

EE5179: Deep Learning for Imaging

Project Report

Image Restoration from Noisy & Blur Images while Preserving Defects of Interest

Team members:

Bhanu Kiran Verma CE21B031

Navin Sriram ED21B044

Anirud N CE21B014

Pranav Debbad ED21B046

GitHub Repository link

(note that the github repo has different branches for different set of codes)

Problem Statement

The project problem statement, given by KLA, was to develop a deep learning or vision transformer-based algorithm to restore noisy and blurred images without obscuring defects of interest, while ensuring that the restored images maintain the visibility and accurate annotation of defects. This was inspired from manufacturing industries, where image restoration is critical for quality control and defect analysis, because images are often degraded due to noise and blur. For this task, we were given a variation of the MVTec Anomaly Detection dataset in which noise and blur have been introduced to degrade the images.

Methodology

We began with a literature review of existing deep learning models for image restoration, focusing on their ability to denoise and deblur images. We replicated codebases of some architectures (mostly CNNs and U-Nets), and tested them on the degraded dataset. We also explored some hybrid approaches that combined different architectures. A few of our implementations, results and ideas are described in the following sections.

1 DeblurGAN with Deep UNet (Best Model)

• Motivation :

- Results produced by UNet were good but had a flaw and fusion encoder gave worse results than UNet by itself, hence we had to remove discolouration somehow, while maintaining minimal artifacts.
- The idea was to use GAN and early stop it to remove the random discolouration while also preserving the image generated by UNet and maintaining minimal artifacts

• Implementation Details :

- Early Stopping at 5th Epoch
- Additional L1 Noise in the masked Region
- MSE Loss between UNet produced image and GAN produced image to ensure minimal deviation from original image hence minimizing artifact generation but also removing discolouration.
- Learning Rate : 1e-4

Object	PSNR	SSIM
Bottle	28.87	0.87
Cable	26.61	0.89
Capsule	32.23	0.85
Carpet	22.20	0.62
Grid	27.56	0.79
Hazelnut	31.55	0.95
Leather	27.15	0.96
Metalnut	29.33	0.86
Pill	33.66	0.90
Screw	30.99	0.89
Tile	24.78	0.75
Toothbrush	29.81	0.89
Transistor	29.68	0.93
Wood	26.86	0.98
Zipper	29.22	0.79

- Optimizer : Adam (β (0.5, 0.999))
- Activation: ReLU
- UNet based Generator

• **Results :**

- Validation Set Average PSNR : 29.03
- Validation Set Average SSIM : 0.86
- Removes discolouration in UNet Output
- Slightly deblurs the UNet Output
- Minimizes artifact generation

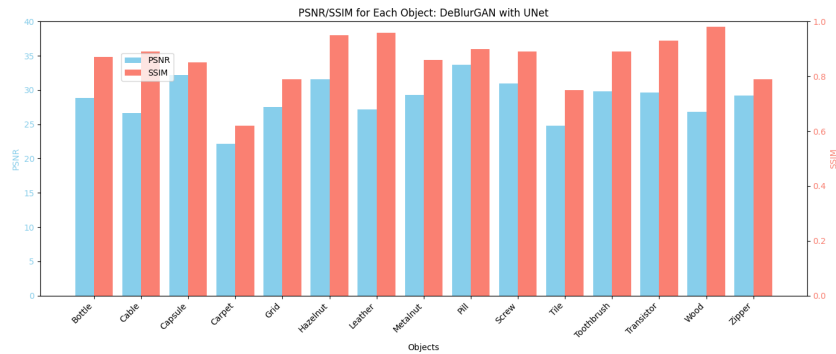
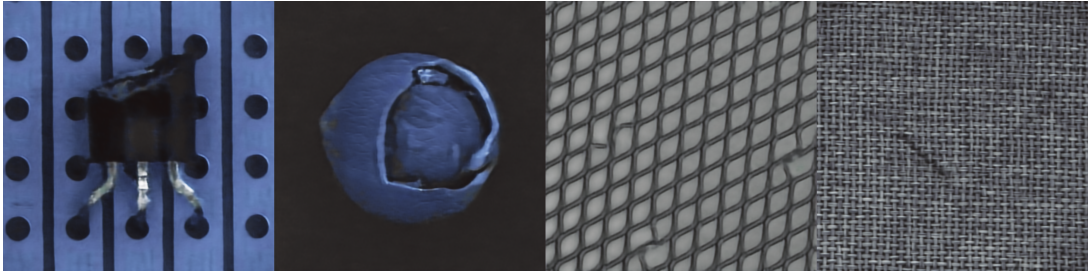
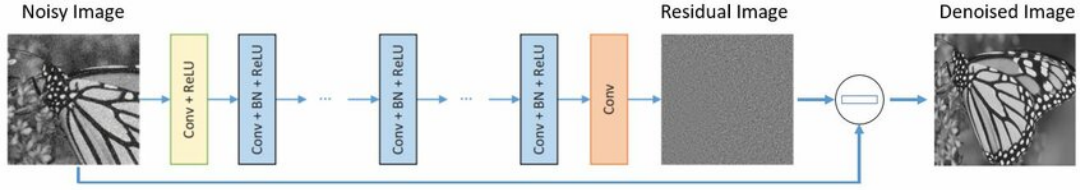


Figure 1: PSNR/SSIM scores object-wise for DeBlurGAN + UNet

2 DnCNN: Residual Learning of Deep CNN for Image Denoising

- **Motivation :** The idea for implementing DnCNN comes from the paper "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising" [1]. DnCNN is a state-of-the-art approach utilizing convolutions with batch norms and residual learning to predict the noise patterns in an image with varying noise levels, subtracting this from the noisy image during test time. Learning the noise (instead of the clean image) reduces the task complexity for the network. It helps the model focus on identifying and removing noise patterns rather than reconstructing the entire image. Since we knew that the noise was random, we wanted to check the denoising performance by trying to predict it. We knew the performance wouldn't be the best possible one, but this is to set the benchmark.
- **Architecture :** The standard DnCNN model has 17 layers of convolutional filters. We use 64 3x3 filters in the first, followed by ReLU Activations. The 15 intermediate layers use filters of the same size with Batch-Norm and the final layer has a single filter with no activations. The architecture is relatively straightforward and simple with L1 loss.



- **Implementation Details :** Images have been taken with the original size of 900 x 900 with all three channels.
 - Learning Rate = 1e-4
 - L1 weight = 1
 - Optimizer = Adam
 - Activation Functions = ReLU
 - Weight Initialization = Orthogonal
- **Results :**
 - Val PSNR: 28.35

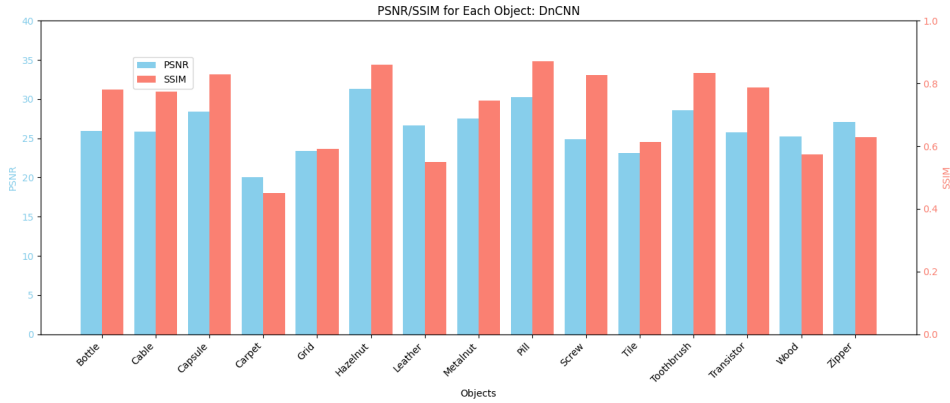


Figure 2: PSNR/SSIM scores object-wise for DnCNN

– **Val SSIM:** 0.714

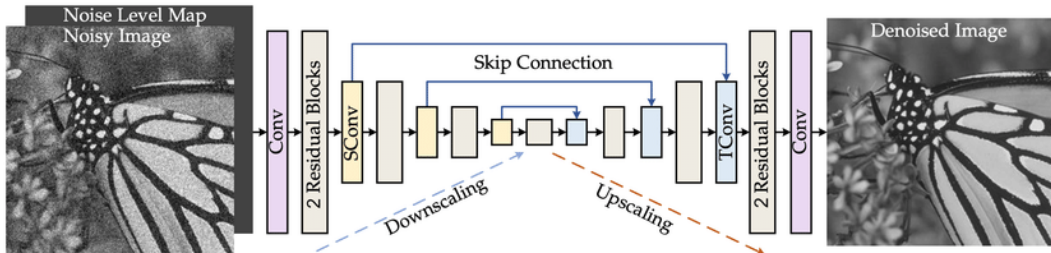
The objectwise PSNR/SSIM scores are shown below in Table 1 and Figure 2:

Object	PSNR	SSIM
Bottle	25.96	0.78
Cable	25.89	0.7741
Capsule	28.43	0.8294
Carpet	20.03	0.4498
Grid	23.36	0.592
Hazelnut	31.29	0.8599
Leather	26.61	0.5497
Metalnut	27.50	0.7461
Pill	30.26	0.871
Screw	24.85	0.827
Tile	23.13	0.6124
Toothbrush	28.60	0.8338
Transistor	25.77	0.7878
Wood	25.27	0.5738
Zipper	27.08	0.6298

Table 1: PSNR and SSIM scores for different objects - DnCNN

DRUNet:

- **Motivation :** The idea for implementing DRuNet comes from the paper ”Deep Plug-and-Play Image Restoration” [2]. The motivation for utilizing DRUNet arises from the need to effectively denoise images while preserving critical defects in various applications. By leveraging a multi-channel input format, DRUNet can integrate a binary defect mask as an additional channel, guiding the model to focus on retaining essential defect details during the denoising process. This approach allows the network to selectively reduce noise in non-defective regions while maintaining the integrity of the defects, addressing the limitations of traditional denoising methods.
- **Architecture :** The DRUNet architecture consists of convolutional layers designed to process the four-channel input (RGB + defect mask). The network employs a U-Net style structure, facilitating skip connections for better feature propagation. The architecture typically includes: - Initial convolutional layers for feature extraction, - Multiple residual blocks to capture complex patterns in the noise, - A final convolutional layer that outputs the denoised image.

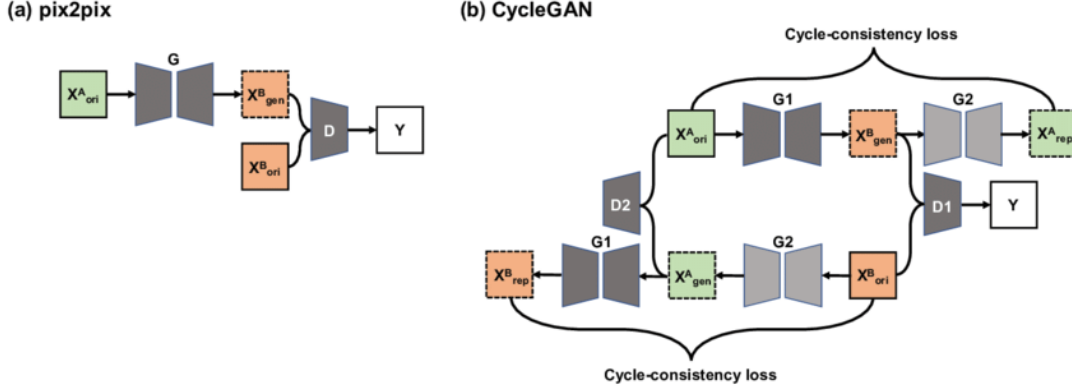


- **Implementation Details :** Images are processed with an original size of 900 x 900 pixels, encompassing all three RGB channels and one binary defect mask channel. - Learning

Rate = 1e-4 - Loss Function = L1 Loss - Optimizer = Adam - Activation Functions = ReLU - Weight Initialization = He Initialization

- **Results :** Val PSNR: 28.35 Val SSIM: 0.8168

3 Pix2Pix-UNet-256



- **Motivation :** The motivation for using Pix2Pix for image denoising lies in its conditional GAN framework [3] and U-Net-based architecture that excels at preserving image structure while generating realistic details. Unlike traditional denoising methods, Pix2Pix's adversarial loss encourages the network to produce outputs that are not only clean but visually plausible, retaining essential textures and fine details that might otherwise be lost
- **Architecture :** The Pix2Pix architecture is a conditional Generative Adversarial Network (cGAN) designed for image-to-image translation tasks, where an input image is transformed into an output image based on specific conditions.

Generator : The generator is structured as a standard U-Net, a type of encoder-decoder network that captures multi-scale features and enables the model to preserve spatial details through skip connections.

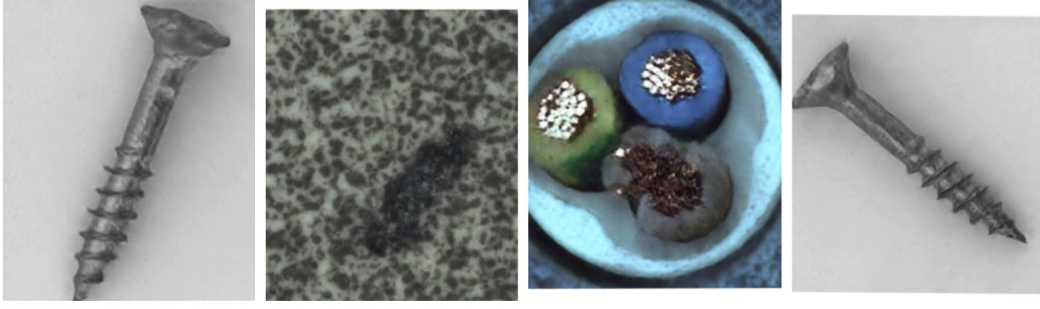
Discriminator : The discriminator in Pix2Pix is a PatchGAN, which evaluates the output image in patches rather than the entire image. This approach focuses on local consistency and texture quality, which is particularly helpful in ensuring that the generated output looks realistic.

Losses : Standard Adversarial loss for fooling the Discriminator and with an additional L1 loss for improving reconstruction accuracy, particularly in the defect region.

- **Implementation Details :** Images have been taken with the original size of 900 x 900 with all three channels
 Generator = UNet-256
 Learning Rate = 2e-4 with lr decay
 GAN Type = lsgan
 Optimizer = Adam ($\beta = 0.5$)
 Activation Functions = ReLU
 Weight Initialization = Orthogonal

- **Results : Wandb logs:** Pix2Pix1 logs Losses:

– G-Gan : 1.263647



- G-L1 : 2.45320993
- D-real: 0.45428
- D-Fake: 0.51708
- **Val PSNR:** 26.30928
- **Val SSIM:** 0.707

4 Pix2Pix-ResNet-9Blocks

- **Motivation :** The motivation for using Pix2Pix for image denoising lies in its conditional GAN framework, using ResNet for image denoising, utilizing the residual learning framework and skip connections, which enable the network to effectively capture and preserve image structure while removing noise. Unlike traditional denoising methods, ResNet’s residual connections allow the network to learn subtle noise patterns without losing essential image details, promoting outputs that are both clean and structurally accurate. This approach helps retain fine textures and intricate features, which might otherwise be smoothed out or lost in standard denoising techniques.

- **Architecture :** The Pix2Pix architecture is a conditional Generative Adversarial Network (cGAN) designed for image-to-image translation tasks, where an input image is transformed into an output image based on specific conditions.

Generator : The generator is structured as a ResNet, a type of residual network that captures hierarchical features through residual connections, enabling the model to effectively learn complex patterns while preserving spatial details. These residual connections

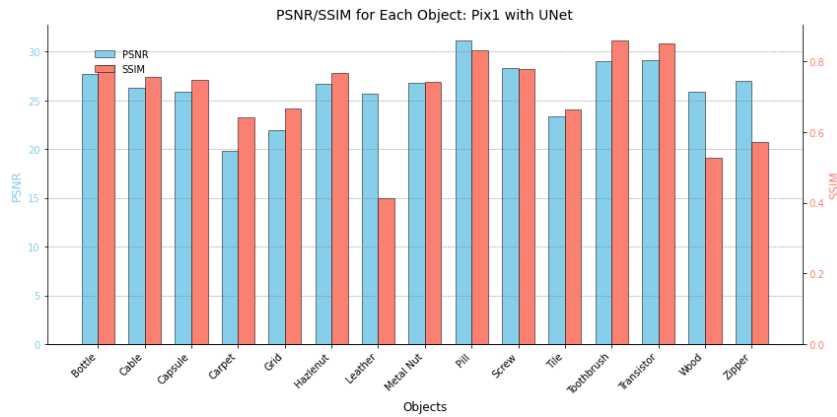


Figure 3: PSNR/SSIM scores object-wise for Pix2pix + UNet

allow feature information to bypass certain layers, reducing the risk of vanishing gradients and helping the network maintain important details

Discriminator : The discriminator in Pix2Pix is a PatchGAN, which evaluates the output image in patches rather than the entire image. This approach focuses on local consistency and texture quality, which is particularly helpful in ensuring that the generated output looks realistic.

Losses : Standard Adversarial loss for fooling the Discriminator and with an additional L1 loss for improving reconstruction accuracy, particularly in the defect region.

- **Implementation Details :** Images have been taken with the original size of 900 x 900 with all three channels
 Generator = ResNet-9-Blocks
 Learning Rate = 2e-4 with lr decay
 GAN Type = lsgan
 Optimizer = Adam ($\beta = 0.5$)
 Activation Functions = ReLU
 Weight Initialization = Orthogonal

- **Results : Wandb logs:** Pix2Pix2 logs

- G-Gan : 0.69486
- G-L1 : 0.30153
- D-real : 0.83204
- D-Fake : 1.97638
- **Val PSNR:** 26.400
- **Val SSIM:** 0.716

5 CycleGAN

- **Motivation :** The motivation behind utilizing CycleGAN [4] for image denoising, even with the availability of noisy-clean pairs, lies in its powerful architecture for learning complex mappings between two domains. While traditional denoising methods often rely on straightforward loss functions to minimize the difference between noisy and clean images,

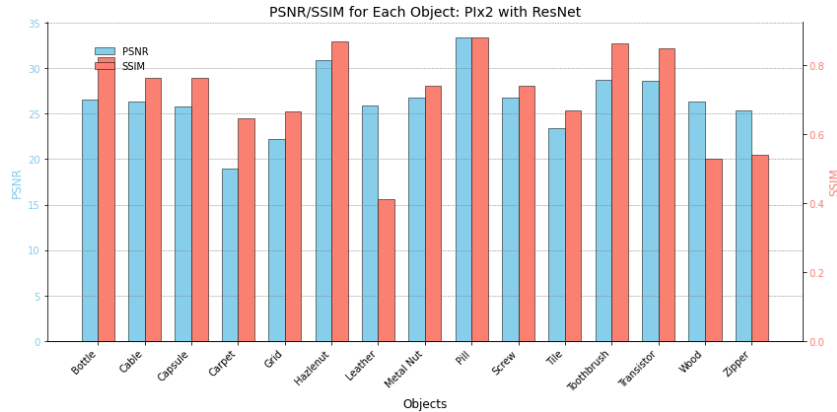


Figure 4: PSNR/SSIM scores object-wise for pix2pix + UNet

CycleGAN introduces the concept of cycle consistency, which ensures that the learned transformation is not only effective but also interpretable. This enables the model to learn to denoise images while preserving important structures

- **Architecture :** CycleGAN employs two generator networks and two discriminator networks to facilitate bidirectional image translation. The architecture includes:

Generator Networks : Two generators G and F that translate noisy images to clean images and vice versa. These generators are typically structured using U-Net or ResNet architectures, allowing them to effectively capture and preserve spatial features during the translation process.

Discriminator Networks : Two discriminators D_{noisy} and D_{clean} that differentiate between real images and generated images in their respective domains, guiding the generators to produce more realistic outputs.

Cycle Consistency Loss: This component of the loss function ensures that an image translated to the target domain can be converted back to the original domain, reinforcing the model's ability to learn meaningful transformations and enhancing the overall quality of the denoised images.

- **Implementation Details :** Images have been taken with the original size of 900 x 900 with all three channels
 Generator = UNet-256
 Learning Rate = 2e-4 with lr decay
 GAN Type = lsgan
 Optimizer = Adam ($\beta = 0.5$)
 Activation Functions = ReLU
 Weight Initialization = Orthogonal
- **Results : Wandb logs:** Cyclegan logs



- Val PSNR: 21.26
- Val SSIM: 0.407

6 UNet

- **Motivation :** Traditional Denoising methods often fail to capture intricate patterns, making deep learning approaches like U-Net valuable. Designed by Ronneberger et al. (2015) for biomedical segmentation [5], U-Net's encoder-decoder structure with skip connections enables precise detail preservation, which is crucial for high-quality denoising and artifact reduction in images.

- **Architecture :** The U-Net architecture is designed as an encoder-decoder network to capture both high-level and fine-grained features:

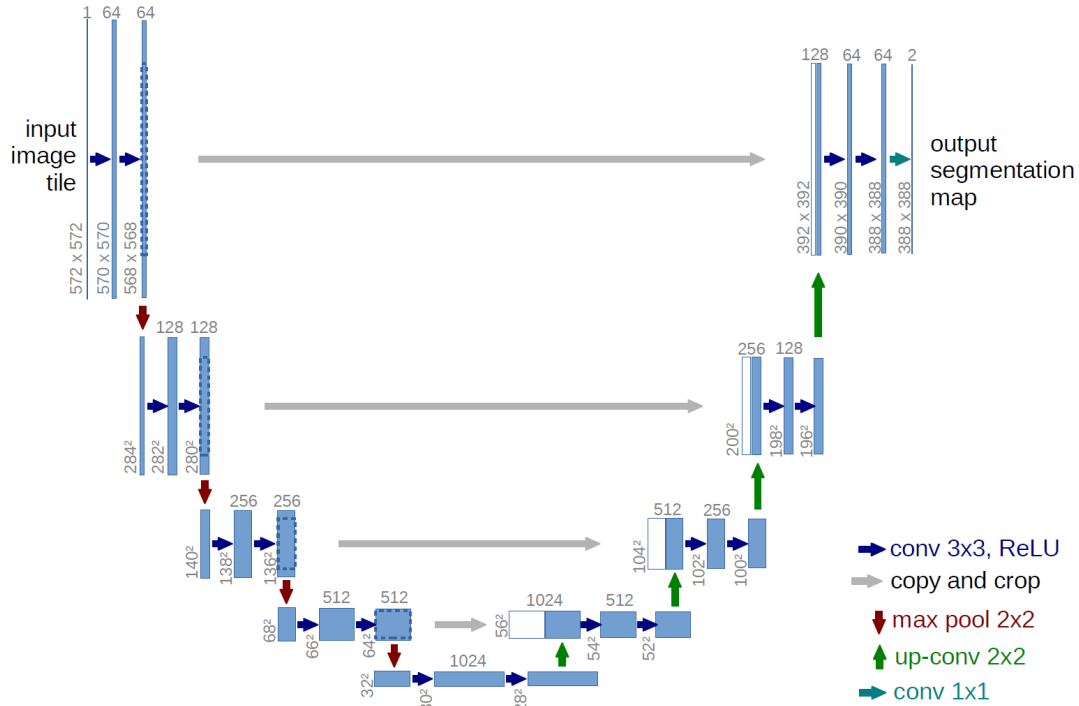
Encoder: The encoder path consists of repeated convolutional and max-pooling layers. Each step includes two convolutions with ReLU activations, followed by max-pooling, which gradually reduces the spatial resolution and increases feature depth, capturing essential details from the input image.

Bottleneck Layer: The bottleneck is the central, most compressed layer that connects the encoder and decoder paths. It captures global context with additional convolutions to help the model understand complex patterns within the image.

Decoder: The decoder mirrors the encoder with upsampling and convolutional layers that progressively restore spatial resolution. Each upsampling is followed by convolutions to refine details and reconstruct the image with a resolution closer to the input.

Skip Connections: Skip connections link each encoder layer to its corresponding decoder layer, providing fine-grained details that were captured at higher resolutions. These connections prevent information loss and aid in producing smoother, artifact-free reconstructions.

Output Layer: A final 1x1 convolution outputs the denoised image with the original resolution. This architecture efficiently balances feature extraction and detail preservation for high-quality image restoration.

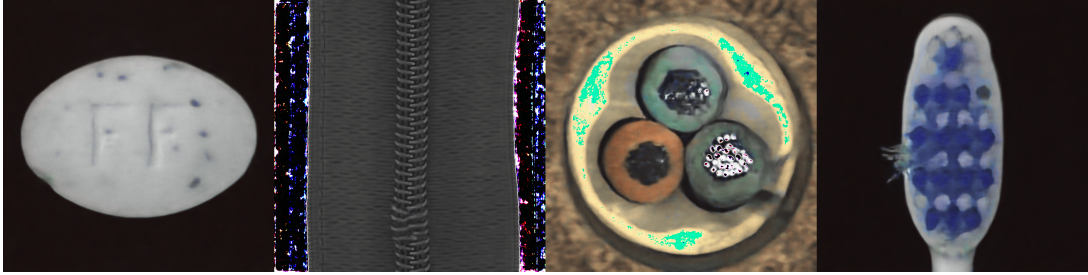


- **Implementation Details:** Images have been taken with original size 900 x 900 instead of the traditional 572 x 572 size in UNet Architectures

- Learning Rate : 1e-4
- Optimizer : Adam
- Activation : ReLU

- **Results :**

- Val PSNR : 28.87
- Val SSIM : 0.87
- Produces discolouration



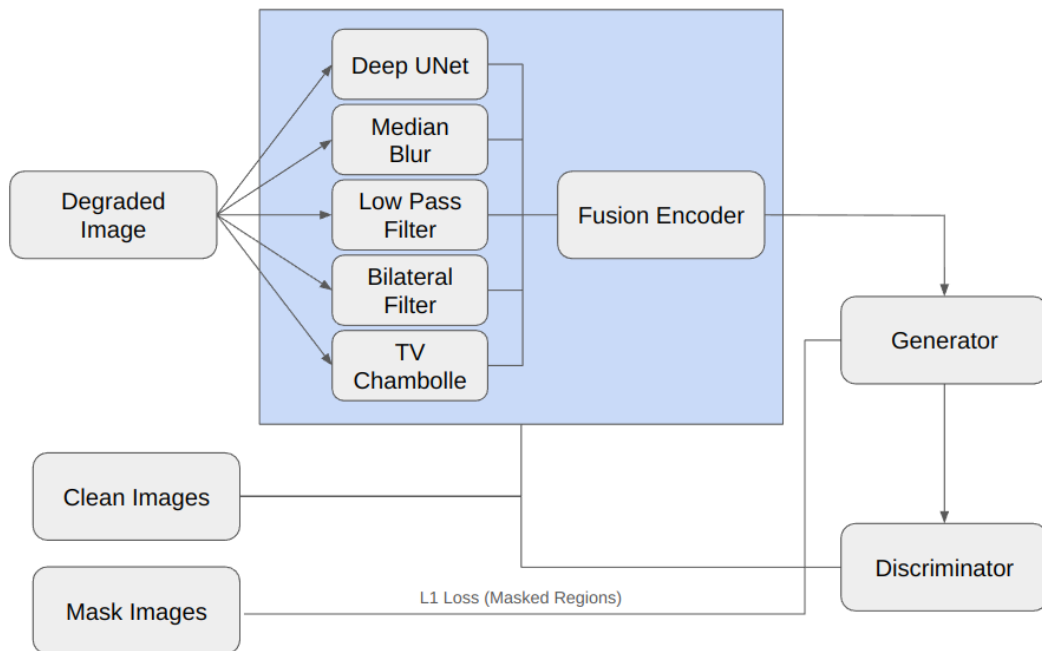
7 Fusion Encoder

• Motivation :

- Results produced by UNet were good but had a flaw i.e. the discolouration in background of the objects, hence we needed a better solution
- Ensemble Learning is a very good approach as it follows the knowledge of the crowd
- We choose Non-Parametric Denoising algorithms as UNet is state-of-the-art model in Deep Learning, hence we went forward with, Median blur, Low Pass Filter in FFT Domain, Bilateral Filter and TV Chambolle denoising

• Architecture :

- We implemented an auto-encoder model which takes in 13 channels as input and gives out an RGB image
- GAN can then further proceed to Deblur the image obtained from Fusion Encoder
- Mask Images were then passed to have an Extra L1 Loss with MSE loss in the masked regions to minimize artifact generation

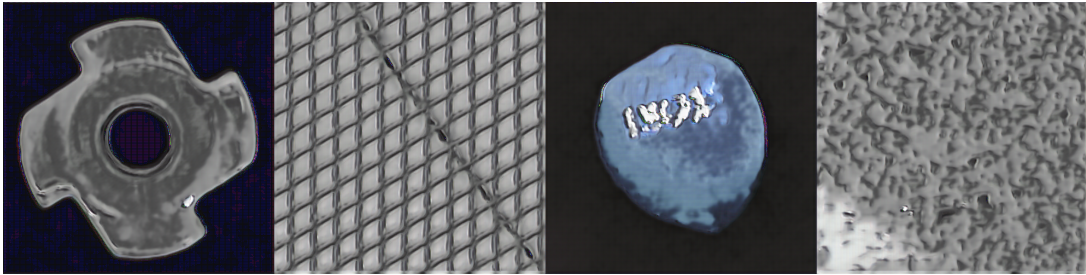


• Implementation Details :

- Learning Rate : $1e-4$
- Optimizer : Adam (β (0.5, 0.999))
- Activation: ReLU
- Fusion Encoder Layers: [12, 64, 128, 256, 512]

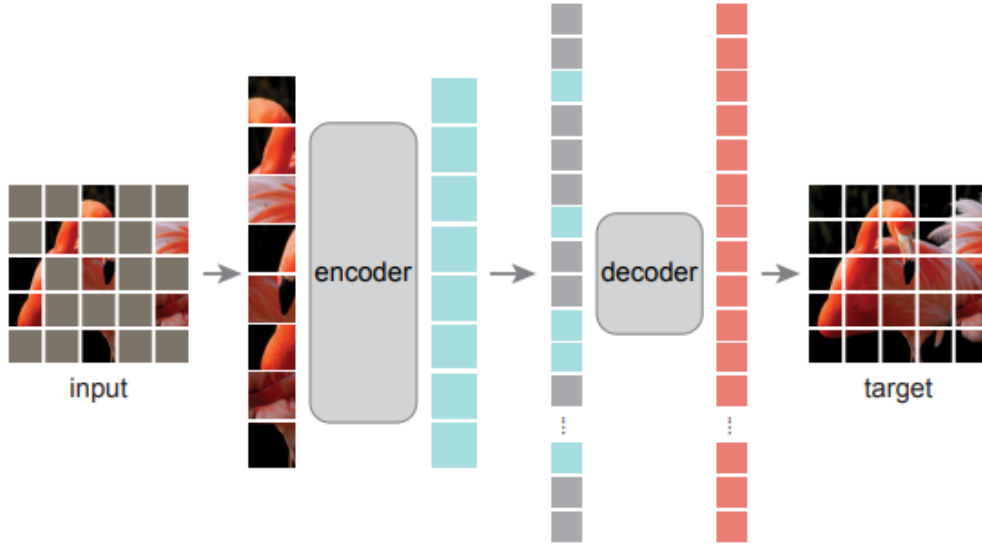
- **Results :**

- Val PSNR : 28.25 (After Auto-Encoder)
- Val SSIM : 0.833 (After Auto-Encoder)
- Val PSNR : 18.72 (After GAN)
- Val SSIM : 0.50 (After GAN)
- The architecture had failed at the fusion encoder phase, the reason being UNet by itself was giving better result than the combination of multiple models
- The Ensemble did not solve the issue of discolouration in background from UNet model, hence we tried a different solution
- Generated Images were having too many artifacts



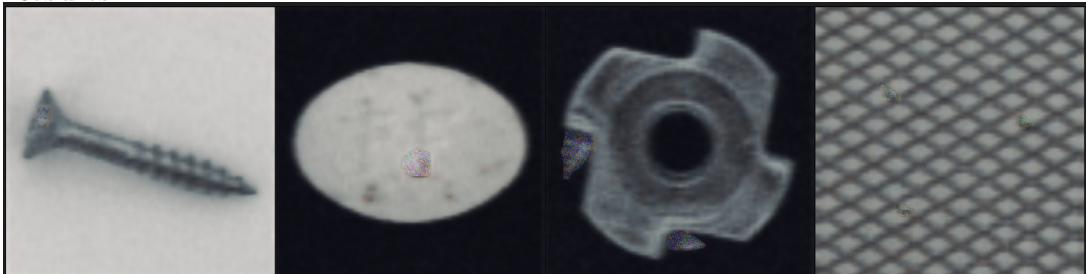
8 Masked Autoencoder

- **Motivation:** The idea of masked autoencoder was taken from the paper "Masked Autoencoders Are Scalable Vision Learners" [6]. The idea is that such a representation of learning would help preserve the information in masked regions. The original paper does random sampling and masks regions randomly to do a better representation learning; here, we already have the set of masks that matter a lot, so we just used this to see how it worked. The idea is the following : The encoder operates only on the visible subset of patches (without mask tokens), and The decoder reconstructs the input from the latent representation along with mask tokens.
- **Architecture:**



The architecture here is a vanilla version of the above. The encoder has three convolutional layers - that increase the channel from 3 to 64 to 128 to 256. Each layer halves the spatial dimensions while increasing the feature channels. The decoder reconstructs the encoded features using three transposed convolutional layers. This has 259 channel input (256 from the encoder features + 3 from the masked regions input). The decoder output is passed through a sigmoid activation to ensure pixel values are between 0 and 1, suitable for normalized image data.

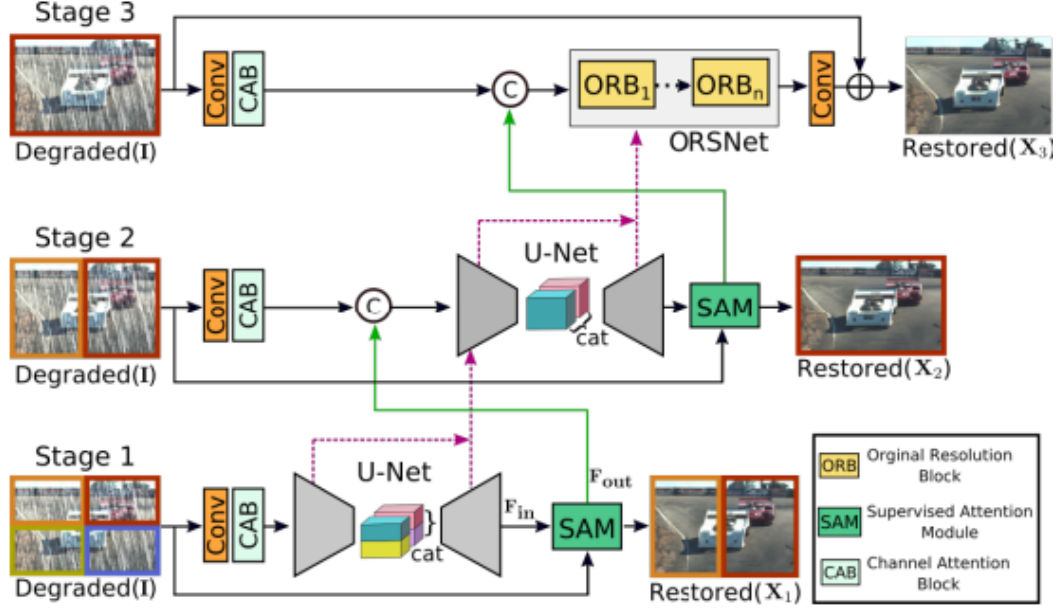
- **Implementation Details :** The learning rate was set to 0.001, and the batch size was set to 32 due to computing power issues. The optimizer is Adam.
- **Results :**



The average PSNR score on Val was found to be 24.5. This showed the need to do better algorithms. As seen in the above image, there are a few artifacts after denoising.

9 MPRNet

- **Motivation:**
The motivation has been taken from [7]. The main proposal is a multi-stage architecture that progressively learns restoration functions for the degraded inputs, breaking the overall recovery process into more manageable steps. In the first two stages, a U-Net architecture is applied with an encoder and decoder to extract useful features. In the final stage, the model operates at the original image resolution to minimize information loss. Cross-stage feature fusion is used to further mitigate any loss of information. Note that this model has not been specifically fine-tuned for masking. We tried this model as an initial baseline and attempt.
- **Architecture:**



The encoder-decoder subnetwork is based on a standard U-Net architecture, incorporating channel attention blocks (CABs) to enhance feature extraction at each scale. CABs are also applied to the feature maps in U-Net skip connections. Instead of using Transposed convolutions to increase the spatial resolution of features in the decoder, bilinear upsampling followed by a convolution layer is used to reduce checkerboard artifacts that often result from Transposed convolutions. A supervised attention module (SAM) is introduced between each stage to achieve a notable performance gain. This module offers two main benefits: it provides ground-truth supervisory signals to support progressive image restoration at each stage, and it generates attention maps that suppress less informative features, allowing only the most relevant ones to propagate to the next stage. To preserve fine details from the input image to the output image, we introduce the original-resolution subnetwork (ORSNet) in the last stage

- **Implementation Details :**

- Trained on 256×256 patches with a batch size of 16 for 100 epochs - due to computer power issues, we couldn't upsample it to 900×900 .
- Optimizer: Adam, with an initial learning rate of 2×10^{-4} , reduced to 1×10^{-6} using cosine annealing.

- **Results :** The average PSNR score was low - around 23 on the Val set. Since this was one of the initial things we tried out and didn't work, we did not focus much on this as we worked further to incorporate the masks given in the data.

10 NLM

- **Motivation:** The motivation was taken from [8]. The idea here is a complete image processing setup, where we replace the color information of a pixel with the average of color intensities that are "similar" and note that these similar pixels need not always be "local" or nearby. Rather than comparing single pixels, it compares entire patches around each pixel. A patch-based comparison ensures that individual colors and overall textures and patterns around each pixel are considered, making it more accurate.

$$NLu(p) = \frac{1}{C(p)} \int f(d(B(p), B(q))) u(q) dq \quad (1)$$

where:

- $d(B(p), B(q))$: Euclidean distance between image patches centered at p and q ,
- f : a decreasing function that weights similarity,
- $C(p)$: a normalization factor ensuring that the weights sum to 1.

• **Architecture:**

In denoising a color image $u = (u_1, u_2, u_3)$, each patch $B = B(p, f)$ is centered at a pixel p with size $(2f + 1) \times (2f + 1)$. For each color channel $i = 1, 2, 3$, the estimate $\hat{B}_i$ is computed by:

$$\hat{B}_i = \frac{1}{C} \sum_{Q=Q(q,f) \in B(p,r)} u_i(Q) w(B, Q), \quad (2)$$

where C is a normalization factor defined as:

$$C = \sum_{Q=Q(q,f) \in B(p,r)} w(B, Q). \quad (3)$$

Here, $B(p, r)$ represents a neighborhood centered at p with size $(2r + 1) \times (2r + 1)$ pixels, and $w(B(p, f), B(q, f))$ is a weight function as in the pixelwise implementation.

By applying this procedure across all patches, multiple estimates for each pixel are obtained. The final denoised image is formed by averaging these estimates:

$$\hat{u}_i(p) = \frac{1}{N^2} \sum_{Q=Q(q,f) | q \in B(p,f)} \hat{Q}_i(p), \quad (4)$$

where $N^2 = (2f + 1)^2$ is the number of possible estimates per pixel.

The patchwise implementation offers improved PSNR due to greater noise reduction in the final aggregation, reducing spurious noise near edges. However, it does not enhance the preservation of details.

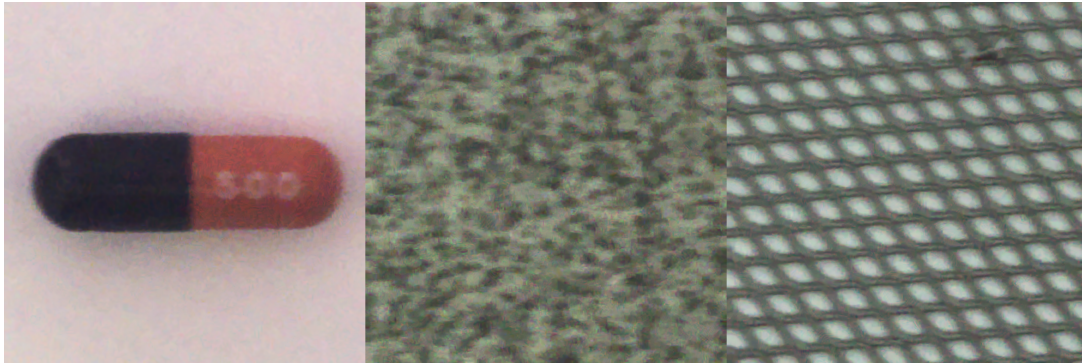
- **Results :** Psnr : 16.52 and ssim : 0.15 - pretty bad results on the val data

11 BM3D

- **Motivation:** This is yet another image processing setup we tried as a preliminary attempt to solve the problem. The original reasons are to validate the need for deep algorithms and, if possible, find a way to use these processed images as inputs to the deep models. Due to time constraints, we couldn't complete this task, but we aim to do it further. The algorithm starts by dividing the noisy image into small overlapping patches called blocks. For each patch, it searches for similar patches within a defined neighborhood in the image. The goal is to group similar patches together because they represent the same underlying image structure. Collaborative filtering uses a 3D transformation - discrete cosine transform to sparse the grouped patches, making denoising easier. Then, the algorithm uses hard thresholding to remove unwanted noise. The above steps are the preliminary steps. Let's go into the weiner filtering part: We further pass the image

to the weiner filter using the preliminary steps. A Wiener filter estimates a desired or target random process by linear time-invariant (LTI) filtering an observed noisy process, assuming known stationary signal, noise spectra, and additive noise. The filtered patches are finally added to the denoised image.

- **Results** : PSNR: 22.82 and SSIM: 0.5766 - a good enough result for a non-parametric image processing model.



Further ideas of dataset augmentation:

We wanted to implement several ideas for augmentation, but we could not do it before the deadline. The main idea is that we used a GAN to generate more noisy and noise-free image pairs, which would help augment the data for further training. This could then be used to train our above networks for better sample efficiency - get more data from less data and train. We could consider this as an adversarial model that generates more datasets for training.

Overall project experience

This has been a wonderful experience, as we found a huge learning curve. We were exposed to the current real-world problems in computer vision and deep learning, and this gave us more insight into even the implementation details like compute power, sample efficiency, etc. We thank both KLA and the professor for such a great learning opportunity.

Team members

Bhanu Kiran Verma CE21B031

Navin Sriram ED21B044

Anirud N CE21B014

Pranav Debbad ED21B046

References

- [1] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.
- [2] Kai Zhang, Yawei Li, Wangmeng Zuo, Lei Zhang, Luc Van Gool, and Radu Timofte. Plug-and-play image restoration with deep denoiser prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):6360–6376, 2022.

- [3] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017.
- [4] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, 2017.
- [5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.
- [6] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15979–15988, 2022.
- [7] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Multi-stage progressive image restoration. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14816–14826, 2021.
- [8] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. Non-Local Means Denoising. *Image Processing On Line*, 1:208–212, 2011. https://doi.org/10.5201/ipol.2011.bcm_nlm.