Training Binary Layers by Self-Shrinking of Sigmoid Slope: Application to Fast MRI Acquisition

This is the author's version of the article that has been presented at IEEE BioCAS2022
The editorial version of the paper is available at http://dx.doi.org/10.1109/BioCAS54905.2022.9948688

Training Binary Layers by Self-Shrinking of Sigmoid Slope: Application to Fast MRI Acquisition

F. Martinini*, A. Enttsel*, A. Marchioni*†, M. Mangia *†, F. Pareschi ‡†, R. Rovatti *†, G. Setti ‡†
*DEI, †ARCES, University of Bologna, Italy -
{filippo.martinini, andriy.enttsel, alex.marchioni, mauro.mangia, riccardo.rovatti}@unibo.it
‡DET, Politecnico di Torino, Italy - {fabio.pareschi, gianluca.setti}@polito.it

***Abstract*—Deep Neural Networks (DNN) have become popular and widespread because they combine computational power and flexibility, but they may present critical hyper-parameters that need to be tuned before the model can be trained. Recently, the use of trainable binary masks in the field of Magnetic Resonance Imaging (MRI) acquisition brought new state-of-the-art results, but with the disadvantage of introducing a bulky hyper-parameter, which tuning is usually time-consuming. We present a novel callback-based method that is applied during training and turns the tuning problem into a triviality, also bringing non-negligible performance improvements. We test our method on the fastMRI dataset.**

***Index Terms*—Binary Layer, DNN, Sigmoid, Callback, Tuning, MRI reconstruction**

I. Introduction

Deep Neural Networks (DNN) have shown a great adaptability power and for the last decade specialists have been extending their use to all the imaginable engineering fields. As a consequence, the design of custom structures in DNN is becoming more and more a common practice. For example, the introduction of binary trainable masks [1], [2] and the use of non-uniform trainable quantizers [3] are all based on peculiar and unique layers.

All DNN share the same learning principles, first a batch of input samples is used to produce a batch of output samples, that is compared with the corresponding reference labels to produce an average loss (or error). Second, the gradients of the loss are backpropagated, and the network parameters are updated in the direction where the loss locally minimum. This step characterizes the method named backpropagation (BP).

In general, to obtain meaningful changes inside the model, BP should propagate neither non-null gradients, nor extremely large gradients, e.g., a gradient close to zero would bring negligible contributions and an exploding one would probably introduce baneful and unstable changes. The two configurations are known as the vanishing gradient problem and the exploding gradient problem [4]. To deal with them, the user is usually asked to tune the learning rate, but when they intrinsically derive from the structure of the model, learning rate tuning is not enough. For example, vanishing/exploding gradient problems reside in DNN built with recurrent layers or LSTM [5], and knowing how to avoid them is a skill one yearns for.

Custom layers must go through a careful design to prevent any arising gradient-related problems. In the design of a trainable binary mask [1], [2], it is necessary to substitute the hard threshold responsible for the binary values with a soft one, where the first occludes the gradient flow (vanishing gradient), the latter grants a swift BP. This solution introduces a hyper-parameter that controls the degree of binarization of the mask. We can imagine BP as a tap whose water flow is controlled by a knob. In [1] the knob is moved from shut to open, while here we try to answer at the question "how much water should flow?". The answer we propose is a gradually decreasing flow training strategy named *Self-Shrinking of Sigmoid Slope*, that is like a knob that is slowly turned close.

The proposed method is adopted in the design of a Magnetic Resonance Imaging (MRI) acquisition system. MRI is a well known non-invasive medical technology used to acquire high-resolution images of the inner structures of the human body. The main disadvantage of MRI is the long sensing time, which takes place in a claustrophobic space. The reduction of the MRI scan duration is one of the most investigated key objective to achieve, and literature provides a rich collection of attempts [6]–[10]. Considering [1] a baseline framework, our work, similarly to [11], [12], contributes in advancing this research direction. In particular, we introduce an efficient and general purpose *binary mask training strategy*, that does not simply remove the need of a hyper-parameter tuning, but brings an overall benefit in term of performance.

The rest of the paper is organized as follows. Section II presents the adopted case study with a DNN embedding a custom layer generating a binary mask, Section III reports the achieved results in the processing of grayscale MRI images and Section IV contains the conclusions.

II. Custom Layer Generating Binary Mask

We refer to the original work presented in [1] as the main case study. This method, named Learning-based Optimization of the Under-sampling PattErn (LOUPE), accelerates the ac-

This is the author's version of the article that has been presented at IEEE BioCAS2022
The editorial version of the paper is available at http://dx.doi.org/10.1109/BioCAS54905.2022.9948688

quisition of an MRI machine by simulating the sensing and the processing of every scan with a novel DNN-based solution.

An MRI system produces a 2D matrix in the spatial domain for each body scan. This is achieved by sensing the frequencies every human body emits when hit by a controlled magnetic field. A traditional MRI system acquires the whole set of frequency components and computes the inverse Fourier Transform to obtain the clinical image. The general idea behind LOUPE (and behind several similar solutions proposed in literature) is to sample only a subset of the entire set of frequency components composing each scan; the direct consequence of the lower sampling rate is the desired acquisition speed-up.

With more details, LOUPE wants to return a high-resolution $N \times N$ image, only acquiring $r \times N \times N$ of its frequencies, where $R = 1/r > 1$ is called the *acceleration rate*. To reach this goal, the whole structure to be trained includes two subsystems: Encoder and Decoder. The first simulates the subsampling acquisition in the frequency domain by multiplying the matrix containing the full set of acquirable frequencies with a binary mask $\boldsymbol{M_\gamma} \in \{0,1\}^{N\times N}$ that selects which frequency components will be physically acquired or discarded at inference time. Ones and zeros in $\boldsymbol{M_\gamma}$ are placed during the training phase and depend on a set of trainable parameters $\boldsymbol{\gamma} \in \mathbb{R}^{N\times N}$. The second part, the Decoder, is a more conventional DNN structure processing the Encoder output and attempting to restore a full resolution image, erasing the artifacts introduced by the undersampling process.

The overall network, composed by both Encoder and Decoder, works as an Autoencoder [13] $\mathrm{Dec}_{\boldsymbol{\theta}}\big(\mathrm{Enc}_{\boldsymbol{\gamma}}(\cdot)\big)$ that, at training time, uses full resolution scans as input and tries to reproduce them as output. The encoder takes a single scan $\boldsymbol{x} \in [0,1]^{N\times N}$ and produces the undersampled frequency content $\boldsymbol{y} \in \mathbb{C}^{N\times N}$ (real and imaginary parts) by applying the binary mask in the frequency domain, i.e.,

$$\boldsymbol{y} = \mathrm{Enc}_{\boldsymbol{\gamma}}(\boldsymbol{x}) = \boldsymbol{M_\gamma} \circ \mathcal{F}(\boldsymbol{x}) \tag{1}$$

where $\mathcal{F}(\cdot)$ and $\mathcal{F}^{-1}(\cdot)$ stand for direct and inverse 2D Fourier transform, and $\circ$ is the Hadamard element-wise product.

The decoder $\mathrm{Dec}_{\boldsymbol{\theta}}(\boldsymbol{y})$ recovers the input scan in two stages: *i*) the first computes $|\mathcal{F}^{-1}(\boldsymbol{y})|$ and yields a first approximation/reconstruction; *ii*) the second stage removes the artifacts by applying pixel-wise adjustments computed by a sub-network $\mathcal{D}_{\boldsymbol{\theta}}$ [1], [2] that is a function of the trainable parameters $\boldsymbol{\theta}$.

$$\hat{\boldsymbol{x}} = \mathrm{Dec}_{\boldsymbol{\theta}}(\boldsymbol{y}) = |\mathcal{F}^{-1}(\boldsymbol{y})| + \mathcal{D}_{\boldsymbol{\theta}}(\mathcal{F}^{-1}(\boldsymbol{y})) \tag{2}$$

The adopted sub-network is a slight modification of the U-net architecture [14].

Once the training is over, we only keep the mask $\boldsymbol{M_\gamma}$ to guide the MRI through the acquisition and the decoder $\mathrm{Dec}_{\boldsymbol{\theta}}(\boldsymbol{y})$ to restore the undersampled images.

At inference time, binary masks are obtained by element-wisely thresholding a sub-output of the sub-network generating $\boldsymbol{M_\gamma}$. Inconveniently, the threshold function introduces a vanishing gradient bottleneck that would hamper the learning of $\boldsymbol{\gamma}$ if used for training, since it makes the gradient of the loss function always null.

To overcome this critical aspect, the authors of [1] create a specific mask generator sub-network for training $\boldsymbol{M_\gamma}$, and substitute the threshold function with a sigmoid function $\sigma_s(\cdot)$ that depends on a scaling factor s, called *slope*.

$$\sigma_s(p) = \frac{1}{1+\exp(-s\,p)} \tag{3}$$

where $\sigma_s(p)$ converges to a threshold function for $s \to \infty$.

The sub-network generating the mask entries is here detailed. The undersampling pattern controlling R is obtained starting from the so called *probability mask* $\boldsymbol{S}(\boldsymbol{\gamma}) = \sigma_t(\boldsymbol{\gamma})$ where each matrix entry represents the probability of that element to be included in $\boldsymbol{M_\gamma}$. Here t is not critical since $\sigma_t(\cdot)$ only serves the purpose of having every element of $\boldsymbol{\gamma}$ mapped in the $[0,1]$. A typical employed value is $t = 5$.

$\boldsymbol{S}(\boldsymbol{\gamma})$ works as a probability mask since, during the training, single matrices $\boldsymbol{M_\gamma}$ are obtained by matching $\boldsymbol{S}(\boldsymbol{\gamma})$ with matrices $\boldsymbol{U}$ containing instances of random variables uniformly distributed in $[0,1]$, i.e.,

$$\boldsymbol{M_\gamma} = \sigma_s(\boldsymbol{S}(\boldsymbol{\gamma}) - \boldsymbol{U}) \tag{4}$$

Here, s is critical since it is the parameter that controls the final degree of resemblance of $\boldsymbol{M_\gamma}$ with a binary mask.

The inverse of the acceleration rate r is the average probability of acquiring every sample, i.e., the average of $\boldsymbol{S}(\boldsymbol{\gamma})$. LOUPE leverages this property by introducing a rescaling layer in the encoder sub-network that imposes $r = R^{-1}$ as the average of $\boldsymbol{S}(\boldsymbol{\gamma})$. As sketched in Fig. 1, the rescaling layer is placed immediately after the layer producing $\boldsymbol{S}(\gamma)$, such that:

$$\boldsymbol{P}_r(\boldsymbol{S}(\boldsymbol{\gamma})) = \begin{cases} \frac{r}{\langle S(\boldsymbol{\gamma})\rangle}\boldsymbol{S}(\boldsymbol{\gamma}) & \text{if } \langle \boldsymbol{S}(\boldsymbol{\gamma})\rangle \geq r \\ \mathbf{1} - \frac{1-r}{1-\langle \boldsymbol{S}(\boldsymbol{\gamma})\rangle}(\mathbf{1} - \boldsymbol{S}(\boldsymbol{\gamma})) & \text{if } \langle \boldsymbol{S}(\boldsymbol{\gamma})\rangle < r \end{cases} \tag{5}$$

where $\langle\cdot\rangle$ is an operator computing the average of the entries of the matrix given as argument. Note that $\boldsymbol{P}_r(\boldsymbol{S}(\boldsymbol{\gamma})) \in [0,1]^{N\times N}$ is still a probability mask but now has $\langle \boldsymbol{P}_r(\gamma)\rangle = r$. This trick gives the user control over R. To complete the process, the output of the rescaling layer contributes to the generation of $\boldsymbol{M_\gamma}$ as before

$$\boldsymbol{M_\gamma} = \sigma_s(\boldsymbol{P}_r(\boldsymbol{S}(\boldsymbol{\gamma})) - \boldsymbol{U}) \tag{6}$$

A. On the importance of s, the Self-Shrinking of Sigmoid Slope

Although $\boldsymbol{M_\gamma}$ is obtained by hard thresholding $\boldsymbol{P}_r(\boldsymbol{S}(\boldsymbol{\gamma}))$ at inference time, the sigmoid function in (6) characterizing the training inevitably leads to the introduction of the hyper-parameter s, which can severely limit the final performance of the model if poorly chosen. In particular, one can fall into

This is the author's version of the article that has been presented at IEEE BioCAS2022
The editorial version of the paper is available at http://dx.doi.org/10.1109/BioCAS54905.2022.9948688

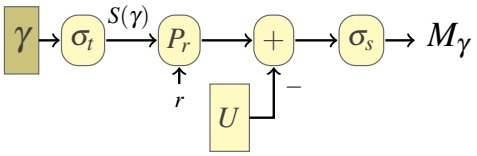

Fig. 1. LOUPE encoder sub network as in [1].

two error classes depending on whether s is too low or too high, respectively: *i*) at training time, many mask entries are far from either 0 or 1, so that hard thresholding significantly changes the mask characterization at inference time; *ii*) at training time, the sigmoid function risks meeting a vanishing gradient problem. It is evident how the tuning of s may require several attempts, resulting in a long procedure. Let us remark that s cannot be a trainable parameter. In fact, if it was, the net would promote $s = 0$, that corresponds to the degenerate case where the $\boldsymbol{M}_\gamma$ is filled with 0.5 only, i.e., undersampling is not performed.

In this work, we propose a method to reduce the burden of slope-like parameters tuning via the use of a *callback* function. Even if we only test our approach within the LOUPE framework, it should be valid for a wide range of problems, e.g., the works in [15], [16].

Confining our analysis within the DNN world, a callback function is a function that affects the training only during certain training time windows, e.g., at the end of an epoch. Callback functions act in parallel with backpropagation, examples are: *i*) Early Stop (ES), that checks whether the loss is descending at the end of each epoch, and stops the training if the loss has not been lowering for a given consecutive number of epochs; *ii*) Reduce Learning Rate on Plateau (RLRP), which, similarly to ES, acts only after a certain amount of epochs during which the model does not improve, and reduces the learning rate; *iii*) Model Checkpoint (MC), which saves the weights of the model, updating them every time the loss gets smaller. The number of epochs after which the callback activates is named *patience*.

We propose and implement a callback that automatically tunes the slope of the sigmoid by gradually increasing s. In particular, every time a patience number of epoch passes without a loss reduction, s is raised. The procedue is repeated until a maximum admitted value value is reached. Once the training is over, thanks to MC, only the weights giving the lowest loss have been saved, and the model with the optimal auto-tuned s is returned. We call our callback Self-Shrinking of Sigmoid Slope (4S). Its parameters are: the patience p and a factor m used to raise the slope every time the callback is triggered, such that $s_{i+1} = ms_i$. We observe that these degrees of freedom are not critical, and one can simply set a broad enough set of s by using $p = 40$ and $m = 1.5$.

4S has a real impact for at least two reasons: *i*) the training process gradually fossilizes $\boldsymbol{M}_\gamma$, until s is high enough to make the mask almost binary, meaning only the decoder is still training hence reducing the performance drop due to the mask binarization; *ii*) it avoids the manual tuning of s.

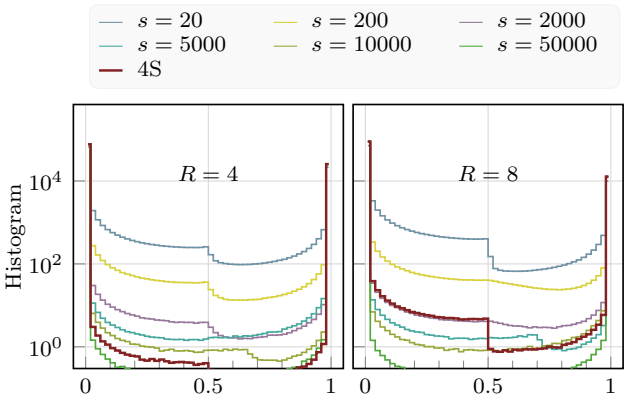

Fig. 2. Histograms for $R = 4$ and $R = 8$ of LOUPE trained with several s values and 4S, that returns two models with $s = 18800$ and $s = 1675$ respectively for $R = 4$ and $R = 8$.

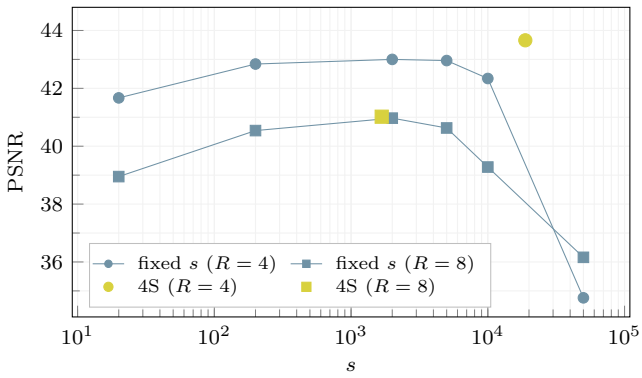

Fig. 3. PSNR for $R = 4$ (circles) and $R = 8$ (squares), trained with several fixed s values (blue points) and 4S (yellow points). 4S models return $s = 18800$ and $s = 1675$ respectively for $R = 4$ and $R = 8$.

III. Numerical Evidences

To assess the quality of our strategy, also following the structure of [1], we use a subset of the publicly available dataset fastMRI [17], already adopted in international competitions such as [18]. The subset of the original dataset consists of 2269 normalized grayscale MRI images with shape 320×320 split into 1895 training images (83.5%), 188 validation images (8.3%) and 186 test images (8.2%). Scans are naturally grouped in *volumes* (every volume represents a different knee), each comprising around 40 slices. Every slice is normalized with the highest magnitude value of its volume.

Coherently with the model we propose, the undersampling mask $\boldsymbol{M}_\gamma$ is not provided by the user but is entirely learnt by the model. All the inference tests adopt a deterministic binarized version of the masks.

We train our models on an Nvidia V100 using Adam optimizer, an initial learning rate of 0.01, and a batch size

This is the author's version of the article that has been presented at IEEE BioCAS2022
The editorial version of the paper is available at http://dx.doi.org/10.1109/BioCAS54905.2022.9948688

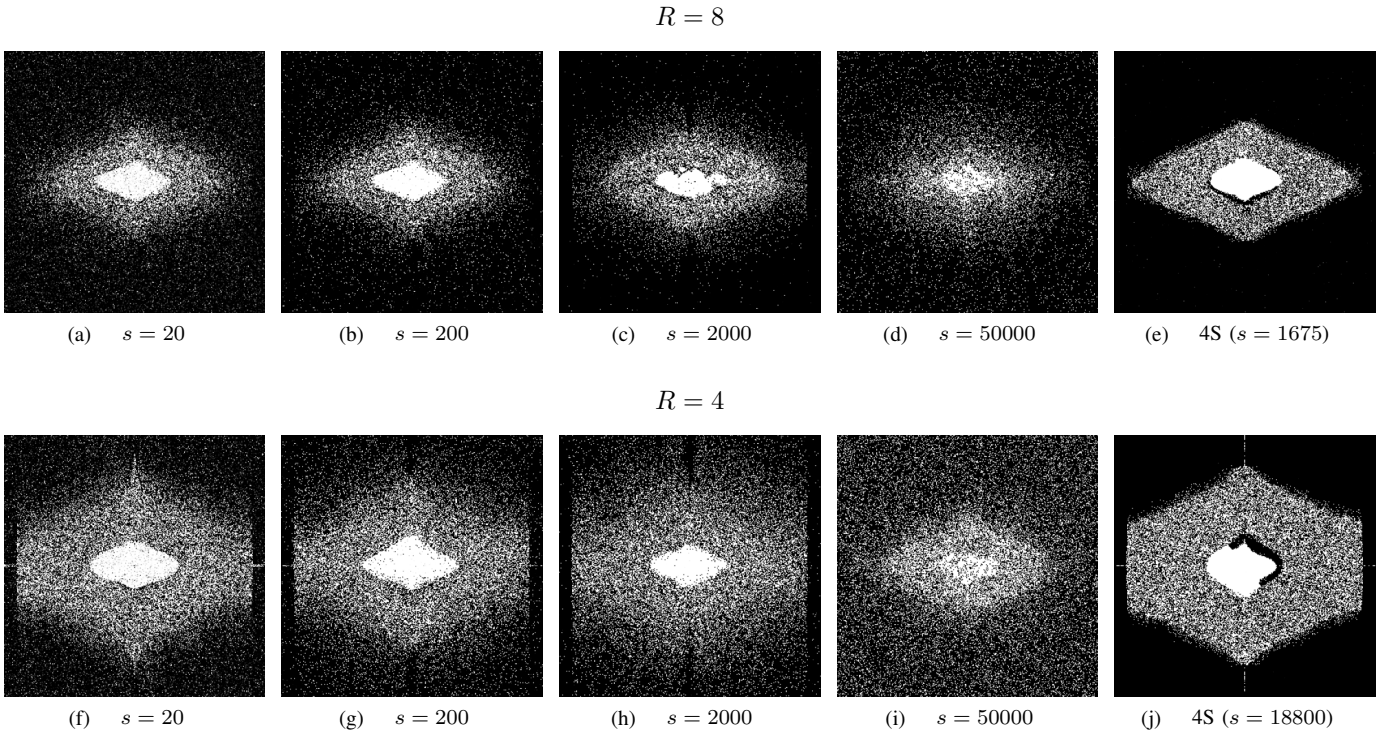

Fig. 4. Probability masks for $R = 8$ (a, b, c, d) and $R = 4$ (f, g, h, i), for cases where s is a hyperparameter (fixed $s = \{20, 200, 2000, 50000\}$) and for 4S (e, j) ($s$ is automatically determined at training time). Without 4S, low values of s produce mask values far from either 0 or 1, so that the thresholding operation at inference time becomes critical. High values of s hamper the training, and produce poor masks.

of 16. To compare the quality of every reconstructed image, we use the Peak Signal-to-Noise Ratio (PSNR) defined as:

$$\text{PSNR} = -10 \log_{10} (\text{MSE}) \tag{7}$$

where MSE is the mean square error between $\boldsymbol{x}$ and the decoder output $\hat{\boldsymbol{x}}$.

In this section, we compare models trained with and without 4S. The training of the latter uses RLRP, ES, and MC. The former are trained using 4S, ES, and MC in a first training round, followed by a second round where RLRP replaces 4S. This strategy has the advantage of letting the model largely explore many s configurations.

To make clear how 4S acts, Fig. 2 shows the histograms for the trained masks entries as in (6) (before hard thresholding) without 4S, considering $s = \{20, 200, 2000, 5000, 10000, 50000\}$, and with 4S. In particular, we randomly draw 1000 masks $\boldsymbol{M}_{\boldsymbol{\gamma}}$ for each configuration and show their average values distribution. Results regard both $R = 4$ and $R = 8$. As expected, high s values return masks containing almost binary values while low s values push several mask entries far from either 0 or 1.

4S shows excellent preservation of the mask binary structure, returning final s equal to 18800 and 1675 for R equal to 4 and 8, respectively. This aspect, coupled with the model performance in terms of PSNR (results displayed in Fig. 3), further evidences the superiority of our strategy and the importance of the choice of s.

Finally, in Fig. 4 we give a visual representation of the masks taken from models trained using $R = 4, 8$, with $s = 20, 200, 2000, 50000$ when using 4S. In accordance with our premises, the mask associated with $s = 20$ contains many non-binary values that spoil its binary structure. Conversely, the mask associated with $s = 50000$ maintains a nice binary configuration but clearly loses the signal adaptation. 4S collects the advantages of both the described configurations, donating a strong binary characterization and structure to the mask.

IV. Conclusions

We present a callback-based novel method to turn the problem of tuning a class of hyper-parameters inside a DNN into a triviality. In particular, we show our method works within the LOUPE framework, which implements a mask generator whose final output critically depends on a hyper-parameter. We demonstrate that one can use our novel method and forget about the tuning, nevertheless obtaining competitive results.

Acknowledgments

This work was supported in part by the Italian Ministry for Education, University and Research (MIUR) under the program "Dipartimenti di Eccellenza" (2018-2022) and in part by the SmartData Center of Politecnico di Torino.

REFERENCES

[1] C. D. Bahadir, A. Q. Wang, A. V. Dalca, and M. R. Sabuncu, "Deep-Learning-Based Optimization of the Under-Sampling Pattern in MRI," *IEEE Transactions on Computational Imaging*, vol. 6, pp. 1139–1152, 2020.

[2] C. D. Bahadir, A. V. Dalca, and M. R. Sabuncu, "Learning-Based Optimization of the Under-Sampling Pattern in MRI," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2019, pp. 780–792.

[3] B. Sun, H. Feng, K. Chen, and X. Zhu, "A Deep Learning Framework of Quantized Compressed Sensing for Wireless Neural Recording," *IEEE Access*, vol. 4, pp. 5169–5178, 2016.

[4] S. Hochreiter, "IM FACH INFORMATIK Untersuchungen zu dynamischen neuronalen Netzen," *Iclr*, no. April, p. 14, 1991.

[5] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, "Advances in optimizing recurrent networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, may 2013, pp. 8624–8628.

[6] S. Ravishankar and Y. Bresler, "MR Image Reconstruction From Highly Undersampled k-Space Data by Dictionary Learning," *IEEE Transactions on Medical Imaging*, vol. 30, no. 5, pp. 1028–1041, 5 2011.

[7] G. Wang, J. C. Ye, K. Mueller, and J. A. Fessler, "Image reconstruction is a new frontier of machine learning," *IEEE Transactions on Medical Imaging*, vol. 37, no. 6, pp. 1289–1296, 2018.

[8] Z. Zhan, J.-F. Cai, D. Guo, Y. Liu, Z. Chen, and X. Qu, "Fast multiclass dictionaries learning with geometrical directions in mri reconstruction," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 9, pp. 1850–1861, 2016.

[9] S. Ma, W. Yin, Y. Zhang, and A. Chakraborty, "An efficient algorithm for compressed mr imaging using total variation and wavelets," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[10] X. Qu, Y. Hou, F. Lam, D. Guo, J. Zhong, and Z. Chen, "Magnetic resonance image reconstruction from undersampled measurements using a patch-based nonlocal operator," *Medical Image Analysis*, vol. 18, no. 6, pp. 843–856, 8 2014.

[11] F. Martinini, M. Mangia, A. Marchioni, R. Rovatti, and G. Setti, "A deep learning method for optimal undersampling patterns and image recovery for mri exploiting losses and projections," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 4, pp. 713–724, 2022.

[12] J. Xie, J. Zhang, Y. Zhang, and X. Ji, "Puert: Probabilistic undersampling and explicable reconstruction network for cs-mri," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 4, pp. 737–749, 2022.

[13] Ian Goodfellow and Yoshua Bengio and Aaron Courville, *Deep Learning*. MIT Press, 2016.

[14] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9351, pp. 234–241, 2015.

[15] F. Juefei-Xu, V. N. Boddeti, and M. Savvides, "Local binary convolutional neural networks," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, 2017, pp. 4284–4293.

[16] N. A. Golilarz and H. Demirel, "Thresholding neural network (TNN) with smooth sigmoid based shrinkage (SSBS) function for image denoising," *Proceedings - 9th International Conference on Computational Intelligence and Communication Networks, CICN 2017*, vol. 2018-January, pp. 67–71, 3 2018.

[17] J. Zbontar *et al.*, "fastMRI: An Open Dataset and Benchmarks for Accelerated MRI."

[18] M. J. Muckley *et al.*, "Results of the 2020 fastMRI Challenge for Machine Learning MR Image Reconstruction," *IEEE Transactions on Medical Imaging*, pp. 1–1, 2021.