LikeNet: A Siamese Motion Estimation Network Trained in an Unsupervised Way

Aria Ahmadi
a.ahmadi@qmul.ac.uk
Ioannis Marras
i.marras@qmul.ac.uk
Ioannis Patras
i.patras@qmul.ac.uk

Multimedia and Vision Research Group
Queen Mary University of London
London, UK

Abstract

This paper addresses the problem of motion estimation using a Convolutional Neural Network (CNN). We propose a network, called LikeNet, that at test time, receives as input a pair of images and calculates at the output a pixel-wise distribution over motion classes. We adopt a Siamese architecture, each branch of which receives the reference frame and the target frame shifted by a specific motion vector. Each of the branches calculates at the output a similarity map that at each pixel it could be interpreted as the probability that the image features in the shifted images match, or equivalently, the probability that the motion vector at a pixel in question is equal to the shift by which the target frame was displaced. The individual similarity maps are merged and further processed by RNNs that implement Conditional Random Fields and impose therefore anisotropic smoothing constraints. The proposed network is trained in an unsupervised manner, that is without the need for ground truth motion fields from synthetic sequences, by optimising at training time the deviation from a feature constancy constraint – this allows for training with unlimited training data. Our method is tested on both synthetic and real image sequences and performs similarly with other state-of-the-art unsupervised methods using a fraction of the number of learned parameters.

1 Introduction

The goal of dense motion estimation is to estimate a motion field that describes the displacement of pixels from a reference frame to a target frame. It is a challenging problem in computer vision with many applications, such as video coding [6, 15] and tracking [17]. Classical motion estimation methods minimize, at test time, a cost function [3, 4] that penalizes deviations from constraints. The most important ones are the intensity/brightness/ colour constancy constraint that states that the intensity/ colour of a pixel does not change by displacement, and the smoothness constraint that states that the motion field of neighbouring pixels should be similar. Several variations and improvements of these constraints have been proposed in the literature, aiming at more realistic modelling - for example by taking into consideration motion discontinuities or intensity changes due to shadows [3, 4, 20]. Such constraints are typically handcrafted and, with the exception of a few methods are not learned from data. In addition, solving an optimization problems at test time typically results in computationally expensive iterative estimation schemes.

Inspired by the success of Deep Learning in several low-level Computer Vision problems, some DNN-based methods have been proposed for motion estimation [5, 9, 18]. Although they achieve promising results, most of them rely on supervised schemes and therefore require groundtruth motion fields from a synthetic datasets. Despite the progress in computer graphics, such synthetic datasets do not accurately characterize realistic scenes. In order to generalize well to an unseen dataset, supervised methods typically need finetuning, also requiring ground truth data on samples from that dataset. Obtaining ground truth motion fields from real scenes is not practical and therefore, developing an accurate method for unsupervised training is desired. Motivated by this, in the recent years, some "unsupervised" methods [1, 12, 16, 19, 31] have been introduced. While progress has been made, the proposed methods still have drawbacks. The unsupervised method proposed recently in [19] involves an upscaling operation at each layer of the latter half of their architecture. This upscaling operation helps with the estimation of motions with larger magnitude, however, since the upscaling is embedded in the architecture, any modification in the number of upscaling layers requires finetuning. The method proposed in [1] demonstrated good performance on a test set using unsupervised training, however training the proposed network in practice is hard. All of those methods treat motion estimation as a regression problem, that is, produce a 2D output with two channels corresponding to the horizontal and vertical motion component.

In this paper, we treat motion estimation as a dense labeling problem. We propose an unsupervised trained Deep Network by adopting a Siamese architecture, with as many branches as motion labels. Each branch of the architecture, receives as input the reference frame and the target frame translated by the motion label in question, and produces the (not normalised) probability map for the motion label in question - that is the (unnormalized) probability that a pixel is translated by the motion vector corresponding to the motion label in question. To the best of our knowledge, this is the first time the problem of motion estimation is treated as a classification problem in a DNN-based framework.

In order to deal with motions with large magnitude, our network is embedded in a classical multiscale scheme. A major issue in multiscale methods is that errors at lower resolutions are propagated to higher ones - for this reason, significant gains can be made by improving the quality of the estimation at the lower levels. To this end, we use Conditional Random Fields (CRFs) at the lowest resolution, implemented as an RNN similar to in [30], so that one can form an end-to-end trainable framework for motion estimation which combines the strengths of deep learning and graphical modelling. More specifically, we employ the CRF to improve the estimated motion field at the lowest resolution of our multi-scale scheme. To the best of our knowledge, this is the first time that CRFs are integrated into a DNN-based framework for motion estimation.

Our network is trained on a simple cost function, without explicit smoothness or other constraints - those are implicitly modelled in the filters of the CNN and are learned from training data. Random, consecutive pairs of frames drawn from videos of the UCF-101 dataset [22] were used for training, while for evaluation both synthetic and realistic datasets were used. We show that the unsupervised trained LikeNet, although is not finetuned on any of the evaluation datasets, performs better, or in par with other unsupervised trained DNN-based methods on both synthetic and real data, even in the case that the other methods are finetuned (in an unsupervised manner) on the target dataset. In addition, when compared to other DNN-based methods, LikeNet model is the smallest in terms of learned parameters - respectively 98% and 42% smaller than FlowNet [7, 19] and SpyNet [18] architectures.

The remainder of the paper is organised as follows. In Section 2 we discuss some works related to ours. In Section 3.1 we present the proposed method, and in Section 4 experimental results and comparisons with other methods in the literature. Finally in Section 5 we draw some conclusions.

2 Related Work

Intensive research has been conducted in the field of motion estimation since Horn and Schunk [8] proposed their classical optical flow variational formulation. Horn and Schunk [8] propose an energy function which minimization penalizes the deviation from the intensity constancy assumption and relaxes a spatial smoothness constraint. For a detailed review of early methods, we refer to [23, 26] and divide more recent methods into two groups: DNN-based methods and other methods which we refer to as classic methods.

To tackle the problem of motion estimation, one of the main assumptions is that the intensity value of the pixels should not change by a displacement. This is referred to as the brightness constancy constraint. Brox et al. [4] consider two more constraints, gradient constancy, and a spatiotemporal smoothness constraint, to derive a variational formulation. Gradient constancy assumption allows for slight intensity changes under a displacement. The smoothness assumption takes into account the neighbouring pixels to take care of the situation where there is no gradient or in the case of aperture problem. Minimizing the energy function in [4] allows for estimation of accurate dense motion field for small motions. Their multi-scale coarse-to-fine approach allows for estimation of larger motions. Several effective methods have been published following the same principle [8, 28].

State-of-the-art approaches [3, 27] use descriptors matched between two frames integrated into a variational approach. Brox et al. in [3] combines the advantage of both energy minimization methods which yield dense motion field for small motions and descriptor matching which allows finding large displacements. They use a segmentation method to find regions in the frame and produce region descriptors that are later used to find a sparse set of hypothesis for correspondences. These hypotheses, initial matches, are then integrated into a variational approach. Their energy function is similar to [4] but, with one additional term which integrates the correspondence information. Most of the descriptor matching approaches rely on rigid descriptors which mean rigid motion hypothesis. Weinzaepfel et al. in [27] first propose a new non-rigid matching algorithm which can retrieve smooth dense correspondence and then they suggest a method for combining the matchings with a variational approach for motion estimation. The method proposed in [20] has three steps: First, to find the matched features between the two frames which forms a sparse set. Second, to perform a densification of this sparse set of matches by computing a sparse-to-dense edge-aware interpolation. Third, they perform one step of variational energy minimization using the dense interpolation as initialization. Although this method does not suffer from previous shortcomings, like the other, it is still relaxing handcrafted constraints.

Inspired by the great success of Deep Neural Networks in several Computer Vision problems [14], a CNN has been proposed by Dosovitskiy et al. in [5] for motion estimation. The method showed performance that was close to the state-of-the-art in a number of synthetically generated image sequences. Dosovitskiy et al. in [5] propose two convolutional hourglass architectures which are trained for motion estimation and differ in some of the lowest convolutional layers. One is fully composed of conventional convolutional layers as receives the input pair as a 2 channel data, FlowNetS. In the other one, the two input frames are fed into two convolutional streams and after 3 convolutional layers, the computed featuremaps are joined through a correlation layer, FlowNetC. The idea behind their architecture is to learn strong features at multiple scales and abstractions and to ease finding the actual correspondences based on these features. They reported outperforming some of the classical state-of-the-art methods. In Flownet 2.0 [9] the concept of end-to-end learning of optical flow was used to improve the quality of the estimation and the performance by slightly sacrificing the speed. To train both FlowNet and Flownet2.0 supervised, synthetically made datasets were used as the existing optical flow datasets are too small. These synthetic datasets consist of random background images on which are overlaid segmented images. These data do not characterize real-world data, but allows for generating arbitrary amounts of samples with custom properties. Another method has been proposed by Ranjan and Black [18] which employs DNNs in a coarse-to-fine scheme for motion estimation. They let the coarse-to-fine scheme to handle larger motions and get a considerable reduction of learned parameters compared to FlowNet while achieving a higher performance compared to FlowNet. However, those methods require ground truth motion fields for training.

The first unsupervised DNN-based methods were proposed in [1, 12]. [1] minimizes the classic intensity constancy constraint linearized by Taylor expansion for training and applies it in a multi-scale scheme during the test. However, this method is not easy to train and there is no mechanism to improve the quality of estimation in the lower resolutions to prevent the propagation of the errors across scales. [12] adopts a similar approach, however instead of linearisation of the motion compensated intensity differences, they utilise a spatial transformer layer Jaderberg et al. [10] and explicitly penalise motion discontinuities. [19] adopt the FlowNetS architecture [5] and train it using the method proposed in Jaderberg et al. [10] considering also gradient and smoothness constancy assumptions. They use an upscaling operation at each layer of the latter half of the exploited architecture. This upscaling operation helps with the estimation of motions with larger magnitude. However, the upscaling is embedded in the architecture and therefore any modification in the number of upscaling layers requires finetuning. All of the methods above treat motion estimation as a regression problem, this is produce a 2 channel output with dimensions equal to those of the input images.

3 LikeNet: a CNN for Motion Estimation

In this section, we present in detail our approach for a CNN for motion estimation, or more precisely a CNN for pixel-level motion class prediction. More specifically, LikeNet treats motion estimation as a classification problem and at test phase, it predicts for each pixel, the motion label/class of the pixel in question. More specifically, at test time, LikeNet receives as input a pair of consecutive frames and outputs a pixel-level distribution over K motion labels/classes.

Formally, let L be a Random Field defined over a set of N discrete variables $\{L_1, \cdots, L_N\}$, where N is the number of image pixels. The domain of each variable L_i is a set of motion labels $M = \{\mathbf{m}_1, \cdots, \mathbf{m}_K\}$, each label corresponding to a motion vector. Clearly, an instantiation of the label field L corresponds to a dense motion field and in this paper, the term m_k will be used to denote both the k^{th} label and the k^{th} motion vector - the interpretation should be clear from the context. Also, let $I \in R^{2\times N}$ denote an input pair of consecutive frames, each of size N. LikeNet receives I as input and outputs a pixel-level distribution over the motion classes, $P(L|I;\theta)$, where θ denotes the model parameters, that is the parameters of the network. The mode of $P(L|I;\theta)$ could be then used as a point estimate of the motion/label field. That is $L^* = \underset{L}{\operatorname{argmax}}\, P(L|I;\theta)$.

3.1 Architecture

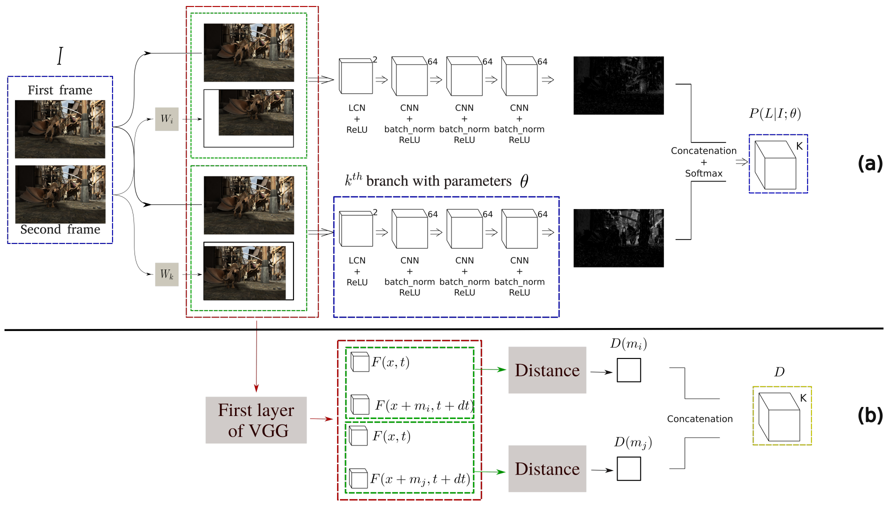

Figure 1: The proposed architecture of LikeNet. For simplicity, only two branches out of K branches (motion classes) are illustrated. The input to the k^{th} branch is the concatenation of the first frame and the second frame shifted with the corresponding motion vector $\mathbf{m}_k$. Block W_k warps its input along with motion vector $\mathbf{m}_k$. LikeNet outputs a pixel-level distribution over the motion classes, $P(L|I;\theta)$.

An overview of the proposed architecture of LikeNet at test time is given in Fig. 1(a). More specifically, to calculate $P(L|I;\theta)$, we propose a Siamese CNN with number of branches equal to K. The input to the k^{th} branch is the concatenation of the first frame and the second frame shifted by the corresponding motion vector $\mathbf{m}_k$. Each branch consists of a Local Contrast Normalization (LCN) layer [11] and three convolutional layers, and calculates a single channel, pixel-level heat-map, which indicates whether the pixels at the same location in the reference and the shifted target frames, match. Since the target frame has been shifted by the motion vector $\mathbf{m}_k$ this heat-map can be interpreted as the probability (not normalized) that a pixel has moved by $\mathbf{m}_k$. In order to obtain a normalized distribution over all motion classes/labels, the concatenation of the heat-maps, calculated by all branches, are passed through a Softmax layer. In other words, the k^{th} branch is responsible for calculating pixel-level probability map $P(L=\mathbf{m}_k|I;\theta) \in [0,1]^N$ that expresses the probability that each pixel in I is displaced by $\mathbf{m}_k$. An arbitrary k^{th} branch and $P(L|I;\theta)$ are highlighted in blue dotted squares in Fig. 1(a).

During training we would like to learn parameters such that at pixel i the probability $P(L_i=\mathbf{m}_k|I;\theta)$ produced by LikeNet is high for the true motion label $\mathbf{m}_k$ and low for the other labels. Our assumption is that, under an appropriate distance measure, the feature differences/distances under the correct motion vector will be lower than the feature distance under an arbitrary motion vector. That is, the features $\mathbf{F}(x_i,t)$ extracted at location x_i (pixel i) in the reference frames, will be more similar to their correspondences $\mathbf{F}(x_i+\mathbf{m}_k,t+dt)$ in the target frame shifted by the correct $\mathbf{m}_k$ (Fig. 1). Formally, we train the network so as to minimize the following cost function:

$$C(I;\theta) = \sum_i \sum_{\mathbf{m}_k \in M} P(L_i = \mathbf{m}_k | I, \theta) D(i, \mathbf{m}_k), \tag{1}$$

where $D(i, \mathbf{m}_k)$ is the distance between $\mathbf{F}(x_i, t+dt)$ and its corresponding aligned pixel in the shifted target frame $\mathbf{F}(x_i + \mathbf{m}_k, t+dt)$. The distance that we use in this paper is:

$$D(i, \mathbf{m}_k) = JSD(\mathbf{F}(x_i + \mathbf{m}_k, t+dt) \| \mathbf{F}(x_i, t)) \tag{2}$$

where JSD denotes the Jensen-Shannon divergence.In our experiments, we chose $\mathbf{F}$ in Eq. (2) to be the features calculated by the first convolutional layer of VGG-16 [21]. While raw intensities/colour could also be used, we have found that those features perform better. These distances are calculated by the branch of the network depicted in Fig. 1(b), which clearly, is used only during training.

Intuitively, we would like that the network outputs higher probability $P(L = \mathbf{m}_k | I; \theta)$ at pixels where, under a shift by $\mathbf{m}_k$, the distance between the corresponding features is smaller. Given that the probability of the motions for each pixel sums to one, minimizing Eq. (1) forces the network to increase the probability of the motion classes for which their corresponding values in D are small.

The proposed formulation relies on a quantisation of the motion vectors to motion labels. With a quantization of each of the horizontal/vertical components to integers, in order to be able to estimate motions of magnitude V, $K = V^2$ branches are needed at test time. For large V this is not practical. For this reason, at test time we embed the trained network in a multi-scale scheme as described in Algorithm 1. In our experiments we use 5 scales with 121, 169, 49, 9, 9, branches from the lowest resolution to the highest resolution, respectively.

Algorithm 1 The algorithm of our proposed framework during the test time.

```
1:  procedure
2:      $I_1$,$I_2$: The two input frames
3:      $\Upsilon$: The estimated motion field
4:      $\beta$: The number of image pyramid levels
5:      $I_1^n$,$I_2^n$: The downsampled versions of $I_1$ and $I_2$ by a factor of $2^n$
6:      $I_{2_w}^n$: The warped version of the second frame
7:      $\Upsilon \leftarrow 0,\ I_{2_w}^n \leftarrow I_2^n,\ n \leftarrow \beta$
8:      while $n > 0$ do
9:          $\Delta\Upsilon \leftarrow CNN(I_1^n, I_{2_w}^n)$ : Calculate the update on the motion field
10:         if $n = \beta$ then
11:             $\Delta\Upsilon \leftarrow CRF(I_1^n, \Delta\Upsilon)$ : Correcting the motion field in the lowest resolution
12:         $\Delta\Upsilon \leftarrow GaussFilt(\Delta\Upsilon)$ : Gaussian filtering of the motion field
13:         $\Upsilon \leftarrow \Upsilon + \Delta\Upsilon$ : Update $\Upsilon$ using the motion field
14:         Up-sample $\Upsilon$ by a factor of $\frac{1}{0.5}$
15:         $n \leftarrow n - 1$
16:         $I_{2_w}^n \leftarrow warp(I_2^n, \Upsilon)$ : Warp $I_2^n$ towards $I_1^n$ using the motion field
17:     end while
18:     Return $\Upsilon$
```

3.2 Graphical model - CRFs

To avoid the propagation of errors across scales, we improve the quality of the estimated motion at the lowest resolution of the multi-scale scheme by using a graphical model, CRF (Fig. 2). In this Section, we provide a brief overview of the CRFs, how we learn their parameters and how we use them at inference for pixel-wise labeling. The Random Variable L that models pixel motion labels form a Markov Random Field when conditioned upon the

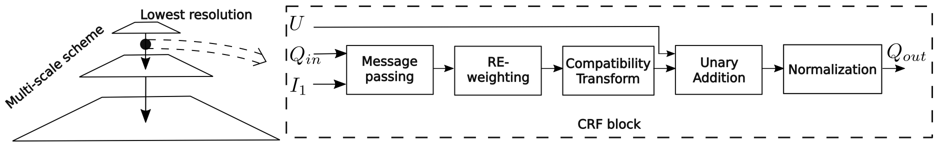

Figure 2: A CRF block at the lowest resolution during test time.

observation I. Given a graph $G = (V,E)$, where $V = \{L_1, L_2, ..., L_N\}$ and the observation I, the pair (I,L) can be modeled as a CRF characterized by a Gibbs distribution $P(L|I) = \frac{1}{Z(I)} exp(-E(L|I))$. Here $E(L)$ and $Z(I)$ are the energy and partition function, respectively. For convenience, we will drop the conditioning on I. In our CRF model, similar to [30], the energy of the label assignment L is given by:

$$E(L) = \sum_{i=1}^{N} \psi_u(L_i) + \sum_{j \in n(i) \setminus i} \psi_p(L_i, L_j) \tag{3}$$

where $\psi_u(L_i)$ is the unary energy of the pixel i taking the label L_i, and the pairwise energy component $\psi_p(L_i, L_j)$ measures the cost of assigning labels L_i, L_j to pixels i,j simultaneously. Also, $n(i)$ represents the neighborhood of pixel i. The unary is obtained from LikeNet which predicts the labels without any smoothness or consistency assumption. The pairwise energies provide an image data-dependent smoothing term that encourages assigning similar motion labels to pixels with similar properties. As in [13], we model pairwise potentials as weighted Gaussians:

$$\psi_p(L_i, L_j) = \mu(L_i, L_j) \sum_{m=1}^{M} \mathbf{w}^{(m)} \gamma_G^{(m)}(f_i, f_j), \tag{4}$$

where each $\gamma_G^{(m)}, m = 1, ..., K$, is a Gaussian kernel applied on feature vectors. The feature vector at pixel i, denoted by f_i, are spatial location and intensity values. The function $\mu(.,.)$, called the label compatibility function, captures the compatibility between different pairs of labels. Minimizing the above CRF energy $E(L)$ yields the most probable motion label assignment L. To obtain the parameters, we follow the RNN-like training scheme proposed in [30]. We set $w^{(m)}$ to 1 and use a compatibility matrix $\mu(L_i, L_j) = \frac{1}{2} e^{-\frac{(L_i - L_j)^2}{2\sigma_c^2}}$ that is parametrized by a single parameter σ_c. This is different to [30] that addresses a labeling problem where there is no natural order/structure in the labels and learns a pairwise compatibility matrix $\mu(.,.) \in \mathbb{R}^{K \times K}$ (where K is the number of labels) – in our case, the labels are structured. An overview of the CRF block, drawn from [30], is illustrated in Fig. 2.

4 Experiments

In order to evaluate our work, we compare LikeNet with a number of state-of-the-art classical and DNN-based methods, supervised and unsupervised, on a number of benchmarks. Likenet is trained for 7k iterations on samples drawn from the realistic action recognition dataset UCF101 and is not fine-tuned on any synthetic dataset. We trained for 7k iterations

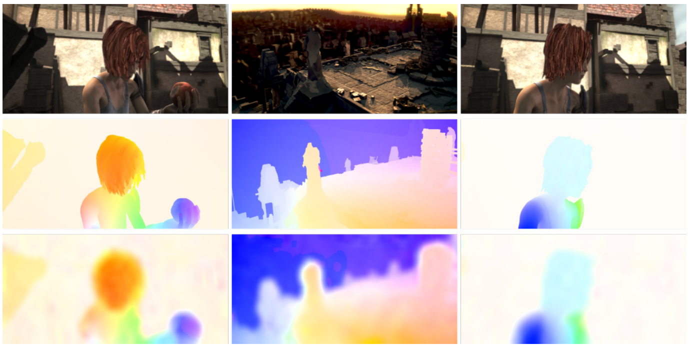

Figure 3: MPI-Sintel examples. (top-to-bottom) Input first frames, groundtruth flows, and predicted flows from LikeNet.

Method	Number of learned parameters
FlowNetS	32,070,472
FlowNetC	32,561,032
SpyNet	1,200,250
LikeNet	**697,028**

Table 1: Compared to other DNN-based methods, LikeNet is the smallest in terms of learned parameters. DSTFlow [19] follows the FlowNetS architecture. UnFlow-C [16] follows the FlowNetC architecture and UnFlow-CS is a FlowNetS architecture stacked on top of a FlowNetC. UnFlow-CSS architecture is composed of a FlowNetS stacked on top of the UnFlow-CS.

on samples drawn from the action recognition dataset named UCF101, and optimized its parameters by adopting Nesterov momentum method with momentum 0.9 [25]. The learning rate starts from 0.1 and drops aggressively by a factor of 2 every 1000 iterations. In our experiments, during training, K is set to 121 to deal with horizontal and vertical motions of maximum magnitude 5 pixels. The motion distribution in the training data can be controlled by resizing the drawn frames, we resized the frames to 96×128 for training.

Visualization of the motion field of some examples, estimated by LikeNet, from MPI-sintel are illustrated in Fig. 3. To quantitatively evaluate our method we report the Average End-point Error (AEE) on both synthetic and real datasets in Table 2. All measures include the occluded areas. We find that LikeNet is performing better than other unsupervised methods that do not use bidirectional schemes even without being fine-tuned on any other synthetic or real dataset, and close to UnFlow, that adopts a bidirectional scheme that helps significantly with occlusions. Let us note that in our Siamese architecture, each branch does the easier task of computing the similarity between corresponding pixels. This considerably reduces the model complexity of LikeNet, each branch of which consists of only 4 layers. The number of parameters that are learned is $697,028$ compared to $1,200,250$, and $32,070,472$ and $32,561,032$ parameters SpyNet [18], FlowNetS and FlowNetC respectively learn, Table 1. LikeNet is about 42% smaller than SpyNet and 98% smaller than FlowNet. While the number of the parameters are not explicitly reported, the DSTFlow method proposed in [19] uses the FlowNetS architecture, and the UnFlow methods build on the FlowNet

		Method	Fine tuned	Sintel clean		Sintel final		KITTI 2015	Middlebury
				train	test	train	test	train	train
Classic		DeepFlow	N	2.66	5.38	3.57	7.21	10.63	0.25
		LDOF (CPU) [3]	N	4.64	7.56	5.96	9.12	18.19	0.44
		LDOF (GPU) [24]	N	4.76	-	6.32	-	18.20	0.36
		EpicFlow [20]	N	2.27	4.12	3.56	6.29	9.27	0.31
		FlowFields [2]	N	1.86	3.75	3.06	5.81	8.33	0.27
		PCA-Layers [29]	N	3.22	5.73	4.52	7.89	12.74	0.66
		PCA-Flow [29]	N	4.04	6.83	5.18	8.65	14.01	.70
Deep neural network based	Sup	FlowNetS [5]	N	4.50	6.96	5.45	7.52	-	1.09
		FlowNetC [5]	N	4.31	6.85	5.87	8.51	-	1.15
		SpyNet [18]	N	4.12	6.69	5.57	8.43		0.33
		FlowNet2 [9]	N	2.02	3.96	3.14	6.02	10.06	0.35
		UnFlow-CS-ft-(KITTI supervised) [16]	Y	-	-	11.99	-	(2.25)	0.64
		UnFlow-CSS-ft(KITTI supervised) [16]	Y	-	-	13.65	-	(1.86)	0.64
	Unsup	UCNNME [1]	**N**	8.94	12.4	10.60	13.78	14.94	2.79
		DSTFlow [19]	**N**	6.93	10.40	7.82	11.11	24.30	-
		DSTFlow(KITTI) [19]	Y	7.10	10.95	7.95	11.8	16.79	-
		DSTFlow(Sintel) [19]	Y	6.16	10.41	7.38	11.28	23.69	-
		DSTFlow(C+K) [19]	Y	7.51	-	8.29	-	22.93	-
		DSTFlow(C+S) [19]	Y	6.47	10.84	6.81	11.27	25.98	-
		UnFlow-C-Cityscapes [16]	N	-	-	8.23	-	10.78	0.85
		UnFlow-C [16]	N	-	-	8.64	-	8.80	0.88
		UnFlow-CS [16]	N	-	-	7.92	-	8.14	0.65
		UnFlow-CSS [16]	N	-	-	7.91	10.22	8.10	0.65
		LikeNet	**N**	**5.7**	**10.02**	**6.49**	**10.69**	**14.66**	**0.78**

Table 2: Average End-point Error (in pixels) of classic and DNN-based methods. LikeNet performs better or in par with other unsupervised methods although it is not finetuned on any of the evaluation datasets and its capacity is considerably smaller that all other DNN-based methods.

architectures, in some cases of considerable complexity (e.g., UnFlow-CSS architecture is a FlowNetC followed by a two FlowNetS).

5 Conclusions

In this work, we propose a new Siamese CNN which solves the motion estimation as a classification problem, named LikeNet. We show that a feature constancy constraint can be used for successfully training LikeNet in an unsupervised manner and without the need for any handcrafted regularization or other constraints. Our CNN is trained on the realistic UCF101 dataset. We show that it performs better than the other state-of-the-art unsupervised methods that do not use bi-directional constraints, and that it can generalize very well to unknown datasets without the need for finetunning. The architecture of the network allows for computational flexibility and prediction of as many motion classes as required. The proposed method is among the very few studies conducted on the unsupervised training of DNNs for motion estimation. For future work, we intend to incorporate bi-directional constraints (i.e., perform motion estimation based on three frames) and investigate on computationally efficient schemes.

References

[1] Aria Ahmadi and Ioannis Patras. Unsupervised convolutional neural networks for motion estimation. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 1629–1633. IEEE, 2016.

[2] Christian Bailer, Bertram Taetz, and Didier Stricker. Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4015–4023, 2015.

[3] Thomas Brox and Jitendra Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(3):500–513, 2011.

[4] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *Computer Vision-ECCV 2004*, pages 25–36. Springer, 2004.

[5] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Höusser, Caner Hazırbaş, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, Dec 2015. URL http://lmb.informatik.uni-freiburg.de//Publications/2015/DFIB15.

[6] Frederic Dufaux and Janusz Konrad. Efficient, robust, and fast global motion estimation for video coding. *Image Processing, IEEE Transactions on*, 9(3):497–501, 2000.

[7] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852*, 2015.

[8] Berthold K Horn and Brian G Schunck. Determining optical flow. In *1981 Technical symposium east*, pages 319–331. International Society for Optics and Photonics, 1981.

[9] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *arXiv preprint arXiv:1612.01925*, 2016.

[10] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.

[11] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE, 2009.

[12] J Yu Jason, Adam W Harley, and Konstantinos G Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *European Conference on Computer Vision*, pages 3–10. Springer, 2016.

[13] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 109–117. Curran Associates, Inc., 2011.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[15] Weiping Li. Mpeg-4 video verification model version 18.0. *ISO/IEC JTC1/SC29/WG11, N3908*, 2001.

[16] Simon Meister, Junhwa Hur, and Stefan Roth. Unflow: Unsupervised learning of optical flow with a bidirectional census loss. In *AAAI*, 2018.

[17] Montse Pardàs and Elisa Sayrol. Motion estimation based tracking of active contours. *Pattern recognition letters*, 22(13):1447–1456, 2001.

[18] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. *arXiv preprint arXiv:1611.00850*, 2016.

[19] Zhe Ren, Junchi Yan, Bingbing Ni, Bin Liu, Xiaokang Yang, and Hongyuan Zha. Unsupervised deep learning for optical flow estimation. In *AAAI*, pages 1495–1501, 2017.

[20] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. *arXiv preprint arXiv:1501.02565*, 2015.

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[22] Khurram Soomro, Amir Roshan Zamir, and Mobarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. In *CRCV-TR-12-01*, 2012.

[23] Deqing Sun, Stefan Roth, and Michael J Black. Secrets of optical flow estimation and their principles. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2432–2439. IEEE, 2010.

[24] Narayanan Sundaram, Thomas Brox, and Kurt Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *European conference on computer vision*, pages 438–451. Springer, 2010.

[25] Ilya Sutskever, James Martens, George E Dahl, and Geoffrey E Hinton. On the importance of initialization and momentum in deep learning. *ICML (3)*, 28:1139–1147, 2013.

[26] Christoph Vogel, Stefan Roth, and Konrad Schindler. An evaluation of data costs for optical flow. In *German Conference on Pattern Recognition*, pages 343–353. Springer, 2013.

[27] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1385–1392. IEEE, 2013.

[28] Manuel Werlberger, Werner Trobin, Thomas Pock, Andreas Wedel, Daniel Cremers, and Horst Bischof. Anisotropic huber-l1 optical flow. In *BMVC*, volume 1, page 3, 2009.

[29] Jonas Wulff and Michael J Black. Efficient sparse-to-dense optical flow estimation using a learned basis and layers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 120–130, 2015.

[30] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537, 2015.

[31] Yi Zhu and Shawn Newsam. Densenet for dense flow. In *Image Processing (ICIP), 2017 IEEE International Conference on*, pages 790–794. IEEE, 2017.