Deep Learning Project Update: Unrolled Primal-Dual Splitting based Deep Network for Optical Flow Estimation

Team Members: Nikola Janjušević
New York University Department of Electrical and Computer Engineering
Brooklyn, NY

Abstract

Deep neural networks (DNNs) are successful in many image processing and computer vision tasks, however, their construction still remains mostly as black box function approximators. Recent works in more principled avenues of DNN building suggests that state-of-the-art performance may be obtained by networks grounded in classical optimization algorithms, at a faction of the learned parameter count. The optical flow problem presents itself as ripe for testing the power of these interpretable architectures as it is a well studied problem in the classical literature yet still difficult to solve, and even more difficult to solve with real-time algorithms. In this work, we investigate an interpretably constructed DNN for optical flow estimation which we call PiBCANet. Preliminary results show promise on standard benchmarks against leading classical and deep learning methods. A Julia implementation is available online at *https://github.com/nikopj/UnrolledFlowNetworks.jl*

1 Introduction

In recent years, deep learning has achieved state-of-the-art results in low-level image-processing and computer-vision tasks. However, despite the extensive literature modeling this problems, leading deep neural networks (DNNs) in problems such as optical flow are still mostly constructed as black box function approximators. Unrolled networks are the subject of recent research activity for more principled DNN design. These networks derive their architectures from classical iterative optimization algorithms via parameterization of linear-operators within the algorithm and truncation to K iterations. They have been demonstrated to yield performance close to state-of-the-art black-box methods at a fraction of their learned parameter count, and in some-cases yield additional robustness due to their more interpretable parameters (Janjušević, Khalilian-Gourtani, and Wang 2021). In this work, we look at applying the algorithm unrolling technique to the optical flow problem primarily in hopes of achieving a more light-weight model to current popular methods such as PWC-Net (Sun et al. 2018) and RAFT (Teed and Deng 2020), which employ millions of parameters.

Optical flow is fundamental problem in computer vision, forming the backbones of such methods as frame interpolation, novel view-synthesis, and video compression. In its most general form, the optical flow problem consists of determining correspondences between sets of frames. In this work, we look at computing a dense, pixel-wise, optical flow field from two input images via a deep neural network. Formally, for input scalar-valued image pair u_0, u_1 defined on domain $\Omega = [0,1]^2$, we seek to find a flow field v such that

$$u_0(\mathbf{x}) = u_1(\mathbf{x} + \mathbf{v}(\mathbf{x})), \quad \forall\, \mathbf{x} \in \Omega. \tag{1}$$

Naturally, the flow field is vector valued function $\mathbf{v}(\mathbf{x}) \in \mathbb{R}^2$. The above constraint is known as the *brightness constancy assumption* (BCA) and foundation of classical variational optical flow methods. As stated, it only accounts for scalar-valued (grayscale) images and additional work is required for vector-valued (ex. RGB) images (see (Rakêt et al. 2011)).

In this work, we derive an optical flow esti-

mation DNN, BCANet, for input grayscale image pairs from an "unrolled" classical method operating on a regularized and linearized version of the brightness constancy assumption. Previous methods (Fan et al. 2018) have considered a similar classical optical flow algorithm for unrolling into a DNN, however, BCANet considers a more general analysis-dictionary regularization model and augmented non-linearities for added differentiability. Furthermore, (2018) focus on optical flow estimation for downstream video classification tasks and do not consider evaluation on the tough optical flow benchmark of the MPI-Sintel dataset (Butler et al. 2012).

2 Background and Related Works

2.1 Classical Methods

Since the work of Horn and Schunck (1981), classical optical flow methods are primarily based on a first order linearized BCA (1),

$$u_0(\mathbf{x}) = u_1(\mathbf{x}+\bar{\mathbf{v}}) + \nabla u_1(\mathbf{x}+\bar{\mathbf{v}})^\top(\mathbf{v}-\bar{\mathbf{v}}), \quad (2)$$

where $\bar{\mathbf{v}}$ is some initial flow field estimate in which the expansion is centered, and ∇ is the spatial gradient operator (i.e. $\nabla u(\mathbf{x}) = [u_x(\mathbf{x}) \quad u_y(\mathbf{x})]^\top \in \mathbb{R}^2$). Note that we've suppressed the dependence of $\bar{\mathbf{v}}, \mathbf{v}$ on $\mathbf{x}$ for ease of notation. As the linearized BCA (2) is only valid in a small range around $\bar{\mathbf{v}}$, methods employing it will need to iteratively update $\bar{\mathbf{v}}$.

In the continuous spatial domain we can formulate an optimization problem as follows. Given a pair of input images u_0, u_1 and initial flow-field estimate $\bar{\mathbf{v}}$,

$$\begin{aligned}&\underset{\mathbf{v}}{\text{minimize}} \;\; \lambda \int_\Omega \psi(\mathbf{v})\, d\mathbf{x} \\ &+ \int_\Omega |\nabla \bar{u}_1(\mathbf{x})^\top(\mathbf{v}-\bar{\mathbf{v}}) + \bar{u}_1(\mathbf{x}) - u_0(\mathbf{x})|\, d\mathbf{x},\end{aligned} \quad (3)$$

where $\bar{u}_1(\mathbf{x}) = u_1(\mathbf{x}+\bar{\mathbf{v}})$ and ψ is our regularizer, commonly chosen as the total-variation $\psi(\mathbf{v}) = \|\nabla \mathbf{v}\|_1$. The absolute-value function on the BCA term of (3) is chosen so as to be robust to outliers.

With TV-regularization, the above (3) is known as a TVL1 problem. In discretized form, without loss of generality consider square images $\mathbf{u}_0, \mathbf{u}_1 \in \mathbb{R}^{\sqrt{N}\times\sqrt{N}}$ and initial flow field $\bar{\mathbf{v}} \in \mathbb{R}^{\sqrt{N}\times\sqrt{N}\times 2}$. We write the TVL1 optical flow problem as,

$$\underset{\mathbf{v}}{\text{minimize}} \;\; \lambda\|\mathbf{D}\mathbf{v}\|_1 + \|\nabla \bar{\mathbf{u}}_1^\top(\mathbf{v}-\bar{\mathbf{v}}) + \bar{\mathbf{u}}_1 - \mathbf{u}_0\|_1, \quad (4)$$

where the inner-product is taken pixel-wise, $\bar{\mathbf{u}}_1 = \mathcal{W}(\mathbf{u}_1, \bar{\mathbf{v}})$ is $\mathbf{u}_1$ warped by the initial flow field and we use convolution with central difference stencils for the spatial derivatives.[1],

$$\begin{aligned}\nabla\mathbf{u} &= [\mathbf{D}_x\mathbf{u}\,;\,\mathbf{D}_y\mathbf{u}]\,,\\ \mathbf{D}_x\mathbf{u} &= [-1\,,\,0\,,\,1] * \mathbf{u},\\ \mathbf{D}_y\mathbf{u} &= [-1\,;\,0\,;\,1] * \mathbf{u},\\ \mathbf{D}\mathbf{v} &= [\mathbf{D}_x\mathbf{v}_x; \mathbf{D}_y\mathbf{v}_x; \mathbf{D}_x\mathbf{v}_y; \mathbf{D}_y\mathbf{v}_y].\end{aligned} \quad (5)$$

A popular method for tackling the discretized TVL1 problem (4) is primal-dual splitting (PDS)[2], whose iterates are given defined in Algorithm 1. Convergence of Algorithm 1 is guaranteed under mild conditions, including $\sigma\tau\|\mathbf{D}\|_2^2 \le 1$. To finish, we define $[\mathbf{z}]_{[-\lambda,\lambda]}[\mathbf{n}] = \max(-\lambda, \min(\lambda, \mathbf{z}[\mathbf{n}]))$ as the element-wise clipping function with threshold λ. Note that we can equivalently express the clipping function via Moreau's identity

$$[\mathbf{z}]_{[-\lambda,\lambda]} = \mathbf{z} - \mathrm{ST}(\mathbf{z}, \lambda),$$

where $\mathrm{ST}(\mathbf{w}, \lambda)[\mathbf{n}] = \mathrm{sign}(\mathbf{w}[\mathbf{n}])\max(0, |\mathbf{w}[\mathbf{n}]| - \lambda)$ is the element-wise shrinkage thresholding function with threshold λ. This fact will be useful for defining a differentiable non-linearities later.

Algorithm 1: TVL1-BCA (Burger, Dirks, and Frerking 2017)

Data: Image pair $\mathbf{u}_0, \mathbf{u}_1$, initial flow field $\bar{\mathbf{v}}$, step-sizes τ, σ, extrapolation parameter $\theta \in [0, 1]$.

for $k = 1, 2, \ldots$

- $\mathbf{w}^{(k+1)} = \left[\mathbf{w}^{(k)} + \sigma\mathbf{D}(\tilde{\mathbf{v}})\right]_{[-\lambda,\lambda]}$
- $\mathbf{v}^{(k+1)} = \mathrm{SST}(\mathbf{v}^{(k)} - \tau\mathbf{D}^\top\mathbf{w}^{(k+1)}, \tau)$
- $\tilde{\mathbf{v}} = \mathbf{v}^{(k+1)} + \theta(\mathbf{v}^{(k+1)} - \mathbf{v}^{(k)})$

Result: Flow field $\mathbf{v}$

[1] (5) described using MATLAB notation: comma for horizontal concatenation and semicolon for vertical concatenation.

[2] a.k.a the Chambolle Pock algorithm or primal dual hybrid gradient method (PDHG).

We further define the element-wise shifted shrinkage-thresholding (in the context of the TVL1 optical flow problem),

$$\mathrm{SST}(\mathbf{v},\tau)[\mathbf{n}] = \mathbf{v}[\mathbf{n}] + \frac{1}{\alpha}\nabla\bar{\mathbf{u}}_1[\mathbf{n}]\left(\mathrm{ST}(\mathbf{r},\tau\alpha) - \mathbf{r}\right)[\mathbf{n}], \tag{6}$$

where $\mathbf{r} = \nabla\bar{\mathbf{u}}_1^\top(\mathbf{v} - \bar{\mathbf{v}}) + \bar{\mathbf{u}}_1 - \mathbf{u}_0$, $\alpha = \|\nabla\bar{\mathbf{u}}_1[\mathbf{n}]\|_2^2$.

Note that the TVL1-BCA solver (Algorithm 1) is composed of convolutions with $\mathbf{D}$, $\mathbf{D}^\top$, pointwise non-linearities, and "skip-connections" – all of which are defining features of modern convolutional neural networks (CNNs).

As presented, the TVL1-BCA solver requires an initial flow estimate $\bar{\mathbf{v}}$. This is often obtained by computing coarse flow fields on J-scale Gaussian pyramid,

$$\begin{gathered}\{(\mathbf{u}_0^j, \mathbf{u}_1^j)\}_{j=0}^J,\ \mathbf{u}_i^0 = \mathbf{u}_i, \quad i = 1,2,\\ \mathbf{u}_i^{(j+1)} = (2\downarrow)\mathbf{h} * \mathbf{u}_i^j,\ i = 1,2,\ j = 0,1,\ldots,J-1,\end{gathered} \tag{7}$$

(with Gaussian blur filter $\mathbf{h}$), where the obtained flow field at scale j, $\mathbf{v}^j$, is upscaled to serve as a initial estimated for the next finer scale, $\bar{\mathbf{v}}^{j-1} = 2\mathbf{g} * (2\uparrow)\mathbf{v}^j$ (with $\mathbf{g}$ bilinear interpolation filter). Furthermore, the estimate $\bar{\mathbf{v}}$ may be updated multiple times, W, per scale. Together we say that the TVL1 solver is embedded in a pyramid iterative coarse-to-fine (CTF) framework. In practice, accelerated convergence of the TVL1 solver is observed if the dual-variable $\mathbf{w}$ is also upscaled to the finer level and used to warm-start the solver variable.

2.2 Deep Learning Methods

Many successful deep networks for optical flow borrow from the classical literature by embedding their network in a pyramid structure (Sun et al. 2018; Ilg et al. 2017), where intermediate computations at a coarser scale are upscaled to a finer scale for further computation. Furthermore, such networks often perform pyramid upsampling and warping on "feature-domain" signals. The proposed BCANet differs from these methods by only working with original grayscale versions of the input image pair. However, feature domain processing and warping is carried out via the dual-variable $\mathbf{w}$.

Other methods (Teed and Deng 2020) iteratively update the optical flow variable at a single scale. Many state-of-the-art methods (Teed and Deng 2020; Sun et al. 2018; Hofinger et al. 2020) use a "cost-volume" of cross-correlations between feature-domain versions of the input image for intermediate computations. While such methods have proven to be high performing, they are often detrimental to the computationally complexity of the method and are less amenable to real-time applications. In contrast, our proposed method has roughly the computational complexity of existing classical methods, which already run in real-time (Monzón, Salgado, and Sánchez 2016).

These deep learning methods contain millions of parameters and use long training schedules to achieve their performance. It is of significant interest in the community to reduce the computational demand of their models in training and inference, even if at the cost of some reduced performance (Sun et al. 2018; Teed and Deng 2020; Ilg et al. 2017; Hofinger et al. 2020).

3 Proposed Method

3.1 BCANet Architecture

In this section, we propose an optical flow deep neural network derived from an unrolled TVL1-BCA solver (BCANet) embedded in a pyramid iterative CTF scheme (PiBCANet). To do so, we first generalize the TVL1-BCA problem by changing the TV regularization for a weighted analysis dictionary regularization, $\psi(\mathbf{v}) = \|\boldsymbol{\lambda} \circ \mathbf{D}\mathbf{v}\|_1$ where $\mathbf{D}$ represents convolution with an M subband, 2-channel filter-bank, $(\mathbf{D}\mathbf{v})_j = \mathbf{d}_j * \mathbf{v}$. We then unroll the resulting solver as

$$\begin{gathered}\mathrm{BCANet}_\Theta(\mathbf{u}_0, \bar{\mathbf{u}}_1, \bar{\mathbf{v}}, \mathbf{w}^{(0)}) = \mathbf{v}^{(K)}, \quad \mathbf{v}^{(0)} = \bar{\mathbf{v}}\\ \mathbf{w}^{(k+1)} = \mathrm{softClip}\left(\mathbf{w}^{(k)} + \mathbf{A}^{(k)}\mathbf{v}^{(k+1)}, \boldsymbol{\lambda}^{(k)}\right)\\ \mathbf{v}^{(k+1)} = \mathrm{softSST}\left(\mathbf{v}^{(k)} - {\mathbf{B}^{(k)}}^\top\mathbf{w}^{(k+1)}, \tau^{(k)}\right)\\ k = 0,1,\ldots,K-1,\\ \Theta = \{\mathbf{A}^{(k)}, {\mathbf{B}^{(k)}}^\top, \boldsymbol{\lambda}^{(k)} \in \mathbb{R}_+^M, \tau^{(k)} \in \mathbb{R}_+\}_0^{K-1}.\end{gathered} \tag{8}$$

$\mathrm{softClip}(\mathbf{z},\lambda) = \lambda\tanh(\mathbf{z}/\lambda)$ is a differentiable clipping function based on the hyperbolic tangent, and softSST is a differentiable shifted shrinkage-thresholding function defined via (6) with ST replaced by

$$\mathrm{softST}(\mathbf{z},\tau) \coloneqq x - \tau\tanh(\mathbf{z}/\tau),$$

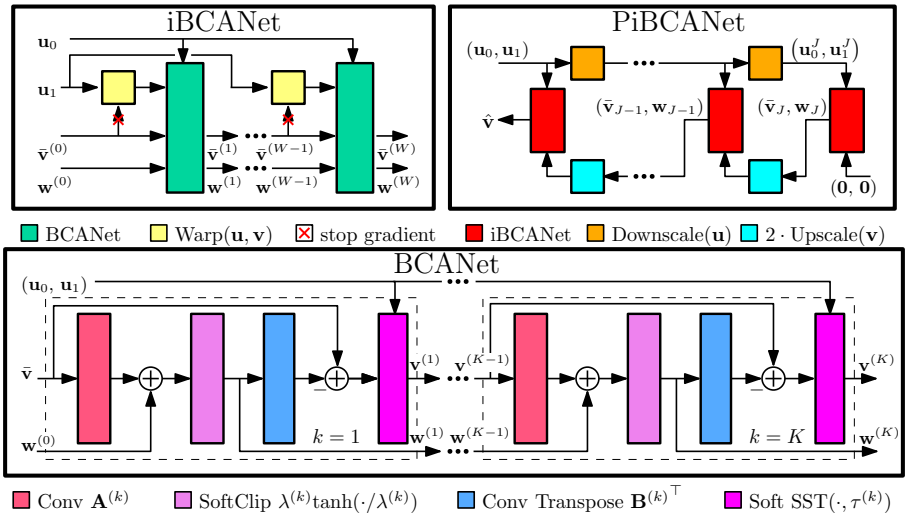

Figure 1: Bottom: BCANet architecture derived via differentiable unrolled TVL1 BCA algorithm. Top-Left: iterative BCANet (iBCANet) derived via iterative optical flow estimation with BCANet at a single image scale/resolution. Top-Right: Pyramid iterative BCANet (PiBCANet) via embedding iBCANet in a pyramid scheme.

as suggested via Moreau's identity. These "soft" varients of the original TVL1 solver's non-linearities are essential to successfully training the network. Furthermore, $\mathbf{A}^{(k)}$ and $\mathbf{B}^{(k)^\top}$ are analysis and synthesis convolution operators operating from $2 \rightarrow M$ and $M \rightarrow 2$ channels respectively. Vector $\boldsymbol{\lambda}^{(k)}$ represents a per-iteration learnable weighted L1-norm variable, and $\tau^{(k)}$ a learnable step-size parameter. A corresponding σ step-size parameter is implicitly learned via the untying of the convoultion operators. This BCANet architecture is shown in Figure 1. The total set of learnable parameters within BCANet is given by Θ (8).

We further ensure differentiability of BCANet via non-zero threshold parameters. We achieve this by parameterizing the thresholds in a log-domain

$$\boldsymbol{\lambda}^{(k)} := \exp\left(\tilde{\boldsymbol{\lambda}}^{(k)}\right), \; \tau^{(k)} := \exp\left(\tilde{\tau}^{(k)}\right).$$

For simplicity of unrolling we also set the extrapolation parameter θ of Algorithm 1 to zero.

3.2 PiBCANet Architecture

To obtain an initial flow estimate to center our BCA taylor expansion, as well as aid in computing large displacements, we embed (several of) our proposed BCANet in a pyramid iterative coarse-to-fine warping scheme which we call PiBCANet. To achieve this, we employ a differentiable warping function via bilinear interpolation. For image $\mathbf{u}$ and flow field $\mathbf{v}$, we compute the warping as

$$\begin{aligned}\mathcal{W}(\mathbf{u},\mathbf{v})[\mathbf{n}] = &(\lceil x\rceil - x)(\lceil y\rceil - y)\mathbf{u}\left[\lfloor x\rfloor, \lfloor y\rfloor\right] \\ &+ (\lceil x\rceil - x)(y - \lfloor y\rfloor)\mathbf{u}\left[\lfloor x\rfloor, \lceil y\rceil\right] \\ &+ (x - \lfloor x\rfloor)(\lceil y\rceil - y)\mathbf{u}\left[\lceil x\rceil, \lfloor y\rfloor\right] \\ &+ (x - \lfloor x\rfloor)(y - \lfloor y\rfloor)\mathbf{u}\left[\lceil x\rceil, \lceil y\rceil\right],\end{aligned} \tag{9}$$

where $x = \mathbf{n}_x + \mathbf{v}_x$ and $y = \mathbf{n}_y + \mathbf{v}_y$. Following the insights of Hofinger et al., we chose to stop back-propagation the warping function. This is justified as the rounding within the function will cause excessive noise. Indeed, it is observed anecdotally that gradient stopping stabilizes training for PiBCANet.

As shown in Figure 1, the iterative warping and flow estimates obtained at a single resolution/scale via (iBCANet) are warm-started by upscaled variables from the previous (coarser) resolution, and the output of the current scale are used similarly for the next (finer) scale. To achieve this, we generate a Gaussian pyramid of the input images with a standard-deviation of $\sigma = 0.8$ pixels, and a filter

side-length of $\lceil 6\sigma - 1 \rceil$. We use the PiBCANet architecture with untied BCANet networks for $J+1$ scales and W warps per scale.

3.3 Loss Function

We use a supervised training loss with supervision at each scale and weight-decay regularization. Specifically, we generate a Gaussian pyramid of the ground-truth flow field, $\mathbf{v}_{\mathrm{gt}}$, and use the average end-point-error (EPE) at each scale and warping,

$$\mathcal{L}\left(\hat{\mathbf{v}}, \mathbf{v}_{\mathrm{gt}}; \{\Theta_{(j,w)}\}_{(j=0,w=1)}^{(J,W)}\right) = \sum_{j=0,w=1}^{J,W} \frac{\alpha^{-j}\beta^{(-W+w)}}{2^{-j}N} \sum_{\mathbf{n}} \|\mathbf{v}_{\mathrm{gt}}^{j}[\mathbf{n}] - \hat{\mathbf{v}}^{(j,w)}[\mathbf{n}]\|_2 + \gamma\|\Theta_{(j,w)}\|_2^2. \tag{10}$$

We refer to the above as the PiEPELoss with parameters α, β, γ.

4 Experiments

The following experiments were implemented in the Flux deep learning package (Innes 2018) of the Julia programming language (Bezanson et al. 2017). Code is available here: *https://github.com/nikopj/UnrolledFlowNetworks.jl*.

4.1 Experimental Setup

Networks: We train a single PiBCANet with $(J+1) = 6$ scales and $W = 1$ warps per scale, totaling in 6 untied BCANets. We use $K = 20$ unrollings with $M = 16$ subbands and filter sizes of $P \times P = 5 \times 5$ in each BCANet. We refer to this network as PiBCANet (5,1) to denote its scales and warpings. We compare this network to its classical counterpart TVL1-BCA (5,6) with $J = 5$ and $W = 1$ which uses static trade-off parameter λ =2e-1.

Initialization: The network is initialized with filters from a standard normal distribution, which are then normalized by the spectral norm of convolution operators. Step-sizes $\tau^{(k)}$ are initialized to 1 corresponding to the maximum step-size allowed in the TVL1-BCA algorithm. Thresholds $\boldsymbol{\lambda}^{(k)}$ are initialized to 1e-1.

Datasets: We train PiBCANet on the Flying Chairs dataset (Dosovitskiy et al. 2015). This synthetic dataset consists of random background images from Flickr with random translations, and randomly generated foreground furniture models also with affine translation. Flying chairs contains ground truth flow fields for each of its 22k image pairs. We evaluate PiBCANet on the MPI-Sintel "clean-pass" training dataset (Butler et al. 2012). This dataset is generated from a CGI movie and contains dynamic motion beyond simple affine maps. MPI-Sintel provides ground truth flow fields and occlusion masks for its 1k training images. These datasets of Flying Chairs and MPI-Sintel are often used together in the literature as they share a similar histogram of flow magnitudes (Ilg et al. 2017).

Training: We train PiBCANet with the ADAM (Kingma and Ba 2014) optimizer (with default parameters) over the Flying Chairs training dataset on the PiEPELoss (10). An initial learning rate of 1e-3 is used and it is halved twice at 5k iterations and 10k iterations, for a total of 15k iterations. A weight decay parameter of γ =1e-4 is used to avoid over-fitting. A PiEPELoss parameter of $\alpha = 1$ is used as the expected loss between scales already contains a factor of 0.5. Random crops of size 256×256, along with random flips and small additive noise are performed online during training for additional robustness. Image pairs are normalized to the range $[0, 1]$ (divided by 255) and also have their joint mean subtracted before being sent to the network. To further stabilize training we clip-gradients to the range $[-1, 1]$.

4.2 Dataset Comparisons

Table 1 compares the performance of the proposed PiBCANet to leading classical and deep learning methods, including the described TVL1-BCA classical solver.

First, we observe the that the trained PiBCANet has an order magnitude less parameters than the leading deep learning models. This is purely due to time constraints, and not a reflection of the limitations of the proposed method. With this in mind, it is not a surprise to see PiBCANet, as shown, is not state-of-the-art method. Second, we see that PiBCANet performs favorably to the TVL1-BCA solver over the Flying chairs validation set and the MPI-Sintel training set. This is a promising result, but more careful tuning of the hyperparameter λ in the classical solver is required for a

Table 1: Performance of classical (IPOL Brox, DF-Auto, TVL1-BCA) and deep learning methods (PWCNet, RAFT) against proposed PiBCANet. Average end-point-error shown on Flying Chairs validation dataset (F) and MPI-Sintel training dataset clean pass (S). For MPI-Sintel dataset, error on all pixels and those with matching correspondences is shown in (all) and (match), respectively.

Method	Params	F	S (all)	S (match)
IPOL Brox (2016)	-	-	5.06	-
DF-Auto (2016)	-	-	5.93	-
TVL1-BCA (5,6)	-	5.84	8.28	6.33
PWCNet (2018)	8.75M	-	2.55	-
RAFT (2020)	5.3M	-	1.43	-
PiBCANet (5,1)	195k	5.33	7.67	5.05

fair comparison. We lastly observe that PiBCANet does not outperform other classical methods. However, these methods (IPOL Brox, DF-Auto) tuned their hyper-parameters on the MPI-Sintel train set. Thus, the comparison is not fair but it does demonstrate that PiBCANet is not yet able to out perform the best-case scenario of classical methods.

4.3 Visual Comparisons

PiBCANet is shown to outperform TVL1-BCA (Algorithm 1) on a held out testset in the average EPE metric. In this section, we show that this metric does not tell the whole story, and that there is much room for improvement in the proposed network. Figure 2 shows a comparison of PiBCANet and TVL1-BCA on a mild example from the MPI-Sintel training subset. Though PiBCANet has definitely learned to produce optical flow estimates, we see that TVL1-BCA performs better than PiBCANet both qualitatively and quantitatively. The classical solver is able to maintain hard edge boundaries, whereas PiBCANet appears too diffuse. From these images alone, it is unclear whether this is simply a property of the network or an implementation/training issue. However, intuition suggests that PiBCANet *should* be able to perform just as well as TVL1-BCA, if not better.

To investigate this further, Figure 3 shows a more difficult example from the Flying Chairs validation dataset, along with the intermediate flows from the TVL1-BCA solver and PiBCANet. First, we note that PiBCANet achieves a lower average EPE on this example, although this fact is not immediately obvious visually. This suggests that PiBCANet's favorable performance in Table 1 may be due to a suppression of outliers.

Second, we observe that the flow fields obtained at coarser scales of PiBCANet (middle row Fig. 3) do not accurately match the ground truth flow. This suggests some error/inadequacies in the training of the model, as intuition suggests that the optical flow predictions should only get better at coarser scales as there is less detail to account for. This problem of poor intermediate flow estimation could be tackled using the α parameter of the pyramid loss (10) or perhaps an training schedule in which the coarser scale BCANet's are trained first and then added into a PiBCANet architecture. Fully end to end training would in theory be favorable as it would allow the network to learn features in the coarser scales that are tailored for aiding the finer scale's computation.

5 Discussion

The proposed PiBCANet has been shown to have some potential in that it can compete with the most basic classical solver at a fraction of its computational requirements. Visual inspection has suggested that further refinement of the model hyperparameters (size, training, loss) may be fruitful and achieving a network that operates at the true capacity of the proposed method.

6 Conclusion

In this report, we presented a novel neural network architecture, PiBCANet, derived from a classical optical flow algorithm. The network consists of an unrolled TVL1 solver embedded in an pyramid iterative coarse-to-fine warping strategy. Critically, the proposed network augments the non-linearities of the original algorithm to be amenable to passing gradients and allowing training, in contrast to the direct parameterization used in (Janjušević, Khalilian-Gourtani, and Wang 2021). The network was shown to out-perform the classical TVL1-BCA algorithm on standard benchmarks, however, visual inspection showed that network is not operating as intended – intermediate/coarser flows are not computed well. This may be a loss function or training problem, and future work should look into ensuring proper flow estimation within the network.

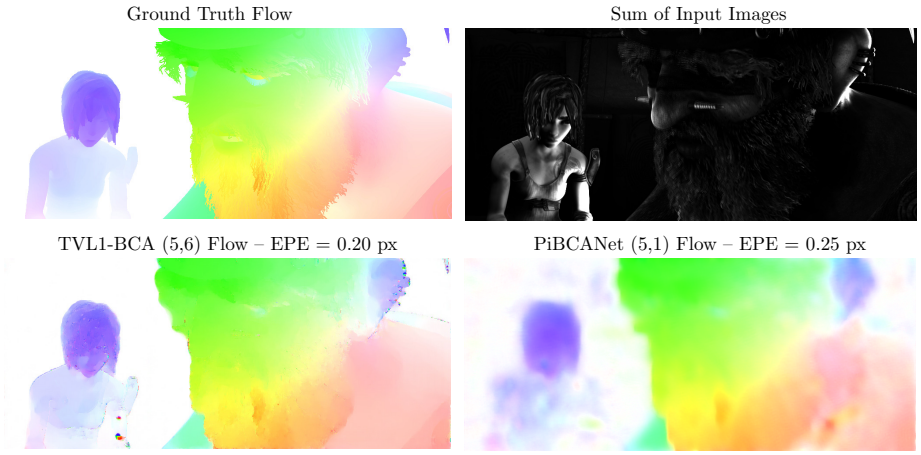

Figure 2: Flow fields obtained for frame 48-49 of "shaman_2" sequence from MPI Sintel training set (Butler et al. 2012). The average motion is small in comparison to Figure 3.

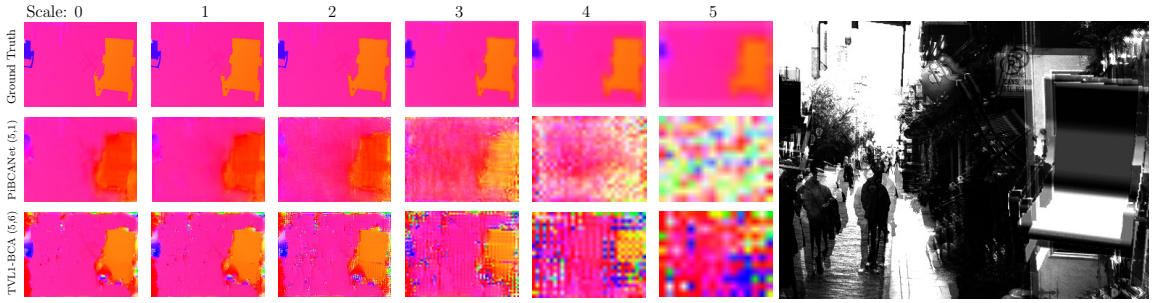

Figure 3: Intermediate flow fields (dispalyed in HSV colorspace) on image pair with large displacement from the Flying Chairs validation set (Dosovitskiy et al. 2015). Left: Flow fields obtained by PiBCANet with 6 scales and 1 warp per scale, TVL1-BCA solver with 6 scales and 6 warps per scale, along with ground truth flow. EPE at scale 0: PiBCANet= 3.52 px, TVL1-BCA= 3.95 px. Right: Sum of grayscale input frames.

If the proposed PiBCANet is shown to overcome the previously mentioned challenges, interesting future work lies ahead in extending the method to an analogous unrolling of classical methods that are able to handle vector valued images.

References

Bezanson, J.; Edelman, A.; Karpinski, S.; and Shah, V. B. 2017. Julia: A fresh approach to numerical computing. *SIAM Review* 59(1):65–98.

Burger, M.; Dirks, H.; and Frerking, L. 2017. 7. on optical flow models for variational motion estimation. In *Variational Methods*. De Gruyter. 225–251.

Butler, D. J.; Wulff, J.; Stanley, G. B.; and Black, M. J. 2012. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.)., ed., *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, 611–625. Springer-Verlag.

Dosovitskiy, A.; Fischer, P.; Ilg, E.; Häusser, P.; Hazırbaş, C.; Golkov, V.; v.d. Smagt, P.; Cremers, D.; and Brox, T. 2015. Flownet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*.

Fan, L.; Huang, W.; Gan, C.; Ermon, S.; Gong, B.; and Huang, J. 2018. End-to-end learning of motion representation for video understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6016–6025.

Hofinger, M.; Bulò, S. R.; Porzi, L.; Knapitsch, A.; Pock, T.; and Kontschieder, P. 2020. Improving optical flow on a pyramid level. In *European Conference on Computer Vision*, 770–786. Springer.

Horn, B. K., and Schunck, B. G. 1981. Determining optical flow. *Artificial intelligence* 17(1-3):185–203.

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

Innes, M. 2018. Flux: Elegant machine learning with julia. *Journal of Open Source Software*.

Janjušević, N.; Khalilian-Gourtani, A.; and Wang, Y. 2021. Cdlnet: Noise-adaptive convolutional dictionary learning network for blind denoising and demosaicing.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Monzón, N.; Salgado, A.; and Sánchez, J. 2016. Regularization strategies for discontinuity-preserving optical flow methods. *IEEE Transactions on Image Processing* 25(4):1580–1591.

Rakêt, L. L.; Roholm, L.; Nielsen, M.; and Lauze, F. 2011. Tv-l1 optical flow for vector valued images. In *EMMCVPR*.

Sun, D.; Yang, X.; Liu, M.-Y.; and Kautz, J. 2018. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Teed, Z., and Deng, J. 2020. Raft: Recurrent all-pairs field transforms for optical flow. In Vedaldi, A.; Bischof, H.; Brox, T.; and Frahm, J.-M., eds., *Computer Vision – ECCV 2020*, 402–419. Cham: Springer International Publishing.