Project Report

Names: Manoj Gembali & Tushar Gwal

Student ID's: A20527288 & 20449419

Title of the Project: *Image Restoration Using Deep Image Prior*

Option 1: Implementation of a Research Paper

Main Paper:

- *Title:* Deep Image Prior
- *Authors:* Dmitry Ulyanov, Andrea Vedaldi, Victor Lempitsky
- *Publication Venue:* Published at IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
- *Year:* 2018
- *DOI/Link:* https://arxiv.org/pdf/1711.10925v4

Team Member Responsibilities:

- *Tushar Gwal* - Project coordination, implementation of denoising and Reconstruction tasks, report writing.
- *Manoj Gembali* - Implementation of Inpainting and Hole Filling tasks, Implementation of maskcreator script, Images Preprocessing, experimental analysis and report writing.

Abstract

This report presents an implementation of the Deep Image Prior (DIP) approach for image restoration, focusing on denoising, inpainting, and hole filling tasks. Unlike traditional supervised techniques requiring large datasets, the DIP approach uses a convolutional neural network (CNN) with randomly initialized weights as an implicit prior for natural image structure. The CNN architecture serves as a restoration tool when trained directly on corrupted images. This report describes the model architecture, training process, and performance evaluation using metrics such as Peak Signal-to-Noise Ratio (PSNR), Mean Square Loss (MSE) and Structural Similarity Index Measure (SSIM). The results show that DIP offers an effective alternative for restoring images without external data.

Table of Content

1. Introduction

- **Problem Statement**
  Image restoration encompasses tasks such as denoising, inpainting, reconstruction, and super-resolution, which aim to recover original image quality from degraded inputs. Conventional image restoration approaches often rely on supervised learning models trained on large datasets. However, obtaining and annotating high-quality datasets is both costly and impractical, particularly in low-resource scenarios or single-image applications, such as specialized medical imaging or remote sensing in unique environments. Additionally, traditional methods can suffer from poor generalization to unseen types of degradation. These challenges highlight the need for a more flexible solution that can operate without large, annotated datasets.

- **Background and Motivation**

  Traditionally, convolutional neural network (CNN) based methods for image restoration rely on explicit priors. A prior encodes assumptions or beliefs about the properties of the target image, guiding its reconstruction. Classical priors, such as Total Variation (TV) [2] and non-local means, are used to reduce noise and enhance structure. These handcrafted priors often fail to capture the complexities of natural images, limiting the performance of traditional restoration methods.

  The Deep Image Prior (DIP) approach offers a unique solution by bypassing the need for external training data. Instead, it leverages the architectural structure of a CNN, trained from scratch on a single corrupted image, to reconstruct and enhance the image. The motivation for this method arises from the observation that CNNs inherently favor natural image structures, which can be exploited for restoration without a dataset. This architecture-based prior is surprisingly effective at tasks such as noise reduction and resolution enhancement, making it an attractive alternative to supervised learning-based techniques.

- **Objectives**
  The goal of this implementation is to explore and validate the *Deep Image Prior* approach by applying it to various image restoration tasks and evaluating its effectiveness.

2. Proposed Solution

2.1 DIP as a novel approach

In traditional approaches, when dealing with degraded images, we need two key components:

- **Data Term**: Measures how well our solution matches the degraded input.
- **Prior Term**: Guides the reconstruction toward natural-looking images.

Traditional reconstruction can be formulated as: $x^* = \arg\min_x E(x; x0) + R(x)$

where:

- **E(x; x0)** is the data term.
- **R(x)** is the prior/regularizer.
- **x0** is the degraded image.
- **x*** is the restored image.

Whereas in Deep Image Prior (DIP), as opposed to traditional image restoration tasks with explicit priors, the approach leverages the structure of a convolutional neural network itself as an implicit prior for natural images. Instead of using pre-trained networks or explicitly defined regularization terms, DIP employs an untrained CNN that maps fixed random noise to an output image. The network weights become the optimized parameters, effectively reparametrizing the image as:

$x = f_\theta(z)$

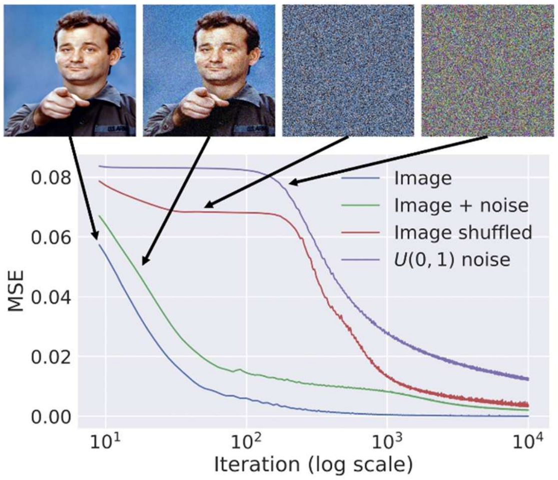
The optimization objective is to find the optimal network parameters **θ*** that minimize the error between the network output and the corrupted image, without any explicit prior term.

$\theta^* = \arg\min_\theta E(f_\theta(z); x0)$

where:

- **f_θ** is an untrained CNN.
- **z** is fixed random noise.
- **θ** are network parameters.
- No explicit prior term **R(x)** is used; the prior is implicit in the CNN architecture.
  The optimization objective is to find the optimal network parameters **θ*** that minimize the error between the network output and the corrupted image, without any explicit prior term.

2.2 Why it Works



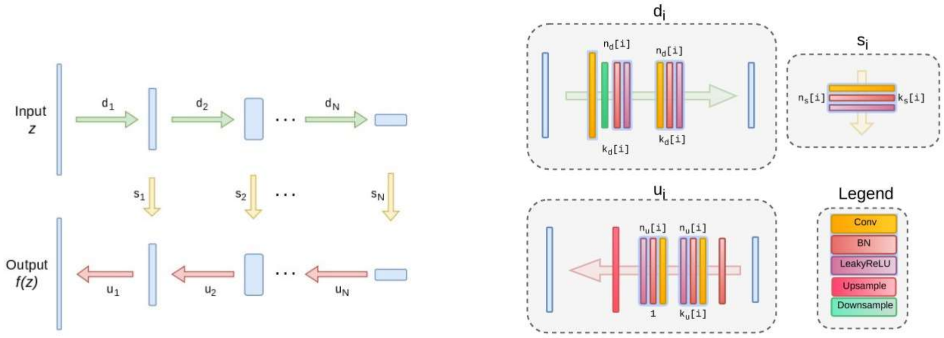
The Deep Image Prior (DIP) works effectively due to key properties of convolutional neural networks (CNNs) that provide a natural fit for image restoration. One major reason is the **inductive bias** of CNNs. Despite their capacity to overfit , CNNs have a strong bias towards structured data [3], meaning that their architecture itself imposes meaningful constraints, which makes them more likely to produce coherent, natural-looking outputs.

Another reason is the **network's natural impedance**. From the figure, when different types of inputs were given to the net, we can observe that CNNs converge quickly on natural image structures but slowly on noise or artifacts. This distinction helps the network resist fitting noise while preserving the image content. Using **early stopping** takes advantage of this difference in fitting speed, allowing restoration without the network overfitting to undesired noise.

Multi-scale structure properties also play an essential role. CNNs progressively refine random noise into meaningful image features through hierarchical organization. **Skip connections** help preserve and enhance details across scales, maintaining image quality. This ability to exploit self-similarities and progressively refine features is a critical factor in DIP's success.

Finally, the CNN architecture itself provides **implicit regularization**. The convolutional layers promote smooth, structured outputs that match natural image statistics, inherently reducing the need for explicitly defined priors or regularization terms. This implicit regularization ensures that DIP produces high-quality restored images, even in the absence of external training data.

2.3 Model Architecture



Encoder-Decoder Structure

The DIP architecture follows an encoder-decoder design inspired by the U-Net as seen in the left figure. This setup is essential in transforming random noise input into coherent image representations:

- **Encoder:** The encoder is responsible for progressively reducing the spatial dimensions of the input while increasing the feature maps' depth. This transformation helps the network capture global features by passing the input through multiple layers of convolutions, each typically followed by a non-linear activation function like **LeakyReLU**. The encoder can be thought of as compressing the image into a set of latent features that represent its core structure.
- **Decoder**: The decoder reconstructs the image from these latent features by progressively increasing the spatial dimensions through **upsampling** and **transposed convolutions**. The upsampling modes typically include **bilinear** or **nearest-neighbor upsampling**, both of which play significant roles in ensuring smooth spatial transitions in the output image. The use of upsampling is a deliberate choice over transposed convolutions to avoid grid artifacts, which can often occur in the latter.

Skip Connections

Skip connections links the corresponding layers of the encoder and decoder which allows the model to retain both high-level and low-level details throughout the transformation. These connections are essential in preserving the fine-grained features that can be lost during the downsampling process in the encoder.

- The skip connections provide shortcuts for the feature maps, allowing the network to merge both local and global information. This design helps the model to better restore fine details, such as edges and textures, which are critical in image restoration tasks like **denoising** and **inpainting**.

Reflection Padding

The network employs **reflection padding** instead of zero padding to minimize border artifacts. Reflection padding extends the image content by mirroring it at the boundaries, which helps in maintaining spatial consistency and prevents the introduction of artificial edges.

3. Implementation Details

3.1. Description Overview

(a) Denoising

- **Data Preparation**: A clean image was loaded, resized, and Gaussian noise was added to generate a noisy version of the image.
- **Input Type**: The network was provided with random noise as input to initialize the restoration process.
- **Network Architecture**: The DIP model used an encoder-decoder structure with skip connections. The encoder reduced spatial dimensions to generate structured features, while the decoder reconstructed the image from these features. Skip connections linked corresponding layers in the encoder and decoder, preserving fine-grained details that might otherwise be lost during the downsampling process.
- **Upsampling Modes**: The decoder used bilinear upsampling, which provided smoother transitions and higher-quality restored images.

(b) Text Inpainting

- **Data Preparation**: The inpainting task involved loading an image along with a binary mask representing missing text regions. The mask was applied to simulate missing parts of the image.
- **Input Type**: Random noise was used as input to initialize the inpainting process.
- **Network Setup**: The DIP model used the same encoder-decoder architecture, and skip connections were included to pass spatial information directly from the encoder to the decoder, helping to preserve details around the masked areas.
- **Upsampling Modes**: The decoder used nearest-neighbor upsampling, which is computationally efficient and provided acceptable visual quality for this task.
- **Loss Computation**: MSE was computed only over the known (masked) regions to guide the network in reconstructing the missing parts of the image.

(c) Hole Filling

- **Data Preparation**: A larger masked region, such as a hole in an image, was created to simulate the challenge of filling in large missing areas.
- **Input Type**: A meshgrid was used as input, allowing better handling of spatial information for larger regions.
- **Input Initialization**: A meshgrid was used for initializing the network input, which allowed better handling of spatial information for larger regions.
- **Network and Training**: The encoder-decoder architecture was used without skip connections for this specific task. Nearest-neighbor upsampling was employed in the decoder to increase spatial dimensions. MSE was computed for the regions covered by the mask to optimize the hole-filling process.
- **Handling Instability**: Loss values fluctuated significantly during training, so the model used a checkpoint mechanism where outputs corresponding to the lowest recorded loss were saved.

(d) Reconstruction

- **Data Preparation**: A grayscale image was used for reconstruction, and Bernoulli noise was applied to simulate random degradation in the image.

- **Input Type**: Random noise was used to initialize the reconstruction process.
- **Network Details**: The DIP architecture consisted of downsampling layers in the encoder to generate key features and upsampling in the decoder to reconstruct the image. Skip connections were used to directly transfer detailed information from the encoder to the decoder, ensuring that both high-level structure and low-level details were preserved during reconstruction.
- **Upsampling Modes**: The decoder utilized bilinear upsampling, which helped produce smoother transitions between pixels, making the reconstructed image more visually coherent.

Common Training Features

Across all tasks, several common training features were utilized:

- **Regularization**: Regularization was achieved by adding noise to the network input at each iteration. This helped to prevent overfitting by encouraging the network to generalize better to the underlying image structure.
- **Checkpoint Mechanism**: A backtracking mechanism was used to handle instability during training. If the PSNR degraded beyond a certain threshold, the model reverted to the previously saved state, ensuring stable and consistent performance.
- **Loss Function**: Mean Squared Error (MSE) was used as the loss function in all tasks to guide the network in minimizing the difference between the output and the target (corrupted or masked) image.

3.2 How to use the program

i. **Dependencies**: Install the required libraries using requirements.yml or set up a conda environment.
ii. **Training:** Modify the image paths and masks if needed, and run the corresponding sections in the script to perform: - Denoising - Inpainting - Hole filling – Reconstruction
iii. **MaskCreator:** Use the maskcreator notebook to generate custom text masks.

3.3 Challenges

- **Text Inpainting Resource Limitation**: For the text inpainting task, we encountered a 'resource exhausted' error while attempting to use the GPU. Switching to a CPU was not feasible due to the extremely long processing time required. Instead, we opted to reduce the input size, which successfully resolved the issue.
- **Hole Filling Checkpoint Issue**: While implementing the checkpoint or backtracking mechanism for the hole filling task as suggested in the paper, we observed that the loss values kept resetting and fluctuated wildly during continuous iterations beyond a certain point. The final output images also appeared blurry. To address this, we decided to continue training despite the loss fluctuations hoping they will converge, but we saved the outputs corresponding to the lowest recorded loss to ensure the best possible quality.

4. Results and Discussion

4.1 Quantitative Results

- **Evaluation Metrics**: Results are measured using metrics like Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM), which assess the quality and fidelity of restored images.

i. ***PSNR (Peak Signal-to-Noise Ratio):*** Measures image quality based on pixel-level differences. Higher values (typically above 30 dB) indicate better quality.

ii. ***SSIM (Structural Similarity Index Measure):*** Assesses image quality based on structural similarities. Values closer to 1 indicate better quality, with >0.95 considered excellent. [4]

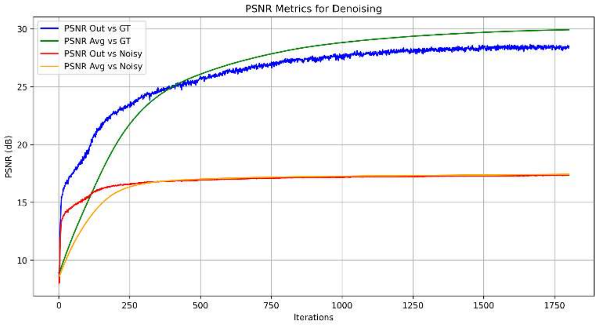
(a) ***Denoising*****:**

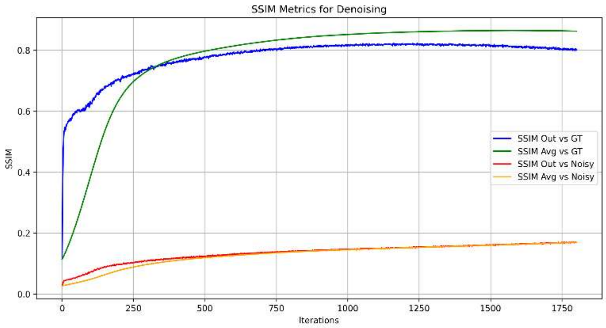
	PSNR	SSIM
Final_vs_GT	26.633632738881374	0.8262966
Avg_vs_GT	29.922273164622798	0.8624169
Final_vs_Noisy	17.159526820560004	0.175373
Avg_vs_Noisy	17.42376855052601	0.16904747

GT (Ground Truth): GT refers to the original, uncorrupted image that serves as the reference for evaluating the quality of the restoration. In image restoration tasks, the ground truth is the ideal image we're trying to recover.

Final: Final refers to the result obtained at the end of the optimization process. In the Deep Image Prior method, this would be the image produced by the network after all iterations of gradient descent have been completed.

AVG (Average): AVG refers to EMA. EMA is Exponential moving Average. Here, we give more weight to the previous output in order to create a smoothed version of the networks' output over iterations to reduce fluctuations and potentially improve stability.

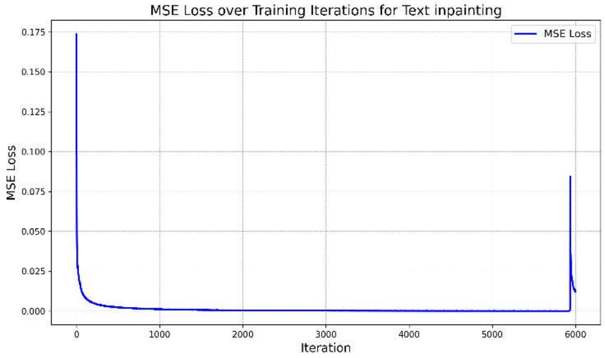


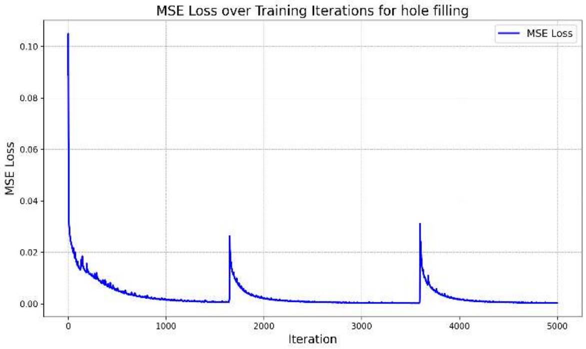


Here from the graph, we can observe that EMA outputs are slightly better than the original output result.

(b) ***Inpainting and Holefilling*****:**

Text Inpainitng	*HoleFilling*
SSIM score: [0.96201557]	SSIM score: [0.9466905]



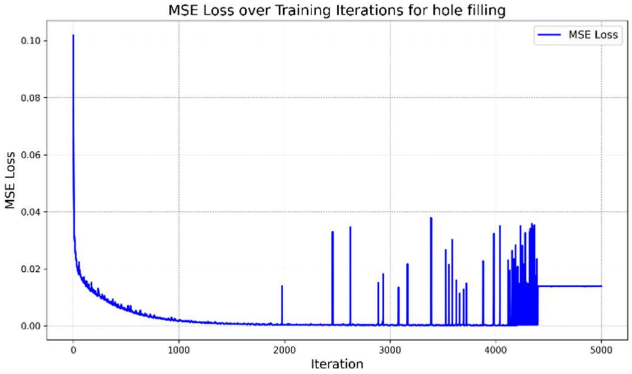


Here, PSNR is not used. This is because PSNR is calculated based on pixel wise values. And for inpainting and Hole filling we have regions of pixel missing. So, calculating PSNR won't give sensible results.

During the training process, we have encountered destability whenever MSE values were low and lead to sudden jump in MSE value and leading to blurry images. The same was mentioned in the paper and they proposed to implement a backtracking mechanism

First, we tried implementing a checkpoint system where we periodically save the networks weights and whenever we observed a sudden increase in MSE we reverted the networks weights to the last checkpoint.

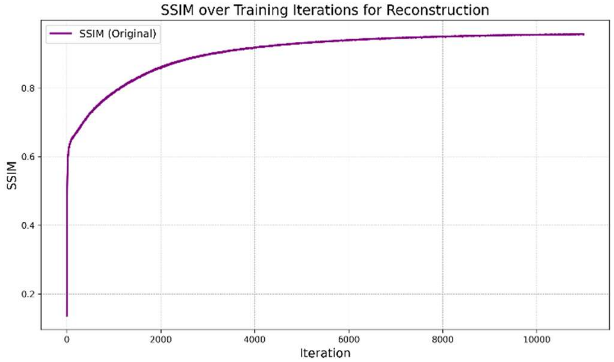
The results were as follows:

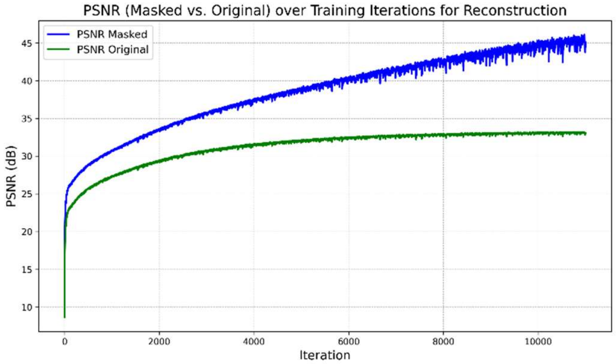


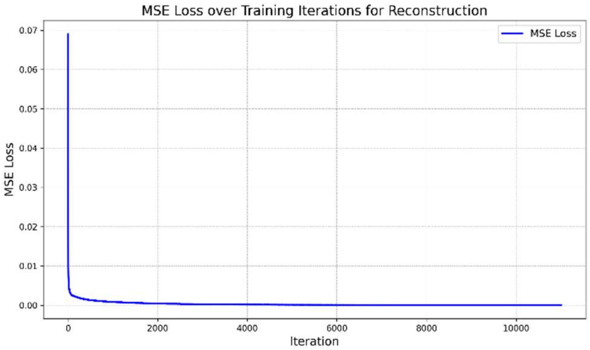
We were unsuccessful with the backtracking implementation. Our network kept fluctuating between getting destabilized as indicated with sudden jumps in MSE values and which would trigger a restoration of the last saved weights. At the end it could not converge. Instead, we skipped the backtracking mechanism and let the training continue even after the destability , but we saved the best performing weights and the corresponding restored image. At the end of the optimization, we use the image with the best metric for evaluation.

(c) **Reconstruction:**

SSIM score: [0.939079]



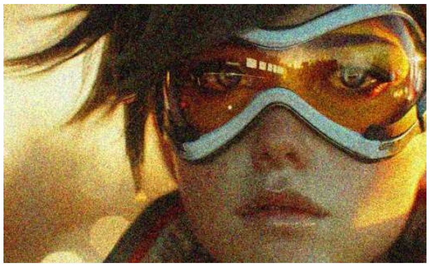


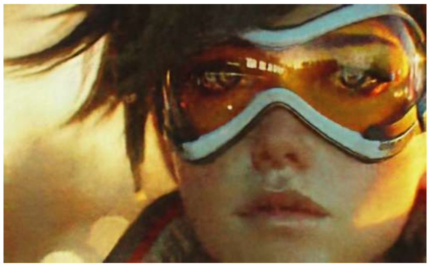


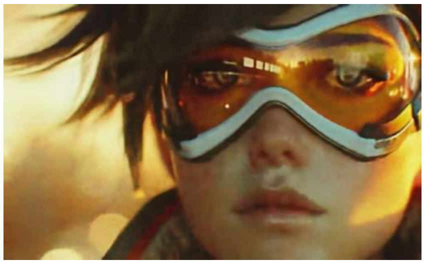
The plots showcase the performance of image reconstruction through SSIM, PSNR, and MSE metrics. SSIM starts low (~0.2) and stabilizes at 0.939, indicating high structural similarity between the reconstructed and original images. The PSNR plot shows significant improvement, with the masked image starting at ~10 dB and the reconstruction reaching ~45 dB, highlighting effective recovery of image quality. The masked MSE loss starts high (~0.07) and steadily decreases to near-zero, reflecting the model's ability to minimize reconstruction errors over iterations.

4.2 Qualitative Results

(a) Denoising:



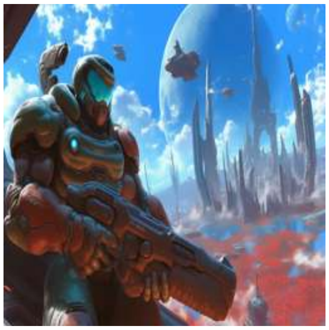




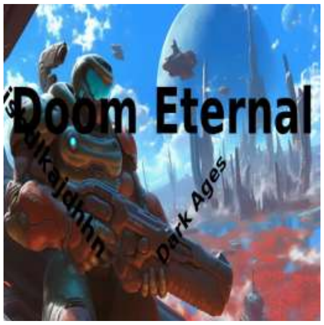
Noisy Image Model Output EMA Output

The noisy image was successfully processed to produce a clean version using DIP. The output shows reduced noise while retaining details, and the EMA (Exponential Moving Average) output further enhances smoothness and visual quality.

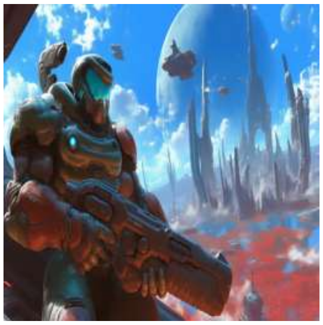
(b) Text Inpainting:



Original Image



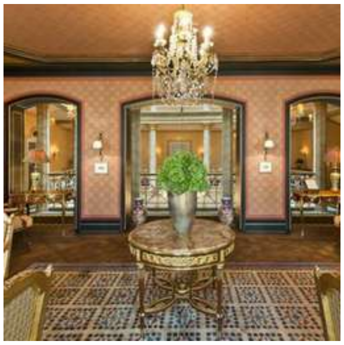
Corrupted Image



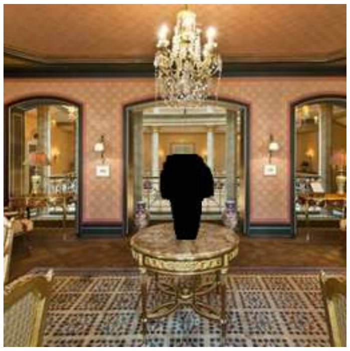
Model Output

The model accurately reconstructed missing text regions in the input image. It used surrounding context to replace the masked areas without introducing significant artifacts. The model outputs align well with the original image, showcasing the model's capability to handle text reconstruction tasks.

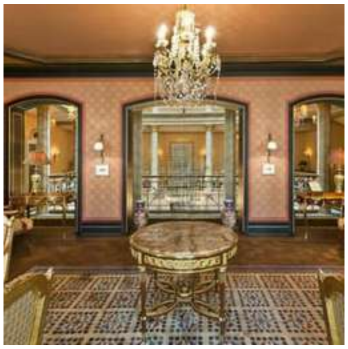
(c) HoleFilling:



Original Image



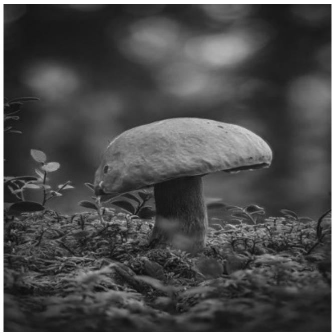
Corrupted Image



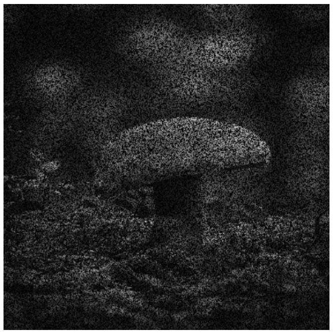
Model Output

The corrupted image with a blacked-out region was restored effectively by the model, filling the missing area with plausible content consistent with the surroundings. The filled region blends seamlessly into the original image, showcasing the model's ability to preserve spatial coherence. This demonstrates strong performance in handling missing data.

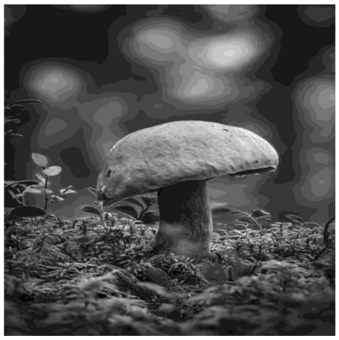
(d) Reconstruction



Original Image



Corrupted Image



Model Output

The reconstruction task involved removing 50% of the pixels from the original image. The model restored the missing regions effectively, producing a visually accurate and high-quality output. Despite losing half of the image data, the model reconstructed the image with remarkable accuracy, indicating its strong generalization capabilities.

5. Conclusion and Future Work

This project successfully implemented the Deep Image Prior (DIP) approach for image restoration, covering tasks such as denoising, inpainting, and reconstruction. The results validate the effectiveness of DIP as an unsupervised learning method, showcasing its potential to handle image restoration tasks without requiring external datasets. While the approach demonstrated notable strengths, certain challenges like computational overhead and blurriness in reconstructed images suggest avenues for further improvement. The insights from this project underline the significance of architectural biases in CNNs for image restoration and pave the way for exploring data-independent neural solutions in related domains.

Strengths:

- *Data Independence:* No reliance on external datasets for training.
- *Versatility:* Effective across various tasks such as denoising, inpainting, and reconstruction.
- *Simplicity:* Minimal pre-processing and intuitive architecture.

Limitations:

- *Resource Intensive:* High computational costs, particularly for large images or complex tasks.
- *Loss Convergence Issues:* Instability during training necessitates additional mechanisms like early stopping and checkpointing.
- *Blurriness in Results:* Lack of finer detail reconstruction for tasks involving large missing regions.

Future Work:

- Experimenting with advanced regularization techniques to stabilize loss fluctuations.
- Investigating alternative architectures or loss functions to enhance fine detail recovery.
- Extending DIP to tasks like super-resolution to evaluate its versatility further.

6. Bibliography

1. The original *Deep Image Prior* paper:
   D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep Image Prior," *arXiv preprint arXiv:1711.10925*, 2018. [Online]. Available: https://arxiv.org/pdf/1711.10925v4
2. L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1–4, pp. 259–268, 1992, doi: 10.1016/0167-2789(92)90242-F
3. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. In: ICLR (2017)
4. Zhou Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," in IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, April 2004

--x--