Multi-Task Accelerated MR Reconstruction Schemes for Jointly Training Multiple Contrasts

Anonymous Author(s)
Affiliation
Address
email

Abstract

Model-based accelerated MRI reconstruction methods leverage large datasets to reconstruct diagnostic-quality images from undersampled k-space. These networks require matching training and test time distributions to achieve high quality reconstructions. However, there is inherent variability in MR datasets, including different contrasts, orientations, anatomies, and institution-specific protocols. The current paradigm is to train separate models for each dataset. However, this is a demanding process and cannot exploit information that may be shared amongst datasets. To address this issue, we propose multi-task learning (MTL) schemes that can jointly reconstruct multiple datasets. We test multiple MTL architectures and weighted loss functions against single task learning (STL) baselines. Our quantitative and qualitative results suggest that MTL can outperform STL across a range of dataset ratios for two knee contrasts.

1 Introduction

To reduce MRI scan time, various iterative reconstruction schemes have been investigated [1–4]. Recently, deep learning approaches, which train a network to estimate the reconstructed image using retrospectively undersampled k-space, have shown superior efficacy over previous non-network based methods [5]. However, these networks require sufficient collection of fully sampled k-space data from similar acquisition protocols as the test-time inference data [6]. For example, to train multiple contrasts, the current paradigm is to collect multiple, fully sampled k-space data for each contrast and train each contrast-specific network separately to avoid domain shift. [6; 7]. Considering the exceptionally large variability of MR images (i.e. different contrasts, orientations, anatomies, pulse sequences), separate training requires a large effort and limits the additional information that can be gained from multiple datasets.

To address this barrier, we propose a novel multi-task learning (MTL) scheme that can jointly train a single network on a variety of datasets. The scheme jointly trains various fully sampled k-space datasets by treating them as different tasks within the same network. The network can train multiple tasks simultaneously and exploit shared, common features to prevent individual tasks from overfitting and foster better performance compared to conventional STL counterparts.

MTL has recently gained traction in various areas [8–15], but has yet to be applied to MRI reconstruction. Our study investigates how this scheme can be useful for jointly training diverse datasets. Specifically, we focus on exploring the performance on two multi-contrast datasets with different data size ratios. We test multiple MTL architectures and multi-task loss functions against STL baselines.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

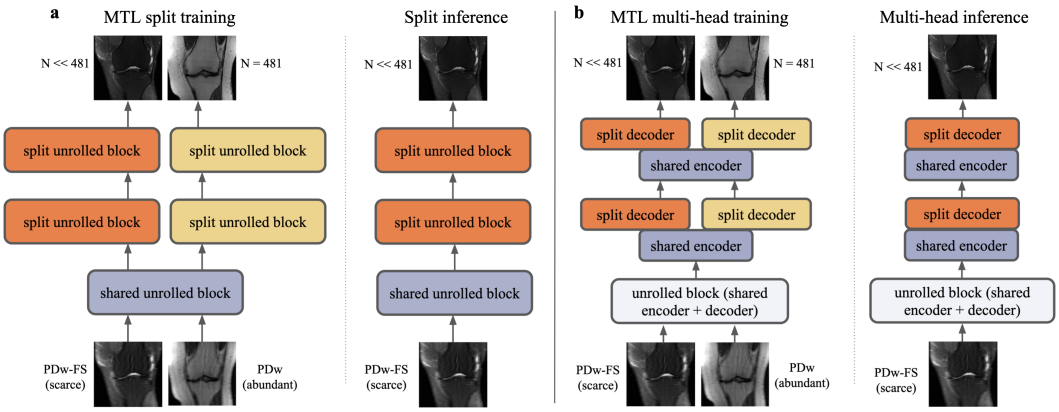

Figure 1: Multi-task learning network architectures built on an unrolled variational network for (a) fully split and (b) shared-encoder-split-decoder blocks.

Our quantitative and qualitative results suggest that the proposed scheme can perform better than STL across all dataset ratios.

2 Methods

The baseline STL network is a typical unrolled network (i.e., Variational Network) composed of a series of unrolled blocks as in [16]. Each unrolled block solves an unrolled iterative optimization problem:

$$\hat{\mathbf{x}} = \text{argmin}_{\mathbf{x}} \frac{1}{2}\|A\mathbf{x} - \tilde{\mathbf{k}}\|^2 + \lambda\Psi(\mathbf{x}), \tag{1}$$

where Ψ is a trainable regularization function modelled as a fully convolutional U-Net; A is a linear operator that consists of sensitivity map projection, Fourier transform, and discrete sampling; $\tilde{\mathbf{k}}$ is a vector of acquired k-space. The MTL network structure is composed of shared layers and task-specific layers, each consisting of multiple unrolled blocks inside the layer. In the following sections, we describe two types of network architectures developed for this study and three weighted loss functions that account for deviations between tasks.

2.1 Network architecture

Two architectures are used in the study: split and multi-head, as shown in Figure 1. Both networks start with a number of shared blocks before splitting into task-specific blocks. In the split architecture, the task-specific blocks do not share further information (1a); we refer to these task-specific blocks as split blocks. In the multi-head architecture, the U-Net encoder continues to be shared amongst tasks, but the decoder is task-specific (1b); we refer to these partially split blocks as multi-head blocks. During inference, a sample is fed through the appropriate shared and task-specific layers. Note that during inference, the MTL architecture is equivalent to STL, and each task can be inferred individually.

2.2 Multi-task loss function

In this study, we consider three different loss weighting schemes. Naive weighting addresses the data imbalance in the loss function by weighting individual task losses in an inverse relation to dataset size, as seen in Equation 2. Uncertainty weighting [9] treats the multi-task network as a probabilistic model and incorporates the task-dependent (or homoscedastic) uncertainty in the loss function. Finally, dynamic weight averaging (DWA) [14] assigns task losses based on the learning speed of each task.

$$\text{naive loss} = \frac{1}{\text{tasks}} \sum_{\text{i}}^{\text{tasks}} (\text{N} - |\text{task}_\text{i}|) \cdot \text{loss}_\text{i} \tag{2}$$

Table 1: Comparison of STL, MTL, and transfer learning for PDw-FS reconstruction

	SSIM			pSNR		
PDw-FS Slices	STL	MTL	Transfer	STL	MTL	Transfer
N = 32	0.813	0.825	0.829	31.67	31.95	32.17
N = 107	0.846	0.851	0.854	33.11	33.25	33.47
N = 253	0.857	0.858	0.862	33.7	33.8	33.94
N = 497	0.861	0.863	0.867	34.03	34.1	34.29

2.3 Dataset

We use two public knee datasets that are described in [16] and available at mridata.org [17]. The datasets contain 19 coronal proton density weighted (PDw) and 20 coronal proton density weighted fat suppression (PDw-FS) knee scans acquired with a 15-channel knee coil. The sensitivity maps of each coil are estimated from a 24×24 block at the center of the k-space using ESPIRiT [18]. 2D slices are treated as separate samples and are randomly divided into training, validation, and test sets. There are 13 PDw volumes (481 slices) and 14 PDw-FS volumes (497 slices) in the training set. In our experiments, PDw simulates the abundant dataset by using all 481 slices, and PDw-FS simulates the scarce dataset by using a percentage (497, 253, 107, or 32 slices) of the 497 slices.

2.4 Training and inference

Our models are implemented in PyTorch and trained on NVIDIA Titan Xp GPUs with 12GB of memory. For the experiments, networks with 12 unrolled blocks are used to ensure convergence. For MTL, different weighted loss functions – naive, DWA, uncertainty – are used for training the network. To assess image quality, magnitude images are normalized between 0 and 1, and peak signal-to-noise ratio (pSNR), structural similarity index (SSIM), and normalized root mean square error (nRMSE) are used. During inference, k-spaces in the test-set are undersampled identically to guarantee fair comparisons.

2.5 Experiments

For MTL, there are an exponential number of ways to compose the shared, split, and multi-head unrolled blocks. After preliminary experiments for two, six, or ten shared blocks in the beginning or middle of the network, we heuristically narrow our selection to two fully shared blocks at the beginning, followed by ten task-specific (either split or multi-head) blocks (see Figure 1). For this study, we mix and match the two aforementioned architectures with the three loss functions (naive, DWA, uncertainty) for a total of six MTL networks (see Appendix). MTL networks are jointly trained using 481 PDw slices and a percentage of PDw-FS slices. We also provide comparisons with transfer learning by taking the PDw baseline and fine-tuning all layers using PDw-FS data [6].

Source code for implementation will be publicly available via GitHub after acceptance of the paper.

3 Results

As seen in Table 1, an MTL network performs better than STL at every dataset ratio. At N = 107 and N = 253, the naive-weighted, split architecture dominates the other MTL architectures (see Appendix). Interestingly, the MTL network not only improves metrics for PDw-FS, but also for PDw at certain ratios (see Appendix).

Qualitative examination also suggests that MTL reduces errors in reconstruction. A comparison between STL and MTL is seen in Figure 2 for two different inference slices. At N = 107, the MTL image is reconstructed by the naive-weighted, split network. Although the naive-weighted, split network gives the best quantitative metrics at N = 253, qualitative observation suggests that the naive-weighted, *multi-head* network, a close runner-up, reconstructs comparably high quality images for N = 253. As such, we choose to show the reconstruction from the naive-weighted, multi-head network

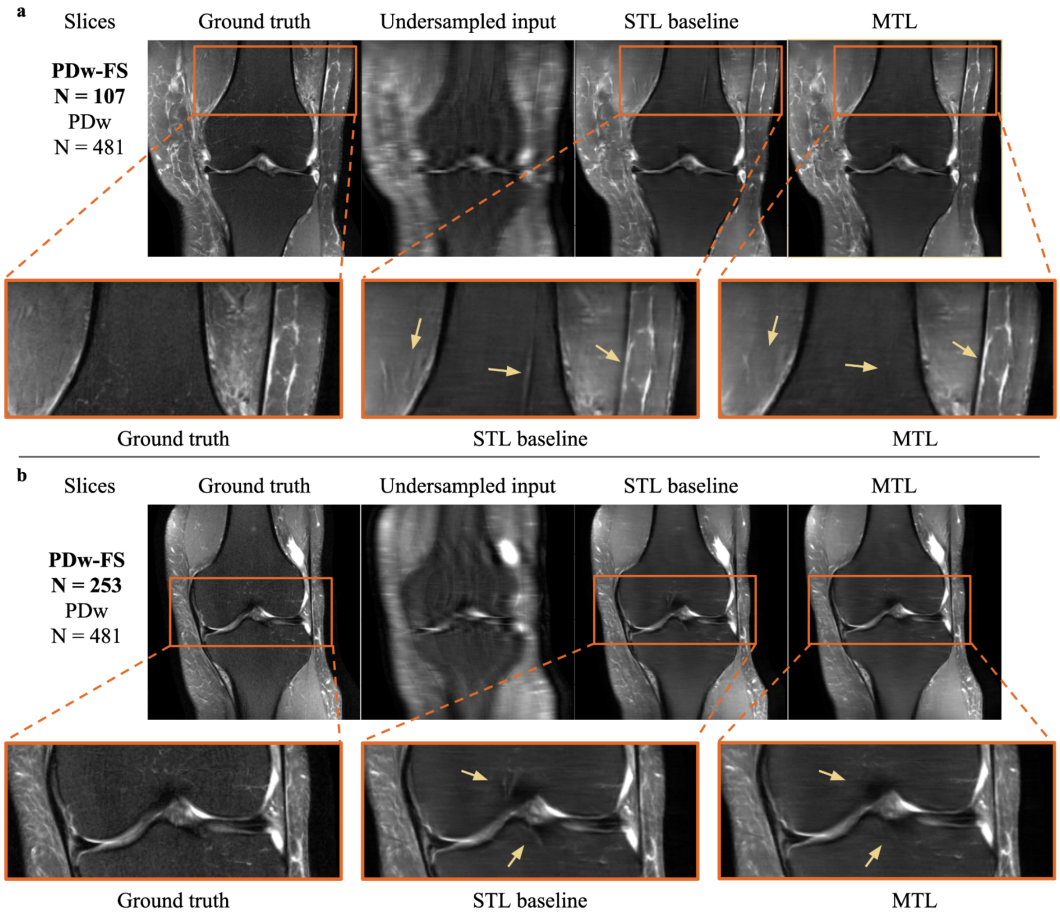

Figure 2: Representative reconstructions for MTL versus STL networks at (a) N = 107 and (b) N = 253 for PDw-FS test dataset. The arrows point to aliasing artifacts present in STL.

for N = 253. Interestingly, transfer learning performs better than MTL quantitatively (see Table 1 and Appendix). Because PDw-FS and PDw data are similar, this result suggests that multi-task learning may be better suited for more dissimilar tasks.

4 Conclusion and discussion

This study explores the use of multi-task learning in jointly training diverse MRI data for reconstruction. Our framework introduces inductive biases in the network by enforcing the sharing of useful information between tasks. We see that MTL performs better than STL baselines across a range of abundant versus scarce ratios, for both PDw and PDw-FS datasets. We did not expect MTL to outperform STL for the abundant data, and this positive sharing of information may guide selection of tasks in future experiments.

Our finding that transfer learning marginally outperforms MTL suggests that MTL may be better suited for more dissimilar tasks such as different orientations, anatomies rather than multi-contrasts. Moreover, it is possible that once we use larger datasets, STL will dominate transfer learning, and MTL's main competitor will be STL.

One noted difficulty is selecting an appropriate architecture and loss function, as negative transfer between the dataset can occur (see Appendix). All in all, our study provides a proof of concept that MTL can be successfully used in jointly training multiple contrasts for MRI reconstruction.

References

[1] Jong Chul Ye. Compressed sensing mri: a review from signal processing perspective. *BMC Biomedical Engineering*, 1, 03 2019. doi: 10.1186/s42490-019-0006-z.

[2] Xinlin Zhang, Di Guo, Yiman Huang, Ying Chen, Liansheng Wang, Feng Huang, and Xiaobo Qu. Image reconstruction with low-rankness and self-consistency of k-space data in parallel mri, 2019.

[3] Anagha Deshmane, Vikas Gulani, Mark Griswold, and Nicole Seiberlich. Parallel mr imaging. *Journal of magnetic resonance imaging : JMRI*, 36:55–72, 07 2012. doi: 10.1002/jmri.23639.

[4] Mark Murphy, Marcus Alley, James Demmel, Kurt Keutzer, Shreyas Vasanawala, and Michael Lustig. Fast ℓ_1-spirit compressed sensing parallel imaging mri: Scalable parallel implementation and clinically feasible runtime. *IEEE Transactions on Medical Imaging*, 31(6):1250–1262, 2012. doi: 10.1109/TMI.2012.2188039.

[5] Anuroop Sriram, Jure Zbontar, Tullie Murrell, Aaron Defazio, C. Lawrence Zitnick, Nafissa Yakubova, Florian Knoll, and Patricia Johnson. End-to-end variational networks for accelerated mri reconstruction, 2020.

[6] Florian Knoll, Kerstin Hammernik, Erich Kobler, Thomas Pock, Michael P Recht, and Daniel K Sodickson. Assessment of the generalization of learned image reconstruction and the potential for transfer learning. *Magnetic resonance in medicine*, 81(1):116–128, 2019.

[7] Mohammad Zalbagi Darestani and Reinhard Heckel. Accelerated mri with un-trained neural networks. *IEEE Transactions on Computational Imaging*, 7:724–733, 2021.

[8] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning, 2016.

[9] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics, 2018.

[10] Michael Crawshaw. Multi-task learning with deep neural networks: A survey, 2020.

[11] Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai, and Luc Van Gool. Multi-task learning for dense prediction tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1–1, 2021. ISSN 1939-3539. doi: 10.1109/tpami.2021.3054719. URL `http://dx.doi.org/10.1109/TPAMI.2021.3054719`.

[12] Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto multi-task learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/685bfde03eb646c27ed565881917c71c-Paper.pdf`.

[13] Sebastian Ruder. An overview of multi-task learning in deep neural networks, 2017.

[14] Shikun Liu, Edward Johns, and A. Davison. End-to-end multi-task learning with attention. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1871–1880, 2019.

[15] Chen Chen, Wenjia Bai, and Daniel Rueckert. Multi-task learning for left atrial segmentation on ge-mri. *Lecture Notes in Computer Science*, page 292–301, 2019. ISSN 1611-3349. doi: 10.1007/978-3-030-12029-0_32. URL `http://dx.doi.org/10.1007/978-3-030-12029-0_32`.

[16] Kerstin Hammernik, Teresa Klatzer, Erich Kobler, Michael P. Recht, Daniel K. Sodickson, Thomas Pock, and Florian Knoll. Learning a variational network for reconstruction of accelerated mri data. *Magnetic Resonance in Medicine*, 79(6):3055–3071, 2018. doi: https://doi.org/10.1002/mrm.26977. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/mrm.26977`.

[17] F Ong, S Amin, SS Vasanawala, and M Lustig. An open archive for sharing mri raw data. In *ISMRM & ESMRMB Joint Annu. Meeting*, page 3425, 2018.

[18] Martin Uecker, Peng Lai, Mark Murphy, Patrick Virtue, Michael Elad, John Pauly, Shreyas Vasanawala, and Michael Lustig. Espirit—an eigenvalue approach to autocalibrating parallel mri: where sense meets grappa. *Magnetic resonance in medicine : official journal of the Society of Magnetic Resonance in Medicine / Society of Magnetic Resonance in Medicine*, 71, 03 2014. doi: 10.1002/mrm.24751.

A Appendix

Tables 2 and 3 provide benchmarks for how well various MTL networks perform, compared to STL baselines and transfer learning. Transfer learning is not shown in Table 3 because transfer learning is fine-tuned for PDw-FS only. Experimental setups are described in detail in Section 2.5. Blank spaces arise from experiments that were not run.

Table 2: SSIM of PDw-FS test dataset for different MTL networks

		PDw-FS Slice Count			
Architecture	MTL Loss	N = 32	N = 107	N = 253	N = 497
STL (PDw & PDw-FS data)		0.813	0.846	0.857	0.861
Transfer learning		0.829	0.854	0.862	0.867
Split	Naive	0.825	0.851	0.858	0.863
Split	DWA		0.847	0.856	
Split	Uncertainty	0.806	0.801	0.855	
Multi-head	Naive	0.819	0.849	0.857	
Multi-head	DWA		0.846	0.857	
Multi-head	Uncertainty	0.822	0.848	0.854	

Table 3: SSIM of PDw test dataset for different MTL networks

		PDw-FS Slice Count			
Architecture	MTL Loss	N = 32	N = 107	N = 253	N = 497
STL (PDw data only)		0.903	0.903	0.903	0.903
Split	Naive	0.891	0.9	0.90	0.905
Split	DWA		0.9	0.899	
Split	Uncertainty	0.901	0.852	0.902	
Multi-head	Naive	0.896	0.899	0.899	
Multi-head	DWA		0.897	0.902	
Multi-head	Uncertainty	0.904	0.903	0.903	

Figure 3 compares transfer learning, STL, and MTL experiments for PDw-FS. These are the same PDw-FS slices shown in Figure 2, with the addition of the transfer learning results. The improvement from MTL to transfer learning is less pronounced than the improvement from STL to MTL. Figure 4 compares STL and MTL for PDw.

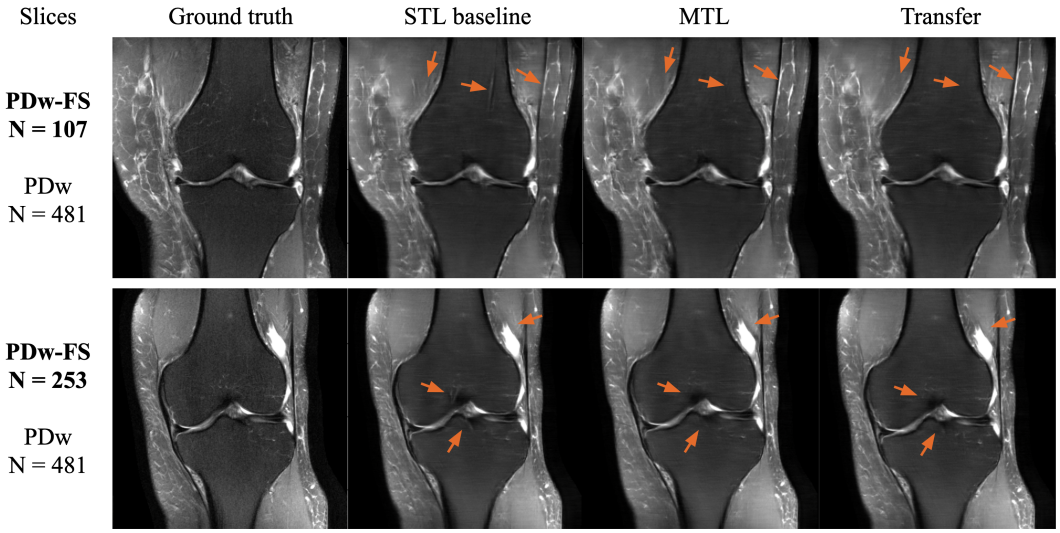

Figure 3: Comparison of STL, MTL, and transfer learning for PDw-FS slices. The first row shows images created by jointly training 107 PDw-FS images with 481 PDw images for STL and MTL. The second row shows the results of jointly training 253 PDw-FS images with 481 PDw images for STL and MTL. For transfer learning, fine-tuning is done with 107 and 253 slices in the first and second rows, respectively.

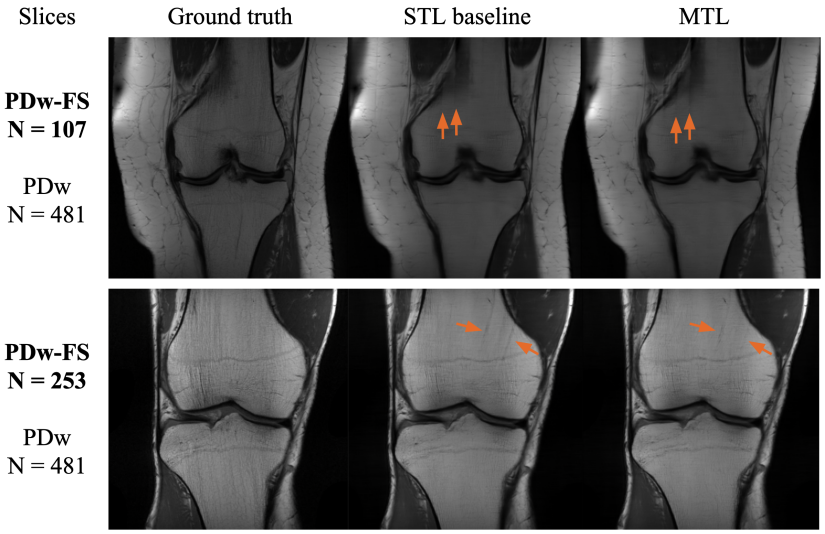

Figure 4: Comparison of STL and MTL for PDw slices. For MTL, the first and second rows show images created by jointly training 107 or 253 PDw-FS images, respectively, with 481 PDw images. The STL baseline in both rows is trained using 481 slices of PDw only. The MTL network shown here is has a multi-head architecture with uncertainty weighting.