Fast MRI Reconstruction

From Compressed Sensing to Deep Learning

Motivation

MRI normally takes 15-90 min.

What if the patient could not bear such a long time in the noisy machine? Or what if the condition requires fast diagnosis?

→ Shortening the time of MRI scanning and reconstruction

Motivation

Possible solution:

- Subsampling k-space data and then estimate the fully-sampled MRI image without reducing the quality.
- Current methods: GRAPPA/CS - noise amplification
- Deep Learning (DL): recognizing patterns and detecting features
- Applying DL to MRI reconstruction

Theory: Compressed Sensing (CS)

Optimization problem:

$$\min_{u} \mathcal{R}(\boldsymbol{u}) + \frac{\lambda}{2} \|A\boldsymbol{u} - \boldsymbol{f}\|_2^2$$

$$A = M \circ \mathcal{F} \circ S$$

Solved by iterative methods such as Gradient Descent

Weakness: low quality of output / reconstruction takes a relatively long time

(Knoll, Hammernik, Zhang, Moeller, Pock, Sodickson, & Akcakaya. 2018)

Theory: Deep Learning (DL General Info)

End-to-End Variational Network

Same optimization problem as in CS $\min_{u} \mathcal{R}(\boldsymbol{u}) + \frac{\lambda}{2} \|A\boldsymbol{u} - \boldsymbol{f}\|_2^2$

Training parameters in Convolutional Neural Network

Unroll iterations (gradient descent) by layer:

$$\boldsymbol{u}^{t+1} = \boldsymbol{u}^t - \underbrace{\sum_{i=1}^{N_k} (K_i^t)^\top \Phi_i^{t\prime}(K_i^t \boldsymbol{u}^t)}_{\text{CNN}} - \lambda^t A^*(A\boldsymbol{u}^t - \boldsymbol{f})$$

(Knoll, Hammernik, Zhang, Moeller, Pock, Sodickson, & Akcakaya. 2018)

Theory: Deep Learning Architecture

(Johnson *et al.* 2020)

Input: raw k-space data
Output: reconstructed image
In one epoch:

1. Estimate sensitivity map using the same architecture
2. Run *VarNetBlock* (one cascade) for 12 times

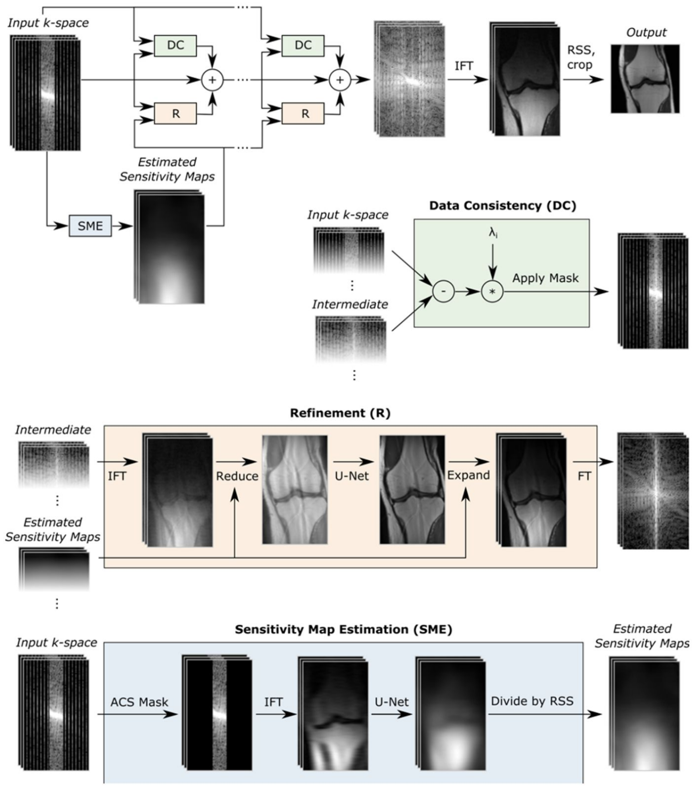
```
    self.sens_net = SensitivityModel(
        sens_chans, sens_pools, num_sense_lines=num_sense_lines
    )
    self.cascades = nn.ModuleList(
        [VarNetBlock(NormUnet(chans, pools)) for _ in range(num_cascades)]
    )

def forward(self, masked_kspace: torch.Tensor, mask: torch.Tensor) -> torch.Tensor:
1.  sens_maps = self.sens_net(masked_kspace, mask)
    kspace_pred = masked_kspace.clone()

    for cascade in self.cascades:
2.      kspace_pred = cascade(kspace_pred, masked_kspace, mask, sens_maps)

    return fastmri.rss(fastmri.complex_abs(fastmri.ifft2c(kspace_pred)), dim=1)
```

(https://github.com/facebookresearch/fastMRI)



Theory: Deep Learning Architecture (Cont'd)

$$k^{t+1} = k^t - \underline{\eta^t M(k^t - \tilde{k})} + \mathcal{F} \circ \varepsilon \circ \mathrm{CNN}\,(\mathcal{R} \circ \mathcal{F}^{-1}(k^t)\,)$$

For each cascade:

```
def forward(
    self,
    current_kspace: torch.Tensor,
    ref_kspace: torch.Tensor,
    mask: torch.Tensor,
    sens_maps: torch.Tensor,
) -> torch.Tensor:
    zero = torch.zeros(1, 1, 1, 1, 1).to(current_kspace)
    soft_dc = torch.where(mask, current_kspace - ref_kspace, zero) * self.dc_weight
    model_term = self.sens_expand(
        self.model(self.sens_reduce(current_kspace, sens_maps)), sens_maps
    )

    return current_kspace - soft_dc - model_term
```

Normalized U-net

(https://github.com/facebookresearch/fastMRI)

Dataset

fastMRI (http://fastmri.med.nyu.edu/): a public open MRI dataset

Multicoil brain validation dataset is used in this project.

Format: Brain / Multicoil / 2D /Field strength: 1.5 or 3 T

Type: T1 / T1 PRE / T1 POST / T2 / FLAIR

Number of coils: 16 or 20 (probably not accurate; randomly checked)

Factors & Variables

Training:

#epochs

training dataset

Training with (5 types or only FLAIR, 40 volumes in total)

Training mask type (Equispaced/Random)

Training acceleration rate (4/8/4+8)

Reconstruction:

Reconstruction mask type (Equispaced/Random)

Reconstruction acceleration rate (4/8/4+8)

Random Subsampling

```
num_cols = shape[-2]
center_fraction, acceleration = self.choose_acceleration()

# create the mask
num_low_freqs = int(round(num_cols * center_fraction))
prob = (num_cols / acceleration - num_low_freqs) / (
    num_cols - num_low_freqs
)
mask = self.rng.uniform(size=num_cols) < prob
pad = (num_cols - num_low_freqs + 1) // 2
mask[pad : pad + num_low_freqs] = True

# reshape the mask
mask_shape = [1 for _ in shape]
mask_shape[-2] = num_cols
mask = torch.from_numpy(mask.reshape(*mask_shape).astype(np.float32))
```

(https://github.com/facebookresearch/fastMRI)

Equispaced Subsampling

```
# determine acceleration rate by adjusting for the number of low frequencies
adjusted_accel = (acceleration * (num_low_freqs - num_cols)) / (
    num_low_freqs * acceleration - num_cols
)
offset = self.rng.randint(0, round(adjusted_accel))

accel_samples = np.arange(offset, num_cols - 1, adjusted_accel)
accel_samples = np.around(accel_samples).astype(np.uint)
mask[accel_samples] = True
```

4 times acceleration: 8% central lines + 17% others
8 times acceleration: 4% central lines + 8.5% others

(https://github.com/facebookresearch/fastMRI)

Evaluation Metrics

Peak Signal-to-Noise Ratio:

$$\mathrm{PSNR}(x,y) = 10\log_{10}\frac{\max(y)^2}{\mathrm{MSE}(x,y)}$$

$$\text{where} \qquad \mathrm{MSE} = \tfrac{1}{n}\left\|x-y\right\|_2^2$$

x is the reconstructed image and y is the ground truth (target)

(Zbontar, Knoll, Sriram, Murrell, Huang, Muckley, Defazio, Stern, Johnson, Bruno. 2018)

Structural Similarity:

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

$$\text{where} \qquad \mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\sigma_x = \left(\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2\right)^{\frac{1}{2}}$$

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y)$$

(Wang, Bovik, Sheikh, Simoncelli. 2004)

Test 1: CS vs. DL

Aim: compare the reconstruction quality

Setting:

CS: BartToolBox

DL: Training:

#epochs =100

Training dataset: 8*5 types = 40 volumes

Training mask type: same as the recon mask type (e-e / r-r)

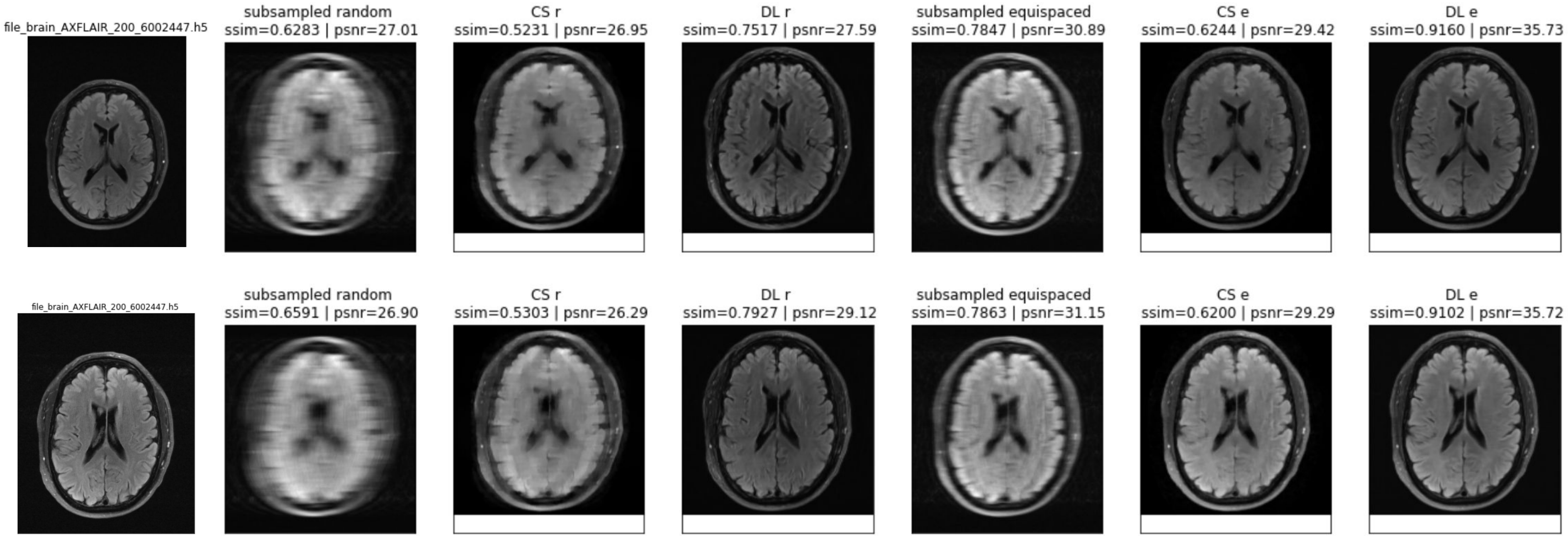
Training acceleration rate: 4+8 mixture

Recon:

"file_brain_AXFLAIR_200_6002447.h5"

Recon mask type: same as the training mask type (e-e / r-r)

Recon acceleration rate: 4



Ground truth

CS r	SSIM	PSNR
Avg	0.42	27.3
Std	0.13	0.66

DL r	SSIM	PSNR
Avg	0.83	30.2
Std	0.05	3.12

CS e	SSIM	PSNR
Avg	0.48	28.8
Std	0.15	0.68

DL e	SSIM	PSNR
Avg	0.93	38.1
Std	0.01	1.67

Test 1: CS vs. DL

Conclusion:

The performance of DL is better than performance of CS.

The gap between CS and DL is smaller in random reconstruction.

Future direction:

Test random scattered subsampling (here it is random line subsampling).

Test 2: Mask Type

Aim: understand the influence of mask type in the training process

Setting:

Training:

#epochs =100

Training dataset = 8*5 types= 40 volumes

Training mask type: equispaced or random

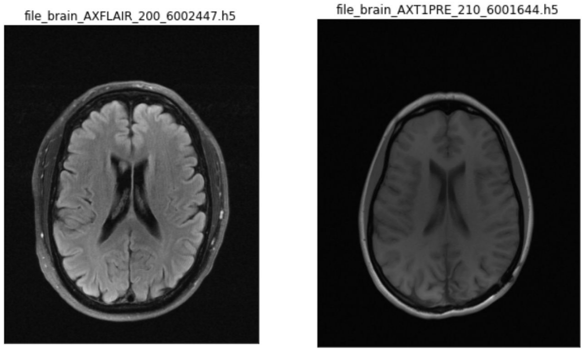
Training acceleration rate = 4+8

Recon:

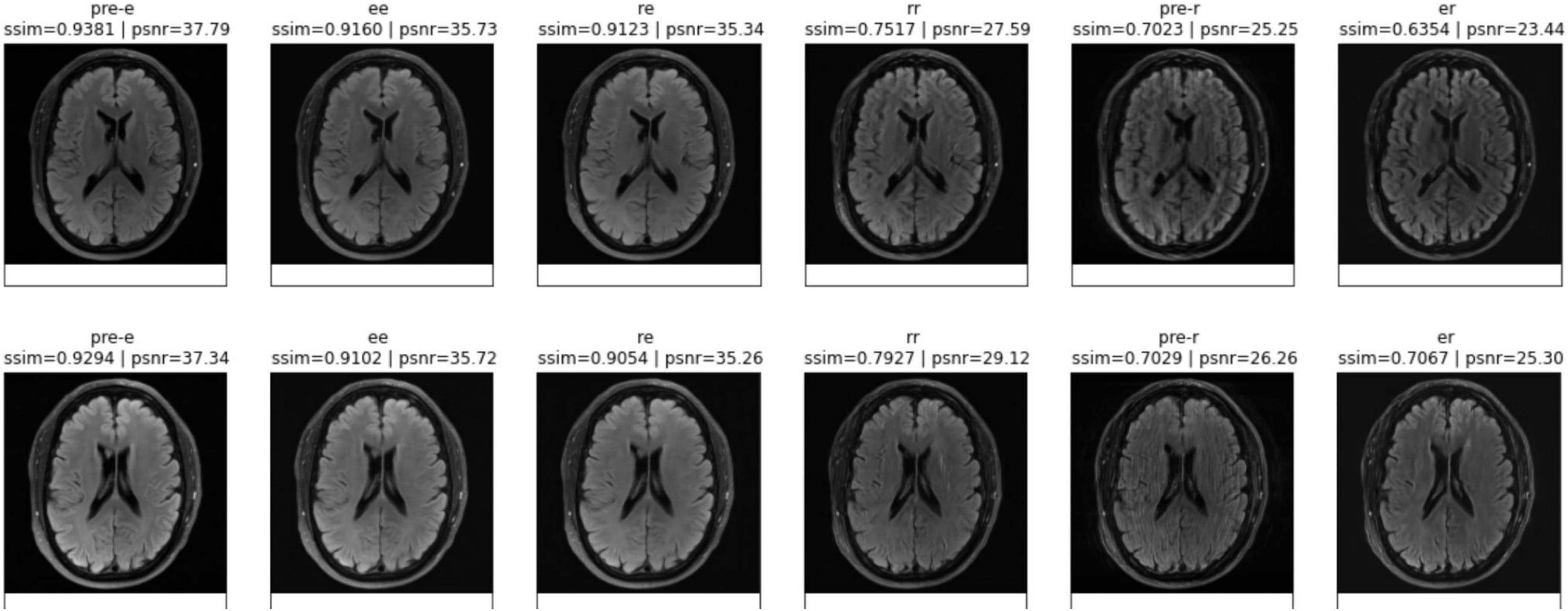
“file_brain_AXFLAIR_200_6002447.h5” & “file_brain_AXT1PRE_210_6001644.h5”

Recon mask type: equispaced or random

Recon Acceleration Rate: 4



Ground truth



pre e	SSIM	PSNR
Avg	0.94	39.5
Std	0.005	1.44

e e	SSIM	PSNR
Avg	0.932	38.1
Std	0.010	1.66

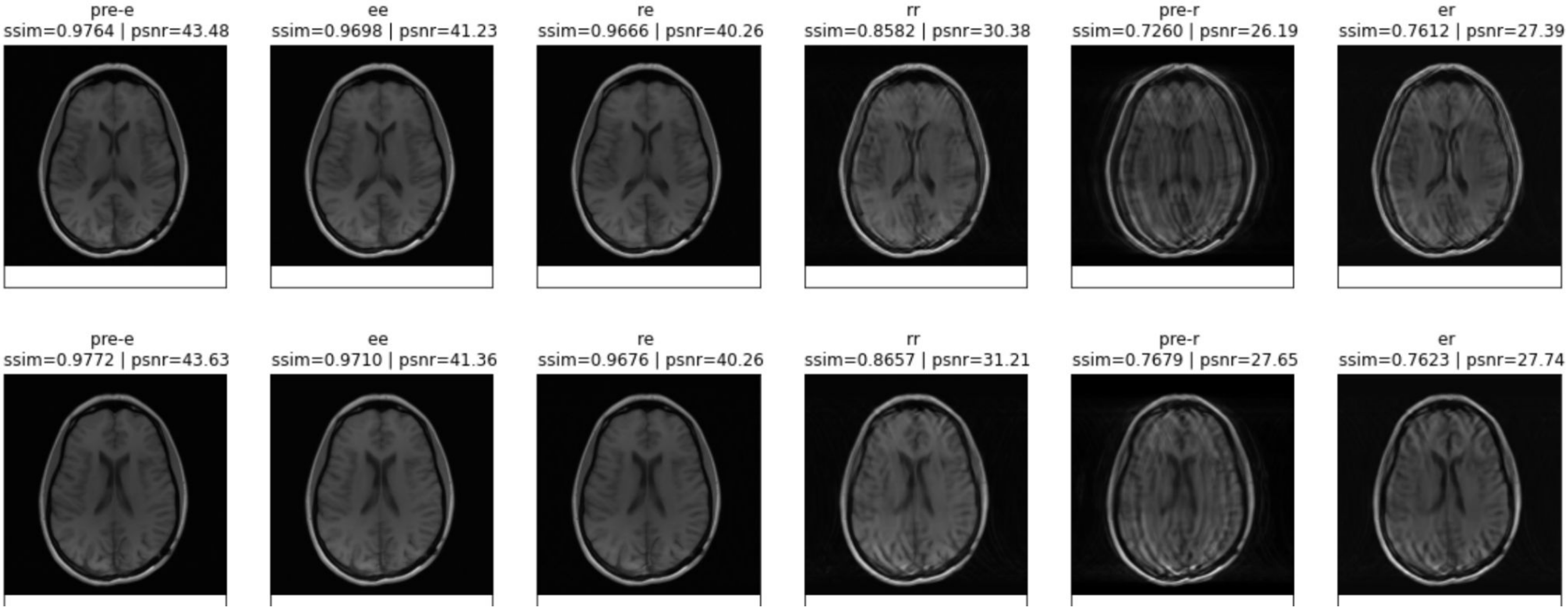
r e	SSIM	PSNR
Avg	0.929	37.6
Std	0.010	1.49

r r	SSIM	PSNR
Avg	0.83	30.2
Std	0.046	3.12

pre r	SSIM	PSNR
Avg	0.79	28.5
Std	0.075	4.50

e r	SSIM	PSNR
Avg	0.73	26.3
Std	0.090	4.80

file_brain_AXFLAIR_200_6002447.h5



pre e	SSIM	PSNR
Avg	0.98	44.4
Std	0.001	0.91

e e	SSIM	PSNR
Avg	0.973	42.4
Std	0.002	1.45

r e	SSIM	PSNR
Avg	0.971	41.6
Std	0.002	1.59

r r	SSIM	PSNR
Avg	0.88	32.5
Std	0.04	3.66

pre r	SSIM	PSNR
Avg	0.83	29.1
Std	0.08	5.47

e r	SSIM	PSNR
Avg	0.78	28.8
Std	0.09	5.06

file_brain_AXT1PRE_210_6001644.h5

Test 2: Mask Type

Conclusion:

Recon performance: pre-e > ee > re >> rr > pre-r ~ er

Possible reason: equispaced also contains some randomness and it’s easier to infer.

Future direction:

Further understanding of why *re* is better than *rr* .

Test 3: Training Acceleration Rate

Aim: understand the influence of acceleration rate in the training process

Setting:

Training:

#epochs =100

Training dataset = 8*5 types = 40 volumes

Training mask type: equispaced

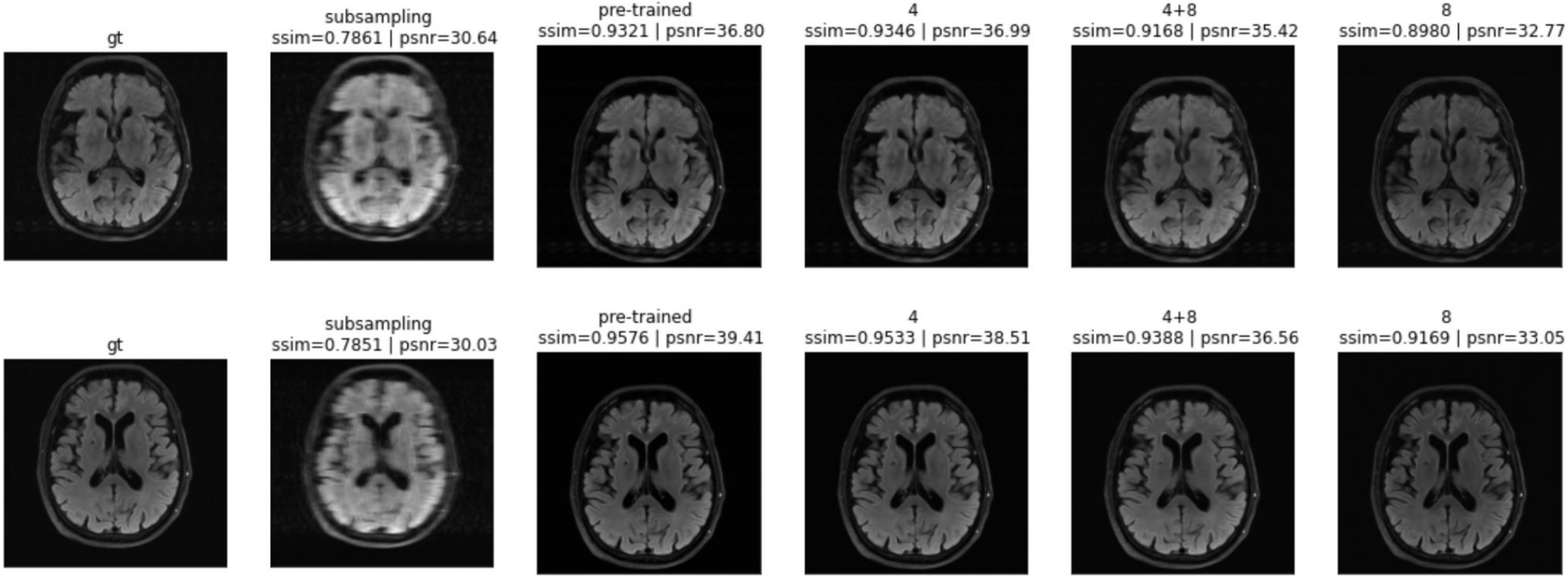
Training acceleration rate = 4+8 / only 4 / only 8

Recon:

“file_brain_AXFLAIR_201_6002868.h5”

Recon mask type: equispaced

Recon Acceleration Rate: 4 / 8

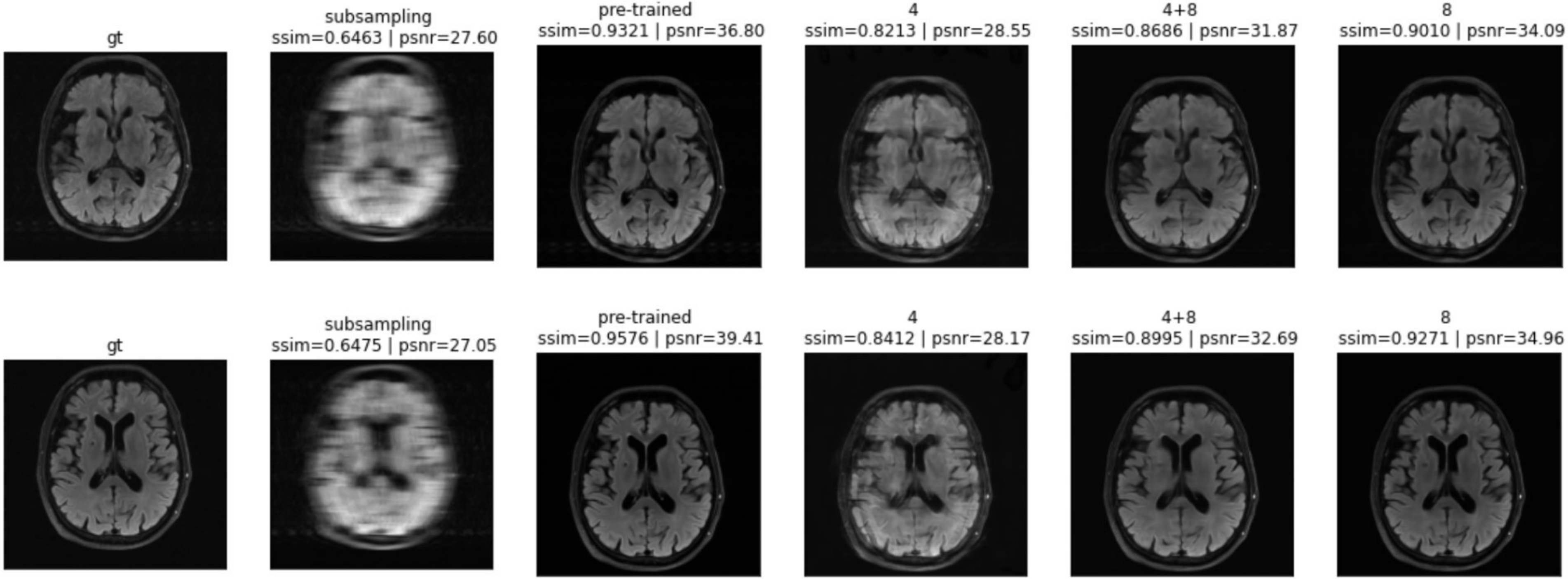


pre	SSIM	PSNR
Avg	0.95	40.6
Std	0.008	2.12

4	SSIM	PSNR
Avg	0.95	40.4
Std	0.006	2.08

4+8	SSIM	PSNR
Avg	0.94	39.1
Std	0.01	2.44

8	SSIM	PSNR
Avg	0.92	35.8
Std	0.01	2.04



pre	SSIM	PSNR	4	SSIM	PSNR	4+8	SSIM	PSNR	8	SSIM	PSNR
Avg	0.95	40.6	Avg	0.87	31.9	Avg	0.91	35.8	Avg	0.93	38.1
Std	0.008	2.12	Std	0.03	3.40	Std	0.02	3.51	Std	0.01	3.02

Test 3: Training Acceleration Rate

Conclusion:

Recon 4: pre-trained > 4 > 4+8 > 8

Recon 8: pre-trained > 8 ~ 4+8 > 4

Future direction:

Testing if we want to reconstruct 8 times acceleration, incorporating a certain amount of 4 times acceleration (or even 2 times acceleration) would help improve the results.

Test 4: Training Data Type

Aim: understand the influence of the type of training dataset

Setting:

Training:

#epochs =100

Training dataset = 8 * 5 types ***vs.*** 40 volumes only FLAIR

Training mask type: equispaced

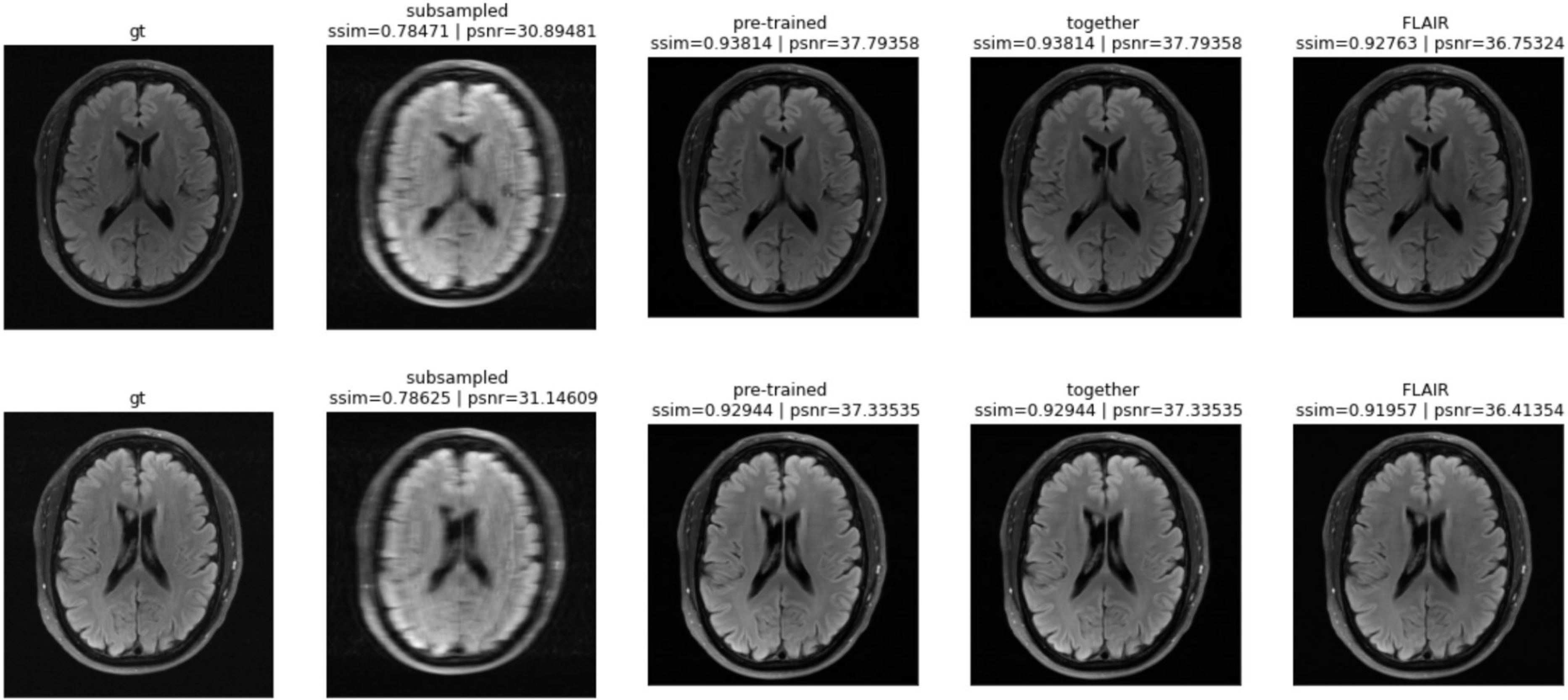
Training acceleration rate = 4+8

Recon:

"file_brain_AXFLAIR_200_6002447.h5" & "file_brain_AXT1POST_201_6002686.h5"

Recon mask type: equispaced

Recon Acceleration Rate: 4

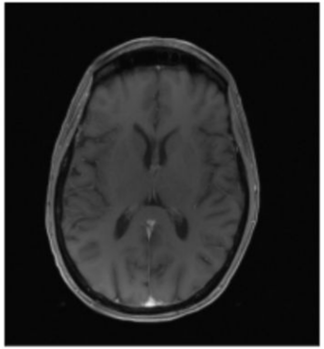


pre	SSIM	PSNR
Avg	0.94	39.5
Std	0.005	1.44

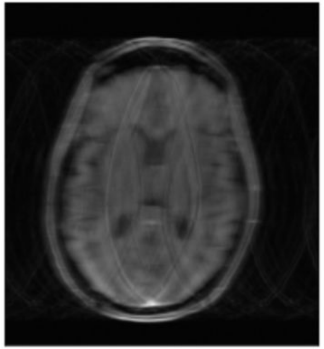
tog	SSIM	PSNR
Avg	0.93	38.1
Std	0.01	1.66

flair	SSIM	PSNR
Avg	0.94	38.8
Std	0.007	1.64

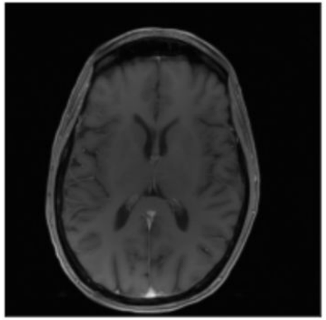
gt



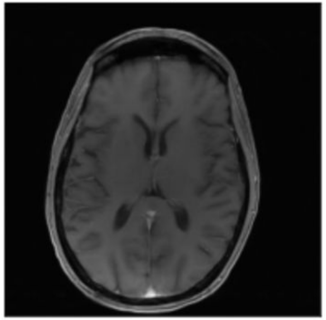
subsampled
ssim=0.78134 | psnr=28.73822



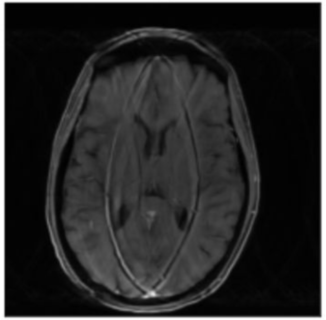
pre-trained
ssim=0.96607 | psnr=39.93233



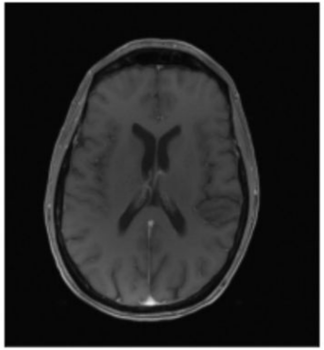
together
ssim=0.95132 | psnr=37.30182



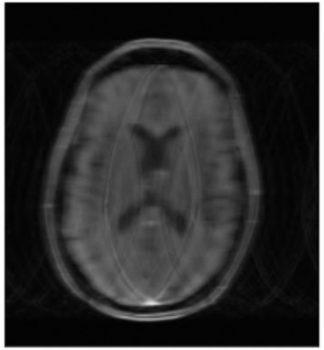
FLAIR
ssim=0.82455 | psnr=27.93074



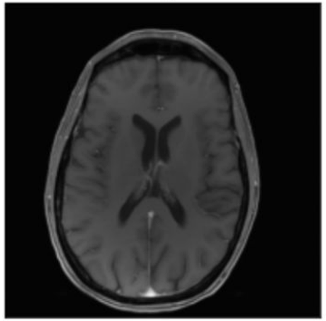
gt



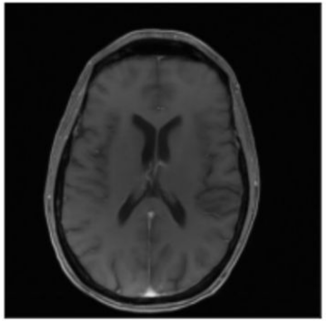
subsampled
ssim=0.78899 | psnr=29.16826



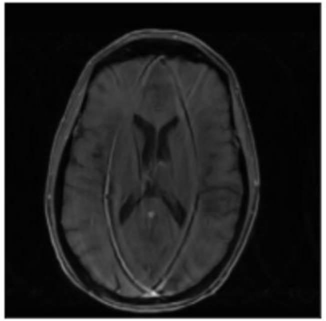
pre-trained
ssim=0.96862 | psnr=41.06179



together
ssim=0.95596 | psnr=38.33681



FLAIR
ssim=0.85062 | psnr=29.10776



pre	SSIM	PSNR
Avg	0.97	42.5
Std	0.002	1.52

tog	SSIM	PSNR
Avg	0.96	40.2
Std	0.005	2.16

flair	SSIM	PSNR
Avg	0.89	33.0
Std	0.043	4.47

Test 4: Training Data Type

Conclusion:

FLAIR only model to recon FLAIR performs almost the same as *together model*.

FLAIR only model to recon other types performs worse than *together model*.

Future direction:

Further verify this idea.

Test 5: Generalization

Aim: if the model is trained on knee dataset, would it perform as good as the model being trained on brain dataset when reconstructing brain images?

Setting:

Training (pre-trained model):

#epochs =100

Training dataset = 8*5 types = 40 volumes

Training mask type: equispaced ?

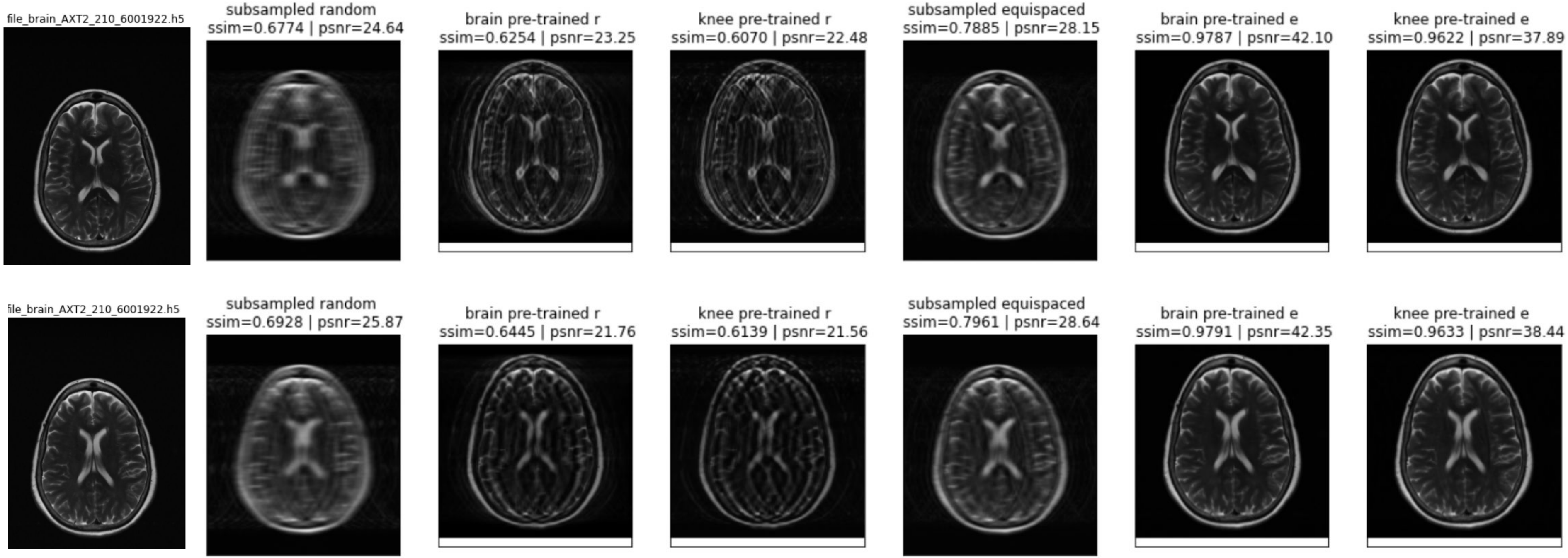
Training acceleration rate = 4+8 ?

Recon:

Brain data: “file_brain_AXT2_210_6001922.h5” & Knee data: “file1000107.h5”

Recon mask type: equispaced / random

Recon Acceleration Rate: 4



Ground truth

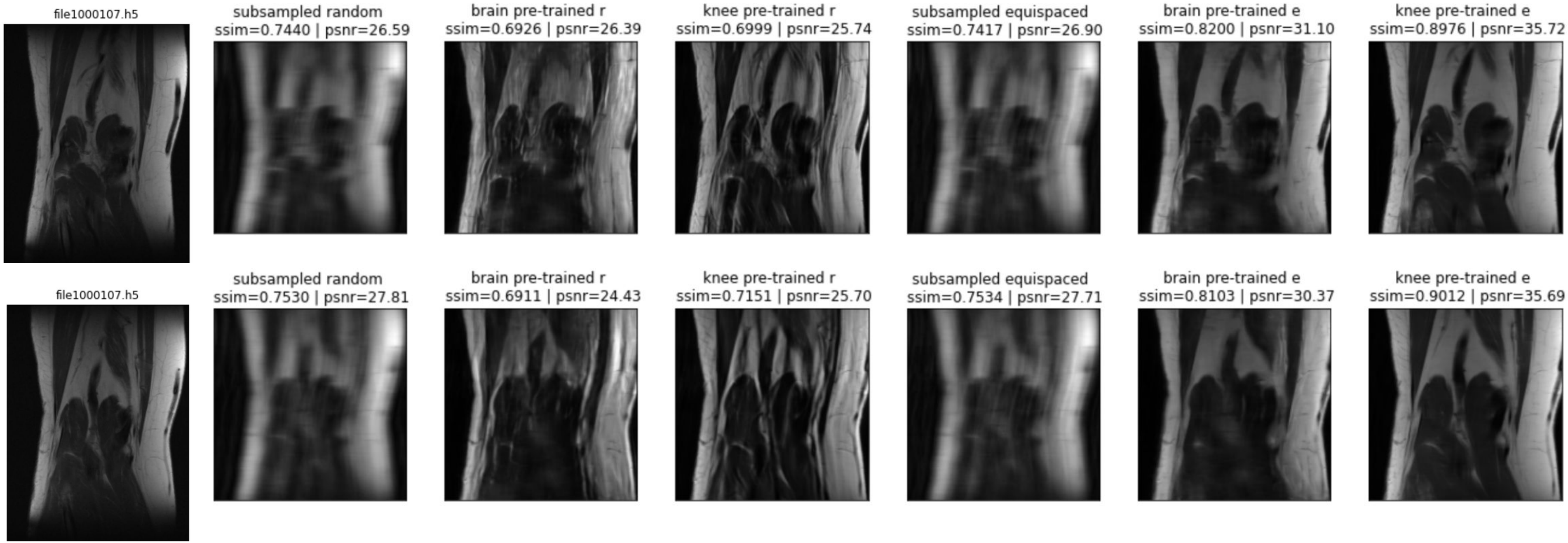
b r	SSIM	PSNR
Avg	0.77	27.13
Std	0.11	6.68

k r	SSIM	PSNR
Avg	0.75	26.2
Std	0.12	6.45

b e	SSIM	PSNR
Avg	0.98	43.8
Std	0.001	1.44

k e	SSIM	PSNR
Avg	0.97	40.6
Std	0.004	1.96

Brain data: file_brain_AXT2_210_6001922.h5



Ground truth

b r	SSIM	PSNR
Avg	0.74	26.65
Std	0.081	3.39

k r	SSIM	PSNR
Avg	0.75	27.1
Std	0.089	4.17

b e	SSIM	PSNR
Avg	0.86	33.2
Std	0.043	2.78

k e	SSIM	PSNR
Avg	0.91	37.0
Std	0.016	1.31

Knee data: file1000107.h5

Test 5: Generalization

Conclusion:

The generalization of the pre-trained model is not quite good.

Reconstructing one type, training on that type?

Potential Reason? This type of DL is still based on the math model.

Future direction:

Verify this assumption.

Test 6: Number of Training Datasets

Aim: whether increasing the number of training datasets would increase the performance of the model.

Setting:

Training:

#epochs =100

Training dataset = 8 * 5 types ***vs.*** 20 * 5 types

Training mask type: equispaced

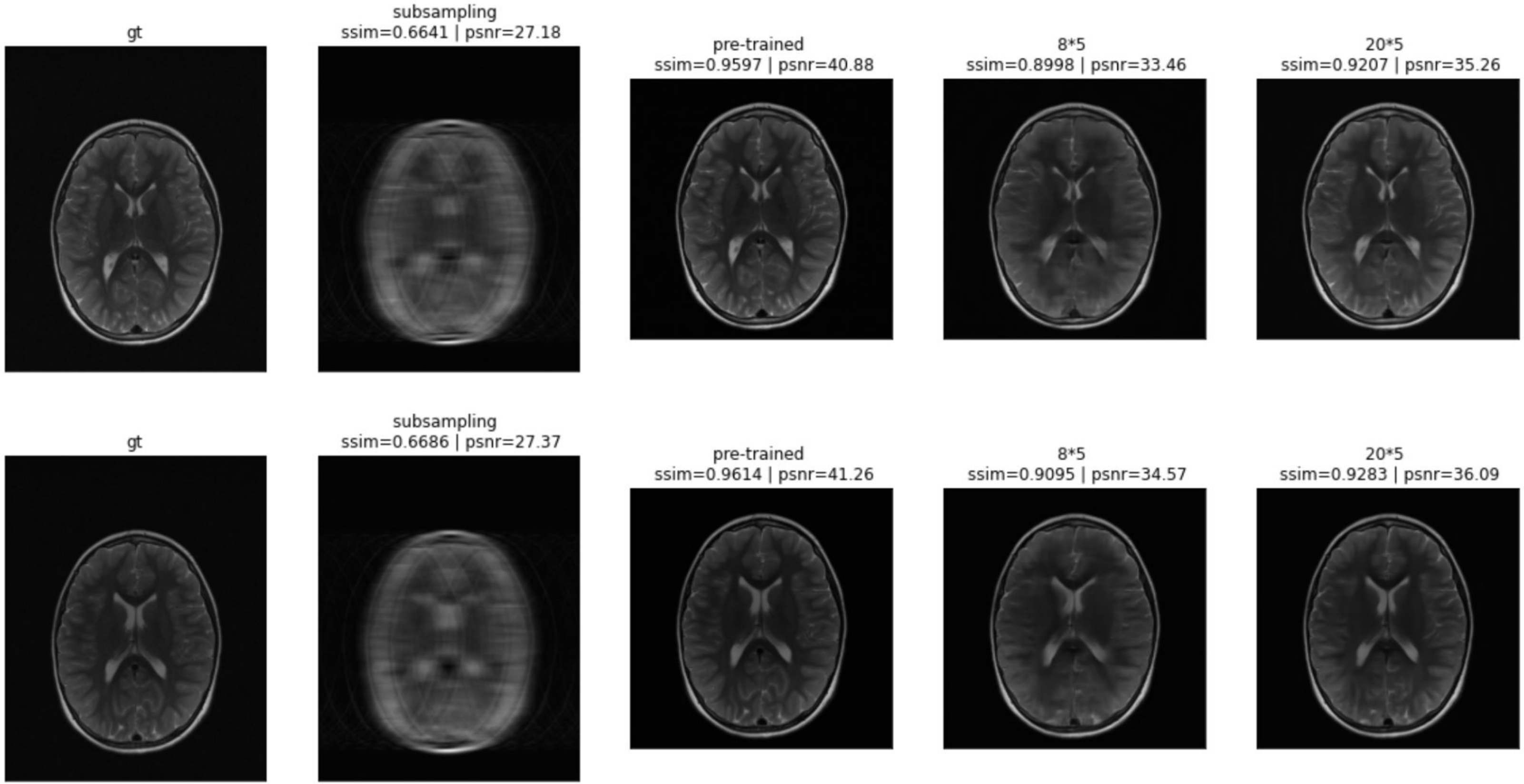
Training acceleration rate = 4+8

Recon:

"file_brain_AXT2_202_2020013.h5"

Recon mask type: equispaced

Recon Acceleration Rate: 8



pre	SSIM	PSNR
Avg	0.96	42.2
Std	0.002	0.84

8*5	SSIM	PSNR
Avg	0.92	36.1
Std	0.013	2.41

20*5	SSIM	PSNR
Avg	0.94	37.7
Std	0.009	2.43

Test 6: Number of Training Datasets

Conclusion:

Increasing number of training datasets could increase the performance of the model.

Future directions:

Find the bound of the relationship between # training datasets and model performance

Adjust other possible parameters to see the performance.

What have I done?

“Deep-Learning Methods for Parallel Magnetic Resonance Image Reconstruction” (2020):

“Is it sufficient to train a single model for all types of MR exams, or are separate models required for scans of different anatomical areas, pulse sequences, acquisition trajectories, and acceleration factors as well as scanner manufacturers, field strengths, and receive coils ? ”

What have I done?

For E2E-VN model, answers might be:

It's better to train the model on

1) all types of MR exams;
2) more training datasets;
3) a mixture of acceleration rates;
4) specific anatomical areas for reconstruction of that particular area;
5) equispaced subsampling

Future Work

- Find a suitable metric to evaluate the image quality
- Check the generalization of E2E-VN on different raw datasets
- Modify/Change the model structure such that it would generate more accurate results
- Interprete DL model to have a better understanding
- Would DL method affect the tumor in the reconstructed image?
- ....

Reference

[1] A. Chambolle and T. Pock. An introduction to continuous optimization for imaging. *Acta Numerica*, 25:161–319, 2016.

[2] K. Hammernik, T. Klatzer, E. Kobler, M. P. Recht, D. K. Sodickson, T. Pock, and F. Knoll. Learning a variational network for reconstruction of accelerated mri data. *Magnetic resonance in medicine*, 79(6):3055–3071, 2018.

[3] F. Knoll, K. Hammernik, C. Zhang, S. Moeller, T. Pock, D. K. Sodickson, and M. Akcakaya. Deep learning methods for parallel magnetic resonance image reconstruction. *arXiv preprint arXiv:1904.01112*, 2019.

[4] S. Roth and M. J. Black. Fields of experts: A framework for learning image priors. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 860–867. IEEE, 2005.

[5] J. Schlemper, J. Caballero, J. V. Hajnal, A. Price, and D. Rueckert. A deep cascade of convolutional neural networks for mr image reconstruction. In *International Conference on Information Processing in Medical Imaging*, pages 647–658. Springer, 2017.

[6] A. Sriram, J. Zbontar, T. Murrell, A. Defazio, C. L. Zitnick, N. Yakubova, F. Knoll, and P. Johnson. End-to-end variational networks for accelerated mri reconstruction. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 64–73. Springer, 2020.

[7] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[8] J. Zbontar, F. Knoll, A. Sriram, T. Murrell, Z. Huang, M. J. Muckley, A. Defazio, R. Stern, P. Johnson, M. Bruno, et al. fastmri: An open dataset and benchmarks for accelerated mri. *arXiv preprint arXiv:1811.08839*, 2018.