KLA Project Report

Sujal (EE21B144) Sujay (ED20B065) Navneet (EE21B094)
Devansh (EE21B037)
https://github.com/sujay-2001/MVTec-Reconstruct.git

November 2, 2024

1 Introduction

This project aims to develop an advanced image restoration algorithm using deep learning techniques, specifically targeting images degraded by noise and blur while preserving critical defects of interest. The main focus is to ensure that restored images maintain the visibility and accurate annotation of these defects. The proposed approach employs using the architecture of SADNet (Spatial-Adaptive Network for Single Image Denoising). Performance is benchmarked using standard image quality metrics such as Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM).

We had tried out various methods like using Restormer model (computationally expensive), making our own network for the task (unsatisfactory result), combining segmentation model with a Denoising Transformer (SwinIR) and SADNet. Out of this SADNet gives the best result keeping the computational constraint in mind (Free subscription of Google Colab).

This work was inspired by [1], [2]. , and [3].

2 Objectives

The primary objectives of the project are:

- Develop an algorithm based on Deep Learning and Vision Transformers to restore images degraded by noise and blur, while preserving critical defects of interest.
- Ensure that the restored images maintain visibility and accurate annotation of defects post-restoration.
- Benchmark the performance of the algorithm using validation metrics such as PSNR and SSIM.

3 Methodology

3.1 Spatial-Adaptive Network for Single Image Denoising(SADNet)

Motivation for chosing SADNet

- The results of SADNet demonstrates that it achieves state-of-the-art performances on both synthetic and real noisy images with a relatively small computational overhead. .
- The network can capture the relevant features from complex image content, and recover details and textures from heavy noise.
- The residual spatial-adaptive block proposed in the paper, introduces deformable convolution to adapt to spatial textures and edges. In addition, using an encoder-deocder structure with a context block to capture multiscale information, it estimates offsets and remove noise from coarse to fine.

We used the architecture of SADNet as it is with changes in the loss function which we deemed as more suitable for this task.

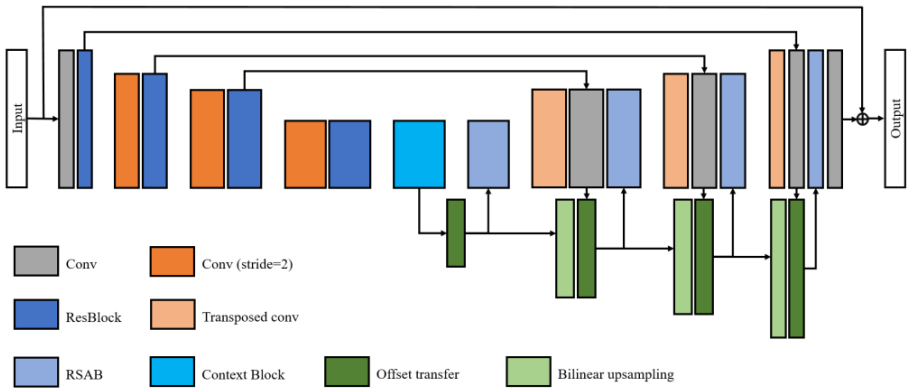


Fig. 1. The framework of our proposed spatial-adaptive denoising network.

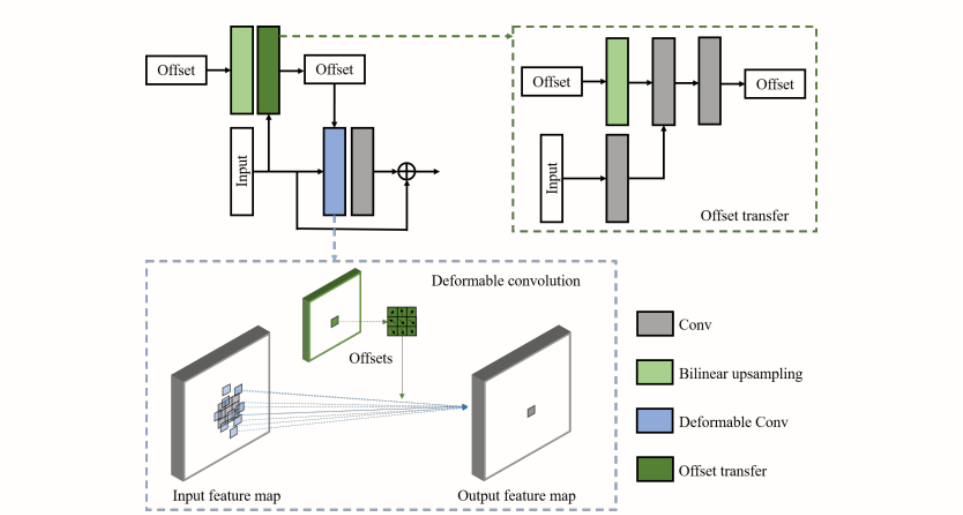


Fig. 2. The architecture of the residual spatial-adaptive block (RSAB). The offset transfer component is shown in the green dashed box. The deformable convolution architecture is shown in the blue dashed box.

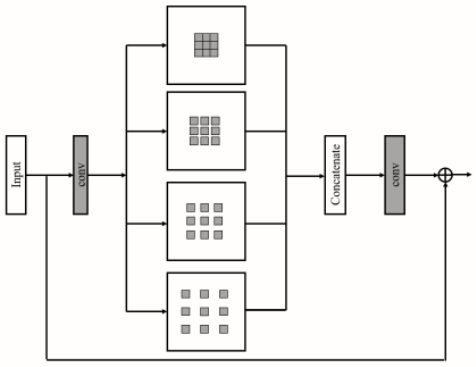


Fig. 3. The architecture of the context block. Instead of downsampling operations, multisize dilated convolutions are implemented to extract different receptive-field features.

4 Validation Metrics

To assess the quality of the restored images, two key validation metrics are used:

- **Peak Signal-to-Noise Ratio (PSNR)**: PSNR is used to measure the ratio between the maximum possible power of a signal and the power of corrupting noise. It helps quantify the quality of the restoration process.
- **Structural Similarity Index Measure (SSIM)**: SSIM measures the similarity between the original and restored images, focusing on preserving structural information and textures, making it suitable for evaluating restoration quality in terms of defect preservation.

5 PSNR and SSIM Formulas

The Peak Signal-to-Noise Ratio (PSNR) is given by:

$$\text{MSE} = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [I(x,y) - K(x,y)]^2$$

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{R^2}{\text{MSE}} \right)$$

The Structural Similarity Index (SSIM) is defined as:

$$\text{SSIM}(I,K) = \frac{(2\mu_I\mu_K + C_1)(2\sigma_{IK} + C_2)}{(\mu_I^2 + \mu_K^2 + C_1)(\sigma_I^2 + \sigma_K^2 + C_2)}$$

where: - $C_1 = k_1 L^2, C_2 = k_2 L^2$

6 Implementation

We implemented a weighted combination of below losses for training

- PSNR loss
- MSE loss
- L1 loss

In the proposed model, we use four scales for the encoder-decoder architecture, and the number of channels for each scale is set to 32, 64, 128, and 256. The kernel size of the first and last convolutional layers is set to 1 × 1, and the final output is set to 1 or 3 channels depending on the input. Moreover, we use 2 × 2 filters for up/down-convolutional layers, and all the other convolutional layers have a kernel size of 3 × 3.

6.1 Weighted Mean Squared Error Loss

The weighted Mean Squared Error (MSE) loss is computed as follows:

- **Element-wise Weight Application:** If a weight tensor w is provided, it is multiplied element-wise with the loss tensor L:

  $$L_{\text{weighted}} = L \odot w$$

  where $\odot$ denotes element-wise multiplication.
- **Reduction:** The weighted loss can be reduced based on the specified reduction mode:
  - If reduction is set to `'none'`, return L_{weighted}.
  - If reduction is set to `'mean'`, compute:

    $$L_{\text{final}} = \frac{\sum L_{\text{weighted}}}{\sum w}$$

    where $\sum w$ is the sum of all weights.
  - If reduction is set to `'sum'`, simply return:

    $$L_{\text{final}} = \sum L_{\text{weighted}}$$

6.2 Weighted L1 Loss

The weighted L1 (Mean Absolute Error) loss is computed similarly:

- **Element-wise Weight Application:** The weight tensor w is applied by multiplying it with the absolute error tensor A:

$$A_{\text{weighted}} = A \odot w$$

- **Reduction:** The final loss can be reduced based on the specified mode:
  - If reduction is set to `'none'`, return A_{weighted}.
  - If reduction is set to `'mean'`, compute:

$$L_{\text{final}} = \frac{\sum A_{\text{weighted}}}{\sum w}$$

    where $A_{\text{weighted}} = |pred - target|$.
  - If reduction is set to `'sum'`, return:

$$L_{\text{final}} = \sum A_{\text{weighted}}$$

7 Experiments

The KLA data was loaded using Dataloader in the Google Colab file. We trained by adopting the multi-scale implementation. A total of 100 epochs were done. The first 50 epochs were trained with image size 512×512 as inputs, the next 25 with image size 640×640 as inputs, and the next 25 with image size 800×800 as inputs. Google Colab's inbuilt T4 GPU was used for training.

We train our model using the ADAM optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ (inbuilt implementation of PyTorch) with a learning rate of 0.0001. The batch size was set to 4. Learning rate scheduler using the Cosine Annealing strategy was applied to the ADAM optimizer with `T_max` = 10, and `eta_min`= 0.0.

For the weighted combination of loss, we used:

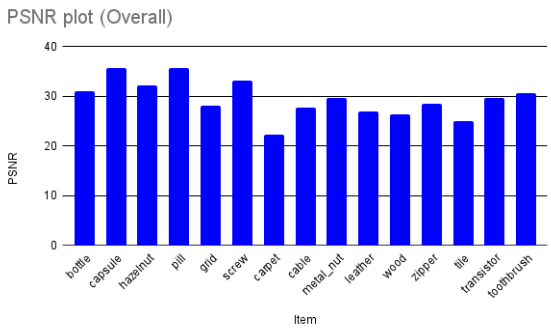
- 0.5 for PSNR loss,
- 1.0 for MSE loss,
- 0.0 for L1 loss.

The rest of the parameters were the same as mentioned in the paper.

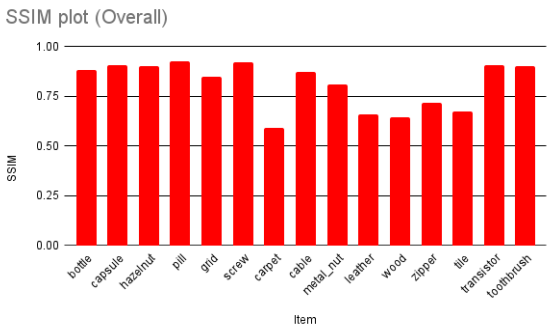
7.1 Quantitative Results

Table 1: Average PSNR and SSIM Values

Metric	Value
PSNR (Overall)	30.0076266
PSNR (Defect)	43.53016483
SSIM (Overall)	0.8212336295
SSIM (Defect)	0.9892118194

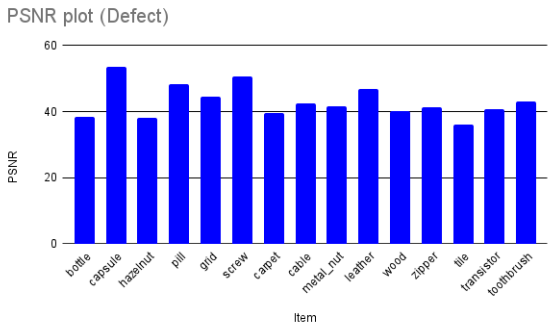


PSNR plot (overall)

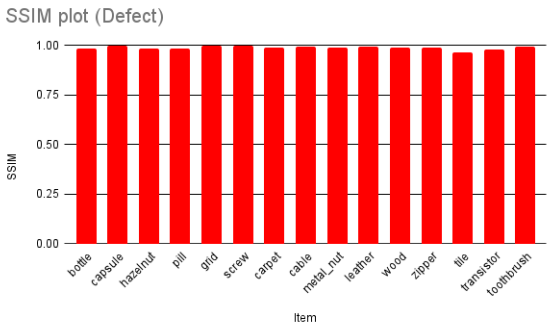


SSIM plot (Overall)

Validation metric



PSNR plot (defect)

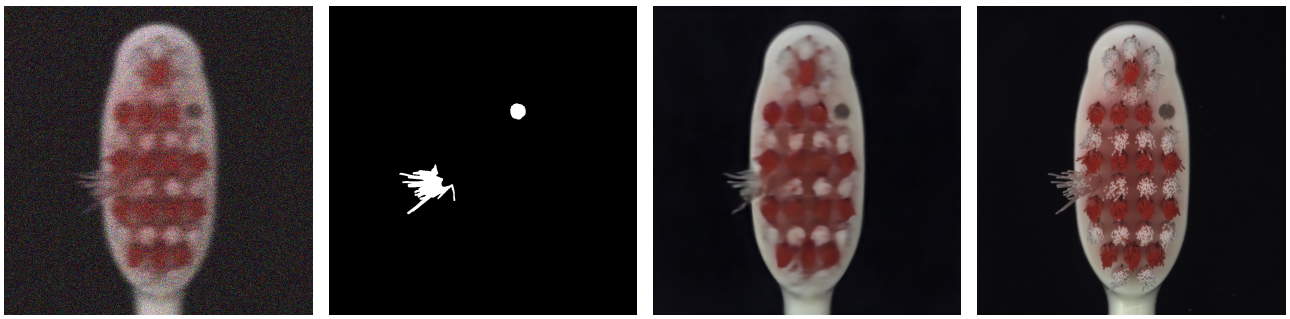


SSIM plot (Defect)

Validation metric

7.2 Qualitative Results

A noticeable smoothness is visible in the Reconstructed images

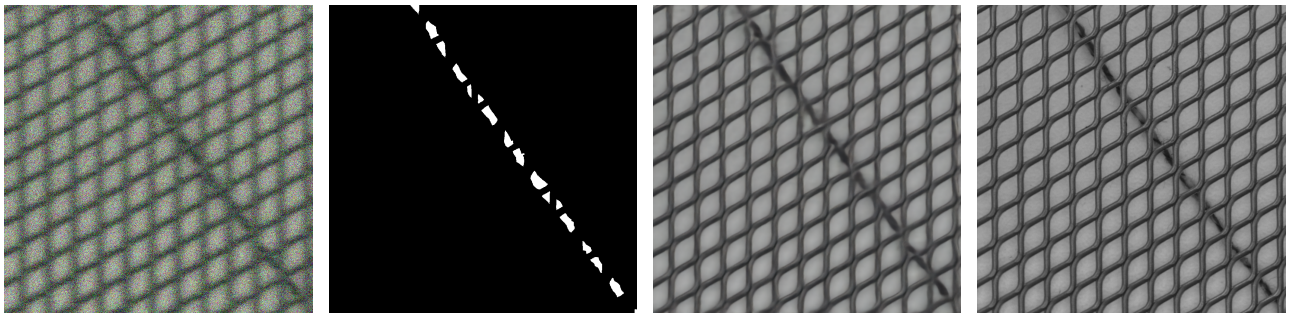


Degraded Masked Reconstructed Clean

Toothbrush

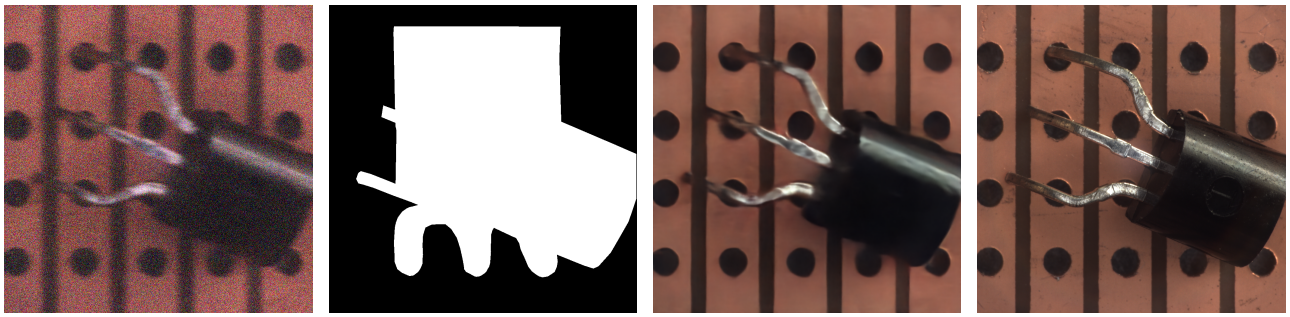
Degraded Masked Reconstructed Clean

Hazelnut



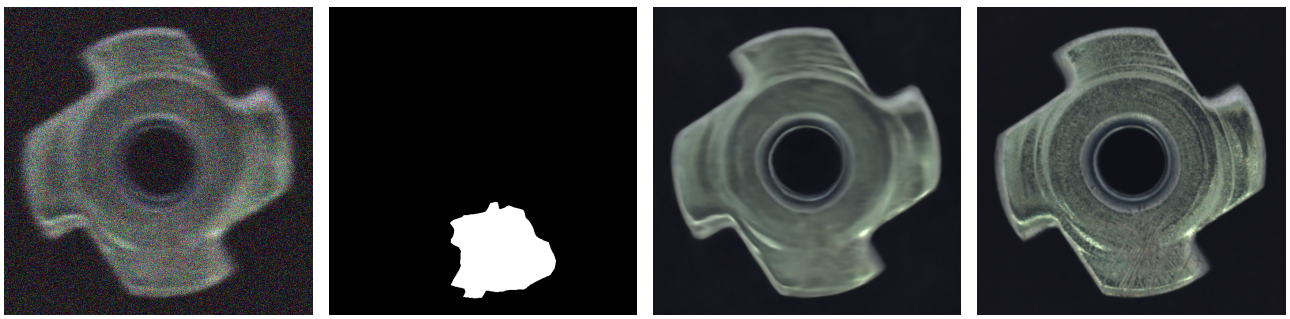
Degraded Masked Reconstructed Clean

Grid



Degraded Masked Reconstructed Clean

Transistor

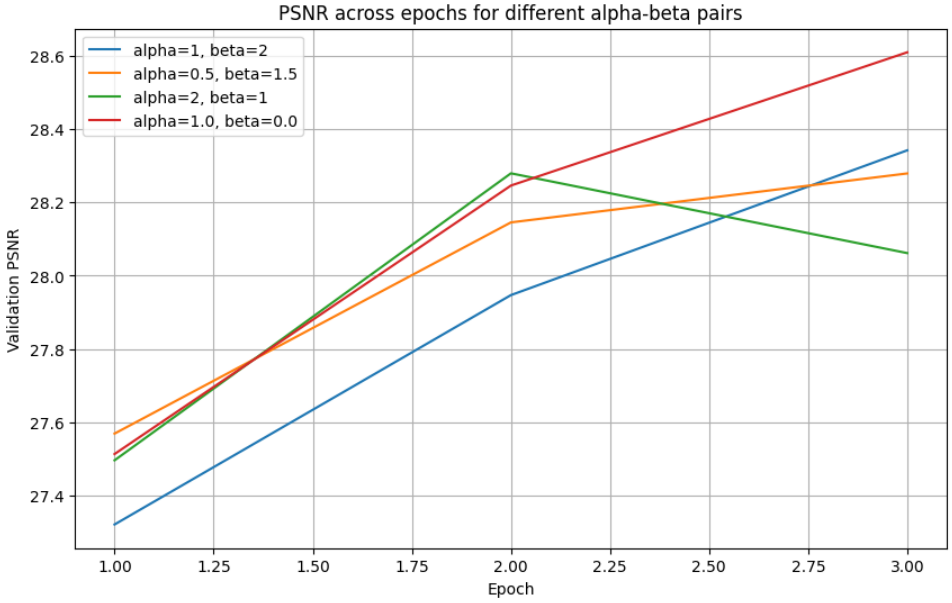


Degraded Masked Reconstructed Clean

Metal nut

8 Ablation study

In this ablation study, we observe that configurations with $\beta \neq 0$ generally achieve higher PSNR than the $\beta = 0$ configuration at higher epochs. The configuration $(\alpha = 1, \beta = 2)$ shows a steeper slope, suggesting it could outperform other configurations if training continues. Conversely, configurations such as $(\alpha = 2, \beta = 1)$ and $(\alpha = 0.5, \beta = 1.5)$ begin to plateau by the third epoch, indicating limited further improvement.



PSNR plot across epochs for different alpha-beta pairs

9 Conclusion

This project uses the Spatial-Adaptive Network for Single Image Denoising (SADNet) architecture to restore images while preserving critical defects. By leveraging the defect masks in tandem with the denoising process, the proposed approach ensures that restored images maintain visibility and accurate annotation of defects, achieving a balance between noise removal and defect preservation. The algorithm's performance is validated using PSNR and SSIM, providing a robust evaluation of its restoration capabilities.

References

[1] Guolei Sun Kai Zhang Luc Van Gool Radu Timofte Jingyun Liang, Jiezhang Cao. Swinir: Image restoration using swin transformer. *https://arxiv.org/abs/2108.10257*, 2021.

[2] Huajun Feng Zhihai Xu Meng Chang, Qi Li. Spatial-adaptive network for single image denoising. *https://arxiv.org/abs/2001.10291*, 2020.

[3] Salman Khan Munawar Hayat Fahad Shahbaz Khan Ming-Hsuan Yang Syed Waqas Zamir, Aditya Arora. Restormer: Efficient transformer for high-resolution image restoration. *https://arxiv.org/abs/2111.09881*, 2021.