IMEXnet - A Forward Stable Deep Neural Network

Eldad Haber[*, 1, 2] Keegan Lensink[*, 1, 2] Eran Treister[3] Lars Ruthotto[4]

Abstract

Deep convolutional neural networks have revolutionized many machine learning and computer vision tasks, however, some remaining key challenges limit their wider use. These challenges include improving the network's robustness to perturbations of the input image and the limited "field of view" of convolution operators. We introduce the IMEXnet that addresses these challenges by adapting semi-implicit methods for partial differential equations. Compared to similar explicit networks, such as residual networks, our network is more stable, which has recently shown to reduce the sensitivity to small changes in the input features and improve generalization. The addition of an implicit step connects all pixels in each channel of the image and therefore addresses the field of view problem while still being comparable to standard convolutions in terms of the number of parameters and computational complexity. We also present a new dataset for semantic segmentation and demonstrate the effectiveness of our architecture using the NYU Depth dataset.

1. Introduction

Convolutional Neural Networks (CNN) have revolutionized many machine learning and vision tasks such as image classification, segmentation, denoising and deblurring (see (Bengio, 2009; LeCun et al., 2015; Goodfellow et al., 2016; Hammernik et al., 2017; Avendi et al., 2016) and references within).

Many different architectures have been proposed and often tailored to specific tasks. In recent years, residual networks (ResNets) have shown to be successful in dealing with many different tasks (Gomez et al., 2017; He et al., 2016b;a; Li et al., 2018). ResNets have many practical advantages (e.g., ease of training and possible reversibility (He, 2017; Chang et al., 2018)) and are also supported by mathematical theory due to their link to ordinary differential equations (Weinan, 2017; Chen et al., 2018; Haber & Ruthotto, 2017; Ma et al., 2018; Lu et al., 2018) and, when dealing with imaging data, partial differential equations (Ruthotto & Haber, 2018).

The connection between ResNets and differential equations has highlighted the issue of forward stability of the network. Roughly speaking, a network is forward stable when it does not amplify perturbations of the input features due to, for example, noise or adversarial attacks. The examples in (Haber & Ruthotto, 2017; Chen et al., 2018) also suggest that stable networks train faster and generalize better. Keeping a network stable requires attention and can be challenging to control (Haber & Ruthotto, 2017; Ciccone et al., 2018; Ma et al., 2018).

While it is possible to apply ResNets to many different vision problems, one should differentiate between problems that have a small-dimensional output and problems that have a large-dimensional output.

In a small-dimensional output problem, the image is reduced in dimension to a small vector. For example, in image classification, a small-dimensional vector in $\mathbb{R}^n$ represents the likelihood of the image to be one of n different classes. In this case, the network is being used for dimensionality reduction. For these problems, the image is typically coarsened a number of times before a prediction is made. The coarsening of the image and the use of convolutions allows far-away pixels to communicate, utilizing long-range correlations in the image.

In a large-dimensional output problem, the network generates several different output images and, most commonly, each output image has at least as many pixels as the input image. An example of this is image segmentation, where each output image represents the probability of each pixel belonging to a certain class. In depth estimation, image denoising, and image deblurring the output image has the same dimension as the input image, and in many cases contains high spatial frequencies that are absent in the original image.

*Equal contribution [1]Department of Earth, Ocean and Atmospheric Sciences, University of British Columbia, Vancouver, Canada [2]Xtract AI, Vancouver, Canada [3]Department of Computer Science, Ben Gurion University of the Negev, Be'er Sheva, Israel [4]Departments of Mathematics and Computer Science, Emory University, Atlanta, GA, USA. Correspondence to: Eldad Haber <ehaber@eoas.ubc.ca>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

For these problems, a straightforward extension of the ResNet architecture may not be sufficient because it is unfavorable to coarsen the image and remove high-frequency spatial information. Without coarsening, one is required to work with the original image resolution and compelled to use very deep networks with sufficiently many convolutional layers to model interactions between far away pixels.

This is known as the *field-of-view* problem and has been studied in (Luo et al., 2016) and it is also common in other image processing techniques such as Total Variation image denoising (Rudin et al., 1992) and anisotropic diffusion (Black et al., 1998; Weickert, 1998). Current CNN architectures are limited to either using additional convolutional layers, coarsening, or a combination of both in order to increase their field-of-view. While the receptive field is in general not a problem for small-dimensional output problems such as classification, where coarsening is already being used for dimensionality reduction, it remains a problem for many large-dimensional output problems. In large-dimensional output problems, image coarsening can be done as a part of the network; however, image interpolation is needed to return to the original image size and provide dense output. This leads to a different architecture, e.g., the U-net (Ronneberger et al., 2015), which is more complicated than simple ResNets, is less well-understood theoretically, and usually requires many more parameters.

In this paper, we introduce the implicit-explicit network, IMEXnet, and apply it to high-dimensional output problems. Our network is based on simple but effective changes to the popular ResNet architecture and is motivated by semi-implicit techniques for partial differential equations. Such techniques are used for time-dependent problems arising in computational fluid dynamics and imaging when global information is passed within a small number of iterations or time steps (Kadioglu et al., 2011; Schönlieb & Bertozzi, 2011). These techniques address both the stability and the field-of-view issues while adding a negligible number of parameters and computational complexity. We differentiate between large and small-dimensional output problems because the proposed methods ability to accelerate the propagation of information is most advantageous for high-dimensional output problems.

The paper is structured as follows. In Section 2, we derive the IMEXnet and explore its theoretical properties. In Section 3, we show that our method can be implemented efficiently in existing machine learning packages and demonstrate that it adds only a marginal cost to simple ResNets in terms of the number of operations and required memory. In Section 4, we conduct numerical experiments on a synthetic dataset that is constructed to demonstrate the advantages and limitations of the method, as well as on the NYU depth dataset. We summarize the paper in Section 5.

2. Semi-implicit Neural Networks

We first briefly review residual neural networks (ResNets) and outline their limitation in terms of stability and field-of-view problem. We then derive the basic idea behind our new implicit-explicit IMEXnet as a modification of ResNets. Finally, we analyze the improved stability of our method and discuss its advantages and disadvantages.

2.1. Residual Networks

Our starting point is the j-th layer of a ResNet that propagates the features $\mathbf{Y}_j$ as follows

$$\mathbf{Y}_{j+1} = \mathbf{Y}_j + h\, f(\mathbf{Y}_j, \boldsymbol{\theta}_j). \tag{1}$$

Here, $\mathbf{Y}_{j+1}$ are the output features of the j-th layer, $\boldsymbol{\theta}_j$ are the parameters that this layer depends on, and f is a nonlinear function. In imaging problems, the parameters $\boldsymbol{\theta}_j$ typically contain convolution kernels as well as scaling and bias parameters for batch or instance normalization. Here $h > 0$ is a step size that is typically set to 1. In particular, we explore the structure proposed in (He et al., 2016a) that has the form

$$f(\mathbf{Y}, \mathbf{K}_1, \mathbf{K}_2, \alpha, \beta) = \mathbf{K}_2 \sigma(N_{\alpha,\beta}(\mathbf{K}_1 \mathbf{Y})). \tag{2}$$

Here $\mathbf{K}_1$ and $\mathbf{K}_2$ are built using (typically 3×3) convolution operators, $N_{\alpha,\beta}(\cdot)$ is a normalization layer that depends on the parameters α and β, and σ is a nonlinear activation function that is applied element-wise.

It is interesting to evaluate the action of this network on some image $\mathbf{Y}_0$. At every layer of this network, each pixel communicates with a 5×5 patch around itself. Therefore, for high-resolution images, many layers are needed in order to propagate information from one side of the image to the other. This is demonstrated in the top two rows of Figure 1 where two delta functions are propagated through a multi-layer ResNet with 20 layers and $h = 1$ and a ResNet with 5 layers and $h = 5$. Comparing the output features it is apparent that the second (4 times less expensive) network is unable to propagate information over large distances.

The above discussion highlights the *field-of-view* problem, i.e., that many convolutional layers are needed to model nonlocal interactions between distant pixels. For problems such as image classification, the image is typically coarsened using pooling layers placed between ResNet blocks. Pooling makes each pixel encompass a larger area and therefore allows information to travel larger distances in the same number of convolution steps. Coarsening is not applicable in tasks that require a high-dimensional output, as it leads to the loss of important local information. In these cases, many layers are needed in order to pass information between different parts of the image. This leads to very high computational cost and storage.

At this point, it is worthwhile recalling the differential equation interpretation given to ResNets proposed in (Weinan, 2017; Haber & Ruthotto, 2017). In this interpretation the ResNet step (1) is viewed as a forward Euler discretization of the ordinary differential equation (ODE)

$$\dot{\mathbf{Y}}(t) = f(\mathbf{Y}(t), \boldsymbol{\theta}(t)), \quad \mathbf{Y}(0) = \mathbf{Y}_0. \tag{3}$$

Here, the features $\mathbf{Y}(t)$ and weights $\boldsymbol{\theta}(t)$ are continuous functions in the (artificial) time that corresponds with the depth of the network. While it is possible to discretize the system using the forward Euler method (resulting in (1)), many other methods can be used. In particular, in (Haber & Ruthotto, 2017; Chen et al., 2018) the midpoint method was used and Runge Kutta methods were proposed. These methods are all *explicit* methods, i.e., the state, $\mathbf{Y}$ at time t_{j+1} is explicitly expressed by the states at previous times. While such methods enjoy simplicity, they suffer from the field of view problem and a lack of stability. Indeed, many small steps are needed in order to integrate the ODE for a long time. In particular, when explicit methods are applied to partial differential equations (PDEs), many time steps are required in order for information to travel on the entire computational mesh. This problem is well-known and documented in the numerical solution of PDEs, e.g. when solving Navier-Stokes equations (Gresho & Sani, 1987), the solution of flow in porous media (Chen et al., 2010), and in cloth simulation for computer graphics (Baraff & Witkin, 1998). Hence, the relation of convolutional ResNets to those PDEs described in (Ruthotto & Haber, 2018), provides an alternative explanation for the field-of-view problem.

2.2. The Semi-Implicit Network

One way to accelerate the communication of information across all pixels is to use implicit methods (Ascher & Petzold, 1998). Such methods express the state at time $\mathbf{Y}_{j+1}$ implicitly. For example, the simplest implicit method for ODEs is the backward Euler method where in order to obtain $\mathbf{Y}_{j+1}$ we solve the nonlinear equation

$$\mathbf{Y}_{j+1} - \mathbf{Y}_j = h\, f(\mathbf{Y}_{j+1}, \boldsymbol{\theta}_{j+1}). \tag{4}$$

The backward Euler method is stable for any choice of h when the eigenvalues of the Jacobian of f have no positive real part. Therefore, it is possible to take arbitrarily large steps in such a network while being robust to small perturbations of the input images due to, for example, noise or adversarial attacks. Unfortunately, implicit methods can be rather expensive. In particular, the solution of the nonlinear equation (4) is a non-trivial task that can be computationally intensive.

Rather than using a fully implicit method, we derive a new architecture using the computationally efficient Implicit-Explicit method (IMEX) (Ascher et al., 1997; 1995). IMEX is commonly used in fluid dynamics and surface formation and has applied also in the context of image denoising (Schönlieb & Bertozzi, 2011). The key idea of the IMEX method is to divide the right-hand side of the ODE into two parts. The first (nonlinear) part is treated explicitly and the second (linear) part is treated implicitly. We design the implicit part so that it can be solved efficiently. In our context, there is no natural division to an explicit and an implicit part and therefore, we rewrite the ODE (3) by adding and subtracting a linear invertible matrix

$$\dot{\mathbf{Y}}(t) = \underbrace{f(\mathbf{Y}(t), \boldsymbol{\theta}(t)) + \mathbf{L}\mathbf{Y}(t)}_{\text{explicit term}} - \underbrace{\mathbf{L}\mathbf{Y}(t)}_{\text{implicit term}} . \tag{5}$$

Here, $\mathbf{L}$ is a matrix that we are free to choose or train. We assume that $\mathbf{L}$ is symmetric positive definite matrix that is ”easy” to invert. As we show next, we can use a particular 3×3 convolution model for $\mathbf{L}$ that has these properties. Next, we use the forward Euler method for the explicit term and a backward Euler step for the implicit term. The forward propagation through the new network that we call IMEXnet then reads

$$\mathbf{Y}_{j+1} = (\mathbf{I} + h\mathbf{L})^{-1}(\mathbf{Y}_j + h\mathbf{L}\mathbf{Y}_j + h\, f(\mathbf{Y}_j, \boldsymbol{\theta}_j)), \tag{6}$$

where $\mathbf{I}$ denotes the identity matrix.

While the forward propagation may appear more complicated than a simple ResNet step, we show below that the computational complexity of the matrix inversion is similar to that of convolution and we emphasize that this construction has some of the favorable properties of an implicit method. Furthermore, we show that through an appropriate choice of the matrix $\mathbf{L}$ the network is unconditionally stable. This implies that no exploding modes will occur throughout the network training. Also, the matrix $(\mathbf{I} + h\mathbf{L})^{-1}$ is dense, i.e., it couples all the pixels in each channel of the image in a single step. For problems where the field-of-view is important, such methods can be very effective.

To demonstrate this fact we refer to the third row of Figure 1. It shows the forward propagation using the semi-implicit IMEX method where we choose $\mathbf{L}$ as a group convolution with the weights

$$\mathbf{L} = \frac{1}{6}\begin{pmatrix} -1 & -4 & -1 \\ -4 & 20 & -4 \\ -1 & -4 & -1 \end{pmatrix}, \tag{7}$$

which is a discrete Laplace operator. We discuss this choice next. Comparing the output images after only 5 time steps to a ResNet with 20 time steps it is apparent that the IMEX method increases the coupling between far away pixels.

2.3. Stability of the Method

We now discuss the selection of the matrix $\mathbf{L}$ and its impact on the stability of the network. To ensure low computational

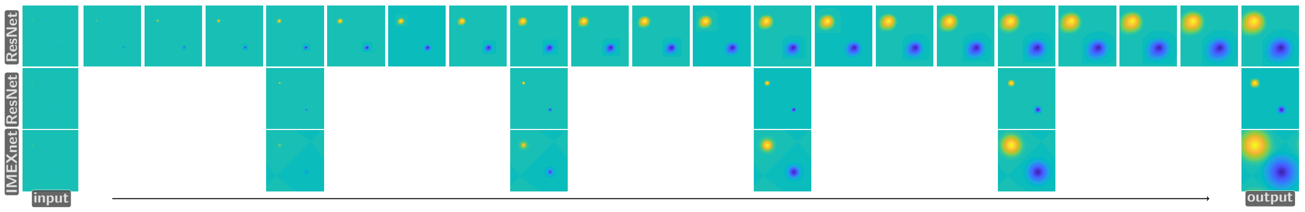

Figure 1. Comparison of explicit and implicit neural networks. Top row: Forward propagation of a test image through an explicit ResNet with 20 time steps. Second row: Explicit ResNet with only five time steps. While the forward propagation is four times faster to evaluate, information is transmitted less effectively. Bottom row: Forward propagation through the proposed semi-implicit network.

complexity, we choose to have $\mathbf{L}$ as a group convolution. Assuming, m channels, this implies that the matrix $\mathbf{L}$ has the form

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}_1 & & \\ & \ddots & \\ & & \mathbf{L}_m \end{pmatrix}.$$

In this way, the implicit step leads to m independent linear systems (which can be parallelized) and it allows us to use tools commonly available in most software packages. As we show in the next section, such a matrix is stored as a 3D tensor and can be quickly inverted.

We now analyze the IMEXnet in a simplified setting and demonstrate that an adequate choice of $\mathbf{L}$ can ensure its stability independent of the step size h. To this end, we show how to choose $\alpha \geq 0$ such that $\mathbf{L} = \alpha\mathbf{I}$ ensures the stability of the forward propagation for the model problem

$$\dot{\mathbf{Y}}(t) = \lambda\mathbf{Y}(t), \quad \mathbf{Y}(t) = \mathbf{Y}_0. \tag{8}$$

Here, $\lambda = -\lambda_{\text{real}} + \imath\lambda_{\text{imag}}$ with $0 \leq \lambda_{\text{real}}$ is a given complex number with non-positive real part. In this case, the norm of the solution $\mathbf{Y}(t) = \exp(\lambda t)\mathbf{Y}_0$ is bounded by the norm of $\mathbf{Y}_0$ for all times t.

As discussed above, the usual ResNet is equivalent to the forward Euler method, and for the model problem (8) reads

$$\mathbf{Y}_{j+1} = (1 + h\lambda)\mathbf{Y}_j.$$

This equation is stable (i.e., $\|\mathbf{Y}_{j+1}\| \leq \|\mathbf{Y}_j\|$) if and only if

$$|1 + \lambda h| \leq 1.$$

Hence, the usual ResNet may be unstable when $|\lambda|$ is large unless h is chosen small enough, which is computationally expensive. Now, consider the semi-implicit model with $\mathbf{L} = \alpha\mathbf{I}$, which can be written as

$$\mathbf{Y}_{j+1} = \frac{1 + h\lambda + h\alpha}{1 + h\alpha}\mathbf{Y}_j,$$

where a large h can be used as long as $0 \leq \alpha$ is chosen to ensure the stability of the scheme. Indeed, since we assume that the real part of λ is non-positive, it is always possible to choose α such that $\|\mathbf{Y}_{j+1}\| \leq \|\mathbf{Y}_j\|$, which implies that stability is conserved independent of h. The above discussion can be summarized by the following theorem:

Theorem 1 *Let $\mathbf{J}$ be a given matrix and consider the linear dynamical system $\dot{\mathbf{Y}}(t) = \mathbf{J}\mathbf{Y}(t)$. Assume that the eigenvalues of $\mathbf{J}$ have non-positive real parts, i.e., can be written as $\lambda = -\lambda_{\text{real}} + \imath\lambda_{\text{imag}}$. Then, if we choose α such that*

$$\frac{|\lambda|^2}{2\lambda_{\text{real}}} - \frac{1}{h} \leq \alpha, \quad \textit{for all } \lambda, \tag{9}$$

the magnification factor between layers in the IMEX method (6) is

$$\left|\frac{1 + h\lambda + h\alpha}{1 + h\alpha}\right| \leq 1, \quad \textit{for all } \lambda,$$

and the method is stable.

The proof of this theorem is straight forward by computing the absolute value of the magnification factor. It is also important to note that as $h \to 0$ we can choose $\alpha = 0$ and keep stability.

Despite its restrictive assumptions, the above theorem provides some intuition about the stability of the forward propagation through the IMEXnet. Here, we choose $\mathbf{J}$ as the Jacobian of the layer f in (2) with respect to the input features. The non-positivity of the real part of the eigenvalues can be ensured by construction (e.g., by setting $\mathbf{K}_1 = -\mathbf{K}_2^\top$ as in (Ruthotto & Haber, 2018)) and a bound on $|\lambda|$ can be obtained by imposing bound constraints on the convolution weights. A precise analysis is beyond the scope of this work since the forward problem is nonlinear and non-autonomous.

In our numerical experiments, we pick α to be relatively large (in the range 1-10). We noticed that around this range of values the method is rather insensitive to its choice.

3. Numerical Implementation and Computational Costs

We show in detail that our network, despite being slightly more complex than the standard ResNet, can be implemented using building blocks that exist in common machine learning frameworks and benefit from GPU acceleration, auto differentiation, etc. In particular, we discuss the computation of the implicit step, that is solving the linear system

$$(\mathbf{I} + h\mathbf{L})\mathbf{Y} = \mathbf{B},$$

where $\mathbf{L}$ is constructed above as group-wise convolution and $\mathbf{B}$ collects the explicit terms.

To solve the system efficiently, we use the representation of convolution in the Fourier space that states that the convolution between a kernel $\mathbf{A}$ and the features $\mathbf{Y}$ can be computed as

$$\mathbf{A} * \mathbf{Y} = \mathbf{F}^{-1}((\mathbf{F}\mathbf{A}) \odot (\mathbf{F}\mathbf{Y})).$$

where $\mathbf{F}$ is the Fourier transform, $*$ is a convolution, and $\odot$ is the Hadamard element-wise product. Here, and in the following, we assume periodic boundary conditions on the image data. This implies that if we need to compute the product of the inverse of the convolution operator defined by $\mathbf{A}$ (assuming it is invertible) with a vector, we can simply element-wise divide by the inverse Fourier transform of $\mathbf{A}$, i.e.,

$$\mathbf{A}^{-1} * \mathbf{Y} = \mathbf{F}^{-1}((\mathbf{F}\mathbf{Y}) \oslash (\mathbf{F}\mathbf{A})).$$

where $\oslash$ applied element-wise division.

In our case the kernel $\mathbf{A}$ is associated with the matrix

$$\mathbf{I} + h\mathbf{L},$$

which is invertible, e.g., when we choose $\mathbf{L}$ to be positive semi-definite. Thus, we define

$$\mathbf{L} = \mathbf{B}^\top \mathbf{B},$$

where $\mathbf{B}$ is a (trainable) group-wise convolution operator. A simple PyTorch code to compute the step is presented in Algorithm 1.

Using Fourier methods we need to have the convolution kernel at the same size as the image we convolve it with. This is done by generating an array of zeros that has the same size of the image and inserting the entries of the convolution into the appropriate places. The techniques is explained in detail in (Nagy & Hansen, 2006).

Let us now discuss the memory and computational effort involved with the method. To be more specific, we analyze the method for a single ResNet layer with m channels, applied to an image of size $s \times s$. Assuming that the stencil size of the convolutions is $\mathcal{O}(1)$, applying such a layer to an image requires a computational cost of $\mathcal{O}(m^2 s^2)$ operations and a memory that is $\mathcal{O}(m^2)$ to store the weights.

For the implicit networks, we have the usual explicit step followed by an implicit step. The implicit step is a group convolution is requires $\mathcal{O}(m(s\log(s))^2)$ additional operations, where the $s\log(s)$ term results from the Fourier transform. Since $\log(s)$ is typically much smaller than m the additional cost of the implicit step is insignificant.

The memory footprint of the implicit step is also very small. It requires only $\mathcal{O}(m)$ additional coefficients. For problems where the number of channels is larger than say, 100 this cost represents less than 1% additional storage. Thus, the improvement we obtain to ResNet comes with a very low cost of both computations and memory.

4. Numerical Experiments

In this section, we conduct two numerical experiments that demonstrate the points discussed above. In the first problem, we experiment with semantic segmentation of a synthetic dataset that we call the Q-tips dataset. We designed this dataset to expose the limitations of explicit methods and demonstrate the improvements of semi-implicit methods. In the second example, we show the advantages of our approach on the NYU Depth Dataset V2 that contains images of different room types together with their depth. The goal of the training here is to predict the depth map given the image of the room. While the two problems are different in their output they share the need for nonlocal coupling across large distances in order to deliver an accurate prediction.

4.1. The Q-tips Dataset

We introduce a synthetic semantic segmentation dataset intended to quantify the effect of a network's receptive field. In this dataset, every image contains a single object composed of a rectangular gray midsection with either a white or black square at each end. We define the object classes according to the combination of markers present, resulting in three classes (white-white, white-black, and black-black). The dataset is specifically designed to require a receptive field that encompasses the entire object for accurate classification. If information from both ends of the object is not available to a pixel in the output, then the problem is ambiguous.

For the following experiments we generate a dataset of 1024 training examples and 64 validation examples. Each 64×64 image consists of a single object of length, l, width, w, and orientation, r, randomly selected from a discrete uniform distribution, where $l \in \mathcal{U}\{32, 60\}$, $w \in \mathcal{U}\{4, 8\}$, and $r \in \mathcal{U}\{-180, 180\}$.

In order to evaluate the effect of our proposed semi-implicit

Algorithm 1 pyTorch implementation of the implicit convolution.

```
def diagImpConv(x, B,h):
    n = x.shape
    m = B.shape
    mid1 = (m[2] - 1) // 2
    mid2 = (m[3] - 1) // 2
    Bp = torch.zeros(m[0],n[2], n[3],device=B.device)
    Bp[:, 0:mid1 + 1, 0:mid2 + 1] = B[:, 0, mid1:, mid2:]
    Bp[:, -mid1:, 0:mid2 + 1]     = B[:, 0, 0:mid1, -(mid2 + 1):]
    Bp[:, 0:mid1 + 1, -mid2:]     = B[:, 0, -(mid1 + 1):, 0:mid2]
    Bp[:, -mid1:, -mid2:]         = B[:, 0, 0:mid1, 0:mid2]

    xh  = torch.rfft(x,  2, onesided=True)
    Bh  = torch.rfft(Bp, 2, onesided=True)
    t   = 1.0/(1.0 + h * (Bh[:, :, :, 0] ** 2 + Bh[:, :, :, 1] ** 2) )
    xBh = torch.zeros(n[0], n[1], n[2], (n[3] + 2) // 2, 2,device=B.device)
    for i in range(n[0]):
        xBh[i, :,:, :, 0] = xh[i, :, :, :, 0]*t
        xBh[i, :,:, :, 1] = xh[i, :, :, :, 1]*t
    xB = torch.irfft(xBh, 2, onesided=True,signal_sizes = x.shape[2:])
    return xB
```

Network	Parameters	IOU	Loss	Accuracy
IMEXnet	2701440	0.926	0.0982	99.56
ResNet	2691648	0.741	0.3332	98.18

Table 1. Comparison of semi-implicit IMEXnet and explicit ResNet on the synthetic Q-tips validation set.

architecture, we train two nearly identical 12-layer IMEXnet with weights that are randomly initialized from a uniform distribution on the interval $[0, 1)$. The opening layer expands the single channel input to 64 channels, and the width is subsequently doubled every 4 layers to result in a 224 channel output before the classifier. Neither network contains any pooling layers, and the convolution layers are padded such that the input and output are of the same resolution. In order to make one of the networks purely explicit, we set $\mathbf{L} = \mathbf{0}$, which will prevent any implicit coupling, effectively resulting in a ResNet. In both cases, we use stochastic gradient descent to minimize the weighted cross entropy loss for 200 epochs with a learning rate of 0.001 and a batch size of 8. The loss is weighted according to the normalized class frequencies calculated from the entire dataset in order to address the class imbalance due to the background.

For comparison we present various error measurements for both networks on the validation dataset in Table 1. Note that the IMEX method did much better in terms of loss and intersection over unions (IOU). The pixel accuracy counts the background and therefore is somewhat misleading. An example of the results on the testing set is plotted in Figure 2.

The table and images demonstrate that the ResNet cannot label the image correctly. The centerpiece of the rod is not continuously segmented with the end of the rod. In particular, the center of the rod is randomly classified as one of three object classes. Adding an implicit layer with a negligible memory footprint and computational complexity manages to resolve the problem, obtaining a near perfect segmentation.

4.2. The NYU Depth Dataset

The NYU-Depth V2 dataset is a set of indoor images recorded by both RGB and Depth cameras from the Microsoft Kinect. Four different scenes from the dataset are plotted in Figure 3. The goal of our network is to use the RGB images in order to predict the depth images. We use a subset of the dataset, made of the kitchen scene in order to train a network to achieve this task. The network contains three ResNet blocks and three bottle-neck blocks and is plotted in Figure 4.

The ResNet has only 506,928 parameters. For the IMEXnet we use an identical network but add implicit layers. This adds only roughly 27,000 more parameters for the implicit network. We use 500 epochs to fit the data. The initial L_2 misfit is 8.2. Using the ResNet architecture we are able to decrease the misfit to 1.10×10^{-2}. The small addition of parameters for the implicit method allows us to fit the data to 2.9×10^{-3}. The convergence of the two methods is presented in Figure 5.

We found that for learning one set of images (i.e. kitchens), it is possible to use rather few examples. For the kitchen dataset we used only 8 training images, 2 validation image and one test image. The results on the test image is plotted in Figure 6. We observe that we are able to obtain good results even when the number of images used for the training is small. These results echo the results obtained for image filtering using variational networks presented in (Riegler et al., 2015).

Comparing the ResNet and the IMEXnet we see that the pre-

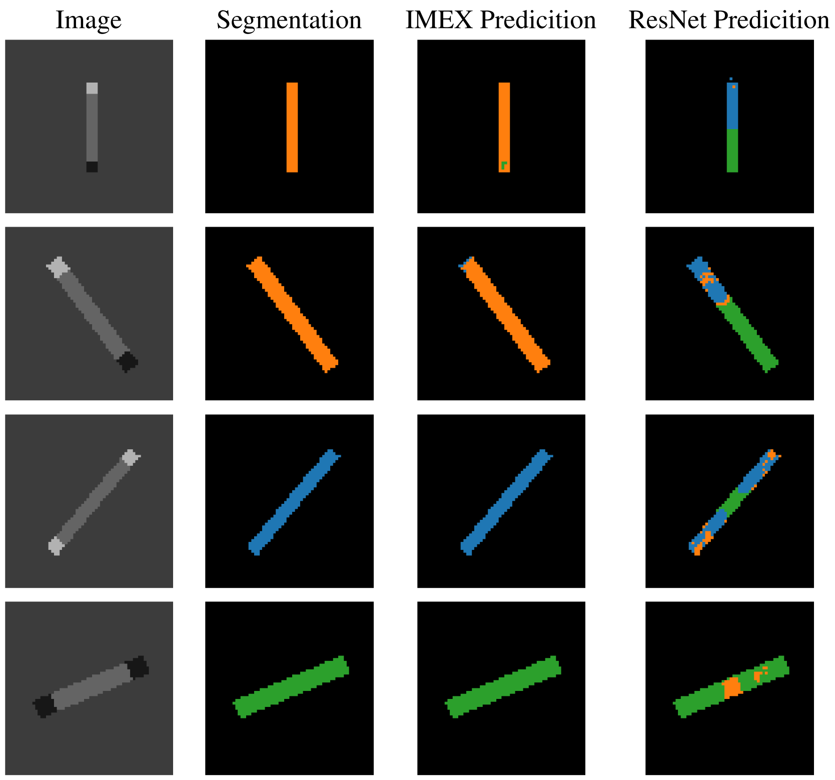

Figure 2. Segmentation results on the Q-tips dataset. We present the image (first column), the ground truth segmentation (second column), our method's prediction (third column), and the ResNet's prediction (last column).

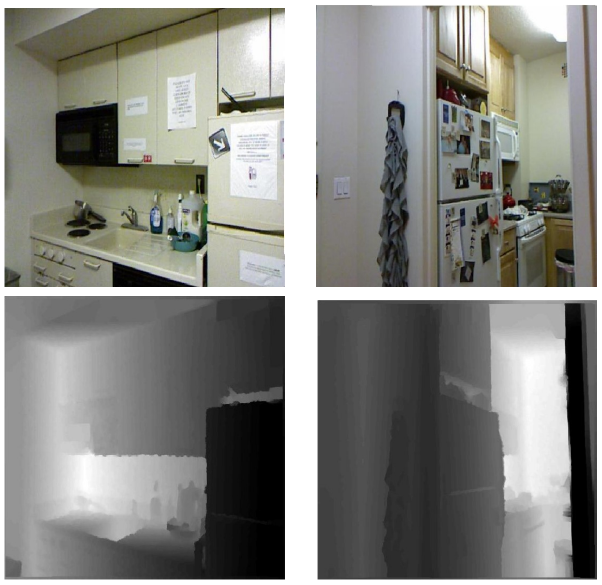

Figure 3. Examples from the NYU depth dataset. Two sets of kitchens are used together with their depth maps.

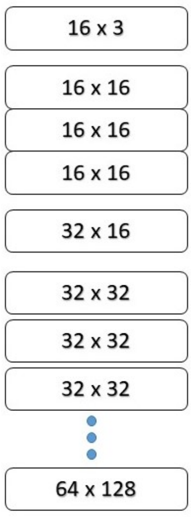

Figure 4. The ResNet used for the NYU dataset.

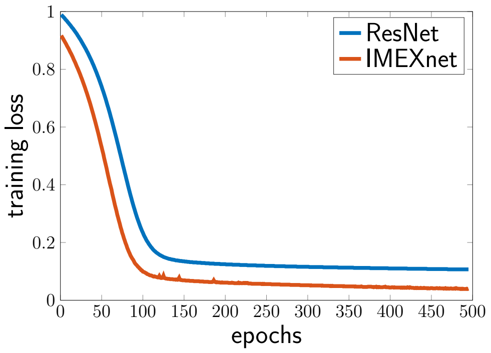

Figure 5. Convergence of the training using ResNet and IMEX methods on the NYU depth dataset.

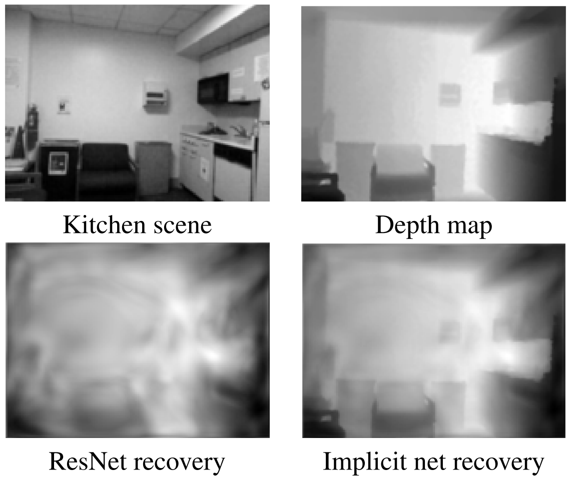

Figure 6. The testing image from the NYU dataset.

dictions of the IMEXnet is smoother than the one obtained from the ResNet. This is not surprising as the implicit step in the IMEXnet can be seen as a smoothing step. We also note that in our numerical experiments we have found that the implicit method is less sensitive to initialization. This is not surprising as the implicit step adds stability. Indeed, if we compare an explicit ResNet with an implicit one with the same kernels, the analysis suggests that while the explicit network may be unstable, the implicit one is stable, assuming $\mathbf{L}$ is chosen appropriately.

5. Summary

In this paper, we introduce the IMEXnet architecture for computer vision tasks that is inspired by semi-implicit IMEX methods commonly used to solve partial differential equations. Our new network extends standard ResNet architectures by adding implicit layers that involve a group-wise inverse convolution operator after each explicit layer. We have discussed and exemplified that this approach can resolve the field of view problem as well as the issue of the forward stability of the network. This makes this type of network suitable for problems where the dimension of the output is similar to the dimension of the input, such as semantic segmentation and depth estimation, and where nonlocal interactions are needed.

We exemplify, using PyTorch, that our method can be implemented efficiently using the available built-in functions. The computational complexity and memory allocation that is added by using the implicit step is small compared with the complexity and memory needed by the ResNet. We have shown that although the method has marginally larger cost compared with ResNet, it can be much more efficient in training as well as in validation on two simple model problems of semantic segmentation and depth estimation from images. Our code is available at `https://github.com/HaberGroup/SemiImplicitDNNs`.

While we have explored one semi-implicit method (5), it is important to realize that we are free to choose other models with similar properties. One attractive option is to remove the matrix $\mathbf{L}$ from the explicit part, and consider the diffusion-reaction problem

$$\dot{\mathbf{Y}}(t) = f(\mathbf{Y}(t), \boldsymbol{\theta}(t)) - \mathbf{L}\mathbf{Y}(t). \tag{10}$$

These types of equations have been used extensively to model nonlinear phenomena such as pattern formation and can have interesting behavior, e.g., it can lead to nonlinear waves. These systems were already studied by Turing (Turing, 1952) and have been studied extensively in (Murray, 1982; Ruuth, 1993; Witkin & Kass, 1991). A similar treatment using an IMEX integration scheme leads to

$$\mathbf{Y}_{j+1} = (\mathbf{I} + h\mathbf{L})^{-1}(\mathbf{I} + h\, f(\mathbf{Y}_j, \boldsymbol{\theta}_j)). \tag{11}$$

Performing a similar analysis than the one in Section 2.3, we see that this network has better stability properties. In particular, we can remove the restriction on having the real parts of the eigenvalues of $\mathbf{J} = \nabla_{\mathbf{Y}} f^{\top}$ to be positive. Since these types of equations are used for pattern formation, this type of network may have advantages when considering an output which has patterns, e.g. segmentation of texture. Detailed experimentation and evaluation of this approach is an item of future work.

Since semi-implicit methods are essential in many fields, we believe that this paper illustrates that they can also play a large role in the field of machine learning using deep nets.

Acknowledgements

EH's work is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). KL's work is supported by the Mitacs Accelerate program and Xtract AI. LRs work is supported by the US National Science Foundation (NSF) awards DMS 1522599 and DMS 1751636

References

Ascher, U. and Petzold, L. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, Philadelphia, 1998.

Ascher, U., Ruuth, S., and Wetton, B. Implicit-explicit methods for time-dependent pde's. *SIAM J. Numer. Anal.*, 32:797–823, 1995.

Ascher, U., Ruuth, S., and Spiteri, R. Implicit-explicit runge-kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics*, 1997. To appear.

Avendi, M., Kheradvar, A., and Jafarkhani, H. 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.

Baraff, D. and Witkin, A. Large steps in cloth simulation. In *COMPUTER GRAPHICS Proceedings, Annual Conference Series*, pp. 3447–3451. SIGGRAPH 98, Orlando, 1998.

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

Black, D., Sapiro, G., Marimont, D., and Heeger, D. Robust anisotropic diffusion. *IEEE Tran. on Image Processing*, 7:442–449, 1998.

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

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural ordinary differential equations. *Arxiv preprint arXiv:1806.07366*, 2018.

Chen, Z., Huan, G., and Ma, Y. *Computational Methods for Multiphase Flows in Porous Media*. SIAM, Philadelphia, 2010.

Ciccone, M., Gallieri, M., Masci, J., Osendorfer, C., and Gomez, F. J. Nais-net: Stable deep networks from non-autonomous differential equations. *CoRR*, abs/1804.07209, 2018. URL http://arxiv.org/abs/1804.07209.

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

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

Gresho, P. M. and Sani, R. L. On pressure boundary conditions for the incompressible Navier-Stokes equations. *Internat. J. Numer. Methods Fluids*, 7:1111–1145, 1987.

Haber, E. and Ruthotto, L. Stable architectures for deep neural networks. *Inverse Problems*, 34(1), 2017.

Hammernik, K., Klatzer, T., Kobler, E., Recht, M. P., Sodickson, D. K., Pock, T., and Knoll, F. Learning a variational network for reconstruction of accelerated mri data. *Magnetic Resonance in Medicine*, 79(6):30553071, Nov 2017. ISSN 0740-3194. doi: 10.1002/mrm.26977. URL http://dx.doi.org/10.1002/mrm.26977.

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

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pp. 630–645. Springer, 2016b.

He, Y.-H. Deep-learning the landscape. *arXiv preprint arXiv:1706.02714*, 2017.

Kadioglu, S., Knoll, D., Sussman, M., and Martineau, R. A second order jfnk-based imex method for single and multi-phase flows. In *Computational Fluid Dynamics 2010*, pp. 549–554. Springer, 2011. doi: 10.1007/978-3-642-17884-9_69.

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

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, pp. 6389–6399, 2018.

Lu, Y., Zhong, A., Li, Q., and Dong, B. Beyond finite layer neural network:bridging deep architects and numerical differential equations. *Thirty-fifth International Conference on Machine Learning (ICML)*, 2018.

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

Ma, C., Wang, J., and E, W. Model reduction with memory and the machine learning of dynamical systems. *CoRR*, abs/1808.04258, 2018. URL http://arxiv.org/abs/1808.04258.

Murray, J. Parameter space for Turing instability in reaction diffusion mechanisms: a comparison of models. *J. Theoretical Biology*, 98:143–162, 1982.

Nagy, J. and Hansen, P. *Deblurring Images*. SIAM, Philadelphia, 2006.

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

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

Rudin, L., Osher, S., and Fatemi, E. Nonlinear total variation based noise removal algorithms. In *Proceedings of the eleventh annual international conference of the Center for Nonlinear Studies on Experimental mathematics : computational issues in nonlinear science*, pp. 259–268. Elsevier North-Holland, Inc., 1992. doi: http://dx.doi.org/10.1016/0167-2789(92)90242-F.

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

Ruuth, S. Implicit-explicit methods for reaction–diffusion problems. *In preparation*, 1993.

Schönlieb, C.-B. and Bertozzi, A. Unconditionally stable schemes for higher order inpainting. *Communications in Mathematical Sciences*, 9:413–457, 06 2011.

Turing, A. M. The chemical basis of morphogenesis. *Phil. Transact Royal Soc. B*, 37:237–242, 1952.

Weickert, J. *Anisotropic Diffusion in Image Processing*. B.G Teubner Stuttgart, 1998.

Weinan, E. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11, 2017.

Witkin, A. and Kass, M. Reaction-diffusion textures. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH ’91, pp. 299–308, New York, NY, USA, 1991. ACM.