Compact CNN for Indexing Egocentric Videos

Yair Poleg Ariel Ephrat Shmuel Peleg Chetan Arora
The Hebrew University of Jerusalem IIIT
Jerusalem, Israel Delhi, India

Abstract

While egocentric video is becoming increasingly popular, browsing it is very difficult. In this paper we present a compact 3D Convolutional Neural Network (CNN) architecture for long-term activity recognition in egocentric videos. Recognizing long-term activities enables us to temporally segment (index) long and unstructured egocentric videos. Existing methods for this task are based on hand tuned features derived from visible objects, location of hands, as well as optical flow.

Given a sparse optical flow volume as input, our CNN classifies the camera wearer's activity. We obtain classification accuracy of 89%, which outperforms the current state-of-the-art by 19%. Additional evaluation is performed on an extended egocentric video dataset, classifying twice the amount of categories than current state-of-the-art. Furthermore, our CNN is able to recognize whether a video is egocentric or not with 99.2% accuracy, up by 24% from current state-of-the-art. To better understand what the network actually learns, we propose a novel visualization of CNN kernels as flow fields.

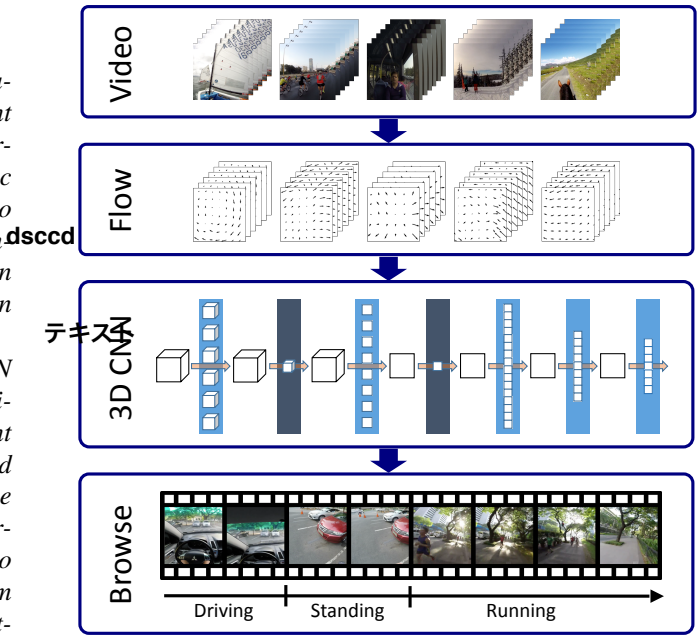

Figure 1. The proposed pipeline for classifying video segments in egocentric video according to the activity of the camera wearer. The CNN takes as input sparse optical flow from several frames arranged in a 3D array. 3D convolutions and 3D pooling are used in the network to preserve temporal structure and facilitate the learning of long term features.

1. Introduction

Recent advances in wearable technologies have made the usage of head mounted camera practical. Such cameras are usually operated in 'always on' mode, providing access to first person point of view which is typically not available with traditional point and shoot cameras. We refer to such videos as egocentric videos. With wearable cameras becoming increasingly affordable, egocentric video recording has become common practice in many areas such as sports, hiking, and law enforcement. While the possibility of sharing one's actions with the community is compelling enough, usage of such cameras for life logging is also on the rise.

Egocentric video gives a novel perspective to capture social and object interactions, but also poses new challenges for the computer vision community. The endless motion from head mounted cameras and the unstructured nature of videos that are shot in an 'always on' mode make egocentric videos challenging to analyze. Recent papers in the field explored a variety of topics such as object recognition [6, 27, 26], activity recognition [3, 5, 28, 31, 22, 32, 21, 20], summarization [16, 18, 1], gaze detection [17] and social interactions [4]. Other tasks such as temporal segmentation [24, 14], frame sampling [35, 25], hyperlapse [15] and camera wearer identification [23, 9, 36] have been explored as well.

Encouraged by the success of neural networks, we investigate CNN architecture for determining the activity of the camera wearer in egocentric videos. Obtaining large enough datasets to train CNNs is always a challenge. In the case of egocentric vision, it is even more difficult due to legal and ethical issues (e.g. privacy). To overcome the

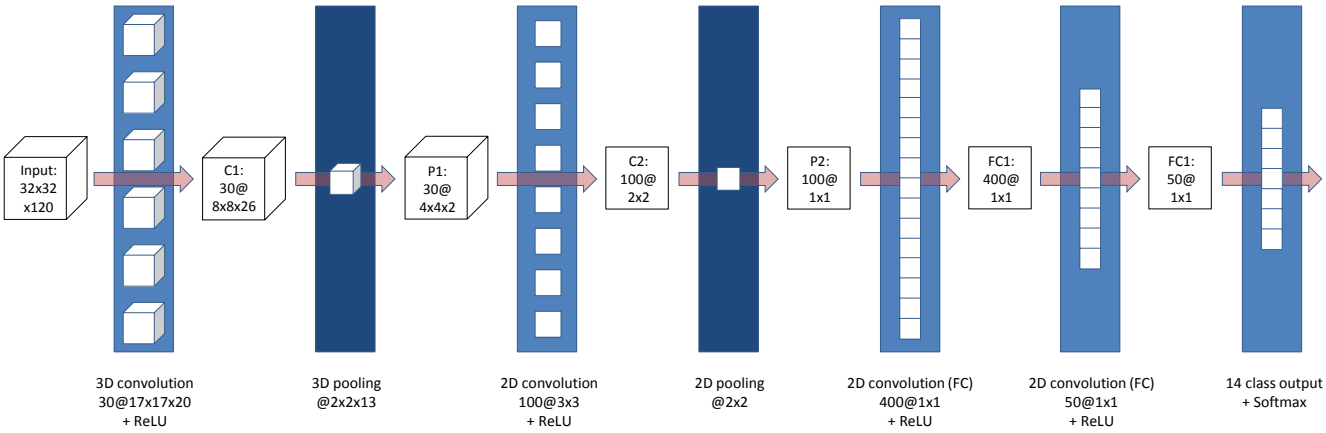

Figure 2. Network architecture. Our network takes as input sparse optical flow computed on a grid of 32×32, stacked over 60 frames. The first hidden layer is a rather long $3D$ convolution followed by $3D$ pooling. Long $3D$ convolutions allow the network to learn features for long term activities, the focus of this paper. Next layers are standard $2D$ convolutional and pooling layers, followed by 2 fully connected layers and terminating with a softmax layer. All hidden weight layers use the ReLU activation function.

data scarcity problem, we propose a compact CNN which is trained on optical flow instead of pixel intensities. Using motion cues for egocentric vision tasks aligns well with recent works in the field (e.g. [28, 14, 29, 24, 23, 33]). Fig. 1 illustrates the proposed pipeline for classifying egocentric videos according to the activity of the camera wearer.

To explore the capabilities of the proposed architecture, we compare it to Poleg *et al*. [24], who used sparse optical flow (without appearance based features) for classifying video into 7 long term activities such as walking, driving, etc. We obtain 89% accurate classification, 19% higher than their original method. We also extend their dataset from 7 to 14 activity classes, and achieve an accuracy of 86% on this new dataset. We also test our network on the task of determining whether a video is egocentric or not. We report accuracy of 99%, up by over 27% on the classification accuracy reported by Tan *et al*. [33]. They suggested to use approximately 50 handcrafted features for this task, based on both intensities and optical flow.

In an attempt to understand what the network actually learns, we analyze the kernels learned by our network for the task of long-term activity recognition. Previous works [30, 11, 7, 2, 10, 37] visualize learned kernels by mapping kernel weights to image intensities. We find this style of visualization difficult for analyzing optical flow based kernels. Instead, we suggest visualizing the learned kernels as optical flow fields. Using this visualization, we show that the proposed CNN learns intuitive and meaningful kernels.

The rest of this paper is organized as follows. We survey related works in Section 2. We present the proposed CNN architecture in Section 3. In Section 4 we evaluate the power of our network on several egocentric tasks and provide insights by analyzing the learned convolutional kernels. We conclude in Section 5.

2. Related Work

Activity Recognition in Egocentric Video There are two main approaches to activity recognition in egocentric video. The first approach is based mainly on appearance, in which object detection, hand recognition and other visual cues are used in order to infer the activity performed by the camera wearer [3, 5, 22, 20, 32]. These methods perform best for short-term tasks in a controlled environment. In this work we focus on long-term activity performed by the camera wearer in the wild.

The second line of works for activity recognition is based on motion analysis. Ogaki *et al*. [21] used motion cues from both an eye-tracking camera and a head-mounted camera to infer egocentric activities. Spriggs *et al*. [31] used inertial motion sensors coupled with a head mounted camera. Our work is based only on a standard head mounted camera. Kitani *et al*. [14] used frequency and motion based features to learn short-term actions in an unsupervised setting. Ryoo and Matthies [28] used global and local motion features to recognize interaction-level activities. The goal of our work is to recognize a set of pre-defined, semantically meaningful, long-term activities.

Recently, Ryoo *et al*. [29] proposed a new feature representation framework based on time series pooling, which is able to abstract detailed short-term/long-term changes in motion/appearance descriptor values over time. In [24] temporal filtering is applied to sparse optical flow in order to recognize long-term activities. Our work is inspired by these works. We generalize the concept of temporal filtering by learning a set of long-term 3D convolution kernels. We apply 3D pooling operators to add robustness against temporal shifts and speed variations in the activities. We report an improvement of 19% on the long term activity recognition task of [24] using our compact CNN architecture.

CNNs for Video using Intensities In recent years several papers have used CNN for video analysis problems. Karpathy *et al*. suggest CNNs for large scale (non egocentric) video classification using a dataset of 1 million YouTube videos, spanning 487 classes [13]. We note the authors' remark that in their architecture there is only a minor difference in the results when operating on single video frame or on a stack of frames. This might indicate that their learned spatio-temporal features do not capture motion well. The fact that motion plays a major role in egocentric vision and the sheer amount of data required to train their network makes their approach less practical for our goals.

Tran *et al*. propose to learn generic features for video analysis by training a deep 3D convolutional neural network [34]. They show that by using $3D$ (spatio-temporal) convolutions, the learned features encapsulate appearance and motion cues. The power of their features is demonstrated on several tasks, including an object recognition task from egocentric videos. In this work we focus on long-term activity recognition from egocentric videos.

CNNs for Video using Optical Flow A compact CNN using sparse optical flow to identify the camera wearer from egocentric video was reported in [9]. Jain *et al*. [10] use a combination of image intensities and instantaneous optical flow from a pair of frames for human pose estimation. Gkioxari and Malik [7] also use instantaneous optical flow and image intensities to perform action recognition and localization. Ji *et al*. [11] use image intensities and optical flow of 5 consecutive frames for human action recognition. To capture the motion component effectively, Simonyan and Zisserman [30] propose a two stream model where one stream handles a stack of image intensities and the other handles a stack of frame-to-frame dense optical flow fields.

Our work differs from all the aforementioned works in several ways: (i) We suggest a CNN architecture tailored for the unique challenges of activity recognition in egocentric video. (ii) Our network is compact with respect to the previous CNNs for activity recognition, enabling training on the relatively small datasets currently available to the egocentric vision community. (iii) The input to our network is built from sparse optical flow (a fixed grid of only 32×32 flow vectors) while all previous CNNs for activity recognition used dense optical flow.

3. CNN Using Sparse Optical Flow

3.1. Network Input

First person activity in egocentric videos is usually manifested over multiple frames. Previous works in egocentric activity recognition have therefore derived features from 2 seconds of video for short term actions [14], and up to 17 seconds of video for long term activities [24]. The input to our network is sparse optical flow derived from 4 seconds. We note the observation made in [30] that their network performs better when it doesn't need to learn to estimate motion implicitly.

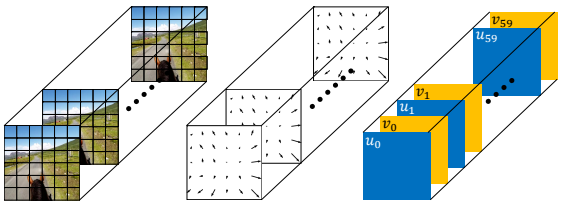

Figure 3. The input to our network is sparse optical flow. Left and Center: We divide each frame into a grid of 32×32 cells and find optical flow independently between each corresponding grid cells of two consecutive frames (one global (x, y) translation pair for each grid cell). Right: We alternately stack the x and y optical flow components corresponding to 60 consecutive frames. The network's input is a data volume of $32{\times}32{\times}120$ elements. Stacking optical flow from such a large number of frames enables the network to learn features for long term activities.

Given a video sequence, we begin by normalizing its frame rate to 15 FPS (frames per second). Next, we divide each video into overlapping blocks of 4 seconds (60 frames). The overlap between consecutive blocks is 2 seconds (30 frames). We then compute sparse optical flow vectors as described in [24, 23, 9] by dividing each frame into a non-overlapping grid of size $32{\times}32$. Optical flow is then estimated between two corresponding grid cells in consecutive frames using LK [19]. An optical flow field is represented by a $32{\times}32{\times}2$ volume, corresponding to the flow in x and y directions. The input to our network is created by stacking 60 such flow fields, resulting in a volume of $32{\times}32{\times}120$ scalars (see Fig. 3).

Formally, let $u_k(i, j)$ and $v_k(i, j)$ denote optical flow in x and y directions between grid cells (i, j) of frame k and frame $k + 1$. Let t denote the time instance (frame number) for which we want to assign an activity label. The input to our CNN at time instance t is the volume $I_t \in \mathbb{R}^{32\times32\times120}$, which is computed from the optical flow as follows:

$$\begin{aligned} I_t(i, j, 2\tau) &= u_{t+\tau}(i, j) \\ I_t(i, j, 2\tau + 1) &= v_{t+\tau}(i, j) \end{aligned} \tag{1}$$

where τ is in the range of $0..59$.

Data normalization for CNNs has become standard practice. We perform the following normalization procedure. Once we obtain all the input volumes I_τ of a certain video dataset, we find the 95th percentile of $u_t(i, j)$ and $v_t(i, j)$ separately and clamp all values to the respective 95th percentile value. We then scale the data to the range $[-1, 1]$.

Figure 4. We extend the dataset of [24] with an additional 16 hours of video, introducing 7 new activity classes. This figure shows representative frames from the 7 new activity classes we introduce (top row) as well as from the original 7 classes. All together, we have 14 activity classes. See Section 4 for details.

3.2. Network Architecture

The first hidden layer in our network C1 is a $3D$ convolutional layer. In this layer we learn 30 kernels of size $17\times17\times20$. The kernels are applied with a spatial stride of 2 and temporal stride of 4 (a stride along the 3rd dimension). By keeping the temporal stride even we ensure that the learned kernels do not mix optical flow from x and y directions. We apply the Rectified Linear Unit (ReLU) nonlinearity to the output of this layer.

The feature maps generated by each of the 30 kernels of C1 are of size $8\times8\times26$. All these outputs are concatenated along the 3rd dimension to give an output volume of size $8\times8\times780$. The exact start time and duration of an activity within an input block can vary. To overcome this, layer P1 pools the feature volume generated by C1 using a 3D max pooling operator of size $2\times2\times13$ and a temporal stride of 13. This means that each feature map generated by a single kernel in C1 is pooled temporally exactly twice. Therefore, the output of P1 is a volume of size $4\times4\times60$.

The second hidden layer C2 is a standard $2D$ convolutional layer with 100 kernels of size 3×3, followed by a ReLU and a max pooling operator of size 2×2. Layers FC1 and FC2 are fully connected of size 400 and 50 respectively. The last layer is a softmax with the number of nodes equal to the number of classes at hand (application dependent). In total, our network contains $287,400$ trainable variables, two orders of magnitude less than [30, 34]. Fig. 2 shows the complete network structure.

3.3. Classification with Temporal Context

The duration of long-term activities can range from a few seconds to minutes. The input to our CNN, on the other hand, covers only 4 seconds. Contradicting short-term actions (e.g. pausing for a second while running) might result in sporadic misclassifications (outliers). As in [9], we add up the softmax scores of η consecutive samples and select the activity label with the highest score. This reflects our prior domain knowledge (e.g. skiing for 4 seconds while running is unlikely). We have explored several values for η in the range of $[0, 30]$ and found that $\eta = 21$, which is equivalent to 44 seconds, gives good results.

3.4. Design Choices

In deep learning, the objective's dependency on hyperparameters is notoriously complex. This problem is amplified when the dataset has its own set of hyperparameters, as in our case (e.g. flow field resolution, temporal receptive field, output filtering etc.). Our primary goal was to design a CNN that performs well on the egocentric tasks at hand, while keeping the number of trainable variables to a minimum to overcome the data scarcity problem.

We first used the 10×5 flow field as in [24]. Temporal receptive fields of $32, 60, 120$, and 240 frames did not yield satisfactory results. We believe this is because the CNN needs more spatial information for the convolution to effectively learn recurring spatial motion patterns in the data. The large 3D kernel size which we converged to ($17\times17\times20$) seems to support this reasoning by indeed learning "meaningful" spatio-temporal patterns.

Increasing the spatial resolution of the flow field to 32×32 and performing a similar temporal length search fared much better. It is possible that a denser optical flow could have performed even better, but with a drastic increase in size of input data and trainable parameters. Reducing the input temporal length to 4 second blocks also helps reduce the number of trainable parameters. Activity duration is accounted for during post-processing, using the scheme described in Sec. 3.3.

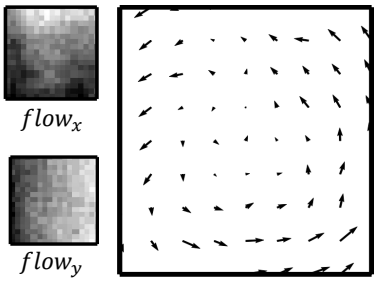

Figure 5. A slice from a $17\times17\times20$ kernel learned by our network for the task of temporal segmentation. The kernel has strong affinity to the *Walking* class. Note: For clarity we show only half of the flow vectors. Visualization of learned kernels by pixel intensities is less informative for a network using optical flow as input. We suggest a visualization where the weights of a $3D$ convolution kernel from two adjacent slices are displayed as an optical flow field. The two images on the left show weights as intensities. The figure on the right shows the same weights as an optical flow field. With this visualization it is clear that the motion pattern learned by the kernel is rotation.

As with CNNs, a better combination of data and network hyperparameters may exist.

3.5. Visualization of Learned Kernels

One way of gaining insight into a fully-trained CNN is through visualization of its learned kernels and input activations [37]. When natural images are used as network input, kernel weights are usually mapped to intensities and visualized as an image. However, the input to our network is a volume composed of stacked flow fields. We therefore suggest visualizing the kernels as vector fields instead of intensity images. We note that each kernel is composed of slices that represent optical flow in the x and y directions. Each such pair is then visualized as flow field. This essentially inverts the stacking process of Equation 1. Fig. 5 shows an example in which the learned z-rotation motion pattern is hard to infer from the intensity images, but becomes clear when visualized as a vector field. While this may seem trivial, to the best of our knowledge it has not done before in the field of CNNs [9, 10, 7, 11, 30].

4. Experiments

In this section we evaluate the performance of the proposed network for several activity recognition tasks in egocentric vision. We do not fine-tune the parameters our network for the different experiments and keep the architecture fixed throughout the experiments.

Implementation Details Our network implementation is based on Caffe [12]. The parameters for the network are fixed as follows: Network weights are initialized using the *Xavier* normalized initialization procedure [8]. The learning rate for all convolutional and fully connected layers is set to 0.01. We use mini-batches of 64 training samples each and stop training after 3000 iterations. It takes 15 minutes to train a network to classify 14 classes with 1300 training samples per class using a single *Nvidia Titan Black* GPU.

Sparse optical flow is estimated using the implementation released by [24]. Their implementation is based on LK [19]. In the rare cases that LK fails to converge, we interpolate the flow value from temporally adjacent frames.

	Recall			
Class	**[24]**	**[29]**	**[34]**	**Ours**
Walking	83%	**91%**	79%	89%
Driving	74%	82%	92%	**100%**
Standing	47%	44%	62%	**79%**
Riding Bus	43%	37%	58%	**82%**
Biking	86%	34%	36%	**91%**
Sitting	62%	70%	62%	**84%**
Static	97%	61%	**100%**	98%
Mean	70%	60%	70%	**89%**

Table 1. Our proposed method does 19% better compared to [24] and [34] on 7 class activity recognition, and 29% better compared to [29]. Our method achieves a high accuracy on *Driving*, while *Sitting, Standing* and *Riding Bus* continue to be the more difficult classes to recognize, as in [24].

4.1. Temporal Segmentation into 7 Classes

Dataset: We evaluate the performance of our architecture on the dataset of [24]. Their dataset contains 65 hours of egocentric videos collected from multiple subjects at Disney World [4], YouTube and others. The subjects perform various tasks, both indoors and outdoors. The dataset contains annotation for 7 activity classes: *Walking, Driving, Riding Bus, Biking, Standing, Sitting, Static*. Two classes out of the 7 (*Static* and *Bus*) had less than 30 minutes of video, which reduced to less than 900 samples per class. In order to train our network, we shot additional sequences for these categories using a *GoPro Hero3* camera. The sequences we shot are similar to the original sequences in the dataset in terms of content (scenario), resolution, FPS and even the camera make. We will release the new sequences and their annotations.

Results: We follow evaluation protocol of [24] for this task. We randomly pick sequences until we have 1300 samples (approximately 90 minutes of video) per class. A sequence from which we take samples for the training set is disqualified from the test set. Table 1 details the classification results. The proposed network is able to reduce the

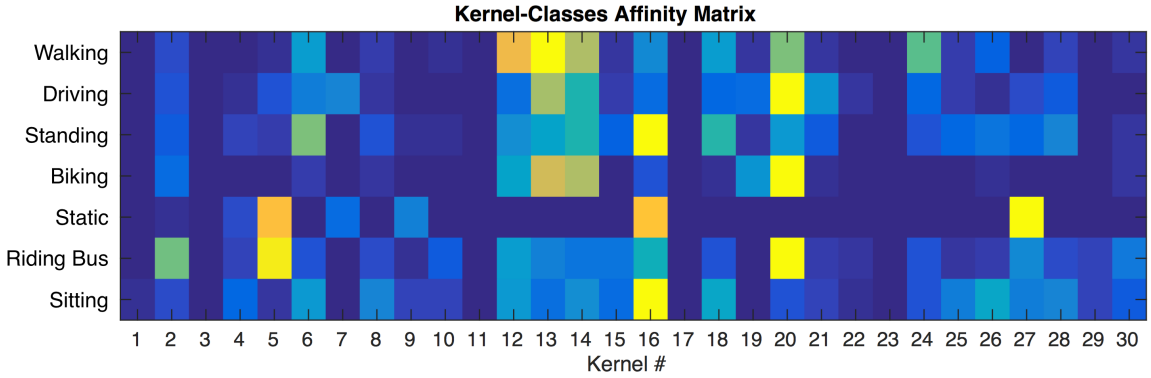

Figure 6. Class-kernel affinity matrix for the first hidden layer. For each sample of a particular class, we find the kernel giving the strongest response. The class-kernel affinity is defined as the number of times the kernel has the strongest response for samples of this class. Brighter colors mean stronger affinity (more votes). We can see in this matrix that kernel #16 has strong affinity to *Standing, Static* and *Sitting*, whereas kernel #20 seems to be popular for *Riding Bus, Biking* and *Driving*. These classes are indeed semantically closer to each other. This indicates that the proposed network may be able to learn meaningful kernels. We cautiously speculate that such affinity matrices can be used to design class hierarchies for classification.

error in all the classes with respect to [24]. The mean error is reduced from 30% to 11%. In addition, we used the code released by [29] and [34] to compare with their methods. Our method outperforms theirs by 29% and 19%, respectively. The worst accuracy is observed for class *Standing*, similar to the current state-of-the-art.

Analysis: To gain insight into what the network is learning, we analyze the kernels of the first convolutional layer (C1). We run the test set through the network again and for each sample find the top 3 kernels that give the highest response with that sample. Each sample then 'casts a vote' to each of these top 3 kernels. The votes are accumulated in an affinity matrix. We expect kernels that received significantly more votes from a particular class to have a strong affinity to that class. Fig. 6 visualizes the affinities between kernel and classes for the 7-class classification task at hand. Brighter colors mean stronger affinity (more votes). We investigate dominant kernels from each class by visualizing their weights as flow fields. Fig. 5 shows the last slice of kernel #24, which has strong affinity with *Walking*. This slice captures rotation about the z-axis, which is a common instantaneous motion in egocentric video captured while walking. We show additional intuitive examples in the next experiment (See Sec. 4.2). It is interesting to see that certain kernels are dominant in several classes. For example, kernel #16 has strong affinities to *Standing, Static* and *Sitting*. Similarly, kernel #20 has strong affinities to *Riding Bus, Biking* and *Driving*. We cautiously regard this observation as another indication that kernels at the first convolutional layer capture characteristic motion patterns.

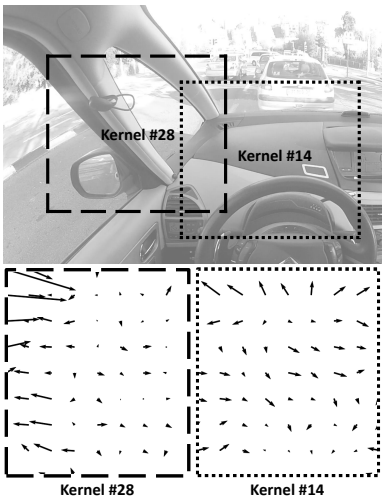

Figure 7. Two slices from two kernels having strong affinity to the *Driving* class (bottom) and possible matching locations (top). The kernels learn 'mixed' flow patterns, that represent locations inside the car and through the car's windows.

4.2. Temporal Segmentation into 14 Classes

Dataset: We extended the original 7-class dataset of [24]. Data for 6 additional classes (14 hours of YouTube videos) was collected by Mechanical Turkers. The *GTEA Gaze+* dataset [5] is used in its entirety to form a 7th new class (*Cooking*). The new classes are: *Running, Stair Climbing, Skiing, Horseback Riding, Sailing, Boxing* and *Cooking*. Fig. 4 shows a representative frame from each new class. The data is distributed evenly across the 7 new classes with roughly 2 hours of footage per class. We have manually annotated the new dataset and will release it and its annotations. All together we have about 82 hours of annotated data for 14 activity classes. Here too we use

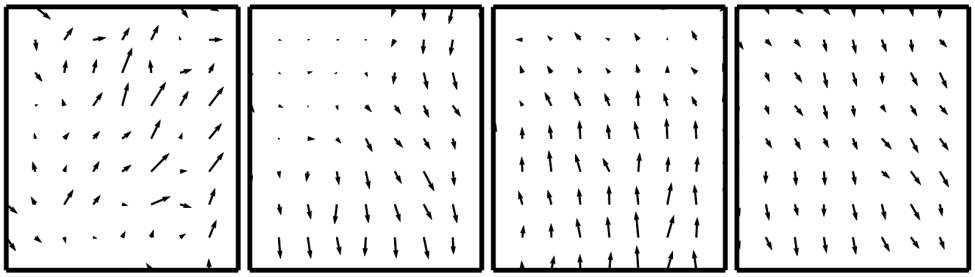

Figure 8. The first 4 slices of the kernel having the strongest affinity for the *Running* class. The flow field visualization helps to see that the kernel learns alternating vertical motions (up/down). It seems that the kernel captures the sharp camera vibration that occurs when the runner's foot hits the ground.

1300 training samples (approximately 90 minutes of video) per class.

Results: We evaluate the performance in terms of Precision, Recall and F-score. The F-score is the harmonic mean of the precision P and recall R, defined as $F = 2PR/(P+R)$. Values range from 0 to 1, where 1 represents perfect performance. Table 2 gives the results of our experiments on the 14 activity classes. We achieve an average recall rate of 86% for this task, a drop of 3% with respect to our performance on the original 7 classes only. This performance drop is expected since we doubled the amount of classes. We note that the network is able to achieve a recall rate of 100% for *Cooking* and 99% for *Sailing* as well as *Static*. Fig. 9 shows the full confusion matrix for this task.

Analysis: Similar to the analysis done in the previous experiment (Sec. 4.1), for each class we find the C1 kernels with the strongest affinity. Fig. 8 shows 4 slices from the kernel that is most associated with *Running*. The kernel's slices capture alternating vertical motions (up/down). It seems that the network learns the sharp camera vibration that occurs when the runner's foot hits the ground. Fig. 7 shows representative slices from the top two kernels associated with class *Driving*. It is interesting to note that kernels learn 'mixed' flow (inside the car and through the car's windows). While the learned flow vectors from inside the car are small and randomly oriented, the vectors from outside the car match those of forward motion.

4.2.1 Evaluation by Transfer Learning

One way of assessing a CNN's generalization capacity is by evaluating the transferability of its learned features to another dataset or task. We attempt this by first training our network on the 7 new classes from the previous experiment (Sec. 4.2). We then fix the weights for all layers in the network except the last, which is initialized randomly. We proceed by training only the last layer on a randomly chosen training subset of the original 7 classes of [24]. Concretely, we train a network with videos of the following 7 classes: *Boxing, Cooking, Skiing, Stair Climbing, Running, Horseback* and *Sailing*. We then transfer the learned weights to a new network and retrain only its classification layer with videos from the following 7 classes: *Walking, Driving, Riding Bus, Standing, Sitting, Static* and *Biking*.

We achieve 73.8% classification accuracy (recall) on the complementary test subset in this setting. As expected, the results are inferior compared to when full network training is done end-to-end. While the accuracy is significantly less than the 89% we report in Table 1, it is still better than the current state-of-the-art. This implies that the learned features of the network are able to generalize to other egocentric activities.

Class	Precision	Recall	F1-Score
Walking	0.93	0.91	0.92
Driving	0.94	0.98	0.96
Standing	0.62	0.59	0.60
Biking	0.92	0.94	0.93
Static	0.44	0.99	0.61
Riding Bus	0.94	0.87	0.91
Sitting	0.73	0.71	0.72
Running	0.91	0.78	0.84
Stair Climbing	1.00	0.59	0.74
Skiing	0.92	0.82	0.87
Horseback	1.00	0.92	0.96
Sailing	0.65	0.99	0.79
Boxing	0.47	0.93	0.62
Cooking	0.80	1.00	0.89
Mean	**0.80**	**0.86**	**0.81**

Table 2. Our network achieves an average recall rate of 86% in the 14 activity classes task. The classes *Drving, Static, Sailing and Cooking* achieve near perfect recall rates. On the other hand, the network struggles with the *Stair Climbing* and *Standing* classes. See confusion matrix in Fig. 9.

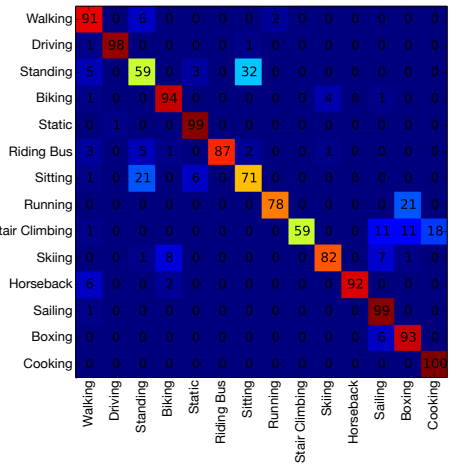

Figure 9. Confusion matrix for 14 activity classes recognition task. A confusion between *Sitting* and *Standing* is evident, similar to [24]. There is room for improving the network's ability to distinguish between *Stair Climbing* and *Sailing*, *Boxing* and *Cooking*.

To further explore this direction, we perform a slightly different experiment. This time, we use the weights learned from training on the 7 new classes to initialize a new network, which we train end-to-end on the original 7 classes. We use only half the training samples (650 instead of 1300), run only 800 iterations (instead of 3000 iterations) and reach an accuracy of 82%. This implies again that the network indeed learns general motion features for egocentric videos. It also shows that given a pre-trained network, generalizing to new classes requires significantly less data and run-time. This is particularly important in egocentric vision due to the data scarcity problem.

4.3. Discriminating Egocentric Videos

Dataset: We use the dataset-of-datasets provided by Tan et al. [33], which contains more than 165 hours of video (approximately 107 hours of egocentric and 59 hours of non-egocentric) covering a diverse set of activities, objects and locations. The egocentric datasets provided have considerable variation, making any attempt to characterize egocentric video as a whole very challenging. Some egocentric videos are shot indoors and with little whole-body movement, while others included walking from indoors to outdoors (and vice-versa). Furthermore, some videos were captured from cameras with wide-angle lenses. Non-egocentric videos were taken from 7 different third-person video datasets containing sports, movie and surveillance footage.

Results: We use the same experimental methodology as suggested by Tan et al. [33]. In the first experiment we divide each dataset in half, and use one half for training and the other for testing. This experiment is referred to as 'Seen', since every dataset has some representation in the training set. The second experiment, referred to as 'Unseen', is performed in an iterative manner. In each iteration, one dataset is left out from the training set and serves as a test set. Our experiments indicate an almost perfect classification accuracy for the 'Seen' experiments, while the accuracy for the 'Unseen' experiments is 90.9%. See Table 3 for a complete comparison. The high accuracy achieved by our network makes it practical for large scale video repositories such as YouTube, etc. This can also potentially automate collecting large amount of samples for new research topics in egocentric vision.

	Seen		Unseen	
	[33]	Ours	[33]	Ours
Egocentric	71%	**99.1%**	62.7%	**90%**
Non-Egocentric	99.3%	**99.4%**	67.1%	**95.3%**
Weighted Mean	75.7%	**99.2%**	63.4%	**90.9%**

Table 3. Comparison with [33] for determining if a video is egocentric or not. Our method achieves high recall rates, making it practical for large scale search applications (e.g., automatic egocentric video collection).

Analysis: We repeat the process of finding the kernels with the strongest affinity to each class and visualize them as optical flow fields. The kernels in this case were more difficult to decipher. We attribute this to the significant inner-class diversity, stemming from the unique properties of each dataset.

5. Concluding Remarks

A convolutional neural network architecture is proposed for long term activity recognition in egocentric videos. The input to the network is a sparse optical flow volume. The network improves the classification accuracy by 19% compared to state-of-the-art for activity-based temporal segmentation. We extend the dataset by 7 more classes and show that the proposed network is able to learn features for the new activities and achieve an overall recall rate of 86% on the 14 activities dataset. Another classification problem supported by our network is discriminating egocentric videos from non-egocentric videos. We achieve a near perfect accuracy of 99.2% in this task. To understand what the network is learning, we visualize the learned kernel as flow fields and analyze affinities between kernels and classes.

Acknowledgment: This research was supported by Intel ICRI-CI and by Israel Science Foundation. Special thanks to Or Sharir for implementation of 3D convolution and pooling.

References

[1] O. Aghazadeh, J. Sullivan, and S. Carlsson. Novelty detection from an ego-centric perspective. In *CVPR*, 2011.

[2] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. *CoRR*, abs/1411.4389, 2014.

[3] A. Fathi, A. Farhadi, and J. M. Rehg. Understanding egocentric activities. In *ICCV*, pages 407–414, 2011.

[4] A. Fathi, J. K. Hodgins, and J. M. Rehg. Social interactions: A first-person perspective. In *CVPR*, 2012.

[5] A. Fathi, Y. Li, and J. Rehg. Learning to recognize daily actions using gaze. In *ECCV*, pages 314–327. 2012.

[6] A. Fathi, X. Ren, and J. Rehg. Learning to recognize objects in egocentric activities. In *CVPR*, pages 3281–3288, June 2011.

[7] G. Gkioxari and J. Malik. Finding action tubes. *CoRR*, abs/1411.6031, 2014.

[8] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

[9] Y. Hoshen and S. Peleg. Egocentric video biometrics. *CoRR*, abs/1411.7591, 2014.

[10] A. Jain, J. Tompson, Y. LeCun, and C. Bregler. Modeep: A deep learning framework using motion features for human pose estimation. *CoRR*, abs/1409.7963, 2014.

[11] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(1):221–231, Jan. 2013.

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

[13] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[14] K. M. Kitani, T. Okabe, Y. Sato, and A. Sugimoto. Fast unsupervised ego-action learning for first-person sports videos. In *CVPR*, 2011.

[15] J. Kopf, M. Cohen, and R. Szeliski. First-person hyperlapse videos. *ACM Transactions on Graphics*, 33(4), August 2014.

[16] Y. J. Lee, J. Ghosh, and K. Grauman. Discovering important people and objects for egocentric video summarization. In *CVPR*, 2012.

[17] Y. Li, A. Fathi, and J. M. Rehg. Learning to predict gaze in egocentric video. In *ICCV*, pages 3216–3223, 2013.

[18] Z. Lu and K. Grauman. Story-driven summarization for egocentric video. In *CVPR*, 2013.

[19] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 2, 1981.

[20] K. Matsuo, K. Yamada, S. Ueno, and S. Naito. An attention-based activity recognition for egocentric video. In *CVPRW*, pages 565–570, June 2014.

[21] K. Ogaki, K. M. Kitani, Y. Sugano, and Y. Sato. Coupling eye-motion and ego-motion features for first-person activity recognition. In *CVPRW*, 2012.

[22] H. Pirsiavash and D. Ramanan. Detecting activities of daily living in first-person camera views. In *CVPR*, 2012.

[23] Y. Poleg, C. Arora, and S. Peleg. Head motion signatures from egocentric videos. In *ACCV*, 2014.

[24] Y. Poleg, C. Arora, and S. Peleg. Temporal segmentation of egocentric videos. In *CVPR*, pages 2537–2544, 2014.

[25] Y. Poleg, T. Halperin, C. Arora, and S. Peleg. Egosampling: Fast-forward and stereo for egocentric videos. In *CVPR*, 2015.

[26] X. Ren and C. Gu. Figure-ground segmentation improves handled object recognition in egocentric video. In *CVPR*, 2010.

[27] X. Ren and M. Philipose. Egocentric recognition of handled objects: Benchmark and analysis. In *CVPRW*, 2009.

[28] M. S. Ryoo and L. Matthies. First-person activity recognition: What are they doing to me? In *CVPR*, 2013.

[29] M. S. Ryoo, B. Rothrock, and L. Matthies. Pooled motion features for first-person videos. *CoRR*, abs/1412.6505, 2014.

[30] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, pages 568–576. 2014.

[31] E. Spriggs, F. D. L. Torre, and M. Hebert. Temporal segmentation and activity classification from first-person sensing. In *CVPRW*, 2009.

[32] S. Sundaram and W. Mayol-Cuevas. High level activity recognition using low resolution wearable vision. In *CVPRW*, 2009.

[33] C. Tan, H. Goh, V. Chandrasekhar, L. Li, and J. Lim. Understanding the nature of first-person videos: Characterization and classification using low-level features. In *CVPRW*, pages 549–556, 2014.

[34] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV (to appear)*, 2015.

[35] B. Xiong and K. Grauman. Detecting snap points in egocentric video with a web photo prior. In *ECCV*, 2014.

[36] R. Yonetani, K. M. Kitani, and Y. Sato. Ego-surfing first person videos. In *CVPR*, 2015.

[37] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833. 2014.