Differentiable Unrolled Alternating Direction Method of Multipliers for OneNet

Zoltán Á. Milacski[1]
srph25@gmail.com
Barnabás Póczos[2]
bapoczos@cs.cmu.edu
András Lőrincz[1]
lorincz@inf.elte.hu

[1] Department of Artificial Intelligence
ELTE Eötvös Loránd University
Budapest, Hungary

[2] Machine Learning Department
Carnegie Mellon University
Pittsburgh, USA

Abstract

Deep neural networks achieve state-of-the-art results on numerous image processing tasks, but this typically requires training problem-specific networks. Towards multi-task learning, the One Network to Solve Them All (OneNet) method was recently proposed that first pretrains an adversarial denoising autoencoder and subsequently uses it as the proximal operator in Alternating Direction Method of Multipliers (ADMM) solvers of multiple imaging problems. In this work, we highlight training and ADMM convergence issues of OneNet, and resolve them by proposing an end-to-end learned architecture for training the two steps jointly using Unrolled Optimization with backpropagation. In our experiments, our solution achieves superior or on par results compared to the original OneNet and Wavelet sparsity on four imaging problems (pixelwise inpainting-denoising, blockwise inpainting, scattered inpainting and super resolution) on the MS-Celeb-1M and ImageNet data sets, even with a much smaller ADMM iteration count.

1 Introduction

In recent years, we have seen an emergence of deep neural networks [17] in many problems, *e.g.*, image [11, 12, 16, 20], text [6, 14, 18] and speech [5, 23] processing. The common theme in these methods is collecting a large supervised data set and training a parametric mapping using gradient descent. This achieves superior test set results compared to manually engineered algorithms. However, each problem typically requires a specific network trained for the actual task. This is computationally expensive and wasteful in the sense that problems defined on the same data may share features. Towards multi-task learning, Chang *et al*. [3] proposed the '*One Network to Solve Them All*' (or *OneNet* in short) method, which has two steps: 1) pretraining an adversarial denoising autoencoder with a hand-designed broad noise family, 2) plugging in the pretrained autoencoder into the *Alternating Direction Method of Multipliers* (*ADMM*) solvers of linear inverse problems with various image formation models (compressive sensing, pixelwise inpainting-denoising, scattered inpainting, blockwise inpainting and super resolution). The authors showed that this method can outperform hand-designed signal priors and is competitive with problem-specific networks.

However, the original OneNet procedure suffers from multiple and interconnected issues that may be overcome. First, it is not an end-to-end trained architecture, as the denoising autoencoder is trained separately from the ADMM algorithm. Second, the inputs to the denoising autoencoder used in pretraining are suboptimal in the sense that their counterparts encountered during the steps of ADMM may be substantially different. Third, the proposed method may require many ADMM iterations to reach satisfying results. Fourth, we empirically observed that the proposed adversarial training procedure brings about realistically looking solutions, but this often hurts performance in terms of *Peak signal-to-noise ratio* (*PSNR*). Lately, Diamond *et al*. [7] proposed the *Unrolled Optimization with Deep Priors* (or *Unrolled Optimization* in short) framework for linear inverse problems, which contains an unfolded differentiable ADMM procedure as a special case that avoids such problems, but they consider problem specialized networks for individual image formation models, that could be overcome by means of the OneNet principle.

In this paper, we address the aforementioned issues by merging the benefits of the OneNet and the Unrolled Optimization frameworks, as we use the same deep parametric prior for multiple linear inverse problems (as in OneNet) and optimize it by unrolling and performing supervised learning (as in Unrolled Optimization). Specifically, we unroll ADMM for finite steps, treat it as a differentiable function, and train the autoencoder end-to-end with it simultaneously using backpropagation, encouraging optimal results by the final step. This way we ensure that the network is able to deal with inputs that are actually encountered in ADMM with significantly reduced iteration count; while still using the same network for the different image formation models. Note that this comes at the cost of sacrificing the original unsupervised learning procedure and applying supervision directly in training phase, leveraging ground truth solutions to a prescribed set of linear inverse problems. We show that our network achieves significantly better or comparable test results on pixelwise inpainting-denoising, scattered inpainting, blockwise inpainting and super resolution while also requiring much fewer ADMM iterations compared to both the original OneNet and Wavelet sparsity on the MS-Celeb-1M and ImageNet data sets. Our scheme is summarized in Fig. 1. The source code for our project is made available for reproducibility[1].

2 Theoretical Background and Related Work

2.1 Notation

Throughout the paper we use the following conventions. A bold face letter refers to a multi-dimensional tensor, a tensor superscript denotes a sample index and a tensor subscript indexes the tensor across the corresponding axes. Following standard machine learning notation, for a 5-dimensional tensor $\boldsymbol{Z} \in \mathbb{R}^{N\times T\times H\times W\times F}$, $z^{(n,t)}_{h,w,f} \in \mathbb{R}$ refers to the pixel value of the n^{th} sample image in the t^{th} iteration at the h^{th} row, w^{th} column and f^{th} feature. N, T, H, W, F stands for the total number of images, iterations, image rows (height), image columns (width) and features (channels), accordingly. When it should cause no confusion, we refer to (h,w) as pixel indices. Functions are specified by normal typesetting, where each "dot" $(\cdot)$ represents a respective argument. $\|\cdot\|_p$ is the ℓ_p norm, vec stands for the vectorization operator, img designates the reshaping to image operator (the inverse of vec) and pinv denotes the Moore – Penrose pseudoinverse.

[1] https://github.com/srph25/diff_unrolled_admm_onenet

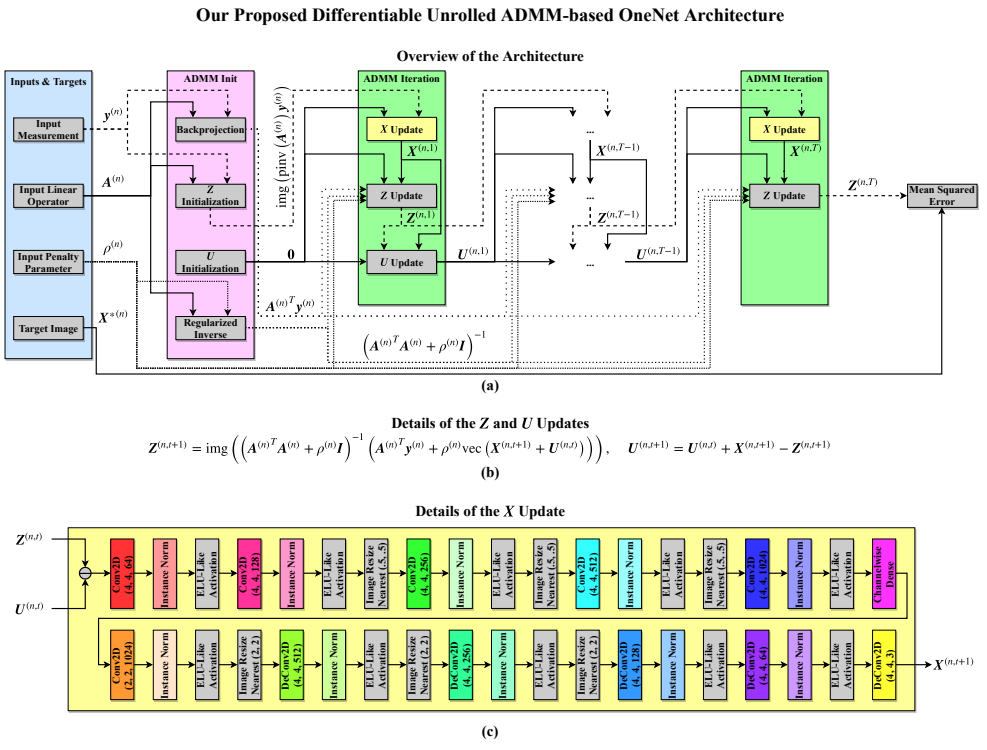


Figure 1: Schematic diagram of our proposed Differentiable Unrolled Alternating Direction Method of Multipliers (ADMM)-based OneNet architecture. Figure is best viewed in color (same color means operations with tied weights, gray designates nonparametric operations, all arrows denote forward propagation). (a) Our architecture unrolls T iterations of ADMM with measurements coming from multiple image formation models while replacing the proximal operator in the $\boldsymbol{X}$ update with a deep convolutional neural network. The parameters of the network are trained to minimize the squared error between the final output $\boldsymbol{Z}^{(n,T)}$ and the ground truth image $\boldsymbol{X}^{*(n)}$. (b) As in ADMM, we update $\boldsymbol{Z}^{(n,t)}$ with an appropriate pseudoinverse, while $\boldsymbol{U}^{(n,t)}$ is a cumulative summation. (c) The network consists of 6 convolutional, 1 channel-wise dense and 5 deconvolutional layers; with instance normalization, exponential linear unit-like activation and nearest neighbor image resizing layers in between.

2.2 Linear Inverse Problems and Signal Priors

The goal of a *linear inverse problem* is to reconstruct an image $\boldsymbol{X}^{*(n)} \in \mathbb{R}^{H\times W\times F}$ from a measurement vector $\boldsymbol{y}^{(n)} \in \mathbb{R}^{d^{(n)}}$ of the form

$$\boldsymbol{y}^{(n)} = \boldsymbol{A}^{(n)} \operatorname{vec}\left(\boldsymbol{X}^{*(n)}\right) + \boldsymbol{\xi}^{(n)}, \tag{1}$$

where $\boldsymbol{A}^{(n)} \in \mathbb{R}^{d^{(n)}\times(H\cdot W\cdot F)}$ is the measurement operator and $\boldsymbol{\xi}^{(n)} \in \mathbb{R}^{d^{(n)}}$ is the noise. (1) is often called the image formation model. For example, in image inpainting, $\boldsymbol{A}^{(n)}$ is pixelwise mask operator; in super resolution, $\boldsymbol{A}^{(n)}$ is a downsampling operator averaging nearby pixels; in compressive sensing, $\boldsymbol{A}^{(n)}$ is a random Gaussian matrix. $\boldsymbol{A}^{(n)}$ is often overcomplete, *i.e.*,

the corresponding linear system is underdetermined, thus the null space of $\boldsymbol{A}^{(n)}$ is non-trivial and there are an infinite number of feasible solutions. The index n represents that one may solve many such problems together, where $\boldsymbol{A}^{(n)}$ may vary considerably.

To facilitate selecting a particular solution out of the feasible set, the reconstruction loss is often regularized with a *signal prior* penalty:

$$\min_{\boldsymbol{X}^{(n)}} \frac{1}{2} \left\| \boldsymbol{y}^{(n)} - \boldsymbol{A}^{(n)} \operatorname{vec}\left(\boldsymbol{X}^{(n)}\right) \right\|_2^2 + \lambda \phi \left(\operatorname{vec}\left(\boldsymbol{X}^{(n)}\right) \right), \tag{2}$$

where $\phi \colon \mathbb{R}^{(H \cdot W \cdot F)} \to \mathbb{R}$ is the signal prior and $\lambda \geq 0$. Note that we apply the same ϕ for each sample index n, so we assume that a common latent structure is present for all $\boldsymbol{X}^{(n)}$. In the previous decade, hand-designed signal priors were widely used as ϕ in the literature. A natural choice is sparsity in some transformation domain, *i.e.*, $\phi\left(\operatorname{vec}\left(\boldsymbol{X}^{(n)}\right)\right) = \left\| \boldsymbol{W} \operatorname{vec}\left(\boldsymbol{X}^{(n)}\right) \right\|_1$, where $\boldsymbol{W}$ is a operator representing either wavelet transformation, image gradient, or some other similar linear operation.

2.3 Alternating Direction Method of Multipliers

The *Alternating Direction Method of Multipliers* (*ADMM*) [2] is an optimization method for solving linear inverse problems in the form of (2). ADMM relies on variable splitting: an auxiliary variable $\boldsymbol{Z}^{(n)}$ is introduced that is constrained to be equal to $\boldsymbol{X}^{(n)}$. This gives us the following optimization problem:

$$\min_{\boldsymbol{X}^{(n)}, \boldsymbol{Z}^{(n)}} \frac{1}{2} \left\| \boldsymbol{y}^{(n)} - \boldsymbol{A}^{(n)} \operatorname{vec}\left(\boldsymbol{Z}^{(n)}\right) \right\|_2^2 + \lambda \phi \left(\operatorname{vec}\left(\boldsymbol{X}^{(n)}\right) \right) \quad \text{s.t.} \quad \boldsymbol{X}^{(n)} = \boldsymbol{Z}^{(n)}, \tag{3}$$

which is equivalent to the original regularized problem (2). The scaled form of the augmented Lagrangian of (3) can be written as

$$L(\boldsymbol{X}^{(n)}, \boldsymbol{Z}^{(n)}, \boldsymbol{U}^{(n)}) = \frac{1}{2} \left\| \boldsymbol{y}^{(n)} - \boldsymbol{A}^{(n)} \operatorname{vec}\left(\boldsymbol{Z}^{(n)}\right) \right\|_2^2 + \lambda \phi \left(\operatorname{vec}\left(\boldsymbol{X}^{(n)}\right) \right) + \frac{\rho^{(n)}}{2} \left\| \boldsymbol{X}^{(n)} - \boldsymbol{Z}^{(n)} + \boldsymbol{U}^{(n)} \right\|_2^2, \tag{4}$$

where $\rho^{(n)} > 0$ is the penalty parameter of the constraint $\boldsymbol{X}^{(n)} = \boldsymbol{Z}^{(n)}$, and $\boldsymbol{U}^{(n)}$ is the tensor of dual variables divided by $\rho^{(n)}$. By alternatively optimizing $L\left(\boldsymbol{X}^{(n)}, \boldsymbol{Z}^{(n)}, \boldsymbol{U}^{(n)}\right)$ over $\boldsymbol{X}^{(n)}$, $\boldsymbol{Z}^{(n)}$, and $\boldsymbol{U}^{(n)}$, ADMM is composed of the following iterations:

$$\boldsymbol{Z}^{(n,0)} = \operatorname{img}\left(\operatorname{pinv}\left(\boldsymbol{A}^{(n)}\right)\boldsymbol{y}^{(n)}\right), \quad \boldsymbol{U}^{(n,0)} = \boldsymbol{0}, \tag{5}$$

$$\boldsymbol{X}^{(n,t+1)} = \arg\min_{\boldsymbol{X}} \frac{\rho^{(n)}}{2} \left\| \boldsymbol{X} - \boldsymbol{Z}^{(n,t)} + \boldsymbol{U}^{(n,t)} \right\|_2^2 + \lambda \phi \left(\operatorname{vec}\left(\boldsymbol{X}\right) \right), \tag{6}$$

$$\boldsymbol{Z}^{(n,t+1)} = \arg\min_{\boldsymbol{Z}} \frac{1}{2} \left\| \boldsymbol{y}^{(n)} - \boldsymbol{A}^{(n)} \operatorname{vec}\left(\boldsymbol{Z}\right) \right\|_2^2 + \frac{\rho^{(n)}}{2} \left\| \boldsymbol{X}^{(n,t+1)} - \boldsymbol{Z} + \boldsymbol{U}^{(n,t)} \right\|_2^2, \tag{7}$$

$$\boldsymbol{U}^{(n,t+1)} = \boldsymbol{U}^{(n,t)} + \boldsymbol{X}^{(n,t+1)} - \boldsymbol{Z}^{(n,t+1)}. \tag{8}$$

The update of $\boldsymbol{X}^{(n)}$ in (6) is the proximal operator of the signal prior ϕ with penalty $\frac{\rho^{(n)}}{\lambda}$, denoted as $\text{prox}_{\phi,\rho^{(n)}/\lambda}\left(\boldsymbol{Z}^{(n,t)}-\boldsymbol{U}^{(n,t)}\right)$. When the signal prior uses ℓ_1-norm, the proximal operator is simply a soft-thresholding. The update of $\boldsymbol{Z}^{(n)}$ in (7) is a least squares problem.

Notice that the ADMM algorithm separates the signal prior ϕ from the linear operators $A^{(n)}$, $n=1,\dots,N$. This enables us to learn a signal prior that can be used with any linear operator.

2.4 One Network to Solve Them All

Chang *et al.* [3] argue that in each iteration t, the input $\left(\boldsymbol{Z}^{(n,t)}-\boldsymbol{U}^{(n,t)}\right)$ to the proximal operator resembles an image, which is ideally mapped to a noiseless image. Consequently, the authors proposed to pretrain an adversarial denoising autoencoder $p(\boldsymbol{\theta},\cdot)$ and use it during inference phase in ADMM in place of the proximal operator in (6):

$$\boldsymbol{X}^{(n,t+1)}=p\left(\boldsymbol{\theta},\boldsymbol{Z}^{(n,t)}-\boldsymbol{U}^{(n,t)}\right). \tag{9}$$

In other words, a modified ADMM is suggested with steps (5), (9), (7) and (8). They call their framework *One Network to Solve Them All* (*OneNet*) as they are using the same pretrained network $p(\boldsymbol{\theta},\cdot)$ with multiple linear inverse problem matrices $\boldsymbol{A}^{(n)}$ (compressive sensing, pixelwise inpainting-denoising, scattered inpainting, blockwise inpainting and super resolution). They showed either significantly improved or on par performance compared to Wavelet sparsity and problem-specific networks, depending on the imaging task.

In pretraining phase, their autoencoder takes a noisy image $\left(\boldsymbol{X}^{*(n)}+\boldsymbol{m}^{(n)}\right)$ as input and is trained to output its noiseless counterpart $\boldsymbol{X}^{*(n)}$, consequently the signal prior ϕ is the indicator function of the set of natural images. The noise $\boldsymbol{m}^{(n)}$ is generated using a hand-designed broad family of noise functions (Gaussian with spatially varying standard deviations). The network parameters $\boldsymbol{\theta}$ are trained to minimize the mean squared error (MSE) and to fool a discriminator $d(\boldsymbol{\omega},\cdot)$ (classifier with sigmoid output that is trained with binary cross-entropy to distinguish between real and $p(\boldsymbol{\theta},\cdot)$-generated samples) with additional regularization terms[2]:

$$\begin{aligned}\min_{\boldsymbol{\theta}}\Bigg\{&\left\|\boldsymbol{X}^{*(n)}-p\left(\boldsymbol{\theta},\boldsymbol{X}^{*(n)}+\boldsymbol{m}^{(n)}\right)\right\|_2^2-0.001\cdot\log\left(d\left(\boldsymbol{\omega},p\left(\boldsymbol{\theta},\boldsymbol{X}^{*(n)}+\boldsymbol{m}^{(n)}\right)\right)\right)\\&+R\left(\boldsymbol{\theta},\boldsymbol{\omega},\boldsymbol{X}^{*(n)},\boldsymbol{m}^{(n)}\right)\Bigg\},\end{aligned} \tag{10}$$

$$\min_{\boldsymbol{\omega}}\left\{-\log\left(d\left(\boldsymbol{\omega},\boldsymbol{X}^{*(n)}\right)\right)-\log\left(1-d\left(\boldsymbol{\omega},p\left(\boldsymbol{\theta},\boldsymbol{X}^{*(n)}+\boldsymbol{m}^{(n)}\right)\right)\right)\right\}. \tag{11}$$

The training with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$ is performed with alternating steps using Adam [15] optimizer, introducing competition between $p(\boldsymbol{\theta},\cdot)$ and $d(\boldsymbol{\omega},\cdot)$. As the network is trained independently from the linear operators $\boldsymbol{A}^{(n)}$, it is expected to generalize well to new operators, however, it also underfits those observed in inference phase within ADMM.

In addition to the above findings addressed in the original paper, we observed two phenomena in practice. First, depending on the linear inverse problem, ADMM with OneNet

[2]The additional regularization terms are argued to make the following ADMM procedure more stable, but we omit them here for simplicity.

may require many iterations (up to 300) to converge, which is too costly for real time applications. Second, the proposed adversarial training procedure for OneNet helps producing visually appealing images, but imposes too strong regularization and slightly hurts the PSNR score of ADMM compared to using the MSE objective on its own. This is due to the definition of PSNR being derived from MSE. Thus, when designing our method, we dropped the adversarial training procedure and focused on using the MSE objective solely to reduce computational complexity.

2.5 Unrolled Optimization with Deep Priors

Diamond *et al.* [7] argue that deep models lack a systematic approach for incorporating prior knowledge of image formation models. Thus the authors propose the *Unrolled Optimization with Deep Priors* framework based on Deep Unfolding [13], in which they truncate a classical iterative optimization algorithm (*e.g.*, Proximal Gradient [4], Iterative Shrinkage Thresholding [1, 8] and even ADMM [2, 21]) and propose using deep convolutional prior architectures $p(\boldsymbol{\theta}^{(t)}, \cdot)$ within the unrolled optimization:

$$\left(\boldsymbol{Z}^{(n,0)}, \boldsymbol{U}^{(n,0)}\right) = \beta\left(\boldsymbol{\theta}^{(0)}, \boldsymbol{y}^{(n)}, \boldsymbol{A}\right), \tag{12}$$

$$\boldsymbol{X}^{(n,t+1)} = p\left(\boldsymbol{\theta}^{(t)}, \boldsymbol{Z}^{(n,t)}, \boldsymbol{U}^{(n,t)}\right), \tag{13}$$

$$\left(\boldsymbol{Z}^{(n,t+1)}, \boldsymbol{U}^{(n,t+1)}\right) = \Gamma\left(\boldsymbol{\theta}^{(t)}, \boldsymbol{y}^{(n)}, \boldsymbol{A}, \boldsymbol{X}^{(n,t+1)}, \boldsymbol{Z}^{(n,t)}, \boldsymbol{U}^{(n,t)}\right), \tag{14}$$

where $\beta\left(\boldsymbol{\theta}^{(0)}, \cdot, \cdot\right)$ is a parametric initialization function and $\Gamma\left(\boldsymbol{\theta}^{(t)}, \cdot, \cdot, \cdot, \cdot, \cdot\right)$ is a parametric data function encoding prior information about the image formation model, for which they use a fixed linear operator $\boldsymbol{A}$ for samples $n = 1, \ldots, N$. The iterations are considered for a finite number of steps T and are treated as one large differentiable parametric composite function. The network parameters $\boldsymbol{\theta}^{(t)}$, $t = 0, \ldots, T$ are trained to minimize the MSE between the final output $\boldsymbol{Z}^{(n,T)}$ and the ground truth solution $\boldsymbol{X}^{*(n)}$:

$$\min_{\boldsymbol{\theta}^{(0)}, \ldots, \boldsymbol{\theta}^{(T)}} \left\| \boldsymbol{X}^{*(n)} - \boldsymbol{Z}^{(n,T)} \right\|_2^2. \tag{15}$$

To solve (15), they use Adam [15]. Their method outperforms previous state-of-the-art results on a broad variety of inverse imaging problems.

However, the authors do not address the case of training a single network with multiple linear operators $\boldsymbol{A}^{(n)}$ coming from different image formation models as in OneNet, as their work focuses on the case of a fixed image formation model $\boldsymbol{A}^{(n)} = \boldsymbol{A}$, $n = 1, \ldots, N$. In other words, they train problem specialized networks and thus achieve better results for the fixed $\boldsymbol{A}$ but fail to generalize to new operators due to overfitting.

3 Methods

3.1 Proposed Solution

In this section, we adopt the Unrolled Optimization with Deep Priors (Sect. 2.5) framework to the OneNet (Sect. 2.4) procedure.

We argue that the input image $\boldsymbol{X}^{*(n)}$ and the hand-designed noise $\boldsymbol{m}^{(n)}$ used in the OneNet framework are suboptimal in terms of later usage as part of ADMM. Notice that during inference phase $p(\boldsymbol{\theta},\cdot)$ takes $\left(\boldsymbol{Z}^{(n,t)}-\boldsymbol{U}^{(n,t)}\right)$ as argument, which may differ considerably from $\left(\boldsymbol{X}^{*(n)}+\boldsymbol{m}^{(n)}\right)$ encountered in pretraining phase. This may hurt ADMM performance and convergence. Consequently, one should expect better results within fewer ADMM iterations when the denoising autoencoder $p(\boldsymbol{\theta},\cdot)$ is trained with the exact inputs from the ADMM scenario, as in Unrolled Optimization. This remark provides the basis of our method.

While collecting samples from $\boldsymbol{Z}^{(n,t)}$ and $\boldsymbol{U}^{(n,t)}$ could be a reasonable choice, instead we choose to take an end-to-end training direction. Specifically, we follow the Unrolled Optimization recipe by defining its iterations to be the those of the modified ADMM ((12)⇔(5), (13)⇔(9), (14)⇔(7)& (8)) with objective (15). Note that this is feasible as the update of $\boldsymbol{Z}^{(n)}$ in (7) can be realized as a multiplication with an appropriate pseudoinverse and the update of $\boldsymbol{U}^{(n)}$ in (8) is just a cumulative summation, thus both of these facilitate backpropagation. We propose using tied weights over the iterations ($\boldsymbol{\theta}^{(t)}=\boldsymbol{\theta}$ for $t=1,\dots,T$). The computational graph of our procedure is depicted in Fig. 1.

This way our procedure is ensured to deal with the actual inputs seen in practice, while also promoting satisfactory results within a much smaller iteration count compared to the original OneNet. In contrast to Unrolled Optimization, performance should not degrade even when using very different linear operators $\boldsymbol{A}^{(n)}$. On the other hand, our network is tailored to the linear operators $\boldsymbol{A}^{(n)}$ seen in training phase and is expected to generalize poorly to new operators, but still overfit less than the original Unrolled Optimization procedure.

3.2 Experimental Setup

During our experiments, we mostly followed the setups (data sets, hyperparameter settings) from the original OneNet paper [3]. We compared the performance of our Differentiable Unrolled ADMM-based OneNet with its original variant (which we left completely untouched) and Wavelet sparsity in terms of PSNR and ADMM iteration count.

Due to memory limitations, we restricted ourselves to a problem set with sparse linear operators $\boldsymbol{A}^{(n)}$ only: pixelwise inpainting-denoising (randomly replacing half of the pixels by zeros and adding Gaussian noise with 0.1 standard deviation, $\rho^{(n)}=0.3$), scattered inpainting (randomly replacing 10 blocks of size 6×6 with zeros, $\rho^{(n)}=0.3$), blockwise inpainting (replacing the center block of size 19×19 with zeros, $\rho^{(n)}=0.2$) and super resolution (downsampling to size 32×32 using box-averaging, $\rho^{(n)}=0.5$).

For all techniques, in testing phase, we chose $N=500$ random images from the held-out test set and considered all linear operators from the problem set.

For our method, in each training step, we constructed mini-batches of size $N=10$ with samples randomly generated for $n=1,\dots,10$: first we chose a random image $\boldsymbol{X}^{*(n)}$ from the training set and a linear operator $\boldsymbol{A}^{(n)}$ from the problem set (together with penalty parameter $\rho^{(n)}$), and calculated the corresponding measurement $\boldsymbol{y}^{(n)}$. Then we treated the triplet $\left(\boldsymbol{y}^{(n)},\boldsymbol{A}^{(n)},\rho^{(n)}\right)$ as inputs and $\boldsymbol{X}^{*(n)}$ as target to the differentiable unrolled ADMM.

We employed the architecture in Fig. 1: (a) we unrolled ADMM for $T=8$ iterations, (b) leaving the updates of $\boldsymbol{Z}^{(n,t)}$ and $\boldsymbol{U}^{(n,t)}$ untouched, (c) but applying a deep network $p(\boldsymbol{\theta},\cdot)$ in the update of $\boldsymbol{X}^{(n,t)}$ consisting of 6 convolutional, 1 channel-wise dense and 5 deconvolutional linear layers, followed by instance normalization [22] and exponential linear-like smooth $\frac{\mathrm{softplus}(2x+2)}{2}-1$ activation [9] and optionally nearest neighbor image resizing layers.

We used Adam [15] as optimizer with learning rate 10^{-4}, $\beta_1 = 0$ and $\beta_2 = 0.9$. To circumvent the exploding gradient problem while unrolling the ADMM, we clipped gradient values to the interval $[-10, 10]$.

We performed our calculations on 2 NVIDIA® GTX1080 GPUs with 8 GB VRAM exploiting data parallelism. $N = 10$ and $T = 8$ were chosen to fit the whole procedure into the GPUs.

We implemented our network in Python 3.6 using Tensorflow 1.14.1 based upon the original OneNet source code [3].

We performed our experiments on the MS-Celeb-1M[4] [10] and ImageNet[5] [19] data sets. The former contains 8,000,000 cropped and aligned face images of 99,892 people from different viewing angles, from which we randomly selected images of 73,678 and 16,204 people as the training and test sets, respectively. The latter contains 12,000,000 training images and 100,000 test images from the Internet. We resized the images to $H \times W = 64 \times 64$ and executed 50,000 training iterations for both autoencoders on both data sets.

4 Results and Discussion

Table 1 shows our quantitative results. Table 2 depicts the corresponding qualitative results for illustrative purposes.

Table 1: Quantitative experimental results for the MS-Celeb-1M and ImageNet data sets. PID, BI, SI, SR and Iter stand for pixelwise inpainting-denoising, blockwise inpainting, scattered inpainting, super resolution and the number of performed ADMM iterations, respectively. Means and standard deviations were computed over 500 randomly chosen test images. Our contribution and the winning numbers are highlighted in bold.

	MS-Celeb-1M (64 × 64)							
	PID		BI		SI		SR	
	PSNR	Iter	PSNR	Iter	PSNR	Iter	PSNR	Iter
Diff. Unr. ADMM OneNet	**26.87** (±1.2)	8.00 (± 0.0)	**28.59** (±2.3)	8.00 (± 0.0)	**30.66** (±1.9)	8.00 (± 0.0)	28.19 (±3.1)	13.00 (± 0.0)
Original OneNet [8]	25.39 (±1.6)	13.96 (± 0.4)	24.41 (±2.4)	222.12 (±99.4)	25.23 (±2.4)	45.43 (± 17.7)	27.15 (±1.9)	6.87 (± 0.6)
Wavelet ℓ_1 Sparsity	20.81 (±1.0)	57.25 (± 12.1)	17.70 (±1.9)	44.42 (±18.1)	28.60 (±2.4)	68.31 (± 25.9)	**28.66** (±1.7)	17.85 (± 0.4)
	ImageNet (64 × 64)							
	PID		BI		SI		SR	
	PSNR	Iter	PSNR	Iter	PSNR	Iter	PSNR	Iter
Diff. Unr. ADMM OneNet	24.91 (±1.8)	8.00 (± 0.0)	**26.59** (±3.0)	8.00 (± 0.0)	**27.92** (±2.5)	8.00 (± 0.0)	26.38 (±2.5)	11.00 (± 0.0)
Original OneNet [8]	**25.44** (±1.8)	17.39 (± 18.0)	22.25 (±2.9)	300.00 (± 0.0)	22.85 (±2.6)	299.53 (± 10.5)	26.11 (±3.0)	25.42 (± 8.4)
Wavelet ℓ_1 Sparsity	20.18 (±1.5)	163.90 (±130.0)	19.33 (±2.9)	250.58 (±93.6)	27.65 (±2.8)	210.71 (±122.4)	**26.70** (±2.4)	282.79 (±56.7)

Overall, one can see both quantitatively and qualitatively that our proposed Unrolled ADMM-based OneNet performs the best on the blockwise inpainting and the scattered inpainting tasks. By visual inspection, our method generates semantically correct blocks, while the 2 baseline schemes are unable to capture the statistics of the observed pixels.

During the pixelwise inpainting-denoising problem, our algorithm is comparable to the original OneNet, as it marginally wins on MS-Celeb-1M, but slightly falls behind on ImageNet.

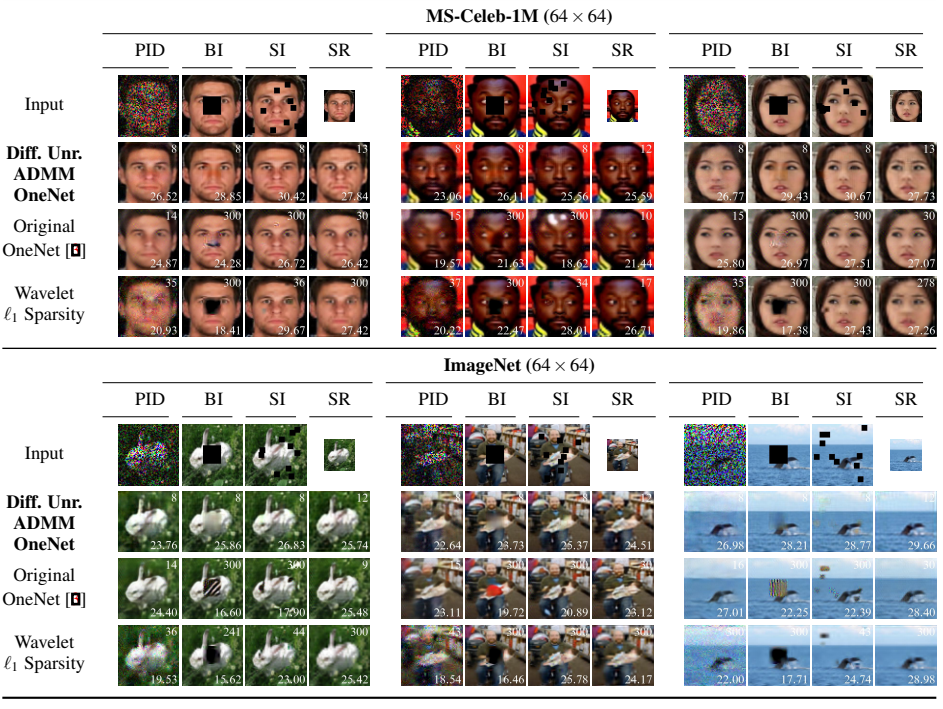
Interestingly, super resolution seems to be more challenging. Due to the small $d^{(n)}$ and large $\rho^{(n)}$, it appears to require more ADMM iterations than $T = 8$ used in training. We found

[3] https://github.com/rick-chang/OneNet

[4] https://www.msceleb.org

[5] http://image-net.org

Table 2: Sample qualitative and quantitative experimental results for the MS-Celeb-1M and ImageNet data sets. PID, BI, SI and SR stand for pixelwise inpainting-denoising, blockwise inpainting, scattered inpainting and super resolution, respectively. The PSNR values against the ground truth and the number of performed ADMM iterations are shown in the bottom-right and top-right corners for each image. Our contribution is highlighted in bold.



empirically that if we let $11 \leq T \leq 13$ during testing here, results for our scheme improve considerably, beating OneNet and approaching the level of Wavelet sparsity on both data sets.

In general, we can say that our scheme successfully avoids complete failure cases often observed for its two rivals, due to effectively resolving the concerns mentioned in Sect. 1.

Finally, we also tested whether adversarial training as in (10), (11) improves our results, and observed the exact same phenomenon as in OneNet: the generated images became sharper, but PSNR scores degraded.

5 Conclusions

We proposed a combination of the Unrolled Optimization and the OneNet frameworks, exploiting supervised learning for multiple image formation models using differentiable ADMM. Considering pixelwise inpainting-denoising, blockwise inpainting, scattered inpainting and super resolution, our method either outperformed or tied the original OneNet procedure and Wavelet sparsity; while using a much smaller ADMM iteration count.

For future work, extending the scope of this paper to dense linear operators (*e.g.*, compressive sensing), video data, adversarial performance measures (*e.g.*, Fréchet Inception Dis-

tance) and larger image sizes all seem worthy to investigate.

Our method is a step towards achieving multi-task learning. It has relevance when storing and running pretrained weights of several problem specific deep neural networks is infeasible, *e.g.*, when doing edge computing on mobile devices and self-driving cars.

Acknowledgments

This research has been supported partially by the European Union, co-financed by the European Social Fund. Grant Nos. are EFOP-3.6.3-VEKOP-16-2017-00001, EFOP-3.6.3-VEKOP-16-2017-00002.

References

[1] A. Beck and M. Teboulle. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIIMS*, 2(1):183–202, 2009.

[2] S. Boyd, N. Parikh, et al. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *FTML*, 3(1):1–122, 2011.

[3] JH. R. Chang, CL. Li, et al. One Network to Solve Them All–Solving Linear Inverse Problems Using Deep Projection Models. In *ICCV*, pages 5888–5897. IEEE, 2017.

[4] Y. Chen, W. Yu, et al. On learning optimized reaction diffusion processes for effective image restoration. In *CVPR*, pages 5261–5269. IEEE, 2015.

[5] CC. Chiu, T. N. Sainath, et al. State-of-the-art speech recognition with sequence-to-sequence models. In *ICASSP*, pages 4774–4778. IEEE, 2018.

[6] J. Devlin, MW. Chang, et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805*, 2018.

[7] S. Diamond, V. Sitzmann, et al. Unrolled Optimization with Deep Priors. *arXiv:1705.08041*, 2017.

[8] K. Gregor and Y. LeCun. Learning Fast Approximations of Sparse Coding. In *ICML*, pages 399–406. Omnipress, 2010.

[9] I. Gulrajani, F. Ahmed, et al. Improved Training of Wasserstein GANs. In *NIPS*, pages 5767–5777, 2017.

[10] Y. Guo, L. Zhang, et al. MS-Celeb-1M: A Dataset and Benchmark for Large Scale Face Recognition. In *ECCV*. Springer, 2016.

[11] K. He, X. Zhang, et al. Deep Residual Learning for Image Recognition. In *CVPR*, pages 770–778. IEEE, 2016.

[12] K. He, G. Gkioxari, et al. Mask R-CNN. In *ICCV*, pages 2980–2988. IEEE, 2017.

[13] J. R. Hershey, J. L. Roux, et al. Deep Unfolding: Model-based Inspiration of Novel Deep Architectures. *arXiv:1409.2574*, 2014.

[14] J. Howard and S. Ruder. Universal Language Model Fine-tuning for Text Classification. In *Proc. 56th ACL*, volume 1, pages 328–339, 2018.

[15] D. P Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980*, 2014.

[16] A. Krizhevsky, I. Sutskever, et al. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*, pages 1097–1105, 2012.

[17] Y. LeCun, Y. Bengio, et al. Deep learning. *Nature*, 521(7553):436, 2015.

[18] M. E. Peters, M. Neumann, et al. Deep contextualized word representations. *arXiv:1802.05365*, 2018.

[19] O. Russakovsky, J. Deng, et al. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015.

[20] S. Sabour, N. Frosst, et al. Dynamic Routing Between Capsules. In *NIPS*, pages 3856–3866, 2017.

[21] J. Sun, H. Li, et al. Deep ADMM-Net for Compressive Sensing MRI. In *NIPS*, pages 10–18, 2016.

[22] D. Ulyanov, A. Vedaldi, et al. Instance Normalization: The Missing Ingredient for Fast Stylization. *arXiv:1607.08022*, 2016.

[23] W. Xiong, L. Wu, et al. The Microsoft 2017 conversational speech recognition system. In *ICASSP*, pages 5934–5938. IEEE, 2018.