EE5179: Deep Learning for Imaging
Project Report

Group Members:
Aswin Balamurugan (CH20B018)
Sanjith Balamurugan (MM20B055)
Shrivarshan K (MM20B058)

Model Architecture:

For this project, we employed a multi-step approach to denoise and restore images from the MVTec dataset.

1. Denoising Stage with Restormer

- The **Restormer model**, a recent state-of-the-art transformer-based architecture, was used in the first stage to perform image denoising. Restormer is particularly adept at handling real-world noise due to its self-attention mechanisms and efficient feature processing, which are explicitly designed for high-quality image restoration tasks. This model is inspired by the work presented in the paper: *Zamir et al., "Restormer: Efficient Transformer for High-Resolution Image Restoration," CVPR 2022*. This paper introduces Restormer's key innovations, including attention within channel dimensions and spatial complexities, which significantly improves denoising performance on high-resolution images. No changes were made to the model, and the pre-trained model was used for denoising.

2. Image Restoration with Custom Model

- After denoising, a custom **PyTorch-based model** was designed for image restoration. This model architecture incorporates residual blocks and channel attention mechanisms to further enhance image quality. The images were first resized to 256*256*3 dimensions before training because of computational constraints.
    - **Residual Blocks**: To capture deeper features and maintain gradient flow through the network, several residual blocks were included. Residual connections allow the model to focus on details without losing low-level information from earlier layers.
    - **Channel Attention**: Channel attention was integrated to focus on informative feature channels selectively, inspired by the idea of channel attention presented in the *Squeeze-and-Excitation Networks* paper (Hu et al., CVPR 2018). This mechanism prioritizes feature channels

essential for the restoration task, enhancing the model's ability to produce clear and sharp outputs.

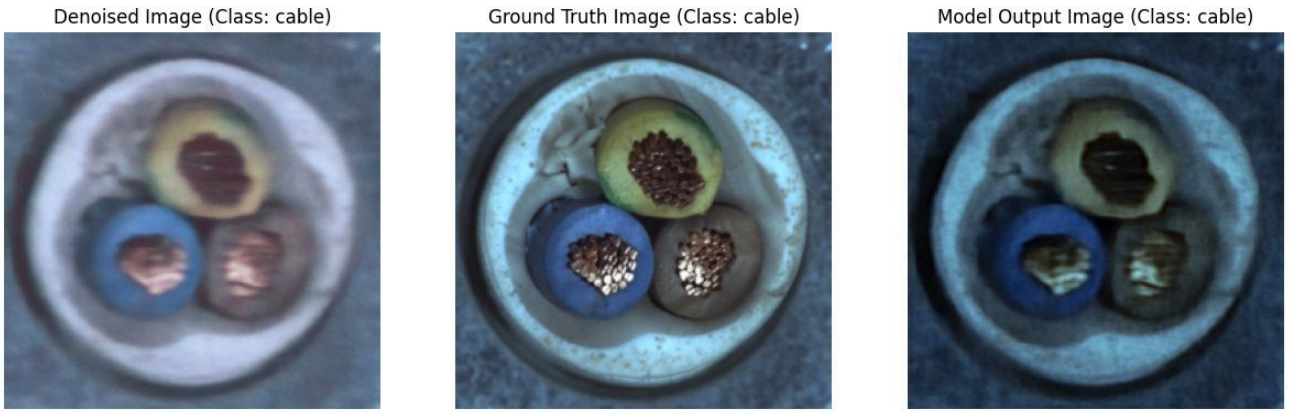
- The model was trained for 20 epochs on Colab's T4 GPU. The training process took around 45 minutes to complete. Optimization was done using Adam with a learning rate of 1e-4. The loss used was Perceptual Loss, which is explained below in detail. The model was then saved in a .pth file for testing purposes.

3. Perceptual Loss with VGG19

- To emphasize perceptual quality over pixel-wise accuracy, **Perceptual Loss** based on VGG19 was used during training. This approach compares the high-level features of the output and target images by extracting feature maps from intermediate layers of a pre-trained VGG19 network (Simonyan and Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 2014). This perceptual loss helps capture fine textures and structural details, which improves visual similarity between the restored and ground truth images.

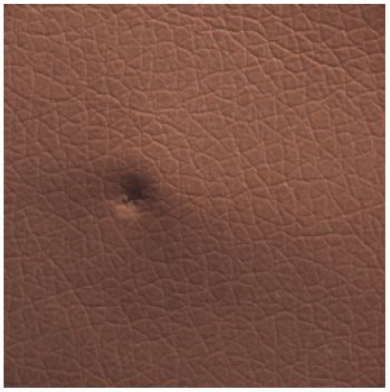
Results:

1. Sample Outputs from Validation Set:



Denoised Image (Class: leather)

Ground Truth Image (Class: leather)



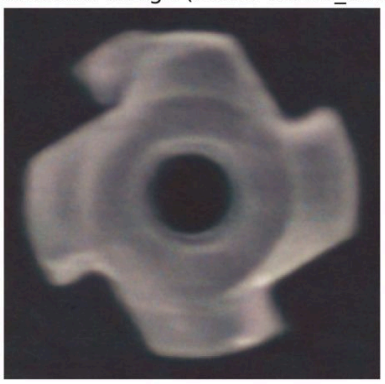
Model Output Image (Class: leather)

Denoised Image (Class: screw)

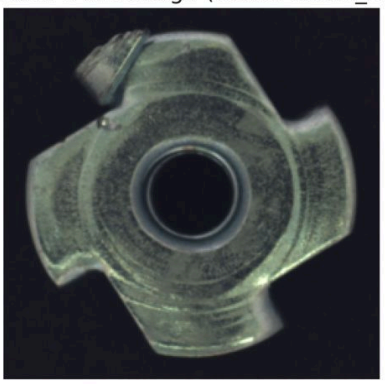
Ground Truth Image (Class: screw)

Model Output Image (Class: screw)

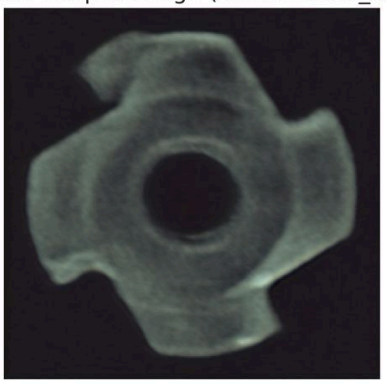
Denoised Image (Class: metal_nut)



Ground Truth Image (Class: metal_nut)

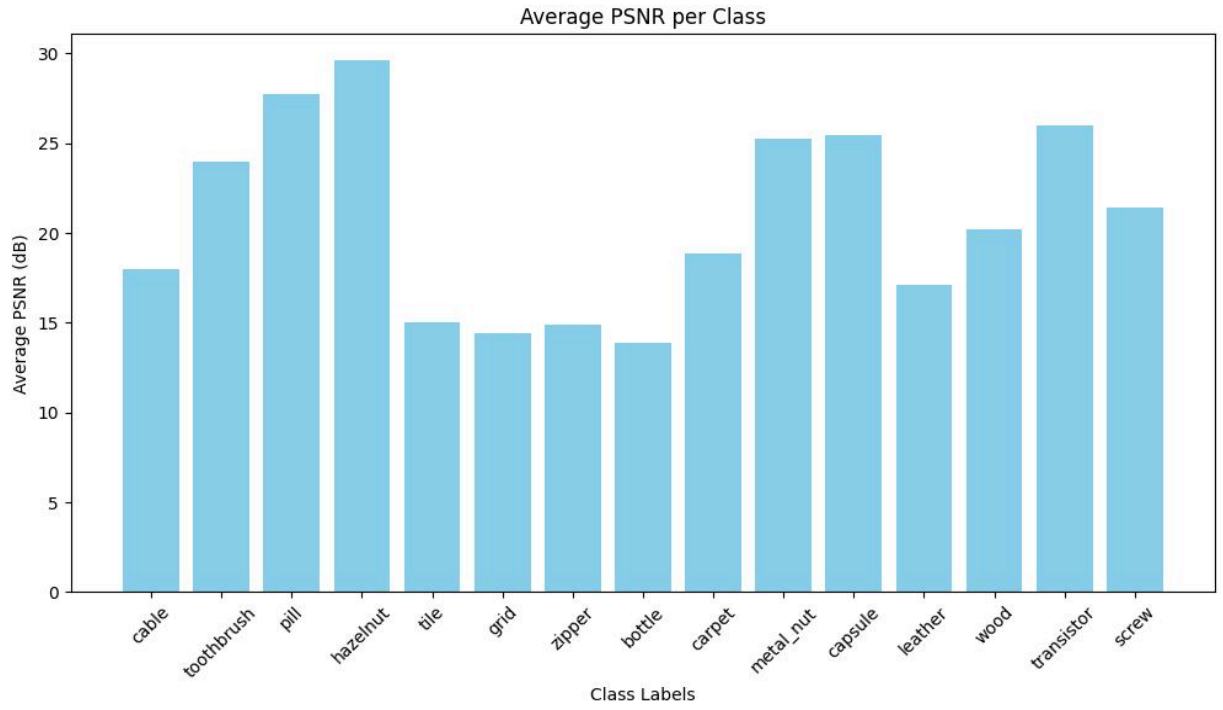


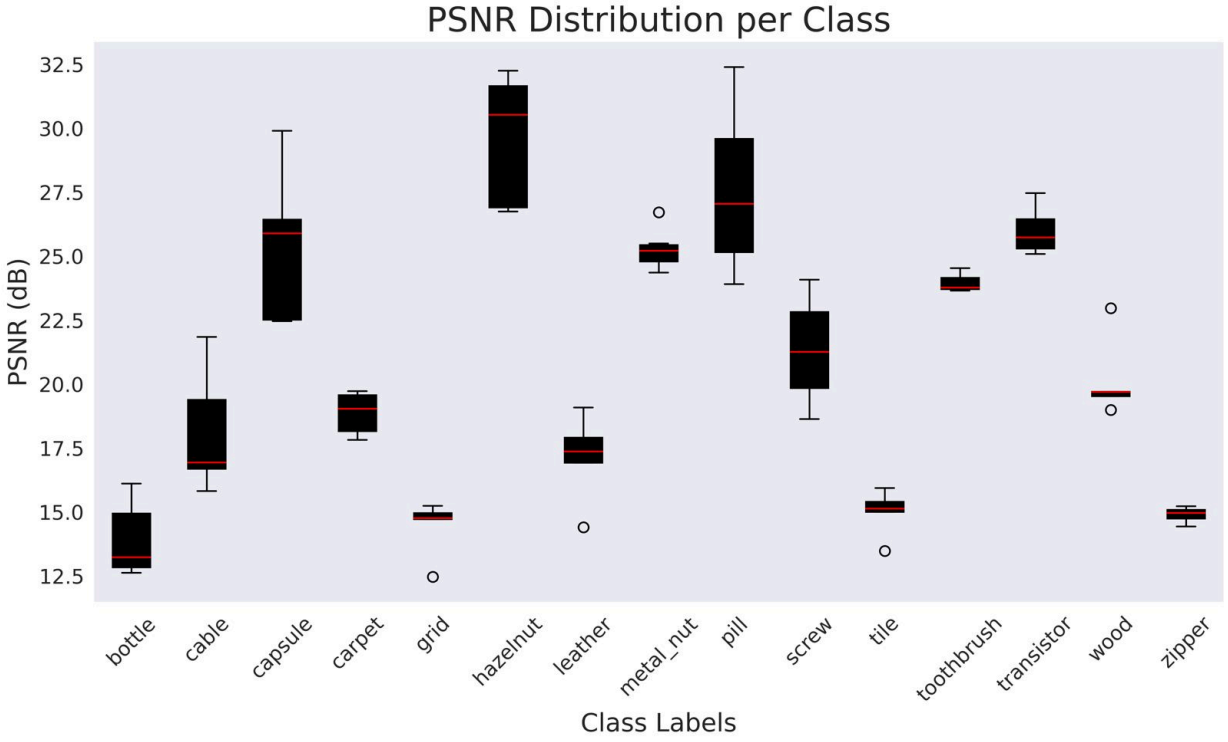
Model Output Image (Class: metal_nut)



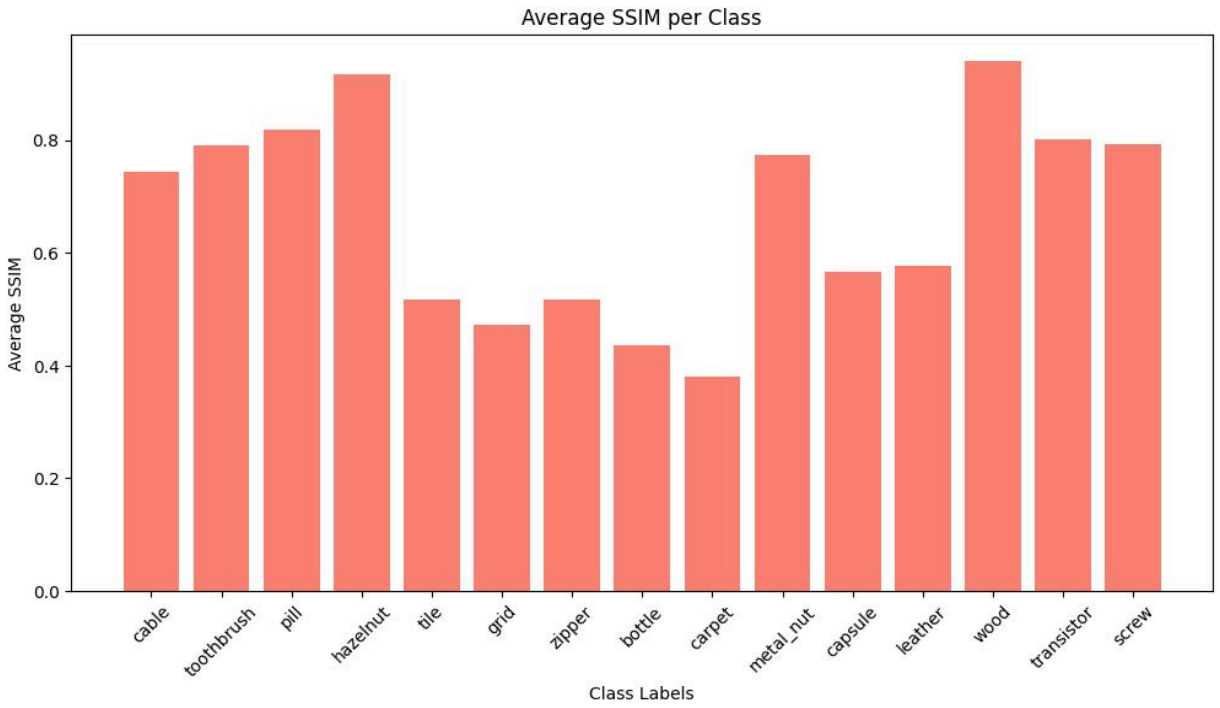
2. PSNR/SSIM Values on Validation Set

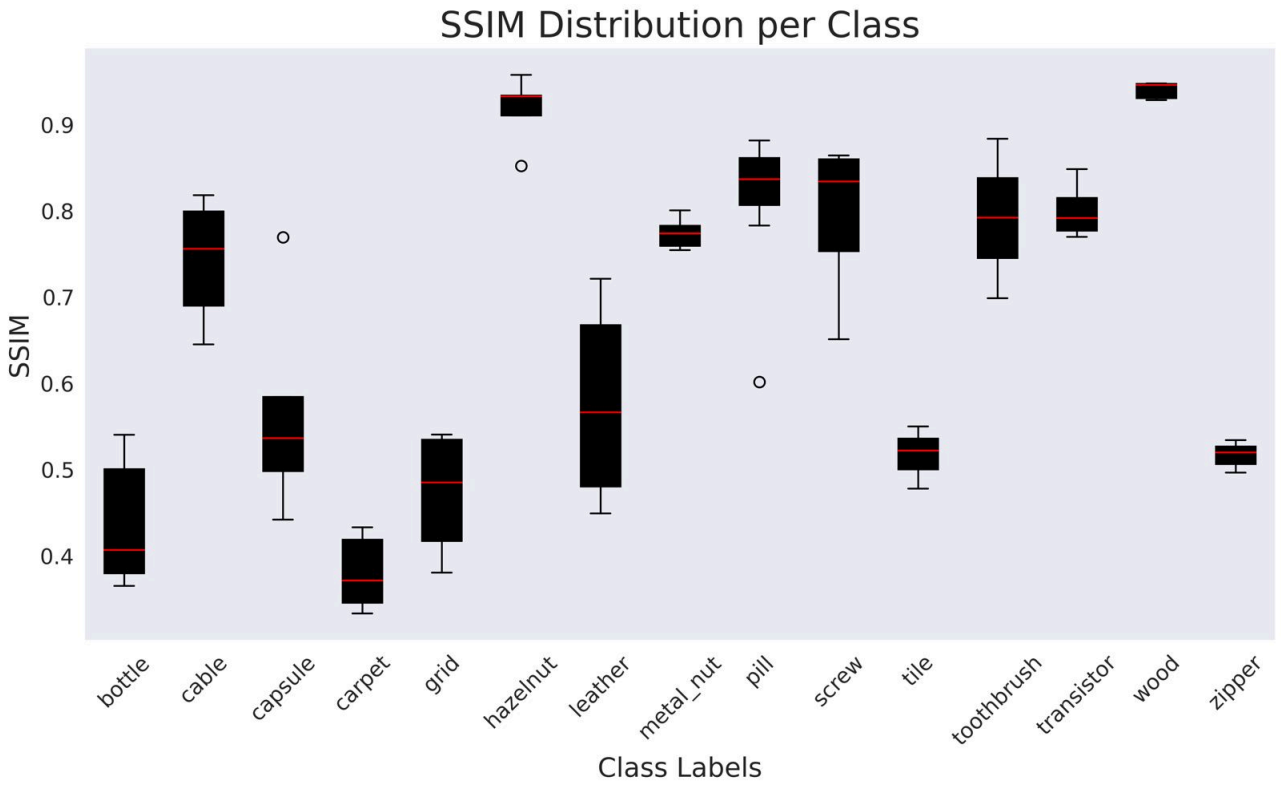
PSNR for each Class:





SSIM for each Class:





3. Average PSNR/SSIM Values:

- **Validation Set**:
  PSNR = **20.79**
  SSIM = **0.67**
- **Train Set:**
  PSNR = **21.03**
  SSIM = **0.69**

Instructions for testing a sample image:

- Open the Testing Notebook in Google Colab or any other system with access to a GPU, which is required to run the model.
- After this, replace the path to test images in the **process_images()** function and the GT_root path. Note that this code assumes that the test images follow the same directory structure as the MVTec train and validation dataset.
- Then, run all the cells in the notebook, and the PSNR and SSIM results are printed at the end. And images are stored in a separate output directory.

Link to GitHub Repo:

https://github.com/uncertainty3/EE5179_Project_Jul-Nov-24

References:

- Zamir, S. W., Arora, A., Khan, S. H., Hayat, M., Khan, F. S., Yang, M.-H., & Shao, L. (2022). **Restormer: Efficient Transformer for High-Resolution Image Restoration**. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 5728-5739. doi: 10.1109/CVPR52688.2022.00566. https://github.com/swz30/Restormer

- Hu, J., Shen, L., & Sun, G. (2018). **Squeeze-and-Excitation Networks**. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7132-7141. doi: 10.1109/CVPR.2018.00745.

- Simonyan, K., & Zisserman, A. (2014). **Very Deep Convolutional Networks for Large-Scale Image Recognition**. *arXiv preprint arXiv:1409.1556*. Available at: https://arxiv.org/abs/1409.1556.

- KAIR GitHub repo: https://github.com/cszn/KAIR

- Hassan, M., Illanko, K., & Fernando, X. N. (2024). **Single Image Super Resolution Using Deep Residual Learning**. *AI, 5*(1), 426-445. https://doi.org/10.3390/ai5010021.