Alexandria University
Faculty of Computers and Data Science
Computing and Data Science Department

GRADUATION PROJECT II

MRI Acceleration using Untrained Neural Networks

2024/2025

MRI Acceleration using Untrained Neural Networks

Presented by:

Ola Mamdouh Mahdy	20221365645
Verina Michel Asham	20221440977
Marly Magdy Kamal	20221440960
Maria Anwar Beshara	20221380729
Eman Ashraf Abd-Elghany	20221457158
Sandra Adel Faouzy	20221445255
Maria Mansour Yousseff	20221453714
Abdelrahman Ahmed Elsayed	20221372760

Supervised by:

Dr. Amr Amin

I declare that no part of the work referred to in this thesis has been submitted in support of an application for another degree or qualification from this or any other University or Institution.

Table of Contents

Acknowledgements

We would like to express our sincere gratitude to everyone who supported us throughout this project.

We are deeply thankful to Dr. Amr Amin for his continuous guidance, feedback, and encouragement. His expertise greatly contributed to the success of this work.

We would also like to acknowledge the support and collaboration of our fellow students that made the project a more enjoyable and productive experience.

This project has been a rewarding experience, and we are thankful for all the support we received along the way.

Abstract

Convolutional Neural Networks have revolutionized image reconstruction tasks such as denoising, super-resolution, and inpainting. However, their reliance on large, high-quality datasets introduces biases and poses challenges in data-scarce fields. Untrained neural networks, such as Deep Image Prior and Deep Decoder, offer a compelling alternative by leveraging their architecture to reconstruct images directly from degraded observations, eliminating the need for external training data.

This project explores accelerated MRI reconstruction using untrained neural networks in two phases. The first phase focuses on enhancing the performance and efficiency of DIP and Deep Decoder through innovative modifications, achieving faster convergence and higher reconstruction quality. Their performance is critically compared with state-of-the-art trained models, showcasing their ability to deliver competitive results without relying on extensive datasets.

In the second phase, these methods are adapted for real-world accelerated MRI reconstruction, addressing data dependency and generalizability limitations. Rather than replacing trained networks, this work complements them, advocating for a paradigm shift toward architectural innovation over dataset reliance. By advancing untrained neural networks, this research aspires to promote fairness, adaptability, and precision in artificial intelligence, particularly in domains where data acquisition is limited or inconsistent.

List of Figures

List of Tables

Chapter One: Introduction

1 INTRODUCTION

Artificial Intelligence (AI) has rapidly become a cornerstone of technological advancement. Neural networks, in particular, have been driving innovations across sectors as diverse as healthcare and finance. However, beneath these achievements lies a pressing issue that we cannot overlook: AI systems' overwhelming reliance on training data. This dependency introduces vulnerabilities – such as data scarcity, imbalance, overfitting, and bias – that go beyond technical challenges, often leading to real-world consequences.

For instance, consider the alarming reality of bias within AI systems. Algorithms hailed for their impressive accuracy in tasks like melanoma detection, often underperform for patients with darker skin tones. This issue arises not from algorithm design, but from the lack of diversity in training data. Similarly, during the COVID-19 pandemic, pulse oximeters performed poorly for individuals with darker skin tones. These failures highlight the risks of relying on narrow or biased datasets, particularly in healthcare, where such disparities can have life-threatening implications.

This raises a critical question: what if neural networks could achieve high performance without being tied to the constraints of biased or insufficient datasets? This question leads us to explore an emerging paradigm in AI – untrained neural networks. Untrained methods, like Deep Image Prior and Deep Decoder, shift the focus from data dependence to the inherent structure of the network's architecture. These models can generate high-quality outputs, while being independent from the biases and limitations imposed by training data.

This shift is especially important in the context of Magnetic Resonance Imaging (MRI), a critical diagnostic tool in healthcare. MRI scans provide unparalleled clarity in imaging soft tissues, aiding in the diagnosis of conditions such as brain tumors, spinal disorders, and more. However, MRI's full potential is often hindered by long scan times, which not only contribute to patient discomfort but also limit accessibility, especially for critically ill or claustrophobic patients. In emergency settings, time can be the difference between life and death, making faster MRI scans a matter of urgency.

This is where untrained neural networks come into play. Unlike traditional methods, untrained neural networks provide a lightweight, adaptable, and data-independent solution. They offer a promising way to address the challenges of accelerating MRI scans – enabling quicker, higher-quality scans without the need for extensive training data. This innovation doesn't just solve a technical problem; it addresses a deeper human need: accessible, efficient healthcare. Imagine MRI technology that reduces patient stress, increases diagnostic capacity, and expands access even in resource-limited environments.

The motivation behind our research is rooted in this vision. Applying untrained neural networks to MRI reconstruction is not just about improving technology; it is about promoting fairness and inclusivity in medical imaging. This work aims to contribute to a more just healthcare system, where technology serves humanity without perpetuating systemic biases or disparities. Through this exploration, we hope to bring AI closer to a future where it serves all people, regardless of skin color, socioeconomic status, or geographic location.

Chapter Two: MRI & Accelerated MRI Challenges

2 MRI &ACCELERATED MRI CHALLENGES

2.1 MAGNETIC RESONANCE IMAGING

2.1.1 Introduction to MRI

Magnetic Resonance Imaging (MRI) is a medical imaging technique that produces detailed images of internal body structures, especially soft tissues like the brain, muscles, and organs. MRI is favored for being non-invasive, and not using ionizing radiation (unlike X-rays or CT scans). Instead, MRI uses magnetic fields and radiofrequency (RF) signals.

The key principle behind MRI is **nuclear magnetic resonance (NMR)**, where protons in a magnetic field absorb and emit RF energy in a measurable way. By manipulating these signals and analyzing their spatial distribution, MRI reconstructs high-resolution 2D or 3D images of tissues.

2.1.1.1 Nuclear Magnetic Resonance (NMR)

Nuclear Magnetic Resonance (NMR) is the fundamental physical principle behind MRI. It refers to the behavior of certain atomic nuclei, like the hydrogen atoms in the human body, when placed in a magnetic field.

MRI works by applying a strong magnetic field to the hydrogen atoms, causing their magnetic moments to align with the field. Then, a RF signal is used to excite the atoms, causing them to flip away from this alignment. When the signal stops, the hydrogen atoms begin to relax back to their original alignment, emitting RF signals in the process. This process is known as **Relaxation**. These emitted signals are detected by the coils in the MRI device and then used to reconstruct an image.

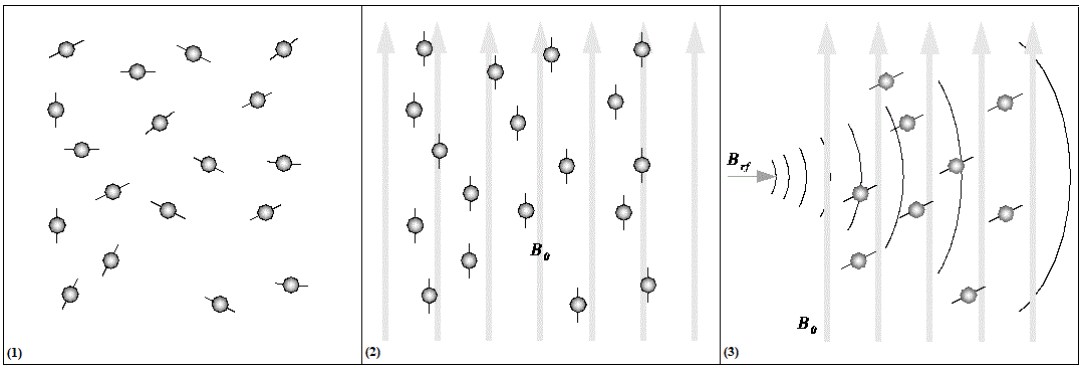


Figure 2.1 Nuclear Magnetic Resonance Visualization

2.1.2 Relaxation and Image Contrast

2.1.2.1 T1 and T2 Relaxation

After the hydrogen atoms in the body are excited by an RF pulse, they gradually return to their original state in a process called relaxation. This happens in two main ways:

- **T1 Recovery** (longitudinal magnetization)is the process by which the hydrogen spins realign with the main magnetic field. It reflects how quickly the longitudinal magnetization recovers. Different tissues have different T1 times, which affects the brightness of tissues in T1-weighted images.
- **T2 Decay** (transverse magnetization) describes how quickly the hydrogen spins lose synchronization. This causes the MR signal to decay. Tissues with slower T2 decay appear brighter in T2-weighted images.

Both relaxation times are key to determining image contrast in MRI.

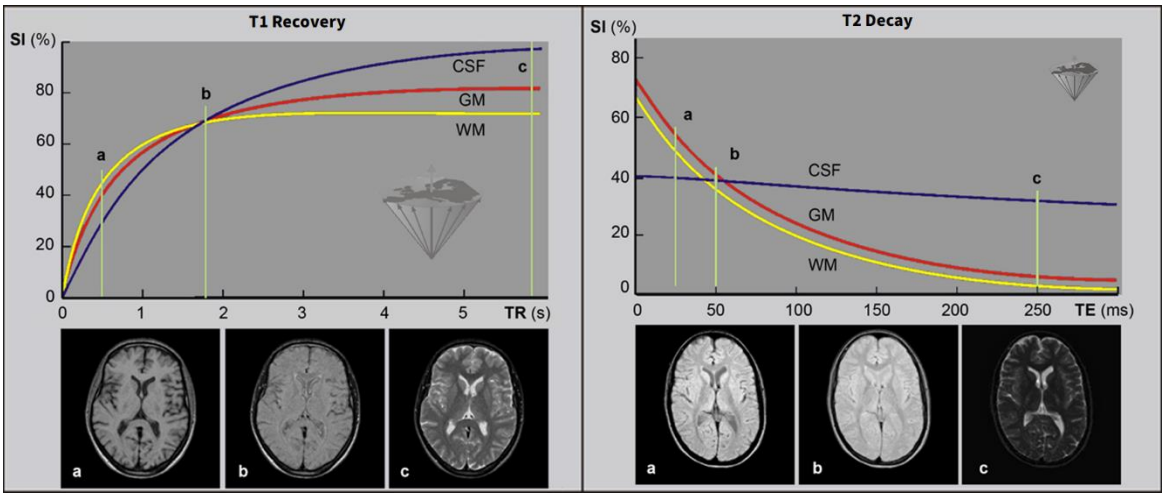


Figure 2.2 T1 Recovery and T2 Decay

2.1.2.2 Image Contrast: TE, TR, and Weighting

In MRI, image contrast is controlled primarily through two timing parameters:

- **TR (Repetition Time):** Time between successive RF pulses
- **TE (Echo Time):** Time between RF excitation and signal measurement

These parameters influence how much T1 and T2 relaxation effects are captured in the signal, and thus determine the "weighting" of the image.

By adjusting TE and TR, different tissue properties can be highlighted. For example:

- **T1-Weighted Images:**
  - Uses short TR and short TE.
  - Highlights anatomy: Fat appears bright, fluid appears dark.
  - Example use: Structural imaging, white/gray matter contrast.
- **T2-Weighted Images:**
  - Uses long TR and long TE.
  - Highlights pathology: Fluid appears bright, fat is intermediate.
  - Example use: Detecting edema, inflammation, or tumors.
- **FLAIR (Fluid Attenuated Inversion Recovery):**
  - Uses very long TR and long TE, with an inversion pulse to suppress fluid (CSF).
  - Similar to T2, but fluid signals are dark, enhancing contrast for nearby lesions.
  - Example use: Detecting legions such as those caused by strokes or sclerosis.

- **Proton Density (PD)-Weighted Images:**
    - Uses long TR and short TE.
    - Contrast is based on the concentration of hydrogen protons in each tissue.
    - Fat and fluid both appear relatively bright.
    - Example use: Joint imaging and subtle tissue contrast.

2.1.2.3 Fat and Fluid Suppression

In certain MRI applications, high signal from fat or fluid can obscure important details. Suppression techniques selectively reduce these signals to improve image contrast:

- **Fat suppression** uses frequency-selective pulses or inversion recovery to reduce the bright signal from fat. Common in joint imaging, breast MRI, and abdominal scans.
- **FLAIR** is a form of fluid suppression, designed to null cerebrospinal fluid (CSF) signal in T2-weighted brain images, making nearby abnormalities more visible.

These techniques are often combined with T1, T2, or PD-weighted imaging, depending on the clinical need.

2.1.3 K-Space and MRI Data

The signal data emitted during relaxation is collected by the MRI scanner. Then, **gradient encoding** is applied to localize where the signal is coming from within the body. This step is essential, because the raw signals alone cannot form an image.

There are two main types of gradient encoding:

- **Frequency encoding** is applied during signal readout. It creates a position-dependent frequency shift along one direction (usually the x-axis), allowing the scanner to distinguish signal contributions based on frequency.
- **Phase encoding** is applied before readout along a second direction (usually the y-axis). It causes spins in different locations to accumulate different phase shifts, which are used to separate signals across that dimension.

By repeating this process, the scanner fills the **k-space** line by line.

K-space is a complex-values matrix that represents the image in the frequence domain. The center of k-space holds low-frequency data (overall image contrast and structure), while the edges contain high-frequency data (fine details and edges).

The spatial image is constructed from the k-space data by applying the **Inverse Fourier Transform**, to convert the data from the frequency space to the image space. Since the acquired k-space data is complex-valued, the reconstructed image after applying the inverse Fourier transform is also complex. The final MRI image used for diagnosis is typically the magnitude of the reconstructed complex image.

2.1.4 MRI Hardware and Coil Configuration

The main hardware components of an MRI scanner are:

- **Main magnet:** generates the strong static field B_0.
- **Gradient coils:** produce controlled magnetic field variations for spatial encoding.
- **RF coils:** responsible for transmitting the RF pulses that excite the hydrogen atoms and receiving the emitted signals.

Most modern MRI systems use separate transmit and receive RF coils, or even multiple receive-only coils. Using multiple coils improves signal sensitivity, allows for parallel imaging techniques, and helps reduce scan times.

2.1.4.1 Single-Coil Systems

Traditional MRI systems use a single receive coil to collect signals from the imaging region. These setups provide high-quality images but can be limited in efficiency when high-resolution or fast imaging is required.

2.1.4.2 Multi-Coil Systems

Modern scanners often use multi-coil (array) systems, which consist of several smaller, spatially distributed receive coils. Each coil in the array picks up the MR signal from a specific region of the body, providing a stronger and more localized signal.

In multi-coil setups, each coil has a unique sensitivity profile, also called a **sensitivity map** – a spatial pattern describing how sensitive that coil is to signals from different locations. These sensitivity maps are crucial for reconstructing the final image,

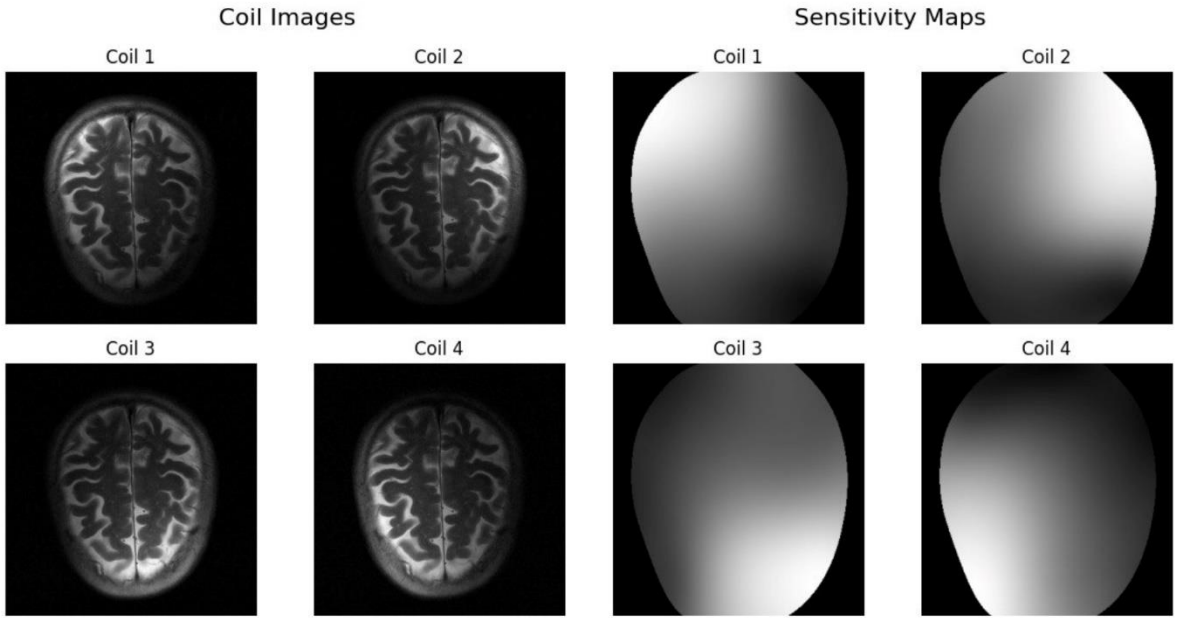


Figure 2.3 Example of Individual Coil Images and Sensitivity Maps in a 4-Coil System

Several methods exist for estimating coil sensitivity maps, like B1 mapping and ESPIRiT, with ESPIRiT being the most popular.

ESPIRiT (Eigenvalue-based Self-consistent Parallel Imaging Reconstruction Technique) operates by extracting calibration data from the fully sampled central k-space region and solving an eigenvalue problem to identify self-consistent sensitivity maps across coils.

2.2 ACCELERATED MRI

2.2.1 Accelerated MRI using Machine Learning

Magnetic Resonance Imaging (MRI) is a widely used medical imaging technique that provides high-resolution images of soft tissues without exposing patients to ionizing radiation. However, conventional MRI scans are time-consuming, often requiring several minutes to hours for data acquisition, leading to patient discomfort and motion artifacts. Accelerated MRI aims to reduce scan times while preserving image quality, making it a critical area of research in medical imaging.

Machine learning, particularly deep learning, has emerged as a powerful tool for accelerating MRI by reconstructing high-quality images from undersampled data. Traditional acceleration methods, such as parallel imaging and compressed sensing, rely on mathematical models and handcrafted priors to reconstruct images from fewer measurements. In contrast, machine learning-based approaches leverage data-driven models to learn complex mappings between undersampled acquisitions and fully sampled images.

2.2.2 FastMRI Dataset Description

The imaging dataset utilized in this work was provided by NYU Langone and consisted of de-identified raw k-space data organized into several sub-dataset groups. The curation of these data was conducted as part of an Institutional Review Board (IRB) approved study.

Sub-Dataset Groups

2.2.2.1 Knee MRI Data

The knee MRI dataset comprised two primary components:

Raw Dataset:
This component included more than 1,500 fully sampled knee MRIs acquired on 3 Tesla and 1.5 Tesla scanners. The raw data consisted of coronal proton density (PD)-weighted images with and without fat suppression.

DICOM Dataset:
In addition to the raw data, DICOM images from 10,000 clinical knee MRIs were provided. The DICOM dataset encompassed multiple imaging contrasts, including coronal PD-weighted images (with and without fat suppression), axial PD-weighted images with fat suppression, sagittal PD-weighted images, and sagittal T2-weighted images with fat suppression.

2.2.2.2 Brain MRI Data

The brain MRI dataset consisted of 6,970 fully sampled brain MRIs acquired on both 3 Tesla and 1.5 Tesla scanners. The raw dataset included axial images acquired using different sequences:

T1-Weighted:
Some T1 acquisitions involved the administration of a contrast agent.

T2-Weighted:
These images provided complementary anatomical information.

FLAIR (Fluid-Attenuated Inversion Recovery):
FLAIR images were included to highlight specific pathologies.

2.2.2.3 Prostate MRI Data

The prostate MRI dataset included data from 312 prostate MRI examinations, all acquired on 3 Tesla magnets. For each exam, the raw dataset comprised:

Axial T2-Weighted Images

Axial Diffusion-Weighted Images

2.2.2.4 Breast MRI Data

The breast MRI dataset consisted of 300 clinical breast MRI examinations, all obtained on 3 Tesla scanners. Each examination in the raw dataset featured:

Axial Dynamic Contrast-Enhanced (DCE) MRI:
The images were acquired using a 3D GRASP sequence.

2.2.3 The 2019 fastMRI Challenge

Realistic Data Acquisition

To faithfully simulate clinical conditions, the challenge utilized actual raw k-space data obtained directly from MRI scanners. The dataset comprised 1,500 consecutive clinical proton-density-weighted MRI examinations of the knee, acquired in the coronal plane. Two types of acquisitions were included: one with frequency-selective fat saturation (COR PD FS) and one without (COR PD). These acquisitions not only offered diverse image contrasts but also varied in signal-to-noise ratio (SNR) by approximately a factor of four. Data collection was performed on multiple clinical systems, including three 3T systems (Siemens Magnetom Skyra, Prisma, and Biograph-mMR) and one 1.5T system (Siemens Magnetom Aera), using standard multi-channel receive coils.

The dataset used

Knee MRI: Data from more than 1,500 fully sampled knee MRIs obtained on 3 and 1.5 Tesla magnets and DICOM images from 10,000 clinical knee MRIs also obtained at 3 or 1.5 Tesla.

Controlled Undersampling and Ground Truth Formation

In this challenge, undersampling was performed retrospectively from fully sampled acquisitions. This approach ensured that the ground truth for each scan was identical to what would have been acquired in an ideal, fully sampled scenario, thereby eliminating potential confounding factors such as variations in patient motion or scanner calibration.

The undersampling scheme was predefined to isolate the image reconstruction problem from the sampling trajectory design. A one-dimensional pseudo-random undersampling pattern was applied in the phase encoding direction, with a fully sampled central k-space region, following the compressed sensing framework.

Evaluation Metrics and Final Ranking

The assessment of image quality in the fastMRI Challenge was multi-faceted. Initially, quantitative metrics such as the Structural Similarity Index (SSIM), normalized mean square error (NMSE), and peak signal-to-noise ratio (PSNR) were employed to facilitate objective comparisons. However, recognizing that these metrics might not fully capture the clinical diagnostic value of the images, the final ranking was determined by an expert panel of musculoskeletal radiologists. This expert evaluation focused on the depiction of anatomical details and pathology, ensuring that the winning solutions were not only mathematically robust but also clinically meaningful.

Flexibility in Data Usage

In order to promote innovation, the challenge did not restrict participants to using only the provided fastMRI dataset. Researchers were allowed to supplement their training data with additional external datasets, provided that such usage was explicitly reported at the time of submission. This policy was designed to level the playing field while encouraging the development of versatile and generalizable reconstruction algorithms.

Experimental Setup and Challenge Tracks

The 2019 fastMRI Challenge was structured to assess advanced MRI reconstruction methods under varied conditions by using multiple submission tracks. Two acceleration factor scenarios were implemented: one at a moderate acceleration of R = 4, which balanced reconstruction difficulty with clinically acceptable image quality based on prior experience, and a high acceleration scenario at R = 8, designed to push methods beyond typical limits to reveal failure modes despite likely suboptimal clinical results. Additionally, the challenge featured two data channel tracks. The multi-coil track (with both 4× and 8× acceleration) provided true multi-channel raw k-space data from clinical scanners, reflecting realistic imaging conditions and targeting experts in MR reconstruction. In contrast, the single-coil track (limited to 4× acceleration) combined multi-channel data into a single channel while preserving its complex nature, thereby simplifying initial processing for researchers in machine learning, computer vision, and image processing—although this simplification increased the reconstruction challenge due to reduced data redundancy.

2.2.4 Enhancements in the 2020 fastMRI Challenge

The 2020 fastMRI Challenge was designed to build upon the experiences of its 2019 predecessor, incorporating key modifications aimed at increasing clinical relevance, improving generalization, and advancing reconstruction performance

2.2.4.1 Rationale for Modifications

Based on extensive feedback and performance analyses from the 2019 fastMRI Challenge, several shortcomings were identified—most notably, issues related to clinical applicability and model generalizability. The following modifications were introduced for the 2020 challenge:

Clinical Relevance:
Transitioning from knee imaging to brain imaging was motivated by the fact that the brain is the most frequently imaged organ in clinical practice. This shift not only increases the impact of the challenge but also better aligns the task with common diagnostic scenarios encountered in hospitals.

Focused Pathology Evaluation:
Whereas 2019 emphasized overall image quality, the 2020 challenge refocused evaluation on the depiction of pathology. By prioritizing the accurate representation of clinically significant features, the challenge ensured that the reconstruction methods would have direct diagnostic utility.

Model Generalization:
A recurring theme in the feedback from 2019 was the limited generalizability of reconstruction models. To address this, the 2020 challenge introduced a "**Transfer**" track, wherein participants were required to test their models on multi-vendor data. This track was specifically designed to assess the robustness of algorithms when applied to images from scanners different from those used in training.

Enhanced Data and Evaluation Strategy:
To further bolster clinical relevance, the single-coil track—despite its popularity in 2019—was removed, **favoring a pure multi-coil approach that mirrors current clinical practices**. Additionally, **the use of pseudo-equispaced subsampling masks replaced the previous random masks.** This adjustment ensured that the undersampling patterns were more in line with those used in vendor sequences, thereby facilitating a smoother transition from research to clinical deployment.

Upgraded Baseline Model:
In 2019, the baseline model was based on a U-Net architecture. However, winning entries predominantly utilized variational network or cascading models. Reflecting this evolution, the 2020 challenge adopted an End-to-End Variational Network as the baseline, providing participants with a stronger foundation from which to develop their methods.

Implementation of the 2020 Challenge Tracks

The 2020 challenge maintained a two-stage evaluation process. Initially, submissions were ranked using the Structural Similarity Index (SSIM) to identify the top three candidates. These finalists were subsequently evaluated by an expert panel of radiologists to ensure that the final results were not only quantitatively sound but also clinically relevant.

The challenge was structured into three key tracks:

Multi-Coil 4X and 8X Tracks:
These tracks continued from 2019, with a moderate acceleration (R = 4) expected to yield clinically acceptable reconstructions, and a high acceleration (R = 8) intended to probe the limits of current methods and reveal their failure modes.

Transfer 4X Track:
Introduced exclusively in 2020, the Transfer track required participants to run their pre-trained models on data obtained from vendors not included in the training set. This track emphasized the generalization capability of the models and provided a more challenging and realistic test of their robustness across different clinical scenarios.

Data Set Expansion and Clinical Considerations
For the 2020 challenge, the dataset was significantly expanded to include brain MRI scans—a departure from the knee MRI data used in 2019. The new dataset encompassed 6,970 scans collected at NYU Langone Health, spanning various sequences (T1, T1 post-contrast, T2, and FLAIR) and field strengths (1.5T and 3T). Additionally, 565 scans were withheld for challenge evaluation, and the Transfer track was further enriched with 329 non-Siemens scans from GE and Philips systems. This expansion not only increased the scale of available data but also introduced variability that is reflective of real-world clinical imaging.

Chapter Three: Deep Learning

3 DEEP LEARNING

Deep learning, a subset of machine learning, employs artificial neural networks with multiple layers (deep neural networks) to learn hierarchical representations directly from data. Unlike traditional methods, deep learning automates the feature extraction process, enabling more effective solutions to complex problems across various domains like healthcare, robotics, and cybersecurity.

Neural networks, the foundational models of deep learning, consist of interconnected layers of neurons that process data hierarchically. While inspired by the structure of the human brain, these models are not biological replicas. The power of deep learning lies in its ability to learn multiple levels of abstraction, automating feature engineering and solving previously intractable problems.

Deep learning models encompass a variety of architectures tailored for specific tasks, such as:

- Artificial Neural Networks
- Convolutional Neural Networks
- Recurrent Neural Networks
- Long Short-Term Memory Networks
- Generative Adversarial Networks
- Transformers

3.1 CONVOLUTIONAL NEURAL NETWORKS

CNNs specialize in processing grid-like data, such as images, by leveraging spatial hierarchies through convolutional, pooling, and fully connected layers. These architectures excel at capturing patterns like edges, textures, and shapes. By using sparse connectivity and shared weights, CNNs are computationally efficient and effective in tasks like image classification.

3.1.1 The Convolution Operation

Convolution combines two functions to produce a third, used in CNNs for feature extraction. For instance, smoothing noisy sensor data involves a weighted average calculated via convolution. In CNNs, filters (or kernels) slide over input data, learning relevant features during training.

3.1.2 Convolution Layers

These layers apply learnable kernels to input data, generating activation maps that highlight detected features. Hyperparameters like kernel depth, stride, and padding control the layer's output. Proper configuration ensures efficient pattern recognition while managing computational complexity.

3.1.3 Pooling Layers

Pooling layers reduce spatial dimensions by summarizing feature values, decreasing computational cost, and introducing translation invariance. Common pooling methods include:

- **Max Pooling**: Captures the maximum value within a region.
- **Average Pooling**: Computes the mean value of a region.

3.1.4 Fully Connected Layers

Fully connected layers integrate extracted features and produce final predictions. Each neuron connects to all neurons in the previous layer, enabling the model to learn complex relationships. These layers often employ SoftMax for multi-class classification.

3.1.5 Activations Functions

Activation functions introduce nonlinearity, enabling networks to model complex relationships. Common functions include:

- **ReLU (Rectified Linear Unit)**: Efficient and avoids vanishing gradients but may cause "dead neurons."
- **Sigmoid**: Maps outputs to [0, 1], suitable for binary classification but prone to vanishing gradients.
- **Tanh**: Maps outputs to [-1, 1], offering zero-centered activations but shares sigmoid's limitations.
- **SoftMax**: Converts outputs into probability distributions for multi-class tasks.

3.1.6 Normalization Techniques

3.1.6.1 Batch normalization

Batch normalization (BN) standardizes the activations of each layer across a mini-batch of data, improving convergence speed and stability. It operates by normalizing inputs to have zero mean and unit variance, followed by a learnable scaling and shift. This technique helps to:

- Reduce internal covariate shift.
- Improve gradient flow through the network.
- Act as a form of regularization, reducing overfitting.

The batch normalization operation for a mini-batch x with mean μ and variance σ^2 is defined as:

$$\dot{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

where ε is a small constant for numerical stability.

The normalized output is scaled and shifted using learnable parameters γ and β:

$$y = \gamma\dot{x} + \beta$$

3.1.6.2 Channel-wise Normalization

Channel-wise normalization is often used in CNN to normalize activations along the channel dimension. Unlike batch normalization, which processes data across a mini-batch, channel-wise normalization focuses on spatial dimensions. This method is especially useful in tasks involving image data.

For an activation map X with shape (H, W, C) channel-wise normalization ensures that each channel has consistent statistics (mean and variance), enhancing training dynamics and feature learning. This is a common component in advanced architectures like **Instance Normalization** and **Group Normalization**.

3.2 CNN ARCHITECTURES

3.2.1 Encoder-Decoder

In deep learning, Encoder-Decoder architectures, otherwise known as sequence-to-sequence models, are a powerful class of neural networks designed to handle tasks related to variable-length input and output sequences. Encoder-decoder models are used to handle sequential data, especially mapping input sequences to output sequences of different lengths, such as neural machine translation and image captioning.

3.2.1.1 Core Components

Encoder-decoder architecture includes two major components: the encoder and the decoder. Both the encoder and decoder can act as separate, fully connected neural networks. These could include RNNs, CNNs, and transformer models. An encoder-decoder architecture typically consists of multiple encoders and multiple decoders.

- **Encoder:** It processes the input data and extracts a compressed representation, also known as context vector or latent representation.
  In image processing, the encoder typically consists of a series of convolutional layers with decreasing spatial dimensions and increasing the depths of feature map. This process gradually extracts higher-level features from the input image.
- **Decoder:** It reconstructs the original input or generates a desired output based on the context vector provided by the encoder.
  The decoder typically consists of a series of convolutional layers with increasing spatial dimensions and reduced depths of the feature maps. This process gradually reconstructs the output image from the compressed representation.

3.2.1.2 Architectures

Hourglass Architecture

The Hourglass architecture is a specialized type of encoder-decoder architecture that has found extensive use in computer vision tasks, specifically in enabling image restoration tasks. The hourglass architecture consists of a symmetric shape akin to the shape of an hourglass.

The input of the network is an image with dimensions $C \times W \times H$ (Channels, Height, Width) has the same spatial resolution as the output of the network.

- **Encoding Path**
  Downsampling is achieved via strides and convolutional via max pooling and downsampling with Lanczos kernel. The input image acts as the starting point. It passes into the encoder for feature extraction.
  Phase 1: Extracting the local patterns (edges and textures) by applying a convolutional layer with F1 filters, Stride is equal to 1. This make the number of feature channels increases to F1 and the output dimensions is $(H \times W \times F1)$.

Downsampling using strided convolution: By applying Stride is equal to 2, the spatial resolution is reduced by half, and the output dimensions becomes $(H/2 \times W/2 \times F1)$

Phase 2: Extracting more complex patterns (edges combined into shapes) by applying a convolutional layer with F2 filters, Stride is equal to 1. This make the number of feature channels increases to F2 and the output dimensions is $(H/2 \times W/2 \times F2)$.
Downsampling using strided convolution: By applying Stride is equal to 2, the spatial resolution is reduced by quarter, and the output dimensions becomes $(H/4 \times W/4 \times F2)$

The process repeats with extra convolutional and downsampling layers, that reduce the spatial resolution (H, W) while increasing the number of feature channels (C). The output dimensions of the Encoder are $(H/16 \times W/16 \times F4)$

- **Bottleneck Layer (Latent Space)**
  The tightest part of the architecture. Represents the most compressed representation of the input image where the spatial resolution is minimal, and the feature channels are maximal. Captures global features and discards noise or redundant information. This compact representation serves as the foundation for the upsampling.

- **Decoding Path**
  Upsampling is done through either the Bilinear upsampling technique or the Nearest neighbor technique. It progressively upsamples the feature maps from the bottleneck layer to reconstruct the original image dimensions.

  Bilinear Upsampling Technique: A standard approach in which each pixel in the upsampled image is given a value based on a weighted average of the four nearest neighbors within the lower-resolution feature map. The result is smoother images.

  Nearest neighbor Upsampling Technique: Assigns each pixel within the upsampled image the value of the nearest pixel within the lower-resolution feature map.
  The output dimensions of the upsampling step is $(H/8 \times W/8 \times F4)$

  Long Skip Connections: After each upsampling step, the upsampled feature maps are then combined with the corresponding feature maps from the encoder path. Combines the global (shape of the image) and local (small details in the image) features fine-grained details can be recovered in the reconstruction, and the output dimensions are $(H/8 \times W/8 \times (F4 + F3))$

  Convolutional Layer: Applied to the combined feature maps. Reduces feature channels. Preserves spatial details and refines high-level features, and the output dimensions are $(H/8 \times W/8 \times F3)$.

  The process repeats with upsampling steps and combining with encoder features. Upsampling the spatial resolution to (H, W) while decreasing the number of feature channels to (C). The output dimensions of the Decoder are $(H \times W \times C)$

- **The Output**
  A reconstructed image has the same dimensions of the input image $(C \times W \times H)$.

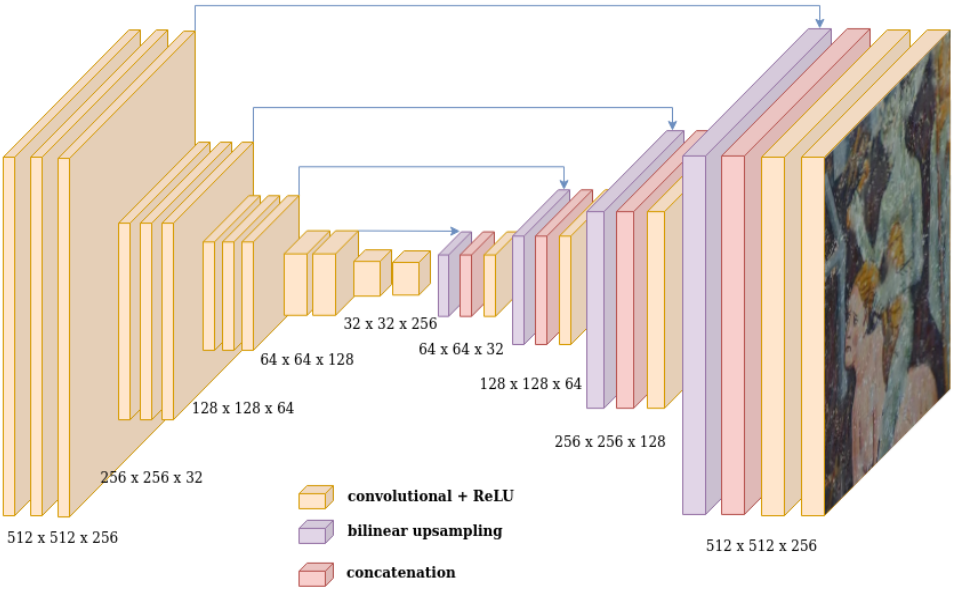


Figure 3.1 The architecture of the "Hourglass" encoder-decoder

3.2.2 Skip Connections

Skip connections are extra connections between nodes in different layers of a neural network that skip one or more layers of nonlinear processing. The introduction of skip (or residual) connections has substantially improved the training of very deep neural networks.

In deep neural networks, especially Convolutional Neural Networks (CNNs), information flows through multiple layers. As the network deepens, the gradients (the signals that guide the network's learning) can become smaller and smaller, eventually vanishing. This makes it difficult for the network to learn from earlier layers and causes information degradation.

Skip connections address these challenges by establishing direct routes for gradients to propagate backward through the network, thereby enhancing training stability and convergence. And for information degradation, skip connections retain and reuse unaltered information from earlier layers. It enhances the network's ability to preserve fine details, especially in tasks requiring high-resolution outputs.

3.2.2.1 Types of Skip Connections

- **Long Skip Connections:** These skip connections have been shown to help recover the full spatial resolution at the network output, making fully convolutional methods. These connections bridge the gap between corresponding layers in the encoder and decoder parts of the network. Usually, layers of similar spatial dimensions are connected, which allows the decoder to recover complex information that has been captured by the encoder. This is realized in the U-Net architecture.
- **Short Skip Connections:** They connect the output of one layer directly to the input of another layer that is only a few layers ahead. They are used along with consecutive convolutional layers that do not change the input dimension. They address the vanishing/exploding gradient problem by providing an alternative path for gradients to flow. This allows deeper networks to be trained more effectively. It is used in ResNet architecture.

3.2.2.2 Architecture

Skip connections are implemented as direct pathways that transfer feature maps from earlier layers to later ones. They operate in two fundamental ways:

Addition: Element-wise addition of feature maps, assuming spatial dims are the same. The output of the skipped layers is added to the output of the layer that receives the skip connection. This summation combines the information from the direct path with the information processed by the skipped layers.
Concatenation: Stacks feature maps along the channel dimension, requiring dimensionality alignment through convolutions if necessary. This architecture heavily uses feature concatenation so as to ensure maximum information flow between layers in the network. This is achieved by connecting via concatenation all layers directly with each other.

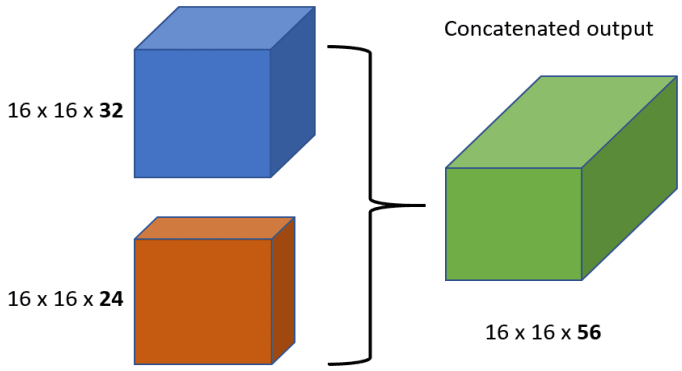


Figure 3.2 The Concatenation method in Skip Connections

Chapter Four: Untrained Methods for Image Restoration

4 UNTRAINED METHODS FOR IMAGE RESTORATION

4.1 DEEP IMAGE PRIOR

4.1.1 Introduction

Deep convolutional networks are widely used in image restoration due to their ability to learn image priors from large datasets. However, the Deep Image Prior (DIP), introduced by Ulyanov et al. (2017), takes a different approach. DIP uses the structure of an untrained neural network as an implicit prior for natural images. By fitting the network to a single degraded image, it leverages the network's architecture alone, without external training data, to solve tasks like denoising, super-resolution, and inpainting.

4.1.1.1 Image priors

Image priors are essential for solving inverse problems like denoising, inpainting, and super-resolution, as they guide restoration by enforcing natural image constraints. Priors help prevent unrealistic outcomes and recover lost details. They fall into three categories:

- Explicit Priors: Handcrafted assumptions (e.g., smoothness, sparsity) defined mathematically.
- Learned Priors: Derived from training on large image datasets using machine learning models.
- Implicit Priors: Not explicitly defined; embedded in network architectures, such as in DIP, which relies on the network's structure to bias reconstructions toward natural images without training.

In the DIP framework, the network's architecture is not merely a tool for representing data but **serves as a handcrafted prior** that guides the optimization process. Specifically, DIP employs U-Net-like “hourglass” architectures with skip connections. These architectures are particularly well-suited for capturing the self-similar and hierarchical patterns often present in natural images. Skip connections also help to combine features at different scales, which benefits the preservation of detailed information and the reconstruction of high-fidelity images.

The idea of using the architecture of a network as a prior enforces the generative capacity of convolutional networks. By fitting the network to a degraded image, its weights effectively parametrize the restored image. Apart from providing a competitive alternative to the conventional learning-based approaches, this also leverages the deep learning and handcrafted priors to bridge the gap, while showing the indispensable capability of the network architecture in dealing with complicated image restoration tasks.

In other words, the network structure is a strong prior that influences the optimization path and leads it toward solutions that are meaningful visually.

4.1.1.2 DIP as an implicit prior

Traditional methods use explicit prior *R(x)* (Regularizers) to enforce the properties of natural images. In contrast, DIP replaces the explicit prior with an **implicit prior** encoded in the structure of the network (hourglass encoder-decoder architecture).

In DIP, the image $\boldsymbol{x}$ is represented or described in terms of a function controlled by a set of parameters $\boldsymbol{\theta}$. Instead of representing an image directly as a grid of pixel values, it is **parameterized** by the network $\boldsymbol{f_\theta}$, where the parameters $\boldsymbol{\theta}$ control the network's behavior and output. This approach shifts the image restoring problem from working directly with pixel values to **optimizing the parameters θ** as:

$$\boldsymbol{x = f_\theta(z)}$$

where:

$\boldsymbol{f_\theta}$**:** is the deep generator network with parameters θ.

z: is the fixed random input tensor.

Thus, the neural network structure acts as an implicit prior, that biases it toward generating images that look natural as a significant amount of information about the image distribution is contained in the structure of the network even without performing any training of the model parameters.

Now, the goal of the optimization problem (Energy minimization) is to find the best parameters **θ,** and the problem is reformulated as:

$$\boldsymbol{\theta^* = argmin\ E(f_\theta(z); x_o)}$$

The restored image is then obtained as: $x^* = f_{\theta^*}(z)$

4.1.1.3 DIP Architecture and the Optimization Process

The choice of architecture does have an impact on the results. The DIP uses a U-Net-like **"hourglass"** architecture with skip connections, where *z* and *x* have the same spatial dimensions and the network has several millions of parameters. As a high-capacity network, it has the ability to fit any image including noise, given enough iterations.

Network Input: The network takes the corrupted image $\boldsymbol{x_o} \in R^{3\times H\times W}$ as an input, and a fixed random initialized tensor $\boldsymbol{z} \in R^{C'\times H'\times W'}$, which can be considered as random noise initialized from a uniform distribution. This random noise allows the network to start the optimization process without any bias from the input.

The optimization process: starts with initializing random parameters θ. The network optimizes θ solely to minimize the loss function based on the corrupted image $\boldsymbol{x_o}$.
The (local) minimizer $\boldsymbol{\theta^*}$ is obtained using an optimizer such as gradient descent. At every iteration the weights θ are mapped to an image $\boldsymbol{x = f_\theta(z)}$. The image *x* is used to compute the task-dependent loss $E(x; x_o)$ The gradient of the loss w.r.t. the weights θ is then computed and used to update the parameters. The optimization gradually refines θ, and the output image becomes similar to the uncorrupted image.

Network Output: The restored image $x^* = f_{\theta^*}(z)$.

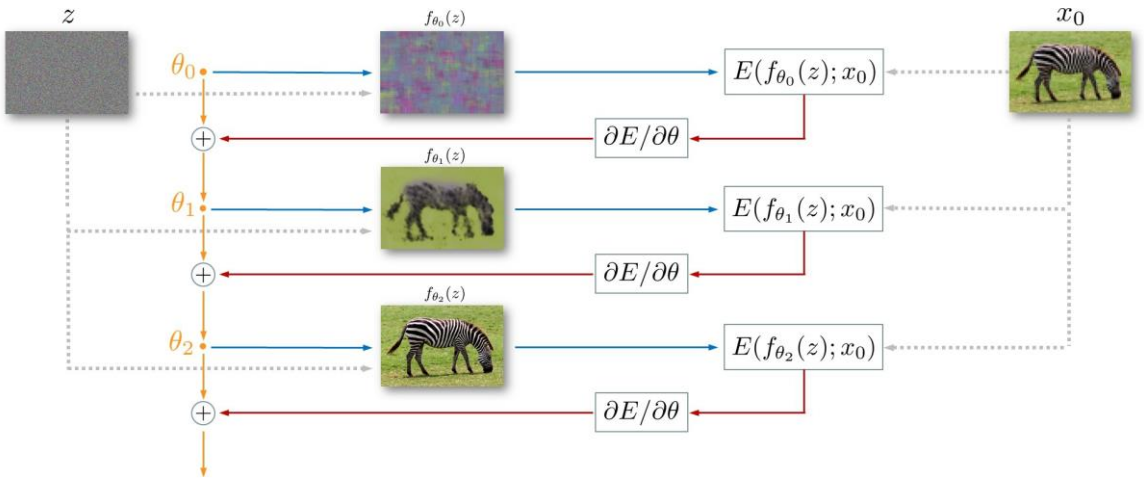


Figure 4.1 Image Restoration using the Deep Image Prior

Starting from a random weights θ we iteratively update them in order to minimize the data term. At every iteration t the weights θ are mapped to an image $x = f_\theta(z)$, where z is a fixed tensor and the mapping f is a neural network with parameters θ. The image x is used to compute the task-dependent loss $E(x; x_o)$. The gradient of the loss w.r.t. the weights θ is then computed and used to update the parameters.

4.1.2 Denoising with Deep Image Prior

Deep Image Prior (DIP) presents an innovative approach to image denoising by leveraging its implicit high impedance to image noise. This characteristic makes it naturally suited to filter out noise from an image without requiring explicit training on large datasets.

4.1.2.1 Problem Formulation

Denoising aims to recover a clean image. x from a noisy observation x_o. Sometimes the degradation model is known as: $x_o = x + \varepsilon$ Where ε follows a particular distribution. However, more often in blind denoising the noise model is unknown. However, more often in blind denoising the noise model is unknown.

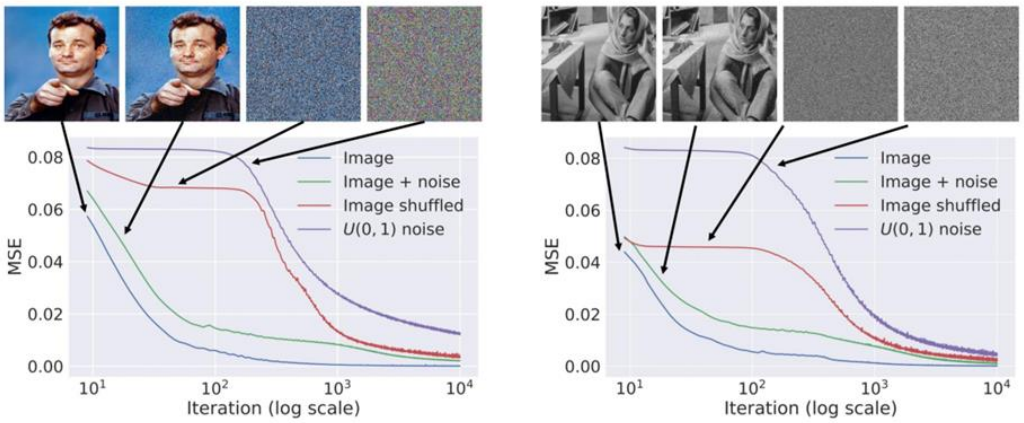


Figure 4.2 Denoising Capability of Deep Image Prior

An experiment illustrating the optimization behavior of Deep Image Prior (DIP) used four image types: a natural image, a noisy version, a scrambled version, and pure white noise. Results show DIP converges faster on (cases 1 and 2) than on noisy or scrambled inputs (cases 3 and 4).

This demonstrates that the network's parametrization inherently resists fitting noise while favoring meaningful patterns in the data. The parametrization exhibits **low impedance** to signal (natural images) and **high impedance** to noise, thus prioritizing meaningful structures during the optimization process. This proves DIP's effectiveness in tasks like denoising. Limiting optimization iterations can help avoid overfitting to noise and improve performance.

4.1.2.2 DIP Denoising Methodology

- **Network Input**
  - DIP uses a random noise tensor as the input, rather than the noisy image.
  - The noisy image serves as the target, guiding the optimization to reveal the clean underlying content.
- **Input Dimensions**
  - The input noise tensor has the same height and width as the target image.
- **Network Setup**
  - DIP employs an encoder-decoder CNN with skip connections.
  - The encoder compresses image features into a latent space.
  - The decoder reconstructs the image at full resolution.
  - Skip connections help retain fine-grained details, such as textures and edges, by bypassing certain layers.
- **Hyperparameter choices :**

$z \in \mathbb{R}^{32 \times W \times H} \sim U(0, \frac{1}{10})$
$n_u = n_d = [128,\ 128,\ 128,\ 128,\ 128]$
$k_u = k_d = [3,\ 3,\ 3,\ 3,\ 3]$
$n_s = [4,\ 4,\ 4,\ 4,\ 4]$
$k_s = [1,\ 1,\ 1,\ 1,\ 1]$
$\sigma_p = \frac{1}{30}$
$\text{num\_iter} = 2000$
$\text{LR} = 0.01$
$\text{upsampling} = \text{bilinear}$

Figure 4.3 DIP denoising architectural hyperparameter

- **Optimization Phase**
  - Gradient descent is used to adjust weights and biases, gradually reducing the loss.
  - The Adam optimizer is used with a learning rate of 0.01, controlling the update step size during training.

4.1.2.3 DIP Denoising Results

To evaluate the denoising performance of the Deep Image Prior (DIP) method, the study used two standard benchmark datasets: Set5 and Set14. These datasets, containing widely

recognized natural images, offered a controlled setting to assess DIP's effectiveness on images degraded with synthetic Gaussian noise. The denoising performance of Deep Image Prior (DIP) was quantitatively evaluated using the Peak Signal-to-Noise Ratio (PSNR) metric.

Despite requiring no training data or explicit regularization, DIP achieved competitive results compared to state-of-the-art supervised methods. For example, on the Barbara and Monarch images, DIP reached PSNR values of 27.6 and 30.24, closely matching those of trained models like ESRGAN (28.2, 28.9) and FFDNet (30.5, 32.95). These results highlight DIP's ability to approximate the performance of advanced learning-based techniques without any training.

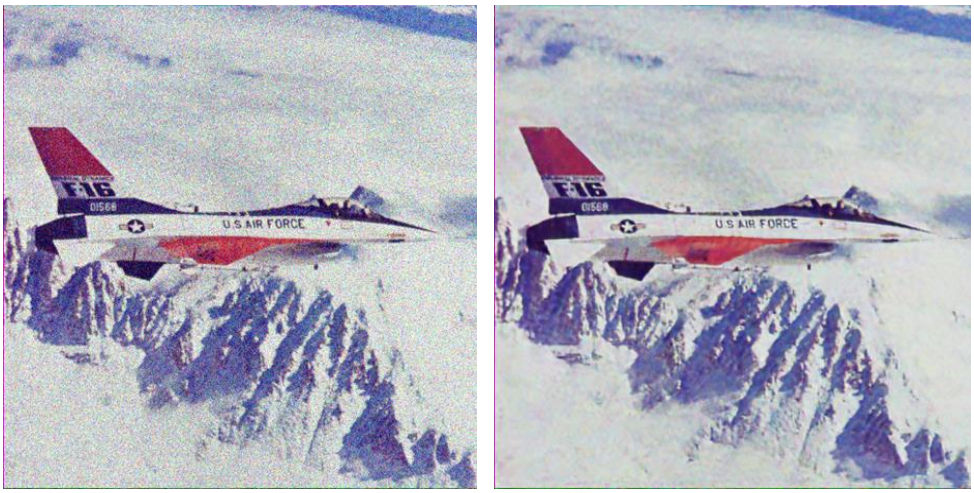


Figure 4.4 DIP Denoising on F16 Image

Observations:

- **Number of Iterations:** DIP's performance is highly sensitive to iteration count. Too many iterations can cause overfitting to noise.
- **Hyperparameters:** Large input depths can capture more detail but increase overfitting risk. A moderate depth (e.g., 32) often provides a good balance between preserving detail and suppressing noise.
- **Skip Connections:** Skip connections help balance between capturing high-level structures and retaining low-level details. They reduce the number of iterations required to achieve good results,
- **Network Architecture:** Deeper networks capture more complex patterns but risk overfitting. Shallower networks are less expressive but may generalize better in some scenarios.

4.1.3 Super Resolution with Deep Image Prior

In this implementation, the default architecture was used with five layers having 128 channels each, kernel sizes of 3, strides of 4, and skip connection kernels of size 1. The hyperparameters included an input noise level of 1/30, 2000 optimization iterations, a learning rate of 0.001, and bilinear interpolation for upsampling.

4.1.3.1 The Lanczos2 kernel

The Lanczos2 kernel plays a key role in the downsampling process for super-resolution tasks by serving as a decimation operator that performs low-pass filtering. Based on a modified sinc function, it is designed to preserve essential image details while minimizing aliasing artifacts, thus improving the quality of reconstructed high-resolution images.

The Lanczos2 kernel is defined as:

$$L(x) = sinc(x).sinc(\frac{x}{a}).box(|x| < a)$$

where $a = 2$ limits the kernel's support to the range $[-2, 2]$, ensuring localized, efficient computation.

The Lanczos2 kernel combines:

1. **Anti-Aliasing:** via the primary sinc function, which filters out high-frequency components that could cause distortion during downsampling.
2. **Detail Preservation:** through the second sinc function, which balances the frequency spectrum to retain crucial low-frequency information.
3. **Finite Support:** provided by the box function, ensuring the kernel affects only nearby pixels, reducing computational complexity and avoiding long-range artifacts.

These components work together to suppress high-frequency components that would cause aliasing in the downsampled image while preserving as much detail as possible during the downsampling process.

4.1.3.2 DIP SR Results

The super-resolution capabilities of the proposed DIP-based approach were evaluated using the Set5 and Set14 benchmark datasets. Images were first downsampled and then super-resolved using three methods: DIP, bicubic interpolation, and the trained ESRGAN model. PSNR was computed by comparing each method's output to the ground truth. The results showed that bicubic interpolation produced noticeably blurred images, while the DIP approach preserved fine details more effectively.

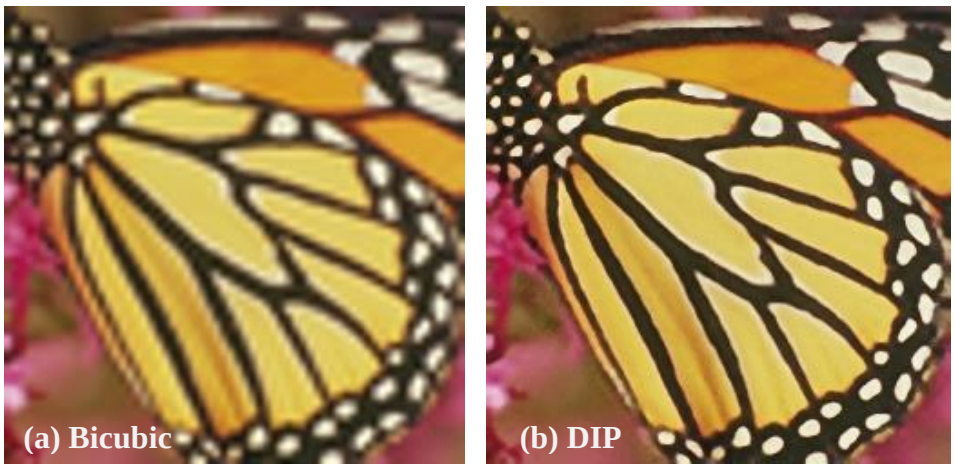


Figure 4.5 DIP SR on Butterfly Image

Limitations

While the Deep Image Prior (DIP) approach demonstrates significant potential in super-resolution tasks, certain limitations were observed:

When dealing with low-resolution images containing fine, high-detail features, the super-resolved outputs sometimes result in pixelated lines. Similarly, for images with text that is heavily blurred, the approach fails to reconstruct the text clearly, leaving it still blurred and undefined. This occurs because DIP operates without using pre-trained models or explicitly defined priors for natural images, relying solely on the structure of the neural network. As a result, it struggles to accurately infer and preserve subtle details, leading to visible distortions in such cases.

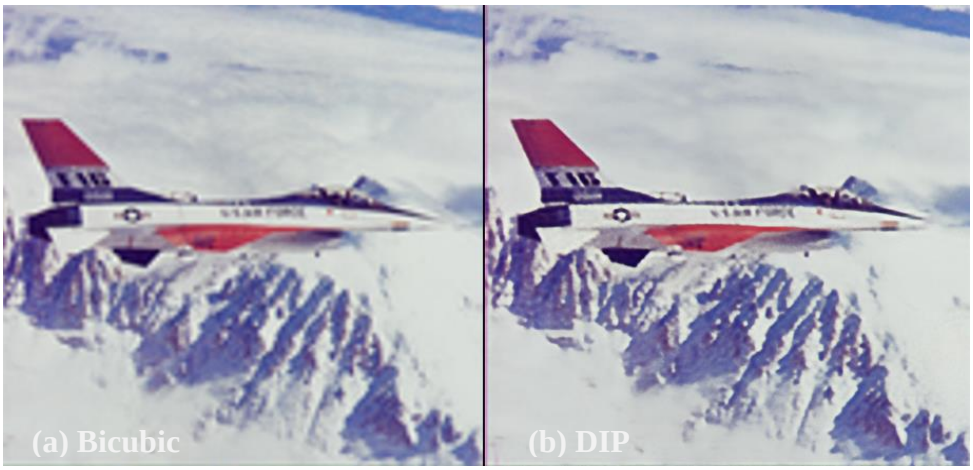


Figure 4.6 DIP SR F16 image

4.1.4 Inpainting with Deep Image Prior

DIP can successfully support inpainting application with impressive results in different types of masks, like large hole inpainting and text inpainting.

Experiments were conducted on different images from the original dataset used in the paper, and on customized dataset. Each image has its own mask of varying shape and size. The objective was to generalize the model by discovering the impact of the hyperparameters on the inpainting performance.

The following hyperparameters were considered during the experiments:

- **Learning Rate**: Experimented with values ranging from 0.001 to 0.01. The optimal value was **0.01**, balancing convergence speed and stability.
- **Number of Iterations**: Testing values included 3500,4000,4500,5000,5500, and 6000 iterations. For most images, **3500 iterations** provided a good balance between image quality and computational cost.
- **Noise Input Type (z)**: Random noise was sampled uniformly between 0 and 0.1, and meshgrid noise. The "**noise**" type was the optimal type for most of the cases.
- **Network Architecture**: The DIP used an Hourglass encoder-decoder architecture with depth ranging from 4 to 6 layers. A 5-layer network with skip-connections and with 128 filters showed the best tradeoff between computational efficiency and reconstruction quality.

- **Standard Deviation of the input noise**: Experimented values 0.0 and 0.03. The optimal value was **0.03.**
- **Input depth:** Experimented values included 1,2,6 and 32. The optimal value was **2**.

4.1.4.1 DIP Inpainting Results

The shape and the size of the mask have a large impact on the visual quality:

- Small mask (small missing areas) is inpainted with high performance as the network rely on the neighbor pixels.
- The text masks are inpainted with accurate fine details without any halo artifacts.
- Large hole inpainting worked successfully for most of the images, but with a few images it gave a blurry effect on the inpainted region.

The Selection of the hyperparameters was guided by systematic experimentation, and the impact of each hyperparameter on the visual quality was observed:

- **Learning rate:** It does not have an impact on the visual effect, but it affects the running time. Decreasing the value leads to increasing the running time and vice versa.
- **Iterations Number:** High numbers of iterations lead to overfitting in the known regions, but degrading performance in the missing areas.
- **Noise Type:** “mesh grid” noise performs successfully with some cases of large holes with real textures and details) but led to blurry areas in other images. Random noise “noise” performs successfully with most of the images.
- **Input Depth:** Large numbers like 6 and 32 lead to a blurry effect on the missing region, but small numbers like 1 and 2 perform successfully with missing parts.
- **Network Architecture:** A large number of filters [256]*5 for upsampling and downsampling layers resulted in overfitting, capturing noise rather than meaningful patterns due to its excessive capacity. But using less number of filters [128]*5 performs a stable and balanced feature representation across all layers.

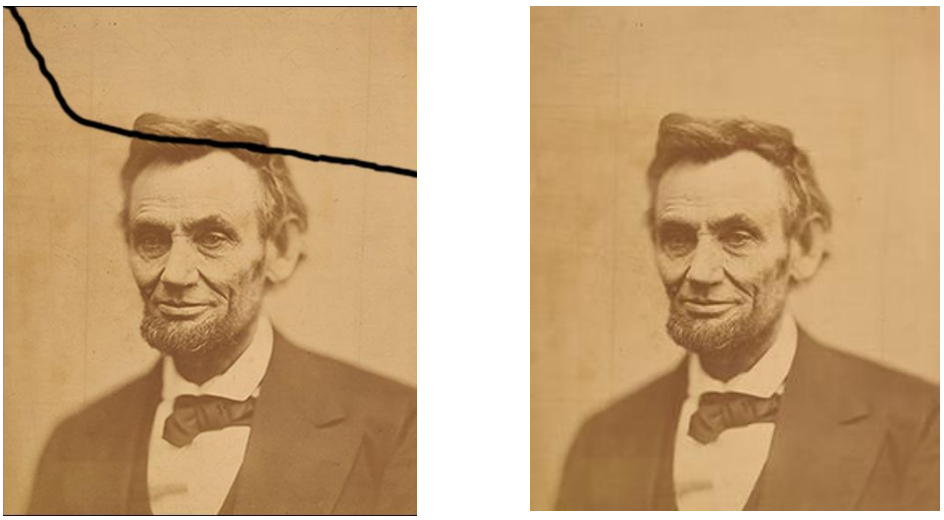


Figure 4.7 DIP Inpainting on Lincoln Image

4.1.5 Limitations of Deep Image Prior

The Deep Image Prior (DIP) has shown remarkable effectiveness across a range of image restoration tasks, such as denoising and inpainting. Nevertheless, its practical implementation is limited by several limitations.

4.1.5.1 General Limitations

- **Computational Time:** DIP is inherently slow due to the optimization process for each image. This involves high computational overhead in running the untrained neural network from scratch for each input image. Then, the problem becomes even more accentuated when dealing with high-resolution images.
- **Sensitivity to hyperparameters:** DIP requires careful tuning of hyperparameters for every single image to achieve the best possible results. Although a wide range of hyperparameters might yield acceptable results, finding the best configuration can be very time-consuming and usually requires iterative trial-and-error strategies. This leads to a lack of generalization of the model.
- **Computational Resources:** DIP requires large computational resources, which may be hard to apply in resource-constrained environments, such as on free-tier GPUs or limited hardware.
- **trial-and-error:** Most previous DIP studies set the number of iterations based on visual inspection or trial-and-error. Relying on trial and error makes the process subjective and prone to inconsistency, especially in unfamiliar scenarios.

4.1.5.2 Inpainting Limitations

- The training process is sensitive to the mask used for inpainting. Large masks lead to slow time due to their size.
- Highly semantic images have slightly blurry effects in the inpainted areas due to the texture and details of the image.
- The lowest loss does not correspond to the best visual result as it sometimes refers to overfitting while training the network. The discrepancy makes it difficult to determine the best output based on the loss value only.

4.1.5.3 Denoising Limitations

- **Overfitting to Noise:** DIP leverages the inductive bias of convolutional neural networks to prioritize signal reconstruction over noise during the early stages of training. However, as the number of iterations increases, the network begins to overfit the noise, degrading the denoising performance.

4.2 DEEP DECODER

4.2.1 Introduction

While the Deep Image Prior (DIP) effectively handles various image restoration**,** it is susceptible to overfitting noise if regularization is absent**, due to its over-parametrization**. To overcome this limitation, the Deep Decoder (DD), proposed by R. Heckel and P. Hand, introduces an under-parameterized, untrained, non-convolutional neural network that addresses DIP’s vulnerability to overfitting.

The Deep Decoder is a simple neural network: it is under-parameterized (has less parameters than the dimension of the generated image) and does not use convolutions. It does not require training and relies entirely on its architecture for assumptions about the data. The network is called the deep decoder due to its resemblance to the decoder part of an autoencoder.

4.2.2 Intuitive Representation of Image Signals Using the Deep Decoder Model

A model efficiently represents a class of signals if it can approximate any signal in that class using the fewest parameters possible.

To benchmark the Deep Decoder, each image was represented using the N-largest wavelet coefficients. Wavelets, which form the basis of JPEG 2000 compression, are renowned for their ability to approximate images with a small number of coefficients.

Compression Factor 32.3: For higher compression ratios, the Deep Decoder slightly outperforms wavelet compression for most images (crosses lie above the red line).

Compression Factor 8: For lower compression ratios, wavelet compression is marginally better.

These findings highlight that the Deep Decoder can effectively represent natural images with very few parameters, without requiring learning or training. The superior performance of wavelets at smaller compression factors is expected, as wavelets can exactly represent any image given enough coefficients. The Deep Decoder, being underparameterized, cannot guarantee zero representation error in such cases.

The primary goal of this experiment was to demonstrate that the Deep Decoder is a strong image model, capable of supporting applications like solving inverse problems. Additionally, the model shows potential for lossy image compression by quantizing its coefficients C and storing them.

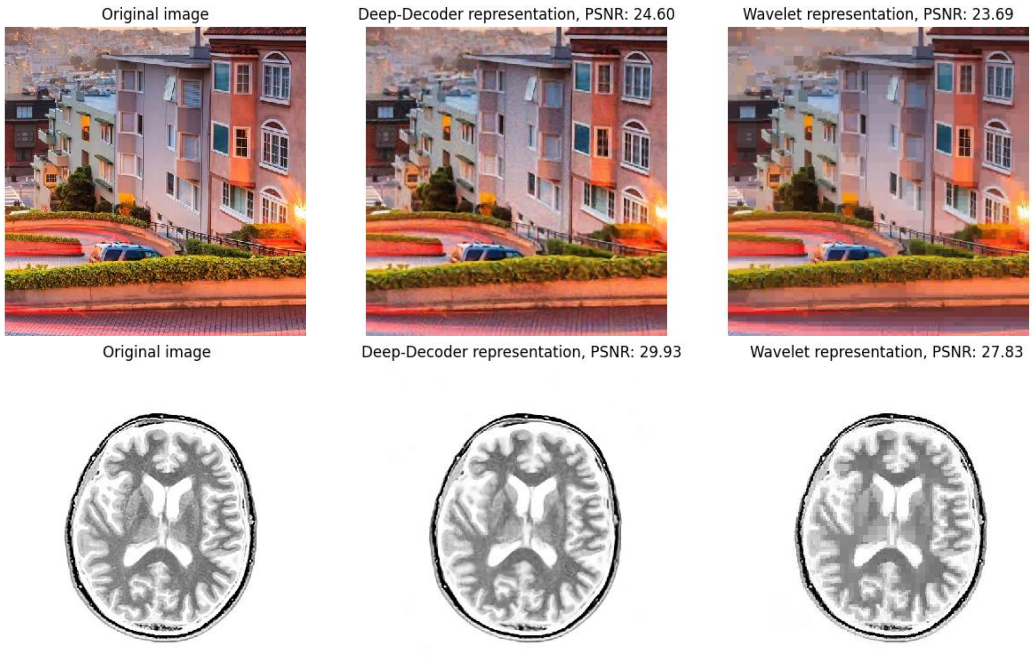


Figure 4.8 Image Compression with compression factor = 32.3

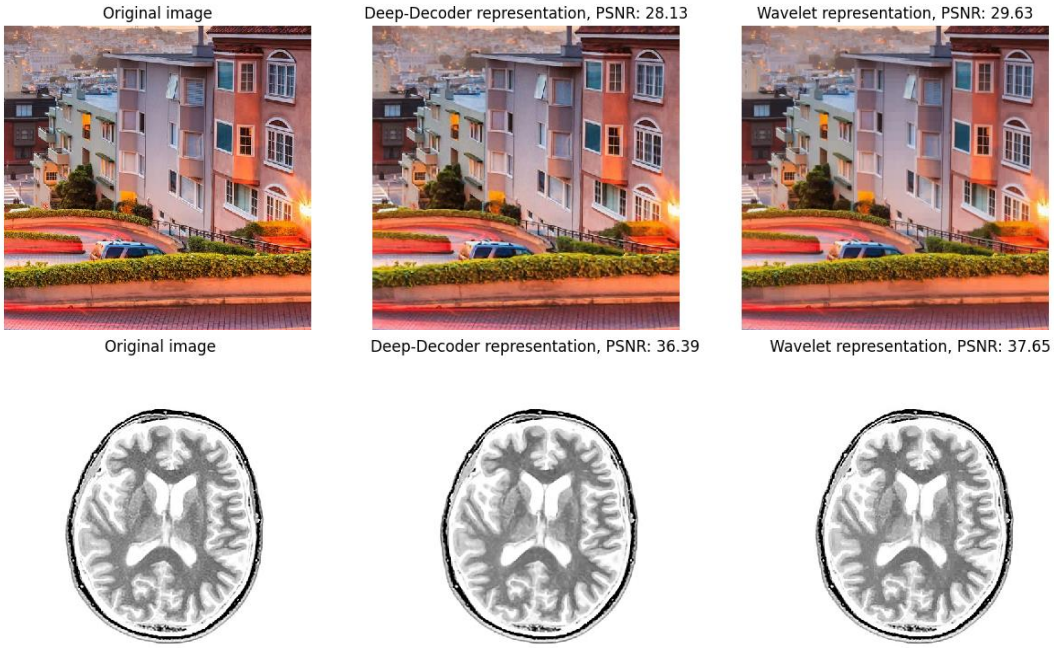


Figure 4.9 Image Compression with compression factor = 8

4.2.3.1 Network Architecture

The Deep Decoder is a decoding-only neural network designed to generate high-resolution images from a fixed, randomly initialized input tensor. Unlike encoder-decoder models, it omits the encoding phase and directly maps the input to a detailed output. The input tensor typically has dimensions of 16×16 with multiple feature channels and is transformed into a high-resolution image.

The network consists of sequential decoding layers that refine the input through pixel-wise linear combinations, bilinear upsampling, ReLU activations, and channel normalization. Each layer increases spatial resolution and adjusts feature channels using learned weight matrices.

Bilinear upsampling increases spatial dimensions, ReLU introduces non-linearity, and channel normalization ensures stable output. This architecture enables the Deep Decoder to effectively generate detailed images without overfitting or requiring convolutional layers.

4.2.3.2 Channel Normalization

Channel normalization in the Deep Decoder plays a key role in stabilizing the values within each feature channel during training. Unlike traditional batch normalization, which normalizes across an entire batch of data, channel normalization operates independently on each channel. It standardizes values by subtracting the channel mean and dividing by its variance, ensuring no single channel dominates learning.

4.2.3.3 Network Parameters

The Deep Decoder's parameters consist solely of the learned weight matrices $C_0, C_1, \dots C_d$, while the input tensor B_0 is fixed and randomly initialized. This minimalist setup allows the model to function entirely based on these learned weights. The total number of parameters depends on the number of layers d, the number of feature channels K_i, and the number of output channels K_{out}.

This parameter count is calculated as: $N = dK^2 + 2dK + 3K$

For standard configurations—such as six layers with 64 or 128 channels—the parameter counts are 25,536 and 100,224, respectively. This is considerably smaller than the number of pixels in a high-resolution image (e.g., 512×512×3), illustrating the model's under-parameterized yet effective design.

4.2.3.4 Architectural Flexibility and Variations

The Deep Decoder's architecture is flexible, allowing modifications to the order and type of operations within its layers. For example, an earlier version of the architecture applied the ReLU activation function before the upsampling operation. However, experiments showed that applying upsampling first, followed by ReLU, resulted in better image reconstruction quality. These adjustments improved the network's performance without increasing its complexity or compromising its simplicity and interpretability.

4.2.3.5 Non-Convolutional Layers

The Deep Decoder differs from traditional CNNs by avoiding spatial operations like 3×3 convolutions that mix information from neighboring pixels. Instead, it uses pixelwise linear combinations of feature channels, processing each pixel independently without spatial coupling. This uniform transformation applies the same weight matrix to all pixels, ensuring consistency across the image. The approach is conceptually similar to 1×1 convolutions in CNNs, where each pixel's feature channels are combined using shared weights, but without involving any spatial context.

By avoiding spatial coupling, the Deep Decoder simplifies its structure and reduces its parameter count, leading to:

- **Improved Resilience to Overfitting:** The absence of spatial connections prevents unnecessary dependencies between pixels.
- **Greater Robustness to Noise:** Without relying on local pixel relationships, the network avoids amplifying corrupted or noisy inputs.
- **Simplified Architecture:** Fewer parameters make the Deep Decoder both efficient and interpretable, while still maintaining strong performance.

4.2.3.6 Overfitting Noise

The Deep Decoder's architecture is naturally resistant to overfitting due to its underparameterized design, using significantly fewer learnable parameters than the number of output pixels. This compact structure prevents the network from memorizing noise, enabling effective denoising without requiring large training datasets.

In contrast, the Deep Image Prior (DIP) uses an overparameterized convolutional network that, while effective for inverse problems, relies on early stopping to avoid overfitting. This dependence on external regularization highlights a key limitation of DIP, whereas the Deep Decoder achieves robustness through its inherently constrained architecture.

The results highlight a fundamental contrast between DIP and the Deep Decoder. DIP, due to its overparameterized architecture, can fit both clean image structures and noise, necessitating external regularization methods like early stopping to prevent overfitting. It also converges to a clean image faster than the Deep Decoder. In contrast, the Deep Decoder's underparameterized design inherently limits its ability to fit noise, even with extended iterations. This is because it maps a low-dimensional input tensor to a high-dimensional output image with far fewer learnable parameters than output pixels. As a result, it naturally

focuses on capturing essential image features while ignoring much of the noise, making it particularly well-suited for tasks like denoising without additional regularization.

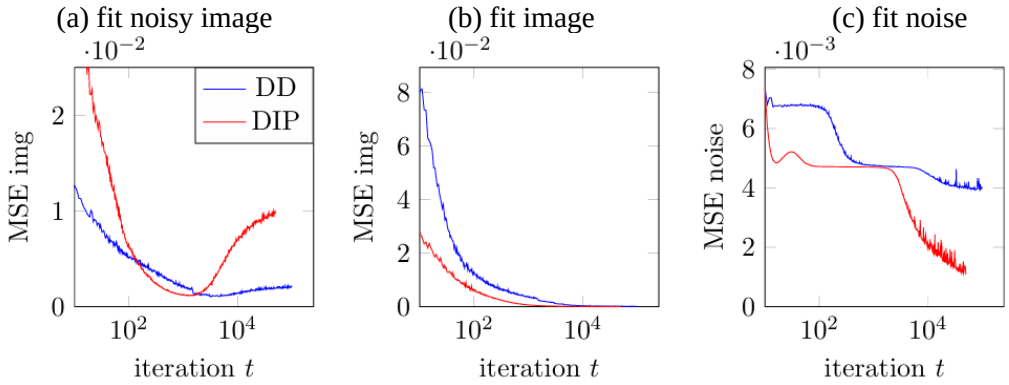


Figure 4.10 Performance of DD and DIP across Three experiments

4.2.3.7 Gaussian Noise and Parameter Limitations

The Deep Decoder attempts to fit the noisy target image by minimizing the squared distance between its output G(C) and the noisy image y. However, due to its underparameterization, the network cannot perfectly fit the noise. This limitation is mathematically expressed in the following equation:

$$min\ ||G(C) - \eta||_2^2 \geq ||\eta||_2^2(\frac{1 - 20K^2 log(n_0)}{n})$$

In this equation, G(C) represents the output generated by the Deep Decoder using its learnable parameters C, and η is the zero-mean Gaussian noise that the model is trying to fit. This noise, η has a covariance matrix I (the identity matrix), which indicates that the noise is independent and identically distributed with zero mean and unit variance. The variable n refers to the total number of pixels in the output image, k is the number of feature channels in the network, and n_0 is the size of the input tensor.

In brief, the equation highlights a key property of the Deep Decoder: its underparameterization inherently prevents it from perfectly fitting noise. The left side of the inequality represents the error between the network's output and the target noise, while the right side acts as a threshold a mathematical lower bound on how small the error can be. This threshold is determined by the network's limited number of parameters relative to the high dimensionality of the output image.

4.2.3.8 Upsampling

In the Deep Decoder, spatial coupling is introduced exclusively through bilinear upsampling, unlike CNNs which achieve spatial coupling via convolutions and pooling layers. Since the Deep Decoder does not use convolutions, it depends entirely on bilinear interpolation to increase resolution. This upsampling method computes new pixel values by averaging the intensities of nearby pixels, thereby introducing spatial dependencies.

For example, upsampling a 16×16 tensor to 32×32 involves inserting new pixels whose values are derived from surrounding pixel values, such as averaging four adjacent pixels (A, B, C, D) to produce a new one: (A + B + C + D) / 4. This limited but effective form of spatial coupling

preserves the Deep Decoder's simplicity and interpretability, while still enabling it to perform well in tasks like denoising and inpainting.

4.2.3.9 Network Input

In the Deep Decoder model, the network input B_1 is fixed throughout the optimization process. The network input B_1 is a matrix where each entry is generated randomly from a uniform distribution. Random generation ensures that the matrix rows are incoherent (meaning that the rows are dissimilar and uncorrelated.) The presence of highly similar and correlated rows restricts the range of images the Deep Decoder can represent.

4.2.4 Denoising with Deep Decoder

The DD excels in denoising due to its ability to represent natural images efficiently while being under-parameterized. This under-parameterization limits the network's capacity to fit noise, effectively filtering out a significant portion of it while preserving the signal.

For our experiments, we use the DD decoder network. The input to the network is a tensor of randomly generated uniform noise. The optimization is performed using the Adam optimizer. The loss is calculated using the least-squares loss defined above.

Model hyperparameters:

```
k=128
num_channels = [128, 128, 128, 128, 128]
numit = 1900
rn = 0.0
```

The parameter k represents the number of latent features. It influences the trade-off between representation error and noise filtering. Larger k values may retain more noise. Smaller k values result in more aggressive noise filtering. The choice of k depends on the noise level, with lower noise requiring larger k values. For example, k = 64 or 128 works well for moderate noise levels, while k=32 is more effective for higher noise (PSNR ≈ 14dB).

4.2.4.1 DD Denoising Results

We test the performance of the DD model and observe that it performs well on standard benchmark test sets, effectively reconstructing and denoising images.

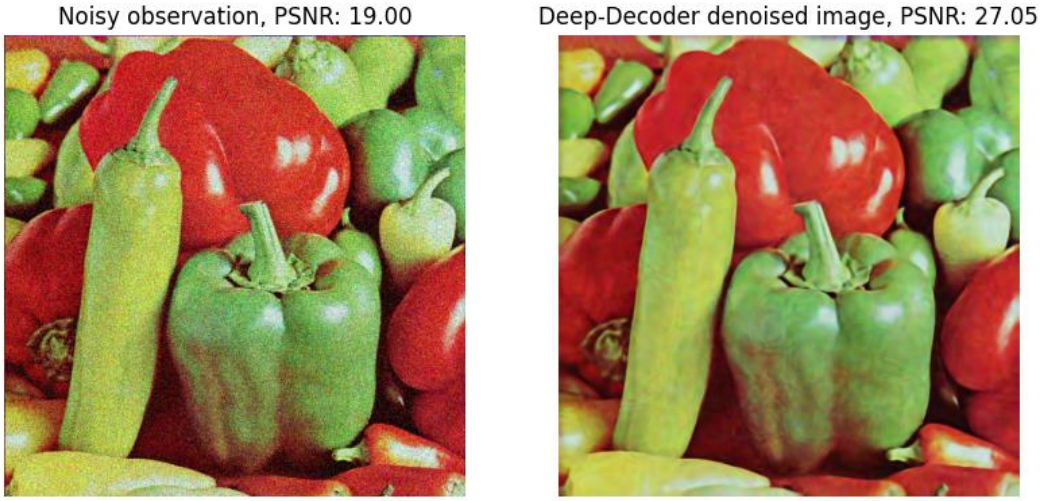


Figure 4.11 DD Denoising on the Peppers test image

When applied to complex real-world images, we observe the DD's limitations. In images that feature complex textures, fine details, or high-frequency components, the DD's under-parametrization restricts its capacity to fully represent these images.

As a result, we can conclude that while the Deep Decoder excels in scenarios where the input images have simple textures and less fine details, but its applicability is limited in practical settings that require high fidelity for fine image features.

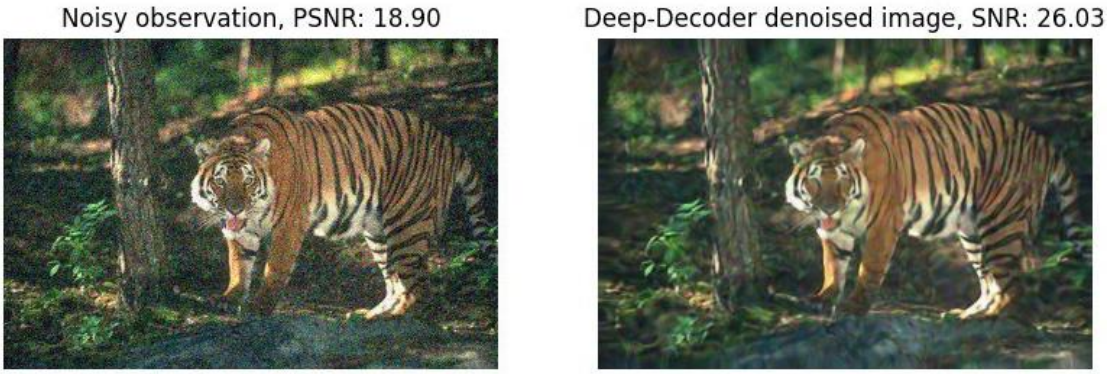


Figure 4.12 DD Denoising in textured images

4.2.5 Inpainting using Deep Decoder

The Deep Decoder was employed for inpainting, where it attempted to reconstruct the missing regions of an image. We tested the inpainting logic on several images using masks of varying sizes, from small to large, to evaluate its robustness and adaptability.

A critical aspect of our study involved analyzing the impact of key hyperparameters on the quality of the inpainted images. The original paper identifies four primary hyperparameters, which we investigated to determine their optimal settings and to understand their effects on image quality and the inpainting areas. The hyperparameters and their default values, as specified in the original code, are as follows: the number of iterations is 40000, the standard deviation of the input noise is 0.005, the learning rate is 0.0025, and the decay of the input noise standard deviation is 500.

4.2.5.1 Hyperparameter Experiments:

- **Number of Iterations**: the number of iterations significantly affects image quality—too many lead to overfitting and unnatural artifacts, while too few result in blurred, detail-lacking outputs. A setting of 10,000 iterations was found to offer a good balance.
- **Input Noise Standard Deviation ("reg_noise_std")**: This parameter also plays a key role: higher values improve robustness in masked areas but blur the image, while lower values preserve sharpness but reduce inpainting quality. A moderate value of 0.05 worked best overall.
- **Learning Rate**: The learning rate mainly influenced training speed rather than output quality; 0.01 provided stable and efficient convergence.
- **Decay of Input Noise Standard Deviation ("reg_noise_std_decayevery")**: This parameter affects overfitting and blur trade-offs, with a decay setting of 1000 achieving a generally effective balance between artifact reduction and inpainting performance.

Chosen Hyperparameters to Generalize the Model

Number of iterations: 10 000 Standard deviation of the input noise: 0.05 Learning rate: 0.01 Decay of Input Noice Standard Deviation: 1000

4.2.5.2 DD Inpainting Results

The shape and size of the mask have a significant impact on the visual quality of the inpainted images. Small masks, which correspond to smaller missing areas, were inpainted with high performance, as the network relied on neighboring pixels to effectively reconstruct the missing regions. Even when the original image contained some artifacts, the inpainting was still done very well. Text masks were inpainted with remarkable accuracy, preserving fine details without introducing any halo artifacts.

(a) image (b) image masked (c) Deep Decoder

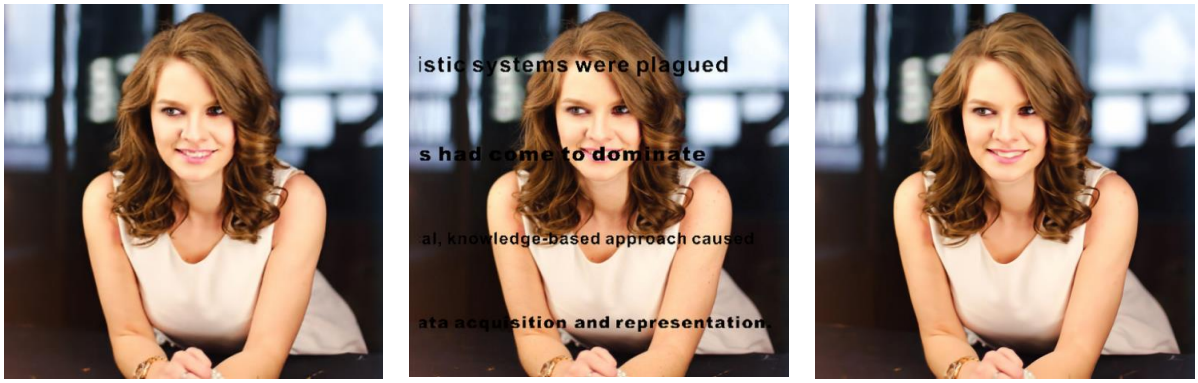


Figure 4.13 DD inpainting results with chosen parameters

The Deep Decoder struggles with large-hole inpainting, often failing to reconstruct complex textures and details in missing regions. For example, when inpainting a zebra against a grassy background, it fills the area with generic green tones rather than replicating the detailed grass texture. This weakness is due to its decoder-only architecture, which lacks hierarchical feature representation and skip connections. In contrast, Deep Image Prior (DIP), with its encoder-decoder structure and skip connections, preserves both high-level structure and fine details, making it more effective in handling large, complex inpainting tasks.

Chapter Five: Comparative Analysis

5 COMPARATIVE ANALYSIS

5.1 DENOISING

In this section, we present a comparative study of four deep learning-based methods for image denoising: two untrained approaches, Deep Image Prior (DIP) and Deep Decoder (DD), and two pre-trained models, FFDNet and ESRGAN. The evaluation is conducted on two standard benchmark datasets, Set14 and Set5, which include images of varying complexity, ensuring a comprehensive assessment of the methods under diverse conditions. Additionally, we test on a subset of photos from the PolyU-Real-World-Noisy-Images-Dataset, which contains 40 scenes captured by 5 cameras from leading brands – Canon EOS (5D Mark II, 80D, 600D), Nikon (D800), and Sony (A7 II)—to evaluate performance under real-world camera noise and challenging conditions such as low lighting. This study highlights the performance gap between untrained and trained methods, with a focus on the potential of untrained approaches to deliver competitive results despite not relying on prior training data. While trained methods, such as FFDNet and ESRGAN, leverage learned priors from extensive datasets to achieve state-of-the-art performance, the untrained methods demonstrate surprisingly close results in certain scenarios, showcasing their adaptability and robustness.While ESRGAN is primarily designed for super-resolution, it was included in this study to explore its capability in a related task like denoising. Although not optimized for denoising, ESRGAN's performance provides valuable insights into the adaptability of super-resolution models to other restoration tasks. Additionally, FFDNet was utilized in the study to compare the performance of trained and untrained methods. Quantitative comparisons are made using the (PSNR) metric, objectively evaluating reconstruction quality. Furthermore, qualitative visual assessments are included to analyze perceptual performance. The results reveal that, in many cases, untrained methods can achieve performance levels comparable to those of trained models, making them a promising alternative, particularly in scenarios where training data or computational resources are limited. These results are derived from experiments conducted on noisy images with a noise level of 30, which exceeds the noise levels typically tested in Deep Image Prior and Deep Decoder studies. Additionally, a non-local filter with h=0.01 was applied to the final results.

5.1.1 Quantitative results

Table 5.1 denoising comparison 1

Method	Barba-ra	zebra	baboon	pepper	Ppt3	monarch	man
Dip	27.60	26.18	22.5	28.29	26.93	30.24	26.22
DD	24.06	26.11	21.35	26.97	25.06	28.96	26.03
FDD	29.5	28.46	23.80	29.68	33.19	33.95	28.73
ESRGAN	23.01	25.54	21.01	28.02	26.61	28.83	25.76

Table 5.2 denoising comparison 2

Method	bridge	lenna	flowers	comic	face	Coast-guard	foreman
Dip	24.97	28.58	26.24	23.47	25.76	26.47	27.49
DD	24.53	28.27	25.63	23.32	26.81	25.56	27.55
FDD	25.90	30.35	28.37	27.14	28.9	29.01	32.15
ESRGAN	23.45	27.97	25.69	23.82	25.81	24.45	27.27

Table 5.3 denoising comparison 3

Method	Fruit	MRI	Astron-aut	castle	baby	bird	F16
Dip	28.57	26.51	30.01	28.06	29.02	24.20	28.85
DD	27.32	26.66	29.18	26.99	28.51	27.65	26.82
FDD	31.4	31.8	34.02	31.27	32.57	31.99	31.94
ESRGAN	26.84	29.03	28.55	25.51	26.37	26.99	25.8

5.1.2 Qualitative results

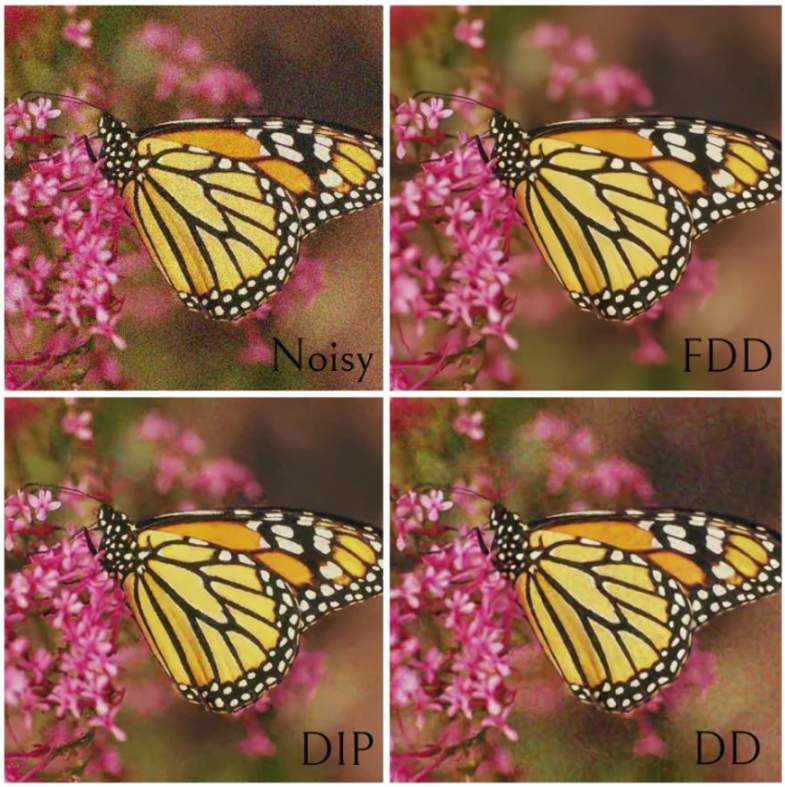


Figure 5.1 monarch denoising comparison

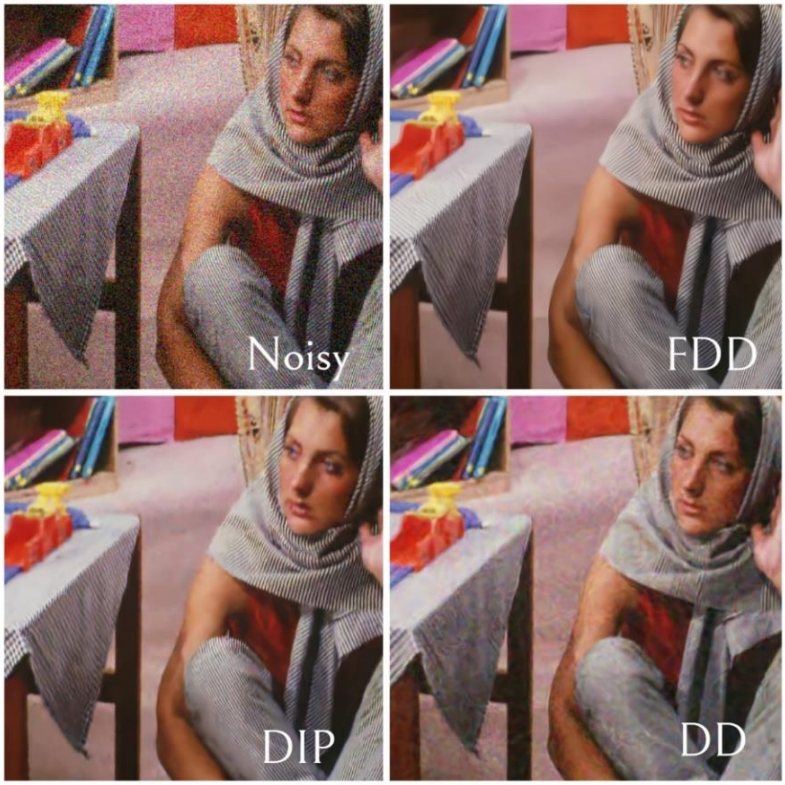


Figure 5.2 barbara denoising comparison

5.1.2.1 Results Interpretation

The comparative study reveals intriguing insights into the strengths and limitations of untrained and trained methods for image denoising. Among the untrained methods, Deep Image Prior (DIP) demonstrates highly promising results. Although it is outperformed by the trained method FFDNet, particularly in terms of Peak Signal-to-Noise Ratio (PSNR), the performance of DIP is remarkable for an untrained approach. Its ability to recover fine details and denoise effectively without relying on pre-trained knowledge highlights its potential as a viable competitor to trained models. This capability is particularly noteworthy in scenarios where training data is unavailable or computational resources are limited . On the other hand, Deep Decoder (DD) exhibits significant limitations. Due to its decoder architecture, it struggles to preserve critical details and edges, which are essential for high-quality image restoration. This deficiency leads to noticeable artifacts and a loss of structural integrity in the denoised images. Consequently, while DD showcases the potential of untrained approaches in simpler tasks, its lack of detail retention makes it unsuitable for deployment in high-fidelity denoising applications.In summary, the study confirms the superiority of trained methods like FFDNet in producing consistently high-quality results. However, the surprising performance of DIP underscores the potential of untrained methods to bridge the gap and compete effectively in certain contexts.

5.1.3 Introduction to Real-World Noise Analysis

While the initial evaluation focused on controlled noise scenarios from standard datasets, real-world images present a different challenge. Factors such as low lighting, high-detail textures, and complex noise patterns often push denoising methods to their limits. To explore these challenges, we examine the performance of the selected methods on the PolyU-Real-World-Noisy-Images-Dataset. Interestingly, the trained method FFDNet, which showed strong

results on synthetic benchmarks, struggles significantly to denoise the low-light, real-camera noise in this dataset. This raises questions about the generalizability of trained methods in unpredictable real-world settings.In the context of real-world noise conditions, Deep Decoder fails to effectively denoise images while preserving critical details. It exhibits significant oversmoothing, struggles to capture finer edges, and ultimately falls short in handling the complexities of real-camera noisy images. **In contrast, Deep Image Prior (DIP) demonstrates exceptional performance in this scenario, successfully denoising challenging images under low-light conditions. Surprisingly, for the first time, DIP outperforms the pre-trained method FFDNet, which struggles to handle the complex noise present in real-world low-light images.**

DIP's success comes with a cost. Achieving optimal results requires a processing time of 8–9 minutes per image on a Google Colab T4 GPU.

Figure 5.3 FDD vs DIP Denoising on Disk Image

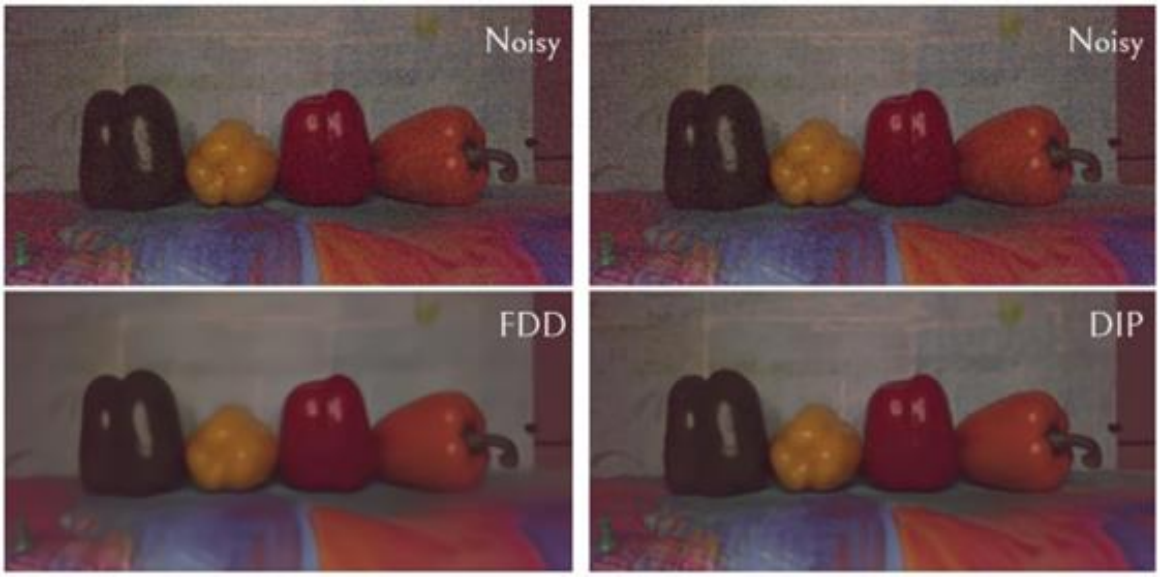


Figure 5.4 Figure 5.4 FDD vs DIP Denoising on Vegetables Image

Table 5.4 Real camera denoising

Method	Gadget desk	Toy car	Vegetables
Dip	31.15	30.51	30.06
DD	27.50	30.01	29.90
FDD	30.50	30.29	29.66

5.1.4 Proposed Method: Enhancing and Accelerating Deep Image Prior

To address the limitations of the Deep Image Prior (DIP) framework in denoising real-world noisy images, particularly its susceptibility to overfitting and computational inefficiencies, this study proposes a series of modifications. These enhancements aim to improve the performance of DIP in challenging noise conditions while maintaining its computational feasibility. The proposed modifications are detailed as follows:

Reduced Network Overparameterization : the original DIP architecture is overparameterized, with approximately 2.2 million parameters. This excessive parameterization increases the risk of overfitting to noise, particularly in real-world scenarios. To mitigate this issue, a streamlined network architecture is proposed:

Proposed Encoder-Decoder Architecture : downsampling Channels: [16, 64, 64, 64, 128], Upsampling Channels: [16, 64, 64, 64, 128] and Skip Connections: [4, 4, 4, 4, 4]. This modified architecture retains the encoder-decoder structure while significantly reducing the total number of parameters to approximately **768,531**. The reduced complexity not only lowers the risk of overfitting but also improves computational efficiency, making the network more suitable for real-world applications.

Enhanced Input Initialization : in the standard DIP implementation, the input to the network is randomly initialized. This approach, however, lacks the ability to leverage prior information about the noisy image, which can lead to suboptimal results. To address this limitation, the proposed method introduces a **denoised image-based initialization**, which provides a more structured and informed starting point for the network.

Input Initialization Process:

Bilateral Filtering: The noisy image is first processed with a bilateral filter to reduce noise while preserving important edges. This step ensures that the subsequent edge detection focuses on structural features rather than noise. Parameters are tuned based on the noise level.

Edge Feature Enhancement: A Laplacian of Gaussian (LoG) filter is applied to detect and enhance prominent image edges. Gaussian smoothing is incorporated to further suppress residual noise.

Amplitude Low-Pass Denoising: The edge-enhanced image is transformed into the frequency domain, where high-frequency noise is attenuated using a low-pass filter. The inverse transformation yields a refined, denoised initialization image.

This initialization strategy provides the network with a meaningful and noise-suppressed input, helping to preserve essential image details and reduce the likelihood of overfitting to noise.

Regularization with Increased Noise : To further address the risk of overfitting, regularization is applied by adding noise to the input during training. In contrast to the default DIP regularization, which uses a noise standard deviation of **1/30**, the proposed method increases the standard deviation to **1/10**. This adjustment enhances the regularization effect, ensuring that the network learns robust image representations rather than overfitting to specific noise artifacts.

Number of Iterations: To balance the trade-off between computational efficiency and denoising quality, the number of training iterations is set to **3000**. This choice ensures sufficient optimization while avoiding excessive computational overhead.

Table 5.5 proposed method denoising

Method	Gadget-desk	Toy car	Vegetables
MOD_Dip	31.4	30.78	30.17

Performance Evaluation of the Proposed Method

The proposed modifications to the Deep Image Prior (DIP) framework achieve comparable performance to the original DIP while addressing key limitations related to efficiency and overfitting.

Quantitative Evaluation

The **Peak Signal-to-Noise Ratio (PSNR)** achieved by the proposed method is on par with the original DIP, demonstrating its ability to maintain quantitative denoising quality. The proposed method reaches this PSNR performance in **approximately half the training time**, significantly improving computational efficiency.

Model Efficiency

The original DIP architecture requires **over 2.2 million parameters**, contributing to increased risk of overfitting and higher computational demands.By contrast, the proposed architecture reduces the parameter count to approximately **750,000**, making the model more efficient and less prone to overfitting while maintaining robust performance.

Qualitative Evaluation

The proposed method **delivers similar or sometimes better qualitative results**, preserving finer details and structural integrity in the denoised images compared to the original DIP.

To demonstrate that the proposed method can achieve performance comparable to the original DIP in **half the training time** while avoiding overfitting to noise, we evaluate it on both benchmark datasets and real-world noisy images. Specifically, we test the method on the **Set14 dataset**, a widely used benchmark in image restoration tasks, to provide quantitative evidence of its effectiveness by comparing Peak Signal-to-Noise Ratio (PSNR) values.

Figure 5.5 DIP vs MOD_DIP Denoising Comparison

Table 5.6 proposed method denoising set14

Method	barbara	zebra	baboon	pepper	Ppt3	monarch	man
Dip_MOD	26.00	26.58	22.23	28.98	26.57	30.62	26.00

Table 5.7 proposed method denoising set14

Method	bridge	lenna	flowers	comic	Face	Coast-guard	forman
Dip_MOD	24.02	29.64	26.04	24.48	27.23	26.56	29.95

Figure 5.6 DIP vs modified DIP Denoising

The average runtime of the original DIP method on a T4 GPU (Google Colab) was approximately 9 minutes, which reduced to 3 minutes when executed on an A100 GPU (Google Colab Pro). In contrast, our proposed modified method achieved a runtime of 4.5 minutes on the T4 GPU, further decreasing to 1.5 minutes on the A100 GPU. This demonstrates that our method not only achieves significant acceleration compared to the original DIP approach but also exhibits superior performance on resource-limited hardware. Furthermore, the consistent reduction in runtime across different GPUs highlights the adaptability and efficiency of our approach in optimizing computational resources.

5.1.5 Conclusion

The Deep Image Prior (DIP) method demonstrates promising results in image denoising. However, our proposed method aims to address two primary limitations of DIP: overparameterization and high computational time. By mitigating these challenges, the proposed approach seeks to enhance the efficiency and scalability of the DIP framework while maintaining or even improving its strong performance in denoising tasks.

5.2 INPAINTING

In this section, we present a detailed comparative analysis of three image inpainting methods: two untrained approaches: Deep Image Prior (DIP) and Deep Decoder, and one pretrained method “deepfillv2”. The goal of this analysis is to evaluate the performance of these techniques across diverse datasets, highlighting their strengths and limitations. The study encompasses two benchmark datasets Set 14 and Set 5 as well as a custom dataset we curated, which includes a variety of images and corresponding masks. Our custom dataset features a mix of real-world images and aged photographs, adding complexity and diversity to the evaluation.

The comparative analysis employs several metrics, including Peak Signal-to-Noise Ratio (PSNR) and qualitative visual assessment, to assess the quality of the inpainted images. While PSNR is a widely used metric, it does not always align with human perception of visual quality. For instance, an inpainted image with a high PSNR value may still exhibit visible artifacts, such as shadows or traces of the original mask. This discrepancy underscores the importance of combining quantitative and qualitative evaluations to achieve a comprehensive understanding of the methods' performance.

Another key aspect of our analysis is the computational efficiency of the methods. We measure and compare the execution times of DIP and Deep Decoder, providing insights into their suitability for real-time or resource-constrained applications. All experiments were conducted on an NVIDIA A100 GPU to ensure consistency and reproducibility of results.

The findings of this study reveal that, in many cases, untrained methods like DIP and Deep Decoder can achieve performance levels comparable to those of trained models. This observation highlights the potential of untrained approaches as viable alternatives in scenarios where access to extensive training data or computational resources is limited

5.2.1 Quantitative Result

Table 5.8 Set 14 Inpainting comparison

Image	DIP		Deep Decoder		Trained model
	PSNR	Time	PSNR	Time	PSNR
baboon	26.20	1m48s	23.45	3m9s	18.27
barbara	32.94	1m46s	25.90	3m15s	30.89
bridge	28.44	1m11s	27.23	2m1s	29.60
coastguard	33.29	1m7s	31.35	1m12s	24.56
comic	29.31	1m3s	28.01	1m7s	26.58
face	32.66	1m3s	31.33	1m10s	23.07
flowers	29.85	1m2s	29.58	1m15s	19.30
foreman	38.96	1m10s	38.15	1m20s	23.04
lenna	32.79	1m32s	32.55	2m33s	34.45
man	30.06	1m8s	28.94	2m15s	31.48
monarch	32.16	1m30s	32.11	3m5s	31.11
pepper	32.6	1m14s	32.55	2m5s	33.72
ppt3	29.63	1m28s	27.52	3m2s	27.70
zebra	30.45	1m37s	28.76	3m15s	29.63

Table 5.9 Set 5 Inpainting comparison

Image	DIP		Deep Decoder		Trained
	PSNR	Time	PSNR	Time	PSNR
baby	35.07	1m19s	35.02	2m 36s	35.53
bird	37.70	1m4s	38.78	1m 13s	36.46
butterfly	29.67	1m2s	32.07	1m 10s	16.04
head	32.66	1m3s	30.69	1m 15s	23.04
woman	36.61	1m5s	35.37	1m 22s	20.54

Table 5.10 Our Dataset Inpainting comparison

Image	DIP		Deep Decoder		Trained
	PSNR	Time	PSNR	Time	PSNR
autumn	20.48	1m16s	21.17	1m30s	18.67
astronaut	34.28	1m46s	34.91	3m12s	35.05
beach	30.26	1m2s	31.90	1m17s	30.93
couch	23.56	1m32s	23.32	1m50s	22.78
zebras	23.77	1m15s	20.92	2m4s	19.25
kate	36.61	1m46s	38.24	3m8s	41.83
lincolne	39.21	1m4s	38.59	1m30s	29.33
mickey	41.47	1m3s	41.18	1m25s	17.92
Old building	15.66	1m23s	15.40	2m17s	15.20
Old photo	37.63	1m25s	38.11	1m40s	26.24
powerpuff	32.85	1m27s	29.45	1m55s	16.20
Scan 1	27.77	1m4s	27.88	1m31s	23.77
Scan 2	24.85	1m6s	24.63	1m37s	15.54
F16	34.69	1m13s	29.06	3m25s	35.50
vase	30.43	1m6s	26.00	1m36s	29.25
mri	31.50	1m47s	30.60	2m43s	30.48

5.2.2 Qualitive Results

Figure 5.7 Scan 2 inpainting comparison

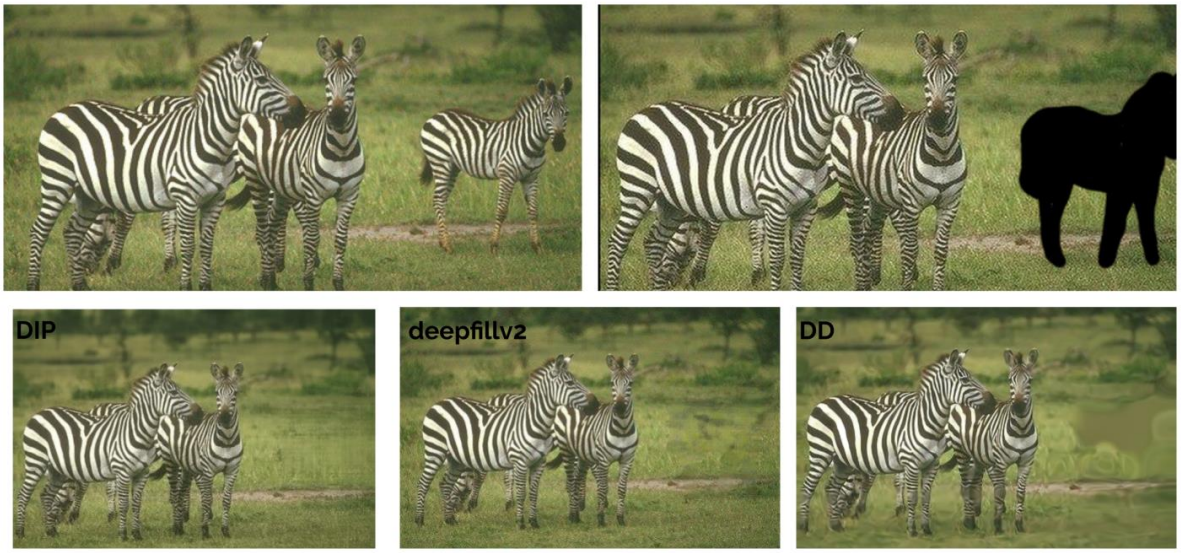


Figure 5.8 zebras inpainting comparison

5.2.3 Conclusion

Based on the results of our comparative analysis, Deep Image Prior (DIP) emerged as the most favorable untrained method for image inpainting. It consistently produced high-quality results that closely matched those of the trained method, making it a reliable alternative in most cases. On average, DIP achieved these results with a computation time of approximately 1 minute and 15 seconds per image, demonstrating a balance between performance and efficiency. These characteristics position DIP as a practical and effective choice for various inpainting scenarios, particularly when computational resources or training data are limited. This conclusion reinforces the potential of untrained neural networks to deliver competitive performance in image restoration tasks.

5.3 SUPER RESOLUTION

We perform a comparative analysis between our models and some popular state-of-the-art traditional and learned super resolution methods. The analysis was performed on the Set 5 and Set 14 standard test datasets.

For traditional methods, we used Bicubic Interpolation. Bicubic Interpolation is an image upscaling method traditionally used for super-resolution. It calculated new pixel values using a weighted average of the 16 nearest surrounding pixels in a 4x4 grid. It provides smoother results compared to other interpolation methods, but isn’t effective for highly detailed images.

For learned methods, we worked with the ESRGAN model, a deep learning-based method that leverages a GAN architecture. ESRGAN employs Residual-in-Residual Dense Blocks (RRDB) to extract features effectively while avoiding degradation. It excels in producing realistic and detailed high-resolution images.

5.3.1 Quantitative Result

Table 5.11 Set14 SR Comparison

Image	Bicubic	DIP		ESRGAN
	PSNR	PSNR	Time	PSNR
baboon	20.30	20.11	1m3s	20.92
barbara	23.57	23.68	1m36s	24.75
bridge	23.27	24.07	1m4s	24.12
coastguard	24.13	24.21	38s	24.83
comic	20.30	20.83	39s	22.14
face	28.82	29.50	35s	29.91
flowers	23.77	24.25	51s	26.22
foreman	27.45	28.41	47s	29.36
lenna	28.84	29.18	1m13s	30.42
man	24.36	24.57	1m6s	26.05
monarch	26.29	28.15	1m17s	31.21
pepper	28.83	29.24	1m2s	29.98
ppt3	20.30	22.45	1m26s	24.60
zebra	23.04	24.05	1m8s	25.89

Table 5.12 Set14 SR Comparison

Image	Bicubic	DIP		ESRGAN
	PSNR	PSNR	Time	PSNR
baby	30.45	29.35	1m12s	32.13
bird	28.43	29.22	51s	32.32
butterfly	21.41	24.37	48s	26.49
head	28.84	28.26	52s	29.94
woman	25.34	26.50	38s	29.12

5.3.2 Qualitative Result

Figure 5.9 Barbara SR Comparaison

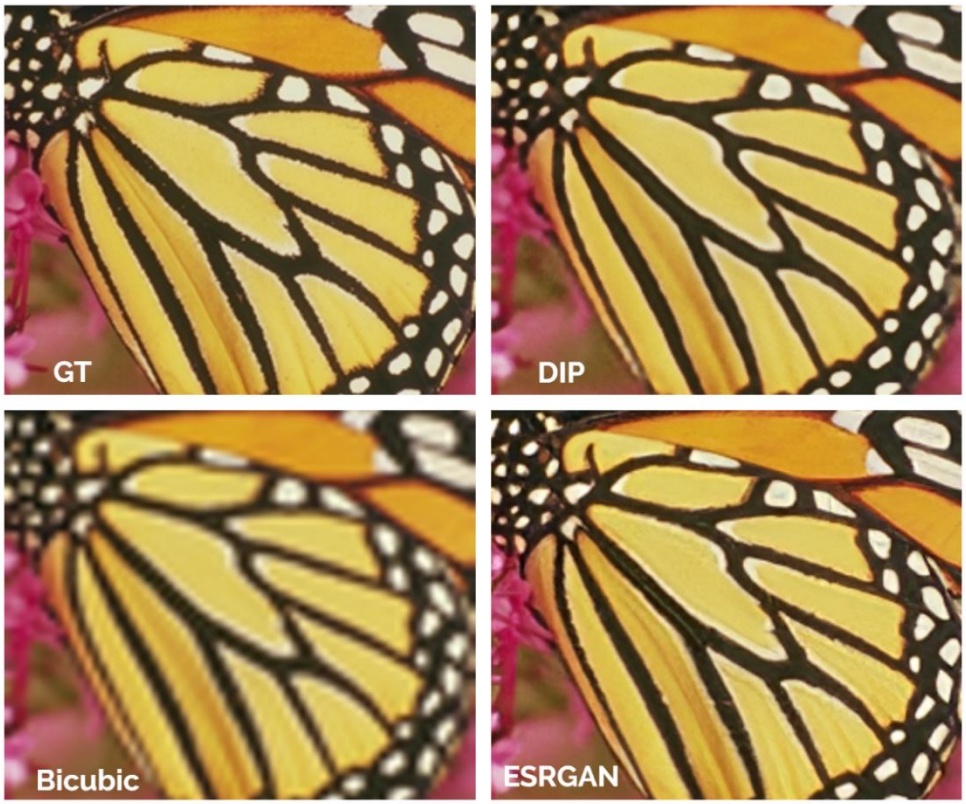


Figure 5.10 Butterfly SR Comparaison

5.3.3 Interpretation and Conclusion

Our analysis reveals that the DIP performs significantly well for super-resolution, although not as effectively as the ESRGAN. DIP, while promising, shows some limitations in this instance. While it does enhance the image beyond the basic Bicubic interpolation, it doesn't quite reach the level of detail and visual appeal achieved by ESRGAN.

Furthermore, it's worth noting that ESRGAN can sometimes generate images that appear even more visually appealing than the original ground truth. This highlights how GAN models go beyond simple reconstruction and introduce artistic enhancements or remove subtle imperfections present in the original image.

However, for applications where maintaining the original integrity of the image is critical, DIP becomes a better and safer choice. For instance, in medical imaging, where any external additions or artistic enhancements could lead to misinterpretation or diagnosis errors, DIP becomes a better choice.

Chapter Six: ConvDecoder for MRI Acceleration

6 CONVDECODER FOR MRI ACCELERATION

6.1 ACCELERATING MRI WITH UN-TRAINED NEURAL NETWORKS

Convolutional Neural Networks (CNNs) have shown great success in image reconstruction tasks, traditionally relying on large datasets for training. However, recent research has demonstrated that un-trained CNNs, such as Deep Image Prior (DIP) and Deep Decoder, can achieve strong performance without any external training data. These networks exploit their architectural structure as an implicit prior, allowing them to perform tasks like denoising, inpainting, and compressive sensing purely based on the input image.

While un-trained networks have been effective in controlled scenarios, their potential for real-world medical imaging applications, particularly in Magnetic Resonance Imaging (MRI), remains underexplored. MRI is a widely used diagnostic tool, but its long scan times pose a significant challenge. Compressed sensing techniques have been introduced to accelerate MRI by reconstructing images from a limited number of measurements. Investigating un-trained neural networks for MRI reconstruction offers a promising approach to further improving scan efficiency without relying on extensive training datasets.

6.1.1 The Philosophy of Compressed Sensing

- **Conventional Compression**: In typical image processing, the full image is captured first. It is then transformed (e.g., using wavelets), and only the most significant coefficients are kept for compression. This approach wastes resources by collecting all the data first and then discarding most of it.
- **Compressed Sensing**: CS revolutionizes this process by acquiring only the important coefficients from the start. This is achieved through two key ideas:
    1. **Transform Sparsity**: Most natural images are sparse in a certain domain (like wavelets or Fourier). This means that, once transformed, most of the coefficients are zero or near zero, and only a few carry significant information.
    2. **Incoherent Sampling**: Instead of collecting all the pixel values, a smaller number of incoherent measurements are taken, ensuring that enough information is captured for accurate reconstruction.

In MRI, two popular approaches for CS reconstruction are **l1-Norm Minimization** and **Total Variation (TV) Minimization**. Both methods exploit sparsity but do so in different ways, enhancing the reconstruction process by preserving important features while discarding redundant data.

6.1.1.1 L1 Norm Minimization

Compressed Sensing (CS) relies on the idea that images are sparse in certain transform domains, such as wavelet or Fourier domains. L1-Norm Minimization enforces this sparsity by promoting solutions where most coefficients are zero or near zero.

In CS-based MRI, the acquired k-space data y is modeled as:

$$y = Ax + n$$

where x is the image to be reconstructed, A is the undersampled Fourier encoding matrix, and n represents noise. The image x is assumed to be sparse in a transform domain Ψ, such as the wavelet domain, meaning α=Ψx is sparse. The reconstruction is achieved by solving the optimization problem:

$$\hat{x} = arg\ min \parallel \Psi x \parallel_1, subject\ to\ \parallel Ax - y \parallel_2 \leq \epsilon$$

where ‖Ψx‖ is the L1-norm, encouraging sparsity, and $\parallel Ax - y \parallel_2$ is the L2-norm, ensuring consistency with the acquired data. The parameter ϵ accounts for measurement noise.

In MRI reconstruction, L1-Norm Minimization is particularly effective in preserving fine details and edges, which are often sparse in the wavelet domain. It is widely used for reconstructing images from highly undersampled k-space data, allowing for faster imaging while maintaining high fidelity.

6.1.1.2 Total Variation (TV) Minimization

TV Minimization takes advantage of sparsity in the image's gradient domain, assuming that natural images consist of piecewise smooth regions with sparse edges. Instead of enforcing sparsity in a transform domain, TV Minimization promotes smoothness in the reconstructed image while preserving important edges.

The reconstruction process is formulated as:

$$\hat{x} = arg\ min\ TV(x), subject\ to\ \parallel Ax - y \parallel_2 \leq \epsilon$$

where the Total Variation is defined as:

$$TV(x) = \sum_{i,j} \sqrt{(\frac{\partial x}{\partial i})^2 + (\frac{\partial x}{\partial j})^2}$$

Here, $\frac{\partial x}{\partial i}$ and $\frac{\partial x}{\partial j}$ represent the image gradients along the horizontal and vertical directions. By minimizing this term, TV regularization reduces noise and artifacts while preserving sharp transitions, such as object boundaries and contours.

In MRI, TV Minimization is particularly beneficial for maintaining sharp edges and boundaries, which are crucial for clinical interpretation. It is also effective in reducing noise and artifacts, making it a valuable tool for high-quality image reconstruction.

6.2 CONVDECODER FOR MULTI-COIL MRI RECONSTRUCTION

A novel variation of Deep Image Prior (DIP) and Deep Decoder, named ConvDecoder, is proposed. This architecture, consisting of up-sampling, convolution, batch normalization, and ReLU blocks, demonstrates superior performance for knee MRI images and performs

comparably to DIP and Deep Decoder for brain MRI images, which typically contain less detail.

The study further establishes that untrained neural networks significantly outperform traditional untrained methods based on sparse and low-rank models. Comparisons are made with Total Variation (TV) minimization and ENLIVE, a representative low-rank-based method. These methods serve as natural baselines since they are widely used in practice, require no training data, and have demonstrated strong performance.

A key limitation of untrained neural networks is their relatively slow reconstruction speed due to the need for iterative network fitting. To address this, an initialization technique is introduced, accelerating reconstruction by a factor of 10.

Untrained methods are also compared with trained approaches using the fastMRI dataset, a benchmark for deep learning-based accelerated MRI. Despite the advantage of trained models in this dataset, untrained methods achieve performance comparable to the U-Net baseline and perform slightly worse than the state-of-the-art Variational Network (VarNet).

VarNet is a deep learning-based MRI reconstruction method that integrates MRI physics directly into the network architecture by iteratively refining the reconstruction through multiple cascades of CNNs and data fidelity steps. Unlike purely data-driven methods (like U-Net), VarNet explicitly incorporates MRI physics using the data consistency layer, leading to more accurate and reliable reconstructions. This design allows VarNet to effectively handle undersampled MRI data, making it one of the most powerful trained methods for MRI reconstruction.

To assess robustness to distribution shifts, trained and untrained methods are tested across different domains. Specifically, models trained on the fastMRI knee dataset are evaluated on the fastMRI brain dataset, where all methods experience performance degradation. Two strategies are introduced to mitigate this issue for untrained methods: meta-learning using a few test domain examples and an auto-tuning technique, enabling untrained networks to match VarNet's performance under distribution shifts. This highlights a critical advantage of untrained neural networks: enhanced robustness to out-of-distribution examples.

6.2.1 Problem Formulation for ConvDecoder-Based MRI Reconstruction

The objective is to reconstruct an image from undersampled k-space measurements in an accelerated multi-coil MRI setup. The acquired measurements follow the model:

$$y_i = MFS_i x + noise, \qquad i = 1, \ldots, nc$$

where S_i represents the coil sensitivity map, F is the 2D discrete Fourier transform (DFT), and M is the undersampling mask that determines which k-space frequencies are acquired and which are omitted. Specifically, if $M(k) = 1$, the k-space frequency at position k is retained (measured), whereas if $M(k) = 0$, the k-space frequency at position k is discarded (not measured). The number of receiver coils is denoted by n_c, and the k-space measurements from each coil are represented as y_i.

In a multi-coil MRI system, multiple receiver coils are placed around the body to acquire signals from different spatial locations. Each coil captures the same image but with different spatial weighting due to coil sensitivity variations. The sensitivity map S_i characterizes how

sensitive a particular coil is to different regions of the image. Coils positioned near specific anatomical structures have higher sensitivity in those areas and lower sensitivity elsewhere.

6.2.1.1 Single-Coil Reconstruction

To illustrate how images can be reconstructed using an untrained neural network, we first consider the case of single-coil reconstruction. In this scenario, the k-space measurement is denoted as $y \in C^M$, where M represents the undersampling mask and F is the discrete Fourier transform. Additionally, the sensitivity map in this case is the identity matrix, meaning that no spatial weighting is applied to the image by the coil.

An untrained neural network $G: R_p \rightarrow R^{c \times w \times h}$with learnable parameters C is used to estimate the image from the given measurements y. The reconstruction process involves two steps. First, the optimal parameter set C is obtained by minimizing the mean-squared loss function:

$$L(C) = \frac{1}{2}|y - MFG(C)|_2^2$$

using an iterative first-order optimization method. Once the optimized parameters C^* are found, the final image estimate is given by $x = G(C^*)$.

Since MRI images contain both real and imaginary components, they are represented using two channels in the network, c=2. The first channel stores the real part of the complex image, which represents the primary intensity information, while the second channel stores the imaginary part, which encodes the phase. The phase information is essential in MRI, as it helps preserve spatial relationships and improve the accuracy of image reconstruction.

6.2.1.2 Multi-Coil Accelerated MRI Reconstruction

In multi-coil MRI imaging, multiple receiver coils capture different k-space measurements of the same underlying image. This redundancy enables various reconstruction techniques using untrained methods. One method is particularly effective for brain images, while the other performs better for knee images. The first approach reconstructs the image using estimated coil sensitivity maps, whereas the second method operates without requiring sensitivity map estimates.

Reconstruction Without Coil Sensitivity Maps

One effective approach, introduced by Arora et al., reconstructs all coil images simultaneously by enforcing a shared structure across them. This method does not rely on sensitivity maps and instead treats the reconstruction as a joint process, where a single untrained neural network generates all coil images at once. Each coil's image is represented by two channels (real and imaginary parts), and the final image is obtained by combining them using the Root-Sum-of-Squares algorithm. This approach ensures consistency across the reconstructed images while improving computational efficiency and reconstruction quality.

The network produces a stack of images, where each coil's image is represented by two output channels, one for the real part and one for the imaginary part. This ensures that all coil images follow the same underlying pattern, helping the network learn common features across them.

The reconstruction is done by optimizing a loss function that minimizes the difference between the predicted and actual k-space measurements for each coil:

$$L(C) = \frac{1}{2}|y - MFG(C)|_2^2$$

Here, $G(C)$represents all the reconstructed coil images, and $G_i(C)$is the image for the i-th coil. By sharing the same network for all coils, this approach produces better-quality reconstructions and runs much faster than reconstructing each coil separately.

Reconstruction With Sensitivity Map Estimates

Instead of reconstructing individual coil images separately and then combining them, this method directly reconstructs the final image while incorporating estimated coil sensitivity maps. The sensitivity maps S_i are estimated **only from the available undersampled k-space data** especially from the ACS region, if present. The ACS region consists of fully sampled central k-space lines, typically located in the low-frequency region, where most of the image's energy is concentrated. ESPIRiT, an eigenvalue-based method for sensitivity estimation, is used to compute these maps by analyzing the correlation between coil images in k-space. This method identifies the dominant subspace of the signal and extracts smooth, spatially varying sensitivity profiles for each coil.

With these estimated sensitivity maps, the reconstruction loss function is formulated as:

$$L(C) = \frac{1}{2}\sum_{i=1}^{n_c}|y - MFS_iG(C)|_2^2$$

Enforcing Data Consistency

Once the network is trained by minimizing the respective loss function, a data consistency step is applied. The k-space measurement consists of undersampled frequency components of the original image at specific positions in the frequency domain. After reconstructing the image, its Fourier representation is updated by replacing the known k-space values with the actual acquired measurements. This step ensures that the reconstructed image remains consistent with the measured data while still benefiting from the learned prior information.

6.2.2 ConvDecoder Architecture

The ConvDecoder is a variation of the Deep Decoder architecture, designed for image reconstruction tasks. Unlike the Deep Decoder, which relies on bi-linear upsampling and 1×1 convolutions, the ConvDecoder uses nearest-neighbor upsampling and 3×3 convolutional layers, allowing it to capture more spatial details.

The ConvDecoder follows a structured design where each layer (except the last one) consists of:

- **Nearest-Neighbor Up-Sampling**: This operation increases the spatial resolution of the feature maps. Unlike the Deep Decoder, which uses bilinear upsampling,

ConvDecoder relies on nearest-neighbor interpolation,which simply copies the nearest pixel's value. This makes it computationally efficient while preserving sharp edges in the upsampled feature maps.

- **3×3 Convolutional Layers**: These layers capture local spatial information, refining features extracted from the input.
- **ReLU Activation Function**: A non-linear activation function that introduces sparsity and enhances feature learning.
- **Batch Normalization (BN)**: BN normalizes each channel independently, stabilizing the training process and improving convergence.

The final layer does not include an up-sampling operation. Instead, it consists of a 1×1 convolutional layer that linearly combines the output channels to generate the final reconstructed image.

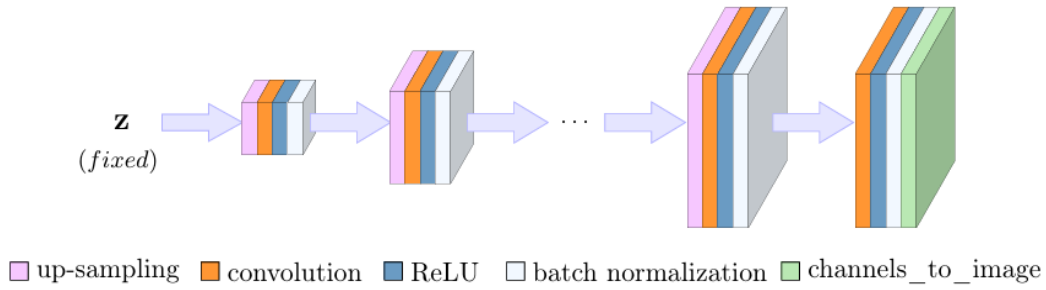


Figure 6.1 ConvDecoder Architecture

6.2.2.1 Network Input and Initialization

The ConvDecoder starts with a fixed input tensor, denoted as $z \in R^{c_0 \times w_0 \times h_0}$, which is drawn from a Gaussian distribution N(0,I), and remains unchanged throughout the reconstruction process. The input has a dimension of 256×10×5, where 256 represents the number of channels

The network consists of (n+1) layers, with the final output being an image $c \in R^{c_{n+1} \times w_{n+1} \times h_{n+1}}$. The default configuration includes 8 layers (including the last one), with 256 channels per layer. The fixed random input, combined with the network's structured design, enables it to reconstruct images from undersampled measurements by progressively refining details across layers.

6.2.2.2 Role of Up-Sampling in Image Reconstruction

Each layer in the ConvDecoder plays a role in building up the final image. To understand this, we can look at what each layer produces during reconstruction. Since each layer has many feature channels (e.g., 256), instead of viewing them all separately, we can combine them in a way that best matches the ground-truth image at that layer's resolution. If a layer outputs a feature map of size $w_i \times h_i$, we resize the ground-truth image to this same size for comparison.

The upsampling layers help the network build the image step by step, with earlier layers capturing the coarse structure and later layers refining finer details.

6.2.2.3 How Network Width Affects Reconstruction Quality

The width of the ConvDecoder, defined as the number of channels per layer, plays a crucial role in determining the quality of image reconstruction. Choosing an appropriate width ensures that the network has enough capacity to capture details without introducing unnecessary complexity.

The network width directly affects how much spatial and structural information is preserved during reconstruction. If the network has too few channels, it struggles to capture fine details, resulting in blurry reconstructions. On the other hand, an excessive number of channels can lead to overfitting, where the network learns noise rather than meaningful patterns, ultimately degrading the reconstruction quality.

The authors of the paper conducted an experiment to evaluate the effect of network width on reconstruction performance. They tested a seven-layer ConvDecoder on three undersampled knee MRI measurements from the fastMRI dataset. The Structural Similarity Index (SSIM) was used to measure the quality of reconstructions.

Key observations from the experiment:

- Very small or very large network widths resulted in poor performance.
- A width of around 200 channels per layer consistently produced the best SSIM scores across different images.
- The experiment was repeated four times to ensure consistency, and the results showed that the optimal width remained stable across different runs.

6.2.2.4 Network Parameter Selection

The ConvDecoder architecture allows for variations in the number of layers and channels per layer. The authors performed a grid search to select the best parameters. The default configuration consists of 8 layers with 256 channels per layer, a kernel size of 3, and no pooling operations (pool = 0). Different setups were tested for various MRI datasets:

- **Brain MRI reconstruction:** A network with 5 layers and 64 channels per layer performed best.
- **Knee MRI reconstruction:** A network with 8 layers and 256 channels per layer yielded optimal performance.

6.3 CONSIDERATIONS FOR EVALUATING RECONSTRUCTION PERFORMANCE

Evaluating MRI reconstruction performance is complex due to multiple challenges. The following key considerations were addressed:

6.3.1 Image Comparison Metrics

Standard image quality metrics, such as Peak Signal-to-Noise Ratio (PSNR), are often inadequate for assessing MRI reconstruction quality. Visual Information Fidelity (VIF) better correlates with diagnostic quality (most relevant by cliniques) compared to widely used metrics like PSNR and Structural Similarity Index (SSIM). However, higher VIF score sometimes corresponds to a visibly worse performance. To ensure a comprehensive evaluation, we report performance using: VIF, PSNR, SSIM, Multi-Scale SSIM (MS-SSIM).

6.3.2 Normalization

Normalization is essential for fair image comparisons. Different normalization techniques significantly impact extracted texture features in medical imaging. Experiments were conducted with three normalization techniques:

- Min-max normalization: that transforms image I to $\frac{I-\min(I)}{\max(I)-\min(I)}$
- Mean-standard deviation (std) normalization on both ground truth and the reconstructed image, that transforms image I to $\frac{I-\text{mean}(I)}{std(I)}$
- Mean-std normalization that is only applied to the ground truth image, using the mean and the standard deviation of the reconstructed image to match the histogram of the ground truth image to the histogram of the reconstructed image.

Based on experiments were conducted on 200 mid-slice multi-coil knee images (4x accelerated) the mean-std normalization that applied to the ground truth image was chosen as normalization technique.

6.3.3 Comparison to Noisy Ground Truth

In some cases, ground-truth images contain measurement artifacts, which can obscure reconstruction performance evaluations. A high-quality reconstruction may still receive a suboptimal score if the reference image is corrupted.

6.3.4 Volume- vs. Image-Based Comparison

When evaluating MRI reconstruction, results can be assessed per image (slice) or as a full 3D volume:

Image-based evaluation: Evaluating each 2D slice of the MRI separately. Metrics are computed for individual slices. Each scan consists of a number of slices, each slice is an image.

Volume-based evaluation: Evaluating the reconstructed MRI as a complete 3D volume. Metrics are computed over the entire 3D scan (multiple slices). Each slice is an image and those images together form a volume.

All the metrics are sensitive to whether the comparison is done on an image or volume level. Experiments showed that the image-based score computation results in a lower range of numbers compared with the volume-based.

6.4 EVALUATING RESULTS

Experiments were conducted to compare the performance of the ConvDecoder with seven methods. Five Untrained methods: ConvDecoder, DIP, Deep Decoder, Total Variation (TV) norm minimization, and ENLIVE (a calibration-less untrained method), and two trained methods: U-Net, and End-to-End Variational Network (VarNet). The evaluation was conducted on the 4x accelerated multi-coil knee measurements, on 8x accelerated multi-coil knee measurements, and on 4x accelerated multi-coil brain measurements.

A grid search was conducted over 10 randomly-chosen training images to optimize the hyperparameters of each untrained neural network (ConvDecoder, DIP, Deep Decoder) for fair comparison.

6.4.1 Evaluation of 4x Accelerated Multi-Coil Knee Measurements

6.4.1.1 Untrained Networks Comparison

First, the comparison was between the ConvDecoder with DIP, and Deep Decoder by averaging over 20 randomly-chosen mid-slice images from different MRI volumes in the validation set. From experiments, it can be seen that:

- DIP introduced noticeable vertical artifacts in the reconstructed images.
- Deep Decoder produced overly smooth images, leading to a loss of texture details.
- ConvDecoder outperformed both methods for knee MRI images, providing the best results among the three untrained networks.

6.4.1.2 Comparison with Trained Models

Subsequent experiments focused on comparing ConvDecoder with established methods: U-Net, VarNet, and ENLIVE.

- **U-Net Training:** A standard U-Net with 8 layers and width factor = 32 was trained using 974 MRI volumes.
- **Variational Network Training:** A 12-cascade end-to-end Variational Network was trained with width factor = 18.
- **Evaluation Setup:** The models were tested on 200 validation set images, each selected as a mid-slice from different MRI volumes. Randomized masks were applied during each run to ensure robust evaluation.

The results show that:

- ConvDecoder achieved a higher VIF score than both U-Net and VarNet (VIF is a metric deemed clinically relevant).
- On other metrics, ConvDecoder performed comparably to U-Net.
- ConvDecoder significantly outperformed TV and ENLIVE, making it a more effective untrained approach.
- Despite ConvDecoder's strong performance, VarNet remained the top-performing model on most evaluation metrics.

6.4.2 Evaluation of 8x Accelerated Multi-Coil Knee Measurements

The effectiveness of ConvDecoder was further assessed on 8x accelerated knee MRI scans. Given the increased acceleration factor, new hyperparameters were selected through another grid search, leading to an optimized ConvDecoder architecture.

Results from experiments:

- ConvDecoder's performance remained comparable to U-Net on most metrics.
- U-Net slightly outperformed ConvDecoder in SSIM and MS-SSIM, but the difference was marginal.
- ConvDecoder significantly outperformed TV and ENLIVE in all evaluated aspects.

6.4.3 Evaluation of 4x Accelerated Multi-Coil Brain Measurements

To validate the findings across different anatomical regions, untrained neural networks were tested on **brain MRI images**.

- Unlike knee images, all three untrained networks (ConvDecoder, DIP, Deep Decoder) performed similarly on brain MRI scans.
- A new data consistency step improved SSIM scores by 8%.
- Since there were no major differences among the three untrained methods, **ConvDecoder was chosen for further comparison** to ensure consistency.

6.5 ACCELERATION OF UNTRAINED NETWORKS 10X FASTER

Un-trained neural networks are very flexible for image reconstruction tasks, but with a drawback: their computational cost considered as higher than trained neural networks. The reason is that un-trained networks rely on iteratively solving an optimization problem, whereas trained networks perform a simple forward pass through learned parameters. To counter this inefficiency, a domain-specific initialization method was proposed that accelerates the optimization by a factor of 10.

For iterative optimization problems, initialization affects the time it takes to get a solution of desired accuracy. The acceleration approach is based on starting from a good initialization.

MRI knee scans can be classified into two categories:

- **Fat-suppressed images:** These remove the bright signal from fat tissues, making it appear dark and allowing other tissues to be seen more clearly (Fat appears dark because a special MRI technique removes its signal.)
- **Non-fat-suppressed images**: These show all tissues, including fat (Fat appears bright because it naturally gives off a strong MRI signal), in their natural brightness.

Each category exhibits distinct but internally consistent statistical properties, including pixel intensity distributions, contrast levels, and histograms. It turns out that a decoder optimized

to reconstruct a non-fat-suppressed image provides a good initialization for other non-fat-suppressed images.

6.5.1 Implementation of the initialization strategy

First, a random non-fat-suppressed knee slice and fit the ConvDecoder until full convergence based on the loss function $L(C) = \frac{1}{2}\sum_{i=1}^{nc}||Yi - MFGi(C)||^2_2$ to reconstruct the selected image from its 4x under-sampled measurements. The trained decoder parameters are stored as a set, denoted as C1. C1 is used as an initialization to reconstruct other non-fat-suppressed images using ConvDecoder.

6.5.2 Results

This approach achieves the same reconstruction accuracy in 10 times less iterations with no performance loss (60 minutes were reduced to 6 minutes). This result shows that only one sample gives sufficient information to represent a specific category of images.

6.6 BETTER PERFORMANCE AT THE COST OF MORE COMPUTATIONS

ConvDecoder significantly outperforms Total Variation (TV) and achieves performance close to U-Net. However, to further enhance the reconstruction quality, an additional method that improves ConvDecoder's performance at the expense of increased computational cost was proposed.

Instead of using a single ConvDecoder instance for image reconstruction, an ensemble averaging technique was used that running **multiple ConvDecoders (k instances) with the same hyperparameters but different random initializations**. The idea is that each ConvDecoder independently reconstructs the same under-sampled MRI image (denoted as **y**), and after all k models have completed their optimization, their **reconstructed outputs are averaged** to form the final image. The reasoning behind it is:

- Each ConvDecoder instance converges to a slightly different solution due to its independent initialization.
- Averaging the outputs reduces noise and **improves structural similarity and peak signal-to-noise ratio (PSNR)**.
- This results in **higher reconstruction accuracy** compared to using a single ConvDecoder instance.

6.6.1 Results

To assess the effectiveness of this approach, an experiment was conducted using **20 independent ConvDecoder runs**, applying the averaging technique to compute the final reconstructed image. The evaluation is performed on **18 randomly selected knee MRI images** from the **fastMRI validation dataset**, including:

- **9 proton-density weighted images** (Proton-density weighting (PD-weighting) produces images where the contrast mainly depends on the number of hydrogen protons in a given tissue), and
- **9 fat-suppressed images**.

The results, are that the averaging method slightly exceeds U-Net and produces performances near those of the end-to-end Variational Network (VarNet). The implication is that, even though ConvDecoder is an untrained network, it can nearly match the efficiency of fully supervised deep learning models if multiple runs are used and the results thus obtained are averaged.

Although the numerical metrics (PSNR, SSIM) show an improvement, the **visual difference between reconstructions with and without averaging is small**.

Even averaging over the output of two decoders has a noticeable impact (more than 0.5 dB) on the reconstruction quality. Note that the improvement rate decreases as we increase the number of decoders.

Chapter Seven: PRIORI – Experimental Development and Evaluation

7 PRIORI – EXPERIMENTAL DEVELOPMENT AND EVALUATION

In this chapter, we present our experimental work on improving MRI reconstruction using the ConvDecoder framework. We begin by outlining the experimental setup used to assess reconstruction quality under various conditions. Building on the original ConvDecoder architecture, we introduce our own modifications to enhance reconstruction quality. We call the final model PRIORI – Priori-based Optimized Reconstruction for MR Imaging

Through this investigation, we aim to demonstrate both the potential and the limitations of ConvDecoder-based approaches, and to contribute insights toward more effective reconstruction strategies in data-limited MRI settings.

Link to the full-resolution result images is available here for better image clarity.

Figure 7.1 QR Code: Results Drive

7.1 BRAIN MRI EXPERIMENTS

A series of experiments was carried out to determine the most effective configuration for the ConvDecoder model in the context of brain image reconstruction from undersampled k-space data. The aim was to balance reconstruction quality with computational efficiency and model stability. After iterative testing, the network was configured with 64 feature channels per layer. This number was found to provide sufficient capacity to capture relevant structural and textural information without introducing unnecessary computational overhead or longer reconstruction time. The model architecture included five convolutional layers, as deeper configurations beyond this point led to minimal improvements while increasing training time and instability.

All layers were assigned a stride of 1, ensuring no downsampling occurred within the network. This choice preserved spatial resolution throughout the decoding process, which is critical for maintaining fine structural details in brain images. The latent input size was set to [8, 4], providing an effective trade-off between model flexibility and control over parameter count. Experiments showed that increasing this size to values like [10, 5] introduced a higher risk of overfitting and resulted in line artifacts in the final image.

A kernel size of 3 was selected for all convolutional layers, a standard choice in image reconstruction tasks. The network's output was configured with a depth of 2 to represent the real and imaginary components of the reconstructed image, in line with the complex-valued nature of MRI data. Finally, the output size was matched directly to the shape of the input k-space slice, ensuring consistent resolution and allowing for direct comparison between the reconstructed and ground truth images in both image and frequency domains.

7.1.1 Sensitivity Map Estimation: SigPy’s ESPIRiT vs. BART’s ecalib

In parallel MRI reconstruction, sensitivity maps describe how each receiver coil in the MRI system "sees" the imaged object with spatially varying intensity and phase. Accurate estimation of these maps is essential for high-quality image reconstruction. Two commonly used tools for this purpose are SigPy’s ESPIRiT and BART’s ecalib.

7.1.1.1 SigPy’s ESPIRiT

SigPy implements the traditional ESPIRiT algorithm, which estimates coil sensitivity maps using a standard calibration process. It begins by selecting a small, fully sampled area from the center of k-space called the Auto-Calibration Signal region. This region contains the crucial low-frequency information used to calibrate the coil sensitivities. A sliding window moves across the ACS region, collecting small overlapping neighborhoods of k-space points from each coil. These collected values are stacked to form a calibration matrix, capturing local spatial relationships across coils.

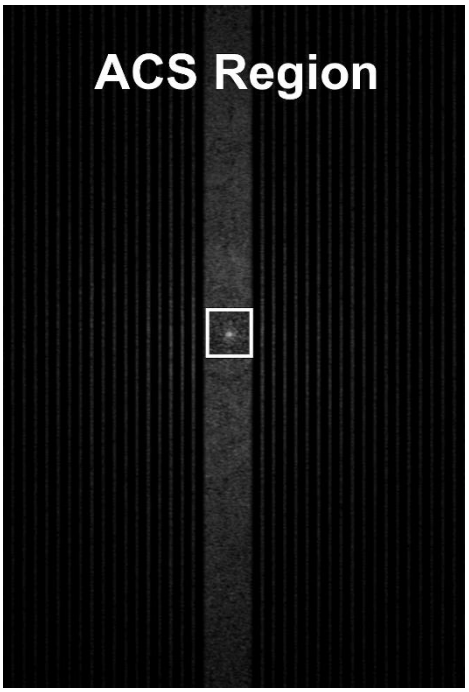


Figure 7.2 Central ACS Region in Undersampled k-Space Data

Next, Singular Value Decomposition (SVD) is applied to identify the principal signal components that are consistently shared across the different coils. Based on these components, ESPIRiT solves an eigenvalue problem, and the dominant eigenvector corresponding to an eigenvalue close to one is selected as the estimate of the coil sensitivity map. Finally, the maps are normalized so that their combined strength remains consistent across the image.

While ESPIRiT is effective, it requires manual specification of the ACS region. It can struggle to maintain accuracy at high acceleration factors if the ACS region is too small, potentially impacting the quality of the reconstructed images.

7.1.1.2 BART's ecalib (ESPIRiT-based)

BART's ecalib method also estimates coil sensitivity maps using ESPIRiT principles but includes several enhancements. Similar to SigPy, it uses the fully sampled ACS region for calibration but it offers the option to either manually specify this region or automatically detect it.

When automatic detection is enabled, ecalib scans the k-space data to identify the largest continuous region around the center of k-space that contains fully sampled lines. This is based on detecting regions where all coils have non-zero values across the same set of k-space lines. This ensures that the ACS region used for calibration has enough reliable low-frequency data, without requiring the user to define it manually.

After selecting the ACS region, ecalib builds a calibration matrix in a similar way to ESPIRiT. However, instead of a one-shot estimation, it uses an iterative regularization process based on nonlinear inversion. This improves robustness, especially with noisy or highly undersampled data. Additionally, it includes advanced phase alignment methods to correct for coil-to-coil phase differences.

BART was selected as the preferred sensitivity map estimation tool in this setup, as it produced clearer, more reliable, and artifact-free sensitivity maps than ESPIRiT in brain MRI reconstruction with ConvDecoder. Its advanced calibration and phase correction consistently improved image quality throughout the reconstruction process.

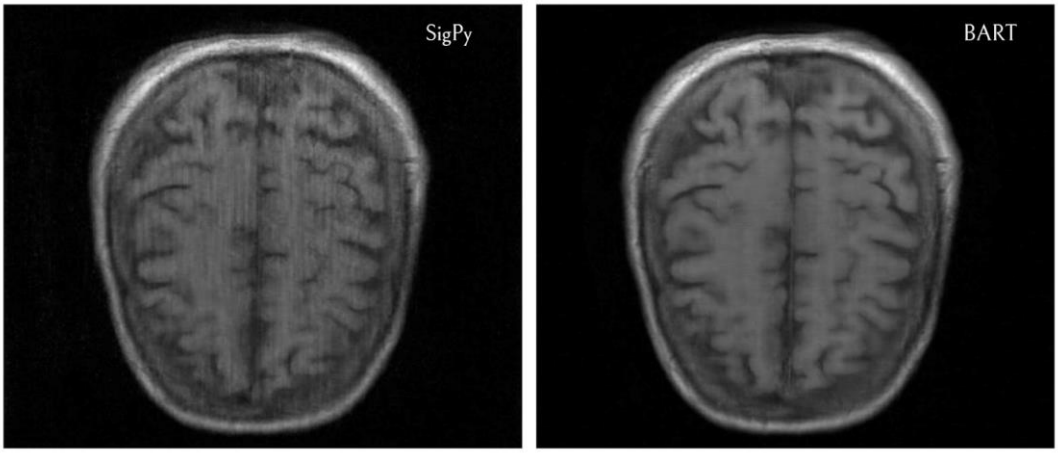


Figure 7.3 Comparison of SigPy and BART ecalib Outputs on a T1-Weighted Image, Showing Cleaner Reconstruction with BART

7.1.2 Initialization for Brain Reconstruction

For MRI brain image reconstruction experiments, multiple images were evaluated as potential candidates for network initialization. The goal was to identify an image that, when used to pre-train the network and save both its weights and input tensor, would provide an effective starting point for reconstructing other images. It was observed that images with lower noise levels and clearer anatomical structures offered better initializations, allowing the network to converge faster and produce higher-quality results with fewer iterations. By saving the weights and input tensor after training on a suitable image, subsequent reconstructions could start from a pre-trained, meaningful representation rather than from random weights and a fresh input tensor. During this process, only the layers with matching shapes between the saved checkpoint and the target model were updated. This ensured compatibility when

reconstructing images with fewer coils, such as 4 or 12, while preserving important learned representations from the initialization image.

7.1.2.1 T1-Weighted Image Initialization

For the T1-weighted brain image reconstruction experiments, file_brain_AXT1_201_6002828 proved to be the most effective. Its network weights and input tensor, saved after 20,000 iterations, consistently provided reliable starting points for other reconstructions. Extending the training beyond 20,000 iterations resulted in diminished performance when used as an initialization for new images, likely due to overfitting to the initial image. Experiments also compared saving weights after training the image with and without the sensitivity map, and incorporating the sensitivity map consistently yielded better reconstruction outcomes and higher evaluation metrics. In some cases, VIF increased by 1 point when using sensitivity map initialization, confirming its importance in preserving coil sensitivity information and improving reconstruction performance.

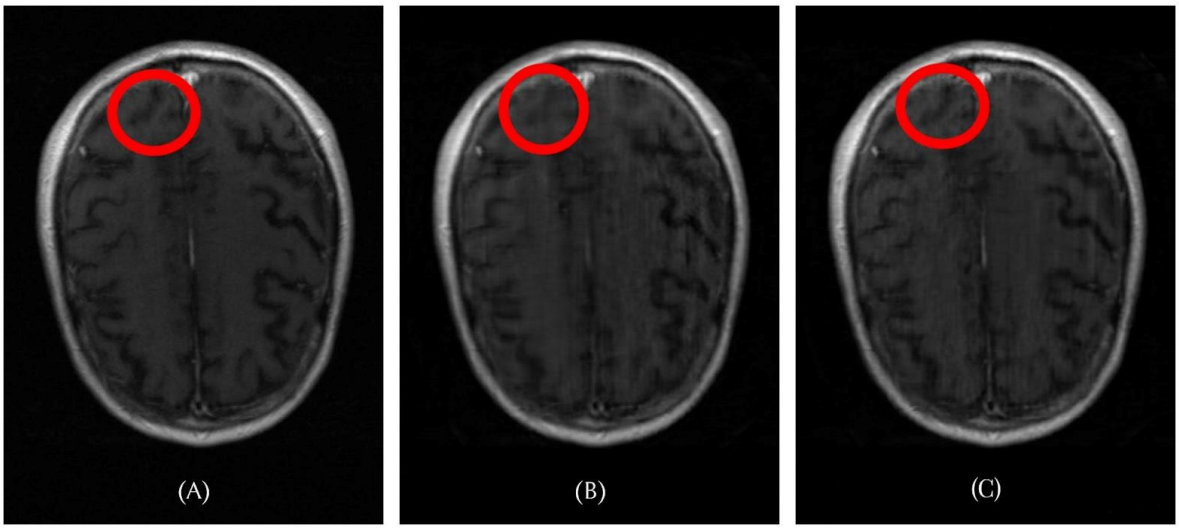


Figure 7.4 Comparison of Reconstructions With and Without Sensitivity Map for Weight Initialization

The previous figure shows three images **A:** Ground Truth, **B:** Without Sensitivity Map Initialization (VIF: 0.92, PSNR: 37.39), **C:** With Sensitivity Map Initialization (VIF: 0.93, PSNR: 38.09).

7.1.2.2 T2-Weighted and FLAIR Image Initialization

We experimented with different initializations for T2-Weighted brain image reconstruction. The final initialization is taken after optimizing the network for the image file_brain_AXT2_206_6000222, after 20,000 iterations and with a learning rate 0.01.

The saved weights were then used as a starting point for other reconstructions. We noticed that most images converge to good results in 10,000 to 12,000 iterations. Compared to random initialization, using these optimized weights reduced convergence time while preserving image quality.

For FLAIR brain images, we performed similar experiments and found that using the same T2-weighted initialization yields good results. Despite modality differences, the structural similarities between T2-weighted and FLAIR brain images make this initialization transferable and effective.

7.1.3 Image-Domain Reconstruction Ensemble Experiments

To investigate potential performance gains through image-domain fusion, several experiments were conducted by combining the reconstructed outputs from different models. The primary objective was to assess whether merging ConvDecoder's reconstructions with other techniques could enhance image quality, particularly under challenging conditions with noisy ground truth and k-space measurements.

7.1.3.1 Experiment 1: ConvDecoder + GRAPPA

In the first experiment, the reconstructed outputs from ConvDecoder and GRAPPA were combined in the image domain to assess whether their complementary characteristics could improve overall image quality. This approach demonstrated improved performance on images containing relatively smooth structures, likely due to the ability of both methods to preserve low-frequency information. However, in cases where the ground truth images and k-space measurements were inherently noisy, this combination underperformed. The noise present in the GRAPPA reconstruction propagated through the combination process, resulting in images with elevated noise levels and degraded visual quality, particularly in areas of high spatial frequency or in images already compromised by acquisition noise.

7.1.3.2 Experiment 2: ConvDecoder + U-Net

The second approach involved combining the reconstructed outputs of ConvDecoder and a U-Net model in the image domain. A series of weighted averages between the two reconstructions was evaluated to determine the optimal contribution from each model. This method effectively balanced the preservation of fine structural details with noise suppression, either improving the Visual Information Fidelity (VIF) score or maintaining it at a level comparable to the highest-performing individual model.

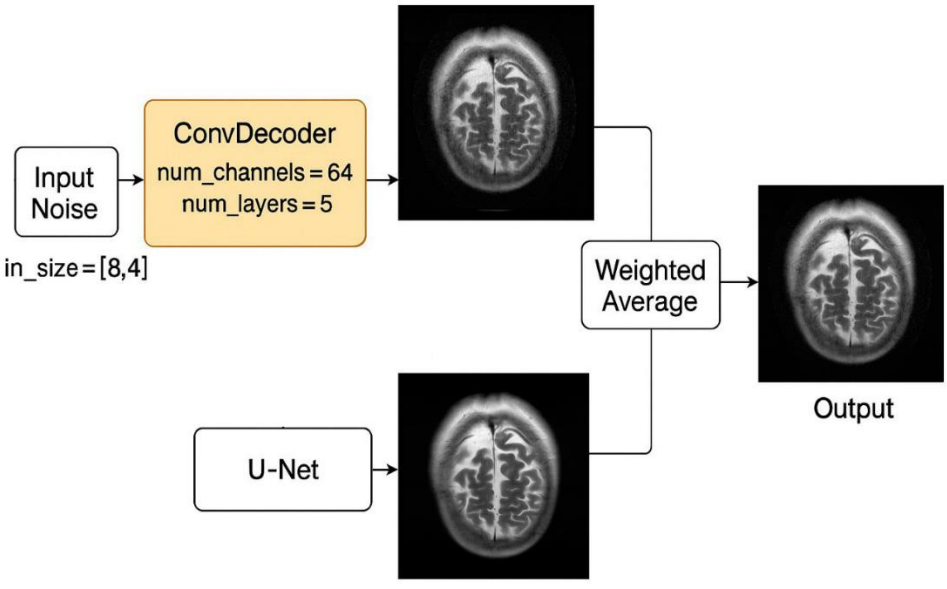


Figure 7.5 Architecture of the ConvDecoder and U-Net Ensemble

The main advantage U-Net contributed was its ability to enhance the sharpness of brain folds and the outer brain boundary, while also reducing noise more effectively in flat, homogeneous areas. However, its tendency to aggressively smooth these uniform regions sometimes removed subtle texture details that ConvDecoder preserved better. By combining both models,

the reconstruction benefited from U-Net's strengths, while retaining ConvDecoder's natural contrast and texture fidelity.

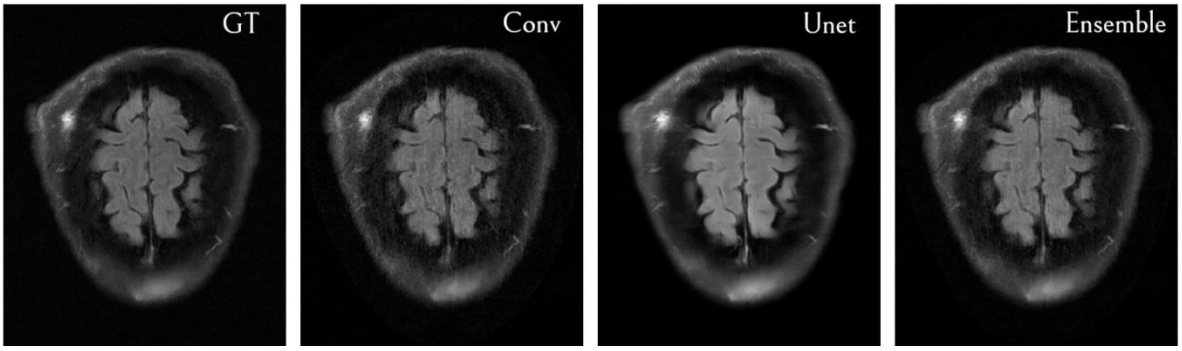


Figure 7.6 Comparison of ConvDecoder and U-Net Reconstruction

7.1.4 Brain Comparative Analysis

To evaluate the performance of the proposed ConvDecoder model on brain MRI data, we conducted a series of comparative experiments, using the Google Colab environment with a T4 GPU. The ConvDecoder network was initialized using the pre-trained weights as described above, and the reconstruction is performed with sensitivity map estimates.

7.1.4.1 Quantitative results

Table 7.1 Scores of the ConvDecoder, DIP, and DD models on Brain Images

	ConvDecoder-SM			***DIP***			***DD***		
	VIF	**SSIM**	**PSNR**	**VIF**	**SSIM**	**PSNR**	**VIF**	**SSIM**	**PSNR**
AXFLAIR_201_6002974	0.84	0.84	31.60	0.77	0.79	30.54	0.74	0.84	30.44
AXFLAIR_201_6002979	0.78	0.84	33.39	0.73	0.81	33.32	0.68	0.87	33.75
AXFLAIR_201_6002910	0.90	0.88	34.28	0.86	0.89	34.32	0.83	0.90	34.21
AXFLAIR_201_6002872	0.87	0.82	30.78	0.84	0.79	30.67	0.80	0.84	30.87
AXT1PRE_203_6000645	0.89	0.90	36.06	0.84	0.87	34.37	0.84	0.89	34.69
AXT1_202_2020590	0.93	0.95	38.14	0.91	0.94	12.64	0.90	0.94	9.98
AXT1_202_2020438	0.92	0.91	36.65	0.86	0.91	35.66	0.85	0.92	35.84
AXT1PRE_209_6001221	0.85	0.89	35.43	0.79	0.89	34.37	0.77	0.88	33.72
AXT2_200_2000161	0.94	0.96	38.41	0.89	0.95	34.75	0.84	0.94	33.11
AXT2_203_2030014	0.89	0.93	35.54	0.79	0.91	31.36	0.70	0.87	29.48
AXT2_202_2020142	0.93	0.93	37.23	0.83	0.93	34.74	0.84	0.94	35.57
AXT2_209_6001469	0.89	0.90	34.17	0.79	0.89	31.40	0.76	0.88	31.27

Table 7.2 Scores of the ConvDecoder vs. Trained Methods on Brain Images

	ConvDecoder-SM			***U-Net***			***Varnet***		
	VIF	**SSIM**	**PSNR**	**VIF**	**SSIM**	**PSNR**	**VIF**	**SSIM**	**PSNR**
AXFLAIR_201_6002974	0.84	0.84	31.60	0.80	0.84	31.75	0.96	0.91	36.12
AXFLAIR_201_6002979	0.78	0.84	33.39	0.75	0.88	34.74	0.89	0.91	37.37
AXFLAIR_201_6002910	0.90	0.88	34.28	0.87	0.90	34.08	0.98	0.93	38.17
AXFLAIR_201_6002872	0.87	0.82	30.78	0.85	0.82	30.95	0.97	0.89	34.96
AXT1PRE_203_6000645	0.89	0.90	36.06	0.89	0.92	35.83	0.91	0.93	37.36
AXT1_202_2020590	0.93	0.95	38.14	0.94	0.95	36.56	0.95	0.97	40.53
AXT1_202_2020438	0.92	0.91	36.65	0.93	0.94	36.99	0.93	0.95	39.85
AXT1PRE_209_6001221	0.85	0.89	35.43	0.87	0.90	35.25	0.92	0.92	37.07
AXT2_200_2000161	0.94	0.96	38.41	0.89	0.95	33.13	0.99	0.98	42.13
AXT2_203_2030014	0.89	0.93	35.54	0.90	0.94	34.22	0.97	0.97	39.21
AXT2_202_2020142	0.93	0.93	37.23	0.94	0.95	36.71	0.98	0.97	41.78
AXT2_209_6001469	0.89	0.90	34.17	0.89	0.92	33.99	0.97	0.94	37.76

Table 7.3 Scores of the ConvDecoder vs Ensemble Method on Brain Images

	ConvDecoder-SM			***Enhanced ConvDecoder-SM***		
	VIF	**SSIM**	**PSNR**	**VIF**	**SSIM**	**PSNR**
AXFLAIR_201_6002974	0.84	0.84	31.60	0.83	0.85	32.95
AXFLAIR_201_6002979	0.78	0.84	33.39	0.78	0.84	34.90
AXFLAIR_201_6002910	0.90	0.88	34.28	0.89	0.89	35.66
AXFLAIR_201_6002872	0.87	0.82	30.78	0.87	0.82	32.23
AXT1PRE_203_6000645	0.89	0.90	36.06	0.89	0.91	36.89
AXT1_202_2020590	0.93	0.95	38.14	0.94	0.95	39.03
AXT1_202_2020438	0.92	0.91	36.65	0.93	0.93	38.06
AXT1PRE_209_6001221	0.85	0.89	35.43	0.86	0.90	36.16
AXT2_200_2000161	0.94	0.96	38.41	0.93	0.97	38.19
AXT2_203_2030014	0.89	0.93	35.54	0.90	0.94	36.42
AXT2_202_2020142	0.93	0.93	37.23	0.93	0.94	38.65
AXT2_209_6001469	0.89	0.90	34.17	0.89	0.89	35.36

7.1.4.2 Qualitative results

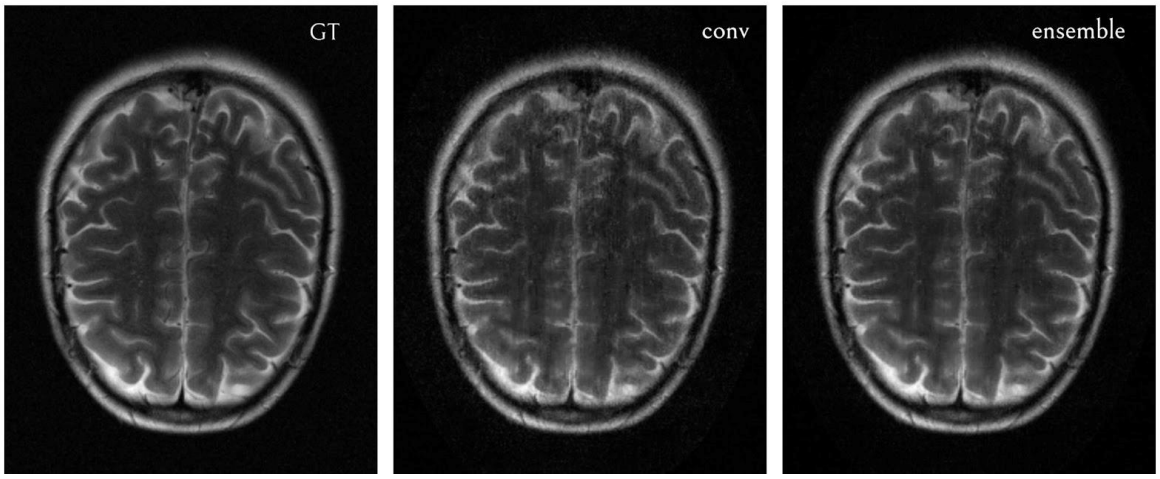


Figure 7.7 Brain Reconstruction of T2 Images (1)

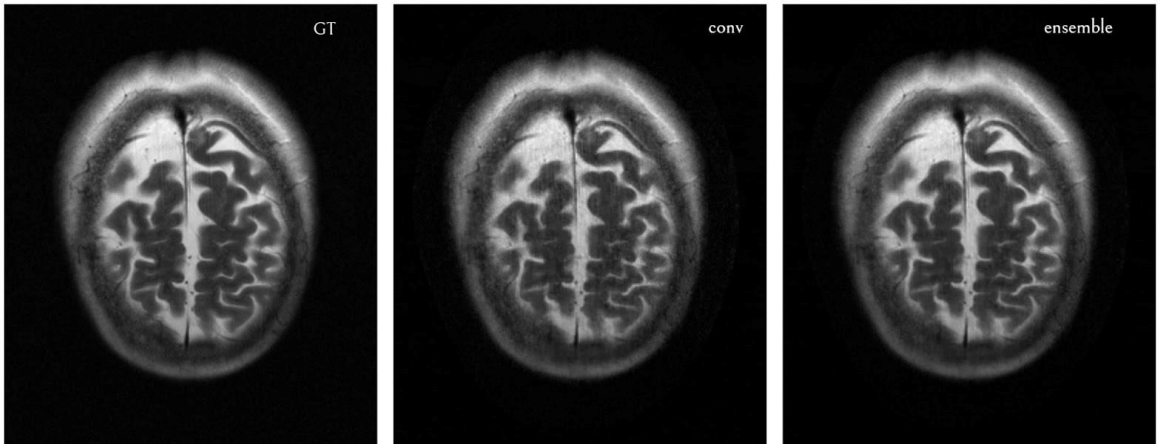


Figure 7.8 Brain Reconstruction of T2 Images (2)

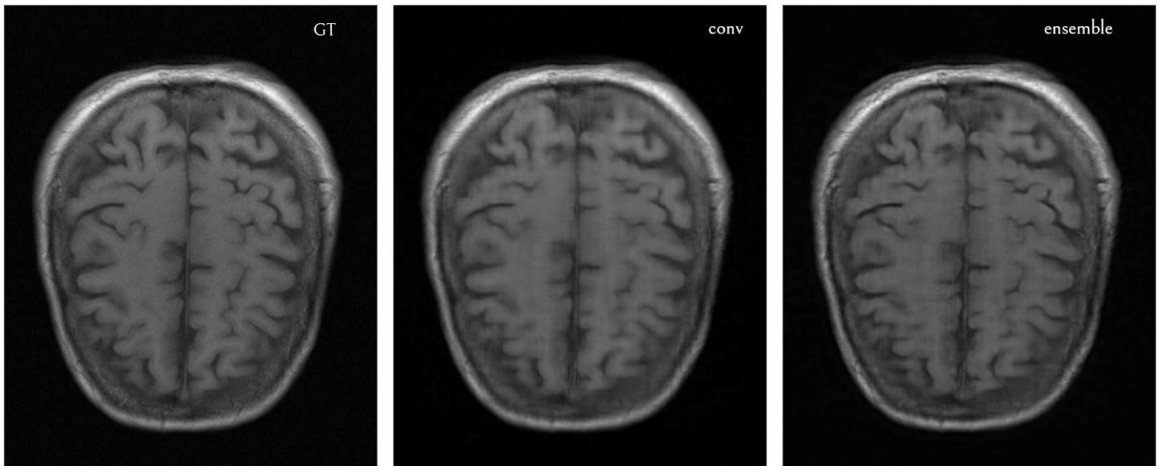


Figure 7.9 Brain Reconstruction of T1 Images (1)

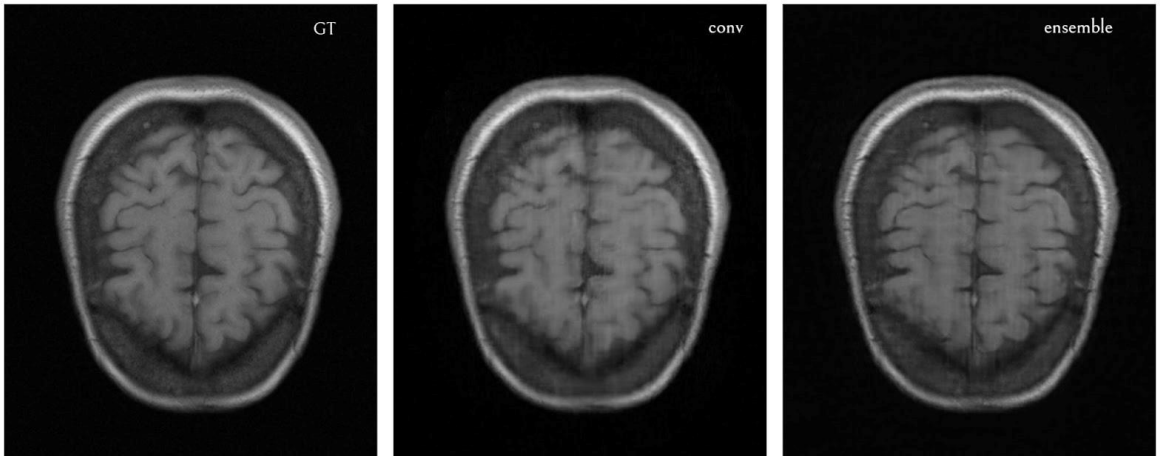


Figure 7.10 Brain Reconstruction of T1 Images (2)

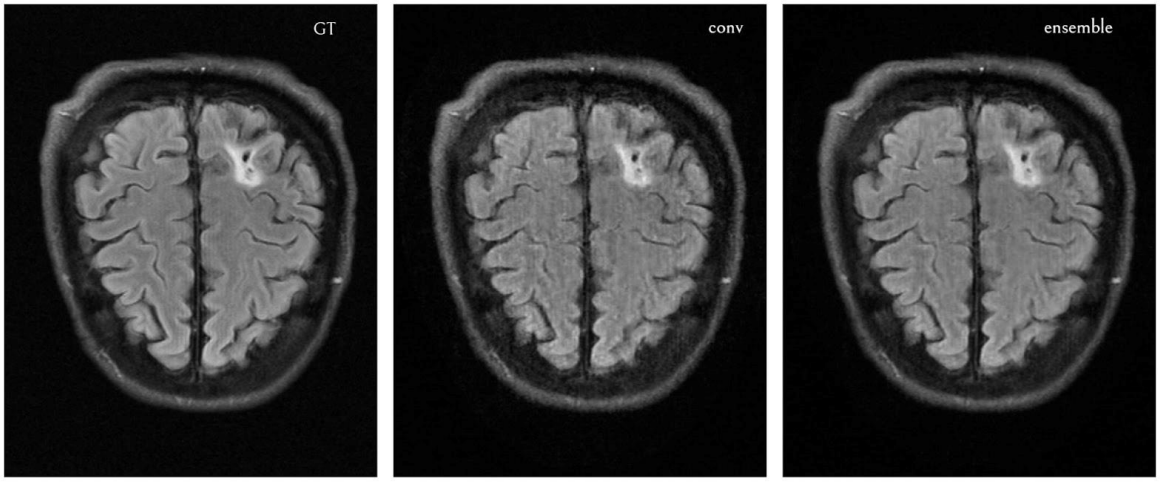


Figure 7.11 Brain Reconstruction of FLAIR Images (1)

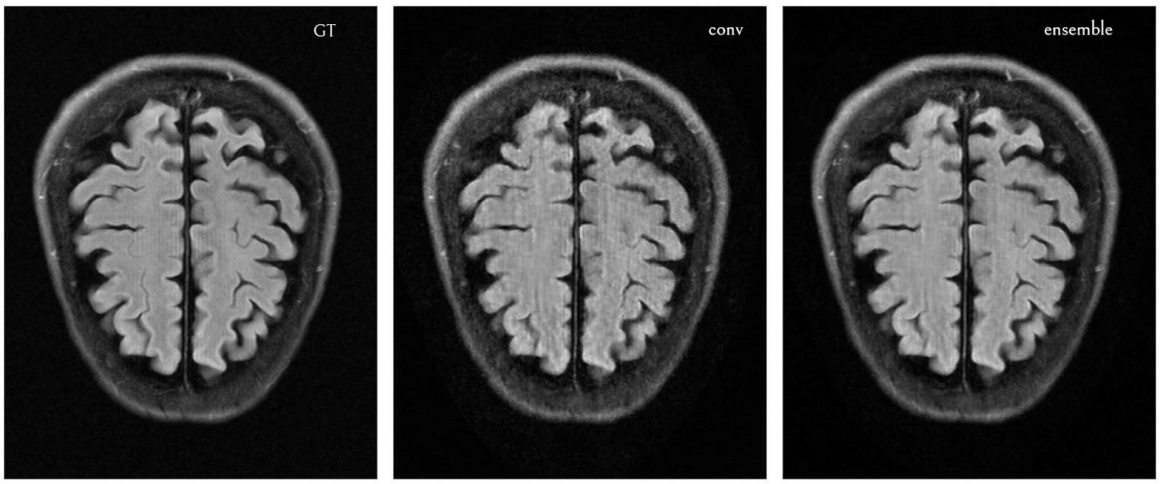


Figure 7.12 Brain Reconstruction of FLAIR Images (2)

7.2 KNEE TRANSFER LEARNING EXPERIMENTS

In this study, we conducted experiments on two distinct types of knee MRI scans: proton density (PD) and proton density with fat suppression (PDFS), comprising 796 and 798 scans, respectively. These two pulse sequences represent commonly used clinical protocols and provide coronal views of the knee joint. The primary distinction between PD and PDFS imaging lies in the treatment of fat signal. In conventional MR imaging, fat exhibits high signal intensity, which can obscure adjacent anatomical structures and reduce diagnostic clarity. To mitigate this issue, fat-suppressed (PDFS) sequences are employed to attenuate fat signals, thereby enhancing the visibility of cartilage, ligaments, and other non-fatty tissues. However, this suppression process introduces additional noise, which makes the reconstruction problem significantly more challenging. As a result, reconstructing high-quality images from PDFS scans requires more sophisticated techniques to effectively denoise and preserve anatomical details. To evaluate the effectiveness of our reconstruction approach, we also conducted a **comparative study** against several established baselines. Specifically, we compared our method based on an untrained neural network with transfer learning and bandwidth-constrained input against:

- **Deep Decoder** (an untrained, parameter-efficient network),
- **Deep Image Prior (DIP)** (a widely used untrained reconstruction technique),
- **U-Net** (a supervised, trained convolutional neural network), and
- **VarNet** (one of the state-of-the-art fully trained models for MRI reconstruction).

This comparative analysis provided valuable insights into the trade-offs between trained and untrained models, especially in the context of noisy data like PDFS scans. Our findings highlight the strengths of each approach under different conditions and underscore the potential of untrained methods when carefully regularized and initialized.

7.2.1 Non-Fat Suppressed MRI Reconstruction

Transfer Learning

We integrated transfer learning into an untrained reconstruction framework by initially training the network on a single fat-suppressed scan and subsequently reusing the learned weights to reconstruct other scans within the same category. This approach resulted in a reduction of reconstruction time by up to a factor of 10.

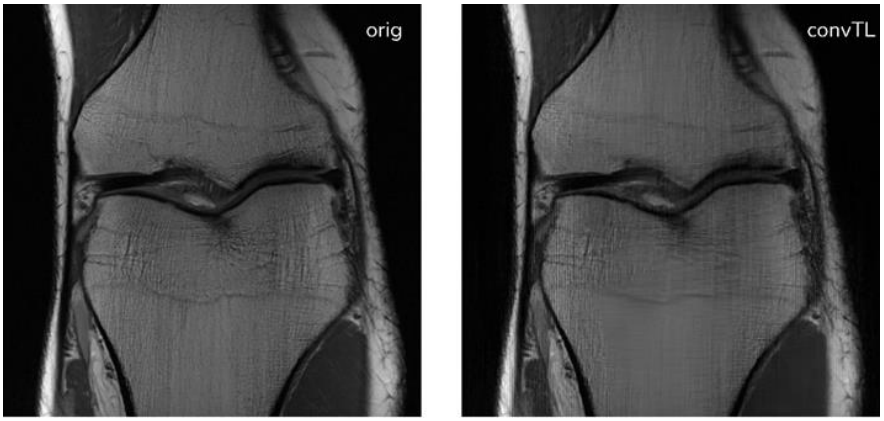


Figure 7.13 Conv decoder with Transfer Learning on -NON-fat-supp knee MRI

7.2.1.1 Qualitative Results

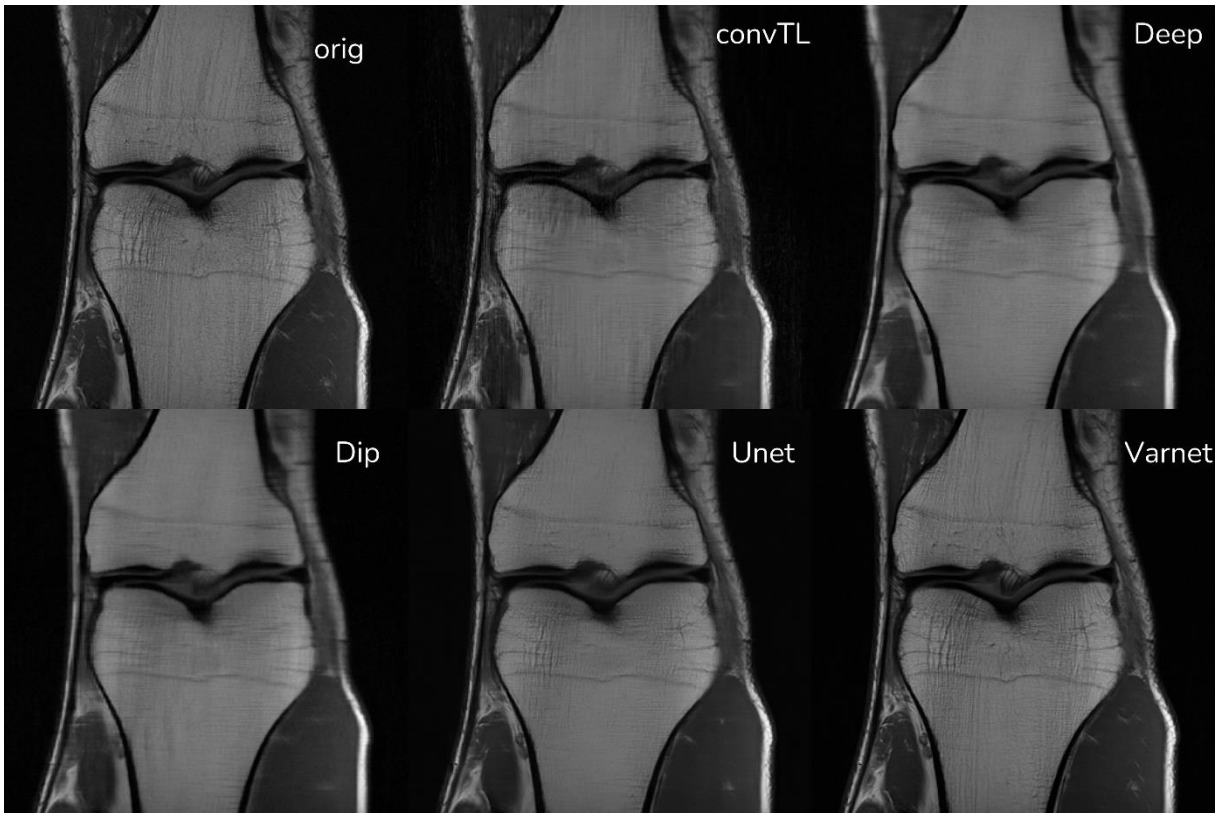


Figure 7.14 Non-Fat-Suppressed Knee Reconstruction Results

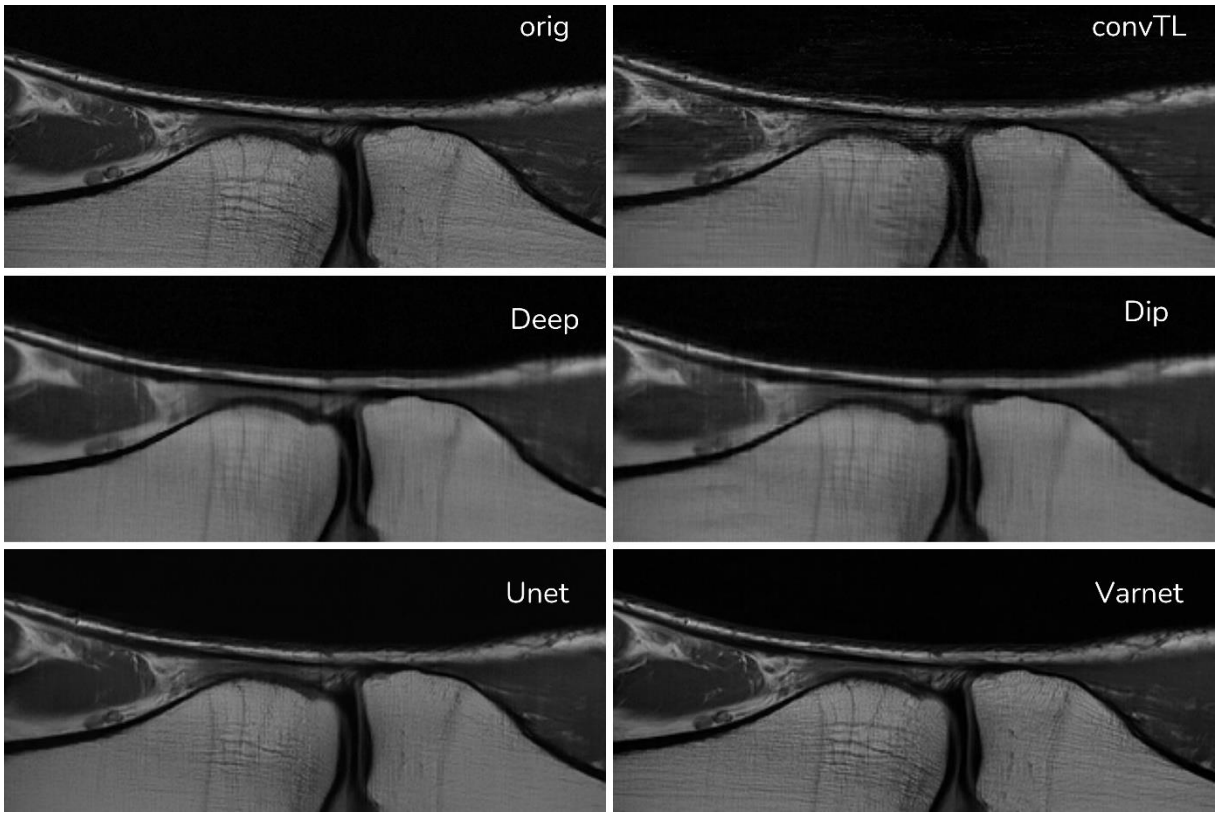


Figure 7.15 Non-Fat-Suppressed Knee Reconstruction Results

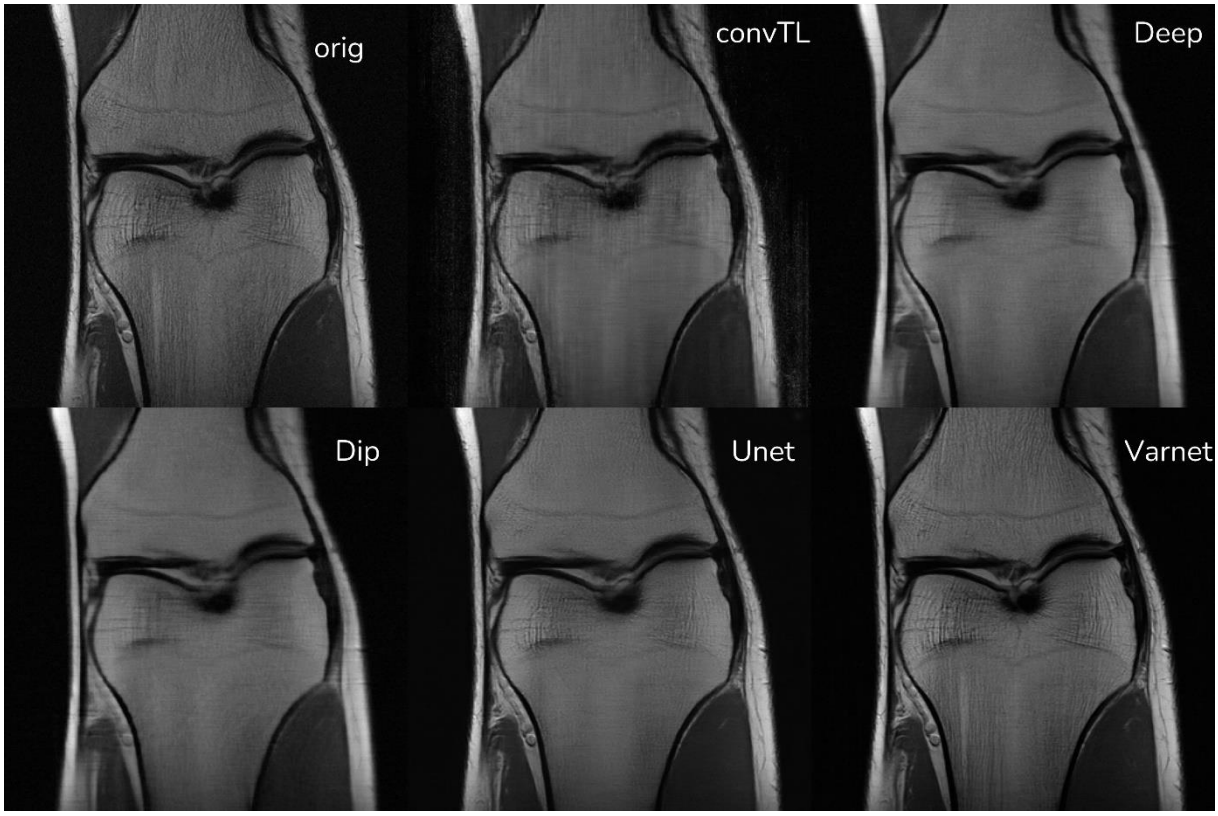


Figure 7.16 Non-Fat-Suppressed Knee Reconstruction Results

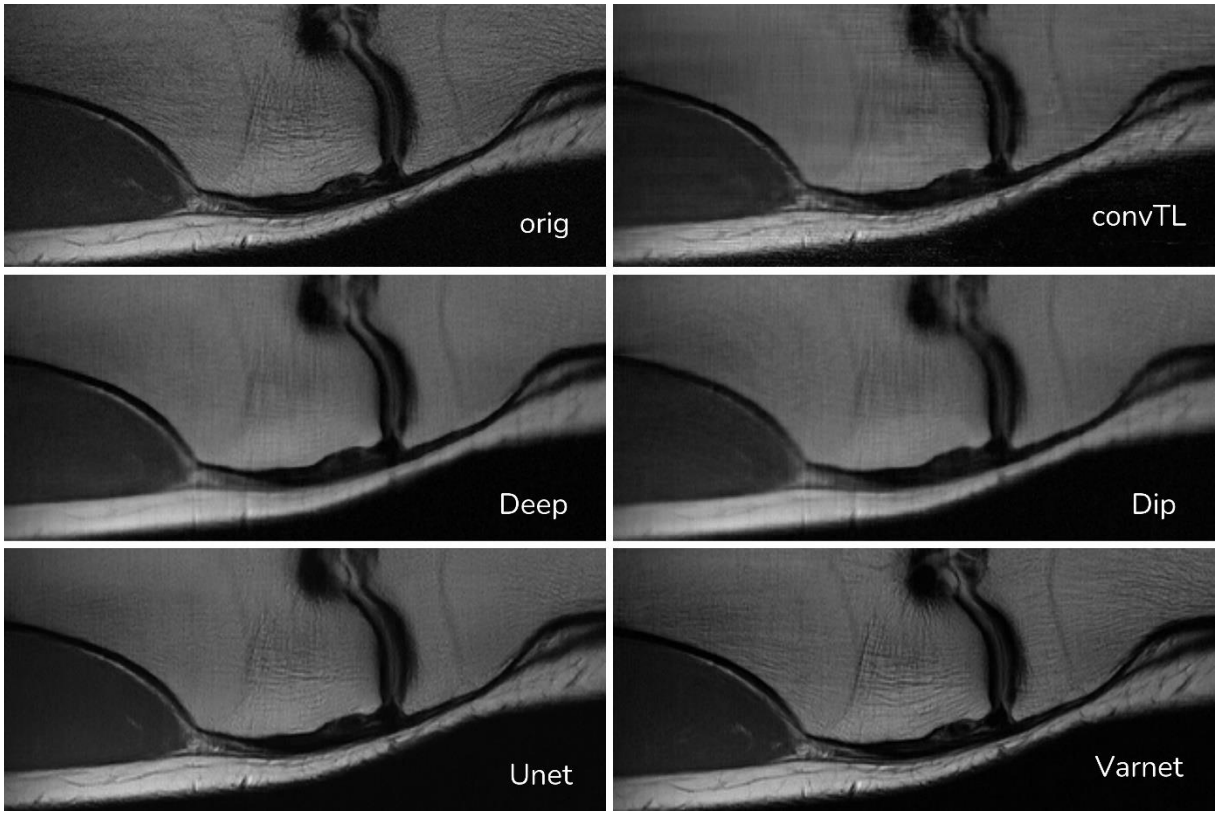


Figure 7.17 Non-Fat-Suppressed Knee Reconstruction Results

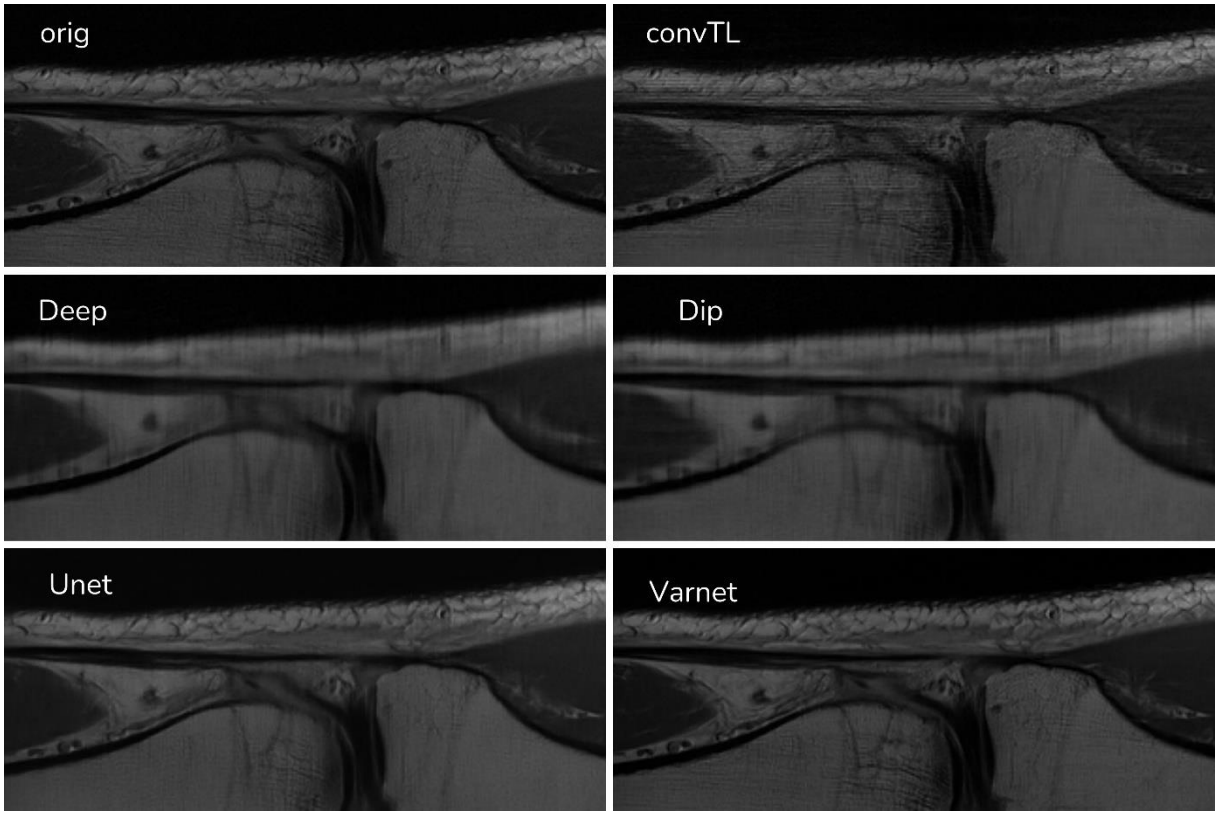


Figure 7.18 Non-Fat-Suppressed Knee Reconstruction Results

7.2.1.2 Quantitative Results

Table 7.4 Scores of the ConvDecoder, DIP, and DD models on Non-Fat-Supp Knee Images

	ConvDecoder-TL			***DIP***			***DD***		
	VIF	**SSIM**	**PSNR**	**VIF**	**SSIM**	**PSNR**	**VIF**	**SSIM**	**PSNR**
KNEE-PD-1000182	0.83	0.82	32.40	0.80	0.84	32.43	0.81	0.85	32.84
KNEE-PD-1000190	0.86	0.83	31.88	0.83	0.83	31.94	0.85	0.85	32.96
KNEE-PD-1001184	0.90	0.88	35.04	0.78	0.86	33.33	0.78	0.86	33.54
KNEE-PD-1000041	0.87	0.83	32.98	0.82	0.83	32.13	0.82	0.83	32.14
KNEE-PD-1000033	0.89	0.89	35.94	0.80	0.87	34.19	0.83	0.87	34.91
KNEE-PD-1000108	0.91	0.93	39.06	0.86	0.93	38.25	0.86	0.93	38.66
KNEE-PD-1000073	0.80	0.83	33.03	0.77	0.81	33.00	0.79	0.80	31.04

Table 7.5 Scores of the ConvDecoder vs. Trained Methods on Non-Fat-Supp Knee Images

	ConvDecoder-TL			*U-Net*			*Varnet*		
	VIF	**SSIM**	**PSNR**	**VIF**	**SSIM**	**PSNR**	**VIF**	**SSIM**	**PSNR**
KNEE-PD-1000182	0.83	0.82	32.40	0.84	0.87	33.46	0.90	0.93	36.95
KNEE-PD-1000190	0.86	0.83	31.88	0.87	0.86	33.3	0.96	0.93	37.57
KNEE-PD-1001184	0.90	0.88	35.04	0.87	0.90	35.53	0.95	0.95	39.51
KNEE-PD-1000041	0.87	0.83	32.98	0.90	0.86	33.39	0.96	0.93	36.88
KNEE-PD-1000033	0.89	0.89	35.94	0.90	0.91	36.11	0.96	0.94	39.78
KNEE-PD-1000108	0.91	0.93	39.06	0.90	0.94	39.00	0.96	0.97	43.8
KNEE-PD-1000073	0.80	0.83	33.03	0.84	0.87	34.68	0.92	0.90	37.36

7.2.2 Fat-Suppressed MRI Reconstruction

Fat-suppressed (PDFS) knee MRI scans are designed to attenuate the high signal intensity of adipose tissue, which can otherwise obscure critical anatomical structures. By suppressing the fat signal, these scans enhance the contrast of soft tissues such as cartilage, ligaments, and fluid-filled regions. However, the suppression process typically results in increased image noise, making reconstruction more complex compared to non-suppressed scans. This presents additional challenges for image restoration and necessitates more robust denoising and reconstruction strategies.

Due to the inherently high noise levels in fat-suppressed (PDFS) knee MRI scans, directly applying untrained neural networks for reconstruction poses significant challenges. In our approach, we incorporated transfer learning within an untrained framework by training the network on a single fat-suppressed scan and reusing its learned weights to accelerate the reconstruction of other scans from the same category. This method yielded up to a 10× reduction in reconstruction time. However, the noisy nature of PDFS scans made it difficult to obtain optimal weights and increased the risk of overfitting due to suboptimal initialization.To address this, we applied **non-local means denoising** exclusively to the reference scan used for weight extraction. Care was taken to ensure that no structural details were lost during denoising. The denoised image was then used to train the network, and the resulting weights were transferred for reconstructing other scans.

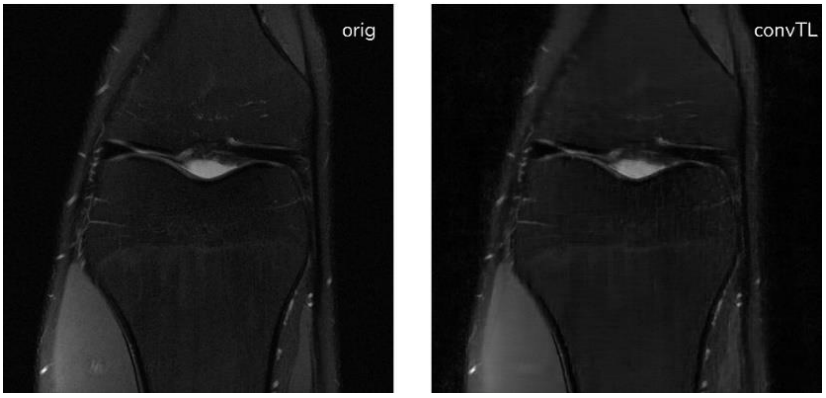


Figure 7.19 Conv decoder with Transfer Learning on fat-supp knee MRI

Quantitative Results

Table 7.6 Scores of the ConvDecoder, DIP, and DD models on Fat-Supp Knee Images

	ConvDecoder-TL			***DIP***			***DD***		
	VIF	**SSIM**	**PSNR**	**VIF**	**SSIM**	**PSNR**	**VIF**	**SSIM**	**PSNR**
KNEE-PDFS-1000090	0.80	0.82	32.58	0.78	0.80	32.00	0.76	0.80	32.08
KNEE-PDFS-1000000	0.72	0.82	33.83	0.70	0.82	32.53	0.68	0.81	32.57
KNEE-PDFS-1002436	0.70	0.85	36.88	0.67	0.83	35.57	0.66	0.82	34.63
KNEE-PDFS-1002067	0.74	0.84	34.39	0.70	0.76	30.57	0.66	0.75	30.11

Table 7.7 Scores of the ConvDecoder vs. Trained Methods on Fat-Supp Knee Images

	ConvDecoder-TL			***U-Net***			***Varnet***		
	VIF	**SSIM**	**PSNR**	**VIF**	**SSIM**	**PSNR**	**VIF**	**SSIM**	**PSNR**
KNEE-PDFS-1000090	0.80	0.82	32.58	0.78	0.81	32.54	0.86	0.84	34.29
KNEE-PDFS-1000000	0.72	0.82	33.83	0.73	0.82	33.70	0.81	0.84	35.92
KNEE-PDFS-1002436	0.70	0.85	35.88	0.72	0.89	37.55	0.79	0.90	38.69
KNEE-PDFS-1002067	0.74	0.84	34.39	0.71	0.77	31.95	0.77	0.87	36.54

Qualitative Results

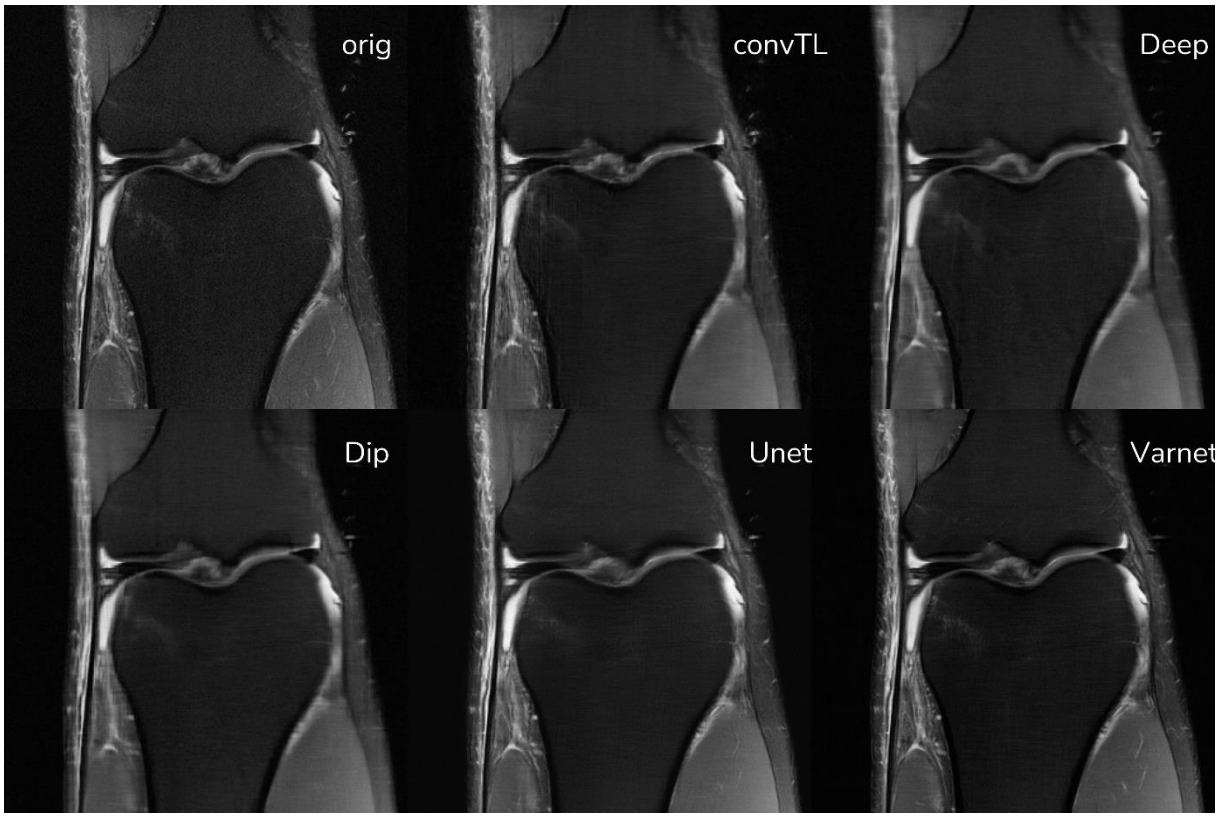


Figure 7.20 comparison between our method & 2 untrained networks & 2 trained network

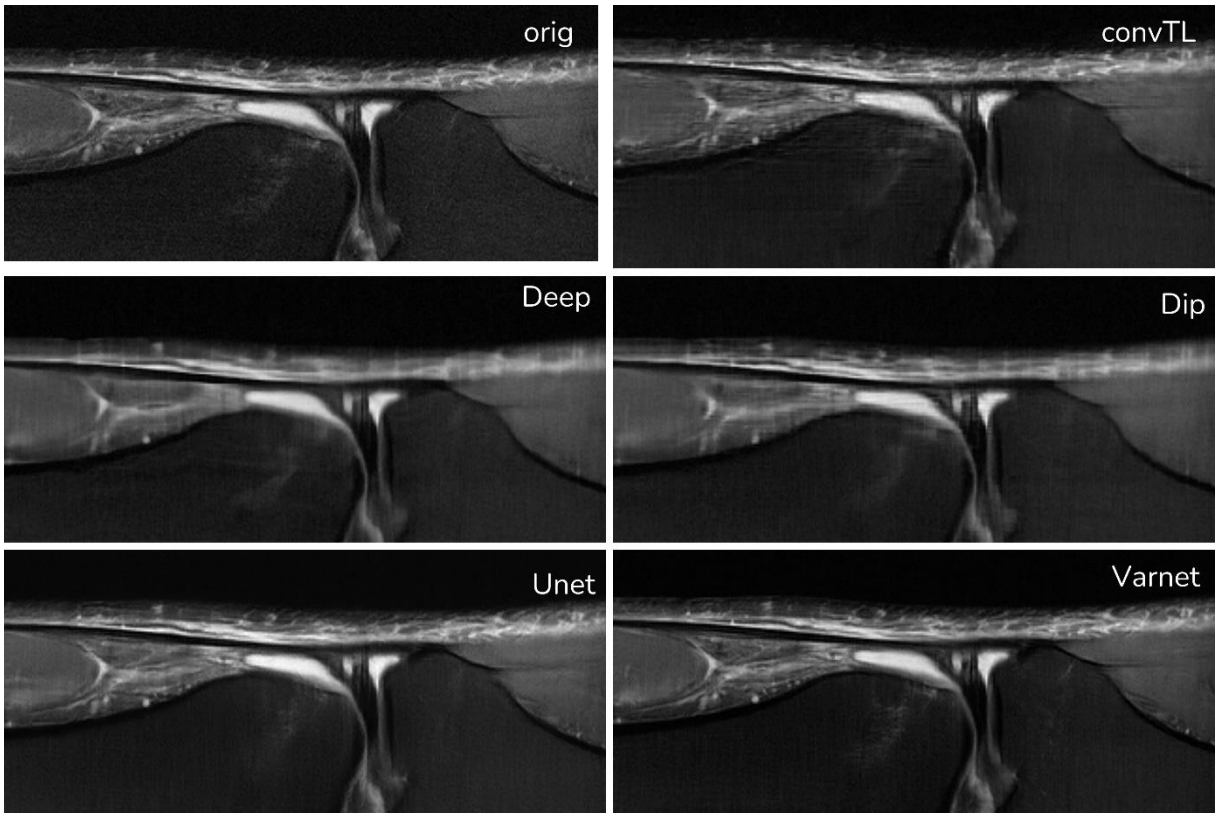


Figure 7.21 zoomed-in comparison between our method & 2 untrained net & 2 trained net

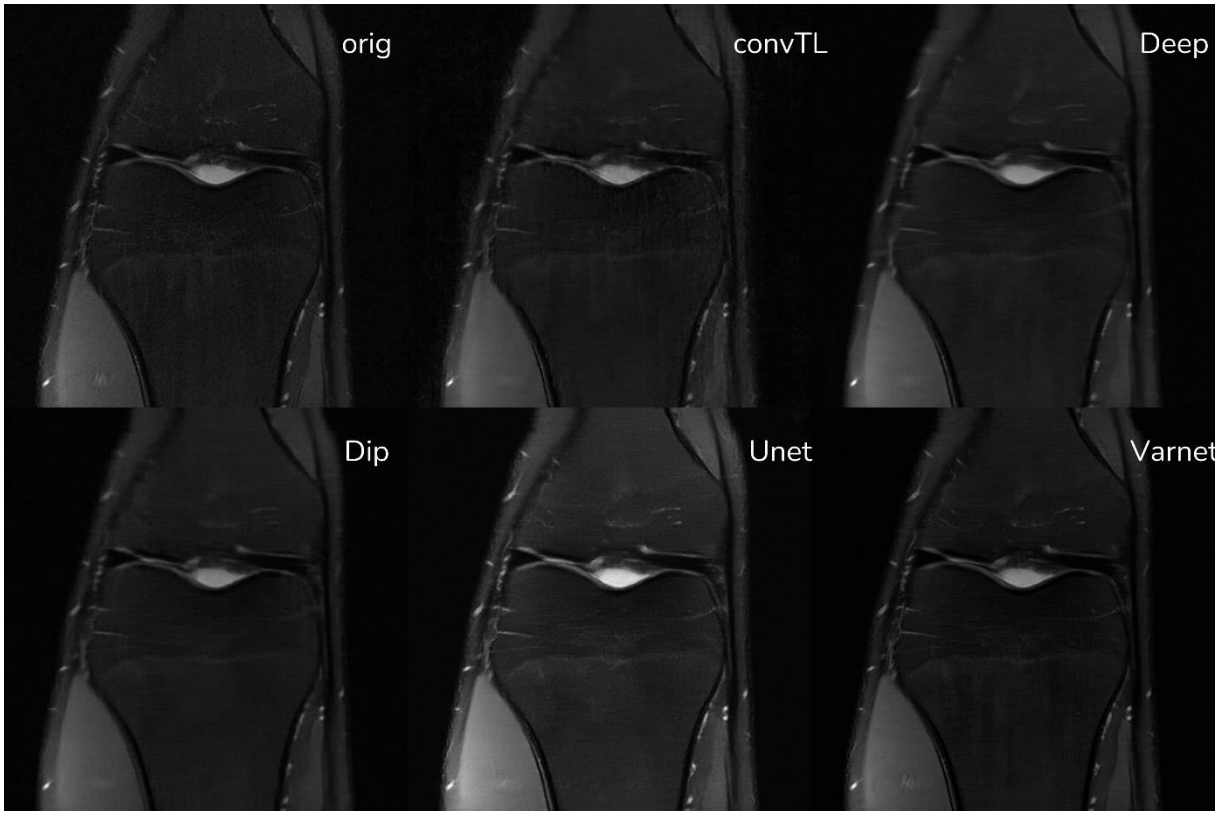


Figure 7.22 comparison between our method & 2 untrained networks & 2 trained

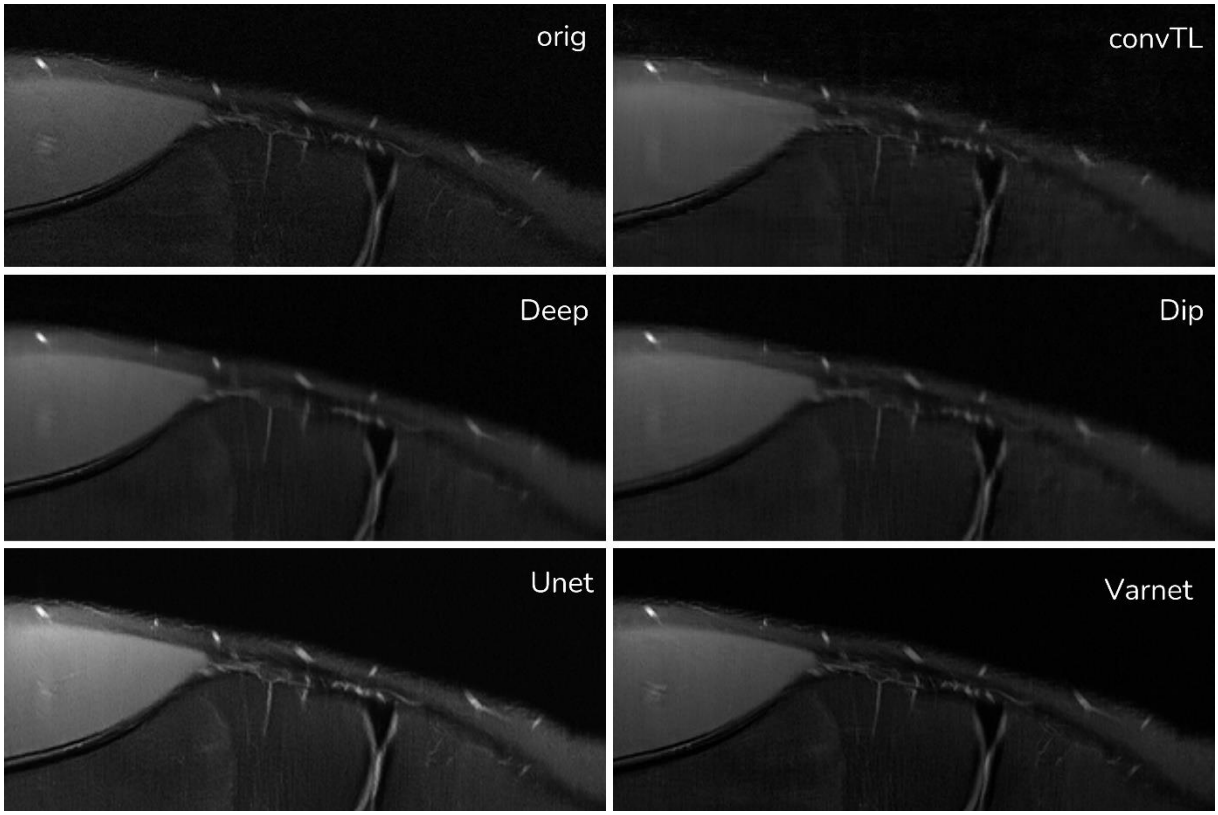


Figure 7.23 zoomed-in comparison between our method & 2 untrained net & 2 trained net

7.3 POST-PROCESSING ENHANCEMENT

To further improve the visual quality of the reconstructed MRI images produced by the ConvDecoder model, a **pre-trained super-resolution (SR) model trained on the FastMRI Brain dataset** was employed as a post-processing step. This enhancement targeted only the T1-weighted Brain MRI and Knee MRI datasets, which exhibited resolution degradation artifacts as a result of accelerated, undersampled MRI reconstruction. The goal was to restore fine anatomical details and improve overall image sharpness without retraining the SR model.

The super-resolution model used in this study is derived from a **multi-task convolutional framework** presented in the paper *"Improving MR image quality with a multi-task model, using convolutional losses"*. Specifically, the **super-resolution subnetwork** of this framework was utilized.

7.3.1 Model Architecture

It is built on a modified version of **SRResNet**, a deep convolutional neural network originally proposed for single-image super-resolution, and was chosen for its ability to generate perceptually realistic reconstructions.

A key innovation in this model is its **dual-path architecture**, which combines information from both the **image domain** and the **frequency domain (k-space)**. The architecture consists of two parallel SRResNet-based subnetworks:

- The **first SRResNet model** operates directly on the image and predicts residuals in the spatial domain.
- The **second SRResNet model** first applies a Fourier Transform to operate in k-space (frequency domain), then outputs residuals which are transformed back to the image space via an Inverse Fourier Transform.

The outputs of these two branches—now both in the image space—are **summed together** to form the final predicted residuals. These residuals are added to the input image to generate the final enhanced output. This parallel processing enables the model to correct high-frequency artifacts and recover image structure using complementary information from both domains.

Each SRResNet subnetwork follows an encoder-decoder design with **twelve convolutional blocks**. The encoder part includes **five downsampling blocks** using strided convolutions, while the decoder contains **five upsampling blocks**. Skip connections link the encoder and decoder at corresponding depths, resembling the structure of a **U-Net**, which helps preserve spatial detail. Each block uses 2D convolutions, Batch Normalization, and LeakyReLU activations, with channel depth increasing at lower levels of the network.

Another essential component of the training strategy was the use of **perceptual (convolutional) loss**. Rather than relying solely on pixel-wise losses like MSE, the model uses a VGG-based feature space loss that better aligns with human visual perception. This encourages the restoration of textures, edges, and fine details that might otherwise be overlooked in conventional loss functions.

7.3.2 Experiments and Results

7.3.2.1 Qualitative Results of T1w Brain MRI

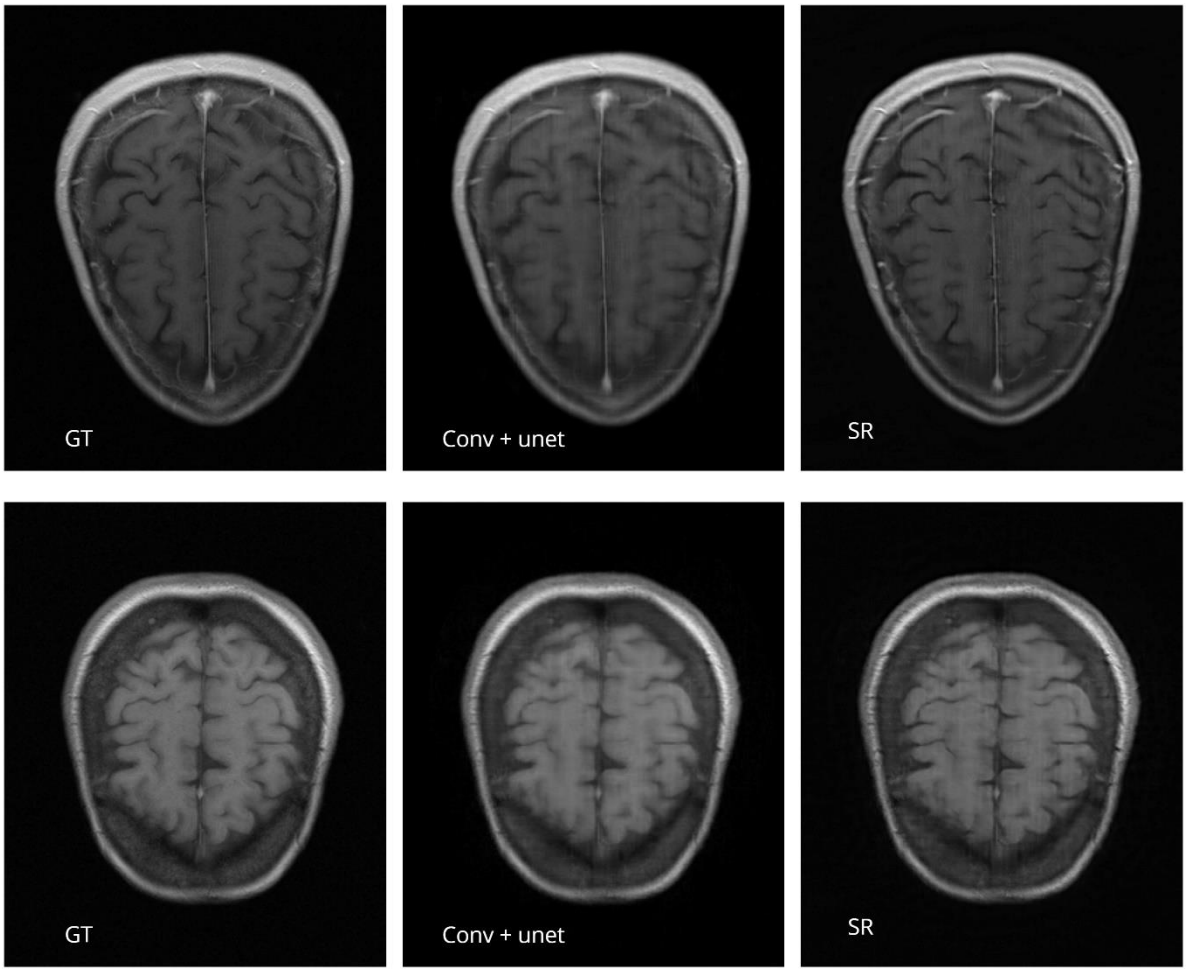


Figure 7.24 Comparison between the GT, the Reconstructed T1w before SR, and MRI after (SR)

7.3.2.2 Quantitative Results of T1w Brain MRI

Table 7.8 T1w Brain Reconstruction Enhancement Scores

	ConvDecoder Before SR			After SR		
	VIF	**SSIM**	**PSNR**	**VIF**	**SSIM**	**PSNR**
AXT1POST_209_6001084	0.90	0.93	37.67	0.91	0.91	35.82
AXT1PRE_203_6000645	.89	.90	36.06	.91	.90	35.36
AXT1PRE_203_6000631	.82	.86	33.23	.87	.85	33.08
AXT1PRE_209_6001221	.86	.90	35.43	.88	.89	35.83

7.3.2.3 Quantitative Results of Knee MRI

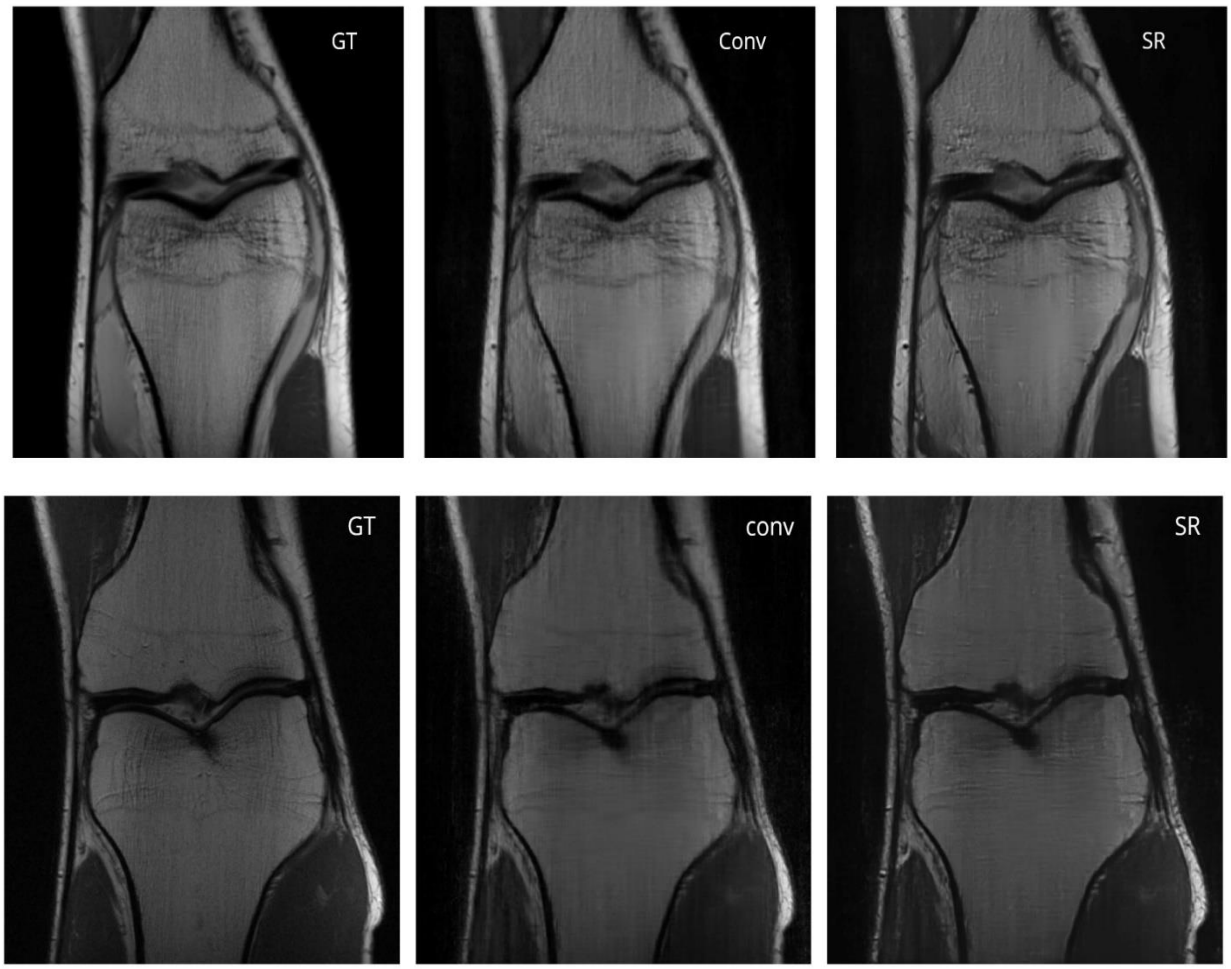


Figure 7.25 Comparison between the GT and the Reconstructed of Non-fat Knee MRI before and after SR

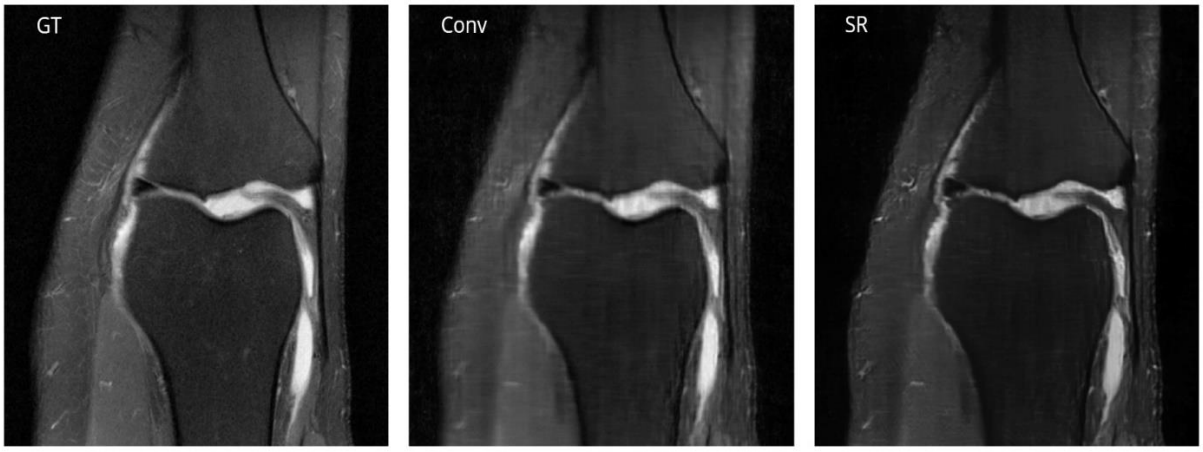


Figure 7.26 Comparison between the GT and the Reconstructed of FS Knee MRI before and after SR

7.3.2.4 Qualitative Results of Knee MRI

Table 7.9 Knee MRI Reconstruction Enhancement Scores

	ConvDecoder Before SR			After SR		
	VIF	**SSIM**	**PSNR**	**VIF**	**SSIM**	**PSNR**
KNEE-PD-1000107	0.84	0.83	32.59	0.87	0.83	33.20
KNEE-PD-1000108	0.91	0.93	39.06	0.93	0.92	39.0
KNEE-PD-1000126	0.80	0.80	32.70	0.83	0.81	33.08
KNEE-PD-1000153	0.82	0.75	30.33	0.85	0.73	30.05
KNEE-NF 1	0.71	0.86	33.98	0.75	0.85	33.60
KNEE-NF 2	0.87	0.81	31.38	0.89	0.82	31.75
KNEE-NF 14	0.88	0.84	32.40	0.90	0.83	30.20
KNEE-NF 20	0.89	0.88	35.22	0.91	0.89	36.09
KNEE-FS 8	0.75	0.71	30.14	0.78	0.70	29.87
KNEE-FS 9	0.69	0.79	32.66	0.72	0.80	32.90

7.4 GENERALIZATION EVALUATION ON EXTERNAL MRI DATASETS

Deep learning-based MRI reconstruction models often face challenges in generalizing across datasets due to variations in acquisition protocols, coil configurations, and anatomical contrasts. While the ConvDecoder model was trained and validated on the FastMRI dataset, its real-world applicability depends on performance across diverse, unseen data. This section evaluates the model's robustness on two external datasets:

- **M4Raw Brain MRI:** A brain dataset with distinct acquisition parameters.
- **CAI2R 100 Knee MRI Cases:** A knee dataset with varied pathologies and imaging protocols.

7.4.1 Description of External Datasets

7.4.1.1 M4Raw Brain Dataset

A publicly available dataset designed for low-field MRI research, M4Raw provides fully sampled, multi-contrast, multi-channel raw k-space data (T1w, T2w, FLAIR) acquired at 0.3 Tesla using a four-channel head coil. Key challenges include:

- **Lower resolution and higher noise** compared to high-field systems (e.g., FastMRI's 1.5T/3T data).
- **Limited coil count**, exacerbating undersampling artifacts (4 coils).

7.4.1.2 100 Knee MRI Cases Dataset

This dataset, provided by NYU's CAI2R, includes 100 knee MRI scans across five sequences:

- Coronal spin density-weighted (with/without fat suppression).
- Axial and sagittal T2-weighted (fat-suppressed).
- Sagittal spin density-weighted.

Each case includes raw k-space (MATLAB format), coil sensitivity maps (15 coils), and reference reconstructions, enabling rigorous evaluation of reconstruction algorithms.

7.4.2 Experiments and Results

7.4.2.1 Experiments on Brain Dataset and Results

We evaluated our ConvDecoder+U-Net ensemble model on M4Raw's test subset including Axial T1w, T2w, and Flair MR images, applying super-resolution post-processing on T1w to enhance low-field reconstructions.

Our untrained ConvDecoder model not only generalized effectively but **exceeded** the performance of a trained U-Net, demonstrating its potential for deployment in low-field MRI scenarios where training data is scarce.

a) Qualitative Results

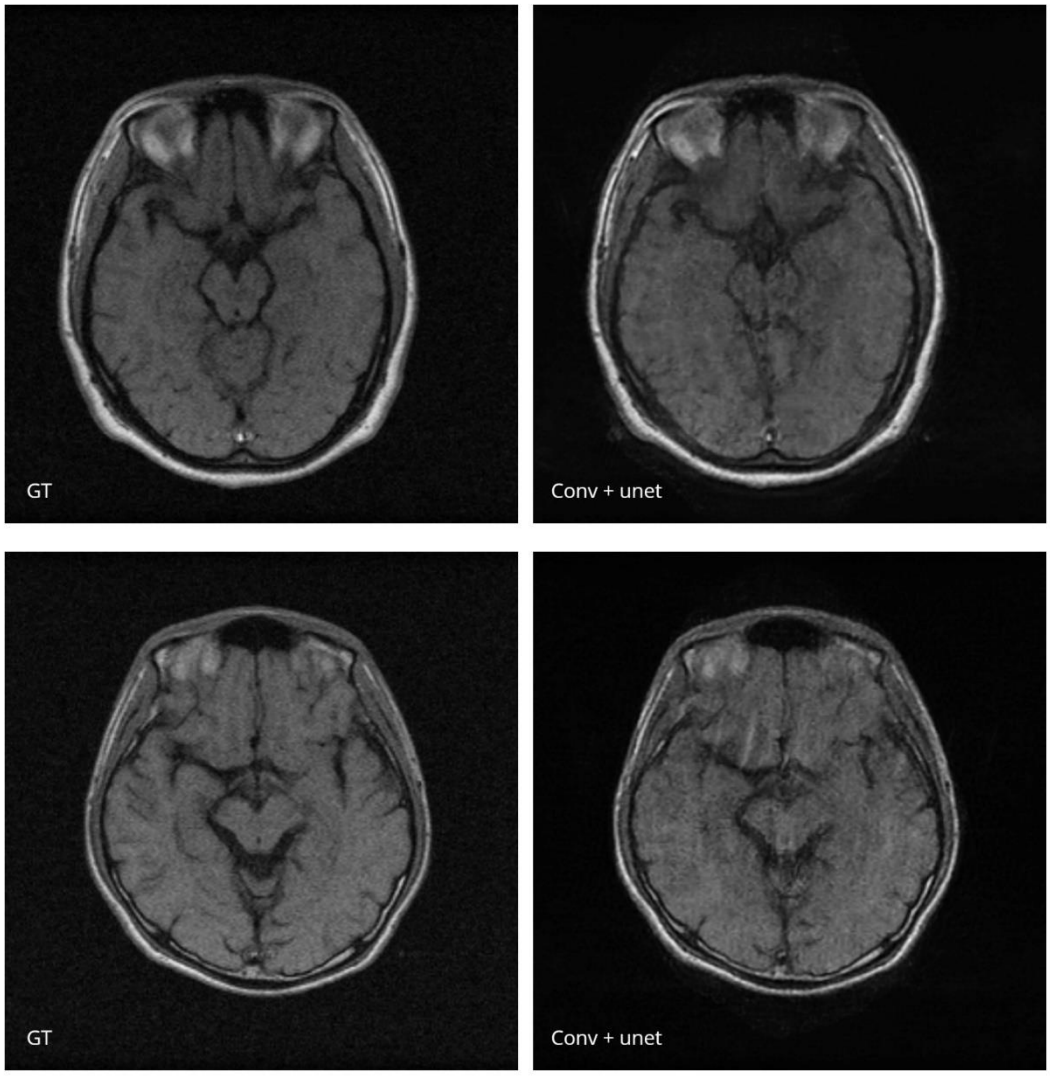
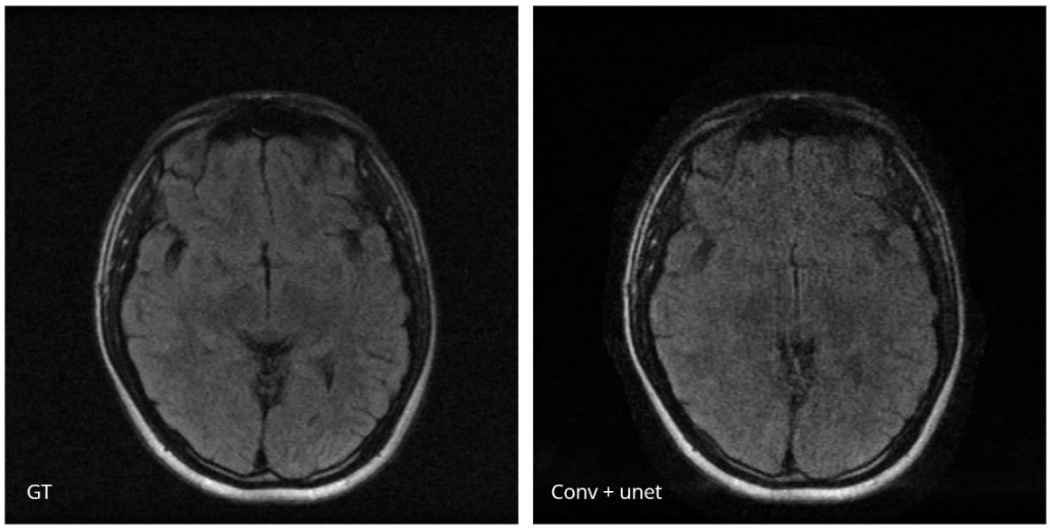


Figure 7.27 Comparison between the Ground Truth and the Reconstructed of T1w Brain MR images from "M4Raw" Dataset



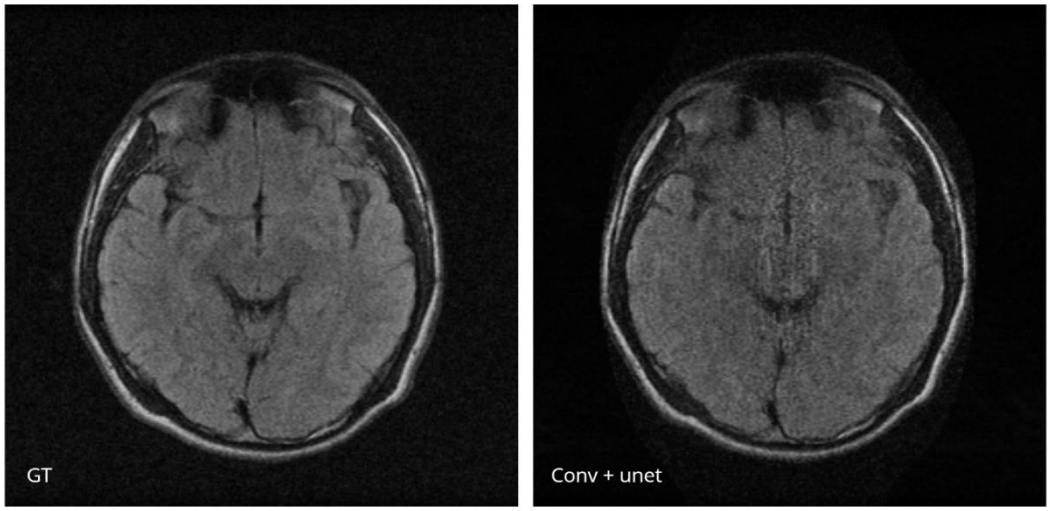


Figure 7.28 Comparison between the GT and the Reconstructed of Flair Brain MR images from "M4Raw" Dataset

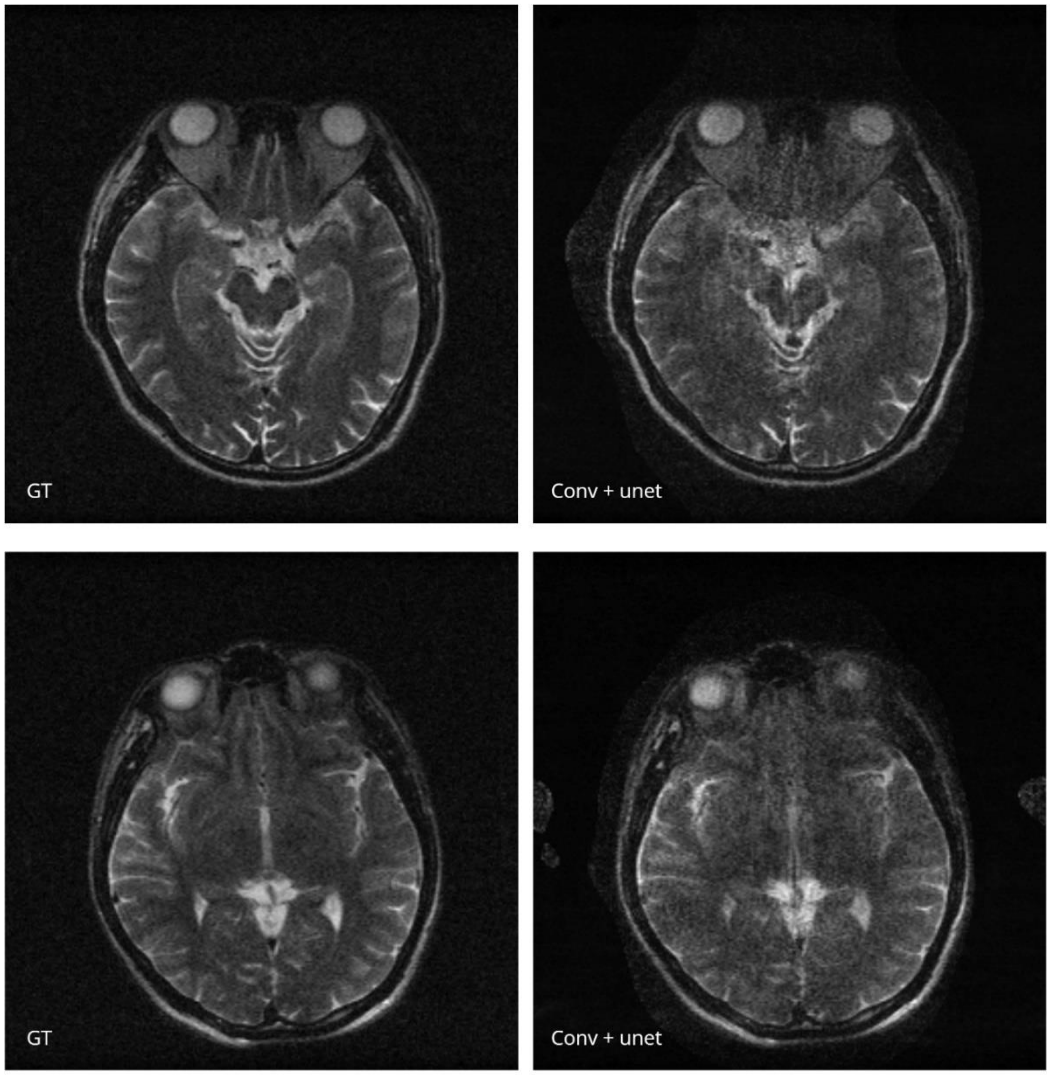


Figure 7.29 Comparison between the Ground Truth and the Reconstructed of T2 Brain MR images from "M4Raw" Dataset

b) Quantitative Results

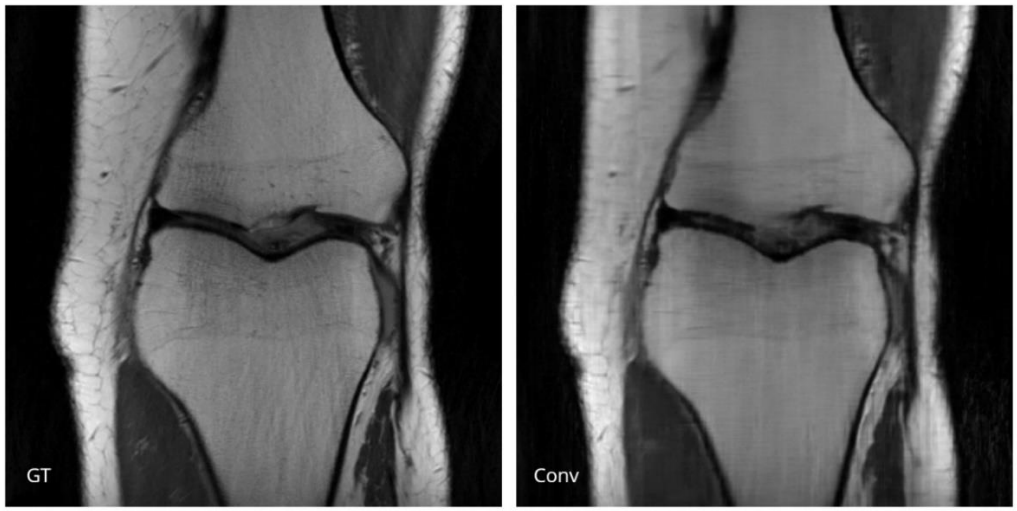
Table 7.10 M4Raw Brain Reconstruction Scores

	ConvDecoder		
	VIF	**SSIM**	**PSNR**
2022101201_T101	0.78	0.71	27.82
2022121703_T106	0.78	0.78	29.2
2022101103_FLAIR01	0.79	0.59	30.02
2022121706_FLAIR02	0.81	0.72	29.20
2022121705_T202	0.72	0.76	29.15
2022101102_T203	0.70	0.74	28.38

7.4.2.2 Experiments on Knee Dataset

We evaluate our Transfer Learning ConvDecoder Model on the “100 Knee MRI Cases” Dataset on the Coronal spin density-weighted with fat suppression and without fat suppression images.

a) Qualitative Result



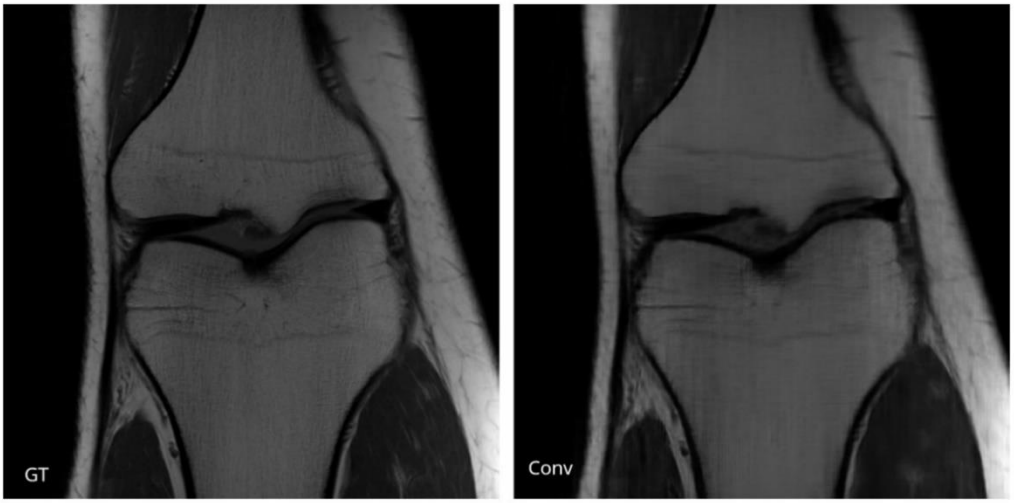


Figure 7.30 Comparison between the GT and the Reconstructed of Non-fat Knee MR images from "100 Knee MRI Cases" Dataset

Figure 7.31 Comparison between the Ground Truth and the Reconstructed of Fat Knee MR images from "100 Knee MRI Cases" Dataset

b) Quantitative Results

Table 7.11 “100 Knee MRI Cases” Dataset Reconstruction Scores

	ConvDecoder		
	VIF	**SSIM**	**PSNR**
KNEE1-NF 14	0.87	0.81	31.38
KNEE-NF 20	0.89	0.88	35.22
KNEE-NF 1	0.71	0.86	33.98
KNEE-NF 2	0.87	0.81	31.38
KNEE-FS 8	0.75	0.71	30.14
KNEE-FS 9	0.69	0.79	32.66

7.5 EVALUATING MULTI-DOMAIN LOSS FUNCTIONS FOR ROBUST IMAGE RECONSTRUCTION WITH SENSITIVITY MAPS INTEGRATION.

This method represents a pivotal component of our proposed framework, as it leverages a composite loss function to enforce consistency across both the frequency and spatial domains while incorporating powerful regularization priors. The design integrates k-space data fidelity, image-space structural accuracy, and sparsity-promoting penalties namely Total Variation (TV) and Wavelet L1 norms to produce high-quality reconstructions even under challenging conditions. A key advantage of this approach is its generalizability: it is fully adaptable across a wide range of MRI modalities, anatomical regions, and acquisition protocols .Furthermore, the integration of sensitivity maps enhances the method's robustness in handling multi-coil data, improving signal separation and channel-wise fidelity. Notably, this approach demonstrates strong resilience to noise, effectively reconstructing anatomically plausible images from inherently noisy acquisitions such as fat-suppressed knee MRIs.

This idea proves particularly powerful due to its ability to operate without any model initialization or transfer learning, achieving a remarkable 10× speed-up comparable to that of transfer learning-based methods. By mitigating both paradigms untrained reconstruction(conv decoder architecture) with multiple loss and transfer learning within a unified design, the framework consistently outperforms conventional architectures such as U-Net and approaches the performance levels of advanced supervised models like VarNet. This balance between computational efficiency and reconstruction quality signifies a major success, demonstrating that a hybrid approach can deliver near state-of-the-art .

$$L(x, k_0) = \eta_1 \|PFSx - k_0\|_1 + \eta_2 \|F^{-1}(PFSx - k_0)\|_2^2 + \rho R\big(F^{-1} DC(FSx, k_0)\big)$$

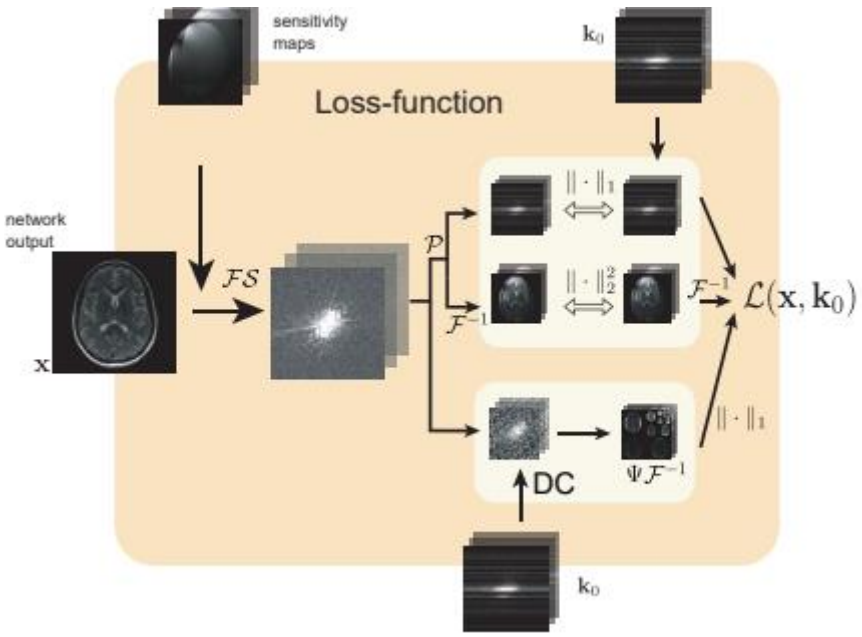


Figure 7.32 Multiple Loss diagram

7.5.1 Knee MRI Reconstruction

This section presents a comparative analysis between the initial version of our model, ConvDecoder with transfer learning, and the enhanced version incorporating Multi-Domain Loss Functions for robust knee MRI image reconstruction.

Qualitative Results

Figure 7.33 comparison 1: convmodel with only transfer learning(TL) vs with multi-loss(ML)

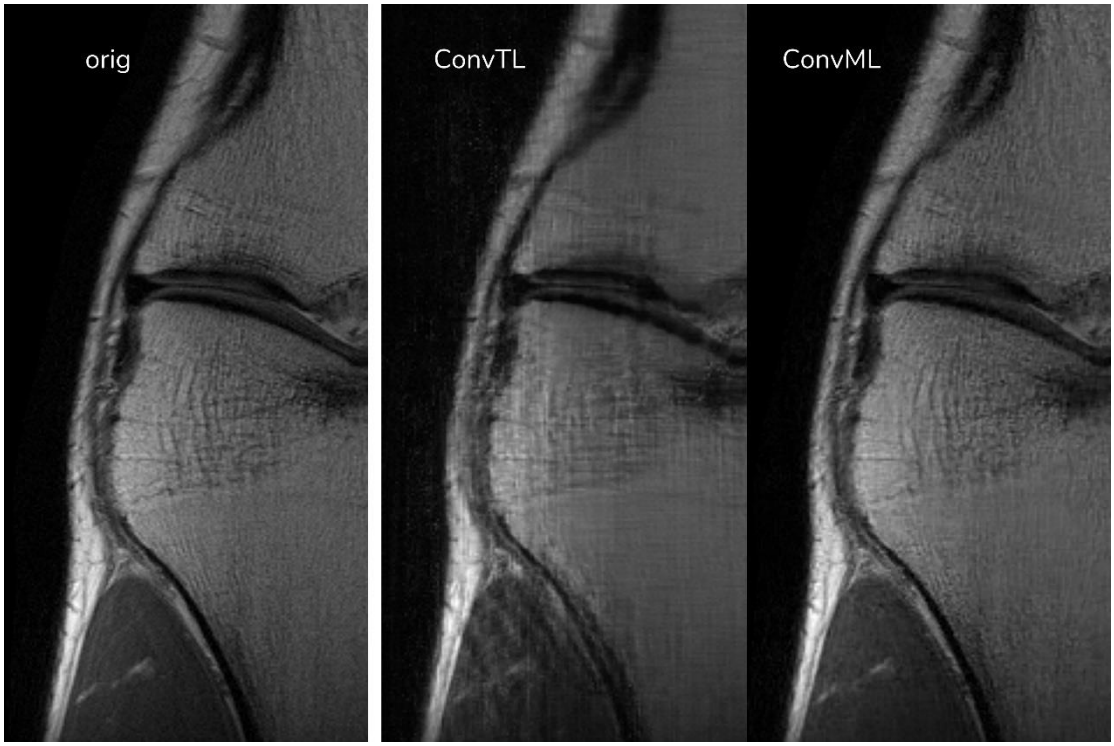


Figure 7.34 comparison 2: convmodel with only transfer learning(TL) vs with multiloss(ML)

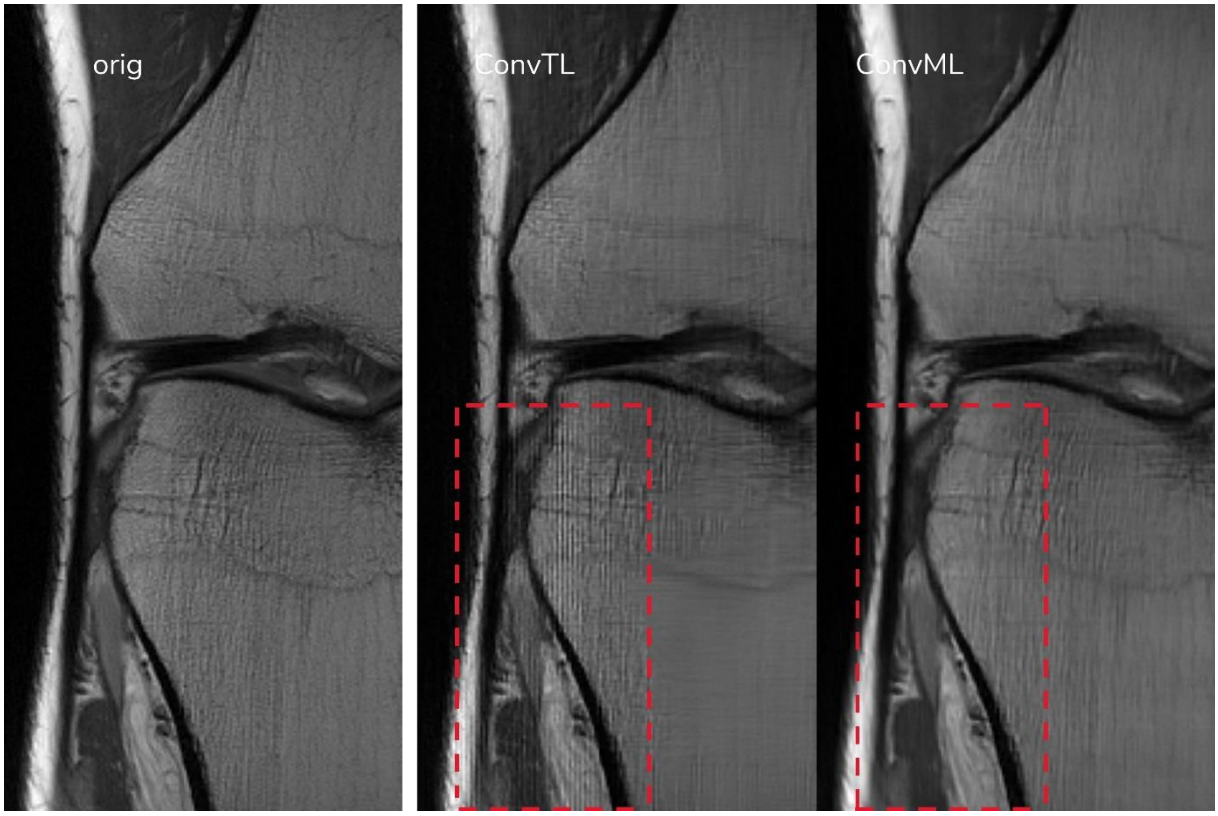


Figure 7.35 comparison 3: convmodel with only transfer learning(TL) vs with multiloss(ML)

7.5.2 Brain MRI Reconstruction

This section presents a comparative analysis between our initial ensemble approach combining ConvDecoder and U-Net and the ConvDecoder with a multi-loss strategy. For T1-weighted images, we ensured a fair comparison by also applying super-resolution to the multi-loss ConvDecoder model before evaluation. In this setup, we observed that initializing the new model with a reference T1 image led to better reconstruction quality than simply increasing the number of iterations. However, the ensemble method slightly outperformed the multi-loss ConvDecoder with super-resolution in T1 reconstructions. In contrast, for T2 and FLAIR images, the multi-loss ConvDecoder consistently delivered better performance, even without initialization.

Qualitative Results

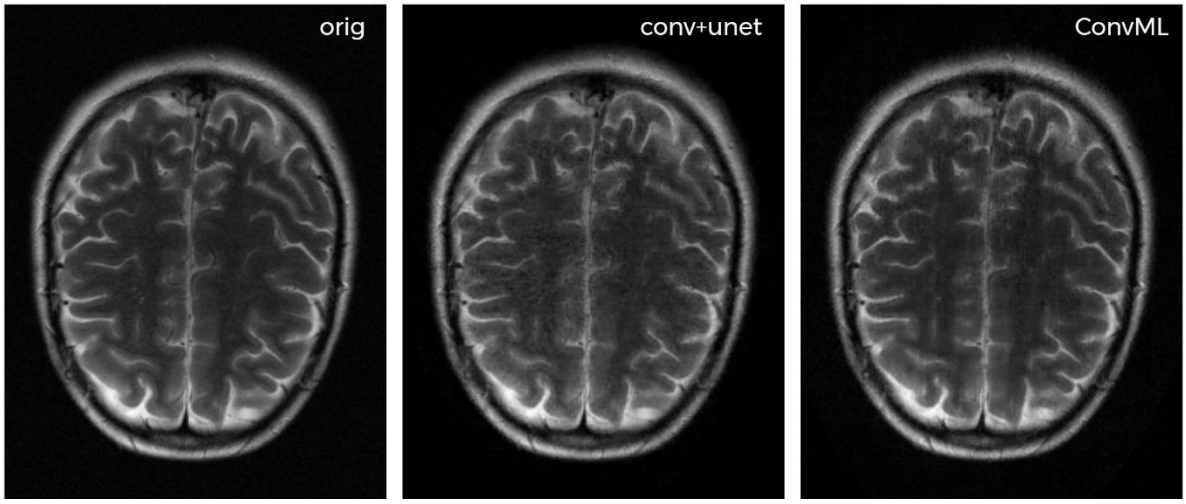


Figure 7.36 : Comparison of Reconstruction Results on T2-Weighted Image

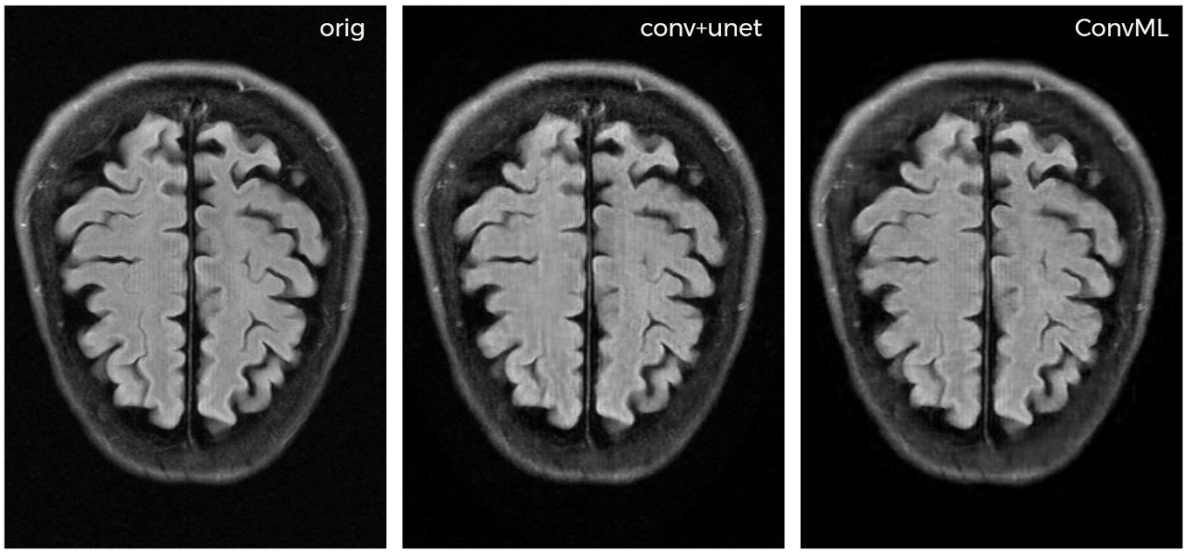


Figure 7.37: Comparison of Reconstruction Results on FLAIR Image

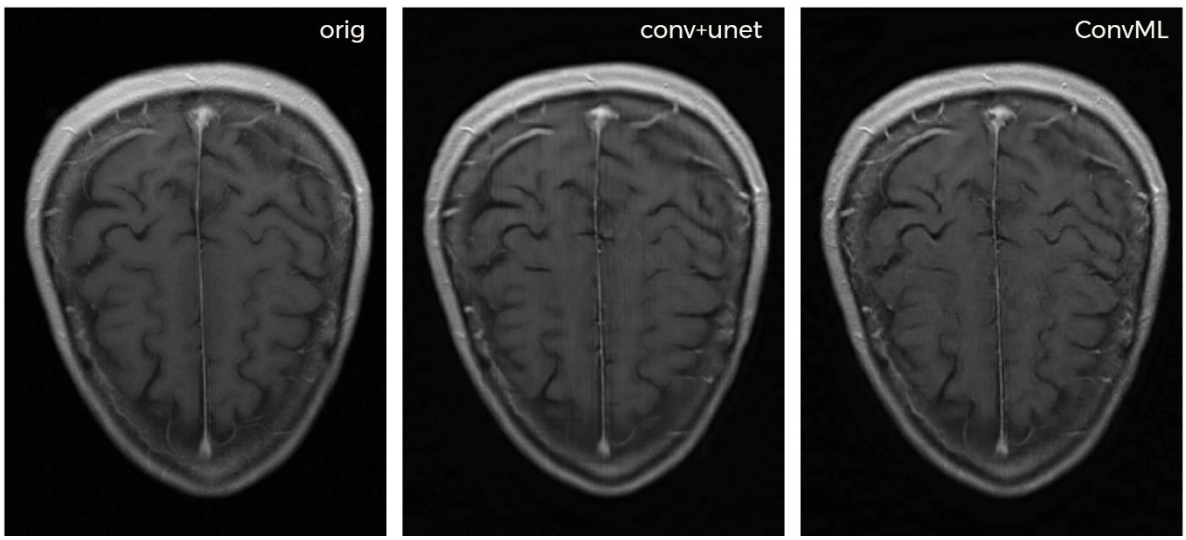


Figure 7.38: Comparison of Reconstruction Results on T2-Weighted Image

Quantitative Results

Table 7.12 Scores og Multi-Loss Model

	ConvTL			*ConvML*			*Varnet*		
	MS-SIM	**SSIM**	**PSNR**	**MS-SIM**	**SSIM**	**PSNR**	**MS-SIM**	**SSIM**	**PSNR**
KNEE-PD-1000182	0.96	0.83	31.98	0.98	0.91	35.79	0.99	0.93	36.88
KNEE-PD-1000031	0.92	0.74	29.92	0.98	0.89	34.22	0.97	0.85	34.07
KNEE-PD-1000041	0.96	0.83	31.98	0.98	0.91	35.79	0.99	0.93	36.88
KNEE-PDFS-1001090	0.96	0.82	32.58	0.96	0.80	33.07	0.97	0.84	34.29
KNEE-PDFS-1002436	0.97	0.85	35.88	0.98	0.90	37.92	0.98	0.90	38.69
KNEE-PDFS-1002067	0.97	0.84	34.39	0.98	0.87	35.82	0.98	0.87	36.54
KNEE-PD-1000153	0.94	0.75	30.33	0.98	0.89	34.24	0.98	0.87	34.63
KNEE-PD-1000126	0.96	0.80	32.7	0.98	0.88	36.50	0.98	0.89	36.78
KNEE-PD-1001184	0.98	0.88	35.04	0.99	0.93	37.34	0.99	0.95	39.51
AXFLAIR_201_6002872	0.97	0.82	32.23	0.98	0.90	34.49	0.97	0.89	34.96
AXFLAIR_201_6002974	0.97	0.85	32.95	0.97	0.90	35.37	0.99	0.91	36.12
AXT1POST_209_6001084	0.98	0.93	37.67	0.98	0.91	36.42	0.99	0.95	39.75
AXT1PRE_203_6000645	0.98	0.91	36.89	0.98	0.91	36.43	0.99	0.93	37.36
AXT2_203_2030014	0.98	0.94	36.42	0.99	0.97	39.25	0.99	0.97	39.21
AXT2_202_2020244	0.98	0.72	33.56	0.99	0.93	35.56	0.99	0.90	36.56

7.5.3 Evaluation on External DICOM Data

To evaluate the generalization ability of the Multi-Loss ConvDecoder beyond the FastMRI dataset, we conducted additional experiments using external MRI scans in DICOM format. We performed this experiment using images of brain, knee, and spine MRI.

First, we extracted the image data from the DICOM files and simulated multi-coil acquisitions by generating coil sensitivity maps. Using these maps, we created fully-sampled Cartesian k-space data through a forward 2D Fourier Transform. To simulate the undersampling conditions used in Cartesian 4x MRI acceleration, we applied a variable-density Gaussian mask to the k-space data. The resulting undersampled data was then processed by our reconstruction pipeline in the same manner as the FastMRI inputs.

The model produced high-quality reconstructions that were visually consistent, indicating good generalization to data outside the original training domain.

Quantitative Results

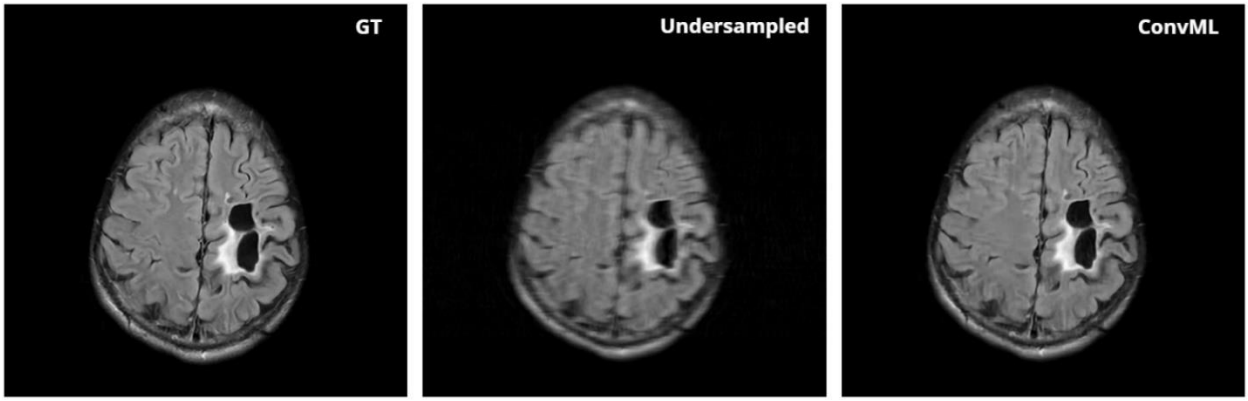


Figure 7.39 Brain DICOM Image CS Results

Figure 7.40 Knee DICOM Image CS Results

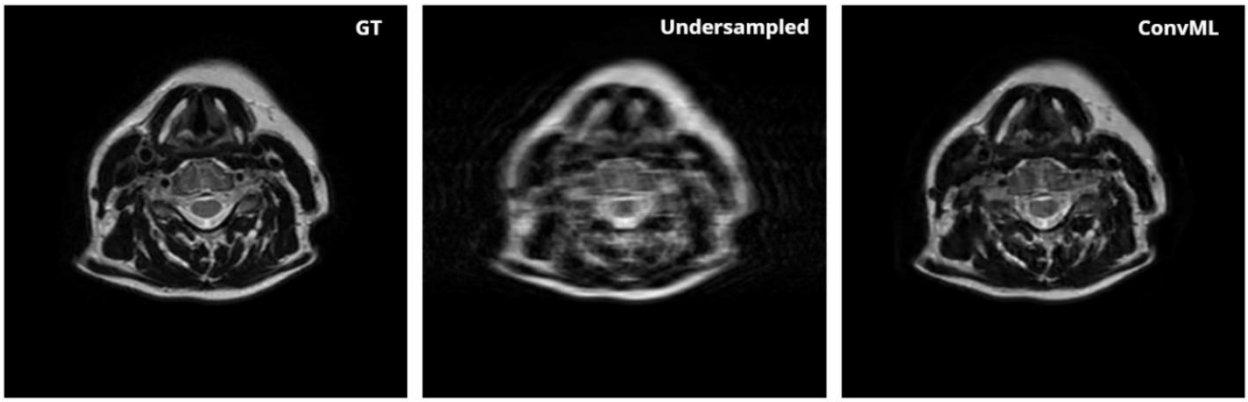


Figure 7.41 Spine DICOM Image CS Results

Qualitative Results

Table 7.13 Cartesian Dicom Scores

	ConvML	
	SSIM	**PSNR**
BRAIN1	0.97	38.15
BRAIN2	0.95	37.72
KNEE	0.90	36.39
SPINE1	0.95	31.68
SPINE2	0.88	31.66

7.5.4 Testing DICOM files using Non-Cartesian MultiLoss

As part of our evaluation, we extended the Multiloss ConvDecoder model to handle **non-Cartesian MRI data** by generating appropriate k-space measurements from DICOM files using non-Cartesian sampling trajectories. Unlike Cartesian sampling, where data points are collected on a uniform grid, non-Cartesian MRI acquires data along more complex trajectories such as **radial,** spiral. These alternative trajectories allow for more flexible and efficient coverage of k-space, especially in the low-frequency regions critical for image quality.

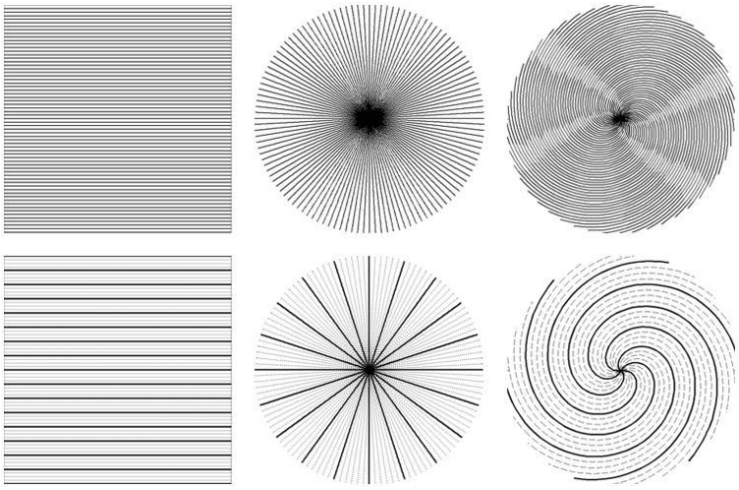


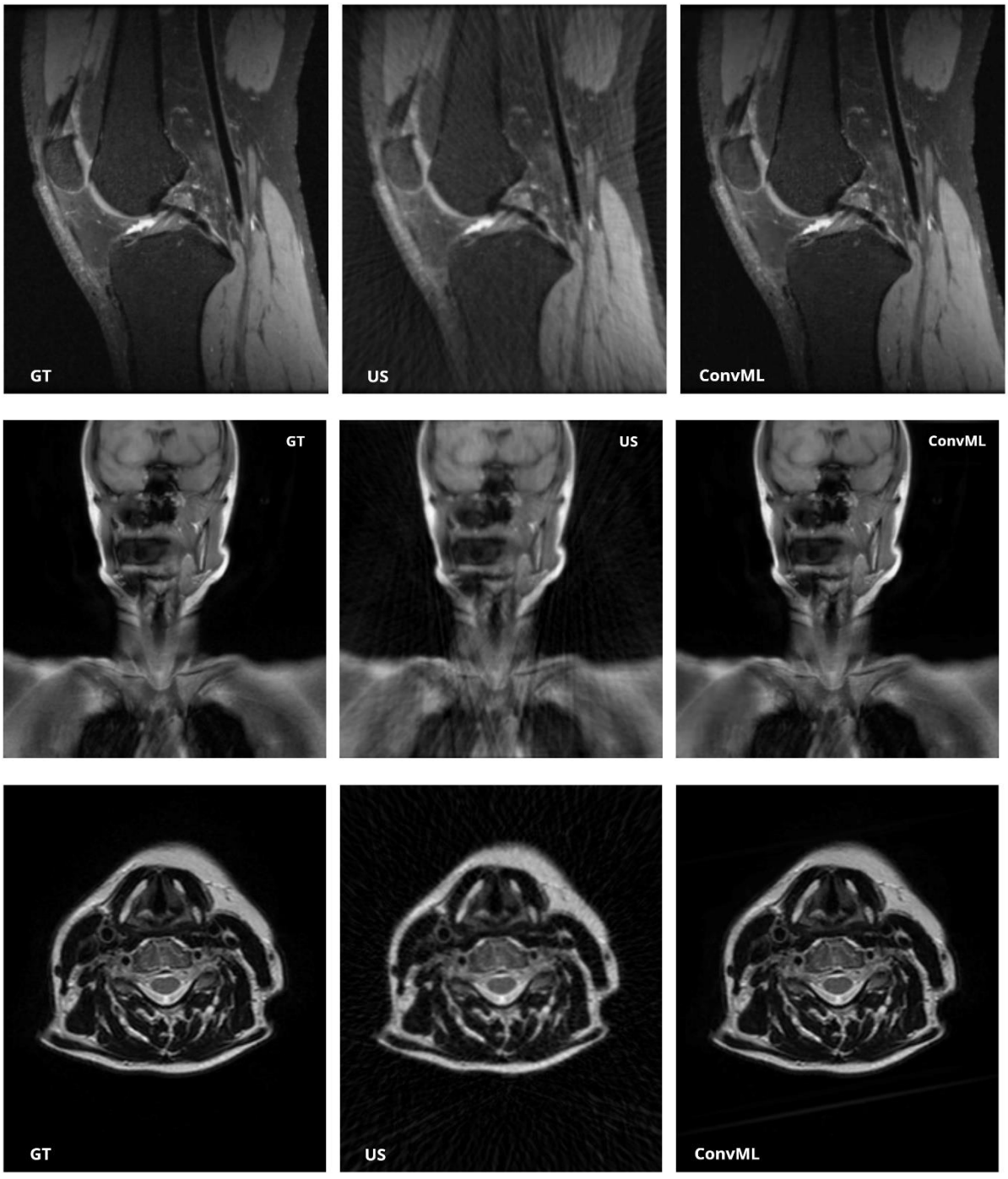
Figure 7.42 Cartesian vs. Non-Cartesian Sampling and Under-sampling Techniques

The use of non-Cartesian sampling introduces unique advantages, particularly in the context of under-sampling for acceleration. Non-Cartesian trajectories naturally lead to **incoherent aliasing artifacts**, which are more amenable to removal using reconstruction techniques based on untrained deep priors. This contrasts with Cartesian under-sampling, where aliasing appears more structured and harder to suppress.

To accommodate the irregular sampling positions, we integrated a **Non-Uniform Fast Fourier Transform (NUFFT)** operator into the forward model of MultiLoss ConvDecoder. This allowed the network to learn a reconstruction from sparse non-Cartesian k-space.

Our results on non-Cartesian-sampled data illustrate the flexibility of Multiloss model in adapting to different acquisition strategies. By comparing reconstructions from Cartesian and non-Cartesian k-space generated from the same DICOM source, we highlight the potential of Multiloss in realistic and diverse MRI settings.

7.5.4.1 Quantitative Results



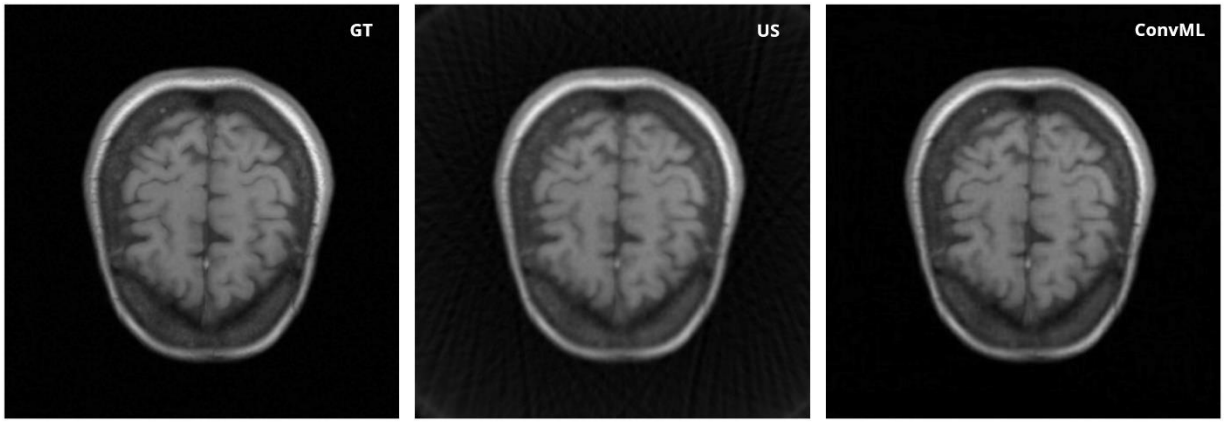


Figure 7.43 Reconstructed Non-Cartesian different MRIs

7.5.4.2 Qualitative Results

Table 7.14 Scores of Non-Cartesian Reconstruction with MultiLoss

	Multiloss	
	SSIM	**PSNR**
KNEE	0.97	38.65
SPINE	0.98	41.59
SPINE2	0.95	40.57
BRAIN	0.97	41.82
BRAIN2	0.97	40.67

Chapter Eight: Software Design

8 SOFTWARE DESIGN

8.1 INTRODUCTION

We created the Pixel Perfect web application to help users enhance their images effortlessly. The goal of Pixel Perfect is to make advanced image enhancement accessible without the need for pre-trained models. This platform provides easy-to-use tools like denoising, super-resolution, and inpainting. Users can upload their images, apply various enhancement techniques, and download the improved results—all in just a few clicks.

8.2 APPLICATION REQUIREMENTS

8.2.1 Functional Requirements

1. Image Processing Selection
    - **Feature Selection:** The authenticated user should be allowed to select either denoising, inpainting, or super-resolution functions.
2. MRI Processing Features:
    - **Denoising, Super-resolution, and Inpainting** for MRI scans, tailored for medical data.
3. MRI Image Upload:
    - **MRI Upload:** Support for uploading MRI files in H5 format, with checks for file size and format validation.
4. Image Upload and Processing
    - **Image Upload:** The user shall upload an image in a format supported by the system, such as JPEG and PNG. The system shall check the image format and size before processing.
    - **Backend Processing:** The uploaded pictures are transferred to the backend, where processing, depending on the selected feature, will be done. The backend will run whatever algorithm is necessary to enhance the quality of the image.
5. Result Presentation
    - **Comparison Display:** After processing, the application should display the original and the processed images side by side for comparison by the user.
    - **Download Processed Image:** There should be the provision for downloading the processed image in the desired format by the user.

- **Download Processed MRI:** Provide the ability to download the processed MRI in H5 format (as well as PNG/JPEG for general purposes).

6. Image Editing
    - **Edit Feature:** The Edit button shall allow the users to edit the preprocessed image by providing basic editing features such as feature of crop, draw and text on image.
7. Features of User Interface
    - **Dark Mode/Light Mode Toggle:** There shall be a toggle switch available to flip the application between dark mode and light mode for better user accessibility and user experience.

8.2.2 Non-Functional Requirements

1. Performance
    - The application must ensure minimum latency in the uploading and processing of images to ensure a responsible user experience
    - Ensure that MRI processing does not cause unacceptable delays due to file sizes and processing complexity.
2. Scalability
    - The system architecture will be able to scale, handling more and more users with image processing requests without degradation in performance.
    - The system should be able to scale to handle large MRI files, which may require more computational resources.
3. Data Integrity:
    - Ensure the processed MRI images retain all essential medical metadata and quality.
4. Security
    - Images should be stored and transmitted with the use of standard encryption schemes. The software should provide data protection regarding legislation.
    - Ensure secure handling of medical data, adhering to necessary privacy laws (e.g., HIPAA).
5. Usability
    - The user interface provided by the system should be as simple and easy to operate as possible so that the broad range of users with diverse computer skills could actuate the services provided by the solution.
6. Reliability

- The application should be highly available, with at least 99.9% uptime, and implement error handling and recovery mechanisms to ensure that the application continues to function even when failure conditions arise.

7. Maintainability
    - The codebase should follow best practices and coding standards to facilitate ease of maintenance and updates. Proper documentation and modular design are key to this.
8. Compatibility
    - The application should be compatible with major web browsers such as Chrome, Firefox, and Safari, and responsive across different devices and screen sizes.

8.3 TOOLS AND TECHNOLOGIES

8.3.1 Front-End Technologies

The front end is responsible for providing an interactive interface to users for uploading images, selecting options, and downloading the processed image.

- **HTML5**: For structuring the web pages.
- **CSS3**: For styling and ensuring a responsive user interface.
- **JavaScript**: For handling user interactions and dynamic content updates.
- **React.js**: For building a modern and reactive user interface. It also provides dynamic transitions and interaction between the backend and frontend.
- **Bootstrap**: For responsive design and pre-built UI components.

8.3.2 Back-End Technologies

The back-end processes user requests, manages authentication, and handles image processing tasks.

- **Python**: Ideal for its extensive libraries and frameworks for image processing and machine learning.
- **Flask:** Python framework for building robust APIs.
- **Node.js**: JavaScript Framework for server-side programming, suitable for data-intensive applications since it uses an asynchronous, event-driven model.

8.3.3 Additional Tools

- **GitHub:** For version control.
- **Google Colab:** Cloud-hosted Jupyter Notebook for collaborative python programming, used for model testing.

8.4 SYSTEM DESIGN

8.4.1 Use Case Diagram

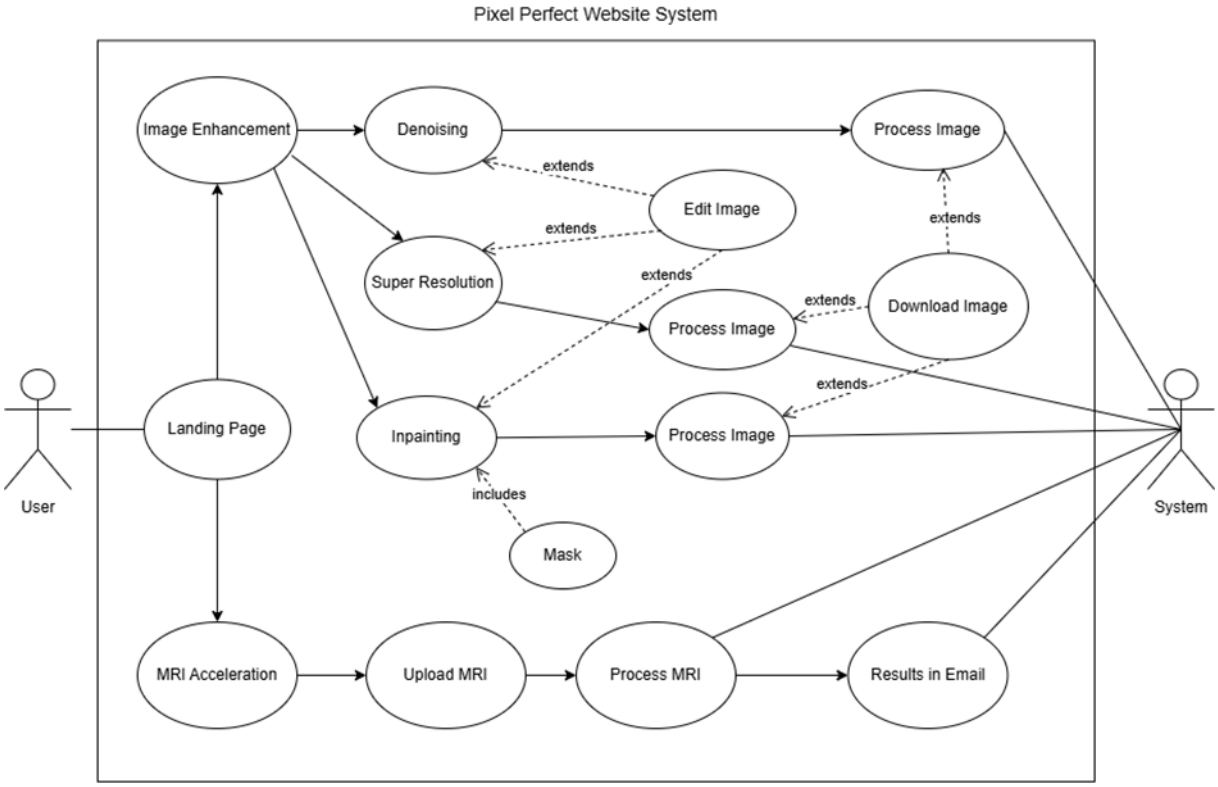


Figure 8.1 Software Use Case Diagram

8.4.2 Sequence Diagram

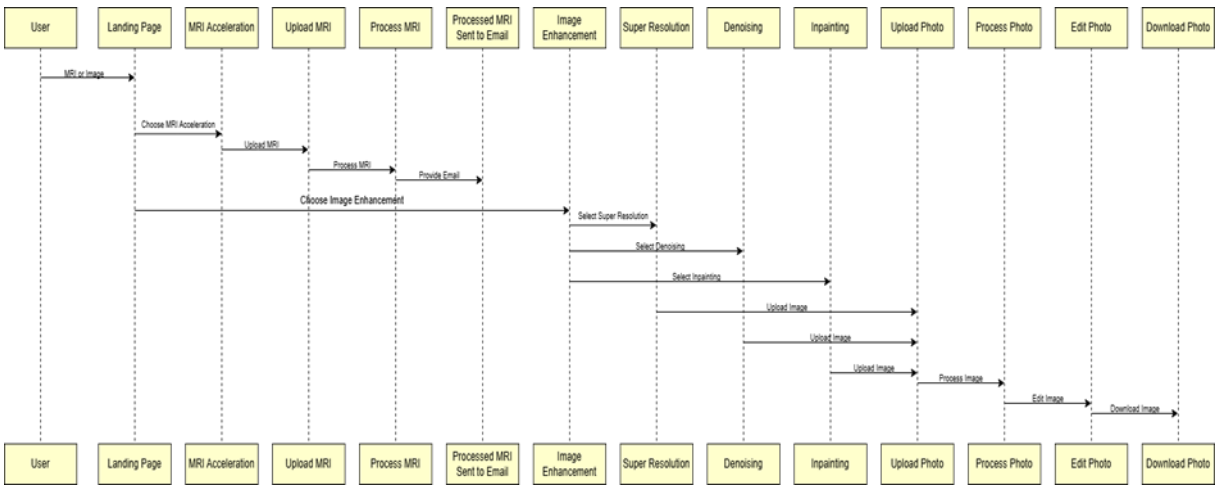


Figure 8.2 Software Sequence Diagram

8.4.3 Class Diagram

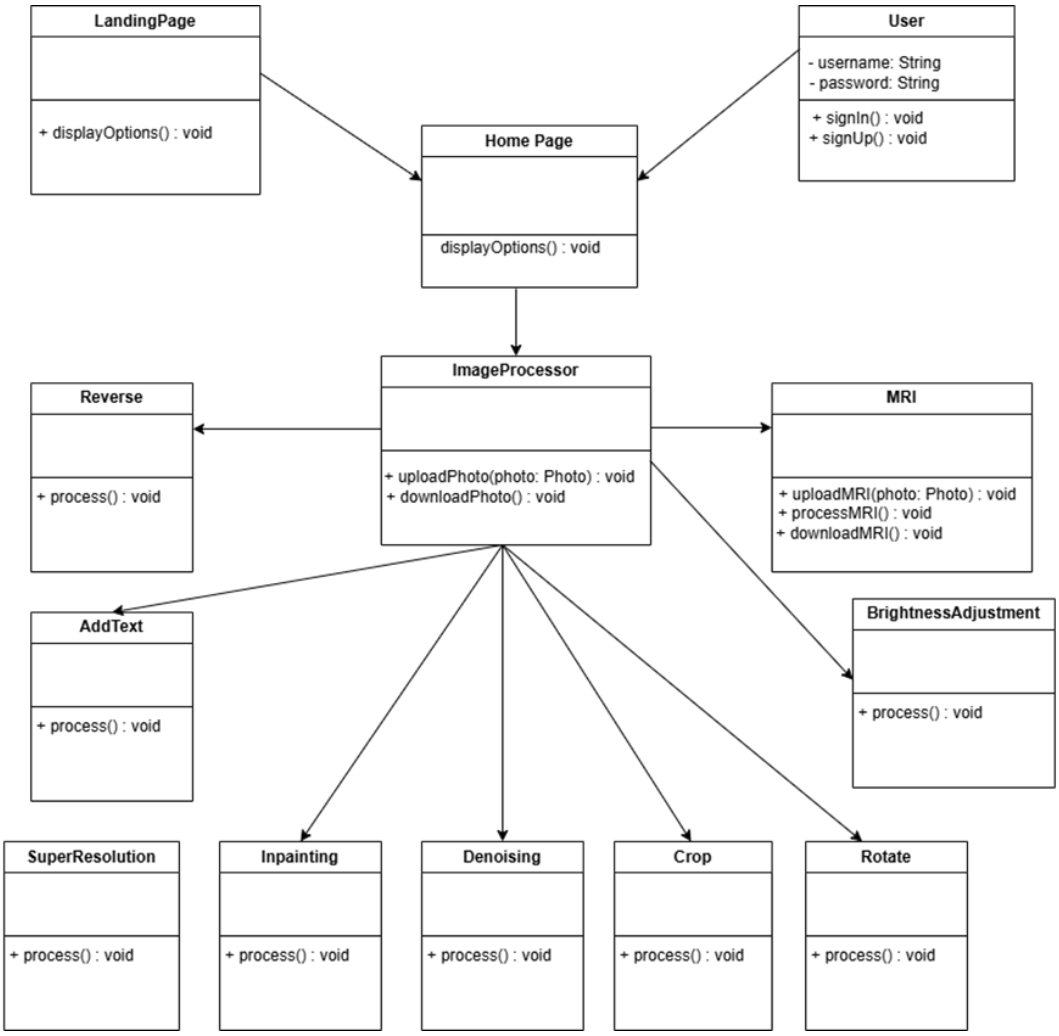


Figure 8.3 Software Class Diagram

8.4.4 Flowchart

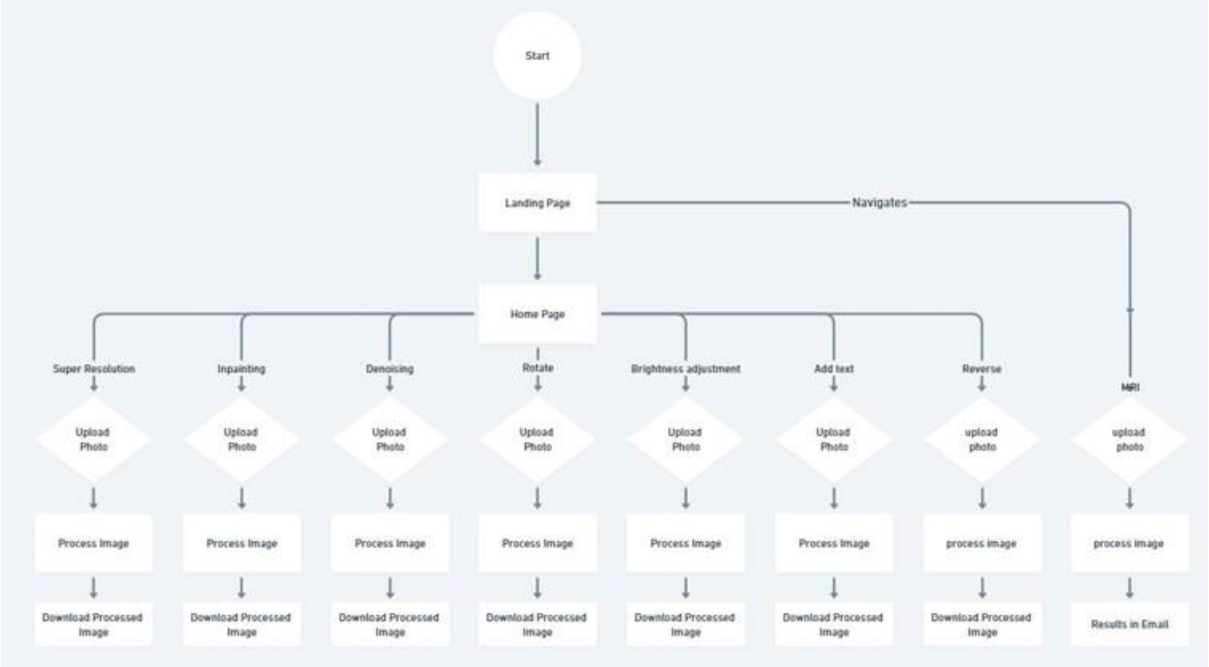


Figure 8.4 Software Flowchart

8.5 DATABASE DESIGN

1. No Image Storage in the Database:
    - Images uploaded by users are processed directly by the untrained models.
    - Since the models do not require persistent storage of images, the database is not used to store image files, reducing storage requirements and simplifying the database schema.
    - Temporary file storage or memory is used during processing, ensuring that no sensitive image data is retained unnecessarily.
2. Untrained Models:
    - The models used in this project are untrained, meaning they do not rely on pre-existing datasets for predictions or processing. Instead, they learn and operate dynamically based on the input provided.
    - This approach ensures flexibility and adaptability for custom tasks while avoiding the complexities of maintaining pre-trained model weights or large datasets.

Chapter Nine: Software Implementation

9 SOFTWARE IMPLEMENTATION

9.1 SOFTWARE DEVELOPMENT METHODOLOGY

Our team adopted an Agile methodology for the development of this project. We used the principles of iterative and incremental development to organize our work. The development process was split into phases, each phase containing one or more features to be implemented. This allowed us to work simultaneously on front-end, and back-end, and machine learning features, and deliver functional iterations of the application progressively.

To ensure effective communication and coordination, we established a flexible timeline to fit our schedules at university. Every few days, we conducted quick status updates to share progress, identify roadblocks, and make minor adjustments as needed. Every two weeks, we held in-person meetings to evaluate progress, discuss challenges, plan our next steps, and ensure alignment with our supervisor's expectations.

This collaborative approach allowed us to remain adaptable and responsive to changes while maintaining a steady momentum throughout the project. It also fostered accountability and transparency within the team, ensuring that everyone stayed engaged and contributed meaningfully to the project's success.

9.2 UI/UX DESIGN

The UI/UX design of our website focuses on delivering a seamless, user-friendly, and visually appealing experience. It incorporates modern design principles, ensuring accessibility and functionality for a wide range of users. One of the standout features of our design is the inclusion of **light and dark modes**, allowing users to choose the theme that best suits their preferences or environment.

To achieve a professional and cohesive aesthetic, **Canvas** was used as the primary design tool for creating visual assets, ensuring high-quality visuals and consistency across the platform.

9.2.1 Website Design Approach

The design process was structured around the following principles:

1. **User-Centered Design**:
    - The interface focuses on delivering intuitive navigation and a streamlined experience, ensuring accessibility for all users.
2. **Visual Aesthetics**:
    - A modern, clean, and engaging design was achieved by selecting harmonious color schemes, typography, and layouts.
3. **Responsiveness**:
    - The design is fully responsive, ensuring seamless usability across various devices, including desktops, tablets, and smartphones.
4. **Consistency**:

- Consistent design elements were applied across all pages to maintain a unified and professional appearance.

Canvas played a pivotal role in designing the website's interface, allowing for the efficient creation of visually appealing assets. Canvas is an online graphic design platform widely used for creating high-quality visuals. It features:

- A drag-and-drop interface for effortless customization.
- A rich library of templates, fonts, icons, and design elements.
- Collaboration tools for team-based workflows.

Canvas was instrumental in crafting key elements for the website, including:

- **Landing Page:** Visually user-friendly design with eye comforting design to make the user experience better.
- **Homepage Banners**: Vibrant and visually engaging banners to highlight the website's features.
- **Custom Icons and Illustrations**: Graphics tailored to each page's functionality.
- **Typography and Color Schemes**: Experimentation with fonts and colors to create a professional and cohesive aesthetic.
- **Supporting Visuals**: Marketing assets and UI components consistent with the website's overall branding.

By using Canvas, we ensured that the design process was both efficient and flexible, enabling the delivery of a polished, professional interface.

9.2.2 Website Pages Overview

The website comprises the following main pages, each designed to provide a seamless user experience and ensure functionality:

1. Landing Page:

- The Landing Page is the first point of interaction for users visiting the website. It serves as an introduction to the platform and its core capabilities.
- The design is minimalistic and clean, aiming to create a professional first impression while highlighting the project's objectives.
- Key elements include a project title, tagline, and a short description, often accompanied by a call-to-action that guides users to explore more.
- Navigation links are placed prominently to direct users to core features such as the MRI Acceleration module, Image processing tools which are denoising, inpainting and super-resolution.

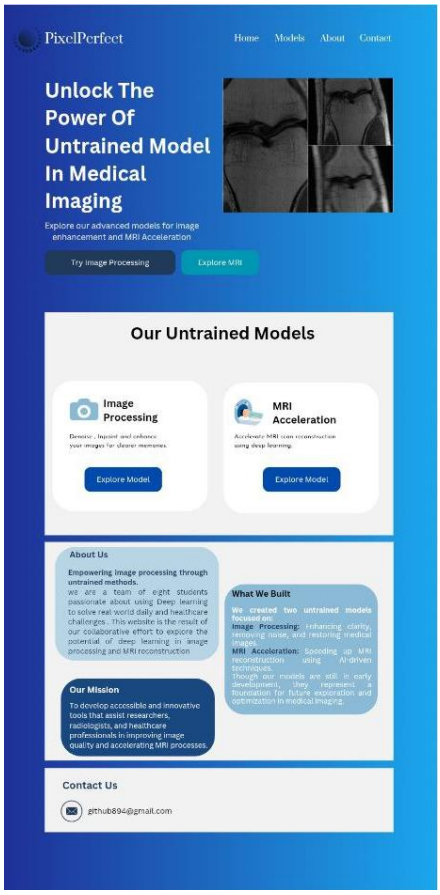


Figure 9.1 Landing Page User Interface

2. MRI Acceleration Homepage:

- This dedicated page focuses specifically on the MRI Acceleration functionality of the project.
- It provides a user-friendly interface where users can upload MRI data, configure acceleration parameters, and visualize the outputs.
- The page is structured to support both new and returning users, offering tooltips or brief guides for ease of use.

Figure 9.2 MRI Acceleration Homepage User Interface

3. MRI Acceleration:
The MRI Reconstruction page is designed with a clean and intuitive interface to guide users through a three-step process for MRI file submission and result retrieval. The interface is built to ensure clarity and ease of use, even for non-technical users.

- The left panel provides a visual step-by-step guide
- Upon uploading the .h5 MRI file, users are prompted to enter their email address through a modal popup form.
- A simple and modern input form allows users to submit their email, which is then used to:

    - Confirm that the MRI file is being processed.
    - Deliver the final reconstructed image directly to their inbox once processing is complete.

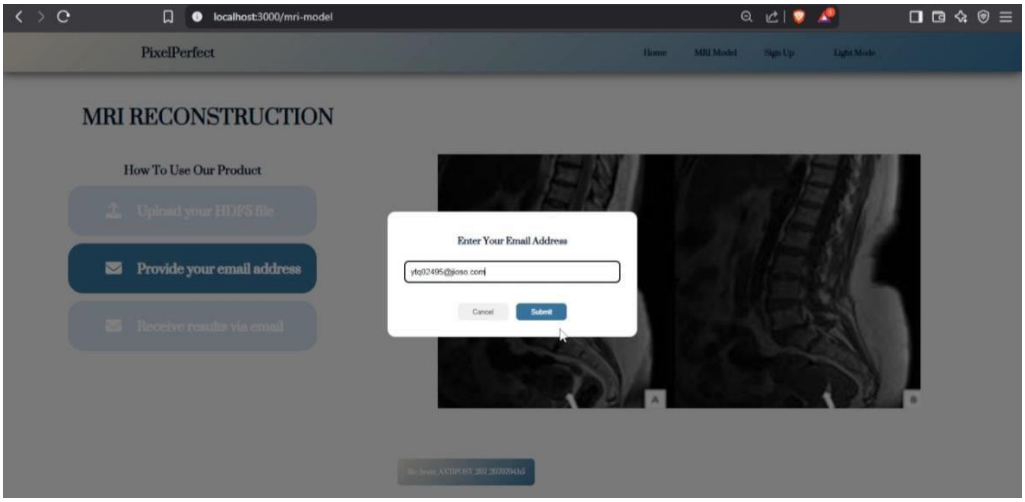


Figure 9.3 MRI Acceleration User Interface

4. Image Processing Homepage:

- The homepage functions as a central hub, offering an overview of the platform's features and services with Image Processing tools.
- It includes a clear navigation bar, vibrant banners designed with Canvas, and quick access links to the key features.

The design emphasizes simplicity and clarity to attract and engage users immediately

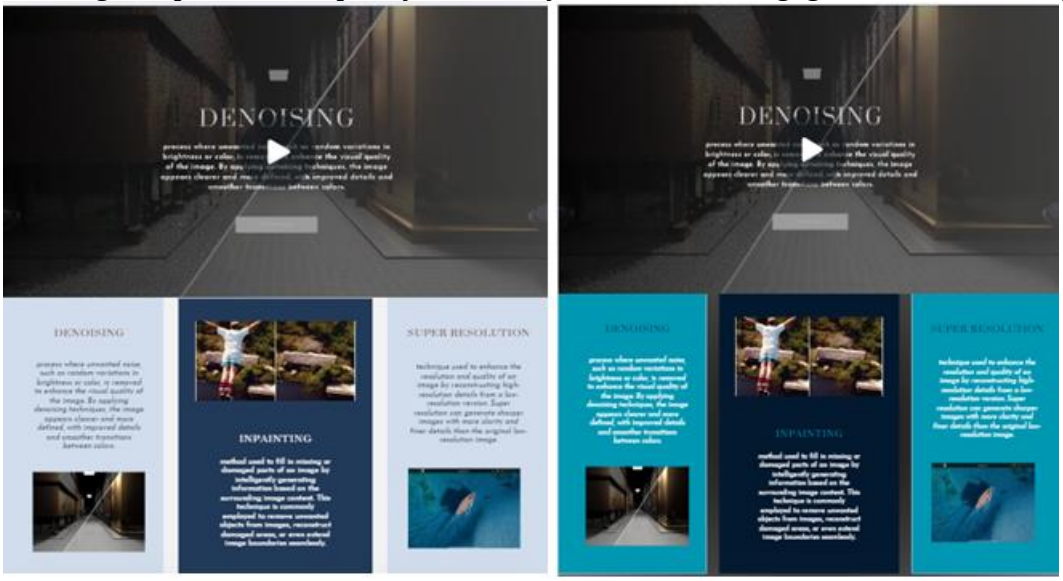


Figure 9.4 Image Processing Homepage User Interface

5. Denoising Page:

- This page enables users to upload and process images for noise reduction.
- The layout is simple and functional, with color schemes adjusted to suit the selected theme (light or dark mode).

Figure 9.5 Denoising User Interface

6. Super-Resolution Page:

- Users can enhance the resolution of their images on this page.
- The interface incorporates icons and buttons designed to look crisp and visually consistent in both modes.

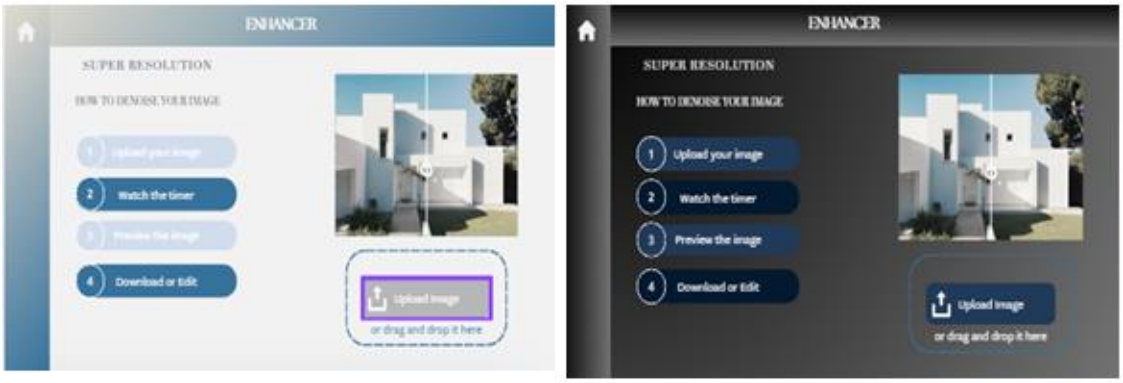


Figure 9.6 Super Resolution User Interface

7. Inpainting Page:

- This page allows users to repair damaged images by filling in missing parts intelligently.
- The design ensures ease of use, with adaptive visuals and backgrounds optimized for light and dark themes.

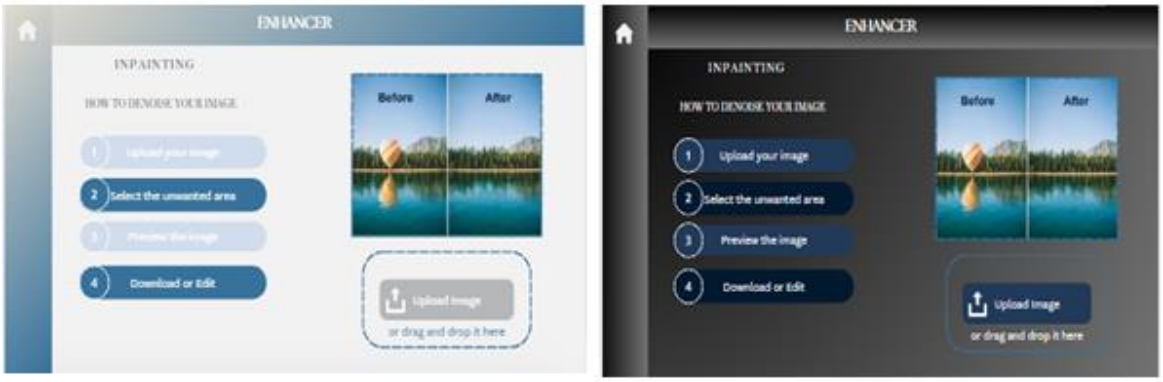


Figure 9.7 Inpainting User Interface

Light and Dark Modes

To enhance usability and accommodate different user preferences, the website includes a light and dark mode toggle.

- Light Mode:
  - Designed for environments with bright lighting.
  - Features lighter backgrounds and dark text for optimal readability.
- Dark Mode:
  - Ideal for low-light environments, reducing strain on the eyes.
  - Incorporates darker backgrounds and light text while preserving vibrancy in key elements like buttons and icons.

The seamless transition between these modes ensures a consistent and user-friendly experience, regardless of the selected theme.

9.3 FRONT-END DEVELOPMENT

The front-end development of the application utilizes React.js, a powerful JavaScript library for building user interfaces, complemented by CSS for styling and responsiveness. React.js was opted due to its component-based architecture that facilitates reusability and effectively manages state.

9.3.1 Technology Stack

- **React.js:** Used for effective creation of dynamic and interactive user interfaces. React's Virtual DOM ensures efficient rendering and hence updates, improving performance.
- **CSS/SCSS:** Used for component styling and the application of responsive design principles. SCSS is a CSS preprocessor that helps create maintainable and scalable stylesheets.
- **Redux:** Utilized for state management to handle complex state interactions across the application coherently and predictably.
- **React Router:** This is used for managing navigation within the single-page application, hence allowing for smooth transitions between different views such as signup, signin, and image processing pages.

9.3.2 Design Principles

1. **Component-Based Architecture**
   - The application is modularized into reusable components for different functionalities, such as image upload interfaces, and displays for processing results. This allows for easier maintenance and scaling of the code in the future.
2. **Responsive Design**
   - By using CSS Flexbox and Grid layouts, an application will guarantee compatibility with every type of device and screen size. Media queries have been set to dynamically switch between layouts to offer the best experience for desktops, tablets, and mobile devices.

3. **Accessibility**
    - Front-end design is per WCAG and hence makes the application accessible even to users with disabilities. Some of the features are keyboard navigability, appropriate contrast ratios, and ARIA attributes.
4. **Implementation of Dark Mode**
    - A feature of theme toggling allows users to toggle between dark and light mode. This utilizes CSS variables and React context-based theming to dynamically set styles based on the currently chosen theme, increasing user comfort and accessibility.
5. **State Management and Data Flow**
    - Effective state management is made possible using Redux. This allows for centralized application state control, thereby guaranteeing predictable data flow. Such a strategy has its positive impacts on debugging, and it enhances the application's robustness and reliability.
6. **Performance Optimization**
    - Code-splitting and lazy loading techniques are employed to reduce initial load times and improve performance. Additionally, React.memo and useCallback hooks are utilized to prevent unnecessary re-renders, ensuring a smooth user experience.
7. **User Feedback and Error Handling**
    - The interface includes visual indicators such as progress spinners and toast notifications to provide users with clear feedback during processing operations and to handle errors gracefully.
8. **Security Practices**
    - Basic client-side validation is implemented for file uploads to ensure only valid and supported file types are processed.

9.3.3 User Interface Components

- **Feature Selection Dashboard:** A clear and user-friendly interface for selecting functionalities (Image Processing, MRI Acceleration).
- **Image Upload Interface:** The area for dropping or selecting image files that are to be uploaded; progress indicators and error handling.
- **Display of the Results of Processing:** Original and processed image on side-by-side comparison with interactive buttons for downloading and editing.
- **Editing Utilities:** Integrated editing features including cropping, drawing, and text addition are provided by utilizing canvas libraries to allow changes in real-time.
- **Theme Toggle:** A responsive switch component allows users to easily switch between dark and light themes.
- **MRI Acceleration Form:** Allows users to upload MRI H5 files, input their email, and submit the data for processing.

The front-end development focuses on user experience, performance, and maintainability to make sure the application will be functional as well as appealing.

9.4 BACK-END DEVELOPMENT

The backend development of the project plays a crucial role in managing data, ensuring smooth transitions between features, and facilitating secure communication between the user interface and the database.

9.4.1 Website Backend

The website backend serves as the core of the project, handling user registration, file uploads, data processing, and communication between the frontend and processing models. It uses a combination of **JavaScript** and **Python** to manage both real-time and asynchronous interactions.

Key responsibilities of the website backend include:

- Model Execution via Python and Flask:
  - The backend integrates several machine learning models hosted in Google Colab, executed via Flask servers built in Python.
  - Flask handles incoming HTTP requests from the frontend and routes them to the corresponding processing pipeline.
- MRI Acceleration (Asynchronous with Email Response):
  - Users upload an .h5 MRI file and provide their email address.
  - A Flask endpoint triggers the MRI model hosted on Colab, and an initial confirmation email is sent to the user.
  - After processing, a second email containing the final image result is sent to the user.
- Image Processing Models (Synchronous with Web Response):
  - The Denoising, Inpainting, and Super-resolution models accept .png or .jpg images.
  - These are processed in real time via Flask, and the output images are rendered directly on the website page.
  - Users receive immediate visual feedback without needing email communication.
- Smooth Transitions:
  - Transitions and dynamic interactions were achieved using **JavaScript**.
  - The backend communicates seamlessly with the frontend to ensure an efficient and user-friendly experience.

9.4.2 API Integration

The backend utilizes APIs to manage communication between the website frontend, Flask servers, external processing environments (Google Colab), and the database.

Key APIs used in the project include:

- Internal API Calls for Processing & Data Flow:
  - Flask-based APIs handle uploads, trigger model execution, and manage result retrieval.
  - The system supports both synchronous (for image models) and asynchronous (for MRI acceleration) workflows

Workflow of API Usage:

1- **Image Processing (Denoising, Inpainting, Super-resolution):**
    - User uploads .png or .jpg file through the web interface.
    - Flask API sends the file to the appropriate model (hosted via Colab or backend server).
    - The processed image is returned immediately and displayed on the result page.
2- **MRI Acceleration:**
    - User uploads an .h5 MRI file and submits their email address.
    - Flask triggers the MRI model running in Colab.
    - A confirmation email is sent indicating that the file is under processing.
    - Once the model finishes, the backend sends a final email with the processed image attached or linked.

This hybrid API system enables fast real-time interaction for image tools and reliable asynchronous processing for heavier MRI data.

The workflow begins when a user accesses the website. They enter their email address, select an image for upload, and click the 'Upload' button. The system processes the image in the backend and delivers the final result to the user via email.

For storage, the system utilizes GitHub as a file repository and CDN. Email delivery is handled through the Resend API. A lightweight Node.js server serves as an intermediary, specifically handling email transmission. This server is located at /server/node/email-proxy.js and receives email parameters from the React frontend.

Important Note: This architecture is designed to distribute API calls effectively, implementing load balancing to reduce server strain when making requests to the Resend API on Google Colab.

The Flask API provides essential functions:

- A filename generator that combines the user's email with the current timestamp
- An image uploader that stores files to GitHub and returns accessible URLs
- An email composer that creates HTML content for image previews
- A mail sender that delivers messages to users

The email process has two stages:

1. An initial "MRI start" notification
2. A follow-up "MRI end" message containing the downloadable image link

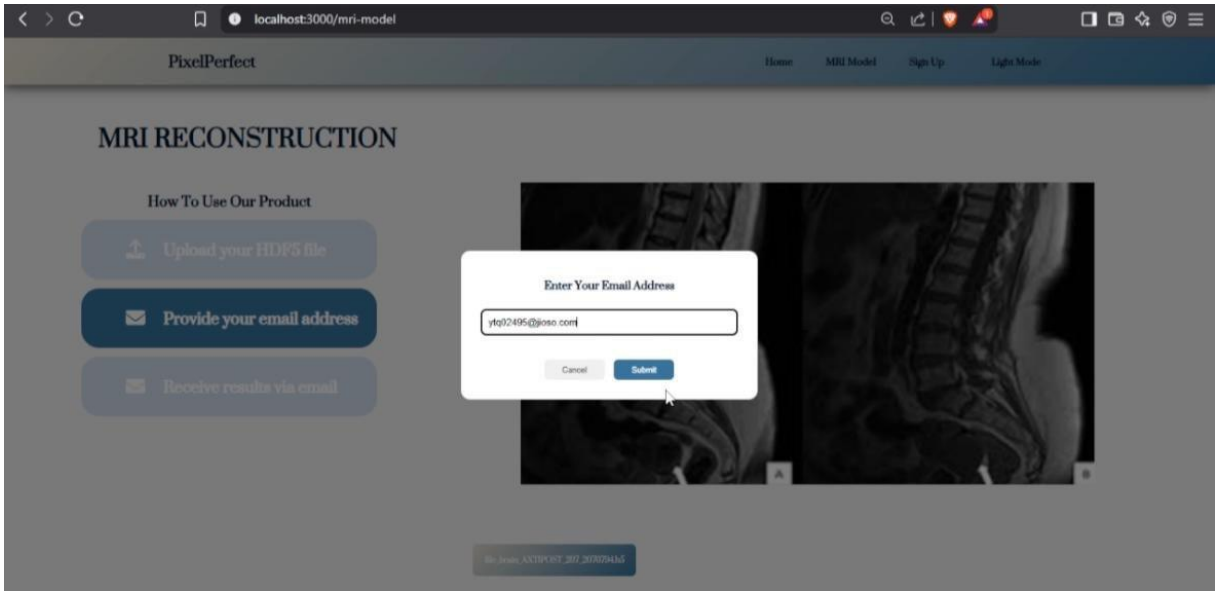


Figure 9.8 Email Notification Preview

Figure 9.9 API Workflow Diagram

References

[1] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep Image Prior," *arXiv preprint arXiv:1711.10925*, Nov. 2020.

[2] R. Heckel and P. Hand, "Deep Decoder: Concise Image Representations from Untrained Non-convolutional Network," *arXiv preprint arXiv:1810.03982*, Oct. 2019.

[3] M. Z. Darestani and R. Heckel, "Accelerated MRI with Un-trained Neural Networks," IEEE Transactions on Computational Imaging, vol. 7, pp. 724-733, Apr 2021.

[4] T. M. Siedler, P. M. Jakob, and V. Herold, "Enhancing quality and speed in database-free neural network reconstructions of undersampled MRI with SCAMPI," Magnetic Resonance in Medicine, vol. 92, no. 3, pp. 1232-1247, Sep. 2024.

[5] Muhammad Umair Danish , "A Comparative Study of Image Denoising Algorithms “,*arXiv preprint arXiv:2412.05490*, Dec. 2024.

[6] "Noise profoundly impacts medical imaging," *Diagnostics*, vol. 14, no. 13, p. 1328, Jul. 2023. [Online]. Available: https://www.mdpi.com/2075-4418/14/13/1328

[7] Ajay Kumar Boyat , Brijendra Kumar Joshi, "A REVIEW PAPER: NOISE MODELS IN DIGITAL IMAGE PROCESSING" *arXiv preprint arXiv:1505.03489*, May 2015.

[8] Onyedinma, E. G. and Onyenwe, I. E., "Image Restoration: A Comparative Analysis of Image De noising Using Different Spatial Filtering Techniques" *arXiv preprint arXiv:2401.09460*, Jan. 2024. [Online].

[9] "AI and Healthcare," *Pew Research Center*, Feb. 2023. [Online]. Available: https://www.pewresearch.org/wp-content/uploads/sites/20/2023/02/PS_2023.02.22_AI-health_REPORT.pdf

[10] M. Brown et al., "Radiology Artificial Intelligence Applications," *RSNA Radiology: Artificial Intelligence*, vol. 5, no. 2, p. 337, Mar. 2023.

[11] A. Lee, "Analyzing the Impact of Diagnostic AI in Modern Healthcare," *Journal of Student Research*, vol. 11, no. 3, pp. 4142–4148, 2023.

[12] "Pulse oximeters don’t work well on darker skin, leading to flawed COVID care," *UCSF Epidemiology and Biostatistics News*, Jan. 2023. [Online]. Available: https://epibiostat.ucsf.edu/news/pulse-oximeters-dont-work-well-darker-skin-leading-flawed-covid-care

[13] F. Merizzi, P. Saillard, O. Acquier, E. Morotti, E. L. Piccolomini, L. Calatroni, and R. M. Dessì, "Deep image prior inpainting of ancient frescoes in the Mediterranean Alpine arc," *arXiv preprint arXiv:2306.14209*, 2023.

[14] Y. Melekhov, J. Ylioinas, J. Kannala, and E. Rahtu, "Image-based localization using hourglass networks," *arXiv preprint arXiv:1703.07971*, 2017.

[15] M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, and C. Pal, "The importance of skip connections in biomedical image segmentation," in *Deep Learning and Data Labeling for Medical Applications*, G. Carneiro et al., Eds. Cham: Springer, 2016, pp. 179–187

[16] G. Xu, X. Wang, X. Wu, X. Leng, and Y. Xu, "Development of skip connection in deep neural networks for computer vision and medical image analysis: A survey," *arXiv preprint arXiv:2405.01725v1*, 2024.

[17] E. Orhan and X. Pitkow, "Skip connections eliminate singularities," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2018.

[18] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *arXiv preprint arXiv:1411.4038*, 2014.

[19] T. van Leeuwen and C. Brune, "Inverse Problems and Imaging." Available: https://tristanvanleeuwen.github.io/IP_and_Im_Lectures/intro.html

[20] Z. Wang, J. Chen and S. C. H. Hoi, "Deep Learning for Image Super-resolution: A Survey," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 10, pp. 3365-3387, Oct. 2021

[21] A. Montanaro, D. Valsesia, and E. Magli, "Exploring the Solution Space of Linear Inverse Problems with GAN Latent Geometry," arXiv preprint arXiv:2207.00460v1 [eess.IV], Jul. 2022.

[22] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. Huang, "Free-Form Image Inpainting with Gated Convolution," arXiv preprint arXiv:1806.03589v2 [cs.CV], Oct. 2019.

[23] I. Singh and N. Neeru, "Performance Comparison of Various Image Denoising Filters under Spatial Domain," Int. J. Comput. Appl., vol. 96, no. 19, pp. 21-30, Jun. 2014, doi: 10.5120/16903-6969.

[24] M. Qureshi, M. Kaleem, and H. Omer, "Journey through k-space: an interactive educational tool," *Biomedical Research*, vol. 28, no. 4, pp. 1475-1481, 2017.

[25] magnetic-resonance.org, "MR Image Contrast." Accessed: May 2025. [Online]. Available: https://www.magnetic-resonance.org/ch/10-02.html

[26] A. Simkó *et al.*, "Improving MR image quality with a multi-task model, using convolutional losses," *BMC Med. Imaging*, vol. 23, no. 1, p. 201, Nov. 2023, doi: 10.1186/s12880-023-01109-z

[27] M. Lyu *et al.*, "M4Raw: A multi-contrast, multirepetition, multi-channel MRI k-space dataset for low-field MRI research," *Sci. Data*, vol. 10, no. 1, p. 308, May 2023, doi: 10.1038/s41597-023-02181-4.

[28] "100 Knee MRI Cases," CAI2R, New York, NY, USA. Accessed: Jun. 7, 2025. [Online]. Available: https://cai2r.net/resources/100-knee-mri-cases/

[29] R. Gupta, "Backend Processing for Image Enhancement: Techniques and Challenges," IEEE Comput. Vision J., vol. 8, no. 2, pp. 210-220, April 2023.

[30] K. Patel, "Integrating image editing functionality within web interfaces," IEEE Transactions on Human-Machine Systems, vol. 51, no. 4, pp. 300-310, Aug. 2021.

[31] J. Doe, "Performance Optimization in Web Applications," IEEE Software Performance, vol. 15, no. 2, pp. 45-53, April 2023.

[32] A. Kumar and S. Singh, "Scalable Architectures for Web-Based Image Processing," IEEE Cloud Computing, vol. 10, no. 1, pp. 25-34, Feb. 2022.

[33] M. Jackson, "React.js: A Comprehensive Overview," IEEE Software, vol. 35, no. 4, pp. 40-47, July 2018.