | 0.9556014538 | 0.8342105239 | 0.2310073035 | 2.538432703 | 78788.77378 |
|  |  |  |  |  |  |  |
| pcb3 | auroc | pixel | pro | f1 | fps | latency |
| origin(wide\_resnet101\_2) | **0.9622772336** | **0.9476521015** | **0.775882073** | **0.186149801** | **3.680258012** | **54615.73601** |
| wide\_resnet50\_2 | 0.9649505019 | 0.925399065 | 0.6900661155 | 0.1584297441 | 4.06934243 | 49393.72969 |
| resnet50 | 0.9448514581 | 0.9242632389 | 0.6468091698 | 0.1228344109 | 4.705721877 | 42713.95659 |
|  |  |  |  |  |  |  |
| pcb4 | auroc | pixel | pro | f1 | fps | latency |
| origin(wide\_resnet101\_2) | **0.9871287346** | **0.9395744205** | **0.670784436** | **0.2738472721** | **2.413311589** | **83288.04326** |
| wide\_resnet50\_2 | 0.9820792079 | 0.9506173134 | 0.7685502565 | 0.4018069273 | 2.354849653 | 85355.76773 |
| resnet50 | 0.9906930327 | 0.961427331 | 0.7911025181 | 0.4575126958 | 2.174823573 | 92421.28992 |

Table 15. modifying the upsampling techniques

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| pcb1 | auroc | pixel | pro | f1 | fps | latency |
| origin(nearest-neighbor interpolation method) | **0.8988999128** | **0.860730648** | **0.743587683** | **0.1286459339** | **4.155907735** | **48124.26376** |
| transpose convolution | 0.8517999649 | 0.7477995157 | 0.7210985128 | 0.1121093182 | 4.208614594 | 47521.5764 |
|  |  |  |  |  |  |  |
| pcb2 | auroc | pixel | pro | f1 | fps | latency |
| origin(nearest-neighbor interpolation method) | **0.9564999938** | **0.9604209065** | **0.8746431025** | **0.2346019114** | **2.384485081** | **83875.55099** |
| transpose convolution | 0.9179000258 | 0.9555775523 | 0.8338686969 | 0.2314618991 | 3.057081899 | 65421.86522 |
|  |  |  |  |  |  |  |
| pcb3 | auroc | pixel | pro | f1 | fps | latency |
| origin(nearest-neighbor interpolation method) | **0.9622772336** | **0.9476521015** | **0.775882073** | **0.186149801** | **3.680258012** | **54615.73601** |
| transpose convolution | 0.9629703164 | 0.9471417665 | 0.7617580839 | 0.1863271324 | 4.725147287 | 42538.35654 |
|  |  |  |  |  |  |  |
| pcb4 | auroc | pixel | pro | f1 | fps | latency |
| origin(nearest-neighbor interpolation method) | **0.9871287346** | **0.9395744205** | **0.670784436** | **0.2738472721** | **2.413311589** | **83288.04326** |
| transpose convolution | 0.985742569 | 0.9343197346 | 0.6320500859 | 0.2615568321 | 2.36962051 | 84823.70877 |

Table 16. modifying the downsampling techniques

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| pcb1 | auroc | pixel | pro | f1 | fps | latency |
| origin(average pooling) | **0.8988999128** | **0.860730648** | **0.743587683** | **0.1286459339** | **4.155907735** | **48124.26376** |
| max pooling | 0.8713000417 | 0.8387266397 | 0.7470378136 | 0.1729450258 | 4.244469062 | 47120.14556 |
|  |  |  |  |  |  |  |
| pcb2 | auroc | pixel | pro | f1 | fps | latency |
| origin(average pooling) | **0.9564999938** | **0.9604209065** | **0.8746431025** | **0.2346019114** | **2.384485081** | **83875.55099** |
| max pooling | 0.9653999805 | 0.9612047076 | 0.8676225709 | 0.2204060953 | 3.154423215 | 63403.03326 |
|  |  |  |  |  |  |  |
| pcb3 | auroc | pixel | pro | f1 | fps | latency |
| origin(average pooling) | **0.9622772336** | **0.9476521015** | **0.775882073** | **0.186149801** | **3.680258012** | **54615.73601** |
| max pooling | 0.910990119 | 0.9267212749 | 0.7189289298 | 0.1330789166 | 4.780443013 | 42046.31233 |
|  |  |  |  |  |  |  |
| pcb4 | auroc | pixel | pro | f1 | fps | latency |
| origin(average pooling) | **0.9871287346** | **0.9395744205** | **0.670784436** | **0.2738472721** | **2.413311589** | **83288.04326** |
| max pooling | 0.9820792079 | 0.8997257352 | 0.5506935513 | 0.2818034282 | 2.43594039 | 82514.33444 |

Table 17. Based on the results, it was observed that a dropout rate of 0.1 yields the best performance. This setting does not lead to significant loss of information, yet effectively prevents overfitting and enhances accuracy.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| pcb1 | auroc | pixel | pro | f1 | fps | latency |
| origin( 0 ) | 0.8988999128 | 0.860730648 | 0.743587683 | 0.1286459339 | 4.155907735 | 48124.26376 |
| 0.1 | **0.898999989** | **0.8562390208** | **0.7270107251** | **0.1206160786** | **4.47840506** | **44658.75626** |
|  |  |  |  |  |  |  |
| pcb2 | auroc | pixel | pro | f1 | fps | latency |
| origin( 0 ) | 0.9564999938 | 0.9604209065 | 0.8746431025 | 0.2346019114 | 2.384485081 | 83875.55099 |
| 0.1 | **0.9572999477** | **0.960673213** | **0.8771057154** | **0.2328795182** | **3.225650656** | **62002.99454** |
|  |  |  |  |  |  |  |
| pcb3 | auroc | pixel | pro | f1 | fps | latency |
| origin( 0 ) | 0.9622772336 | 0.9476521015 | 0.775882073 | 0.186149801 | 3.680258012 | 54615.73601 |
| 0.1 | **0.9633662701** | **0.9474352598** | **0.7628547987** | **0.1892269804** | **4.602215686** | **43674.61538** |
|  |  |  |  |  |  |  |
| pcb4 | auroc | pixel | pro | f1 | fps | latency |
| origin( 0 ) | 0.9871287346 | 0.9395744205 | 0.670784436 | 0.2738472721 | 2.413311589 | 83288.04326 |
| 0.1 | **0.9861385822** | **0.935488224** | **0.6422683888** | **0.2650309947** | **2.104136172** | **95526.13688** |

Table18. In comparing different activation functions for the residual blocks, experimental results indicated that the MILU activation function outperforms SiLU. The superiority of MILU could be attributed to its ability to better manage the gradient flow and provide more dynamic range in activation, which is crucial for handling complex patterns in anomaly detection tasks.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| pcb1 | auroc | pixel | pro | f1 | fps | latency |
| origin(silu) | 0.898999989 | 0.8562390208 | 0.7270107251 | 0.1206160786 | 4.47840506 | 44658.75626 |
| Mish | **0.9087999463** | **0.863191545** | **0.691985849** | **0.1286731701** | **4.436703847** | **45078.51028** |
|  |  |  |  |  |  |  |
| pcb2 | auroc | pixel | pro | f1 | fps | latency |
| origin(silu) | 0.9572999477 | 0.960673213 | 0.8771057154 | 0.2328795182 | 3.225650656 | 62002.99454 |
| Mish | **0.9451000094** | **0.9564200044** | **0.8396495676** | **0.2174383397** | **3.371241702** | **59325.32215** |
|  |  |  |  |  |  |  |
| pcb3 | auroc | pixel | pro | f1 | fps | latency |
| origin(silu) | 0.9633662701 | 0.9474352598 | 0.7628547987 | 0.1892269804 | 4.602215686 | 43674.61538 |
| Mish | **0.9660395384** | **0.9515909553** | **0.7901185943** | **0.188078019** | **4.775007132** | **42094.17796** |
|  |  |  |  |  |  |  |
| pcb4 | auroc | pixel | pro | f1 | fps | latency |
| origin(silu) | 0.9861385822 | 0.935488224 | 0.6422683888 | 0.2650309947 | 2.104136172 | 95526.13688 |
| Mish | **0.9892079234** | **0.9453049898** | **0.7340442309** | **0.3155216032** | **2.267844553** | **88630.41329** |

Table 19. In this study, the findings from the PCB category are adapted to other domains, focusing on identifying the optimal parameters v1, v2, w and n\_heads. Increasing nhead weight improves anomaly detection by better focusing on key image areas through multi-head attention. However, excessively high nhead can cause overfitting and computational complexity, so balancing detail capture and model efficiency is crucial.In summary, the Visa dataset's features are generally easier to discern compared to pixels, leading to a higher weighting for features. However, the proportion varies across different categories. For instance, in the PCB series, there is a significant emphasis on pixel precision due to the detailed nature of the images. Conversely, for categories with curved shapes like capsules and cashews, pixel precision can be almost negligible.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | auroc | pixel | pro | f1 | fps | latency |
| pcb1 | 0.9754999876 | 0.8386648893 | 0.5715666429 | 0.1245437327 | 4.593978171 | 43535.25257 |
| pcb2 | 0.9927999973 | 0.9299597144 | 0.7583173855 | 0.2211874914 | 1.659051487 | 120550.8096 |
| pcb3 | 0.9853465557 | 0.9333935976 | 0.7351767782 | 0.1656654098 | 5.019471562 | 40044.05594 |
| pcb4 | 0.9892079234 | 0.9453049898 | 0.7340442309 | 0.3155216032 | 2.267844553 | 88630.41329 |
| pipe\_fryum | 0.9153999686 | 0.9502823353 | 0.7146063581 | 0.3206277692 | 7.257187922 | 20669.16299 |
| candle | 0.9901000261 | 0.9756087065 | 0.933150949 | 0.3976588377 | 7.723328585 | 25895.57052 |
| capsules | 0.9898333549 | 0.9780046344 | 0.8915693296 | 0.5987913655 | 3.514089346 | 45530.99942 |
| cashew | 0.9747999907 | 0.9313732386 | 0.6254780248 | 0.5235213167 | 0.5235213167 | 20234.16686 |
| chewinggum | 0.9754000902 | 0.7638657689 | 0.4060399769 | 0.07809419781 | 4.093335814 | 36644.92893 |
| fryum | 1 | 0.9066184163 | 0.8328368344 | 0.4490907669 | 7.01730247 | 21375.73528 |
| macaroni1 | 0.9775999784 | 0.9685521722 | 0.9296662499 | 0.2296142763 | 9.267431894 | 21580.95169 |
| macaroni2 | 0.9872000813 | 0.9765995741 | 0.9776408202 | 0.3562729779 | 9.326440436 | 21444.40866 |
| mean | 0.9794323295 | 0.9248523364 | 0.759174465 | 0.3150491454 | 5.188581963 | 42178.03798 |
| rank | 5 | 4 | 14 | 1 | 1 | 1 |
| total | 29 | 29 | 29 | 29 | 29 | 29 |