Cross Pixel Optical-Flow Similarity for Self-Supervised Learning

(2018)

Aravindh Mahendran, James Thewlis, Andrea Vedaldi

Notes

Contributions

The authors propose a new self-supervised algorithm by using the optical flow that can be generated from videos in an unsupervised manner, and use stand alone from to train a model in a self-supervised setting. To avoid the difficulty of detecting specific details about the motion from a single frame, we learn to embed pixels into vectors that cluster together when the model believes that the corresponding pixels are likely to move together. This is obtained by encouraging the inner product of the learned pixel embeddings to correlate with the similarity between their corresponding optical-flow vectors. But even objects that can move together may not do so all the time, and this was addresses by using a contrastive loss.

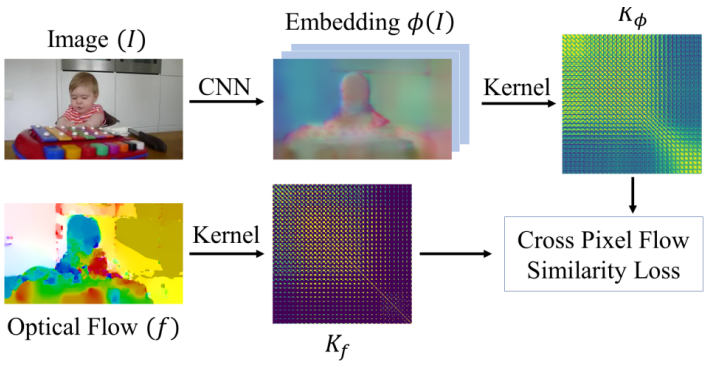

Fig. 1: We propose a novel method to exploit motion information represented as optical-flow, to supervise the learning of deep CNNs. We learn a network that predicts per-pixel embeddings $\phi(I)$ such that the kernel computed over these embeddings (K_ϕ) is similar to that over corresponding optical-flow vectors (K_f). This allows the network to learn from motion cues while avoiding the inherent ambiguity of motion prediction from a single frame.

Method

The objective is to train a neural network that maps each pixel of the input image into a D dimensional embeddings, and training objective is that the similarity between two embedding vectors must be close the corresponding flow vectors, this objective is based on the idea that things that move together should be grouped together. Given two D-dimensional embeddings vectors ϕ_p and ϕ_q of two pixels p and q and their corresponding optical flow f_q, f_p, their two kernel matrices should match over all pixels (given that the embeddings are extracted from a given image I using a CNN with parameters θ: $\phi(I, p|\Theta) = \phi(p)$):

$$K_\phi(\phi_p, \phi_q) \approx K_f\left(f_p, f_q\right)$$

where $K_\phi : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}, K_f : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ are kernels that measure the similarity of the CNN embeddings and flow vectors, respectively.

Kernels The authors propose the usage of two kernels, Gaussian for optical flows (σ is learned) and RBF for pixel embeddings:

$$K_\phi\left(\phi_p, \phi_q\right) := \frac{1}{4}\frac{\phi_p^T \phi_q}{\left\|\phi_p\right\|_2 \left\|\phi_q\right\|_2}, \quad K_f\left(f_p, f_q\right) := \exp\left(-\frac{\left\|f_p - f_q\right\|_2^2}{2\sigma^2}\right)$$

Each one of these kernels is a matrix of size $\Omega \times \Omega$, where Ω is the set of pixels, each row captures the similarity of a given pixel with respect to all other pixels in the image.

Similarity Loss Function To train the model, we need to define a similarity loss function to compare the kernels of pixel embeddings and flow vectors:

- Kernel Target Alignment (CPFS-KTA): for two kernel matrices K, K', their similarity is given by
$$\mathcal{L}_{KTA}\left(K, K'\right) = \sum_{pq} K_{pq} K'_{pq} / \sqrt{\sum_{pq} K_{pq}^2 \sum_{pq} K'_{pq} 2}$$
- Cross-Entropy (CPFS-CE): To use cross entropy loss, we first need to start by computing the target and source distribution. Each column of the two kernel matrices are normalized (using softmax), so that each one represents the probability that a given pixel belongs to the same segment as the other ones. With two distributions $S_\phi(\cdot, q).S_f(\cdot, q)$, one for flow vectors, considered as targets and one for pixel embeddings, This way embedding is tasked with inducing a kernel such that its corresponding distribution matches that of the flow vectors. the CR loss can then be computed:
$$\mathcal{L}_{CE}(\Theta) = -\sum_q \sum_p S_f(p, q) \log S_\phi(p, q)$$
- Cross-Entropy Reversed (CPFS-CE-rev): in this case, the two distributions are reversed, this time the a pixel p, the distribution $S_\phi(\cdot, q)$ must be a delta distribution around $q' = \operatorname{argmax} S_f(\cdot, q)$, this natural effect of a flipped cross entropy loss, and can be best approximated by having the embeddings of the pixel and flow vector to be similar $\phi_p \cong \phi_{q'}$ but also forces all the other to be anti correlated $\phi_p \approx -\phi_q \forall q \neq q'$.

Embedding network Based on AlexNet, the embeddings are generated using hypercolumns, the hypercolumn contains interpolated values from different levels of the network, and is then projected to a lower D-dimensional (D=16) space using an MLP. The model is trained with a sparsity trick, where only a same sampled pixels are used to train the model at each iteration, which reduces the computation and training time, and also gives better convergence given that the majority of pixel in the same image are redundant. The hypercolums are built using conv1, pool1, conv3, pool5 and fc7 of AlexNet. The embeddings are obtained after passing the hypercolumns through an MLP and L2 normalizing them.

Results

Table 1: Pascal VOC Comparison for three benchmarks: VOC2007-classification (column 4) %mAP, VOC2007-Detection (Column 5) %mAP and VOC2012-Segmentation (Column 6) %mIoU. The rows are grouped into four blocks (1) The limits of no-supervision and human supervision, (2) motion/video based self-supervision, (3) Our models and the baseline, (4) others. The third column [ref] indicates which publication the reported numbers are borrowed from. Full table in supplementary material.

	Method	Supervision	[Ref]	Cls.	Detection	Seg.
	Krizhevsky *et al.* [30]	Class Labels	[60]	79.9	56.8	48.0
	Random	-	[41]	53.3	43.4	19.8
Motion cues	Agrawal *et al.* [2]	Egomotion	[10]	63.1	43.9	-
	Jayaraman *et al.* [26]	Egomotion	[26]	-	41.7	-
	Lee *et al.* [32]	Time-order	[32]	63.8	46.9	-
	Misra *et al.* [35]	Time-order	[35]	-	42.4	-
	Pathak *et al.* [42]	Video-seg	[42],Self	61.0	50.2	-
	Wang *et al.* [55]	Track + Rank	[29, 55]	63.1	47.5	-
	CPFS-CE	Optical-flow	Self	64.2	**50.8**	41.4
	CPFS-CE-rev	Optical-flow	Self	63.6	49.9	39.5
	CPFS-KTA	Optical-flow	Self	**65.3**	50.5	**41.5**
	Ours direct cls.	Optical-flow	Self	63.2	46.1	38.8
	Other Cues - State of the art	Varied	[19, 38, 36]	73.3 [19]	55.5 [38]	40.6 [36]

Table 2: ImageNet LSVRC-12 linear probing evaluation. A linear classifier is trained on the (downsampled) activations of each layer in the pretrained model. Top-1 accuracy is reported on ILSVRC-12 validation set. The column [ref] indicates which publication the reported numbers are borrowed from. We finetune Pathak *et al.*'s [42] model along with ours as they do not report these benchmark in their paper.

	Method	Supervision	[ref]	Conv1	Conv2	Conv3	Conv4	Conv5
	Krizhevsky *et al.* [30]	Class Labels	[61]	19.3	36.3	44.2	48.3	50.5
	Random	-	[61]	11.6	17.1	16.9	16.3	14.1
	Random-rescaled [29]	-	[29]	17.5	23.0	24.5	23.2	20.6
Motion	Pathak *et al.* [42]	Video-seg	Self	**15.8**	23.2	29.0	29.5	25.4
	CPFS-CE	Optical-Flow	Self	14.9	**25.0**	**29.5**	**30.1**	**29.1**
	CPFS-CE-rev	Optical-Flow	Self	15.3	24.8	27.7	27.8	26.3
	CPFS-KTA	Optical-Flow	Self	14.8	24.6	29.2	29.5	28.1
	Ours direct cls.	Optical-Flow	Self	14.0	23.0	26.4	26.7	24.8
Other cues	Doersch *et al.* [8]	Context	[61]	16.2	23.3	30.2	31.7	29.6
	Gidaris *et al.* [19]	Rotation	[19]	18.8	31.7	**38.7**	38.2	**36.5**
	Jenni *et al.* [27]	-	[27]	19.5	**33.3**	37.9	**38.9**	34.9
	Mundhenk *et al.* [36]	Context	[36]	**19.6**	31.4	37.0	37.8	33.3
	Noroozi *et al.* [37]	Jigsaw	[39]	18.2	28.8	34.0	33.9	27.1
	Noroozi *et al.* [39]	Counting	[39]	18.0	30.6	34.3	32.5	25.7
	Noroozi *et al.* [38]	Jigsaw++	[38]	18.2	28.7	34.1	33.2	28.0
	Noroozi *et al.* [38]	CC+Jigsaw++	[38]	18.9	30.5	35.7	35.4	32.2
	Pathak *et al.* [41]	In-Painting	[61]	14.1	20.7	21.0	19.8	15.5
	Zhang *et al.* [60]	Colorization	[61]	13.1	24.8	31.0	32.6	31.8
	Zhang *et al.* [61]	Split-Brain	[61]	17.7	29.3	35.4	35.2	32.8