Deep Feature Flow for Video Recognition

Xizhou Zhu[1,2*] Yuwen Xiong[2*] Jifeng Dai[2] Lu Yuan[2] Yichen Wei[2]

[1]University of Science and Technology of China [2]Microsoft Research

ezra0408@mail.ustc.edu.cn {v-yuxio,jifdai,luyuan,yichenw}@microsoft.com

Abstract

Deep convolutional neutral networks have achieved great success on image recognition tasks. Yet, it is non-trivial to transfer the state-of-the-art image recognition networks to videos as per-frame evaluation is too slow and unaffordable. We present deep feature flow*, a fast and accurate framework for video recognition. It runs the expensive convolutional sub-network only on sparse key frames and propagates their deep feature maps to other frames via a flow field. It achieves significant speedup as flow computation is relatively fast. The end-to-end training of the whole architecture significantly boosts the recognition accuracy. Deep feature flow is flexible and general. It is validated on two video datasets on object detection and semantic segmentation. It significantly advances the practice of video recognition tasks. Code would be released.*

1. Introduction

Recent years have witnessed significant success of deep convolutional neutral networks (CNNs) for image recognition, *e.g.*, image classification [23, 39, 41, 16], semantic segmentation [28, 4, 50], and object detection [13, 14, 12, 34, 8, 27]. With their success, the recognition tasks have been extended from image domain to video domain, such as semantic segmentation on Cityscapes dataset [6], and object detection on ImageNet VID dataset [36]. Fast and accurate video recognition is crucial for high-value scenarios, *e.g.*, autonomous driving and video surveillance. Nevertheless, applying existing image recognition networks on individual video frames introduces unaffordable computational cost for most applications.

It is widely recognized that image content varies slowly over video frames, especially the high level semantics [45, 51, 21]. This observation has been used as means of regularization in feature learning, considering videos as unsupervised data sources [45, 21]. Yet, such data redundancy and continuity can also be exploited to reduce the computation cost. This aspect, however, has received little attention for video recognition using CNNs in the literature.

Modern CNN architectures [39, 41, 16] share a common structure. Most layers are convolutional and account for the most computation. The intermediate convolutional feature maps have the same spatial extent of the input image (usually at a smaller resolution, *e.g.*, $16\times$ smaller). They preserve the spatial correspondences between the low level image content and middle-to-high level semantic concepts [47]. Such correspondence provides opportunities to cheaply propagate the features between nearby frames by spatial warping, similar to optical flow [17].

In this work, we present *deep feature flow*, a fast and accurate approach for video recognition. It applies an image recognition network on sparse key frames. It propagates the deep feature maps from key frames to other frames via a flow field. As exemplifed in Figure 1, two intermediate feature maps are responsive to "car" and "person" concepts. They are similar on two nearby frames. After propagation, the propagated features are similar to the original features.

Typically, the flow estimation and feature propagation are much faster than the computation of convolutional features. Thus, the computational bottleneck is avoided and significant speedup is achieved. When the flow field is also estimated by a network, the entire architecture is trained end-to-end, with both image recognition and flow networks optimized for the recognition task. The recognition accuracy is significantly boosted.

In sum, deep feature flow is a fast, accurate, general, and end-to-end framework for video recognition. It can adopt most state-of-the-art image recognition networks in the video domain. Up to our knowledge, it is the first work to jointly train flow and video recognition tasks in a deep learning framework. Extensive experiments verify its effectiveness on video object detection and semantic segmentation tasks, on recent large-scale video datasets. Compared to per-frame evaluation, our approach achieves unprecedented speed (up to $10\times$ faster, real time frame rate) with moderate accuracy loss (a few percent). The high performance facilitates video recognition tasks in practice. Code

*This work is done when Xizhou Zhu and Yuwen Xiong are interns at Microsoft Research Asia

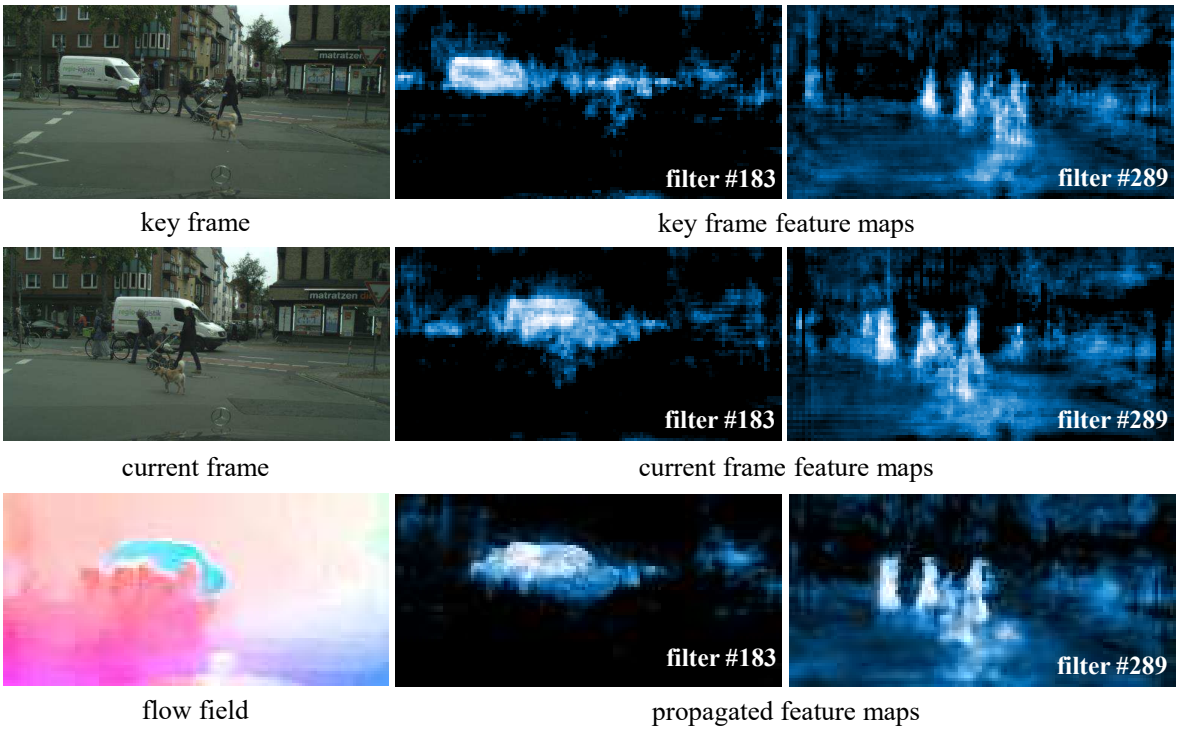


Figure 1. Motivation of proposed *deep feature flow* approach. Here we visualize the two filters' feature maps on the last convolutional layer of our ResNet-101 model (see Sec. 4 for details). The convolutional feature maps are similar on two nearby frames. They can be cheaply propagated from the key frame to current frame via a flow field.

would be released.

2. Related Work

To our best knowledge, our work is unique and there is no previous similar work to directly compare with. Nevertheless, it is related to previous works in several aspects, as discussed below.

Image Recognition Deep learning has been successful on image recognition tasks. The network architectures have evolved and become powerful on image classification [23, 39, 41, 15, 20, 16]. For object detection, the region-based methods [13, 14, 12, 34, 8] have become the dominant paradigm. For semantic segmentation, fully convolutional networks (FCNs) [28, 4, 50] have dominated the field. However, it is computationally unaffordable to directly apply such image recognition networks on all the frames for video recognition. Our work provides an effective and efficient solution.

Network Acceleration Various approaches have been proposed to reduce the computation of networks. To name a few, in [48, 12] matrix factorization is applied to decompose large network layers into multiple small layers. In [7, 33, 18], network weights are quantized. These techniques work on single images. They are generic and complementary to our approach.

Optical Flow It is a fundamental task in video analysis. The topic has been studied for decades and dominated by variational approaches [17, 2], which mainly address small displacements [43]. The recent focus is on large displacements [3], and combinatorial matching (e.g., DeepFlow [44], EpicFlow [35]) has been integrated into the variational approach. These approaches are all hand-crafted.

Deep learning and semantic information have been exploited for optical flow recently. FlowNet [9] firstly applies deep CNNs to directly estimate the motion and achieves good result. The network architecture is simplified in the recent Pyramid Network [32]. Other works attempt to exploit semantic segmentation information to help optical flow estimation [37, 1, 19], *e.g.*, providing specific constraints on the flow according to the category of the regions.

Optical flow information has been exploited to help vision tasks, such as pose estimation [31]. This work exploits optical flow to speed up general video recognition tasks.

Exploiting Temporal Information in Video Recognition T-CNN [22] incorporates temporal and contextual information from tubelets in videos. The dense 3D CRF [24] proposes long-range spatial-temporal regularization in semantic video segmentation. STFCN [10] considers a spatial-temporal FCN for semantic video segmentation. These works operate on volume data, show improved recognition accuracy but greatly increase the computational cost.

By contrast, our approach seeks to reduce the computation by exploiting temporal coherence in the videos. The network still runs on single frames and is fast.

Slow Feature Analysis High level semantic concepts usually evolve slower than the low level image appearance in videos. The deep features are thus expected to vary smoothly on consecutive video frames. This observation has been used to regularize the feature learning in videos [45, 21, 51, 49, 40]. We conjecture that our approach may also benefit from this fact.

Clockwork Convnets [38] It is the most related work to ours as it also disables certain layers in the network on certain video frames and reuses the previous features. It is, however, much simpler and less effective than our approach.

About speed up, Clockwork only saves the computation of some layers (e.g., $1/3$ or $2/3$) in some frames (e.g., every other frame). As seen later, our method saves that on most layers (task network has only 1 layer) in most frames (e.g., 9 out of 10 frames). Thus, our speedup ratio is much higher.

About accuracy, Clockwork does not exploit the correspondence between frames and simply copies features. It only reschedules the computation of inference in an off-the-shelf network and does not perform fine-tuning or re-training. Its accuracy drop is pretty noticeable at even small speed up. In Table 4 and 6 of their arxiv paper, at 77% full runtime (thus 1.3 times faster), Mean IU drops from 31.1 to 26.4 on NYUD, from 70.0 to 64.0 on Youtube, from 65.9 to 63.3 on Pascal, and from 65.9 to 64.4 on Cityscapes. By contrast, we re-train a two-frame network with motion considered end-to-end. The accuracy drop is small, *e.g.*, from 71.1 to 70.0 on Cityscape while being 3 times faster (Figure 3, bottom).

About generality, Clockwork is only applicable for semantic segmentation with FCN. Our approach transfers general image recognition networks to the video domain.

3. Deep Feature Flow

Table 1 summarizes the notations used in this paper. Our approach is briefly illustrated in Figure 2.

Deep Feature Flow Inference Given an image recognition task and a feed-forward convolutional neutral network $\mathcal{N}$ that outputs result for input image $\mathbf{I}$ as $\mathbf{y} = \mathcal{N}(\mathbf{I})$. Our goal is to apply the network to all video frames $\mathbf{I}_i, i = 0, ..., \infty$, fast and accurately.

Following the modern CNN architectures [39, 41, 16] and applications [28, 4, 50, 13, 14, 12, 34, 8], without loss of generality, we decompose $\mathcal{N}$ into two consecutive sub-networks. The first sub-network $\mathcal{N}_{feat}$, dubbed *feature network*, is fully convolutional and outputs a number of intermediate feature maps, $\mathbf{f} = \mathcal{N}_{feat}(\mathbf{I})$. The second sub-network $\mathcal{N}_{task}$, dubbed *task network*, has specific structures for the task and performs the recognition task over the feature maps, $\mathbf{y} = \mathcal{N}_{task}(\mathbf{f})$.

k	key frame index
i	current frame index
r	per-frame computation cost ratio, Eq. (5)
l	key frame duration length
s	overall speedup ratio, Eq. (7)
$\mathbf{I}_i, \mathbf{I}_k$	video frames
$\mathbf{y}_i, \mathbf{y}_k$	recognition results
$\mathbf{f}_k$	convolutional feature maps on key frame
$\mathbf{f}_i$	propagated feature maps on current frame
$\mathbf{M}_{i\to k}$	2D flow field
$\mathbf{p}$, $\mathbf{q}$	2D location
$\mathbf{S}_{i\to k}$	scale field
$\mathcal{N}$	image recognition network
$\mathcal{N}_{feat}$	sub-network for feature extraction
$\mathcal{N}_{task}$	sub-network for recognition result
$\mathcal{F}$	flow estimation function
$\mathcal{W}$	feature propagation function, Eq. (3)

Table 1. Notations.

Consecutive video frames are highly similar. The similarity is even stronger in the deep feature maps, which encode high level semantic concepts [45, 21]. We exploit the similarity to reduce computational cost. Specifically, the feature network $\mathcal{N}_{feat}$ only runs on sparse key frames. The feature maps of a non-key frame $\mathbf{I}_i$ are propagated from its preceding key frame $\mathbf{I}_k$.

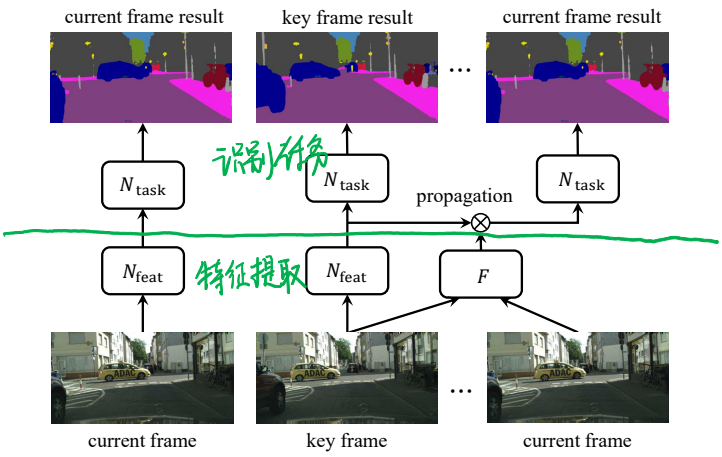
The features in the deep convolutional layers encode the semantic concepts and correspond to spatial locations in the image [47]. Examples are illustrated in Figure 1. Such spatial correspondence allows us to cheaply propagate the feature maps by the manner of spatial warping.

Let $\mathbf{M}_{i\to k}$ be a two dimensional flow field. It is obtained by a flow estimation algorithm $\mathcal{F}$ such as [26, 9], $\mathbf{M}_{i\to k} = \mathcal{F}(\mathbf{I}_k, \mathbf{I}_i)$. It is bi-linearly resized to the same spatial resolution of the feature maps for propagation. It projects back a location $\mathbf{p}$ in current frame i to the location $\mathbf{p} + \delta\mathbf{p}$ in key frame k, where $\delta\mathbf{p} = \mathbf{M}_{i\to k}(\mathbf{p})$.

As the values $\delta\mathbf{p}$ are in general fractional, the feature warping is implemented via bilinear interpolation

$$\mathbf{f}_i^c(\mathbf{p}) = \sum_{\mathbf{q}} G(\mathbf{q}, \mathbf{p} + \delta\mathbf{p}) \mathbf{f}_k^c(\mathbf{q}), \quad (1)$$

where c identifies a channel in the feature maps $\mathbf{f}$, $\mathbf{q}$ enumerates all spatial locations in the feature maps, and $G(\cdot, \cdot)$ denotes the bilinear interpolation kernel. Note that G is two dimensional and is separated into two one dimensional kernels as



(a) per-frame network (b) deep feature flow (DFF) network

Figure 2. Illustration of video recognition using per-frame network evaluation (a) and the proposed deep feature flow (b).

Algorithm 1 Deep feature flow inference algorithm for video recognition.

input: video frames $\{\mathbf{I}_i\}$
$k = 0$; ▷ initialize key frame
$\mathbf{f}_0 = \mathcal{N}_{feat}(\mathbf{I}_0)$
$\mathbf{y}_0 = \mathcal{N}_{task}(\mathbf{f}_0)$
for $i = 1$ **to** ∞ **do**
  if $is\_key\_frame(i)$ **then** ▷ key frame scheduler
    $k = i$ ▷ update the key frame
    $\mathbf{f}_k = \mathcal{N}_{feat}(\mathbf{I}_k)$
    $\mathbf{y}_k = \mathcal{N}_{task}(\mathbf{f}_k)$
  else ▷ use feature flow
    $\mathbf{f}_i = \mathcal{W}(\mathbf{f}_k, \mathcal{F}(\mathbf{I}_k, \mathbf{I}_i), \mathcal{S}(\mathbf{I}_k, \mathbf{I}_i))$ ▷ propagation
    $\mathbf{y}_i = \mathcal{N}_{task}(\mathbf{f}_i)$
  end if
end for
output: recognition results $\{\mathbf{y}_i\}$

$$G(\mathbf{q}, \mathbf{p} + \delta\mathbf{p}) = g(q_x, p_x + \delta p_x) \cdot g(q_y, p_y + \delta p_y), \quad (2)$$

where $g(a, b) = max(0, 1 - |a - b|)$.

We note that Eq. (1) is fast to compute as a few terms are non-zero.

The spatial warping may be inaccurate due to errors in flow estimation, object occlusion, etc. To better approximate the features, their amplitudes are modulated by a "scale field" $\mathbf{S}_{i\to k}$, which is of the same spatial and channel dimensions as the feature maps. The scale field is obtained by applying a "scale function" $\mathcal{S}$ on the two frames, $\mathbf{S}_{i\to k} = \mathcal{S}(\mathbf{I}_k, \mathbf{I}_i)$.

Finally, the feature propagation function is defined as

$$\mathbf{f}_i = \mathcal{W}(\mathbf{f}_k, \mathbf{M}_{i\to k}, \mathbf{S}_{i\to k}), \quad (3)$$

where $\mathcal{W}$ applies Eq.(1) for all locations and all channels in the feature maps, and multiples the features with scales $\mathbf{S}_{i\to k}$ in an element-wise way.

The proposed video recognition algorithm is called *deep feature flow*. It is summarized in Algorithm 1. Notice that any flow function $\mathcal{F}$, such as the hand-crafted low-level flow (e.g., SIFT-Flow [26]), is readily applicable. Training the flow function is not obligate, and the scale function $\mathcal{S}$ is set to ones everywhere.

Deep Feature Flow Training A flow function is originally designed to obtain correspondence of low-level image pixels. It can be fast in inference, but may not be accurate enough for the recognition task, in which the high-level feature maps change differently, usually slower than pixels [21, 38]. To model such variations, we propose to also use a CNN to estimate the flow field and the scale field such that all the components can be jointly trained end-to-end for the task.

The architecture is illustrated in Figure 2(b). Training is performed by stochastic gradient descent (SGD). In each mini-batch, a pair of nearby video frames, $\{\mathbf{I}_k, \mathbf{I}_i\}$[1], $0 \le i - k \le 9$, are randomly sampled. In the forward pass, feature network $\mathcal{N}_{feat}$ is applied on $\mathbf{I}_k$ to obtain the feature maps $\mathbf{f}_k$. Next, a *flow network* $\mathcal{F}$ runs on the frames $\mathbf{I}_i, \mathbf{I}_k$ to estimate the flow field and the scale field. When $i > k$, feature maps $\mathbf{f}_k$ are propagated to $\mathbf{f}_i$ as in Eq. (3). Otherwise, the feature maps are identical and no propagation is done. Finally, task network $\mathcal{N}_{task}$ is applied on $\mathbf{f}_i$ to produce the result $\mathbf{y}_i$, which incurs a loss against the ground truth result. The loss error gradients are back-propagated throughout to update all the components. Note that our training accommodates the special case when $i = k$ and degenerates to the per-frame training as in Figure 2(a).

The flow network is much faster than the feature network, as will be elaborated later. It is pre-trained on the Flying Chairs dataset [9]. We then add the scale function $\mathcal{S}$ as a sibling output at the end of the network, by increasing the number of channels in the last convolutional layer appropriately. The scale function is initialized to all ones (weights and biases in the output layer are initialized as 0s and 1s, respectively). The augmented flow network is then fine-tuned as in Figure 2(b).

The feature propagation function in Eq.(3) is unconventional. It is parameter free and fully differentiable. In back-propagation, we compute the derivative of the features in $\mathbf{f}_i$ with respect to the features in $\mathbf{f}_k$, the scale field $\mathbf{S}_{i\to k}$, and the flow field $\mathbf{M}_{i\to k}$. The first two are easy to compute using the chain rule. For the last, from Eq. (1) and (3), for

[1]The same notations are used for consistency although there is no longer the concept of "key frame" during training.

each channel c and location $\mathbf{p}$ in current frame, we have

$$\frac{\partial \mathbf{f}_i^c(\mathbf{p})}{\partial \mathbf{M}_{i\to k}(\mathbf{p})} = \mathbf{S}_{i\to k}^c(\mathbf{p}) \sum_{\mathbf{q}} \frac{\partial G(\mathbf{q}, \mathbf{p}+\delta\mathbf{p})}{\partial \delta\mathbf{p}} \mathbf{f}_k^c(\mathbf{q}). \quad (4)$$

The term $\frac{\partial G(\mathbf{q},\mathbf{p}+\delta\mathbf{p})}{\partial \delta\mathbf{p}}$ can be derived from Eq. (2). Note that the flow field $\mathbf{M}(\cdot)$ is two-dimensional and we use $\partial\delta\mathbf{p}$ to denote $\partial\delta p_x$ and $\partial\delta p_y$ for simplicity.

The proposed method can easily be trained on datasets where only sparse frames are annotated, which is usually the case due to the high labeling costs in video recognition tasks [29, 11, 6]. In this case, the per-frame training (Figure 2(a)) can only use annotated frames, while DFF can easily use all frames as long as frame $\mathbf{I}_i$ is annotated. In other words, DFF can fully use the data even with sparse ground truth annotation. This is potentially beneficial for many video recognition tasks.

Inference Complexity Analysis For each non-key frame, the computational cost ratio of the proposed approach (line 11-12 in Algorithm 1) and per-frame approach (line 8-9) is

$$r = \frac{O(\mathcal{F}) + O(\mathcal{S}) + O(\mathcal{W}) + O(\mathcal{N}_{task})}{O(\mathcal{N}_{feat}) + O(\mathcal{N}_{task})}, \quad (5)$$

where $O(\cdot)$ measures the function complexity.

To understand this ratio, we firstly note that the complexity of $\mathcal{N}_{task}$ is usually small. Although its split point in $\mathcal{N}$ is kind of arbitrary, as verified in experiment, it is sufficient to keep only one learnable weight layer in $\mathcal{N}_{task}$ in our implementation (see Sec. 4). While both $\mathcal{N}_{feat}$ and $\mathcal{F}$ have considerable complexity (Section 4), we have $O(\mathcal{N}_{task}) \ll O(\mathcal{N}_{feat})$ and $O(\mathcal{N}_{task}) \ll O(\mathcal{F})$.

We also have $O(\mathcal{W}) \ll O(\mathcal{F})$ and $O(\mathcal{S}) \ll O(\mathcal{F})$ because $\mathcal{W}$ and $\mathcal{S}$ are very simple. Thus, the ratio in Eq. (5) is approximated as

$$r \approx \frac{O(\mathcal{F})}{O(\mathcal{N}_{feat})}. \quad (6)$$

It is mostly determined by the complexity ratio of flow network $\mathcal{F}$ and feature network $\mathcal{N}_{feat}$, which can be precisely measured, *e.g.*, by their FLOPs. Table 2 shows its typical values in our implementation.

Compared to the per-frame approach, the overall speedup factor in Algorithm 1 also depends on the sparsity of key frames. Let there be one key frame in every l consecutive frames, the speedup factor is

$$s = \frac{l}{1 + (l-1) * r}. \quad (7)$$

Key Frame Scheduling As indicated in Algorithm 1 (line 6) and Eq. (7), a crucial factor for inference speed is when to allocate a new key frame. In this work, we use a simple fixed key frame scheduling, that is, the key frame duration length l is a fixed constant. It is easy to implement and tune. However, varied changes in image content may require a varying l to provide a smooth tradeoff between accuracy and speed. Ideally, a new key frame should be allocated when the image content changes drastically.

How to design effective and adaptive key frame scheduling can further improve our work. Currently it is beyond the scope of this work. Different video tasks may present different behaviors and requirements. Learning an adaptive key frame scheduler from data seems an attractive choice. This is worth further exploration and left as future work.

	FlowNet	FlowNet Half	FlowNet Inception
ResNet-50	9.20	33.56	68.97
ResNet-101	12.71	46.30	95.24

Table 2. The approximated complexity ratio in Eq. (6) for different feature network $\mathcal{N}_{feat}$ and flow network $\mathcal{F}$, measured by their FLOPs. See Section 4. Note that $r \ll 1$ and we use $\frac{1}{r}$ here for clarify. A significant per-frame speedup factor is obtained.

4. Network Architectures

The proposed approach is general for different networks and recognition tasks. Towards a solid evaluation, we adopt the state-of-the-art architectures and important vision tasks.

Flow Network We adopt the state-of-the-art CNN based *FlowNet* architecture (the "Simple" version) [9] as default. We also designed two variants of lower complexity. The first one, dubbed *FlowNet Half*, reduces the number of convolutional kernels in each layer of FlowNet by half and the complexity to $\frac{1}{4}$. The second one, dubbed *FlowNet Inception*, adopts the Inception structure [42] and reduces the complexity to $\frac{1}{8}$ of that of FlowNet.

The three flow networks are pre-trained on the synthetic Flying Chairs dataset in [9]. The output stride is 4. The input image is half-sized. The resolution of flow field is therefore $\frac{1}{8}$ of the original resolution. As the feature stride of the feature network is 16 (as described below), the flow field and the scale field is further down-sized by half using bilinear interpolation to match the resolution of feature maps. This bilinear interpolation is realized as a parameter-free layer in the network and also differentiated during training.

Feature Network We use ResNet models [16], specifically, the ResNet-50 and ResNet-101 models pre-trained for ImageNet classification as default. The last 1000-way classification layer is discarded. The feature stride is reduced from 32 to 16 to produce denser feature maps, following the practice of DeepLab [4, 5] for semantic segmentation, and R-FCN [8] for object detection. The first block of the conv5 layers are modified to have a stride of 1 instead of 2. The holing algorithm [4] is applied on all the 3×3 convolutional

kernels in conv5 to keep the field of view (dilation=2). A randomly initialized 3×3 convolution is appended to conv5 to reduce the feature channel dimension to 1024, where the holing algorithm is also applied (dilation=6). The resulting 1024-dimensional feature maps are the intermediate feature maps for the subsequent task.

Table 2 presents the complexity ratio Eq. (6) of feature networks and flow networks.

Semantic Segmentation A randomly initialized 1×1 convolutional layer is applied on the intermediate feature maps to produce $(C+1)$ score maps, where C is the number of categories and 1 is for background category. A following softmax layer outputs the per-pixel probabilities. Thus, the task network only has one learnable weight layer. The overall network architecture is similar to DeepLab with large field-of-view in [5].

Object Detection We adopt the state-of-the-art R-FCN [8]. On the intermediate feature maps, two branches of fully convolutional networks are applied on the first half and the second half 512-dimensional of the intermediate feature maps separately, for sub-tasks of region proposal and detection, respectively.

In the region proposal branch, the RPN network [34] is applied. We use $n_{\mathrm{a}} = 9$ anchors (3 scales and 3 aspect ratios). Two sibling 1×1 convolutional layers output the $2n_{\mathrm{a}}$-dimensional objectness scores and the $4n_{\mathrm{a}}$-dimensional bounding box (bbox) regression values, respectively. Non-maximum suppression (NMS) is applied to generate 300 region proposals for each image. Intersection-over-union (IoU) threshold 0.7 is used.

In the detection branch, two sibling 1×1 convolutional layers output the position-sensitive score maps and bbox regression maps, respectively. They are of dimensions $(C+1)k^2$ and $4k^2$, respectively, where k banks of classifiers/regressors are employed to encode the relative position information. See [8] for details. On the position-sensitive score/bbox regression maps, position-sensitive ROI pooling is used to obtain the per-region classification score and bbox regression result. No free parameters are involved in the per-region computation. Finally, NMS is applied on the scored and regressed region proposals to produce the detection result, with IoU threshold 0.3.

5. Experiments

Unlike image datasets, large scale video dataset is much harder to collect and annotate. Our approach is evaluated on the two recent datasets: Cityscapes [6] for semantic segmentation, and ImageNet VID [36] for object detection.

5.1. Experiment Setup

Cityscapes It is for urban scene understanding and autonomous driving. It contains snippets of street scenes collected from 50 different cities, at a frame rate of 17 fps. The train, validation, and test sets contain 2975, 500, and 1525 snippets, respectively. Each snippet has 30 frames, where the 20^{th} frame is annotated with pixel-level ground-truth labels for semantic segmentation. There are 30 semantic categories. Following the protocol in [5], training is performed on the train set and evaluation is performed on the validation set. The semantic segmentation accuracy is measured by the pixel-level mean intersection-over-union (mIoU) score.

In both training and inference, the images are resized to have shorter sides of 1024 and 512 pixels for the feature network and the flow network, respectively. In SGD training, 20K iterations are performed on 8 GPUs (each GPU holds one mini-batch, thus the effective batch size $\times 8$), where the learning rates are 10^{-3} and 10^{-4} for the first 15K and the last 5K iterations, respectively.

ImageNet VID It is for object detection in videos. The training, validation, and test sets contain 3862, 555, and 937 fully-annotated video snippets, respectively. The frame rate is 25 or 30 fps for most snippets. There are 30 object categories, which are a subset of the categories in the ImageNet DET image dataset[2]. Following the protocols in [22, 25], evaluation is performed on the validation set, using the standard mean average precision (mAP) metric.

In both training and inference, the images are resized to have shorter sides of 600 pixels and 300 pixels for the feature network and the flow network, respectively. In SGD training, 60K iterations are performed on 8 GPUs, where the learning rates are 10^{-3} and 10^{-4} for the first 40K and the last 20K iterations, respectively.

During training, besides the ImageNet VID train set, we also used the ImageNet DET train set (only the same 30 category labels are used), following the protocols in [22, 25]. Each mini-batch samples images from either ImageNet VID or ImageNet DET datasets, at $2:1$ ratio.

5.2. Evaluation Methodology and Results

Deep feature flow is flexible and allows various design choices. We evaluate their effects comprehensively in the experiment. For clarify, we fix their default values throughout the experiments, unless specified otherwise. For feature network $\mathcal{N}_{feat}$, ResNet-101 model is default. For flow network $\mathcal{F}$, FlowNet (section 4) is default. Key-frame duration length l is 5 for Cityscapes [6] segmentation and 10 for ImageNet VID [36] detection by default, based on different frame rate of videos in the datasets..

For each snippet we evaluate l image pairs, (k, i), $k = i - l + 1, ..., i$, for each frame i with ground truth annotation. Time evaluation is on a workstation with NVIDIA K40 GPU and Intel Core i7-4790 CPU.

Validation of DFF Architecture We compared DFF with several baselines and variants, as listed in Table 3.

[2] http://www.image-net.org/challenges/LSVRC/

method	training of image recognition network $\mathcal{N}$	training of flow network $\mathcal{F}$
Frame (oracle baseline)	trained on single frames as in Fig. 2 (a)	no flow network used
SFF-slow	same as *Frame*	SIFT-Flow [26] (w/ best parameters), no training
SFF-fast	same as *Frame*	SIFT-Flow [26] (w/ default parameters), no training
DFF	trained on frame pairs as in Fig. 2 (b)	init. on Flying Chairs [9], fine-tuned in Fig. 2 (b)
DFF fix $\mathcal{N}$	same as *Frame*, then fixed in Fig. 2 (b)	same as *DFF*
DFF fix $\mathcal{F}$	same as *DFF*	init. on Flying Chairs [9], then fixed in Fig. 2 (b)
DFF separate	same as Frame	init. on Flying Chairs [9]

Table 3. Description of variants of deep feature flow (*DFF*), shallow feature flow (*SFF*), and the per-frame approach (Frame).

Methods	Cityscapes (l = 5)		ImageNet VID (l = 10)	
	mIoU(%)	runtime (fps)	mAP(%)	runtime (fps)
Frame	71.1	1.52	73.9	4.05
SFF-slow	67.8	0.08	70.7	0.26
SFF-fast	67.3	0.95	69.7	3.04
DFF	**69.2**	5.60	**73.1**	20.25
DFF fix $\mathcal{N}$	68.8	5.60	72.3	20.25
DFF fix $\mathcal{F}$	67.0	5.60	68.8	20.25
DFF separate	66.9	5.60	67.4	20.25

Table 4. Comparison of accuracy and runtime (mostly in GPU) of various approaches in Table 3. Note that, the runtime for *SFF* consists of CPU runtime of SIFT-Flow and GPU runtime of *Frame*, since SIFT-Flow only has CPU implementation.

- *Frame*: train $\mathcal{N}$ on single frames with ground truth.
- *SFF*: use pre-computed large-displacement flow (e.g., SIFT-Flow [26]). *SFF-fast* and *SFF-slow* adopt different parameters.
- *DFF*: the proposed approach, $\mathcal{N}$ and $\mathcal{F}$ are trained end-to-end. Several variants include *DFF fix* $\mathcal{N}$ (fix $\mathcal{N}$ in training), *DFF fix* $\mathcal{F}$ (fix $\mathcal{F}$ in training), and *DFF seperate* ($\mathcal{N}$ and $\mathcal{F}$ are separately trained).

Table 4 summarizes the accuracy and runtime of all approaches. We firstly note that the baseline *Frame* is strong enough to serve as a reference for comparison. Our implementation resembles the state-of-the-art DeepLab [5] for semantic segmentation and R-FCN [8] for object detection. In DeepLab [5], an mIoU score of 69.2% is reported with DeepLab large field-of-view model using ResNet-101 on Cityscapes validation dataset. Our *Frame* baseline achieves slightly higher 71.1%, based on the same ResNet model.

For object detection, *Frame* baseline has mAP 73.9% using R-FCN [8] and ResNet-101. As a reference, a comparable mAP score of 73.8% is reported in [22], by combining CRAFT [46] and DeepID-net [30] object detectors trained on the ImageNet data, using both VGG-16 [39] and GoogleNet-v2 [20] models, with various tricks (multi-scale training/testing, adding context information, model ensemble). We do not adopt above tricks as they complicate the comparison and obscure the conclusions.

SFF-fast has a reasonable runtime but accuracy is significantly decreased. *SFF-slow* uses the best parameters for flow estimation. It is much slower. Its accuracy is slightly improved but still poor. This indicates that an off-the-shelf flow may be insufficient.

The proposed *DFF* approach has the best overall performance. Its accuracy is slightly lower than that of *Frame* and it is 3.7 and 5.0 times faster for segmentation and detection, respectively. As expected, the three variants without using joint training have worse accuracy. Especially, the accuracy drop by fixing $\mathcal{F}$ is significant. This indicates a jointing end-to-end training (especially flow) is crucial.

We also tested another variant of *DFF* with the scale function $\mathcal{S}$ removed (Algorithm 1, Eq (3), Eq. (4)). The accuracy drops for both segmentation and detection (less than one percent). It shows that the scaled modulation of features is slightly helpful.

Accuracy-Speedup Tradeoff We investigate the tradeoff by varying the flow network $\mathcal{F}$, the feature network $\mathcal{N}_{feat}$, and key frame duration length l. Since Cityscapes and ImageNet VID datasets have different frame rates, we tested $l = 1, 2, ..., 10$ for segmentation and $l = 1, 2, ..., 20$ for detection.

The results are summarized in Figure 3. Overall, DFF achieves significant speedup with decent accuracy drop. It smoothly trades in accuracy for speed and fits different application needs flexibly. For example, in detection, it improves 4.05 fps of *ResNet-101 Frame* to 41.26 fps of *ResNet-101 + FlowNet Inception*. The $10\times$ faster speed is at the cost of moderate accuracy drop from 73.9% to 69.5%. In segmentation, it improves 2.24 fps of *ResNet-50 Frame* to 17.48 fps of *ResNet-50 FlowNet Inception*, at the cost of accuracy drop from 69.7% to 62.4%.

What flow $\mathcal{F}$ should we use? From Figure 3, the smallest *FlowNet Inception* is advantageous. It is faster than its two

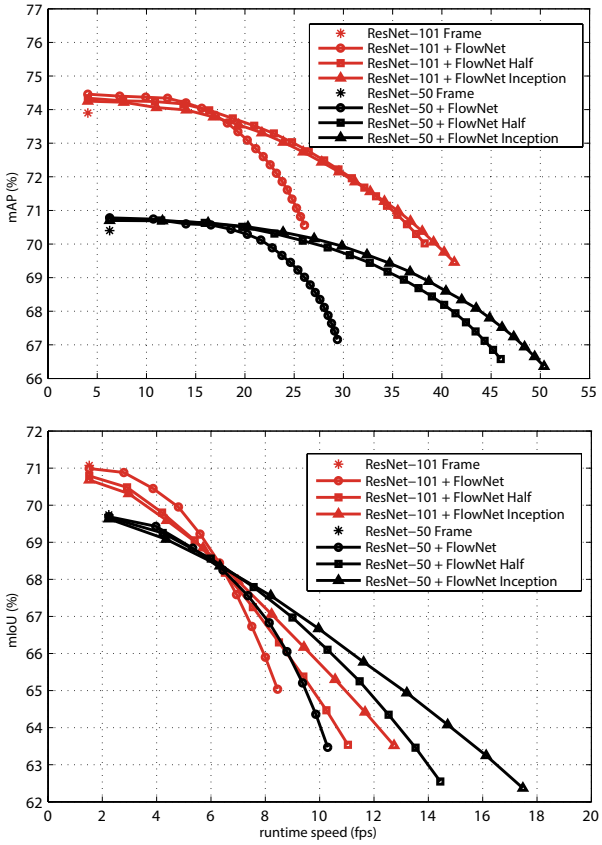


Figure 3. (better viewed in color) Illustration of accuracy-speed tradeoff under different implementation choices on ImageNet VID detection (*top*) and on Cityscapes segmentation (*bottom*).

counterparts at the same accuracy level, most of the times.

What feature $\mathcal{N}_{feat}$ should we use? In high-accuracy zone, an accurate model *ResNet-101* is clearly better than *ResNet-50*. In high-speed zone, the conclusions are different on the two tasks. For detection, *ResNet-101* is still advantageous. For segmentation, the performance curves intersect at around 6.35 fps point. For higher speed, *ResNet-50* becomes better than *ResNet-101*. The seemingly different conclusions can be partially attributed to the different video frame rates, the extents of dynamics on the two datasets. The Cityscapes dataset not only has a low frame rate 17 fps, but also more quick dynamics. It would be hard to utilize temporal redundancy for a long propagation. To achieve the same high speed, *ResNet-101* needs a larger key frame length l than *ResNet-50*. This in turn significantly increases the difficulty of learning.

Above observations provide useful recommendations for practical applications. Yet, they are more heuristic than general, as they are observed only on the two tasks, on limited data. We plan to explore the design space more in the future.

Split point of $\mathcal{N}_{task}$ Where should we split $\mathcal{N}_{task}$ in $\mathcal{N}$? Recall that the default $\mathcal{N}_{task}$ keeps one layer with learning weight (the 1×1 conv over 1024-d feature maps, see Section 4). Before this is the 3×3 conv layer that reduces dimension to 1024. Before this is series of "Bottleneck" unit in ResNet [16], each consisting of 3 layers. We back move the split point to make different $\mathcal{N}_{task}$s with 5, 12, and 21 layers, respectively. The one with 5 layers adds the dimension reduction layer and one bottleneck unit (conv5c). The one with 12 layers adds two more units (conv5a and conv5b) at the beginning of conv5. The one with 21 layers adds three more units in conv4. We also move the only layer in default $\mathcal{N}_{task}$ into $\mathcal{N}_{feat}$, leaving $\mathcal{N}_{task}$ with 0 layer (with learnable weights). This is equivalent to directly propagate the parameter-free score maps, in both semantic segmentation and object detection.

# layers in $\mathcal{N}_{task}$	Cityscapes (l=5)		ImageNet VID (l=10)	
	mIoU(%)	runtime (fps)	mAP(%)	runtime (fps)
21	69.1	2.87	73.2	7.23
12	69.1	3.14	73.3	8.04
5	69.2	3.89	73.2	9.99
1 (default)	69.2	5.60	73.1	20.25
0	69.5	5.61	72.7	20.40

Table 5. Results of using different split points for $\mathcal{N}_{task}$.

Table 5 summarizes the results. Overall, the accuracy variation is small enough to be neglected. The speed becomes lower when $\mathcal{N}_{task}$ has more layers. Using 0 layer is mostly equivalent to using 1 layer, in both accuracy and speed. We choose 1 layer as default as that leaves some tunable parameters after the feature propagation, which could be more general.

Due to limited space, please see example results and more details in the online version of this paper.

6. Future Work

Several important aspects are left for further exploration. It would be interesting to exploit how the joint learning affects the flow quality. We are unable to evaluate as there lacks ground truth. Current optical flow works are also limited to either synthetic data [9] or small real datasets, which is insufficient for deep learning.

Our method can further benefit from improvements in flow estimation and key frame scheduling. In this paper, we adopt FlowNet [9] mainly because there are few choices. Designing faster and more accurate flow network will certainly receive more attention in the future. For key frame scheduling, a good scheduler may well significantly improve both speed and accuracy. And this problem is definitely worth further exploration.

We believe this work opens many new possibilities. We hope it will inspire more future work.

References

[1] M. Bai, W. Luo, K. Kundu, and R. Urtasun. Exploiting semantic information and deep matching for optical flow. In *ECCV*, 2016. 2

[2] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, 2004. 2

[3] T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *TPAMI*, 2011. 2

[4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015. 1, 2, 3, 5

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint*, 2016. 5, 6, 7

[6] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 1, 5, 6

[7] M. Courbariaux, Y. Bengio, and J.-P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NIPS*, 2015. 2

[8] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *NIPS*, 2016. 1, 2, 3, 5, 6, 7

[9] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, and V. Golkov. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015. 2, 3, 4, 5, 7, 8

[10] M. Fayyaz, M. H. Saffar, M. Sabokrou, M. Fathy, and R. Klette. STFCN: spatio-temporal FCN for semantic video segmentation. *arXiv preprint*, 2016. 2

[11] F. Galasso, N. Shankar Nagaraja, T. Jimenez Cardenas, T. Brox, and B. Schiele. A unified video segmentation benchmark: Annotation, metrics and analysis. In *ICCV*, 2013. 5

[12] R. Girshick. Fast R-CNN. In *ICCV*, 2015. 1, 2, 3

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 2, 3

[14] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014. 1, 2, 3

[15] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015. 2

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2, 3, 5, 8

[17] B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial intelligence*, 1981. 1, 2

[18] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations. *arXiv preprint*, 2016. 2

[19] J. Hur and S. Roth. Joint optical flow and temporally consistent semantic segmentation. In *ECCV CVRSUAD Workshop*, 2016. 2

[20] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 2, 7

[21] D. Jayaraman and K. Grauman. Slow and steady feature analysis: higher order temporal coherence in video. In *CVPR*, 2016. 1, 3, 4

[22] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, and X. Wang. T-cnn: Tubelets with convolutional neural networks for object detection from videos. In *CVPR*, 2016. 2, 6, 7

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2

[24] A. Kundu, V. Vineet, and V. Koltun. Feature space optimization for semantic video segmentation. In *CVPR*, 2016. 2

[25] B. Lee, E. Erdenee, S. Jin, and P. K. Rhee. Multi-class multi-object tracking using changing point detection. *arXiv preprint*, 2016. 6

[26] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman. Sift flow: dense correspondence across difference scenes. In *ECCV*, 2008. 3, 4, 7

[27] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. Reed. Ssd: Single shot multibox detector. 2016. 1

[28] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1, 2, 3

[29] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 5

[30] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, and C.-C. Loy. Deepid-net: Deformable deep convolutional neural networks for object detection. In *CVPR*, 2015. 7

[31] T. Pfister, J. Charles, and A. Zisserman. Flowing convnets for human pose estimation in videos. In *ICCV*, 2015. 2

[32] A. Ranjan and M. J. Black. Optical flow estimation using a spatial pyramid network. *arXiv preprint*, 2016. 2

[33] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. *arXiv preprint*, 2016. 2

[34] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1, 2, 3, 6

[35] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow. In *CVPR*, 2015. 2

[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015. 1, 6

[37] L. Sevilla-Lara, D. Sun, V. Jampani, and M. J. Black. Optical flow with semantic segmentation and localized layers. In *CVPR*, 2016. 2

[38] E. Shelhamer, K. Rakelly, J. Hoffman, and T. Darrell. Clockwork convnets for video semantic segmentation. In *ECCV*, 2016. 3, 4

[39] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 2, 3, 7

[40] L. Sun, K. Jia, T.-H. Chan, Y. Fang, G. Wang, and S. Yan. Dl-sfa: deeply-learned slow feature analysis for action recognition. In *CVPR*, 2014. 3

[41] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 1, 2, 3

[42] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 5

[43] J. Weickert, A. Bruhn, T. Brox, and N. Papenberg. A survey on variational optic flow methods for small displacements. In *Mathematical models for registration and applications to medical imaging*. 2006. 2

[44] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. DeepFlow: Large displacement optical flow with deep matching. In *CVPR*, 2013. 2

[45] L. Wiskott and T. J. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural computation*, 2002. 1, 3

[46] B. Yang, J. Yan, Z. Lei, and S. Z. Li. Craft objects from images. In *CVPR*, 2016. 7

[47] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 1, 3

[48] X. Zhang, J. Zou, K. He, and J. Sun. Accelerating very deep convolutional networks for classification and detection. *TPAMI*, 2015. 2

[49] Z. Zhang and D. Tao. Slow feature analysis for human action recognition. *TPAMI*, 2012. 3

[50] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr. Conditional random fields as recurrent neural networks. In *ICCV*, 2015. 1, 2, 3

[51] W. Zou, S. Zhu, K. Yu, and A. Y. Ng. Deep learning of invariant features via simulated fixations in video. In *NIPS*, 2012. 1, 3