Project 1: Deep Learning Dynamic MRI Reconstruction

熊闻野 吴家兴 杨人一 夏博扬 杨丰敏

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 and a 3D ResNet to explore the temporal correlation. We generate variable density undersampling patterns with acceleration factor 5 and central k-space sampling, 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

1 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.

1.1 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 generate 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.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

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.

1.2 Undersampling Masks Plotting

Figure 1 below shows the undersampling mask for one dynamic frame and undersampling masks in the ky-t dimension. You can find the figure in `undersampling_mask.png`.

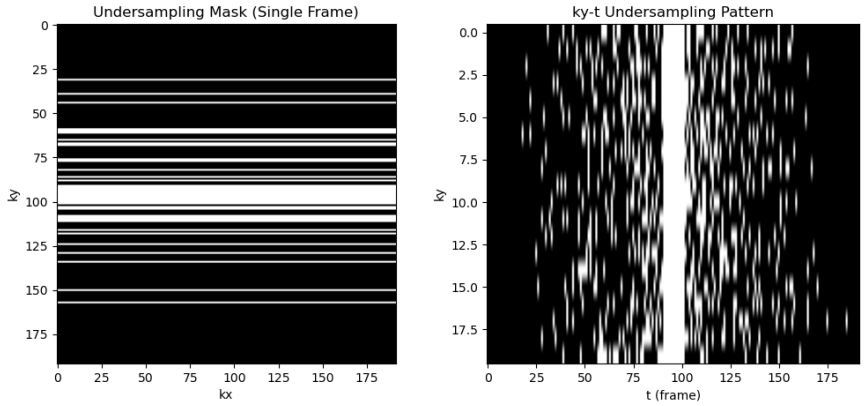


图 1: Undersampling Mask

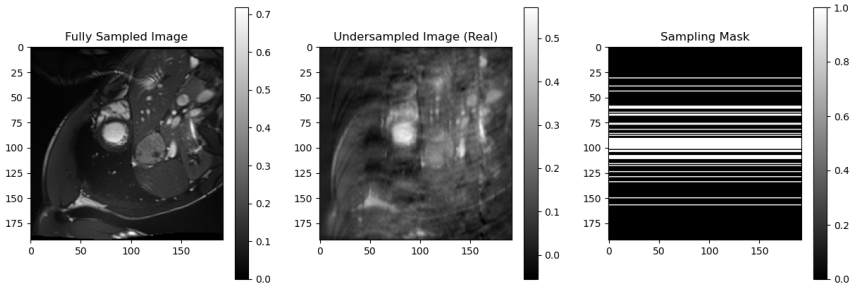
Thus we can see that the sampling mask is not the same for different dynamic frames. That's because sampling positions are randomized per frame to disrupt coherent aliasing artifacts in time.

1.3 Aliased Images Depiction and Comparison

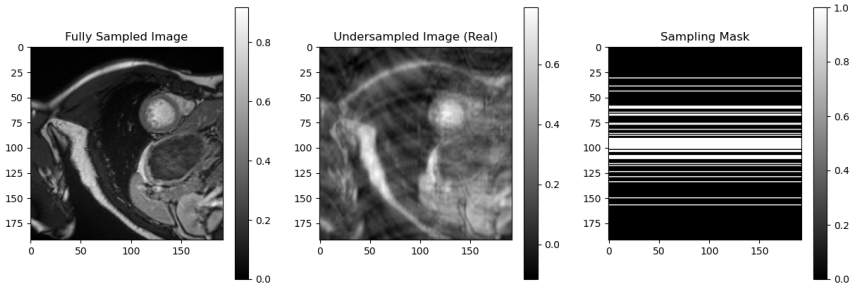
From the stem of this question we can know that, given a fully sampled image m, the aliased image b is computed as $F^{-1}UFm$, where F is the Fourier transform.

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.

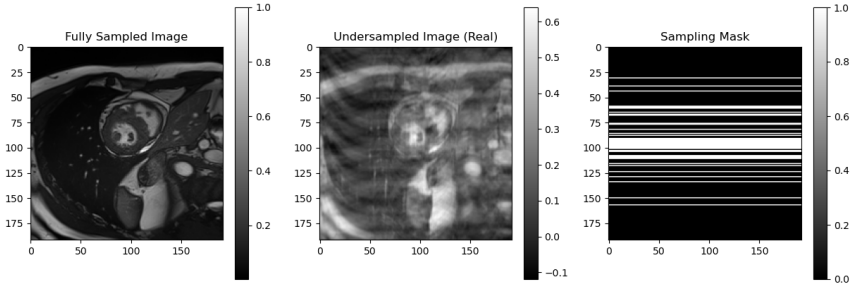
Figure 2 below compares the fully sampled image, aliased image, and sampling mask in Fourier domain. Here we choose the first 3 images generated to better compare the aliased images with the fully sampled image.



(a) comparison image 0



(b) comparison image 1



(c) comparison image 2

图 2: compare the fully sampled images with aliased images

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.

2 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)

2.1 Dataset Partition

We know that the full dataset containing 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. This step is done by the code below:

```
dataset = TensorDataset(inputs, labels)
train_size, val_size, test_size = 114, 29, 57
train_set, val_set, test_set = torch.utils.data.random_split(
    dataset, [train_size, val_size, test_size])
```

Listing 1: Dataset Partition

2.2 CNN Network Designing and Training

Our denoising network consists of two independent 2D UNet modules processing separate input channels, whose outputs are stacked and fed into a 3D ResNet. This design can effectively utilize both spatial information (2D UNet processing) and temporal information (capturing dynamic changes via 3D ResNet) of the image to better achieve the denoising task.

We train our network with Adam optimizer, initial learning rate 10^{-4}, weight decay 10^{-4}, batch size 10, and $L2$ loss (MSE). Note that training and validation losses are logged per epoch.

All the training loss and validation loss are recorded in the file named `output.txt`. And the loss curves for 800 epochs are plotted in figure 3 below.

In the figure 4, figure 5, and figure 6 below, there are some example testing images before and after deep learning reconstruction. Here we only show half of the output we obtained, the complete outcome can be seen in the folder named /textttimages.

Comment on the denoising/dealiasing performance:

1. Denoising: The network significantly reduces random noise in the input image, smoothing the image while protecting important structural features such as edges and details.

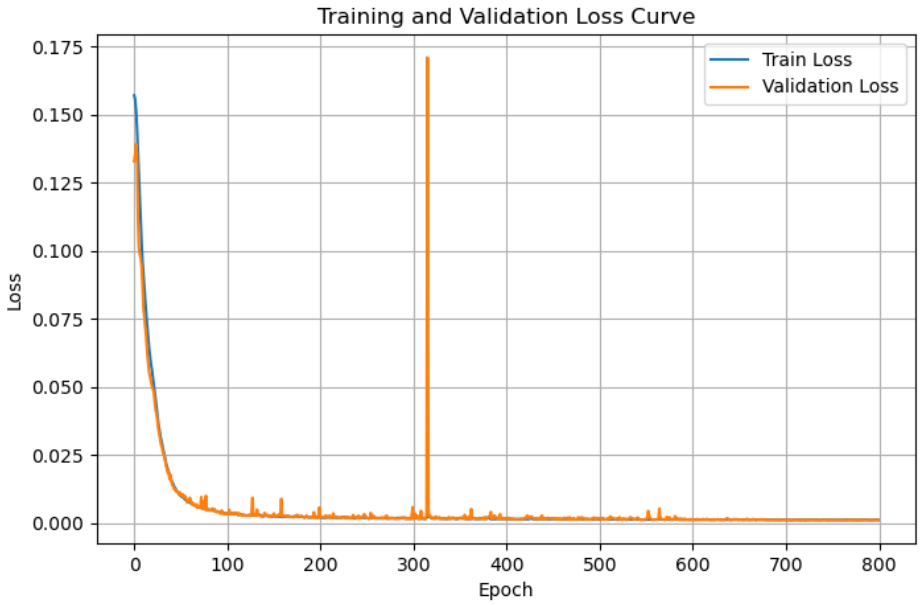


图 3: Training Loss and Validation Loss

2. Dealiasing: Our network reduces aliasing artefacts and enhances the realism of image textures and structures, avoiding unnatural distortions such as “streaks” and “swirls”.

2.3 Quantitative Evaluation

We can calculate the PSNR and SSIM for each testing slice before and after deep learning reconstruction.

3 Impact of Dropout and Dynamic Learning Rate

We add drop out to our network training, and the code can be found in `class UNet`.

```
self.dropout = nn.Dropout(p=0.3)
```

Listing 2: Add Dropout

And a cosine annealing learning rate schedule with warm-up is employed, which can be found in `function train()`.

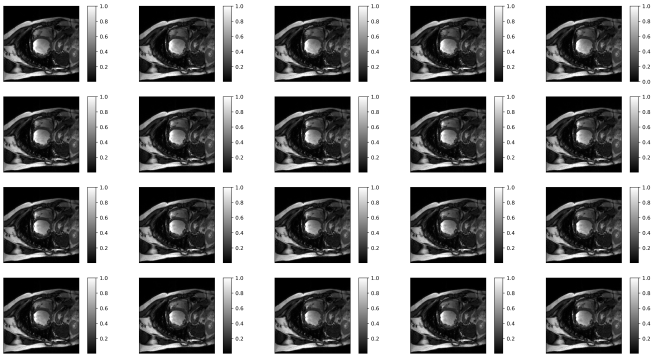
```
lr = lr_scheduler(epoch, warmup_epochs, warmup_lr, initial_lr,
    num_epochs)
for param_group in optimizer.param_groups:
  param_group['lr'] = lr
```

Listing 3: Add Dynamic Learning Rate

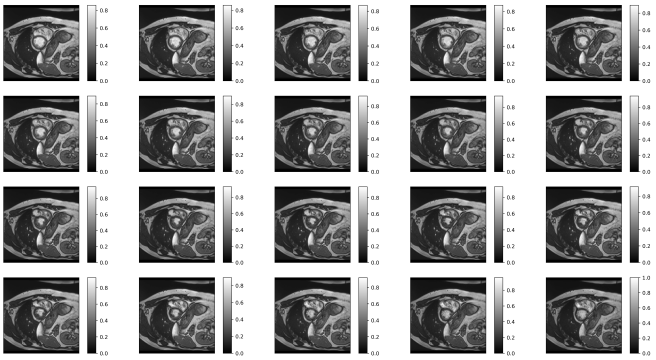
4 Compare L1 with L2

We replace the L2 loss with L1 loss in the training pipeline, retrain using the same architecture and training parameters.

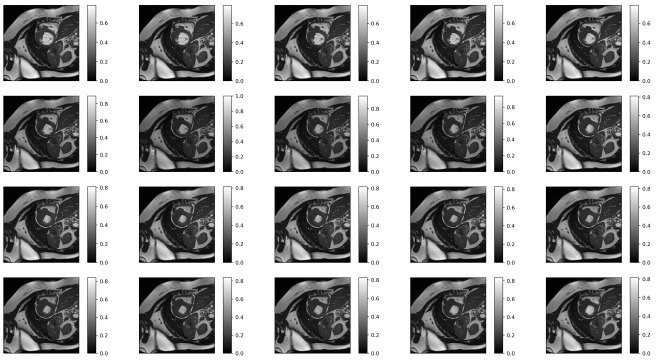
5 Unrolled Deep Learning Reconstruction Network



(a) fully sampled image 0

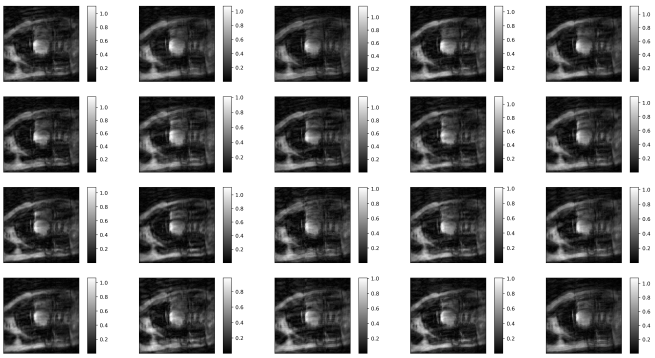


(b) fully sampled image 1

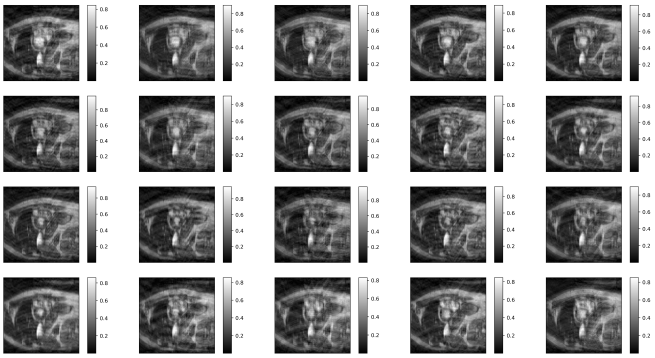


(c) fully sampled image 2

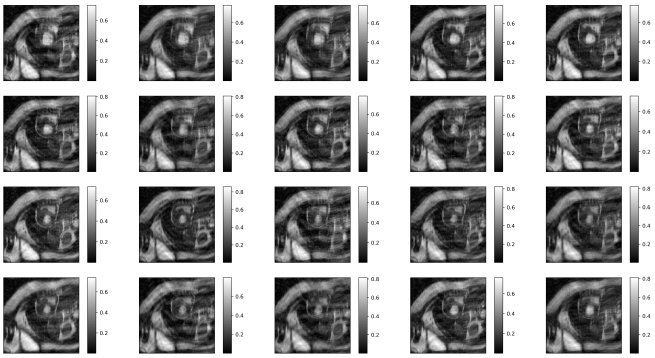
图 4: Fully Sampled Images (Ground Truth)



(a) under sampled image 0

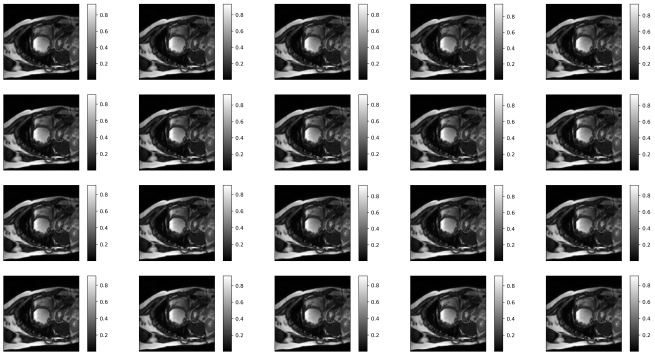


(b) under sampled image 1

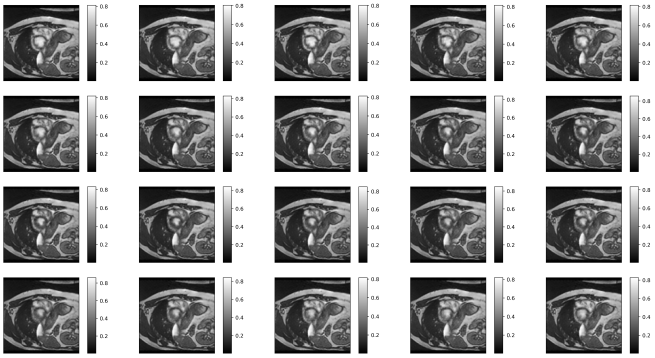


(c) under sampled image 2

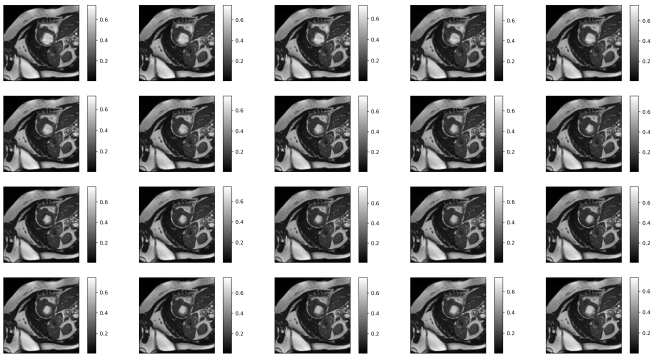
图 5: Under Sampled Images



(a) reconstruction image 0



(b) reconstruction image 1



(c) reconstruction image 2

图 6: Reconstruction Images