Cost Function Unrolling in Unsupervised Optical Flow

Gal Lifshitz and Dan Raviv

Abstract—Steepest descent algorithms, which are commonly used in deep learning, use the gradient as the descent direction, either as-is or after a direction shift using preconditioning. In many scenarios calculating the gradient is numerically hard due to complex or non-differentiable cost functions, specifically next to singular points. This has been commonly overcome by increased DNN model sizes and complexity. In this work we propose a novel mechanism we refer to as Cost Unrolling, for improving the ability of a given DNN model to solve a complex cost function, without modifying its architecture or increasing computational complexity. We focus on the derivation of the Total Variation (TV) smoothness constraint commonly used in unsupervised cost functions. We introduce an iterative differentiable alternative to the TV smoothness constraint, which is demonstrated to produce more stable gradients during training, enable faster convergence and improve the predictions of a given DNN model. We test our method in several tasks, including image denoising and unsupervised optical flow. Replacing the TV smoothness constraint with our loss during DNN training, we report improved results in all tested scenarios. Specifically, our method improves flows predicted at occluded regions, a crucial task by itself, resulting in sharper motion boundaries.

1 INTRODUCTION

The $L^1$ norm of the gradients of a given function, also known as its Total Variation (TV) semi-norm, and more specifically its estimation, has been a significant field of study in robust statistics [1], [2]. Even prior to the sweeping AI era, many approaches to Computer Vision problems, such as image restoration, denoising and registration [3], [4], [5], have used a TV regularizer, as it represents the prior distribution of pixel intensities of natural images [6]. Its main advantage is its robustness to small oscillations such as noise while preserving sharp discontinuities such as edges.

Historically, solving the TV problem has been a challenging task, mainly due to the non-differentiability of the $L^1$ norm at zero. As a result, differentiable relaxations for $L^1$ have been proposed, such as the Huber [7] and Charbonnier [8] loss functions. Further approaches involved iterative mechanisms which have demonstrated superior results in many tasks [3], [5], [9]. Introducing trainable Deep Neural Networks (DNNs) to Computer Vision has brought a significant performance boost, and specifically the commonly used auto-derivation frameworks [10], [11], [12] have provided quick and easy tools to solve complex functions. Indeed, these frameworks commonly bypass the $L^1$ non-differentiability either by differentiable relaxations or by simply using one of its sub-gradients, i.e.

$$\frac{d|x|}{dx} = \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases}$$

which are highly discontinuous. Inspired by the vast research carried for many years before the deep learning era, we claim that non-differentiable cost functions should be dealt with greater care.

Proximal iterative methods for learning complex functions have shown superiority in many Image Processing, as well as Computer Vision tasks [14], [15], [16]. In these works, axiomatic optimization algorithms are unrolled and each iteration is mapped to a sub-network having its own learnable parameters. The resulting network architectures consist of task specific stacked modules which may be trained end-to-end in both supervised and unsupervised manners. Performance boost has been achieved either by increasing the number of stacked modules [14] or introducing intermediate modules [17], [18], both resulting in increased model complexity and size.

In this paper, to the best of our knowledge, we are the first to shift unrolling to the cost function, improving the optimization of a given DNN model while preserving its architecture, as opposed to baselines featuring unrolled architectures. Specifically, we present a novel pipeline we refer to as Cost Unrolling, in which we obtain a differentiable alternative for the the commonly used TV smoothness regu-
lizarization. Following the well known Alternating Direction Method of Multipliers (ADMM) \[9\] algorithm, our initial optimization problem is decomposed into a set of iterative sub-problems, each one featuring a differentiable quadratic cost function. Our unrolled cost function is then defined as the accumulation of all unrolled sub-cost functions. Gradients generated during training are shown to be much more stable in the regions where the gradients of the original cost function are hard to evaluate or undefined, improving convergence.

We demonstrate the effectiveness of our unrolled cost function in synthetic experiments including image denoising, as well as the unsupervised optical flow task, which is the focus of this paper. The lack of available labeled data has made unsupervised optical flow learning a significant field of interest. Unsupervised training commonly consist of a measure for image consistency and TV smoothness regularization. Despite extensive research carried in the field, a major challenge which remains open is occlusions. The occluded pixels, pixels within a reference image with no correspondence at the target image, are masked out when measuring image consistency. As a result, occluded regions, which are highly correlated to motion boundaries, are mostly dominated by the gradients of the smoothness constraint during model training. Interestingly, we find our method particularly beneficial at detecting motion in the occluded regions.

Our method is introduced to two recently published unsupervised optical flow baselines ARFlow \[13\] and SMURF \[19\]. Replacing their used TV smoothness constraint with our unrolled cost during all phases of training produces improved results on both MPI Sintel \[20\] and KITTI 2015 \[21\] unsupervised optical flow benchmarks. Particularly, we report EPE reduced by up to 15.82% on occluded pixels, where the smoothness constraint is dominant, allowing the detection of much sharper motion boundaries, as is highly visible in figures \[1\] and \[6\].

1.1 Contributions
We summarize our contributions as follows:

- Present the Cost Unrolling mechanism for deep learning of TV regularized problems.
- Demonstrate our proposed pipeline in the image denoising and unsupervised optical flow domains.
- Report improved results on unsupervised optical flow benchmarks, achieved without modifying model architecture or increasing complexity.

2 RELATED WORK
Total Variation (TV) minimization has been a significant field of research in robust statistics \[1\]. It was introduced to Image Processing and Computer Vision tasks by \[3\] and showed superior performance over previous methods mainly due to its edge preserving property. However, solving TV regularized problems has shown to be non-trivial, mainly as the $L^1$ norm is non-differentiable at zero \[2\]. Differentiable relaxations were proposed by \[4\], \[5\], \[22\], however iterative methods have proven more effective \[4\], \[11\], \[5\]. A vast research regarding $L^1$ relaxations is given in \[23\].

Introducing trainable DNN models to Computer Vision has brought a significant performance boost to many tasks, such as image denoising \[24\], graph matching \[25\] and optical flow \[26\]. The lack of labeled data available for training has pushed many DNN algorithms towards unsupervised learning. Not surprisingly, the TV smoothness regularization can be found in the cost functions of many unsupervised works both past and recent. As DNN algorithms use gradient descent algorithms for optimization, the non-differentiability of the $L^1$ norm at zero has been commonly bypassed either by differentiable approximations \[27\] or by using its sub-gradients (eq. \[1\], \[17\], \[28\], \[29\]).

Regularizer learning, a recent field of research, attempts to learn regularizers through incorporating designated learnable units. Learned analysis operators for regularized image restoration were proposed in \[30\]. Variational Networks, introduced in \[31\], featured a multi-residual \[32\] structure inspired by the proximal gradient update steps.

Algorithm unrolling is an iterative structured approach for learning complex functions, in which iterative operations derived from axiomatic algorithms determine the model architecture. A vast review of the topic is presented in \[15\]. Axiomatic algorithms, such as the Alternating Direction Method of Multipliers (ADMM) \[9\], Half Quadratic Splitting (HQS) \[33\] or Primal-Dual method \[34\], have inspired many deep algorithm unrolling frameworks. Indeed, ADMMNet \[35\] introduced a DNN model for compressed sensing with architecture derived from ADMM, USRNet \[16\] learned Single Image Super-Resolution (SISR) using a deep framework inspired by HQS and \[14\] demonstrated an iterative structure derived from a Primal-Dual solver, aimed to refine the optical flow predicted by a FlowNet \[35\] model.

Differentiable Programming (DP) is another approach for transferring axiomatic models to differentiable models, such as DNNs, demonstrating promising results. Indeed, \[37\] proposed unsupervised clustering using an NN model, and \[38\] proposed a learnable composition generator for single image dehazing, trained using an adversarial loss.

However, all of these frameworks feature closed form update steps derived from the learned objective function, which are mapped into task specific sub-networks, increasing model size and complexity. As opposed to previous works, we demonstrate improving the optimization of a given DNN model while preserving its architecture. We accumulate the iterative sub-objectives, unrolled following the ADMM algorithm \[9\], to obtain a differentiable alternative to the original non-differentiable TV smoothness regularization. While previous works designed dedicated unrolled learnable units, we present a pipeline that can be applied to any model architecture.

Learning in iterations has shown to be very effective at both image registration and optical flow tasks. While FlowNet and FlowNet 2.0 \[36\], \[39\] were the first to learn optical flow, they were surpassed in both supervised and unsupervised settings, by the well known PWC-Net \[40\] which featured a course-to-fine iterative scheme. RAFT \[26\] introduced a fully iterative architecture which enabled a significant performance boost, as well as its expansion to the unsupervised settings SMURF \[19\]. Further approaches,
such as [14], [17], [18], managed to improve the results of existing frameworks by adding intermediate modules and parameters for better flow refinement, which increase model size and complexity.

Here we demonstrate that unlike all other methods, improved predictions of a given model can be achieved simply through improving the computed gradients during training.

3 Cost Function Unrolling

A general formulation of the cost function used for unsupervised learning consists of a data term, measuring the likelihood of a given prediction over the given data, as well as a prior term, constraining probable predictions. In this work we consider the TV regularized unsupervised cost function and derive the iterative optimization steps for its minimization. These are then unrolled to obtain our novel smoothness regularizer. Denote by $\theta$ the set of trainable parameters of a DNN model $\mathcal{F}_\theta$, its predicted output $\mathbf{F}$ and the set of unlabeled training data $\mathcal{I}$. The unsupervised TV regularized cost function takes the form:

$$
\mathcal{L}(\mathbf{F}, \mathcal{I}) = \Phi(\mathbf{F}, \mathcal{I}) + \lambda \| \nabla \mathbf{F} \|_1
$$

where $\Phi$ is a differentiable function measuring the likelihood of $\mathbf{F}$ over the training data $\mathcal{I}$, $\nabla \mathbf{F}$ are its spatial gradients, $\| : \|_1$ is the $L^1$ norm and $\lambda$ is a hyperparameter controlling regularization.

3.1 Unrolling the Unsupervised Cost Function

Our goal is to minimize the objective function in (2). Following the ADMM [8] algorithm, we derive the iterative update steps minimizing (2). Let us examine the following optimization problem:

$$
\hat{\mathbf{F}} = \arg \min_{\mathbf{F}} \{ \Phi(\mathbf{F}, \mathcal{I}) + \lambda \| \nabla \mathbf{F} \|_1 \} \tag{3}
$$

We rewrite (3), introducing an auxiliary variable $\mathbf{Q}$:

$$
\hat{\mathbf{F}}, \hat{\mathbf{Q}} = \arg \min_{\mathbf{F}, \mathbf{Q}} \{ \Phi(\mathbf{F}, \mathcal{I}) + \lambda \| \mathbf{Q} \|_1 \}
\text{ s.t. } \mathbf{Q} = \nabla \mathbf{F}
\tag{4}
$$

The augmented Lagrangian is then constructed as follows:

$$
\mathcal{L}_\rho(\mathbf{F}, \mathbf{Q}, \mathbf{Q}) = \Phi(\mathbf{F}, \mathcal{I}) + \lambda \| \mathbf{Q} \|_1
+ \langle \mathbf{Q}, \nabla \mathbf{F} \rangle_2 + \frac{\rho}{2} \| \mathbf{Q} - \nabla \mathbf{F} \|_2^2
\tag{5}
$$

where $\mathbf{Q}$ is the Lagrange Multipliers tensor and $\rho$ is a penalty parameter. Substituting $\mathbf{Q} = \frac{\rho}{\lambda}$ for simplicity, we iteratively optimize $\{ \mathbf{F}, \mathbf{Q}, \beta \}$ through solving the sub-problems (6a), (6b) and update the multipliers (6c) in each iteration:

$$
\hat{\mathbf{F}} = \min_{\mathbf{F}} \left\{ \Phi(\mathbf{F}, \mathcal{I}) + \frac{\rho}{2} \| \mathbf{Q} - \nabla \mathbf{F} + \beta \|_2^2 \right\}
$$

$$
\hat{\mathbf{Q}} = \min_{\mathbf{Q}} \left\{ \lambda \| \mathbf{Q} \|_1 + \frac{\rho}{2} \| \mathbf{Q} - \nabla \mathbf{F} + \beta \|_2^2 \right\}
$$

$$
\beta = \beta + \langle \mathbf{Q} - \nabla \mathbf{F} \rangle
\tag{6c}
$$

The solutions to the problems in (6a) and (6b) are referred to as the ADMM update steps for $\mathbf{F}$ and $\mathbf{Q}$, respectively.

1. For the rest of the discussion, we expand the $L^1$, $L^2$ norm and dot product to tensors $\mathbf{A}, \mathbf{B}$ as $\| \mathbf{A} \|_1 = \sum_i |A_i|$, $\| \mathbf{A} \|_2^2 = \sum_i |A_i|^2$ and $\langle \mathbf{A}, \mathbf{B} \rangle = \sum_i A_i B_i$, where $A_i, B_i$ are corresponding elements of $\mathbf{A}, \mathbf{B}$.

Algorithm 1 Unrolled Smoothness Constraint

Inputs:
- Model prediction $\mathbf{F}$
- No. of cost function update steps $T$, $\{\alpha_t\}_{t=1}^T$
- Hyperparameters $\lambda, \rho$

1: procedure UNROLLED_SMOOTH($\mathbf{F}, T, \lambda, \rho$)
2: $\beta^{(0)} \leftarrow 0$
3: $\mathbf{Q}^{(0)} \leftarrow 0$
4: $\nabla \mathbf{F} \leftarrow \text{COMPUTEGRADIENT($\mathbf{F}$)}$
5: for $t \leftarrow 1, T$ do
6: $\ell(t)(\mathbf{F}) \leftarrow \frac{1}{2} \| \mathbf{Q}^{(t-1)} + \beta^{(t-1)} - \nabla \mathbf{F} \|_2^2$
7: $\mathbf{Q}^{(t)} \leftarrow S_{\lambda/\rho} \left( \nabla \mathbf{F} - \beta^{(t-1)} \right)$ ▷ Update $\mathbf{Q}$
8: $\beta^{(t)} \leftarrow \beta^{(t-1)} + \mathbf{Q}^{(t)} - \nabla \mathbf{F}$ ▷ Update $\beta$
9: end for
10: $\mathcal{L}_{\text{sm}}^T(\mathbf{F}) \leftarrow \frac{1}{T} \sum_{t=1}^T \alpha_t \ell(t)(\mathbf{F})$
11: end procedure

Output: Smoothness constraint $\mathcal{L}_{\text{sm}}^T(\mathbf{F})$

Algorithm 2 Model Training

Inputs:
- Model parameters $\theta$, initialized DNN model $\mathcal{F}_\theta$
- Set of training data $\mathcal{I}$
- Likelihood measure $\Phi$
- Smoothness hyperparameters $T, \lambda, \rho$

1: procedure MODELTRAINING($\mathcal{F}_\theta, \mathcal{I}, \Phi$)
2: for $k \leftarrow 1, K$ do ▷ Training iterations
3: $\mathcal{B} \leftarrow \{ x_i \sim \mathcal{I} \}_{i=1}^N$, ▷ Sample data batch
4: $\mathbf{F} \leftarrow \mathcal{F}_\theta(\mathcal{B})$, ▷ Model Prediction
5: $\mathcal{L}_{\text{sm}}^T(\mathbf{F}) \leftarrow \text{UNROLLED_SMOOTH($\mathbf{F}, T, \lambda, \rho$)}$
6: $\mathcal{L}_{\text{dif}}(\mathbf{F}, \mathcal{B}) \leftarrow \Phi(\mathbf{F}, \mathcal{B}) + \mathcal{L}_{\text{sm}}^T(\mathbf{F})$

Update Parameters:
7: $\theta \leftarrow \text{optimizer($\theta, \mathcal{L}_{\text{dif}}$)}$
8: end for
9: end procedure

Output: Trained DNN model $\mathcal{F}_\theta$

3.2 Solving the Sub-Optimization Problems

Solutions to the sub-optimization problems in equations (6a) and (6b) are now discussed. While the problem in equation (6a) has a closed form solution, deriving a closed form solution for the problem in (6b) is not trivial, as $\Phi$ can be any function. Substituting $\mathbf{C} = \nabla \mathbf{F}$, (6b) reduces to:

$$
\min_{\mathbf{Q}} \left\{ \lambda \| \mathbf{Q} \|_1 + \frac{\rho}{2} \| \mathbf{Q} - \mathbf{C} + \beta \|_2^2 \right\}
$$

$$
= \min_{\mathbf{Q}} \left\{ \lambda \sum_i |Q_i| + \frac{\rho}{2} \sum_i |Q_i - C_i + \beta_i|^2 \right\}
$$

$$
= \sum_i \min_{Q_i} \left\{ \lambda |Q_i| + \frac{\rho}{2} |Q_i - C_i + \beta_i|^2 \right\}
\tag{7}
$$

i.e. the problem is separable and we may solve independently for each tensor element. The objective in (7) is con-
Fig. 2. \textit{Unrolled smoothness constraint block diagram.} In each training iteration, given a flow prediction, its spatial gradient $\nabla \mathbf{F}$ is derived. Initialized at $\beta^{(0)} = Q^{(0)} = 0$, the Soft Thresholding and Multipliers Update steps are carried for update steps $t \in \{1, ..., T\}$ to produce $(Q^{(t)}, \beta^{(t)})_{t=1}^{T}$, which are then used together with $\nabla \mathbf{F}$ to construct our smoothness constraint $\mathcal{L}_{\text{sm}}^T$ as in (11).

\textbf{3.3 Unrolled Cost Function}

Inspired by the ADMM update steps given in equations (6a, 6b, 6c), our unrolled smoothness constraint consists of the normalized accumulation of all the iteratively generated constraints in the form of (10), mainly:

\[
\mathcal{L}_{\text{sm}}^T(\mathbf{F}) = \frac{1}{T} \sum_{t=1}^{T} \alpha_t \ell^{(t)}(\mathbf{F})
\]

\[
= \rho \frac{1}{2} \sum_{t=1}^{T} \alpha_t \| Q^{(t-1)} + \beta^{(t-1)} - \nabla \mathbf{F} \|^2_2
\]

where:

\[
Q^{(t)} = S_{\lambda/\rho} \left( \nabla \mathbf{F} - \beta^{(t-1)} \right)
\]

\[
\beta^{(t)} = \beta^{(t-1)} + \left( Q^{(t)} - \nabla \mathbf{F} \right)
\]

$Q^{(0)}, \beta^{(0)}$ are initialized to zeros, $T$ is a hyperparameter stating the number of update steps carried, and $\{ \alpha_t \}$ adjust the weight of each step.

Suppose we limit the number of update steps to $T = 2$. Following (12a), (12b) yields:

\[
Q^{(1)} = S_{\lambda/\rho} \left( \nabla \mathbf{F} - \beta^{(0)} \right) = S_{\lambda/\rho} \left( \nabla \mathbf{F} \right)
\]

\[
\beta^{(1)} = \beta^{(0)} + \left( Q^{(1)} - \nabla \mathbf{F} \right) = S_{\lambda/\rho} \left( \nabla \mathbf{F} \right) - \nabla \mathbf{F}
\]

which results in the following truncated smoothness constraint:

\[
\mathcal{L}_{\text{sm}}^2(\mathbf{F}) = \frac{\rho}{4} \left\{ \alpha_1 \| \nabla \mathbf{F} \|^2_2 + 2 \alpha_2 \| S_{\lambda/\rho} \left( \nabla \mathbf{F} \right) - \nabla \mathbf{F} \|^2_2 \right\}
\]

Recall that minimizing the TV of a function promotes sparse output gradients. In fact, eq. (14) can be viewed as Tikhonov regularization over $\nabla \mathbf{F}$, which implicitly yields a differentiable alternative for TV minimization. Specifically, note that its minimization involves the minimization of the $L^2$ distance between the true and sparsified output gradients. Thus, it promotes sparse output gradients. Experiments presented in sections 4.3.5 and 4.3.6 provide empirical support for the optimality of $T = 2$, mainly as it manages to both promote sparse output gradients and maintain computational complexity.

Interestingly, our truncated constraint given in eq. (14) can be considered as a further relaxation of the well known Huber loss [7]. The Huber loss function (given in eq. (17)) is essentially the Moreau envelope of the $L^1$ loss function:

\[
r_H(\nabla \mathbf{F}; k) = \min_{\mathbf{y}} \| \mathbf{y} - \nabla \mathbf{F} \|^2_2 + k \| \mathbf{y} \|_1
\]

\[
= \| S_k(\nabla \mathbf{F}) - \nabla \mathbf{F} \|^2_2 + k \| S_k(\nabla \mathbf{F}) \|_1
\]

The main difference between ours and the Huber loss functions is the $L^1$ norm replaced by $L^2$, gaining $L^2$ behavior everywhere as opposed to Huber. We claim that this further relaxation featured in our method, inspired by the ADMM update steps, enables faster convergence and thus improves DNN training. We verify this empirically through rigorous experimentation in section 4.

In conclusion, our modified differentiable cost function takes the form:

\[
\mathcal{L}_{\text{diff}}(\mathbf{F}, \mathcal{T}) = \Phi(\mathbf{F}, \mathcal{T}) + \mathcal{L}_{\text{sm}}^T(\mathbf{F})
\]
Following ADMM, we have derived a quadratic regularizer, which consists of soft thresholding operations, thus essentially enforces $L^1$ minimization. Note that here, instead of explicitly refining $F$ in each iteration as suggested by the iterative ADMM update steps in (6a),(6b),(6c), we optimize all iteratively generated smoothness constraints simultaneously given a model prediction. Our proposed smoothness constraint computation and training scheme are summarized in algorithms 1 and 2 respectively. A block diagram is given in figure 2. Note that our method affects only the training phase of a given DNN model, hence no complexity is added during inference. Interestingly, as demonstrated in section 4, setting $T = 2$, we manage to preserve time and memory consumption during training as well.

4 EXPERIMENTATION

We study the effectiveness of our method through conducting experiments in three domains: piece-wise constant (PC) signal prediction (section 4.1), image denoising (section 4.2) and unsupervised optical flow (section 4.3). All of our models are implemented in Pytorch [10].

We further compare our unrolled cost to standard TV smoothness regularization, as well as $L^1$ relaxations, Huber [7] and Charbonnier [8] loss functions. The Huber regularizer over a prediction $x$ is defined as $R_H(x; k) = \sum_i r_H(x_i; k)$, where:

$$ r_H(x_i; k) = \begin{cases} \frac{1}{2}x_i^2, & |x_i| < k \\ k|x_i| - \frac{1}{2}k^2, & |x_i| \geq k \end{cases} \tag{17} $$

and $x_i$ are the elements of $x$. It aims to take advantage of the $L^1$ norm robustness to outliers, but suggests better convergence replacing $L^1$ by $L^2$ for points near its optimum. The Charbonnier regularizer is defined as $R_C(x) = \sum_i r_C(x_i)$, where:

$$ r_C(x_i) = \sqrt{x_i^2 + \epsilon^2} \tag{18} $$

Following the discussion in [23], we focus on small $k, \epsilon$ values. Note that both functions approach the $L^1$ norm only for $k, \epsilon \to 0$, and resemble the $L^2$ norm otherwise. We show that our iterative approach is superior to these $L^1$ relaxations, achieving improved results and faster convergence.

4.1 Piece-wise Constant (PC) Signal Prediction

Firstly, we perform a series of experiments over synthetic 1D Piece-wise Constant (PC) signals in order to study the generated gradients and convergence of our smoothness constraint. In each experiment, we optimize a small NN model to predict a randomly generated full PC signal given only a fixed fraction of its samples, performing GD iterations until convergence. Each cost function consists of a data term, comparing the samples and corresponding predictions, as well as a smoothness regularizer. Note that although the data term is supervised, it contains knowledge of the original signal only at the sampled points. As a result, the full signal reconstruction is heavily dependent on the choice of smoothness term. In fact, these experiments are rather simplified 1D reductions of what is commonly known in unsupervised registration tasks as occluded regions featuring no correspondence, making the smoothness constraint primary during training on these regions.

4.1.1 Implementation

We generate 1D PC signals $y(x), x \in [a,b]$ as linear combinations of random rectangular pulses, and sample them uniformly to generate our data $\{(x_i, y_i)\}_{i=0}^{N-1}$ where $y_i = y(x_i)$. In each experiment, we optimize a small NN model, consisting of 4 FC layers to predict the full signal given its samples, using GD iterations. Defining our data term $\Phi (f, y) = \frac{1}{N} \sum_{i=0}^{N-1} (f_i - y_i)^2$, where $\{f_i\}_{i=0}^{N-1}$ are the...

![Fig. 3. PC signal prediction - training. In each experiment we train a simple model to predict a full randomly generated PC 1D signal given a fraction of its samples and smoothness regularization. We train using our method, as well as standard TV and common $L^1$ relaxations. Displayed are mean and standard deviation of gradient norms (b) and prediction errors (c) measured during training while performing several independent experiments of the best configurations. An example of gradients recorded at a selected point is given in (a).](image-url)
Fig. 4. Image denoising qualitative examples. Displayed are images corresponding to two example scenarios - Zebra (top) and Peppers (bottom). From left to right: original, corrupt and reconstructed images obtained by TV regularization (DIP-TV) and our method. The corrupt images are labeled with the added noise levels, and each reconstruction is labeled with its corresponding PSNR [dB] with respect to the original image. Patches and corresponding pSSIM scores are highlighted in colored bounding boxes.

Table 2: Image denoising qualitative examples results. Both PSNR [dB] and pSSIM scores for selected patches of interest are summarized for the two scenarios given in fig. 4. The added noise level is described by $\sigma$.

| Image     | $\sigma$ | Method | PSNR [dB] | pSSIM (blue) |
|-----------|----------|--------|-----------|--------------|
| Zebra     | 50       | DIP-TV | 24.71     | 0.6973       |
|           |          | Ours   | 24.83     | 0.7309       |
| Peppers   | 75       | DIP-TV | 23.68     | 0.5032       |
|           |          | Ours   | 23.92     | 0.5433       |

4.1.2 Results

Table 1 summarizes measured final gradient norms and prediction errors averaged over our conducted experiments of our best configurations. Corresponding gradient norms and prediction errors recorded during training, as well as an example of gradients recorded at a selected point are given in figures 3(b),(c),(a) respectively. Recall that TV regularization promotes piece-wise constant solutions. In fact, one of its most notable shortcomings is the staircasing phenomenon\[2\], in which solutions are random piece-wise constant signals featuring equal TV. This severe instability is caused by the discontinuous nature of the TV gradients generated during training. Indeed, the staircasing phenomenon is highly evident in our TV experiments, in terms of severe gradient oscillations and non-decreasing gradient norm shown in figures 3(a),(b). More visualizations are given in Appendix A (available online). On the other hand, our smoothness constraint manages to promote deterministic, smooth but edge preserving solutions, with its gradients rapidly decreasing to zero, enabling the fastest convergence.

An extensive hyperparameter study of our proposed method, as well as Huber and Charbonnier is given in Appendix A.2 (available online). The best configurations are summarized in figure 3 revealing perhaps the most significant benefit of our smoothness constraint. Both Huber and Charbonnier essentially relax the $L^1$ loss function by modifying its shape around its optimum making it differentiable. Indeed, this enables gradients generated during training to decay towards reaching the optimum as opposed to TV, as is seen in figures 3(a),(b). Despite being made differentiable, training using the proposed $L^1$ relaxations still suffers from evident oscillations as a result of their constructed shapes, which result in slower convergence.

We optimize our model using our unrolled cost and compare to standard TV, Huber and Charbonnier, measuring the prediction error and the backpropagating gradients’ norm during training. These enable studying and comparing the behavior of the tested loss functions. The prediction error is defined as $\frac{1}{N} \sum_{i=0}^{N-1} |f_i - y_i|$, where $f_i, y_i$ are the predicted and ground-truth full signals of length $L_i$, respectively. The gradient’s norm is computed as $\frac{1}{L} \sum_{i=0}^{L-1} \left\| \nabla \mathcal{L} \right\|_F$, where $\mathcal{L}$ is the overall loss function used for training. In order to resemble actual training of an NN model, all gradients were computed using auto-differentiation. We conduct several independent experiments, during which we record the backpropagating gradients as well as the prediction errors. Further visualizations as well as implementation details and hyperparameter examinations are given in Appendix A (available online).
Indeed, as shown in Appendix A.2 (available online), the convergence rate and errors of both Huber and Charbonnier vary heavily as function of the chosen $k, \epsilon$ values. Instead of letting a DNN model to minimize the difficult (perhaps relaxed) TV smoothness constraint, we de facto provide further guidance during training, through forcing $L^2$ similarity between the true and explicitly computed sparsified versions of its prediction gradients. In fact, explicitly computing the sparsified output gradients during the forward pass replaces the difficult gradient computation during backpropagation. This, together with the well behaved $L^2$ minimization, produces the most stable gradients during training, featuring a rapid decrement. As is demonstrated in figure 5, our smoothness constraint converges to its optimum more than two times faster than all other baselines.

### 4.2 Image Denoising

Broadening the demonstration of our proposed method, we test our unrolled smoothness constraint in image denoising. We recreate several experiments from the well known Deep Image Prior (DIP) [41] baseline, which were furtherly enhanced in [42] by adding TV regularization (DIP-TV), particularly in low SNRs. The proposed baseline consists of training the weights of a DNN model $F_\theta$ to predict a clear image $\hat{x}$ given a noisy image $y$ and fixed random noise $z$ as its input. Optimizing the model parameters, it has been shown that the implicit regularization obtained by its structure (DIP), as well as the added explicit TV regularization (DIP-TV), enable the prediction of visually pleasing natural images in low SNRs. Using $\Phi(\hat{x}, y) = \|\hat{x} - y\|_2^2$, where $\hat{x} = F_\theta(z)$ is the model output, our used cost function takes the form:

$$L_{dis}(\hat{x}, y) = \|\hat{x} - y\|_2^2 + L_{sm}(\hat{x}; \lambda, \rho) \tag{20}$$

We optimize using both our unrolled cost and TV. Replacing TV regularization with our unrolled smoothness constraint, our method achieved improved results, both quantitatively and qualitatively.

#### 4.2.1 Implementation

We remain consistent with the scheme described in [41, 42]. A U-Net [43] shaped CNN model with convolution layers added to its skip connections is selected. Its parameters are optimized using an Adam optimizer, given a noisy image and random noise as its input, which remain fixed during training. In each scenario, a noisy image is generated using Additive White Gaussian Noise (AWGN) with variance $\sigma^2$. The used test data is a combination of 14 grayscale and color images commonly used in image denoising baselines (see Appendix B, available online). Our cost function consists of a data term as described in eq. (20), as well as a smoothness regularizer. We compare images reconstructed using standard TV and our smoothness constraint in each experiment. Further implementation details are given in Appendix B (available online).

#### 4.2.2 Results

Results averaged over our used test set are given in table 3 (a detailed collection of results for each scenario is provided in Appendix B (available online). Furthermore, results of two example scenarios are given in table 2 and figure 4 (further qualitative results are given in Appendix B, available online). Our method produces improved results on the test set, particularly at low SNRs. Moreover, our method demonstrated improved restoration capabilities of edges in piece-wise smooth regions (see examples given in fig. 4). In order to quantify this, we measure the Structural Similarity coefficient [44] between original and reconstructed images, specifically on selected image patches of interest (pSSIM). The used patches and corresponding pSSIM coefficients are highlighted in colored bounding boxes (fig. 4) and summarized in table 2. In conclusion, our method demonstrates improved PSNR particularly in low SNRs, and improved image boundaries preservation, increasing pSSIM by up to 10%.

#### 4.3 Unsupervised Optical Flow

We finally test our proposed method in the real-world unsupervised optical flow problem. We introduce our smoothness constraint to two well known recent unsupervised optical flow baselines, ARFlow [13] and SMURF [19], featuring two different flow backbones considered highly impactful. Here we demonstrate that replacing TV regularization with our unrolled smoothness constraint during training produces improved flow predictions of both models, particularly at the occluded regions.

Optical flow is defined as a mapping $F : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, which aligns pixels given a pair of images $I_1, I_2$. In the unsupervised optical flow setting, our data term takes the form $\Phi(F, I) = \varphi(I_1, I_2)$, where $\varphi$ is a differentiable photometric loss, measuring image consistency between a reference image and a target image warped using $F$. In order to remain consistent with previous optical flow baselines (including ARFlow and SMURF), masked TV regularization is used, replacing $\nabla F$ with $C = W \odot \nabla F$ in equation 2 and line 4 in algorithm 1 where $W$ is a deterministic importance matrix, aiming to decrease the penalty on object boundaries, and $\odot$ stands for point-wise multiplication. $W$ is defined as

$$W = \exp\left\{-|\nabla I_1|\right\}$$

$$= \exp\left\{-\alpha \left[ |\partial I_1| \odot |\partial I_1| \right] \right\} \tag{21}$$

3. Point-wise multiplication is done through broadcasting the channels of $W$ to corresponding channels of $\nabla F = \left[ \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial v}{\partial x}, \frac{\partial v}{\partial y} \right]$.  

| Method | Ave. PSNR [dB] vs. Noise Levels |
|--------|---------------------------------|
|        | $\sigma = 25$                  | $\sigma = 50$ | $\sigma = 75$ | $\sigma = 100$ |
| DIP-TV | 27.91                           | 24.72          | 22.16          | 20.13          |
| Ours   | 27.94                           | 24.73          | 22.22          | 20.22          |

**TABLE 3**

Summarized image denoising results. PSNRs measured between original and reconstructed images are averaged over the used test set, for each noise level. The measured average SNR levels (corresponding to $\sigma = 25, 50, 75, 100$) are 13.36, 7.84, 3.82 and 1.32 [dB].
Fig. 5. Unrolled cost vs. TV relaxations - qualitative examples. Given are Sintel (top) and KITTI (bottom) flows predicted by ARFlow and SMURF baselines, respectively, using our method vs. standard TV, Huber and Charbonnier baselines. The reference images and GT occlusion masks (white stands for occluded) are also given. Training using our smoothness regularizer produces flows which are significantly more accurate at the occluded regions which are correlated with motion boundaries.

| Baseline  | Method          | Sintel Clean Occ. | Sintel Final Occ. | KITTI 2015 Occ. |
|----------|-----------------|-------------------|-------------------|-----------------|
| ARFlow†  | Standard TV     | 8.77              | 9.97              | 6.51            |
|          | Charbonnier     | 8.96              | 10.10             | 6.23            |
|          | Unrolled Cost   | 7.91              | 9.16              | 5.48            |
| SMURF†   | Standard TV     | 8.88              | 10.46             | 6.43            |
|          | Charbonnier     | 8.89              | 10.48             | 6.63            |
|          | Unrolled Cost   | 8.24              | 9.88              | 5.62            |

where \( I_1 \) is a reference image and \( \alpha \) is a hyperparameter controlling the masking operation. In summary, our cost function used for unsupervised optical flow training is:

\[
L_{\text{dif}} (F, I) = \varphi (I_1, I^F_2) + L^T_{\text{sm}} (F; \lambda, \rho)
\]

4.3.1 Model Architectures

We test our method on two well known recent flow backbones, the light weight Pyramid, Warping, Cost volume (PWC-Net) \([40]\) and Recurrent All-pairs Field Transforms (RAFT) \([26]\), featured in the ARFlow \([13]\) and SMURF \([19]\) baselines, respectively.

The PWC-Net backbone, featured in the ARFlow baseline, roughly consists of a multi-scale feature extraction pyramid and a Warp, Correlation, Flow (WCF) unit which is shared across all pyramid levels. At each pyramid level, a coarse flow estimate from a higher level is upsampled and fed to the WCF unit, together with the corresponding level features to produce a finer flow estimate. The RAFT backbone, featured in the SMURF baseline, consists of feature and context extractors, a 4D cost volume capturing multi-scale all-to-all feature correlations, and a recurrent GRU-based update unit. The update unit performs iterative refinement of the predicted flow until convergence. We refer
| Method                | Sintel Clean | Sintel Final | KITTI 2015  |
|-----------------------|--------------|--------------|-------------|
|                       | Train        | Test         | Train       | Test         | (F1)         |
| CoT-AMFlow ★ [18]     |              | 3.96         | -           | 5.14         | -            | 10.34%       |
| STFlow ★ [45]         | 2.91         | 6.12         | (3.59)      | 6.63         | 3.56         | 13.83%       |
| UPFlow ★ [17]         | 2.33         | 4.68         | (2.67)      | 5.32         | 2.45         | 9.38%        |
| ARFlow-MV ★ [13]      | 2.73         | 4.49         | (3.69)      | 5.67         | (3.46)       | 11.79%       |
| SMURF ★ † [19]        | 1.71         | 3.15         | (2.58)      | 4.18         | (2.00)       | 6.83%        |
| DDFlow [46]           | 2.92         | 6.18         | (3.98)      | 7.40         | 5.72         | 14.29%       |
| UIFlow-train [29]     | 2.50         | 5.21         | (3.39)      | 6.50         | (2.71)       | 11.13%       |
| SimFlow [47]          | 2.86         | 5.92         | (3.57)      | 6.92         | 5.19         | 13.38%       |
| ARFlow [13]           | 2.79         | 4.78         | (3.73)      | 5.89         | 2.85         | 11.80%       |
| UnrolledCost (Ours)   | 2.75         | 4.69         | (3.61)      | 5.80         | 2.87         | 10.81%       |

**TABLE 5**

MPI Sintel & KITTI 2015 official unsupervised optical flow benchmark results. We report AEPE and F1 rates for recently published unsupervised methods on both the MPI Sintel [20] and KITTI 2015 [21] optical flow benchmarks. Methods featuring a different (heavier) backbone are marked with ★. Multi-frame results are marked with †. Brackets “()” indicate results of models trained using their validation set. Best methods in each category are in bold. Missing results are marked as “-“. By adapting our unrolled cost to the ARFlow [13] baseline we not only achieve improved results, but also report the best results for a PWC-Net based backbone on both benchmarks at the time of submission. Moreover, we find our method highly effective on the motion boundaries, as shown in the qualitative results (figures 1, 6).

Fig. 6. Qualitative benchmark results. We compare qualitative flow benchmark results of ours, the ARFlow [13] and UIFlow [29] baselines. Both ARFlow and UIFlow are methods adopting a PWC-Net [40] based backbone, reporting the best results on the MPI Sintel [20] and KITTI 2015 [21] benchmarks, respectively. We find adapting our unrolled cost to a PWC-Net based backbone outperforms both baselines, particularly at the motion boundaries.

Thus, occluded regions are learned using the smoothness constraint alone.

Finetuning both our ARFlow and SMURF models for the MPI Sintel benchmark is done using the official training set, consisting of 1,041 image pairs, each rendered with 2 levels of difficulty: the clean pass featuring shading only, and the final pass featuring also motion blur, defocus blur and atmospheric effects. For the KITTI 2015 benchmark, we use the KITTI 2015 multi-view extension for finetuning our models. We exclude frames related to validation from our training set, i.e. use only frames below 9 or above 12 in each scene. Further details regarding the train-test data splitting and training scheme used in our experiments are given in Appendix C (available online).

As carried in both ARFlow and SMURF baselines, self-supervision from pseudo-random augmentations is acquired during finetuning. Image and corresponding flow augmentations denoted

\[ T^{\text{img}} : I_k \mapsto \tilde{I}_k \]  
\[ T^{\text{flow}} : F \mapsto \tilde{F} \]  

the reader to [13, 19] for further details.

4.3.2 Datasets and Training

We train and evaluate our method on three well-known optical flow benchmarks: the synthetic Flying Chairs [36] and MPI Sintel [20], and real-world autonomous driving KITTI 2015 [21]. We rigorously follow the unsupervised training schemes of both ARFlow and SMURF baselines, consisting of pre-training and finetuning steps. Both baselines propose further finetuning in a multi-frame setting, which we do not carry.

Pre-training our ARFlow model for the MPI Sintel benchmark is done using the raw Sintel movie, consisting on 12,466 extracted image pairs, split according to scene shots. For the KITTI 2015 benchmark, we pre-train our ARFlow model on the KITTI raw dataset [48], consisting of real road scenes captured by a car-mounted stereo camera rig. Our SMURF model is pre-trained on the ‘training’ portion of the Flying Chairs dataset, for both benchmarks. No data labels are used in any step of our training. Occluded pixels are masked out during the evaluation of the data term in eq. (22) following the method suggested in [27] (unsupervised).
perform a combination of appearance, spatial and occlusion pseudo-random deformations. An initial flow estimate $\mathbf{F}_1$ is computed given a pair of images $I_1, I_2$. The image pair then undergoes an image augmentation $\mathcal{T}^{img}$ resulting in a deformed pair of images $\tilde{I}_1, \tilde{I}_2$. A second flow estimate $\mathbf{F}_2$ is generated next over the deformed images $\tilde{I}_1, \tilde{I}_2$ and a deformed flow $\tilde{\mathbf{F}}_1$ is obtained through applying the corresponding flow augmentation $\mathcal{T}^{flow}$ over $\mathbf{F}_1$. Finally, a self-supervision term measuring the error $\mathbf{F}_2 - \tilde{\mathbf{F}}_1$ is constructed and added to the cost function in equation (22) during finetuning. Further implementation details are given in Appendix C (available online).

### 4.3.3 Flow in Occluded Regions

Occluded pixels, i.e. pixels with no correspondence, are highly correlated with motion boundaries. Although crucial to many CV applications, performance on the occluded regions minorly affects the errors averaged over full images. These regions are of great interest in this work as they are majorly affected by the gradients of the smoothness constraint during training. We conduct several experiments measuring performance particularly in the occluded regions. We train our ARFlow and SMURF models on both the MPI-Sintel and KITTI 2015 data, using our smoothness constraint as well as standard TV (similar to ARFlow, SMURF baselines) and the $L^1$ relaxation methods described at the beginning of section 4. Measured validation AEPE rates of our unrolled cost, as well as the standard TV, Huber and Charbonnier methods are given in table 4. A qualitative example is given in figure 5. Our method outperforms all tested baselines. As expected, modifying the smoothness regularizer has little effect on the performance in the non-occluded regions ($Noc.$). However, measuring the performance on the occluded regions, our method reduces TV EPE rates by up to 15.82% enabling the detection of sharper motion boundaries with no added complexity as is highly visible in figure 5.

### 4.3.4 Comparison with State of the Art

We compare our trained ARFlow model with recently published unsupervised methods on the MPI Sintel and KITTI 2015 optical flow benchmarks in table 5. Qualitative examples are given in figure 6. The used measures are the Average End-Point-Error (AEPE) and F1 score, measuring outliers percentage (error > 3px). All reported MPI Sintel results are of models which used the entire official training data for both training and validation. KITTI 2015 results either share our training scheme of using the multi-view extension excluding frames related to validation, train using the KITTI raw dataset only [27] or use the entire multi-view extension for training, including the image pairs used for validation [29]. Comparing to the ARFlow [13] and UFlow [29] baselines is particularly interesting, as both are recently published baselines featuring a PWC-Net backbone over dual images only (like ours). Our method managed to improve the reported results of the ARFlow [13] baseline on both MPI Sintel and the KITTI 2015 unsupervised optical flow tasks.

### Table 6

| # Update Steps | Sintel Clean | Sintel Final |
|---------------|-------------|-------------|
| $T = 1$       | 2.43        | 3.25        |
| $T = 2$       | 2.35        | 3.19        |
| $T = 4$       | 2.35        | 3.16        |

**Number of cost function update steps.** Finetuning using several $T$ values, we find that increasing the number of update steps brings a performance boost.

### Table 7

| Baseline | Method       | Memory  | Runtime |
|----------|--------------|---------|---------|
| ARFlow   | Standard TV  | 4.326Gi | 51ms    |
|          | Unrolled Cost (Ours) | 4.285Gi | 51ms    |
| SMURF    | Standard TV  | 4.86Gi  | 800ms   |
|          | Unrolled Cost (Ours) | 4.80Gi  | 800ms   |

**Computational complexity.** Our method consumes slightly less memory during training and preserves runtime, hence replacing TV regularization with our unrolled cost enables improved results with no added complexity.
flow benchmarks, particularly at the motion boundaries, as is highly visible in all qualitative examples (figures 1 and 2). Moreover, our method reports the best results using a PWC-Net based backbone on dual images at the time of submission.

4.3.5 Number of Cost Function Update Steps

We study the effects of varying $T$ values, both on the performance of the accumulated gradients during training and on the performance of the trained model during inference. Cost function update terms $C^{(t)}$ and norms of sparsified flow gradients $Q^{(t)}$, recorded during training on both MPI Sintel and KITTI 2015, are given in figure 7. It is evident that both terms converge at $T = 2$, and that update terms generated at $T > 2$ have little effect on the gradients generated during training. This realization justifies our truncation in equation (16). Furthermore, we perform several experiments on the MPI Sintel data (see Appendix C.1.4, available online, for train-test data subsplit). We use an ARFlow model which has been pretrained on the MPI Sintel raw movie setting $T = 2$, but set various $T$ values during finetuning. The results of chosen $T$ values are compared in table 6 and a qualitative example is shown in figure 8. As expected, we find that increasing $T$ enables capturing finer motion details. However, it can be seen here as well that the gain for setting $T > 2$ is relatively small. We conclude that setting $T = 2$ is optimal as it both achieves good performance and enables preserving computational complexity during training, as is elaborated next (in 4.3.6).

4.3.6 Computational Complexity

Preserving computational complexity during training is achieved through the following. Firstly, we compute our smoothness term at the flow original resolution, which is 1/4 or 1/8 image resolution for ARFlow or SMURF, respectively (this is commonly done in flow baselines), as opposed to upsampling prior to TV computation. Secondly, we limit $T = 2$ update terms in our smoothness constraint as explained earlier. Our method does not increase model size, as no learned parameters are added. As our method deals with model training alone, it does not affects inference at all. We compare both memory consumption and training time of our method with standard TV in table 7. Measurements are carried using a single RTX2080 GPU. We use a batch size of 4 image pairs and an image size of $384 \times 832$ for the ARFlow baseline, and a batch of 1 image pair and image size of $368 \times 496$ for SMURF. Under these settings, we conclude that our method produces the results described above, while consuming slightly less memory and preserving training time.

5 CONCLUSIONS

We introduced the concept of Cost Unrolling, improving DNN optimization without changing its architecture or increasing complexity. Following our method, we derived a novel smoothness constraint replacing the non-differentiable TV semi norm, commonly used in unsupervised works. Our derived smoothness constraint is shown to produce more stable gradients during training, thus it enables faster convergence and improved predictions of a given DNN model. We have demonstrated the effectiveness of our method in a synthetic signal prediction problem as well as image denoising and real-world unsupervised optical flow. Our unrolled cost achieved superior results in all tested scenarios. We believe that the proposed framework can be applied on top of other model architectures for boosting their results next to non-differentiable optimum solutions.

ACKNOWLEDGMENTS

This work is partially funded by the Yitzhak and Chaya Weinstein Research Institute for Signal Processing.

REFERENCES

[1] P. Huber, J. Wiley, and W. InterScience, Robust statistics. Wiley New York, 1981.

[2] K. Bredies, K. Kunisch, and T. Pock, “Total generalized variation,” SIAM Journal on Imaging Sciences, vol. 3, no. 3, pp. 492–526, 2010.

[3] L. I. Rudin, S. Osher, and E. Fatemi, “Nonlinear total variation based noise removal algorithms,” Physica D: Nonlinear Phenomena, vol. 60, no. 1, pp. 259–268, 1992. [Online]. Available: https://www.sciencedirect.com/science/article/pii/016727899290242f

[4] J. Savage and K. Chen, “An improved and accelerated non-linear multigrid method for total-variation denoising,” International Journal of Computer Mathematics, vol. 82, no. 8, pp. 1001–1015, 2005. [Online]. Available: https://doi.org/10.1080/00207160500099634

[5] C. Zach, T. Pock, and H. Bischof, “A duality based approach for realtime tv-l 1 optical flow,” in Joint pattern recognition symposium. Springer, 2007, pp. 214–223.

[6] J. Huang and D. Mumford, “Statistics of natural images and models,” in Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149), vol. 1, 1999, pp. 541–547 Vol. 1.

[7] P. J. Huber, “Robust estimation of a location parameter: Annals mathematics statistics, 35,” 1964.

[8] P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud, “Determination edge-preserving regularization in computed imaging,” IEEE Transactions on image processing, vol. 6, no. 2, pp. 298–311, 1997.

[9] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” Found. Trends Mach. Learn., vol. 3, no. 1, p. 1122, Jan. 2011. [Online]. Available: https://doi.org/10.1561/2200000016

[10] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in Advances in Neural Information Processing Systems 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d\’Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[11] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” arXiv preprint arXiv:1408.5093, 2014.

[12] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/
A. Chambolle and T. Pock, “A first-order primal-dual algorithm for convex problems with applications to imaging,” Journal of mathematical imaging and vision, vol. 40, no. 1, pp. 120–145, 2011.

J. Sun, H. Li, Z. Xu et al., “Deep admm-net for compressive sensing mri,” in Advances in neural information processing systems, 2016, pp. 10–18.

A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, “Flownet: Learning optical flow with convolutional networks,” in Proceedings of the IEEE international conference on computer vision, 2015, pp. 2758–2766.

X. Peng, Y. Li, I. W. Tsang, H. Zhu, J. Lv, and J. T. Zhou, “Xai beyond classification: Interpretable neural clustering,” The Journal of Machine Learning Research, vol. 23, no. 1, pp. 227–254, 2022.

H. Zhu, X. Peng, Y. Chandrasekhar, L. Li, and J.-H. Lim, “DehazeNet: When image dehazing meets differential programming,” in ICCV, 2018, pp. 1234–1240.

E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, “Flownet 2.0: Evolution of optical flow estimation with deep networks,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 2462–2470.

D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, “PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume,” 2018.

D. Ulyanov, A. Vedaldi, and V. Lempitsky, “Deep image prior,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 9436–9454.

J. Liu, Y. Sun, X. Xu, and U. S. Kamilov, “Image restoration using total variation regularized deep image prior,” in ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019, pp. 7715–7719.

O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in International Conference on Medical image computing and computer-assisted intervention. Springer, 2015, pp. 234–241.

Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” IEEE transactions on image processing, vol. 13, no. 4, pp. 600–612, 2004.

L. Tian, Z. Tu, D. Zhang, J. Liu, B. Li, and J. Yuan, “Unsupervised learning of optical flow with cnn-based non-local filtering,” IEEE Transactions on Image Processing, vol. 29, pp. 8429–8442, 2020.

P. Liu, I. King, M. R. Lyu, and J. Xu, “Ddflow: Learning optical flow with unlabeled data distillation,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 8770–8777.

W. Im, T.-K. Kim, and S.-E. Yoon, “Unsupervised learning of optical flow with deep feature similarity,” in European Conference on Computer Vision. Springer, 2020, pp. 172–188.

A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset,” International Journal of Robotics Research (IJRR), 2013.
Dr. Dan Raviv is a faculty member in Tel Aviv University, Israel, in the Engineering department, school of Electrical engineering. His work is focused in the intersection between Machine learning, Computer Vision, and non-rigid Geometry. He graduated from the Technion, Israel Institute of Technology, in the Computer Science faculty, where he specialized in metric geometry for non-rigid objects, following several years working in MIT, USA, developing new learnable computer vision models. Dan is the recipient of the prestigious biennial award of SIAM/Imaging Sciences, a graduate of the Technion’s excellence program, and leading the Geometric Deep Learning laboratory in his department.