Abstract

Convolutional Neural Networks (CNN) have recently seen tremendous success in various computer vision tasks. However, their application to problems with high dimensional input and output, such as high-resolution image segmentation or 3D medical imaging, has been limited by two factors. First, in the training stage, it is necessary to store network activations for back propagation. Second, in the inference stage, a few copies of the image are typically stored to be concatenated to other network states deeper in the network. In these settings, the memory requirements associated with storing activations can exceed what is feasible with current hardware. For the problem of image classification, reversible architectures have been proposed that allow one to recalculate activations in the backwards pass instead of storing them, however, such networks do not perform well for problems such as segmentation. Furthermore, currently only block reversible networks have been possible because pooling operations are not reversible.

Motivated by the propagation of signals over physical networks, that are governed by the hyperbolic Telegraph equation, in this work we introduce a fully conservative hyperbolic network for problems with high dimensional input and output. We introduce a coarsening operation that allows completely reversible CNNs by using the Discrete Wavelet Transform and its inverse to both coarsen and interpolate the network state and change the number of channels. This means that during training we do not need to store the activations from the forward pass, and can train arbitrarily deep or wide networks. Furthermore, our network has a much lower memory footprint for inference. We show that we are able to achieve results comparable to the state of the art in image classification, depth estimation, and semantic segmentation, with a much lower memory footprint.

1 Introduction

Deep Convolutional Neural Networks have recently solved some very challenging problems in computer vision ranging from image classification, segmentation, deblurring, shape from shading, and more [Krizhevsky et al., 2012, Ronneberger et al., 2015, Tao et al., 2018, Bengio, 2009, LeCun et al., 2015, Goodfellow et al., 2016, Hammerink et al., 2017, Avendi et al., 2016].

The recent success of neural networks has been attributed to three main factors. First, the massive amount of data that is being collected allows the training of complex models with hundreds of millions of parameters. Second, stochastic gradient descent has worked surprisingly well for such problems despite their non-convexity. Lastly, the ability to accelerate training through the use of graphical processing units (GPU’s) has made training complex models on the previously mentioned massive amounts of data feasible. Specifically related to the latter points, new and better neural networks architectures such as ResNet [He et al., 2016] and the UNet [Ronneberger et al., 2015] have been proposed that increase the networks stability, permitting the effective utilization of computational resources, obtaining a less nonlinear problem, and reducing training costs.
While such advances have been key to recent progress, we continue to face a number of challenges that current algorithms have yet to solve. As we push the scale at which deep learning is applied, computational time and memory requirements are quickly exceeding what is possible with current hardware. Memory is especially an issue with deep networks or when the size of the input and output are large. Examples include semantic segmentation and image deblurring or cases where 3D convolutions are required such as medical imaging, seismic applications, and video processing. In these cases, memory constraints are a limiting factor when dealing with large scale problems in reasonable computational time and resources. In the training phase, all previous states of the network, i.e. its activations, must be stored in order to compute the gradient. In the inference stage it is common for successful architectures to store various previous states [Ronneberger et al., 2015] or multiple scales [Zhao et al., 2017], again requiring the network to store activations. For these problems, the memory requirements are quickly over-reaching our ability to store the activations of the network.

Beyond just the obvious implications of working with large scale data, such as high resolution images, network depth and width are a significant factor in the memory footprint of a network. For fixed width networks, the depth of the network allows us to obtain more nonlinear models and obtain more expressive networks [Hanin, 2017]. Moreover, for problems in vision, networks depth plays another important role, since convolution is a local operation information propagates a fixed distance each layer. This implies that the output is determined with information available from a limited patch of the input, the size of which is determined by the number of layers and the width of the convolution kernel. The size of this patch is know as the receptive field [Luo et al., 2016]. By coarsening the image, the receptive field of the network grows and allows learning non-local relations. However, coarsening the image comes with the price of reducing the resolution of the final output. For problems such as classification this is not an issue as we desire a reduction in dimensionality, however for problems with high dimensional output, e.g. semantic segmentation, high resolution output is required. In these cases, the image is still coarsened to achieve the desired receptive field, however interpolation is used to regain resolution. A consequence of this is that fine image details or high frequency content is typically missing since the coarsening and subsequent interpolation is not conservative. This is the reason skip connections need to be used in the U-Net architecture [Ronneberger et al., 2015]. These skip connections require additional memory that can be significant and limit the ability of the network even in the inference stage, limiting the use of the networks on edge devices.

Width allows the network to learn more features, increasing the capability of the network to learn. However, the width comes at considerable price in terms of memory and computational effort. While it is clear that while network depth and width are a significant factor in the memory footprint, they are critical to the networks success.

In this work we introduce a new network architecture that addresses all the above difficulties. The network is fully conservative and reversible. Our formulation is motivated by the propagation of signals over physical networks. In physical networks, such as biological nets, signals can propagate in both directions. Indeed, a continuous formulation of a network involves the Telegraph equation (see [Zhou and Luo, 2018] and references within) which, upon discretization leads to a different formulation than the canonical ResNet. Similar to signal propagation in physical networks, our propagation is fully conservative. Conservation implies that, although our network has some similarities to the structure of a ResNet, when we coarsen the image we do not lose information and we can exactly recover any previous state of the network. This means that in the training phase we do not need to store all of the activations, and the memory footprint is independent of the network’s depth. Additionally, in the inference phase we do not need to store previous states of the network since our network does not loose information. Furthermore, the global signature of our network allows us to connect all pixels in the image, thus the network has the full image as its field of view. This allows us to learn local as well as global features.

The rest of the paper is structured as follows. In Section 2 we review background material on ResNets, reversible ResNets, and wavelets. We discuss how one can combine low resolution channels in order to obtain a single high resolution image and introduce our new network architecture. In Section 3 we experiment with our network on a number of problems and show its efficiency and finally, in Section 4 we summarize the paper.
2 Building Blocks of the Fully Hyperbolic Network

We start by reviewing the architecture of a canonical ResNet as presented in [He et al., 2016]. A ResNet is a particular architecture of a neural network. Let \( Y_0 \) represent the data or initial state. We view the data as a matrix. Each column (vector) in the matrix is a particular example. Let us focus on one example. For problems in imaging we can reorganize the example as a tensor which represents a vector image with \( n_c \) channels and a resolution of \( n_x \times n_y \).

The initial state \( Y_0 \) is transformed by the network using the following expression

\[
Y_{j+1} = W_j Y_j + f(Y_j, \theta_j)
\]  

(2.1)

Here, \( Y_j \) is the state at layer \( j \) and \( \theta_j \) are network parameters, e.g. convolution kernels, normalization parameters, and biases, and finally \( f \) is a nonlinear function. The matrix \( W_j \) is the identity matrix if the number of channels in \( Y_{j+1} \) is equal to the number of channels in \( Y_j \) and is chosen as zero otherwise. In this work we particularly use convolutional neural networks with \( \theta_j = \{K_j, b_j\} \), where \( K_j \) are convolution kernels and \( b_j \) are biases (for batch norm we do not use bias but use batch norm parameters). We choose a symmetric nonlinear function that can be expressed as

\[
f(Y_j, \theta_j) = -K_j^T f(K_j Y_j + b_j),
\]

which has been shown to have favorable theoretical and experimental properties [Ruthotto and Haber 2018].

For many, if not most applications, the number of channels grows with depth, while at the same time the state becomes coarser. Since this is done in a non-invertible way, the network loses information. To change resolution, coarsening or pooling is applied. We consider average pooling which corresponds to linear interpolation. While this network has been proven to work well on a number of applications, and in particular, image classification, it has a few drawbacks. In particular, the network is non-reversible. That means that given the state \( Y_j \) we can compute \( Y_{j+1} \) however, if we are given \( Y_{j+1} \) it is not trivial to compute \( Y_j \).

A reversible network has a few advantages. One of the most important is, that it does not require the storage of the activations when computing the gradients, allowing for arbitrary long networks independent of memory [Chang et al., 2018]. In order to obtain a reversible network it was proposed in [Chang et al., 2018] to use a hyperbolic network with the form

\[
Y_{j+1} = 2W_j Y_j - W_{j-1} Y_{j-1} + f(W_j Y_j, \theta_j)
\]  

(2.2)

Here, again, we use \( W_{j-1} \) and \( W_j \) when the resolution is changed and the number of channels are increased and set \( W_{j-1} = W_j = I \) when the resolution does not change. The network is clearly reversible since, as long as the number of channels are not changed and \( W = I \), given \( Y_{j+1} \) and \( Y_j \) it is straight forward to compute \( Y_{j-1} \). Reversibility allows for the computation of the gradient without the storage of the activations. This is done by stepping backwards and computing the states and their derivatives in the backward pass. This of course does not come for free as the computational cost for computing derivatives is doubled. However, for problems with deep networks, memory is typically the limiting factor in training and not the computational cost.

The origin of such a network can be traced to a nonlinear Telegraph equation [Zhou and Luo 2018] which can be written as

\[
\ddot{Y} = f(Y, \theta(t)).
\]  

(2.3)

The Telegraph equation describes the propagation of signals over physical and biological networks and therefore it is straightforward to extend its use to deep neural networks where signals are propagated over artificial networks. A leapfrog finite difference discretization of the second derivative reads

\[
\dot{Y} \approx \frac{1}{h^2} (Y_{j+1} - 2Y_j + Y_{j-1}).
\]

This leads to the proposed network (2.2) where the \( h^2 \) term is absorbed into the network parameters. The network is named hyperbolic as it imitates a nonlinear hyperbolic differential equation (2.3).

For hyperbolic networks, whenever \( W_{j-1} = W_{j-1} = I \), one can compute \( Y_{j-1} \) given \( Y_{j+1} \) and \( Y_j \). This implies that one need not store most of the activations and can recompute them on the backward pass. For large scale problems and deep networks, where the memory required to store
activations exceeds what is possible with current hardware, this may be the only way to compute gradients. A sketch of such a network is presented in Figure 1(a).

While we show in our numerical examples that for classification problems this network can perform well, we are interested in problems where both the input and the output are dense. For these problems we modify the network to obtain even more favorable properties. A simple extension for such problems is to add layers to the network that regain the original resolution [Shelhamer et al., 2017].

When such a network is applied to the data, going from the low resolution latent state to high resolution output requires interpolation. However, interpolating the low level features introduces interpolation artifacts and damps high frequencies. To regain high frequencies, the high resolution image is then supplemented with the previous states, such as at different scales [Zhao et al., 2017] or higher resolutions [Ronneberger et al., 2015]. For problems with many levels, this shortcut adds many degrees of freedom. Furthermore, for many problems skip connections, such as the ones used by the popular UNet, imprint the initial image over the final one. In addition to this, these architectures are not reversible and therefore their application to high dimensional data may be limited by the feasibility of storing activations during training. Finally, the high resolution states that are added via skip connections require additional memory in the inference step that could be prohibitive on edge devices.

Let us now introduce a fully hyperbolic network that overcomes the above issues. To this end we focus our attention on the layers where the resolution and number of channels are changed. Let \( \tilde{Y}_j \) be the \( j \)-th state obtained at layer \( j \). Our goal is to obtain a new state \( Y_j \) that has a coarser resolution with more channels. This of course can be done by simply convolving \( \tilde{Y}_j \) by a few kernels and then, coarsening each one of them which can be written as

\[
Y_j = \begin{pmatrix} P \\ P \\ \vdots \\ P \end{pmatrix} \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{pmatrix} \tilde{Y}_j
\]  

Here, \( A_i \) are convolution matrices and \( P \) are restriction matrices. The resulting state \( Y_j \) has now \( n \) channels, each channel has lower resolution compared to the original state.
Consider the matrix

\[ W_j = \begin{pmatrix} \text{PA}_1 \\ \text{PA}_2 \\ \vdots \\ \text{PA}_n \end{pmatrix} \] (2.5)

In general, the matrix, \( W_j \), is rectangular or not invertible. However, if we construct the matrix in a way that it is square and invertible then it is possible to decrease the resolution and add channels or to increase the resolution and reduce the number of channels without losing any information and, more importantly, without interpolation. This will enable us to start with a fine scale image, reduce its resolution while at the same time increasing the number of channels, and then increase its resolution and reduce the number of channels all without losing any information.

While it is possible to learn the matrices \( A_i \), this may add considerable complexity to the method. This is because, while it is straightforward to build \( W_j \) as square, it is not obvious how to enforce its invertibility. Furthermore, even if the matrix is invertible, inverting it may not be simple, making the process too expensive for practical purposes. Although it is difficult to learn an appropriate matrix \( W \), it is possible to choose one that possesses all the above qualities.

We propose to use the discrete wavelet transformation (DWT) as the invertible operator \( W \) that coarsens and increases the number of channels. The DWT is a linear transformation of a discrete grid function [Truchetet and Laligant, 2004], and is commonly used in image processing. In its simplest form at a single level, the DWT applies four filters to an image and decomposes it into four coarser images each containing distinct information. The important point is that the DWT is invertible, that is, it is possible to use the four low resolution images in order to explicitly and exactly reconstruct the fine resolution image. A simple example is plotted in Figure 2.

While there are many possible wavelets that can be used, here we chose to use the Haar wavelet. The Haar wavelet uses four simple filters and can be interpreted as simple convolution with subsampling (stride). The first is simply an averaging kernel that performs image sub-sampling. The final three are a vertical derivative, a horizontal derivative, and a diagonal derivative. In essence, the wavelet transform gives a recipe to either coarsen the image and increase the number of channels or reduce the number of channels and increase the resolution. This is exactly the property that is needed for our network. Another advantage of using the DWT is that it decreases the number of parameters of the network, since the parameters of the opening channels are pre-determined and need not be learned.

Using the above ingredients the hyperbolic network is simply (2.2) with \( W_j \) being the wavelet transform. For classification problems only a downward pass is needed and therefore the architecture (2.2) is sufficient. For problems where the final resolution is required to be similar to the original image we divide the network into two. A downward pass where the image is coarsened and the number of channels are increased and an upward pass where the image is refined and the number of channels are decreased.
\[
\text{Down} \quad Y_{j+1} = 2W_jY_j - W_{j-1}Y_{j-1} + f(W_jY_j, \theta_j) \quad j = 1 \ldots n
\]
\[
\text{Up} \quad Y_{j+1} = 2W^{-1}_jY_j - W^{-1}_{j-1}Y_{j-1} + f(W^{-1}_jY_j, \theta_j) \quad j = n + 1 \ldots 2n
\]

The final output has the same resolution as the input image and, since both parts of the network are reversible, the entire network is reversible. Another aspect of the network has a large receptive. This is achieved by coarsening the image and working with different resolutions. The architecture (2.2) is used for classification and the architecture (2.6) is used for segmentation and other tasks that require high resolution.

In certain applications, such as image classification, it may be desirable to remove information and encode the image into a lower dimensional latent space. In this case, while reversibility might still be desired to reduce the memory footprint, it may not be beneficial to have a fully conservative network. To this end, we consider a coarsening operation, which we call WavePool, to coarsen an image and reduce the dimensionality of the state while keeping as much information as possible. We begin by focusing on the pooling layers of a network where the current standard of using strided convolutions to simultaneously double the width and halve the spatial resolution, resulting in a net halving of the dimensionality of the state. This operation can be seen as first opening the state to twice the width, and then removing every other pixel. Instead of doing this we apply a bottleneck by first applying the DWT to the state which reduces the resolution by half and quadruples the number of channels. We then apply a convolution which reduces the number of channels by half, resulting in a net transformation that halves the resolution and doubles the width. While the outputs are the same dimension, because the first half of our bottle neck is fully conservative, we don’t lose any information until we apply a layer that restricts the width. In this case, we learn a group of parameters which combine the channels from the DWT instead of removing pixels from every channel. We show in [5] that local conservation, such as the form of pooling described above, could be beneficial even in the case we don’t require the entire network to be conservative.

### 3 Numerical Experiments with Hyperbolic Networks

In this section we experiment with our network on three different problems. The first is the CIFAR10 image classification problem, the second is the estimation of depth from images given by the NYU Depth V2 dataset and the third is an image segmentation problem using the CamVid dataset.

#### 3.1 Image Classification of CIFAR10

In this subsection we experiment with the CIFAR10 dataset [Krizhevsky and Hinton, 2009]. For this task we compare two of our proposed networks, the first of which is a fully hyperbolic network with 34 layers. Every initial image has three channels, and the network is coarsened 4 times such that the final resolution is \(2 \times 2\) pixels, at which point the state is vectorized such that the final number of features used for classification is 768. We also experiment with a a non-reversible 18 layer ResNet that uses WavePool layers instead of strided convolutions to coarsen the state. We compare the results from both of our networks with a standard ResNet18 using strided convolutions. Since the WavePool layer requires extra parameters, we increase the ResNet18’s final unit to a width of 663 such that they have the same number of learned parameters.

We train the HyperNet for 300 epochs using SGD with no momentum or weight decay. The initial learning rate is set to \(1e^{-1}\), and is reduced by a factor of 10 every 100 epochs. The WavePool and ResNet18 are trained for 350 epochs using SGD with a weight decay of 0.0005 and a momentum of 0.9. The initial learning rate is \(1e^{-1}\) and is reduced by a factor of 10 after 200 and 300 epochs. The results are is presented in Table [1]

The results show that the WavePool achieves a higher classification accuracy than the network with strided convolutions, which implies that the operation maintains more information. As expected, the results also demonstrate that for such a task a fully conservative network may not be beneficial. While our architecture can be used for image classification, it has many more advantages when considering problems such as depth estimation or semantic segmentation where the memory requirements in training and inference are much higher.
| Network Type                        | Number of Parameters | Validation Accuracy |
|------------------------------------|----------------------|---------------------|
| HyperNet34                         | 2,865,922            | 85.30               |
| ResNet18 (Strided Convolutions)    | 16,280,033           | 93.55               |
| ResNet18 (WavePool Coarsening)     | 16,262,826           | 93.97               |

Table 1: Comparison of validation results for our proposed networks on CIFAR10 using ResNet18 as a benchmark.

### 3.2 Depth Estimation from NYU Depth V2

In our next experiment we use our architecture on the NYU-Depth dataset. The data is a set of images recorded by both visible and depth cameras from a Microsoft Kinect. The data contain four different scenes and our goal network is to use the visible images in order to predict the depth images. Rather than using all images, we train a 40 block hyperbolic network on a subset of classroom images. The first 20 blocks are from the downward pass and the remaining 20 are from an upward pass. We coarsen the images every five steps, resulting in four coarsening steps followed by four reconstruction steps. An example of one of the images its depth map and the recovered depth map using our network is plotted in Figure 3.

The HyperNet has roughly 56M parameters. An equivalent ResNet has close to 62M parameters due to the opening layers and an equivalent U-Net has close to 66M parameters. We used 500 epochs to fit the data. The initial L2 misfit is 2.3 and we are able to reduce this misfit to 0.01 in 150 epochs. The results on a validation image is presented in Figure 3 where good agreement of our prediction is achieved. Quantitatively, the results are similar to those obtained in [Riegler et al., 2015] where a non-reversible network was used and training requires significantly more memory.

![Figure 3: An example from the validation set of the NYU Depth V2 dataset and our hyperbolic nets prediction.](image)

### 3.3 Semantic Segmentation from CamVid

Semantic image segmentation is one of the prime examples where the benefits of the proposed reversible and conservative network matter. To this end, we test our method on the CamVid dataset. The data consist of a few video segments that are coarsely sampled in time, resulting in RGB images and corresponding 12 class label images. See Figure 4 for an example of an image and corresponding true segmentation. We train and test on the dataset as prepared by a standard data loader[^1] All data and labels are 360 × 480 pixels and there are 367 training, 233 testing, and 101 evaluation pairs. Specifically, we use the network structure (2.6) with 24 blocks; 12 in the downward-pass (coarsening resolution) and 12 in the reconstruction part. After every three network blocks, we coarsen the image and simultaneously increase the number of channels. The coarsest resolution of each image is 45 × 60 pixels and we have 768 channels before refining the image. Training starts at a learning rate of 0.1 for the Adam optimization algorithm for 50 epochs with a batch size equal to one. We use average the cross entropy loss of the output and apply a mean frequency class weighting to balance the classes.

The results are summarized in Table 2. As can be observed, the results are close to the state of the art results reported in [Badrinarayanan et al., 2015]. We note that our model has not been pre-trained, where as many of the results we compare with use pre-trained weights for the encoder. Compared to the SegNet-Bilinear, which is most similar to our proposed model as it doesn’t use learned interpolation, we see that we achieve slightly higher global pixel accuracy. While we under-perform

[^1]: https://github.com/meetshah1995/pytorch-semseg
in terms of the average class accuracy, it has been noted in [Badrinarayanan et al., 2015] that global average is a better indicator of a smooth segmentation.

| Model               | Global Accuracy | Class Average |
|---------------------|-----------------|---------------|
| HyperNet            | 79.7            | 52.8          |
| SegNet-Bilinear [Badrinarayanan et al., 2015] | 77.9            | 61.1          |
| SegNet-Basic [Badrinarayanan et al., 2015] | **82.7**        | 62.0          |
| FCN-Basic [Badrinarayanan et al., 2015] | 81.7            | **62.4**      |

Table 2: Results of the HyperNet (2.6) and various benchmark for the CamVid test set. Here we present the results obtained in [Badrinarayanan et al., 2015] using median frequency class balancing, where as our results were obtained with mean frequency class balancing.

4 Conclusions and summary

In this work we have introduced a new architecture for deep neural networks. The architecture is motivated by the propagation of signals over physical networks where hyperbolic equations are used to describe the behavior of signal propagation. The network can be interpreted as a leapfrog discretization of the nonlinear Telegraph equation. The equation and its corresponding discretization are conservative, which implies that the network propagates the energy of the initial condition throughout all layers without decay. Similar to other networks, in order to obtain non-local behavior coarsening is used and, at the same time, the number of channels is increased. In order to coarsen the image conservatively, we use the discrete wavelet transform. Such a transform has a natural property that it increases the number of channels while reducing the resolution, while conserving all the information in the image due its invertibility.

There are a number of advantages of hyperbolic networks and to our fully conservative networks in particular. Information retention implies that there are no vanishing or exploding gradients to the network, which can make training easier. However, the biggest advantage of such a network is in its reversibility. The network does not require the storage of activations order to compute derivatives. This enables the training of very deep networks even for problems where the input and output size is very large, with only doubling the computational cost of the propagation. The coarsening and refinement of the images in our network does not require interpolation. Instead, coarsening is done by the wavelet transform and refinement is done by its inverse. As a result, information is not damped during the coarsening process, and we can perform exact image refinement without the need of interpolation that introduces interpolation artifacts. The ability to change resolution without damping also allows us to drop the skip connections that are commonly used in the UNet architecture, and this implies that we are able to reduce the memory footprint of our network in inference, which can be very important for edge computing.

While information retention is generally a good property it may not be needed for all problems. Some problems, such as image classification, may favor some information loss and for these cases our network may not be optimal. Nonetheless, we believe that for large scale problems and for problems in 3D our network can be a key in efficient training and inference.
References

M. Avendi, A. Kheradvar, and H. Jafarkhani. A combined deep-learning and deformable-model approach to fully automatic segmentation of the left ventricle in cardiac mri. *Medical Image Analysis*, 30:108 – 119, 2016.

V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561, 2015. URL [http://arxiv.org/abs/1511.00561](http://arxiv.org/abs/1511.00561).

Y. Bengio. Learning deep architectures for AI. *Foundations and Trends® in Machine Learning*, 2(1):1–127, 2009.

B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert, and E. Holtham. Reversible architectures for arbitrarily deep residual neural networks. In *AAAI Conference on AI*, 2018.

A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse. The reversible residual network: Backpropagation without storing activations. In *Adv Neural Inf Process Syst*, pages 2211–2221, 2017.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

K. Hammernik, T. Klatzer, E. Kobler, M. P. Recht, D. K. Sodickson, T. Pock, and F. Knoll. Learning a variational network for reconstruction of accelerated mri data. *Magnetic Resonance in Medicine*, 79(6):3055–3071, Nov 2017. ISSN 0740-3194. doi: 10.1002/mrm.26977. URL [http://dx.doi.org/10.1002/mrm.26977](http://dx.doi.org/10.1002/mrm.26977).

B. Hanin. Universal function approximation by deep neural nets with bounded width and relu activations. *arXiv preprint arXiv:1708.02691v3*, 2017.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeeseer, 2009.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

W. Luo, Y. Li, R. Urtasun, and R. Zemel. Understanding the effective receptive field in deep convolutional neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4898–4906. Curran Associates, Inc., 2016.

G. Riegler, R. Ranftl, M. Rüther, T. Pock, and H. Bischof. Depth restoration via joint training of a global regression model and cnns. *BMVC*, 58, 2015.

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL [http://arxiv.org/abs/1505.04597](http://arxiv.org/abs/1505.04597).

L. Ruthotto and E. Haber. Deep neural networks motivated by partial differential equations. *arXiv preprint arXiv:1804.04272*, 2018.

E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):640–651, 2017. doi: 10.1109/TPAMI.2016.2572683. URL [https://doi.org/10.1109/TPAMI.2016.2572683](https://doi.org/10.1109/TPAMI.2016.2572683).

X. Tao, H. Gao, Y. Wang, X. Shen, J. Wang, and J. Jia. Scale-recurrent network for deep image deblurring. *CoRR*, abs/1802.01770, 2018. URL [http://arxiv.org/abs/1802.01770](http://arxiv.org/abs/1802.01770).

F. Truchetet and O. Laligant. Wavelets in industrial applications: a review. *Wavelet Applications in Industrial Processing II*, 5607, 2004. doi: 10.1117/12.580395. URL [https://doi.org/10.1117/12.580395](https://doi.org/10.1117/12.580395).
H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. doi: 10.1109/cvpr.2017.660. URL http://dx.doi.org/10.1109/CVPR.2017.660

Y. Zhou and Z. Luo. A crank-nicolson collocation spectral method for the two-dimensional telegraph equations. *Journal of Inequalities and Applications*, 2018:137, 2018. URL https://journalofinequalitiesandapplications.springeropen.com/articles/10.1186/s13660-018-1728-5