On instabilities of deep learning in image reconstruction - Does AI come at a cost?

Vegard Antun, Francesco Renna, Clarice Poon, Ben Adcock, Anders C. Hansen

Deep learning, due to its unprecedented success in tasks such as image classification, has emerged as a new tool in image reconstruction with potential to change the field. In this paper we demonstrate a crucial phenomenon: deep learning typically yields unstable methods for image reconstruction. The instabilities usually occur in several forms: (1) tiny, almost undetectable perturbations, both in the image and sampling domain, may result in severe artefacts in the reconstruction, (2) a small structural change, for example a tumour, may not be captured in the reconstructed image and (3) (a counterintuitive type of instability) more samples may yield poorer performance. Our new stability test with algorithms and easy to use software detects the instability phenomena.

The test is aimed at researchers to test their networks for instabilities and for government agencies, such as the Food and Drug Administration (FDA), to secure safe use of deep learning methods.

The importance of stable and accurate methods for image reconstruction for inverse problems is hard to over estimate. These techniques form the foundation for essential tools across the physical and life sciences such as Magnetic Resonance Imaging (MRI), Computerised Tomography (CT), fluorescence microscopy, electron tomography, Nuclear Magnetic Resonance (NMR), radio interferometry, lensless cameras etc. Moreover, stability is traditionally considered a necessity in order to secure reliable and trustworthy methods used in, for example, cancer diagnosis. Hence, there is an extensive literature on designing stable methods for image reconstruction in inverse problems [1–4].

Artificial intelligence (AI) techniques such as deep learning and neural networks [5] have provided a new paradigm with new techniques in inverse problems [6]–[14] that may change the field. In particular, the reconstruction algorithms learn how to best do the reconstruction based on training from previous data, and through this training procedure aim to optimise the quality of the reconstruction. This is a radical change from the current state of the art both from an engineering, physical and mathematical point of view.

AI and deep learning has already changed the field of computer vision and image classification [15]–[18], where the performance is now referred to as super human [19]. However, the success comes with a price. Indeed, the methods are highly unstable. It is now well established [20]–[22] that high performance deep learning methods for image classification are subject to failure given tiny, almost invisible perturbation of the image. An image of a cat may be classified correctly, however, a tiny change, invisible to the human eye, may cause the algorithm to change its classification label from cat to a fire truck, or another label far from the original.

In this paper we establish the instability phenomenon of deep learning in image reconstruction for inverse problems. A potential surprising conclusion is that the phenomenon may be independent of the underlying mathematical model. For example, MRI is based on sampling the Fourier transform whereas CT is based on sampling the Radon transform. These are rather different models, yet the instability phenomena happen for both sampling modalities when using deep learning.

There is, however, a big difference between the instabilities of deep learning for image classification and our results on instabilities of deep learning for image reconstruction. Firstly, in the former case there is only one thing that could go wrong: a small perturbation results in a wrong classification. In image reconstruction there are several potential forms of instabilities. In particular, we consider three crucial issues: (1) instabilities with respect to tiny perturbations, (2) instabilities with respect to small structural changes (for example a brain image with or without a small tumour), (3) instabilities with respect to changes in the number of samples. Secondly, the two problems are totally unrelated. Indeed, the former problem is, in its simplest form, a decision problem, and hence the decision function ("is there a cat in the image?") to be approximated is necessarily discontinuous. However, the problem of reconstructing an image from Fourier coefficients, as is the problem in MRI, is completely different. In this case there exist stable and accurate methods that depend continuously on the input. It is therefore paradoxical that deep learning leads to unstable methods for problems that can be solved accurately in a stable way.

The networks we have tested are unstable either in the form of (1) or (2) or both. Moreover, networks that are highly stable in one of the categories tend to be highly unstable in the other. The instability in form of (3), however, occur for some networks but not all. The new findings raise two fundamental questions:

(i) Does AI, as we know it, come at a cost? Is instability a necessary by-product of our current AI techniques?

(ii) Can reconstruction methods based on deep learning always be safely used in the physical and life sciences? Or, are there cases for which instabilities may lead to, for example, incorrect medical diagnosis if applied in medical imaging?

The scope of this paper is on the second question, as the first question is on foundations, and our stability test provides the starting point for answering question (ii). However, even if instabilities occur, this should not rule out the use of deep learning methods in inverse problems. In fact, one may be able to show, with large empirical statistical tests, that the artefacts caused by instabilities occur infrequently. As our test reveals, there is a
Figure 1: Perturbations \( r_j \) with \(|r_1| < |r_2| < |r_3|\) are added to the image \( x \). Upper row images (1)-(4): original image \( x \) and perturbations \( x + r_j \). Lower row images (1)-(4) reconstruction from \( A(x + r_j) \) using the Deep MRI network \( f \), where \( A \) is a subsampled Fourier transform (33\% subsampling), see Methods and SI for details. Rightmost column: reconstruction from \( Ax \). Lower row images (1)-(4): original image \( x \) and perturbations \( x + r_j \). Lower row images (1)-(4): original image \( x \) and perturbations \( x + r_j \). Lower row images (1)-(4): original image \( x \) and perturbations \( x + r_j \). Lower row images (1)-(4): original image \( x \) and perturbations \( x + r_j \).

A myriad of different artefacts that may occur, as a result of the instabilities, suggesting vast efforts needed to answer (ii). A detailed account is in the conclusion.

The instability test

The instability test is based on the three instability issues mentioned above. We consider instabilities with respect to changes in the sampling. The test checks whether the network is able to recover these small structural changes it is likely to recover other details of the same size. On the other hand, if the network fails on these basic changes, it is likely to fail on other details as well. The symbols can, of course, be specified if the network fails on these basic changes, it is likely to fail on other details as well. The symbols can, of course, be specified depending on the specific application. Our choice is merely for illustration.

**Important note:** When testing stability, both with respect to tiny perturbations and with respect to small structural changes, the test is always done in comparison with a state-of-the-art (SoA abbreviated) stable method in order to check that any instabilities produced by the neural network is due to the network itself and not because of ill-conditioning of the inverse problem. The state-of-the-art methods used are based on compressed sensing and sparse regularisation [25]–[27]. These methods often come with mathematical stability guarantees [28], and are hence suitable as benchmarks (see the Methods for details).

**Changing the number of samples in the sampling device (such as the MRI or CT scanner).** Typical state-of-the-art methods share a common quality; more samples imply better quality of the reconstruction. Given that deep learning neural networks in inverse problems are trained given a specific sampling pattern, the question is how robust is the trained network with respect to changes in the sampling. The test checks whether the quality of the reconstruction deteriorates with more samples. This is a crucial question in applications. For example the recent implementation of compressed sensing on Philips MRI machines allows the user to change the undersampling ratio for every scan. This means that if a network is trained on 25% subsampling, say, and suddenly the user changed the subsampling...
ratio to 35% one would want an improved recovery. If the quality deteriorates or stagnates with more samples, this means that one will have to produce networks trained for each and every combination of subsampling that the machine allows for. Finally, due to the other instability issues, every such network must individually be empirically statistically tested to detect whether the occurrence of instabilities is rare or not. It is not enough to test on only one neural network, as their instabilities may be completely different.

Testing the test

We test six deep learning neural networks selected based on their strong performance, wide range in architectures, difference in sampling patterns and subsampling ratios, as well as their differences in training data. The specific details about the architecture and the training data of the tested networks can be found in the supplementary information (SI).

Important note: The tests performed are not designed to test deep learning against state-of-the-art in terms of performance on specific images. The test is designed to detect the instability phenomenon. Hence, the comparison with state-of-the-art is only to verify that the instabilities are exclusive only to neural networks based on deep learning, and not due to an ill-conditioning of the problem itself. Moreover, as is clear from the images, in the unperturbed cases, the best performance varies between neural networks and state-of-the-art. The list of networks is as follows:

AUTOMAP (AM): This is a neural network for low resolution single coil MRI with 60% subsampling. The training set consists of brain images with added white noise to the Fourier samples.

DAGAN [11]: This network is for medium resolution single coil MRI with 20% subsampling, and is trained with a variety of brain images.

Deep MRI [10]: This neural network is for medium resolution single coil MRI with 33% subsampling. It is trained with detailed cardiac MR images.

Ell 50 [9]: Ell 50 is a network for CT or any Radon transform based inverse problem. It is trained on images containing solely ellipses (hence the name Ell 50). The number 50 refers to the number of lines used in the sampling in the sinogram.

Med 50 [9]: Med 50 has exactly the same architecture as Ell 50 and is used for CT, however, it is trained with medical images (hence the name Med 50) from the Mayo Clinic database. The number of lines used in the sampling from the sinogram is 50.

MRI-VN [12]: This network is for medium to high resolution parallel MRI with 15 coil elements and 15% subsampling. The training is done with a variety of knee images.

Stability with respect to tiny perturbations

Below follows the description of the test applied to some of the networks where we detect instabilities with respect to tiny perturbations.

Deep MRI: In this test we perturb the image $x$ with a sequence of perturbations $\{r_j\}_{j=1}^3$ with $|r_1| < |r_2| < |r_3|$ in order to simulate how the instabilities continuously transform the reconstructed image from a very high quality reconstruction to an almost unrecognisable distortion. This is illustrated in the lower row of Figure 1. Note that the perturbations are almost invisible to the human eye as demonstrated in the upper row of Figure 1. The $r_j$ perturbations are created by early stopping of the algorithm iterating to solve for the optimal worst case perturbation. The purpose of this experiment is to demonstrate how the gradual change in perturbation create artefacts that may be hard to verify as non-physical. Indeed, the worst case perturbation $r_3$ causes clearly a reconstruction that, in a real world situation, can be dismissed by a clinician as non-physical. However, for the
smallest $r_1$ we have a perturbation that is completely invisible to the human eye, yet it results in a reconstruction that is hard to dismiss as non-physical, and provides an incorrect representation of the actual image. Such examples could potentially lead to incorrect medical diagnosis. Note that state-of-the-art methods are not affected by the perturbation as demonstrated in the rightmost column of Figure 4. However, although this network is highly unstable with respect to tiny perturbations, it is highly stable with respect to small structural changes, see the 4th row of Figure 4.

**AUTOMAP:** This experiment is similar to the one above, however, in this case we add $\tilde{r}_1, \ldots, \tilde{r}_4$ to the measurements $y = Ax$, where $|\tilde{r}_1| < |\tilde{r}_2| < |\tilde{r}_3| < |\tilde{r}_4|$ and $A$ is a subsampled discrete Fourier transform. In order to illustrate how small the perturbations are we have visualised $|x + r_j|$ in the first row of Figure 4 where $y + \tilde{r}_j = A(x + r_j)$. To emphasise how the network reconstruction completely deforms the image we have, inspired by the second test on structural changes, added a small structural change in form of a heart that gradually disappears completely in the network reconstruction. This is demonstrated in the second row of Figure 4 and the third row of Figure 4 contains the reconstruction done by a state-of-the-art method. Note that the worst case perturbations are completely different to the ones failing the Deep MRI network. Hence, the artefacts are also completely different. These perturbations are white-noise like and the reconstructions from the network provide a similar impression. As this is a standard artefact in MRI, it is, however, not clear how to protect against the potential bad tiny noise. Indeed, a detail may be washed out, as shown in the experiment (note the heart inserted with slightly different intensities in the brain image), but the similarity between the standard artefact may make it difficult to judge that this is an untrustworthy image.

**MRI-VN:** In this case we add one perturbation $r_1$ to the image, where $r_1$ is produced by letting the algorithm searching for the worst perturbation run until it has converged. The results are shown in the first two columns of Figure 3 and the conclusion is the same for the MRI-VN net as for Deep MRI and AUTOMAP: perturbations barely visible to the human eye, even when zooming in, yield substantial misleading artefacts. Note also that the perturbation has no effect on the state-of-the-art-method.

**Med-50:** Here we add a perturbation $r_2$ that is also produced by running the algorithm until it has converged, and the results are visualised in the last two columns of Figure 3. The Med-50 network is moderately unstable with respect to tiny perturbations compared to Deep MRI, AUTOMAP and MRI-VN, however, severe artefacts are clearly seen. It is worth noting that this network is used for the Radon transform, which is, from a stability point of view, a more unstable operator than the Fourier transform when considering its inverse.

**Stability with respect to small structural changes**

Instabilities with respect to small structural changes are documented below.

**Ell-50:** This network provides a stark example of instability with respect to structural perturbation. Indeed, none of the de-
Figure 4: First four rows: Images $x_j$ plus structured perturbations $r_j$ (in the form of text and symbols) are reconstructed from measurements $y_j = A_j(x_j + r_j)$ with neural networks $f_j$ and state-of-the-art (SoA) methods. The networks are: $f_1 = \text{Ell 50}$, $f_2 = \text{DAGAN}$, $f_3 = \text{MRI-VN}$, $f_4 = \text{Deep MRI}$. The sampling modalities $A_j$ are: $A_1$ is a subsampled discrete Radon transform, $A_2$ is a subsampled discrete Fourier transform (single coil simulation), $A_3$ is a superposition of subsampled discrete Fourier transform (parallel MRI simulation with 15 coils elements), $A_4$ is a subsampled discrete Fourier transform. Note that Deep MRI has not been trained with images containing any of the letters or symbols used in the perturbation, yet it is completely stable with respect to the structural changes. The same is true for the AUTOMAP network (see first column of Figure 2). Last row: The figures show PSNR as a function of the subsampling rate for different networks. The red line indicates the subsampling ratio that the networks were trained for.
tails are visible in the reconstruction as documented in the first row of Figure 4.

**DAGAN:** This network is not as unstable as the Ell-50 network with respect to structural changes. However, as seen in the second row of Figure 4, the blurring of the structural details are substantial, and the instability is still critical.

**MRI-VN:** This is an example of a moderately unstable network when considering structural changes. Note, however, how the instability coincides with the lack of ability to reconstruct details in general. This is documented in the third row of Figure 4.

**Deep MRI:** To demonstrate how the stability with respect to small structured changes coincides with the ability to reconstruct details, we show how stable the Deep MRI network is. Observe also how well the details in the image are preserved in the fourth row of Figure 4. Here we have lowered the subsampling ratio to 25% even when the network is trained on 33% subsampling ratio. We want to point out that none of the symbols, nor any text, has been used in the training set.

### Stability with respect to more samples

Certain convolutional neural networks will allow for the flexibility of changing the amount of sampling. In our test cases all of the networks except AUTOMAP have this feature, and we report on the stability with respect to changes in the amount of samples below and in the last row of Figure 4.

**Ell 50/Med 50:** Ell 50 has the strongest and most fascinating decay in performance as a function of an increasing subsampling ratio. Med 50 is similar, however, with a less steep decline in reconstruction quality.

**DAGAN:** The reconstruction quality deteriorates with more samples similar to the Ell 50/Med 50 networks.

**VN-MRI:** This network provides reconstructions where the quality stagnates with more samples as opposed to the decay in performance witnessed in the other cases.

**Deep MRI:** This network is the only one that behaves aligned with standard state-of-the-art methods and provides better reconstructions when more samples are added.

### Conclusion

The main conclusion of this paper is that the new paradigm of inverse problems, through deep learning, typically yields unstable solutions. How is this related to the network architecture and training set? Moreover, does the subsampling pattern play a role? It is important, however, to observe (as in the 4th row of Figure 4 and 1st column of Figure 2) that there are perfectly stable networks with certain convolutional neural networks will allow for the flexibility of changing the amount of sampling.

(1) **Tiny perturbations lead to a myriad of different artefacts.** Different networks yield different artefacts and instabilities, and as Figures [1][2][3] reveal there is no common denominator. Moreover, the artefacts may be difficult to detect as non-physical. Thus, several key questions emerge: given a trained neural network, which types of artefacts may the network produce? How is the instability related to the network architecture, training set and also subsampling patterns?

(2) **Variety in failure of recovering structural changes.** There is a great variety in the instabilities with respect to structural changes as demonstrated in Figure [3], ranging from complete removal of details to more subtle distortions and blurring of the features. How is this related to the network architecture and training set? Moreover, does the subsampling pattern play a role? It is important, however, to observe (as in the 4th row of Figure 4 and 1st column of Figure 2) that there are perfectly stable networks with respect to structural changes, even when the training set does not contain any images with such details.

(3) **Networks must be retrained on any subsampling pattern.** The fact that more samples may cause the quality of reconstruction to either deteriorate or stagnate means that each network has to be retrained on every specific subsampling pattern, sub-sampling ratio and dimensions used. Hence, one may in practice need hundreds of different network to facilitate the many different combinations of dimensions, subsampling ratios and sampling patterns.

(4) **Universality - Instabilities regardless of architecture?** We have deliberately chosen networks with rather different architectures. Although our sample size of network is too small to conclude, it seems that instabilities may happen regardless of choice of architecture. However, different architectures may yield very different instabilities.

(5) **Rare events? - Empirical tests are needed.** As misleading artefacts caused by instabilities may or may not be rare events, a vast amount of empirical statistical testing is needed in order to establish safe use of deep learning methods in image reconstructions. However, such tests will have to be carried out on each specific network created on each specific subsampling pattern and dimension, yielding a potential vast research effort. Moreover, the search for effective neural networks for inverse problems must take into account the complex instability phenomena raised here, not only generalisation.

### References

[1] S. F. Gull and G. J. Daniell, ‘Image reconstruction from incomplete and noisy data’, *Nature*, vol. 272, no. 5655, p. 686, 1978.

[2] V. Studer, J. Bobin, M. Chahid, H. Moussavi, E. Candès and M. Dahan, ‘Compressive fluorescence microscopy for biological and hyperspectral imaging’, *Proc Natl Acad Sci USA*, vol. 109, no. 26, pp. 1679–1687, 2011.

[3] H. Engl, M. Hanke and A. Neubauer, *Regularization of inverse problems*, ser. Mathematics and Its Applications. Springer Netherlands, 1996.

[4] P. C. Hansen, ‘The l-curve and its use in the numerical treatment of inverse problems’, in *Computational inverse problems in electrocardiology, advances in computational bioengineering*, WIT Press, 2000, pp. 119–142.

[5] Y. LeCun, Y. Bengio and G. Hinton, ‘Deep learning’, *Nature*, vol. 521, no. 7553, p. 436, 2015.

[6] B. Zhu, J. Z. Liu, S. F. Cauley, B. R. Rosen and M. S. Rosen, ‘Image reconstruction by domain-transform manifold learning’, *Nature*, vol. 555, no. 7697, p. 487, Mar. 2018.

[7] R. Strack, ‘AI transforms image reconstruction’, *Nature Methods*, vol. 15, p. 309, Apr. 2018.

[8] M. T. McCann, K. H. Jin and M. Unser, ‘Convolutional neural networks for inverse problems in imaging: A review’, *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 85–95, Nov. 2017.

[9] K. H. Jin, M. T. McCann, E. Froustey and M. Unser, ‘Deep convolutional neural network for inverse problems in imaging’, *IEEE Transactions on Image Processing*, vol. 26, no. 9, pp. 4509–4522, 2017.

[10] 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*, Springer, 2017, pp. 647–658.

[11] G. Yang, S. Yu, H. Dong, G. Slabaugh, P. L. Dragotti, X. Ye, F. Liu, S. Arridge, J. Keegan, Y. Guo *et al.*, ‘DAGAN: Deep de-aliasing generative adversarial networks for fast compressed sensing MRI reconstruction’, *IEEE Transactions on Medical Imaging*, 2017.
[12] 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, vol. 79, no. 6, pp. 3055–3071, 2018.

[13] A. Lucas, M. Iliadis, R. Molina and A. K. Katsaggelos, ‘Using deep neural networks for inverse problems in imaging: Beyond analytical methods’, IEEE Signal Processing Magazine, vol. 35, no. 1, pp. 20–36, 2018.

[14] M. Mardani, Q. Sun, S. Vasanawala, V. Panyan, H. Montajemi, J. Pauly and D. Donoho, ‘Neural proximal gradient descent for compressive imaging’, in Advances in neural information processing systems, Curran Associates, Inc., 2018, pp. 9596–9606.

[15] M. Elad, ‘Deep, deep trouble - Deep learning’s impact on image processing, mathematics, and humanity’, SIAM News, vol. May, 2017.

[16] R. Girshick, J. Donahue, T. Darrell and J. Malik, ‘Rich feature hierarchies for accurate object detection and semantic segmentation’, in IEEE conference on computer vision and pattern recognition, Jun. 2014, pp. 580–587.

[17] A. Krizhevsky, I. Sutskever and G. E. Hinton, ‘Imagenet classification with deep convolutional neural networks’, in Advances in neural information processing systems, 2012, pp. 1097–1105.

[18] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba and A. Oliva, ‘Learning deep features for scene recognition using places database’, in Advances in neural information processing systems, 2014, pp. 487–495.

[19] K. He, X. Zhang, S. Ren and J. Sun, ‘Delving deep into rectifiers: Surpassing human-level performance on imagenet classification’, in IEEE international conference on computer vision, 2015, pp. 1026–1034.

[20] C. Kanbak, S.-M. Moosavi-Dezfooli and P. Frossard, ‘Geometric robustness of deep networks: Analysis and improvement’, in IEEE Conference on computer vision and pattern recognition, 2018, pp. 4441–4449.

[21] S. M. Moosavi-Dezfooli, A. Fawzi and P. Frossard, ‘Deepfool: A simple and accurate method to fool deep neural networks’, in IEEE Conference on computer vision and pattern recognition, 2016.

[22] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi and P. Frossard, ‘Universal adversarial perturbations’, in IEEE Conference on computer vision and pattern recognition, Jul. 2017, pp. 86–94.

[23] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow and R. Fergus, ‘Intriguing properties of neural networks’, in International conference on learning representations, 2014.

[24] A. Fawzi, S. M. Moosavi Dezfooli and P. Frossard, ‘The robustness of deep networks – A geometric perspective’, IEEE Signal Processing Magazine, vol. 34, no. 6, Nov. 2017.

[25] E. J. Candès, J. Romberg and T. Tao, ‘Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information’, IEEE Trans. Inform. Theory, vol. 52, no. 2, pp. 489–509, 2006.

[26] D. L. Donoho, ‘Compressed sensing’, IEEE Trans. Inform. Theory, vol. 52, no. 4, pp. 1289–1306, 2006.

[27] L. I. Rudin, S. Osher and E. Fatemi, ‘Nonlinear total variation based noise removal algorithms’, Physica D: Nonlinear Phenomena, vol. 60, no. 1-4, pp. 259–268, 1992.

[28] B. Adcock, A. C. Hansen, C. Poon and B. Roman, ‘Breaking the coherence barrier: A new theory for compressed sensing’, Forum of Mathematics, Sigma, vol. 1-84, 2017.

Supplementary Information is included in our submission.

Acknowledgments The authors would like to thank Dr. Cynthia McCoilough, the Mayo Clinic, the American Association of Physicists in Medicine, and the National Institute of Biomedical Imaging and Bioengineering for allowing the use of their data in the experiments. F. Renna acknowledges support from the European Union’s Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 655282 and the FCT grant SFRH/BPD/118714/2016. B. Adcock acknowledges support from the NSERC grant 611675. ACH thanks NVIDIA for a GPU grant in form of a Titan X Pascal and acknowledges support from a Royal Society University Research Fellowship, the UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/L003457/1 and a Leverhulme Prize 2017.

Correspondence Correspondence should be addressed to A. Hansen (ach70@cam.ac.uk).

Code and data All code available from [https://github.com/vegarant/Invfool](https://github.com/vegarant/Invfool). Parts of the data is not publicly available, but can be obtained by contacting the owner of the data.
Methods

The specific setups for deep learning and neural networks in inverse problems are typically rather specialised for each type of network. However, the main idea can be presented in a rather general way. Given an under-sampled inverse problem

$$Ax = y, \quad A \in \mathbb{C}^{m \times N}, \quad m < N$$

there is typically an easy linear way of approximating the solution $x$ from the measurement vector $y$. We will denote this linear operator by $H \in \mathbb{C}^{m \times N}$. In the MRI case, when $A$ is a subsampled discrete Fourier transform, often $H = A^*$. Note that in the MRI case $x$ is complex valued and we actually display the magnitude image $|x|$. An example is illustrated in Figure 2.

In the CT case $H$ could be $A^*$, however, this gives very poor results (as demonstrated in Figure 9), and thus $H$ is usually a discretisation of the filtered back projection (FBP). The problem is, as displayed in Figure 9 that the reconstruction $\hat{x} = Hy$ may still be rather poor. The philosophy of deep learning is quite simple; improve this reconstruction by using learning. In particular, inspired by deep learning in image denoising [29], given training images $\{x_1, \ldots, x_n\}$ and poor reconstructions $\{Hx_1, \ldots, Hx_n\}$, train a neural network $f : \mathbb{C}^N \rightarrow \mathbb{C}^N$ such that

$$\|f(Hx_j) - x_j\| \ll \|Hx_j - x_j\|, \quad j = 1, \ldots, n.$$  \hfill (2)

The hope is that (2) will hold for other images as well, not just the training examples $\{x_1, \ldots, x_n\}$.

The construction process of the neural network $f$ is typically done as follows. First one decides on a particular class (architecture) of neural networks $\mathcal{NN}$. Then one decides on a cost function $\text{Cost} : \mathcal{NN} \times \mathbb{C}^N \times \mathbb{C}^C \times \mathbb{C}^N \rightarrow \mathbb{R}$ and tries to solve the optimisation problem of finding

$$f \in \arg \min_{h \in \mathcal{NN}} \sum_{j=1}^n \text{Cost}(h, HAx_j, Ax_j, x_j).$$  \hfill (3)

The task of finding a good class $\mathcal{NN}$ and a good cost function $\text{Cost}$ is an engineering art on its own. All the networks we test, except for AUTOMAP, are trained with some form of a "warm start" in form of a linear operator $H$, however, AUTOMAP is based on directly solving the problem

$$f \in \arg \min_{h \in \mathcal{NN}} \sum_{j=1}^n \text{Cost}(h, Ax_j, x_j),$$  \hfill (4)

without any reference to the reconstructions $Hx_j$. We refer to SI for details regarding the training of the networks. Note that the instability phenomenon is independent of the choice of (3) or (4), and the operator $H$ may be viewed as just adding a layer to the network. Thus, we will in general talk about a network $f$ that takes the measurements $y = Ax$ as input.

Describing the test

Before describing the algorithm for creating the unstable perturbations, it is convenient to have a short review of the framework for establishing instabilities for neural networks for image classification. For a detailed review of such methods, see [24] and the references therein. The basic idea is as follows. Let $g : \mathbb{R}^d \rightarrow [0, 1]^C$ be an image classification network with $C$ different classes, so that $g(x)$ is a $C$-dimensional vector containing the probabilities associated to the different classes for a given input image $x$. Let $k_g : \mathbb{R}^d \rightarrow \{1, \ldots, C\}$ where $k_g(x) = \arg \max_{c \in \{1, \ldots, C\}} g(x)$ is the image classifier. For a given norm $\| \cdot \|$ on $\mathbb{R}^d$, we can then define the optimal, meaning smallest, unstable perturbation $r^* \in \mathbb{R}^d$, for an image $x \in \mathbb{R}^d$ as

$$r^*(x) \in \arg \min_r \|r\| \quad \text{subject to} \quad k_g(x + r) \neq k_g(x),$$  \hfill (5)

where we write $r^*(x)$ to indicate that this is a perturbation for the image $x$.

It is clear that one cannot apply the approach in (5) to the problem of finding instabilities of neural networks for the inverse problem. Indeed, (5) is designed for a decision problem a la "is there a cat in the image?". In inverse problems there is no decision problem but rather the following, slightly simplified issue:

Reconstruct $x$ from $y = Ax$, $A \in \mathbb{C}^{m \times N}$.  \hfill (6)

Thus, if we are given a neural network $f : \mathbb{C}^m \rightarrow \mathbb{C}^N$ designed to solve (6), and we want to search for instabilities imitating (5), we would end up with the problem of finding

$$\hat{r}(x) \in \arg \min_r \|r\| \quad \text{subject to} \quad \|f(y + Ar) - f(y)\| \geq \delta,$$  \hfill (7)

for some $\delta > 0$, where $y = Ax$ for some $x$. Note that (7) has a clear disadvantage in that it may be infeasible for different values of $\delta$. Hence, a slightly different, constrained Lasso inspired variant, may be worth considering:

$$\hat{r}(x) \in \arg \max_{r} \|f(y + Ar) - f(y)\| \quad \text{subject to} \quad \|r\| \leq \theta,$$  \hfill (8)

for some $\tau > 0$. In the case of (8) we do not have any issues regarding infeasibility. However, a third option could be an unconstrained Lasso inspired version of (8) given by

$$r^*(y) \in \arg \min_r \frac{1}{2} \|f(y + Ar) - f(y)\|^2_2 - \frac{\lambda}{2} \|r\|^2_2$$  \hfill (9)

(here we have specified the norm), where $\lambda > 0$. Note that (8) is not the only possibility. In particular, one could consider the more general setting

$$r^*(y) \in \arg \min_r \frac{1}{2} \|f(y + Ar) - p(x)\|^2_2 - \frac{\lambda}{2} \|r\|^2_2,$$  \hfill (10)

where $p : \mathbb{C}^N \rightarrow \mathbb{C}^N$. In this case $r^*$ will obviously depend on $p$, and the quality of the artefacts produced by $r^*$ may differ greatly as $p$ changes. Indeed, this is the motivation for allowing this extra variable. In this paper we focus on (10) and consider $p(x) = f(Ax)$ as in (9) and $p(x) = x$.

However, the first part of our test could indeed be carried out by a different optimisation problem. Moreover, we anticipate that there will be other methods for creating instabilities for neural networks for inverse problems that will be as reliable and diverse as what we present here. Note that (10) is set up to find perturbations in the image domain. We do this deliberately as this provides an easy way to compare the original image with a perturbed image and deduce whether the reconstruction of the perturbed image is acceptable/unacceptable. However, one could set up (10) so that the perturbation is in the sampling domain as well. In the following we describe the test in detail and the methodology.

Stability with respect to tiny perturbations

The neural network $f : \mathbb{C}^m \rightarrow \mathbb{C}^N$ is a non-linear function. In practice this makes the problem of finding a global maximum of the optimization problem in (10) impossible, even for small values of $m$ and $N$. In the following we will provide a method that aims at locating local maxima of (10) by using a gradient search method. In particular, given an image $x \in \mathbb{R}^N, A \in \mathbb{C}^{m \times N}$ and $y = Ax$ as in (1) let

$$Q_y^p(r) = \frac{1}{2} \|f(y + Ar) - p(x)\|^2_2 - \frac{\lambda}{2} \|r\|^2_2$$  \hfill (11)

be the objective function. A most natural method to solve (10) is gradient ascent with momentum. Thus, the method uses the
Figure 5: (Under-sampled MRI problem) $x$ is the original image (left). $\Omega$ is the set of indices corresponding to the subsampling (middle). $\hat{x} = A^* y$ (right) is the poorly reconstructed image, where $y = Ax$, $A = P_\Omega F$ and $F$ is the 2-dimensional discrete Fourier transform transform. Here $P_\Omega$ is the projection onto the span of $\{e_j\}_{j \in \Omega}$ where the $e_j$s denote the canonical basis.

Figure 6: (Under-sampled CT problem) $x$ is the original image (left). $\hat{x}_1$ (middle) is the very poorly reconstruction, where $\hat{x}_1 = A^* y$, $y = Ax$ and $A$ is a subsampled discrete Radon transform. $\hat{x}_2$ (right) is the poorly reconstructed image, where $\hat{x}_2 = B y$ and $B$ is a discrete filtered filtered back projection (FBP).

Gradient of $Q^\theta$ in conjunction with two parameters $\gamma > 0$ (the momentum) and $\eta > 0$ (the learning rate) in each step towards a local maximum.

Algorithm 1 Finding unstable perturbation for inverse problems

1: Input: Image: $x$, neural network: $f$, sampling matrix: $A$, maximum number of iterations: $M$.
2: Output: Perturbation $r_M$
3: Initialize: $y \leftarrow Ax$, $v \leftarrow 0$, $i \leftarrow 1$, $r_0 \sim \text{Unif}([0,1]^N)$, $0 < \lambda, \gamma, \eta, \tau$. Set $Q_{\theta}^y(r)$ as in Equation (11).
4: $r_0 \leftarrow \tau r_0$
5: while $i \leq M$ do
6: $v_{i+1} \leftarrow \gamma v_i + \eta \nabla_r Q_y(r_i)$
7: $r_{i+1} \leftarrow r_i + v_{i+1}$
8: $i \leftarrow i + 1$
9: return $r_M$

This means that there are three parameters $\lambda > 0$, $\gamma > 0$ and $\eta > 0$ to be set, and hence the perturbation $r$ found by the algorithm will depend on these. The complete algorithm is presented in Algorithm 1 where $r_0$ is initialised randomly. Note that the parameter $\tau$ used in Algorithm 1 is simply a scaling factor needed as the input images may have values in different ranges.

Note that for $u = y + Ar$, the gradient of $Q^\theta$ is given by

$$\nabla_r Q^\theta_y = A^* \nabla_u g(u) - \lambda r, \quad g(u) := \|f(u) - p(x)\|^2$$ (12)

where $\nabla_u g(u)$ can be computed efficiently using back propagation. Note also that at each iteration this gradient is left multiplied by the adjoint $A^*$.

Algorithm 2 Finding unstable perturbation for Radon problems

1: Input: Image: $x$, neural network: $f$, sampling matrix: $A$, FBP operator: $B$, maximum number of iterations: $M$.
2: Output: Perturbation $r_M$
3: Initialize: $y \leftarrow Ax$, $v \leftarrow 0$, $i \leftarrow 1$, $r_0 \sim \text{Unif}([0,1]^N)$, $0 < \lambda, \gamma, \eta, \tau$. Set $g(u)$ as in Equation (12).
4: $r_0 \leftarrow \tau r_0$
5: while $i \leq M$ do
6: $v_{i+1} \leftarrow \gamma v_i + \eta B \nabla_u g(y + Ar_i) - \lambda r_i$
7: $r_{i+1} \leftarrow r_i + v_{i+1}$
8: $i \leftarrow i + 1$
9: return $r_M$

Just as in the case when training neural networks using stochastic gradient descent with momentum, choosing the parameters $\gamma$ and $\eta$ is an art of engineering. We are in a similar situation with our algorithm, and the optimal choices of $\gamma, \eta$ are based on empirical testing. Such experimenting with parameters also motivates experimenting with other parts of the algorithm. For example, when considering Radon measurements, we found that setting the optimal values of $\gamma$ and $\eta$ could be rather diffi-
Figure 7: The experiment from Figure 2 is repeated, however, by using a different $p$ in Algorithm 1. In particular, Figure 2 is produced by using $p(x) = x$, however, this figure is produced by using $p(x) = f(Ax)$. Note the substantial difference in the quality of the artefacts in the AUTOMAP reconstruction compared to Figure 2.

cult. However, by replacing $A^*$ in (12) by $B \in \mathbb{R}^{N \times m}$ being a discretisation of a Filtered Back Projection (FBP), this problem could be overcome and we therefore use Algorithm 2 in the case of Radon samples.

It should be mentioned that there are different variations of discretisations of the filtered backprojection for Radon problems. The discretisation $B \in \mathbb{R}^{N \times m}$ used in our experiment is the one provided by MATLAB R2018b.

### State-of-the-art comparison method

All of our tests are done against state-of-the-art benchmark methods using established techniques based on sparse regularisation and compressed sensing [25]–[28].

There are many variations in the literature using X-lets and Total Variation (TV) techniques separately or in combination. Our main algorithm is based on the re-weighting technique suggested in [30]. This idea was refined in [31] and [32], by combining both X-lets (shearlets in this case) and TV. This is our main algorithm of choice used in this paper. We refer the reader to [32] for details, however, a short summary can be described as follows. The algorithm allows for both Fourier and Radon sampling, however, the current implementation only allows for single coil MRI in the Fourier case. The idea is to solve iteratively the problem

$$
\min_z \sum_{j=1}^J \lambda_j \|W_j \Psi_j z\|_1 + \text{TGV}_\alpha^2(z) \quad \text{subject to } Az = y,
$$

where the $\lambda_j$s are weights, $W_j$ is a diagonal weighting matrix, $\Psi_j$ is the $j$th subband in a shearlet transform [32], and $\text{TGV}_\alpha^2$, $\alpha = (\alpha_1, \alpha_2)$ is the second order Total Generalised Variation operator. The $\text{TGV}_\alpha^2$ operator consist of a first order term (TV) weighted by $\alpha_1$ and a second order (generalised) term weighted by $\alpha_2$. In each iteration step the weights $\lambda_j$ and weighting matrices $W_j$ are updated.

In particular, the minimisation problem is casted into an unconstrained formulation

$$
\min_z \sum_{j=1}^J \lambda_j \|W_j \Psi_j z\|_1 + \text{TGV}_\alpha^2(z) + \beta \|Az - y\|_2^2,
$$

and solved via split Bregman iterations. This means that the problem is decoupled into two portions, one accounting for the $\ell_1$-norm term and one for the $\ell_2$-norm term.

In particular, on denoting by $\Psi'$ a composite operator including (1) the effect of multi-scale X-lets transform in different levels including the weights $\lambda_j$, (2) the first order (TV) term of $\text{TGV}_\alpha^2$ and (3) the second order term of the same operator, and by adding a further splitting variable $d = \Psi' z$, it is possible to write the $k$-th split Bregman iteration as

$$
\begin{align*}
(z_{k+1}, d_{k+1}) &= \arg \min_{z,d} \|W d\|_1 + \frac{\mu}{2} \|Az - y\|_2^2 \\
&\quad + \frac{\beta}{2} \|d - \Psi' z - b_k\|_2^2, \\
b_{k+1} &= b_k + \Psi' z_{k+1} - d_{k+1}, \\
y_{k+1} &= y_k + A z_{k+1}.
\end{align*}
$$

During each iteration, the $(z,d)$-minimisation problem is solved using one or multiple non-linear block Gauss-Seidel iterations, which alternate between minimising $z$ and $d$. Also, in contrast with the re-weighting strategy originally presented in [30], the weights in $W$ are updated not only after convergence to the solution of the $\ell_1$ minimisation problem, but weight updates are incorporated in the split Bregman iterations.

In the above iterations we have allowed for a slight abuse of notation. We are using $\mu = (\mu_1, \mu_2, \mu_3)$ and split the sum $\|d - \Psi' z - b_k\|_2^2$ into three separate parts, depending on which of the terms of $\Psi'$ they come from, and weight each partial sum...
the measurements, based on the environmental conditions of the sensing problem. The corresponding measurement matrix is then written as
\[ A_{pf} = \begin{bmatrix} P_{\Omega}F & \ldots & 0 \\ \vdots & \ddots & \vdots \\ P_{\Omega}F & \ldots & 0 \end{bmatrix} \in \mathbb{C}^{m' \times N} \]

where \( m' = cm \). Notice that for this sampling operator we might have \( m' > N \), which means that the corresponding linear system may be overdetermined. Given an image \( x \in \mathbb{R}^N \) we let \( y = A_{pf}x \) and use the SPGL1 algorithm [33] for solving
\[
\text{minimize} \| z \|_1 \quad \text{subject to} \quad \| A_{pf} \Psi^{-1}z - y \|_2 \leq \delta,
\]

where \( \Psi \in \mathbb{R}^{N \times N} \) is the wavelet transform corresponding to the periodised Daubechies 2 wavelet with 3 levels. In all the experiments we set \( \delta = 0.01 \).

**Non-uniqueness of the test - parameter dependency**

Note that the test we provide can never become unique. Indeed, we choose to solve (10) with different choices of \( p \), however, (7) and (8) could also be viable alternatives. Moreover, all of these approaches depend on parameters \( \delta, \theta, \) and \( \lambda \) that have to be specified, and different values give different worst-case perturbations. In addition, Algorithm [1] and Algorithm [2] designed to solve (10) depend on the parameters \( \gamma, \eta, \) and \( \tau \). Moreover, note also that there is no built-in halting criteria in Algorithm [1] and Algorithm [2] but rather the parameter \( M \) controlling the number of iterations. Thus, the stability test can never become a unique test, but instead a collection of algorithms depending on different parameters. Hence, an appropriate use of the test means running Algorithm [1] and Algorithm [2] with different parameters. This is also what is done in this paper, however, only the results based on the final parameters are displayed in the figures. The final parameters chosen are listed in Table 1.

### Table 1: Summary of the different choices of parameters leading to the results reported in Figures 1-3 and 7.

| Neural Network   | \( \lambda \) | \( \gamma \) | \( \eta \) | \( \tau \) | \( p(x) \) |
|------------------|---------------|-------------|-----------|-----------|-----------|
| Deep MRI         | 0.001         | 0.9         | 0.01      | 0.01      | \( f(Ax) \) |
| AUTOMAP Fig. 2   | 0.1           | 0.9         | 0.001     | \( 10^{-6} \) | \( f(Ax) \) |
| AUTOMAP Fig. 7   | 0.1           | 0.9         | 0.001     | \( 10^{-6} \) | \( x \) |
| MRI-VN           | 1             | 0.9         | 0.0005    | 0.001     | \( f(Ax) \) |
| MED 50           | 20            | 0.9         | 0.0005    | 0.005     | \( f(Ax) \) |

Notice in particular that \( \mu_3 = \alpha_2 = 0 \), hence we are only using shearlets and a TV term as our regularizers. In all setups, we run the algorithm until convergence, i.e. between 50 and 500 iterations.

The above approach is used in all examples except for the tests using the MRI-VN network which is designed for parallel MRI. For this imaging modality we have the following reconstruction problem. Let \( \Omega \subseteq \{1, \ldots, N\} \) be \( |\Omega| = m \) and \( P_{\Omega} \in \mathbb{R}^{m \times N} \) be the projection operator onto the canonical basis i.e. \( P_{\Omega} x = \langle x \rangle_{\Omega} \). Let \( F \in \mathbb{C}^{N \times N} \) be the discrete Fourier transform (DFT) matrix. The Fourier sampling matrix can then be written as \( A_{pf} = P_{\Omega}F \), for a given sampling pattern \( \Omega \). In parallel Fourier imaging we receive information from multiple coils elements at the same time. This is modeled by introducing diagonal sensitivity matrices \( S_1, \ldots, S_c \in \mathbb{C}^{N \times N} \) which weight the measurements, based on the environmental conditions of the experiment.

This method has been used for reconstruction from Fourier and Radon measurements, using two different setups. For the case of Fourier measurements, discrete shearlets are generated with 3 scales and with directional parameters \([1 2 2] \) (see [31] and [32] for details). The optimisation parameters are set as follows:

- \((\mu_1, \mu_2, \mu_3) : (5000, 10, 20),\)
- \((\alpha_1, \alpha_2) : (1, 1),\)
- \(\beta : 10^5,\)
- \(c : 10^{-8},\)

Where \( c \) is a parameter which is added in the denominator, of the updating rule, for the weights \( W_{i} \), to avoid division by zero. Similarly, image reconstruction from Radon measurements are obtained by using shearlets with 4 scales and directional parameters \([0 0 1 1] \) and with the following parameter setup:

- \((\mu_1, \mu_2, \mu_3) : (500, 0, 0),\)
- \((\alpha_1, \alpha_2) : (1, 0),\)
- \(\beta : 50,\)
- \(c : 10^{-8},\)

The above approach is used in all examples except for the tests using the MRI-VN network which is designed for parallel MRI. For this imaging modality we have the following reconstruction problem. Let \( \Omega \subseteq \{1, \ldots, N\} \) be \( |\Omega| = m \) and \( P_{\Omega} \in \mathbb{R}^{m \times N} \) be the projection operator onto the canonical basis i.e. \( P_{\Omega} x = \langle x \rangle_{\Omega} \). Let \( F \in \mathbb{C}^{N \times N} \) be the discrete Fourier transform (DFT) matrix. The Fourier sampling matrix can then be written as \( A_{pf} = P_{\Omega}F \), for a given sampling pattern \( \Omega \). In parallel Fourier imaging we receive information from multiple coils elements at the same time. This is modeled by introducing diagonal sensitivity matrices \( S_1, \ldots, S_c \in \mathbb{C}^{N \times N} \) which weight the measurements, based on the environmental conditions of the experiment.
for the flexibility of the CNNs in our tests. The reason is that they all depend on the operator $H \in \mathbb{C}^{N \times m}$, as described in Equation 2, by considering it as a non-learnable first layer. As the $H$ is non-learnable, this allows for flexibility in our choice of $m$, since we know how to construct $H$ for various values of $m$.

In the last row of Figure 4 in the main manuscript we have varied the number of samples $m$ and measured the image quality of the networks reconstructions using the peak signal-noise-ratio (PSNR) between the magnitude images of the original and the reconstructed image. Figure 4 shows all of the networks, except the AUTOMAP network, which learns a mapping directly from the measurement domain without using a non-learned layer $H$. Below follows the description of each of the experiments visualised in the last row of Figure 4.

**Ell 50**: We created 25 sinograms of images containing ellipses similar to the data in the network’s training and test set. The sinograms were created with 1000 uniformly spaced angles (views) using the formula for the Radon transform of an ellipse. We then considered an acceleration factor $k \in \{2, \ldots, 30\}$, by subsampling every $k$-th line among the 1000 views. The FBP of the subsampled sinogram was given to the network and the PSNR of the reconstruction was computed against the FBP of all 1000 views.

**Med 50**: We used a test set, provided by the authors of [9], consisting of 25 CT images from the Mayo Clinic. These images were synthetically sampled, using the discrete Radon transform from MATLAB, at the same angles as the Ell 50 network. The subsampled sinograms were mapped back into the image domain using a FBP and reconstructed using the network. The PSNR values were computed with the original image as ground truth.

**DAGAN**: We used 20 MR images of brain tissue from the test set, and subsampled these images using the 1D Gaussian sampling patterns provided by the authors of [11]. These patterns have been generated for subsampling rates 1%, 5%, 10%, 20%, 30%, 40% and 50%.

**MRI-VN**: We used image data from the networks test set, and picked one image slice from 10 different patients. The image data was subsampled with uniformly spaced lines (center lines...
was always included), at subsampling rates 5%, 10%, 15%, 20%, 25%, 30%, 35%, and 50%. The PSNR was computed with the magnitude image of the fully sampled images as reference.

**Deep-MRI:** We used 30 image slices from one MRI scan, and subsampled each slice using lines sampled according to a Gaussian distribution. Extra caution was taken, so that all lines sampled at a low sampling rate, was included at higher sampling rates. We sampled with an acceleration rate $2, \ldots, 14$.

It should be noted that measuring image quality is a delicate issue. We point out that no comparison based on the last row of Figure 2 on the reconstruction quality should be made between the networks, as the PSNR is unfit to measure image quality between different types of images [34]. However, we are only interested in the change in PSNR, as a function of subsampling percentage, for each specific network.

### References

[9] K. H. Jin, M. T. McCann, E. Froustey and M. Unser, ‘Deep convolutional neural network for inverse problems in imaging’, *IEEE Transactions on Image Processing*, vol. 26, no. 9, pp. 4509–4522, 2017.

[11] G. Yang, S. Yu, H. Dong, G. Slabaugh, P. L. Dragotti, X. Ye, F. Liu, S. Arridge, J. Keegan, Y. Guo et al., ‘DAGAN: Deep de-aliasing generative adversarial networks for fast compressed sensing MRI reconstruction’, *IEEE Transactions on Medical Imaging*, 2017.

[24] A. Fawzi, S. M. Moosavi Dezfooli and P. Frossard, ‘The robustness of deep networks – A geometric perspective’, *IEEE Signal Processing Magazine*, vol. 34, no. 6, Nov. 2017.

[25] E. J. Candès, J. Romberg and T. Tao, ‘Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information’, *IEEE Trans. Inform. Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[26] D. L. Donoho, ‘Compressed sensing’, *IEEE Trans. Inform. Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[27] L. I. Rudin, S. Osher and E. Fatemi, ‘Nonlinear total variation based noise removal algorithms’, *Physica D: Nonlinear Phenomena*, vol. 60, no. 1–4, pp. 259–268, 1992.

[28] B. Adcock, A. C. Hansen, C. Poon and B. Roman, ‘Breaking the coherence barrier: A new theory for compressed sensing.’, *Forum of Mathematics, Sigma*, vol. 1-84, 2017.

[29] H. C. Burger, C. J. Schuler and S. Harmeling, ‘Image denoising: Can plain neural networks compete with BM3D?’, in *IEEE Conference on computer vision and pattern recognition*, Jun. 2012, pp. 2392–2399.

[30] E. J. Candès, M. B. Wakin and S. P. Boyd, ‘Enhancing sparsity by reweighted $\ell^1$ minimization’, *Journal of Fourier analysis and applications*, vol. 14, no. 5-6, pp. 877–905, 2008.

[31] J. Ma, M. Marz, S. Funk, J. Schulz-Menger, G. Kutyniok, T. Schaeffter and C. Kolbitsch, ‘Shearlet-based compressed sensing for fast 3D cardiac MR imaging using iterative reweighting’, *Physics in Medicine & Biology*, vol. 63, no. 23, p. 235004, 2018.

[32] J. Ma and M. März, ‘A multilevel based reweighting algorithm with joint regularizers for sparse recovery’, *ArXiv preprint arXiv:1604.06941*, 2016.

[33] E. van den Berg and M. P. Friedlander, ‘Probing the pareto frontier for basis pursuit solutions’, *SIAM Journal on Scientific Computing*, vol. 31, no. 2, pp. 890–912, 2008.

[34] Q. Huynh-Thu and M. Ghanbari, ‘Scope of validity of PSNR in image/video quality assessment’, *Electronics letters*, vol. 44, no. 13, pp. 800–801, 2008.
Supplementary Information

1 Overview

The Supplementary Information (SI) contains all the extra material on neural networks that is useful in order to understand and reproduce all the experiments done in the paper. In particular, the SI displays the variety of different architectures and training sets used in the various experiments. The neural networks considered are:

(i) **AUTOMAP** [6]: The AUTOMAP neural network we test is for low resolution single coil MRI with 60% subsampling. In the paper [6] one mentions 40% subsampling, but this apparent discrepancy is simply due to different interpretation of the word subsampling. We use the traditional meaning in sampling theory referring to x% subsampling as describing that the total amount of samples used are x% of full sampling. The actual network used in our experiment is trained by the authors of [6] and provided through private communication. The details of the architecture and training data are summarised in §2.1.

(ii) **DAGAN** [11]: This network is for medium resolution single coil MRI with 20% subsampling. The network weights are not available online, however, complete instructions on how to reproduce the network used in [11] are accessible. Moreover, the advice from the authors (through private communication) of [11] was to follow these guidelines to obtain the network. Thus, based on these instructions, we have retrained a network that reproduces the results in [11]. The details of the training data and architecture are summarised in §2.2.

(iii) **Deep MRI** [10]: This neural network is for medium resolution single coil MRI with 33% subsampling. The network used in our experiments is trained by the authors of [10], can be found online and we summarise the details on training data and architecture in §2.3.

(iv) **Ell 50** [9]: Ell 50 is a network for CT or any Radon transform based inverse problem. The number 50 refers to the number of lines used in the sampling in the sinogram. The training of the network is done by the authors of [9]. The network can be obtained online, and all the details can be found in §2.4.

(v) **Med 50** [9]: Med 50 has exactly the same architecture as Ell 50 and is used for CT, however the training is done on a different dataset. The network is trained by the authors of [9] and network weights have been obtained through private communication. The details are summarised in §2.4.

(vi) **MRI-VN** [12]: This network is for medium to high resolution parallel MRI with 15 coil elements and 15% subsampling. In order to show a variety of subsampling ratios we have trained this network on a smaller subsampling percentage than what the authors of [12] originally (25% and 33%) did in their paper. As we already have 33%, and 20%, we want a test on even lower subsampling rates. All the remaining parameters are kept as suggested in the code provided by the authors of [12], except for the subsampling ratios and batch size (due to memory limitations). All the details are documented in §2.5.

2 Deep learning and neural networks for inverse problems

The goal of deep learning in inverse problems is to learn a neural network \( f : \mathbb{C}^m \rightarrow \mathbb{C}^N \) taking the measurements \( y = Ax \) (where \( A \) is the matrix representing the sampling modality) as input and producing an approximation to \( x \) as its output. Many of the networks considered in this work are not directly applied to the measurements \( y \), but attempt to take advantage of the knowledge of the structure of the forward operator \( A \) by first applying a transformation to the measurements \( y \), in particular we obtain \( \tilde{x} = Hy \). The transformation \( H \) represents the adjoint operator when the forward operator is defined in the Fourier domain and a discretised filtered back projection (FBP) operator for the case of Radon measurements.

2.1 AUTOMAP

2.1.1 Network architecture

The AUTOMAP network [6] is proposed for image reconstruction from Radon measurements, spatial non-Cartesian Fourier sampling, misaligned Fourier sampling and undersampled Cartesian Fourier samples. In this work we have tested the network trained for image reconstruction from undersampled Cartesian Fourier samples. In contrast with the other networks considered in this work, the AUTOMAP network provides a direct mapping of the Fourier measurements to the image domain without applying the adjoint operator \( H \) as a first step.

The authors of [6] have not made their code publicly available, and the weights from their paper [6] had not been stored. However, they kindly agreed to retrain their network for us and save the weights. The network architecture they trained deviates
slightly in some of activation functions reported in their paper [6], however, the network was trained on the same data and sampling pattern. Below we describe the network architecture we received. The training parameters and data, are reported as in the paper [6].

The input of the AUTOMAP network, as described in [6] and in Figure 10 takes a complex $n \times n$ image of measurements as input. In the case of subsampling, one may interpret the $n \times n$ image as being zero padded in the coordinates that are not sampled. In the tests, $n = 128$, and in the actual implementation the input is represented by the complex measurement data $y \in \mathbb{C}^m$ with $m = 9855$ ($60\%$ of $n^2$) in this experiment. Such data is reshaped into a vector of length $2m$ with real entries before being fed into the network. The first two layers of the network a fully connected matrices of size $25000 \times 2m$ and $n^2 \times 25000$. The first fully connected layer is followed by a hyperbolic tangent activation function. The second fully connected layer is followed by a layer which subtracts the mean from the output of the second layer. The output is then reshaped into an $n \times n$ image.

Next follows two convolutional layers with filter size $5 \times 5$, 64 feature maps and stride $1 \times 1$. The first convolutional layer is followed by a hyperbolic tangent function, while the other is followed by a rectified linear unit (ReLU). Finally, the output layer deconvolves the 64 feature maps provided by the second convolutional layer with $7 \times 7$ filters with stride $1 \times 1$. The output of the network is an $n \times n$ matrix representing the image magnitudes.

### 2.1.2 Training parameters

The loss function used for training consisted of two terms, $L_{SE}$ and $L_{PEN}$. Here $L_{SE}$ is the $\ell_2$-norm of the difference between the ground truth magnitude image and the magnitude image predicted by the network. Similarly the $L_{PEN}$ is an $\ell_1$-penalty on the outputs of the activations following the second convolutional layer ($C2$). The total loss was then computed as

$$L_{\text{TOTAL}} = L_{SE} + \lambda L_{PEN}$$

with $\lambda = 0.0001$. The network is trained using the RMSProp algorithm (see for example [http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf](http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf) as referred to in [6]) with minibatch size 100, learning rate 0.00002, momentum 0, and decay 0.9. The number of training epochs is 100.

### 2.1.3 Training data

The training dataset consists of 50,000 images taken from 131 different subjects from the MGH-USC HCP public dataset [35]. For each image, the central $256 \times 256$ pixels were cropped and subsampled to a resolution of $128 \times 128$ pixels. Before training, the images were preprocessed by normalizing the entire dataset to a constant value defined by the maximum intensity of the dataset. Fourier data were obtained by subsampling the Cartesian $k$-space using a Poisson-disk sampling pattern with $60\%$ undersampling [36].

In order to increase the network robustness against translation, the following data augmentation scheme was applied. New images were created from each image in the training dataset by tiling together four reflections of the original image. Then, the so obtained $256 \times 256$ image was cropped to a random $128 \times 128$ selection. The routines used to implement the AUTOMAP network were written in TensorFlow [2].

### 2.2 DAGAN

#### 2.2.1 Network architecture

The DAGAN network was introduced in [11] to recover images from Fourier samples, with particular emphasis on MRI reconstruction applications. The DAGAN network assumes measurements $y = Ax$, where $A$ is a subsampled discrete Fourier transform. The network:

- Figure 10: The AUTOMAP architecture (figure from [6]).
input of the network is represented by the noisy magnitude image \( \tilde{x} = |Hy| \), which is obtained by direct inversion of the zero-filled Fourier data, in particular, \( H = A^* \).

The recovery algorithm presented in [11] is based on a conditional generative adversarial network (GAN) model, which consists of a generator network, used for the image reconstruction, and a discriminator network, measuring the quality of the reconstructed image. The generator network adopted in [11] has a U-net structure, similar to that used in [9], and its objective is to produce the recovered image. In [11] the authors propose two almost identical architectures, and train them with different loss functions. Below we will describe their "refinement" architecture trained with what is referred to as Pixel-Frequency-Perceptual-GAN-Refinement loss in the paper. The refined version is also our choice, as this architecture and training performed the best in the paper and in our tests as well. The network was not made publicly available, and based on advice from the authors of [11] we trained the network ourselves.

The architecture of the generator network, which is reported in Figure 11, contains 8 convolutional layers and 8 deconvolutional layers each followed by batch normalization (BN) [37]. The batch normalization layers after the convolutional layers are followed by leaky ReLU (lReLU) activations with slope equal to \( 0.2 \) for \( x < 0 \), while the batch normalization layers after the deconvolutions are followed by a ReLU activation. The generator network also contains skip connections, i.e., connections that copy the output of a layer directly to the input of a layer further down in the hierarchy. The skip connections are used to concatenate mirrored layers (see Figure 11). The filter kernels used for the convolutional and deconvolutional layers have size \( 4 \times 4 \) with stride \( 2 \times 2 \). The number of filters in each convolutional/deconvolutional layer increases/decreases according to Figure 11.

The last deconvolutional layer is followed by a hyperbolic tangent activation function. A global skip connection, adding the input to the network and the output from the hyperbolic tangent function, is then followed by a ramp function clipping the output values of the image to the range \([-1, 1]\). Adding this last skip connection means that the network is actually approximating the residual error between the network input \( \tilde{x} = |Hy| \) and the image of interest.

The generator network is trained jointly with a discriminator network, which aims to distinguish between the output of the generator network and ground truth images. For further information on this network, we refer to [11].

### 2.2.2 Training parameters

The loss function used to train the DAGAN network consists of four different terms. First, an image domain mean square error (MSE) loss, \( \mathcal{L}_{\text{MSE}} \), which accounts for the \( \ell_2 \) distance between the output of the generator network and the ground truth image. Second, a frequency domain MSE loss, \( \mathcal{L}_{\text{MSE}} \), which enforces consistency between the output of the generative network in the frequency domain and the acquired Fourier measurements. Third, a perceptual loss term, \( \mathcal{L}_{\text{VGG}} \), which is computed by using a pretrained VGG-16 described in [38]. In particular, the VGG-16 network was trained over the ImageNet database and the output of its conv4 layer was used to compute the loss term by considering the \( \ell_2 \)-norm of the difference between the VGG-16 output corresponding to the ground truth image and the generator network output. Finally, the fourth term, \( \mathcal{L}_{\text{GEN}} \) is computed using a cross entropy loss on the output of the discriminator network. Adding these four terms together gives us the loss

\[
\mathcal{L}_{\text{TOTAL}} = \alpha \mathcal{L}_{\text{MSE}} + \beta \mathcal{L}_{\text{MSE}} + \gamma \mathcal{L}_{\text{VGG}} + \tau \mathcal{L}_{\text{GEN}}, \quad \alpha, \beta, \gamma, \tau > 0.
\]
The same early stopping rule as given in their implementation. However, from the code, the DAGAN network was trained using data from a MICCAI 2013 grand challenge dataset.

### 2.2.3 Training data

The DAGAN network was trained using data from a MICCAI 2013 grand challenge dataset. We used the early stopping number with initial learning rate \( 0.0001 \), momentum 0.5, and minibatch size 25. The learning rate was halved every 5 epochs. We applied the same early stopping rule as given in their implementation. This is based on measuring the \( L_{\text{MSE}} \) loss between the training set and validation set. We used the early stopping number 10. In total this resulted in 15 epochs of training.

### 2.2 DeepMRINet

#### 2.2.1 Network architecture

The Deep MRI net is used to recover images from their subsampled Fourier measurements. Its architecture is built up as a cascade of neural networks, whose input is represented by the blurry image obtained by direct inversion of the measurements, i.e., \( \hat{x} = H y \).

The networks in the cascade are convolutional neural networks (CNN) designed as follows:

\[
CN_{i} \hat{x} = \hat{x} + C_{i} \rho(C_{i-1} \hat{x} + b_{i} + b_{i}^{c}) \cdots \rho(C_{i} \hat{x} + b_{i} + b_{i}^{c}) + b_{i}^{c},
\]

where \( \rho(z) = \max\{0, z\} \) is the ReLU activation function, whereas \( C_{i} \) and \( b_{i}^{c} \) represent trainable convolutional operators and biases, respectively, for the \( i \)th network. These networks are then tied together and interleaved with data consistency layers (DC), which have the objective to promote consistency between the reconstructed images and the Fourier measurements. The DC layers are defined as:

\[
DC_{\lambda} \hat{x}, y, \Omega \rightleftharpoons \frac{1}{1+\lambda} g_{\lambda}(F \hat{x}, y, \Omega), \quad \text{where} \quad g_{\lambda}(z, y, \Omega) = \begin{cases} z & k \notin \Omega \\ \frac{z + \lambda y_{k}}{1+\lambda} & k \in \Omega \end{cases}
\]

Here \( F \) represents the Fourier operator, and \( \Omega \) is the set of indices corresponding to the measurements acquired in the \( k \)-space. We point out that in the limit \( \lambda \to \infty \), the \( g_{\lambda} \) function simplifies to \( y_{k} \) if \( k \in \Omega \) and \( z_{k} \) otherwise.

In practice, a DC layer performs a weighted average of the Fourier coefficients of the image obtained as the output of a CNN in the cascade and the true samples \( y \). The parameter \( \lambda \) can either be trained or kept fixed. In [10], it is not specified whether \( \lambda \) is learned or not, however, from the code it is clear that \( \lambda \) is chosen to be \( \infty \).

The complete network can now be written as:

\[
f(y, \Omega) = DC_{\lambda_{n}}(\cdots DC_{\lambda}(CN_{n}((CN_{1}(H y)), y, \Omega) \cdots), y, \Omega),
\]

and its architecture is reported in Figure 12. In particular, the architecture used to produce the results in [10] and those reported in this paper contains 5 CNNs interleaved with 5 DC layers. Each CNN contains 5 convolutional layers, all with kernel size 3 × 3 and

https://github.com/nebulaV/DAGAN
http://masiweb.vuse.vanderbilt.edu/workshop2013/index.php/
http://tensorlayer.readthedocs.io
https://github.com/js3611/Deep-MRI-Reconstruction
parameters documented in the paper [10]. The DeepMRINet was trained using a loss function with two terms, $L_{\text{MSE}}$ and $L_{\text{WEIGHTS}}$. The $L_{\text{MSE}}$ term computed the mean squared error (MSE) between the true (complex valued) image and the predicted (complex valued) image, while the $L_{\text{WEIGHTS}}$ computed the $\ell_2$-norm of the weights. The loss function was then computed as

$$L_{\text{TOTAL}} = L_{\text{MSE}} + 10^{-7} L_{\text{WEIGHTS}}.$$  

The network weights were initialized using He initialization [19] and the Adam [39] optimizer was used for training. This optimizer takes as input a learning rate (step size) $\alpha$, and two exponential decay parameters $\beta_1$ and $\beta_2$ related to a momentum term. We refer to [39] for further explanations of these parameters. The network was trained with $\alpha = 10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and batch size equal 10.

### 2.3.3 Training data

The DeepMRINet was trained using data from five subjects from the MR dataset used in [41], which consists of 10 fully sampled short-axis cardiac cine scans. Each of these scans was then preprocessed, using the SENSE [42] software, into $256 \times 256$ valued frames of size $256 \times 256$. Synthetic MRI measurements were then obtained by sampling retrospectively the reconstructed images in k-space according to a Cartesian undersampling masks. During training, whereas a fixed undersampling rate of 33% was used, different undersampling masks were randomly generated in order to allow the network to recover images from measurements obtained with different undersampling masks. In particular, training images were fully sampled along the frequency-encoding direction but undersampled in the phase-encoding direction, according to the scheme described in [43] (center frequencies were always included in the subsampling patterns).

To prevent overfitting, data augmentation was performed by including rigid transformations of the considered images in the training datasets.

The code used to implement the DeepMRINet was written in Python using the Theano^\[^9\] and Lasagne^\[^9\] libraries.

### 2.4 FBPConvNet – The Ell 50 and Med 50 networks

The Ell 50 and Med 50 networks were proposed in [9] under the name FBPConvNet. The networks are trained to reconstruct images from Radon measurements. The networks have identical architecture and are trained using the same algorithm, with the same set of hyper parameters. The only difference between the training of the two networks, is the dataset they have been trained on. Below, we will describe the architecture and the training procedure of both the networks. We will then describe the datasets for the two networks in separate sections.

#### 2.4.1 Network architecture

The Ell 50 and Med 50 networks are trained for reconstructing $x$ from measurements $y = Ax$ where $A \in \mathbb{R}^{m \times N}$ is a Radon sampling operator, sampling 50 uniformly spaced radial lines. Rather than learning a mapping from $y$ to $x$ directly, the networks take advantage of a discrete filtered back projection $\tilde{x} = H y$ to $x$. The operator $H$ can be seen as a non-learnable first layer in the network.

The network contain several convolutional and deconvolutional layers, all of which (except the last) are followed by a batch normalization (BN) layer and a ReLU activation function. The (de)convolutional layers use filter size $3 \times 3$, stride $1 \times 1$ and a varying number of filters. We will not describe the full architecture in detail, as it can be seen in Figure 13, with the relevant number of filters, skip connections, max-poolings ($2 \times 2$) and concatenations. We do, however, point out that the network applies a final global skip connection, so that rather than learning a mapping from $\tilde{x}$ to $x$ the network is trying to learn the “noise” $x - \tilde{x}$.

#### 2.4.2 Training parameters

The network weights were provided by the authors of [9] and obtained based on the training procedure as described in their paper [9]. The loss function used to train the networks is the $\ell_2$ difference between the network output and the ground truth, and the networks are trained using the stochastic gradient descent algorithm with momentum. The learning rate varies from 0.01 to 0.001, whereas the momentum is set to 0.99, and the minibatch size is equal to 1. During training, gradients are clipped to the interval

^{9}\text{We used MATLABs \texttt{iradon} command to represent this operator}

^{10}\text{We used MATLABs \texttt{iradon} with linear interpolation and a 'Ram-Lak' filter to represent this operator}
I_{\text{max}} \] with \( I_{\text{max}} = 10^{-2} \), to prevent the divergence of the cost function. The networks are trained for 101 epochs, and the code used to implement the networks is written in MATLAB using the library MatConvNet\(^{12}\).

### 2.4.3 Ell 50 – Training data

The Ell 50 network is trained from the filtered back projection of 475 synthetic sinograms containing the Radon transform of ellipses of random intensity, size, and location. The dynamic range of the back projected images is adjusted so that image values are contained in the interval \([-500, 500]\). The Radon transform of an ellipse has an analytic formula, and hence this formula was used to create sinograms of such images using 1000 uniformly spaced lines (views). Measurement data are obtained by retaining 50 radial lines out of the 1000 views. The ground truth images were obtained by applying filtered back projection to fully sampled sinograms (i.e., 1000 radial lines). This approach is motivated by the fact that in applications, one will never have access to the underlying actual ground truth image. Data augmentation is also applied to the training data, by considering horizontal and vertical mirroring of the original images.

### 2.4.4 Med 50 – Training data

Med 50 is trained on synthetic images obtained from 475 real in-vivo CT images from the Low-dose Grand challenge competition database provided by the Mayo Clinic. The sinograms used for this training were synthetically generated from high quality CT-images using MATLAB radon command. The same approach as for the Ell 50 network was used, where one sampled 1000 view and used this as ground truth. The network was trained from 50 of these views.

### 2.5 MRI Variational Network (MRI-VN)

#### 2.5.1 Network architecture

The MRI Variational Network (MRI-VN) presented in [12] is designed to reconstruct images from undersampled MRI data, sampled using 15 coil elements. Thus, we use the sampling operator \( A = A_{\text{pf}} \) as described in the methods section, with \( c = 15 \).

The network structure is inspired by the unfolding of a variational minimization problem including a fidelity term and a regularization term defined according the Fields of Experts model [44]. In particular, each iteration of the corresponding Landweber method [45] corresponds to a layer of the resulting neural network. More specifically, the implementation considered in this work

\[ \text{https://www.vlfeat.org/matconvnet} \]
consists of $T = 10$ layers/iterations that can be expressed as follows:
\[
 u^{t+1} = u^t - (K^T \psi(K^t u^t) + \lambda^t A^*(Au^t - y)), \quad 0 \leq t < T
\]
where $u^0 = Hy$ is the complex image obtained by applying $H = A_{pfj}^*$. We will describe each of the remaining components of this network separately.

We start by noticing that the images $u^t \in \mathbb{C}^N$ (stacked as a vector in this simplified description) are complex valued, and can therefore described by its real and imaginary components $u^t_{re}$ and $u^t_{im}$, respectively. We will alternate between the representations.

The operator $K^T : \mathbb{C}^N \rightarrow \mathbb{R}^{N \times N_T}$ acts as follows on $u^t$:
\[
 K^t u^t = K^t_{re} u^t_{re} + K^t_{im} u^t_{im},
\]
where $K^t_{re}, K^t_{im} : \mathbb{R}^N \rightarrow \mathbb{R}^{N \times N_k}$ are learnable convolutional operators, with $N_k$ filters (channels), filter size $11 \times 11$ and stride $1 \times 1$. We will comment on the value of $N_k$ later.

The $\psi : \mathbb{R}^{N \times N_k} \rightarrow \mathbb{R}^{N \times N_k}$ is a non-linear activation function in the network. For each filter/channel, $i = 1, \ldots, N_k$ it applies the non-linear function
\[
 \phi^i(z) = \sum_{j=1}^{N_{w_{ij}}} w_{ij} \exp \left( -\frac{(z - \mu_j)^2}{2\sigma^2} \right),
\]
pointwise to each component $z$ in that channel. Here $\{w_{ij}\}_{i=1,j=1}^{N_k,N_w}$, with $N_{w_{ij}} = 31$, are weights which are learnt during the training phase. The nodes $\mu_j$ are non-learnable, and distributed in an equidistant manner on the interval $[-I_{max}, I_{max}]$, for a fixed value $I_{max}$, commented on below. The $\sigma$ is also non-learnable and equals $2I_{max}/N_{w_{ij}}$.

The operator $(K^t)^T : \mathbb{R}^{N \times N_k} \rightarrow \mathbb{C}^N$, maps $z \rightarrow (K^t_{re})^T z + (K^t_{im})^T z$, where $i$ is the imaginary unit, and $(K^t_{re})^T, (K^t_{im})^T$ are the transpose of $K^t_{re}, K^t_{im}$, respectively. The matrices $A, A^*$ are the matrix $A_{pfj}$ and its adjoint, while $\lambda^t$ is a learnable scalar. The remaining operations should be clear from Equation (13).

During training, each of the filters in $K^t_{re}$ and $K^t_{im}$ were restricted to have zero mean and have unit Euclidean norm. This was done to avoid a scaling problem with the weights $w_{ij}$.

To reproduce this network, we use the code published by the authors of [12]. Parts of this code uses slightly different parameters, than what was used in the original paper. In particular, the value $N_k = 24$ was chosen, rather than $N_k = 48$, as used in the paper. The value of $I_{max}$ was also changed from 150, in the paper, to 1 in the code. The change of the $I_{max}$ value is motivated by another change, also made in the published implementation, namely the scaling of the $k$-space values. In [12] the $k$-space volumes (with $n_{sl}$ slices) was normalized by the factor $\sqrt{n_{sl}10000}/||y_{volume}||_2$, whereas in the code this have been changed to scaling each $k$-space slice $y$ with $1/||Hy||_2$. This change has been made to make their implementation more streamlined. Whenever there has been a conflict between the two sources, we have chosen the version found in the code.

2.5.2 Training parameters

The MRI-VN network is trained using the $\ell_2$-norm as loss function. In particular, since MRI reconstruction are typically assessed through magnitude images, the error is evaluated by comparing smoothed version of magnitude images
\[
 |x|_\epsilon = \sqrt{(\text{Re}(x))^2 + (\text{Im}(x))^2 + \epsilon},
\]
with $\epsilon = 10^{-12}$. The network parameters that minimize the loss function are determined using the inertial incremental proximal gradient (IIPG) optimizer (see [12], [46] for details). Optimization is performed for 1000 epochs, with a step size of $10^{-3}$. Training data is arranged into minibatches of size 5. In the original paper, the batch size was set to 10, but due to memory limitations we had to adjust this.

2.5.3 Training data

The authors in [12] considered 5 datasets for different types of parallel MR imaging protocols, and trained one VN for each dataset. In this work, we have trained a VN for one of these protocols, namely Coronal Spin Density weighted with Fat Suppression. The training data consisted of knee images from 10 patients. From each patient we used 20 slices of the knee images, making up a total of 200 training images. The raw $k$-space data for each slice consisted of 15, $k$-space images of size $640 \times 368$, each with a precomputed sensitivity map. The sensitivity map was computed by the authors of [12], using ESPIRiT [47].

The raw data was obtained using a clinical 3T system (Siemens Magnetom Skyra) using an off-the-shelf 15-element knee coil. The raw data was subsampled retrospectively by zeroing out 85% of the $k$-space data. In [12] they test both a regular sampling scheme and a variable density pattern as proposed in [48]. In our work, we used a regular sampling scheme, where the 28 first central $k$-space lines were sampled, and the remaining lines were placed equidistantly in $k$-space. No data augmentation was used.

The code was implemented in Python with a custom made version of Tensorflow\textsuperscript{14} which was partly implemented in C++/CUDA with cuDNN support. All the code and data have been made available online by the authors of [12].

---

\textsuperscript{14} https://github.com/VLOGroup/mri-variationalnetwork
\textsuperscript{15} https://github.com/VLOGroup/tensorflow-icg
References

[6] B. Zhu, J. Z. Liu, S. F. Cauley, B. R. Rosen and M. S. Rosen, ‘Image reconstruction by domain-transform manifold learning’, *Nature*, vol. 555, no. 7697, p. 487, Mar. 2018.

[9] K. H. Jin, M. T. McCann, E. Froustey and M. Unser, ‘Deep convolutional neural network for inverse problems in imaging’, *IEEE Transactions on Image Processing*, vol. 26, no. 9, pp. 4509–4522, 2017.

[10] 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*, Springer, 2017, pp. 647–658.

[11] G. Yang, S. Yu, H. Dong, G. Slabaugh, P. L. Dragotti, X. Ye, F. Liu, S. Arridge, J. Keegan, Y. Guo et al., ‘DAGAN: Deep de-aliasing generative adversarial networks for fast compressed sensing MRI reconstruction’, *IEEE Transactions on Medical Imaging*, 2017.

[12] 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*, vol. 79, no. 6, pp. 3055–3071, 2018.

[19] K. He, X. Zhang, S. Ren and J. Sun, ‘Delving deep into rectifiers: Surpassing human-level performance on imagenet classification’, in *IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[35] Q. Fan, T. Witzel, A. Nummenmaa, K. R. Van Dijk, J. D. Van Horn, M. K. Drews, L. H. Somerville, M. A. Sheridan, R. M. Santillana, J. Snyder et al., ‘MGH–USC human connectome project datasets with ultra-high b-value diffusion MRI’, *Neuroimage*, vol. 124, pp. 1108–1114, 2016.

[36] M. Uecker, F. Ong, J. I. Tamir, D. Bahri, P. Virtue, J. Y. Cheng, T. Zhang and M. Lustig, ‘Berkeley advanced reconstruction toolbox’, in *Proc. Intl. Soc. Mag. Reson. Med.*, vol. 23, 2015, p. 2486.

[38] K. Simonyan and A. Zisserman, ‘Very deep convolutional networks for large-scale image recognition’, in *International conference on learning representations*, 2015.

[44] S. Roth and M. J. Black, ‘Fields of experts’, *International Journal of Computer Vision*, vol. 82, no. 2, p. 205, 2009.

[45] L. Landweber, ‘An iteration formula for fredholm integral equations of the first kind’, *American journal of mathematics*, vol. 73, no. 3, pp. 615–624, 1951.