BME1312 Project1 Report: Deep Learning for MRI Reconstruction

Wenye Xiong
2023533141
xiongwy2023@shanghaitech.edu.cn

Renyi Yang
2023533030
yangry2023@shanghaitech.edu.cn

Jiaxing Wu
2023533160
wujx2023@shanghaitech.edu.cn

Boyang Xia
2023533073
xiaby2023@shanghaitech.edu.cn

Fengmin Yang
2023533183
yangfm2023@shanghaitech.edu.cn

Abstract

This project uses deep learning to reconstruct high-quality dynamic MRI images from undersampled data. We propose a deep-learning-based denoising framework combining two independent UNet modules (processing real and imaginary components separately) and a 3D ResNet to explore the temporal correlation. We generate variable density undersampling patterns with acceleration factor 5 and 11 central k-space lines per frame, analyze the resulting aliasing artifacts, and evaluate reconstruction performance with PSNR and SSIM metrics. Additionally, we investigate the effects of dropout, dynamic learning rate schedules, and compare L1 versus L2 losses. Finally, we explore an unrolled network architecture incorporating data consistency layers between cascaded instances of our base reconstruction network. Access our code, dataset, and results at `https://github.com/XiongWenye/Deep-Learning-Dynamic-MRI-Reconstruction`.

1 Introduction

Dynamic magnetic resonance imaging (MRI) enables visualization of physiological processes by capturing multiple images over time, but faces a fundamental tradeoff between spatial resolution, temporal resolution, and scan duration. Accelerated MRI acquisition through k-space undersampling can address this challenge but introduces aliasing artifacts that degrade image quality. This project explores deep learning approaches to reconstruct high-quality dynamic MRI images from significantly undersampled k-space data (acceleration factor 5).

We develop and evaluate a novel reconstruction framework with three key components: (1) a dual-branch UNet architecture to separately process real and imaginary components of complex MRI data, (2) a 3D ResNet to leverage temporal correlations across dynamic frames, and (3) data consistency layers in an unrolled network configuration to enforce fidelity to the acquired measurements. Throughout our experiments, we investigate how different design choices—including dropout regularization, learning rate schedules, loss functions, and network architecture—affect reconstruction quality as measured by Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM).

This report documents our methodology, implementation details, experimental results, and analysis across five main sections: undersampling pattern generation, network architecture design, ablation

studies on regularization techniques, comparison of loss functions, and exploration of unrolled network architectures. Our findings provide practical insights for designing effective deep learning solutions for dynamic MRI reconstruction.

2 Variable Density Random Undersampling Pattern Generation and Aliasing Artifacts Analysis

Note: You may need `train.py`, line 29, function `variable_density_mask()` and line 456, function `process_data()` for reference in this section.

2.1 Undersampling Pattern Generation

We were asked to generate a variable density random undersampling mask U matching the cine dataset size, with acceleration factor 5 and 11 fully sampled central k-space lines per frame. The mask values are 1 at sampled locations and 0 otherwise.

The provided code generates a 4D undersampling mask (shape [batch, frames, height, width]) using the following steps:

1. Initialization: A zero-initialized mask is created with the input shape.
2. Central Region Sampling: For each frame, 11 central k-space lines are sampled to retain low-frequency spatial information.
3. Variable Density Probability Distribution: A quadratic probability density function is defined, decreasing with distance from the k-space center.
4. Random Peripheral Sampling: The remaining lines are sampled randomly based on the probability distribution we obtained.
5. Dynamic Frame Variability: Each dynamic frame independently generates unique random peripheral lines to ensure temporal incoherence of aliasing artifacts.

2.2 Undersampling Masks Plotting

Figure 1 shows the undersampling mask for one dynamic frame and undersampling masks in the ky-t dimension.

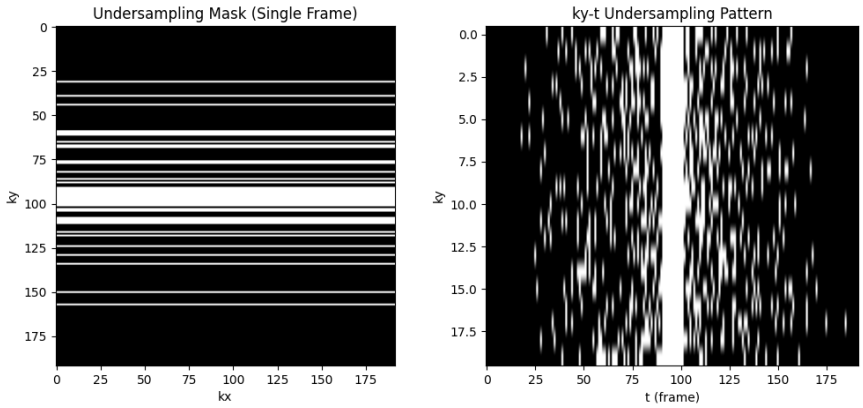


Figure 1: Undersampling Mask for one dynamic frame and undersampling masks in the ky-t dimension.

Thus we can see that the sampling mask is not the same for different dynamic frames. That's because sampling positions are randomized per frame to disrupt coherent aliasing artifacts in time. Figure 2 shows masks for multiple frames.

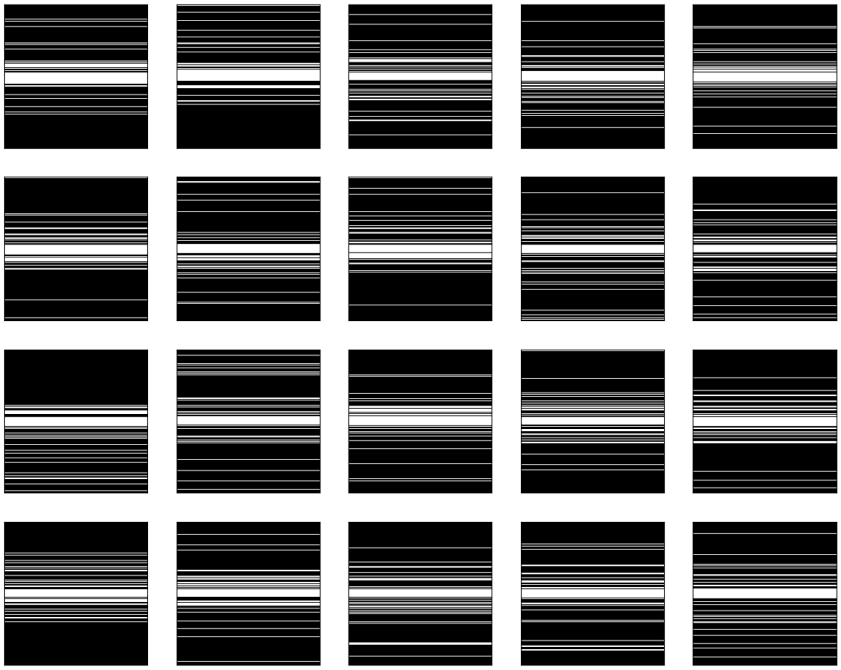


Figure 2: Multiple sampling masks showing the variable density patterns across different temporal frames.

2.3 Aliased Images Depiction and Comparison

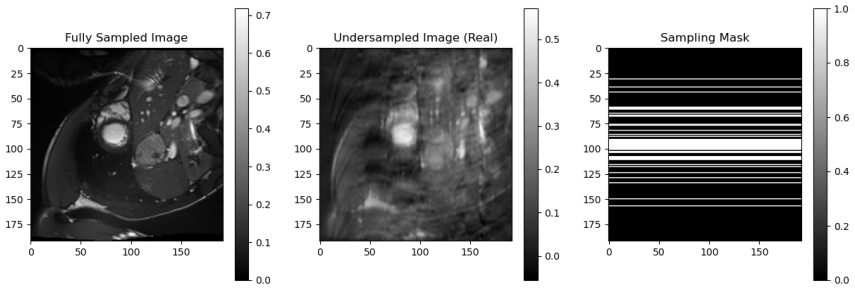
From the stem of this question we can know that, given a fully sampled image m, the aliased image b is computed as:

$$b = F^{-1}UFm$$

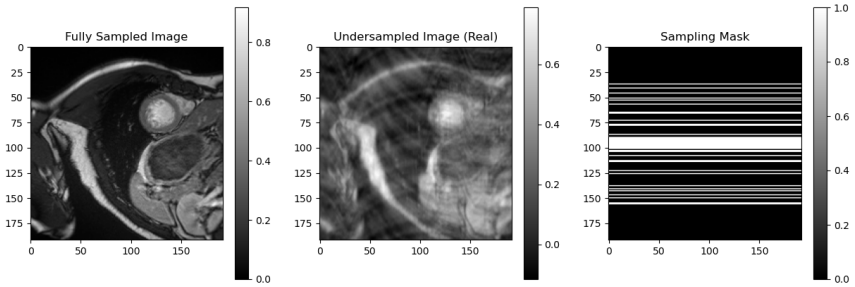
where F is the Fourier transform.

In function `process_data()`, we first load the dynamic MRI dataset dataset from the `cine.npz` file, and convert the data into PyTorch tensor labels, which are used as ground truth for subsequent processing. Then we call the function `variable_density_mask()` that we've just implemented to generate a mask, and then apply the mask to the k-space data, and convert the fully sampled image data to the frequency domain by Fourier transform (`image2kspace`). After that, we undersample the k-space data through the mask, and perform an inverse Fourier transform (`kspace2image`) on the undersampled k-space to generate an aliased image.

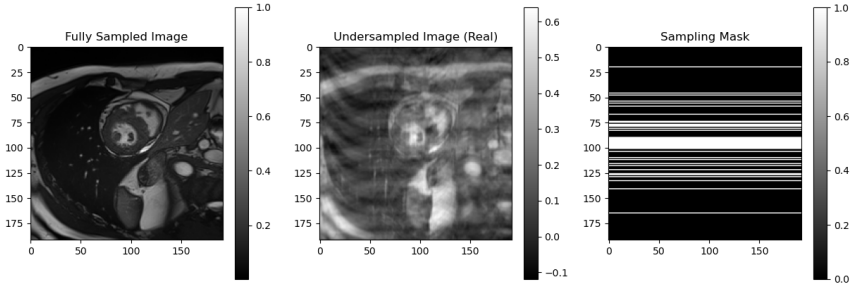
Figure 3 compares the fully sampled image, aliased image, and sampling mask in Fourier domain. Here we choose the first 3 images generated to better compare the aliased images with the fully sampled image.



(a) Comparison for frame 0



(b) Comparison for frame 1



(c) Comparison for frame 2

Figure 3: Comparison between fully sampled (left), undersampled (middle), and corresponding sampling mask (right) for different frames.

Description of aliased images: Since the low-frequency information in the center k-space is usually better preserved, the main structure of the image is displayed more clearly; however, due to the lack of high-frequency information, the edges of the image are blurred with repetitive ripples (i.e., artifacts), especially in dynamic imaging, where the artifacts vary with time frames.

Comparison of aliasing image and fully sampled image: fully sampled image retains complete high-frequency information, with clear and detailed edges and no artifacts, whereas aliasing image has reduced resolution due to insufficient high-frequency sampling and produces artifacts.

3 Dataset Partition, Denoising Network Construction and Evaluation

Note: You may need `train.py`, line 113, class `UNet`; line 345, class `ResNet` and line 497, function `train()` for reference in this section. (Some related functions are also needed)

3.1 Dataset Partition

The full dataset contains 200 samples is split into training, validation, and testing sets with ratios 4:1:2, resulting in 114 training, 29 validation, and 57 testing samples.

3.2 CNN Network Designing and Training

Our reconstruction network consists of three components:

1. **Dual 2D UNet Architecture:** Two independent UNet modules process the real and imaginary parts of the complex MRI data separately. Each UNet features an encoder-decoder structure with skip connections. A key component is our implementation of dual attention mechanisms: channel attention that recalibrates feature maps by explicitly modeling interdependencies between channels, and spatial attention that focuses on informative regions by selectively emphasizing important spatial locations. These attention modules significantly enhance feature representation by suppressing irrelevant features and highlighting diagnostically important structures. The UNet architecture also incorporates dropout (p=0.3) for regularization and LeakyReLU activation for improved gradient flow during training.
2. **Concatenation:** The outputs of the two UNet branches are concatenated along the channel dimension, preserving the complementary spatial information extracted from both real and imaginary components.
3. **3D ResNet (Temporal Fusion):** While the UNet branches effectively extract spatial features, temporal correlation between consecutive frames remains unexploited. To address this, we employ a lightweight 3D ResNet that processes the concatenated features through 3D convolutions and residual connections. This temporal fusion module captures dynamic changes across the MRI sequence, leveraging both short-term and long-term dependencies between adjacent frames. By incorporating both spatial context (from UNets) and temporal dynamics (from ResNet), our model effectively suppresses time-varying artifacts and produces temporally coherent reconstructions. A final 1x1x1 convolution maps the spatio-temporal features to the output channels.

This design (shown in Figure 4) effectively utilizes both spatial information (2D UNet processing) and temporal information (capturing dynamic changes via 3D ResNet) for the reconstruction task.

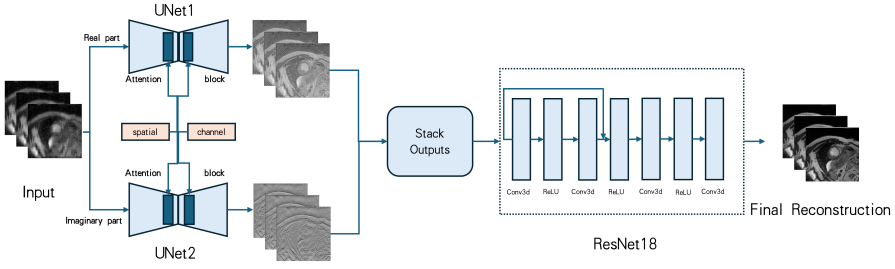


Figure 4: Overall architecture of our proposed reconstruction network with dual UNet branches for real and imaginary components and 3D ResNet for temporal fusion.

We first train a baseline network **without** dropout and dynamic learning rate schedule, using a constant learning rate. The parameters are: Adam optimizer, initial learning rate 10^{-4}, weight decay 10^{-4}, batch size 10, and $L2$ loss (MSE). Training and validation losses are logged per epoch.

All the training loss and validation loss are recorded in the file named `output/No_opt_output.txt`. The loss curves for 800 epochs are plotted in Figure 5.

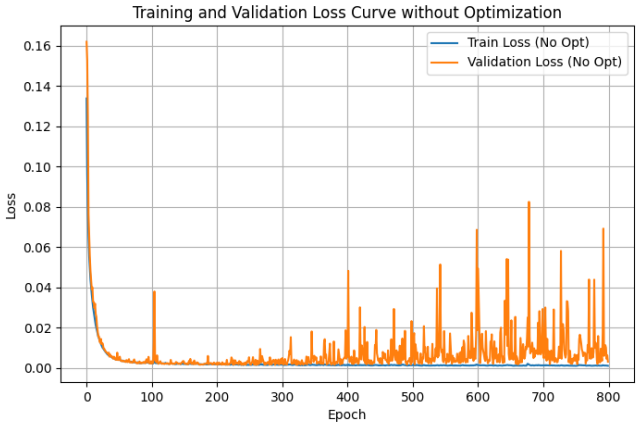
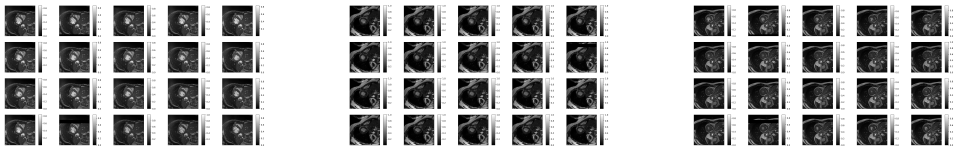


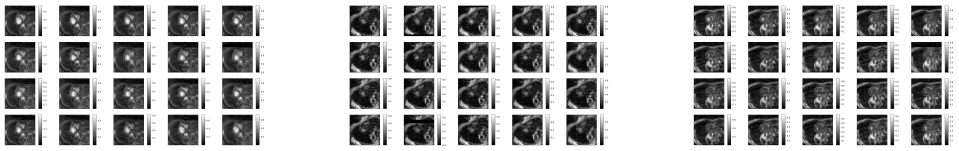
Figure 5: Training Loss and Validation Loss (No Dropout/Dynamic LR)

In Figures 6-8, we show example testing images before and after deep learning reconstruction using this baseline model. The complete results can be found in the folder `images/No_opt`.



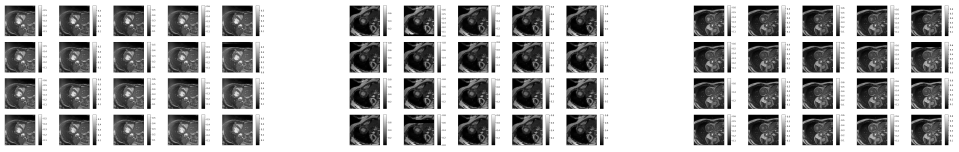
(a) Frame 0 (b) Frame 1 (c) Frame 2

Figure 6: Fully Sampled Images (Ground Truth)



(a) Frame 0 (b) Frame 1 (c) Frame 2

Figure 7: Under Sampled Images (Input to Network)



(a) Frame 0 (b) Frame 1 (c) Frame 2

Figure 8: Reconstruction Images (Baseline Model - No Dropout/Dynamic LR)

Comment on the denoising/dealiasing performance (Baseline Model):

1. Denoising: The network reduces some noise compared to the undersampled input, but residual noise and artifacts remain.

2. Dealiasing: The network attempts to reduce aliasing artifacts, but the reconstruction quality is limited, and some blurring and artifact patterns persist.

3.3 Quantitative Evaluation (Baseline Model)

After calculating the PSNR and SSIM for each testing slice before and after deep learning reconstruction using the baseline model (no dropout/dynamic LR), we get:

1. The mean of PSNR: 24.15409541, the standard deviation of PSNR: 1.85842901
2. The mean of SSIM: 0.74309104, the standard deviation of SSIM: 0.03733812

These metrics indicate moderate reconstruction quality. The validation loss curve (Figure 5) shows fluctuations and a potential increase towards the end, suggesting overfitting, which limits performance.

4 Impact of Dropout and Dynamic Learning Rate

Note: Detailed reconstruction images, related training loss, validation loss and their loss curves for the optimized model can be found in `assets/Training Loss and Validation Loss.png`, `output/output.txt` and `images/output`.
We now train our full model, incorporating dropout (p=0.3 in UNet) and a cosine annealing learning rate schedule with warm-up. The code can be found in `class UNet` and `function train()`. The training parameters are otherwise the same (Adam, initial LR 10^{-4}, weight decay 10^{-4}, batch size 10, L2 loss, 800 epochs).

By calculating the PSNR and SSIM again for this optimized model, we can compare it with the baseline (Table 1).

Table 1: Compare PSNR and SSIM with and without Dropout/Dynamic LR

Optimization?	PSNR mean	PSNR std dev	SSIM mean	SSIM std dev
Before Recon	23.463965	2.866855	0.572510233579521	0.088547643071673
No	24.15409541	1.85842901	0.74309104	0.03733812
Yes (Full Model)	29.08446121	1.93235576	0.84434632	0.03711063

Compared to the baseline, adding dropout and a dynamic learning rate significantly improves both PSNR and SSIM.

- The mean PSNR increases from 24.15 to 29.08, indicating a substantial reduction in noise and error, making the reconstructed image much closer to the ground truth.
- The mean SSIM increases from 0.743 to 0.844, indicating a significant improvement in structural similarity. The model better captures the real structure of the image.
- The standard deviations remain relatively low and similar, suggesting consistent performance across the test set for both models, but the optimized model achieves this consistency at a much higher quality level.

These enhancements are due to:

1. Dropout acts as a regularizer, reducing overfitting by preventing the network from relying too heavily on specific neurons or features in the training data. This improves generalization to unseen data and makes the model more robust to undersampling artifacts.
2. The dynamic learning rate schedule (cosine annealing with warm-up) helps the training process. The warm-up phase stabilizes initial training, the cosine decay allows for faster convergence initially and then fine-tuning with smaller steps later, helping the model find a better minimum and improve detail recovery.

The loss curves for the optimized model (Figure 9) show much more stable convergence compared to the baseline (Figure 5).

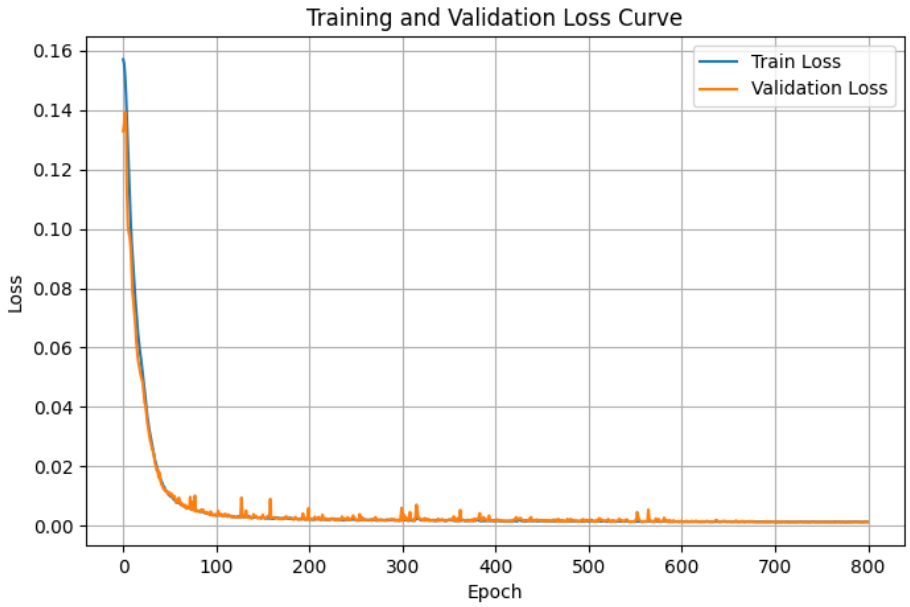


Figure 9: Training Loss and Validation Loss (Optimized Model with Dropout/Dynamic LR, L2 Loss)

5 Compare L1 with L2 Loss

Note: Detailed reconstruction images, related training loss, validation loss and their loss curves can be found in `assets/Training Loss and Validation Loss L1.png`, `output/L1_Loss_output.txt` and `images/L1`.
We replace the L2 loss (MSE) with L1 loss (MAE) in the training pipeline, retraining using the same optimized architecture (dropout, dynamic LR) and training parameters. We then compare the new PSNR and SSIM with the results from the optimized L2 model (Table 2).

Table 2: Compare PSNR and SSIM with L1 vs L2 Loss (Optimized Model)

Loss	PSNR mean	PSNR std dev	SSIM mean	SSIM std dev
Before Recon	23.463965	2.866855	0.572510233579521	0.088547643071673
L1	29.15108941	2.24063664	0.84389277	0.04208510
L2	29.08446121	1.93235576	0.84434632	0.03711063

We can see that L1 loss yields slightly higher mean PSNR, while L2 loss yields slightly higher mean SSIM and lower standard deviations for both metrics in the testing dataset.

This phenomenon can be explained as follows:

- PSNR (Peak Signal-to-Noise Ratio) measures pixel-wise accuracy. L1 loss, by minimizing the mean absolute error, effectively reduces pixel-level discrepancies, thus improving PSNR.
- SSIM (Structural Similarity Index) evaluates structural similarity, focusing on local patterns of luminance, contrast, and structure. L1 loss does not explicitly enforce structural consistency. Small spatial misalignments or distortions, which might minimally impact PSNR, can significantly lower SSIM.

In short, L1 loss favors pixel-wise precision but may compromise local structural integrity. A potential solution could be a composite loss function combining L1 with a structure-aware loss like SSIM loss or a perceptual loss.

The training curve for L1 loss (Figure 10) appears stable. While L1 loss can sometimes promote sparsity, L2 loss often leads to smoother results and is more sensitive to large errors. In this case, both loss functions yield comparable high-quality reconstructions. However, the original configuration with L2 loss achieved slightly better stability (lower standard deviation in metrics) and lower final validation loss values.

Ultimately, the choice of loss function should be guided by the final objective:

- If precise pixel recovery is the goal, prioritizing PSNR with L1 or L2 loss may suffice.
- If maintaining perceptual quality and structural fidelity is crucial (e.g., in clinical imaging), incorporating structure-aware losses is strongly recommended.

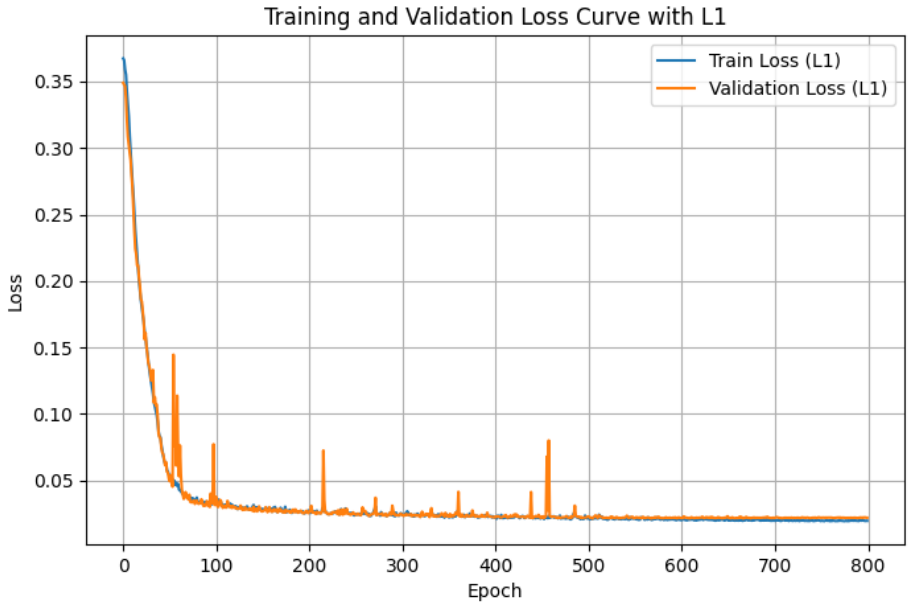


Figure 10: Training Loss and Validation Loss (Optimized Model with L1 Loss)

5.1 Application of Perceptual Loss and Edge Loss in MRI Reconstruction

In MRI reconstruction tasks, especially under highly undersampled conditions, achieving both high numerical accuracy and perceptually pleasing images is crucial. Traditional pixel-wise losses such as Mean Squared Error (MSE) tend to produce blurry reconstructions, as they overly penalize small pixel mismatches without considering the structural or perceptual quality of the image.

Perceptual loss addresses this limitation by measuring the difference between the reconstructed and ground truth images in the feature space of a pre-trained convolutional neural network. This encourages the model to preserve high-level semantic features and fine textures.

Edge loss, on the other hand, focuses explicitly on preserving sharp structural details. By applying an edge detection operator (such as Sobel filters) to both the reconstructed and ground truth images and minimizing the difference between their edge maps, the network is guided to accurately recover critical high-frequency components, such as organ boundaries and anatomical structures, which are vital in medical imaging.

Combining perceptual loss and edge loss with traditional pixel-wise losses leads to reconstructions that are not only numerically superior but also visually much sharper and structurally more faithful to the original anatomy.

Due to the restrictions of the current project, we did not implement perceptual loss and edge loss in our training pipeline. We recommend exploring these advanced loss functions in future work

to further enhance the quality of MRI reconstructions, especially in challenging undersampling scenarios.

6 Unrolled Deep Learning Reconstruction Network

We explored an unrolled network architecture incorporating data consistency (DC) layers between cascaded instances of our base reconstruction network (Dual UNet + 3D ResNet). The DC layer enforces consistency with the acquired k-space data after each denoising step using the formula: $k_{out} = U \odot k_{undersampled} + (1 - U) \odot F(x_{denoised})$, where $k_{undersampled}$ is the initially acquired undersampled k-space data, $x_{denoised}$ is the output of the denoising network, F is the Fourier transform, and U is the undersampling mask. The output image is then $x_{out} = F^{-1}(k_{out})$.

We trained models with 2 cascades (Cascade 2) and 3 cascades (Cascade 3) of the base network + DC layer. Training used the optimized setup (dropout, dynamic LR, L2 loss).

6.1 Training Details and Results

- **Cascade 2:** Required 18GB GPU memory, trained for 300 epochs. Average time per epoch: 12 seconds.
- **Cascade 3:** Required 24GB GPU memory, trained for 300 epochs. Average time per epoch: 360 seconds (6 minutes).

The performance metrics are compared with the original single network in Table 3. The loss curves for Cascade 3 are shown in Figure 11.

Table 3: Comparison of Original and Unrolled Networks

Model	Epochs	Avg Epoch Time	GPU Mem	Loss	PSNR	SSIM
Original	800	$\sim$ 6 sec	$\sim$ 10GB	0.00135	29.084	0.844
Cascade 2	300	$\sim$ 12 sec	18GB	0.00143	28.866	0.834
Cascade 3	300	$\sim$ 360 sec	24GB	0.00137	28.958	0.807

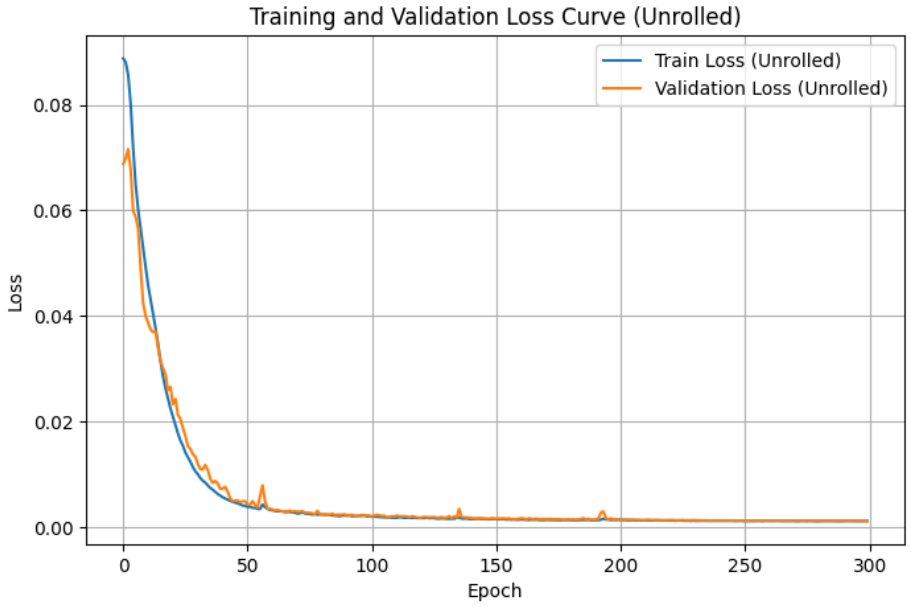


Figure 11: Training and Validation Loss Curves for the 3-Cascade Unrolled Network

6.2 Discussion

- **Resource Intensity:** Increasing cascades significantly increased GPU memory usage and training time per epoch.
- **Performance:** Despite the added complexity and data consistency steps, the cascaded models (trained for 300 epochs) did not outperform the original single network (trained for 800 epochs). Cascade 3 showed slightly better PSNR than Cascade 2 but worse SSIM than both others. The validation loss for Cascade 3 also appears less stable.
- **Potential Reasons for Limited Improvement:**
    1. **Insufficient Training Data:** The dataset size (200 samples) might be too small for these deeper unrolled networks. More data or augmentation might be needed.
    2. **Base Network Complexity/Convergence:** The base network is already quite large. It might not have fully converged even after 800 epochs in the original setup, potentially hindering effective iterative refinement in the unrolled setting.
    3. **Training Constraints:** The significantly increased training time and memory limited the unrolled models to 300 epochs. Longer training might be required for convergence.

Further investigation with more data, potentially a smaller base network, and extended training times would be necessary to fully assess the potential of this unrolled architecture for our specific problem.