First-Person Action Recognition: a temporal and motion approach

Valeria Sorrenti
Politecnico di Torino
s276146@studenti.polito.it

Alessio Mongoli
Politecnico di Torino
s267501@studenti.polito.it

Simone Santia
Politecnico di Torino
s275090@studenti.polito.it

Abstract

In the last years there has been a growing interest in analysing human daily activity from data collected by wearable cameras. The automatic recognition of egocentric activities has several potential health applications such as assistive and rehabilitative technology. The first-person action recognition is more challenging than third person activity recognition due to unavailability of wearer's pose and sharp movements in the video's caused by the natural head motion of the wearer. Most works addressed this issue with two stream architectures, one dedicated appearance of objects involved in the video, another dedicated to extracting motion features from optical flow. In this paper, starting from an architecture in a single stream with an auxiliary task for motion features, we try to integrate another auxiliary task such as the identification of the correct order to learn better temporal correlations between frames.

1. Introduction

Extract information from videos has always been challenging in computer vision. Many studies and researches has been made in security, autonomous driving, human-computer interaction, robotics and many other fields. Historically, attention has always been on third-person action and activity recognition, where a lot of good results are achieved. In the last years, researches on first-person action recognition has been accomplished, this is a field where studies are in evolution yet. When we move on first-person action recognition arise many issues. One of these is the presence of strong egomotion due to the frequent movements of the actor, because camera is mounted on the actor. Another one is the scarcity of information on egomotion, hence there isn't enough material for training deep neural networks. Most egocentric videos contain action where arms of actor interact with objects, so the only visible part is the arms and the hands of actor interacting with these object.

Recent work, pays attention on two aspects: visual appearance of the object of interest and the motion information. Information on appearance is extract processing RBG images, instead, the movement component is handled by temporal stream that takes in input optical flow extracted from adjacent frames [7].

Many recent approaches implement two-stream architecture [7] with attention modules used for identifying the regions of the frames that have more useful information.

Other approaches holds only one stream and adds auxiliary tasks on the network, in such a way that train in the same way appearance and motion features.

Our approach starts from architecture of [5], we use a single stream network adding two self-supervised tasks. The backbone of our network is a ResNet-34 followed with a standard ConvLSTM, for exploiting appearance features. At the end of convolutional block of ResNet-34 a Motion Segmentation block [5] exploits movements features and we added an Order Prediction Block that uses temporal order of the frames [3]. Auxiliary tasks have a strong advantage, the information produced by these is directly encoded in the inner layers of the backbone. The first auxiliary block focuses on object movements, a piece of information that help the network for the action recognition task. The last one block has the aim to recover the temporal coherence of video by observing how object move in the scene.

As we will see later, with the addition of the last auxiliary task accuracy rate it is slightly higher then the previous computed in [5], this means that the to learn temporal order of frame is useful for the main task.

At first, we will discuss about related work and self-supervised learning, then, we will explain our architecture and our experiment.

2. Related Work

2.1. RGB and Warp Flow approaches

When pass from third action recognition to first action recognition there are several problems to hold in consideration, such as camera motion or the occlusion of the hand, that make hard to understand what is the object related to the action. At first, several methods use a single RGB frame for extracting appearance information and neglect how the

spatial-temporal changes. One possible solution in order to consider the spatio-temporal dependencies between frames is the use of a variant of RNN called convolutional long short-term memory (convLSTM) [9], which the presence of convolutional gates allow the propagation of a memory tensor that enables the preservation of spatial structure.
One solution, to overcome the obstacle of the occlusion of the hand is proposed by [1]. They use the idea of class activation map (CAM) as spatial attention to identify the appropriate regions that can discriminate the activity under consideration from others.
But majority of the approaches use two stream networks [7].

- The spatial part in form of individual frame appearance, carries out information about hand segmentation and objects depicted in the video (RGB).

- The temporal part, as motion across the frames, conveys the movement of the observer and the objects for action recognition. The temporal stream processes a short sequence of adjacent flow frames. A possible problem is camera motion and some paper [10] [4] propose to use stacked warp optical flow images obtained by subtracting the camera motion from the optical flow.

After the training of each stream, we concatenate the outputs obtained using the models of each stream that give the best accuracy.

2.2. Self-Supervised Learning and Motion Segmentation Task

Self-Supervised Learning is a technique that allows to learn from unlabeled data. It is a subset of unsupervised learning where outputs are derived by machines that label, categorize, and analyze information on their own then draw conclusions based on connections and correlations. With the introduction of an auxiliary task, it is possible to encode into the first layers of a network knowledge that proves beneficial as initialization when solving a classification problem on related data. About video classification, these kind of tasks, use motion cues and temporal dimension. The last one has been firstly proposed in [2], where convolutional network is train to segment a static frame using motion data obtained from videos. Another auxiliary task that exploit motion cues and optical flow cues is Motion Segmentation [5], its aim is to minimize the discrepancies between a binary map labeling pixels as either moving or static (binary classification) and the object movements predicted by the network when observing a single static RGB frame. the ground-truth motion maps are computed with a method explained in [2], images are improved through the Improved Dense Trajectory (IDT) proposed in [8], it is used to compensate the effect of camera motion.

2.3. Order Prediction Block

Starting from the network of 2 [5], we propose to use sequence sorting as surrogate task for training a CNN so that it learn useful visual representation to recover the temporal coherence of video by observing how objects move in the scene. For this work we implement in our architecture as auxiliary task the Order Prediction Network [3].
We use to n shuffled frames from a video as our input. Similar to the Jigasaw puzzle problem in the spatial domain [2], we formulate the sequence sorting problem as a multi-class classification task. For each tuple of n frames, there are n! possible permutations. But some action are both coherent forward and backward permutations into same class, so we select only the first n!/2. Finally, out of this we select a set of P permutations by following the maximum Hamming distance based algorithm in [6].

3. Architecture Overview

In this section, gradually we explain the implementation of each variation before to present ours.

3.1. Two-stream Network

In the first model we first extract a small number of representative RGB frames for each input video segment. We use a ResNet-34 network pre-trained on imagenet as the backbone architecture to extract image features. Then in order to perform temporal encoding we use a convolutional long short-term memory (ConvLSTM) module. In this way the network track changes in both temporal and spatial dimension in order to lean how an activity evolves. The equations governing the functioning of convLSTM are:

$$i_t = \sigma({w_x}^i * I_t + {w_h}^i * h_{t-1} + b^i) \tag{1}$$

$$f_t = \sigma({w_x}^f * I_t + {w_h}^f * h_{t-1} + b^f) \tag{2}$$

$$\overline{c}_t = \tanh({w_x}^{\overline{c}} * I_t + {w_h}^{\overline{c}} * h_{t-1} + b^{\overline{c}}) \tag{3}$$

$$c_t = \overline{c}_t \odot i_t + c_{t-1} \odot f_t \tag{4}$$

$$o_t = \sigma({w_x}^o * I_t + {w_h}^o * h_{t-1} + b^o) \tag{5}$$

$$h_t = o_t \odot \tanh(c_t) \tag{6}$$

Where i_t, f_t, o_t, c_t and h_t represent the input state, forget state, output state, memory state and hidden state, respectively, of the convLSTM. The trainable weights and biases of the ConvLSTM are represented by w and b. The $\odot$ represent the convolutional operation and the Hadamard product. In ConvLSTM all the gate activations and the hidden states are 3D tensors as opposed to the case with standard fully-connected LSTM which are vectors. The convLSTM layer uses 512 filters in all the gates with a filter size of 3 × 3 and stride 1. When all the input frames are send throught

the network, the memory of the convLSTM module ct is selected as the future describing the entire video frames. Spatial average pooling operation is the applied on ct to obtain the video feature descriptor which is then fed to the classifier layer consisting in a fully connected layer for generating the class-category score. This architecture is trained in two stage. In the first stage we train only the classifier and the ConvLSTM, this part is trained from scratch, in the second stage we train also the last part of Resnet34, in this case in order to avoid overfitting use the first training stage that give a sort of pretraining of ConvLSTM classifier.
From this architecture, has been add Cam before the ConvLSTM [7] as a spatial attention clue to modulate the features extracted from the last convolutional layer of the Resnet. The Cam for class c, $M_c(i)$ can be presented as

$$M_c(i) = \sum_l {w_l}^c f_l(i) \tag{7}$$

Where $f_l(i)$ is the activation of a unit *l* in the final convolutional layer at spatial location i and w_c is the weight corresponding to class *c* for unit *l*. In the architecture 1, for each input frame we compute the Cam using the winning class. The CAM obtained is converted to a probability map by applying the softmax along spatial dimension. This spatial attention is the multiplied with the output of the final convolutional layer using the Hadamard product. This procedure identifies the appropriate regions that can discriminating one action from another in the scene. After this procedure, the image features with spatial attention is led to convLSTM.
Another way to extract temporal features and capturing complex motion dynamics is to exploit optical flow data. Optical flow is a per pixel prediction and the main idea is that it assumes a brightness constancy, meaning it tries to estimate how the pixels brightness moves across the screen over time. This network uses five stacked optical flow images as the input in order to better encode the motion changes occurring in the video. For this, they train a ResNet-34 pre-trained on ImageNet [7].
By concatenating the output of RGB network and Flow network, and then adding a new fully-connected layer on top to get the class-category score, we obtain a two streams network. This network need to be trained in two or more stages and the parameter optimization tend to be difficult [7].

3.2. Single-stream Network with Motion Segmentation

The design of this architecture starts from the network described in the previous subsection [7]. The idea is to hold a single stream architecture adding a self-supervised task (MS) described in [5], removing optical flow stream. In this way spatial and temporal clues are exploited jointly in the egocentric action recognition process. With this design we expect a faster algorithm then the previous described, and an improvement on accuracy rate thanks to the contribution of both motion and spatial features. From action recognition block (explained above), we first extract a small number N of representative RGB frames for each input video segment. While the backbone processes input frames, motion segmentation block replaces the work of optical flow, paying attention on spatial features.
The pretex MS task serves two purposes, it acts as a data-dependent regularizer for action representation learning and it aims at helping the appearance stream learn an embedding that encodes motion clues as well, in this way learned features are more rich and expressive for the main action recognition task.
The action recognition and MS task share a common trunk that is completed by two task-specific heads. The first objective of the learning step consists of estimating the model parameters that minimize a loss $\mathcal{L}_c$ for action recognition head. This loss function is based on cross entropy between the predicted and the true labels:

$$\mathcal{L}_c(x, y) = -\sum_{i=1}^{n} y_i \cdot \log(g(x_i)) \tag{8}$$

Motion Segmentation block receives in input the features extracted from conv5_x block of ResNet. Input size is $7 \times 7 \times 512$, so the only convolutional block reduces their channels to 100. Thenn there is a fully connected layes of size s^2, (thus 49) followed by a softmax and it is train with a loss $\mathcal{L}_{ms}$ based on the pre-pixel cross entropy between the computed label image *l* and the ground truth (which is first downsampled to a size $s \times s$ and then vectorized). the estimated motion map *l* is obtained as a function of both image embedding *x*, which depends on (θ_f), and MS head parameters (θ_{ms}). So we can define loss $\mathcal{L}_{ms}$:

$$\sum_{i=1}^{n} \sum_{k=1}^{N} \sum_{j=1}^{s^2} {m_i}^k(j) \cdot \log({l_i}^k(j)) \tag{9}$$

where *m* is the ground-truth.
Ground-truth on MS is images extract by IDT module [8] described before. The value of each pixel of these images can be black or white, for highlighting region in movement. In other words, MS block do a binary classification, it classifies each pixel and tell us if pixel is in movement or not.
In Figure 2 we can find an overview of the net proposed until this moment and in [5].

We have implemented this task also like a regression problem. The final layer of the neural network has one neuron and the value it returns is a continuous numerical value. The activation function used is a ReLU, this results in a numerical value greater than zero.
Loss function used is Mean Squared Error (MSE), this finds

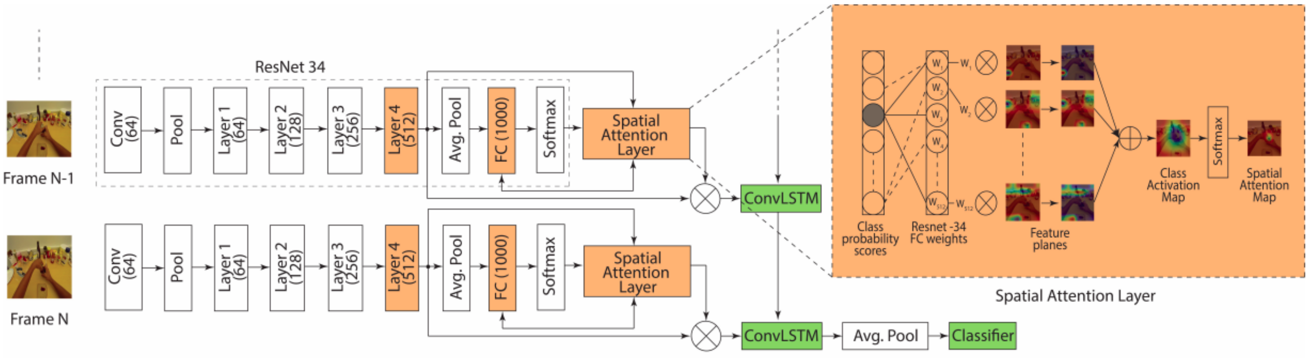


Figure 1. Block diagram of the proposed egocentric action recognition schema. We use ResNet-34 for frame-based feature extraction and spatial attention map generation, and convLSTM for temporal aggregation. Colored blocks indicate trained network layers, we use a two-stage strategy: stage 1 green, stage 2 green+orange.

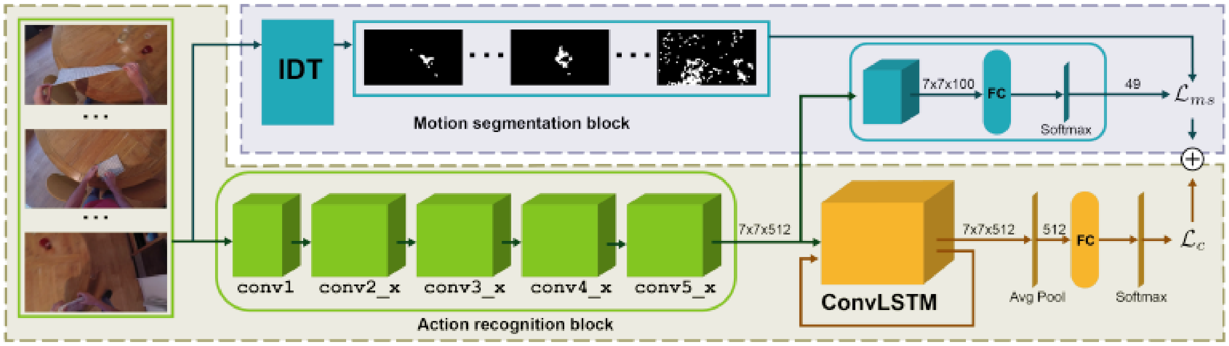


Figure 2. SparNet architecture.

the average squared difference between the predicted value and the true value. MSE function computed is:

$$MSE = 1/n \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \qquad (10)$$

where $\hat{y}$ is the predicted value and y is the true value. Loss function is:

$$\mathcal{L}_m sreg = 1/n \sum_{i=1}^{n} \sum_{j=1}^{s^2} (m_i(j) - l_i(j))^2 \qquad (11)$$

where l is the predicted value and m is the true value.

3.3. Single-stream Network with Motion Segmentation and Order Prediction block

As for the initial implementation, we use the SparNet network with MS task [5]. The OPN has two main components: pairwise extraction and order prediction (Figure 3). The first step receives in input the features extracted from conv5_x block of ResNet-34, reduced in 1024 features (fc6). We take fc6 from every pair of frames for extractions to provide information of the relationship of a frame with another (fc7). Finally, the order prediction is then base on the concatenation of all pairwise feature extractions after one fully connected layer (fc8).
It is trained with a $\mathcal{L}_{op}$ based on cross entropy between the order prediction p and the ground-truth z (the correct permutation applied). It can be defined as:

$$\mathcal{L}_{op} = - \sum_{i=1}^{n} z_i \cdot \log(p(x_i)) \qquad (12)$$

At the end, the optimal model of the net is obtained by solving the following optimization problem.

$$\begin{aligned} &\underset{\theta_M}{\operatorname{argmax}} \, \mathcal{L}(x, y, m|\theta_M) = \\ &\mathcal{L}_c(x, y|\theta_f, \theta_c) + \mathcal{L}_{ms}(x, m|\theta_f, \theta_{ms}) + \\ &\mathcal{L}_{op}(x, z|\theta_f, \theta_{op}) \end{aligned} \qquad (13)$$

Where $\theta_M = \{\theta_f, \theta_{ms}, \theta_c, \theta_{op}\}$.

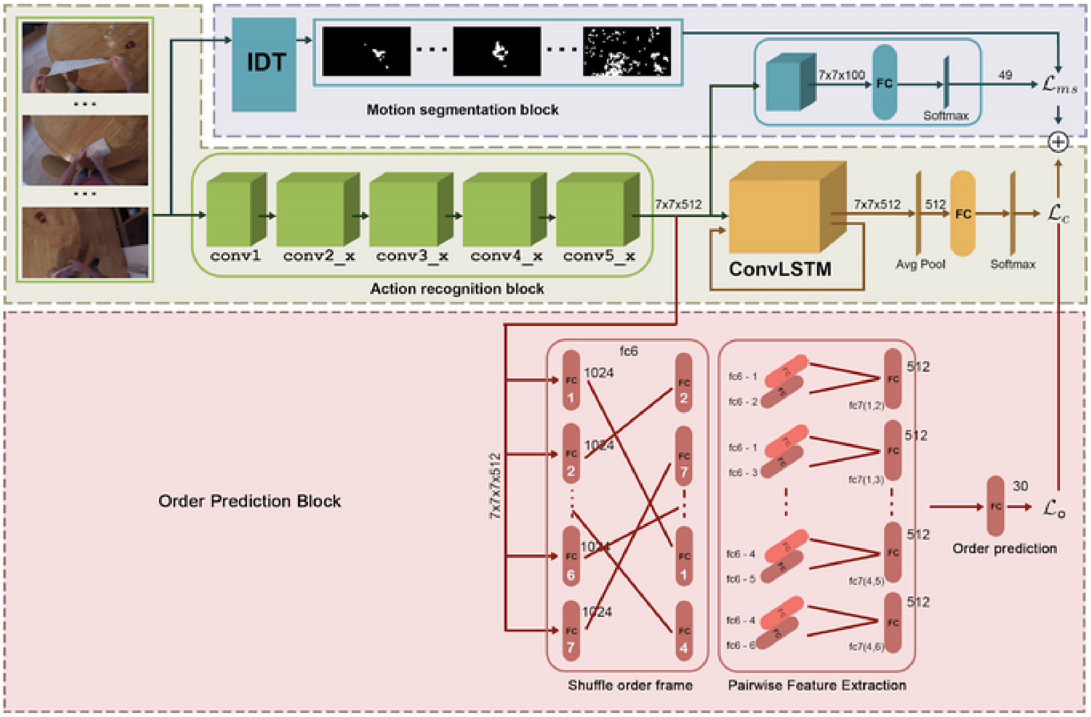


Figure 3. Overview architecture.

All self-supervised loss has the same relevance of the classification loss during the training.

4. Experiments

In this section, we will introduce the dataset used, some implementation details, the description of the training parameters used in the various implementation and the results obtained with the different architecture described in the previous section. Finally we will show the effectiveness of our proposal and the benefit brings by OP block with the improvement of accuracy rate.

4.1. Dataset

All this activity recognition method is evaluated on GTEA 61 dataset. GTEA-61 is an egocentric dataset that includes videos depicting 61 daily activities performed by 4 different subjects. We use subjects S1,S3 and S4 as training sets and subject S2 for validation. Each video of the dataset is already divided into frames, scaled at 256 with the extracted warp flow information. There are three folders: processed_frame which contains RGB frames and ground-truth motion maps for task 2; flow_x_processed and flow_y_processed which contain x and y Warp Flow data. In 4 there is an example of images of the dataset used.

4.2. Implementation Details

In this section we will discuss about step implementation and results for each architecture and set of parameters.

4.2.1 Two-Stream Network

Following paper [7], we run these configurations: ConvLSTM, ConvLstm-attention, Temporal-warp flow, Two stream. For all the configurations, we use a ResNet-34 network pretrained on ImageNet for extracting frame level features. For the configuration with attention we compute the Cam using the winning class through average pooling and a fully connected layer. The Cam obtained, is then converted to a probability map by applying a softmax along the spatial dimensions. This attention map is the multiplied with the output of the last convolutional layer of ResNet-34 with the Handamard product. Instead, in the configuration without attention, do not calculate the Cam and the features of the last convolutional layer are fed to ConvLSTM module with 512 hidden units for temporal encoding. The networks are trained in two stage in order to avoid overfitting section 3.1. We train experiments with both five and seven frames as inputs and we use the following parameter:

Instead the temporal network uses five stacked warp flow images as input in order to better encode the motion changes. Warp optical flow is obtained by subtracting the

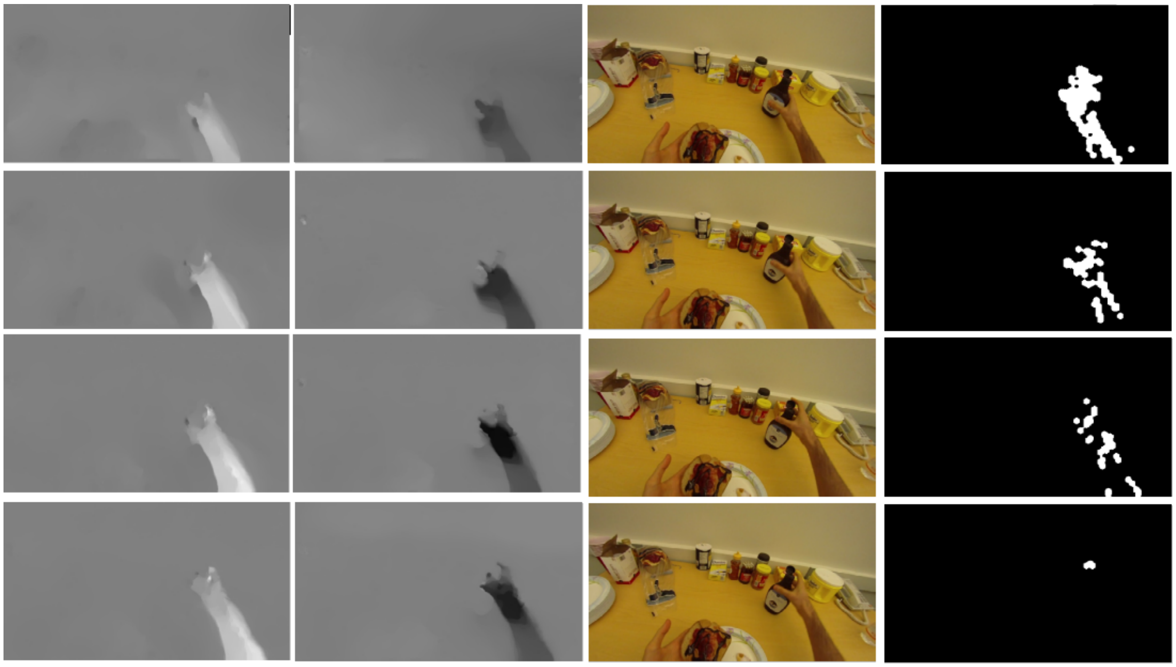


Figure 4. In the first and second column we find images of optical flow, in the second images of RGB, in the last column we find the ground-truth of Motion Segmentation block.

Parameters	**Stage 1**	**Stage 2**
Epochs	200	150
Learning Rate	10^{-3}	10^{-3}
Batch Size	32	32
Dropout	0.7	0.7

camera motion from the optical flow. The temporal network is trained for 750 epochs with a batch size of 32 and learning rate of 10^{-3}.
Once the spatial and temporal network are trained, we can concatenate the output of the two network as described in section 3.1. The network is trained for 250 epochs with learning rate of 10^{-3}, which is reduced by a factor of 0.1 after each epoch.
We evaluated the performance of various configurations on the ”S2” subject of GTEA 61 dataset. We can see that the best results for the discussed architectures are obtained with 16 frames. By adding attention To ConvLSTM we have an improvement of 2% respect ConvLSTM, because the network focus on the regions of the object.

The best results are with the two-stream network that gain a 15% of performance. The network allows to combine complementary features in a way that each activity class becomes more discernible from one another. We have worst results respect paper [1] because we use only 7 or 16 frames as input respect 25 frames. Another possible reason for the result is due by the different splitting of training and validation set.
All results gained with the different configurations are in 1.

4.2.2 Single-stream Network with Motion Segmentation block

We also make experiment using the SparNet proposed in [5], which doesn't need to be trained in three stage like Two-stream architecture. At beginning, we use as input 7 and 16 frames and the same parameters of the previous experiments. Pre-processing consists of resizing the image at height of 256 pixels, and then randomly crops the image of size 224×224. Following the paper [5], we also use some techniques of data augmentation like corner cropping and random horizontal flipping. Starting from this architecture, we try to implement some variants. One variant consists by adding the attention mechanism on Conv5_x; the other variant consists to consider the self-supervised task as a regression problem no longer like a binary classification (moving or static). This second variant is obtained by adding a Relu as activation function and to replace the previous number of neurons with 1. For each variation, we made three runs changing a couple of hyperparameters between Learning rate, Frame, Batch size and weight decay.

As we can see, in the last set of hyperparameters we try to increase the number of frame but in the same time we decrease the batch size for memory issues.

Configurations	Accuracy% 5 frames	Accuracy% 7 frames	Accuracy% 16 frames
ConvLSTM	-	45.55	52.83
ConvLSTM-Attention	-	47.41	54.72
Temporal Warp-Flow	44.83	-	-
Two-stream(joint train	-	56.03	70.70

Table 1. Results on the first architecture on section 3.1.

LR	Frame	Epochs	Batch Size	Weight Decay
1^{-5}	16	150	64	4^{-5}
1^{-4}	16	150	64	4^{-4}
1^{-4}	20	150	8	4^{-5}

Table 2. Set of hyperparameters used for experiments with MS block.

We evaluate the performance of SparNet and his variants taking as baseline the results described before with ResNet34+ConvLstm. The result of his analysis is summarized in the table 3, where we show for different variants of baseline and SpaNet the accuracy taking both 7 and 16 frames as input.

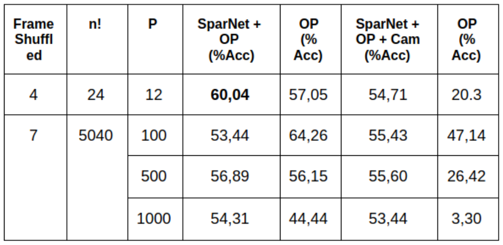
The table 3 suggest that we have with SparNet we achieve 56.6 (16 frames) of accuracy, therefore we have an accuracy gain of 3.77%. It also possible to see that the introduction of the Cam on SparNet, causes a decrease in accuracy of 2.45% respect the baseline taking in consideration 7 frames, but when we increase the number of frame the negative effect disappear and we achieve an accuracy 59,43% (+2.89%) . After a first analysis, we run an hyperparameter optimization and the result are showed in the table 4.

From the table 4 we can see that the Ms achieve an accuracy 64,95% taking as input 20 frames and batch size of 8. We cannot increase the number of frames for the memory issue. The introduction of CAM causes some issue, the same issue of the paper [5]. For this reason, in our implementation we choose an implementation without CAM.

4.2.3 Single-Stream Network with MS and OP block

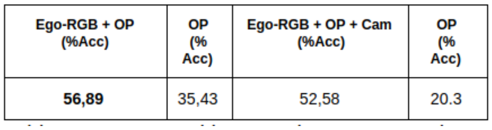
Overall OP block has only two parameters: *n* shuffled frames and *P*, the cardinality of the permutations subset. To train the network, we use the same parameters used so far on SparNet: 150 epochs, lr 0.0001 and step-size [25, 75].
Now we evaluate the impact of the self-supervised auxiliary task on the same dataset. The baseline for our model is SparNet with and without Attention.
For each experiment we made three runs to try to achieve as much as possible their reproducibility. Result are shown in Table 5, where we compare our approach with different frame shuffled and P permutations.

It is interesting to check whether the OP classifier is producing meaningful results per-se, besides supporting generalization for the ego-motion classifier. Order shuffled frames is an hard task but reducing the number of frames and with help of other task of the network, we get a good result for OP task without CAM (in addition to improving the overall result). In Figure 9, the first row shows the accuracy, without CAM, over the learning epochs for the ego-motion and OP classifiers indicating that both grows, differently for training with attention in the second row, as for Motion Segmentation.



Frame Shuffled	n!	P	SparNet + OP (%Acc)	OP (% Acc)	SparNet + OP + Cam (%Acc)	OP (% Acc)
4	24	12	**60,04**	57,05	54,71	20.3
7	5040	100	53,44	64,26	55,43	47,14
		500	56,89	56,15	55,60	26,42
		1000	54,31	44,44	53,44	3,30

Figure 5. Results (3 runs avg) on SparNet with 7 frames.



Ego-RGB + OP (%Acc)	OP (% Acc)	Ego-RGB + OP + Cam (%Acc)	OP (% Acc)
56,89	35,43	52,58	20.3

Figure 6. Accuracy without Motion Segmentation. The attention, like in MS, causes accuracy drops.

We hypothesize that the reasons it could be that both task try to observing how object move in the scene but CAM making the network to focus on the regions consisting of the object and OP needs to be more general.
The figure 10 shows the OP accuracy when changing the number of permutation classes P. Obviously, the performance decreases when the task becomes more difficult.
To try to improve our results, with the same architecture block, we can try the preatraning-finetuning paradigm for evaluating the utility of learnt features. Pretraining is the process of optimizing the weights of a randomly initialized CNN for an auxiliary task that is not the same as the target task. Finetuning is the process of modifying the weights of a pretrained network for the given target task.
ResNet-34 has 0.46M parameters to train up to fc7.
With 300 epoch, lr 10-3, step-size [50,100,150] and a

Configurations	Accuracy% 7 frames	Accuracy% 16 frames
ConvLSTM	47.05	52.83
ConvLSTM-attention	49.41	56.64
MS		
ConvLSTM	49.27	56.67
ConvLSTM-attention	46.96	59.43
Regression		
ConvLSTM	50.86	56.60
ConvLSTM-attention	49.13	55.54

Table 3. Results with MS block.

LR	Frames	Epochs	Batch Size	Weight Decay	MS	MS+CAM	MS+CAM+REG	MS+REG
1^{-5}	16	150	64	4^{-5}	54.72%	39.62%	40.57%	42.45%
1^{-4}	16	150	64	4^{-4}	60.38%	53.77%	54.72%	53.77%
1^{-4}	20	150	64	4^{-5}	64.95%	61.85%	59.79%	60.82%

Table 4. Results with MS block. The best result is highlighted.

dropout rate of 0.5, we train all weights of action recognition block. We take 4 frames for each video and we randomly shuffle the samples to form an input for training network, then the path is the same of OP block. Figure 7.

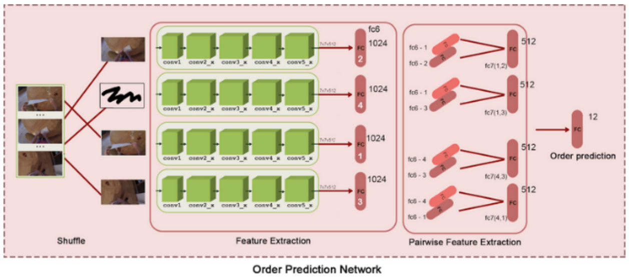
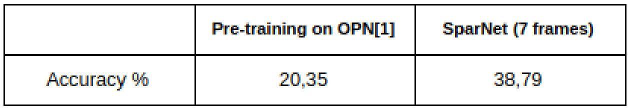


Figure 7. Overview order prediction Network.

Finally, we use the weights of pretrained model on SparNet. The results are shown on Table 8.



	Pre-training on OPN[1]	SparNet (7 frames)
Accuracy %	20,35	38,79

Figure 8. Accuracy preatraning-finetuning paradigm.

We get no improvement with this model. There are many possible reasons, one of all, the dimension of our dataset. To train entire model we use all dataset available, 457 sample, but is not enough. For example ImageNet contains over 14 million images with 1.2 million of them assigned to one of a 1000 categories. Another important problem is the computational power.

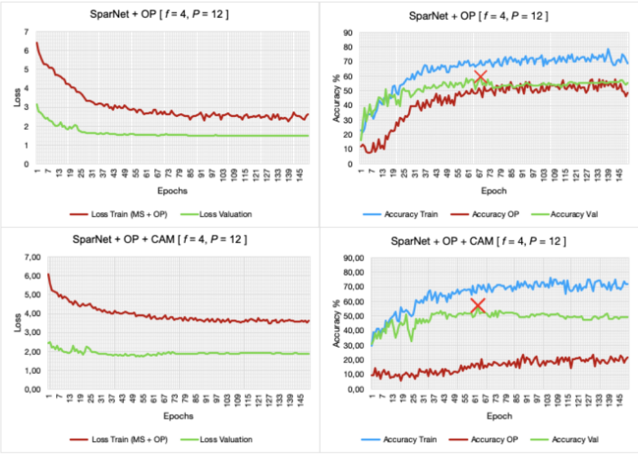


Figure 9. Analysis of the behaviour of SparNet with OP task. The red x are the best accuracy obtained. The trend is similar both losses and accuracies. Despite randomization, the models find the best accuracy in the same epoch.

5. Conclusions

In this paper, we work on the problem of recognition of egocentric activities. We have seen three networks that adopt or combine different approaches to extract functionality from every single frame. In the end we tried to add a second task to the network to recognize the temporal order of the frames and allow CNN to learn a useful visual representation by observing how objects move in the scene. However there are limitations to our approach. First, if two or more frames are the same or very similar, it is impossible to distinguish from each other, by any method. As a consequence, our method would not work reliably on perfectly periodic cases. Another limitation is computation.

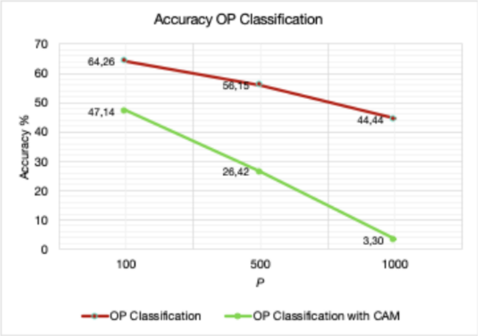


Figure 10. Analysis of the behaviour of the order classifier on the SparNet.

But we believe there are many untapped potentials in self-supervised learning and in the future it will provide a viable alternative to expensive human annotation.

References

[1] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proc. CVPR*, 2017.

[2] P. D. T. D. Deepak Pathak, Ross B. Girshick and B. Hariharan. Learning features by watching objects move. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*.

[3] M. S. M. Y. H. Lee, J.Huang. Unsupervised representation learning by sorting sequence. In *ICCV*, 2017.

[4] Z. W. Y. Q. D. L.-X. T. Limin Wang, Yuanjun Xiong and L. V. Gool. Temporal segment networks: Towards good practices for deep action recognition. In *Proc. CVPR*, 2016.

[5] A. B. Mirco Planamente and B. Caputo. Joint encoding of appearance and motion features with self-supervision for first person action recognition, 2020. arXiv:2002.03982 [cs.CV].

[6] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision (ECCV)*, 2016.

[7] O. L. Swathikiran Sudhakaran. Attention is all we need: Nailing down object-centric attention for egocentric activity recognition. In *arXiv:1807.11794 [cs.CV]*, 2018.

[8] H. Wang and C. Schmid. Action recognition with improved trajectories. In *IEEE International Conference on Computer Vision*, Sydney, Australia, 2013.

[9] H. W. D.-Y. Y. W.-K. W. Xingjian Shi, Zhourong Chen and W. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Proc. NIPS*, 2015.

[10] Z. Y. Yin Li and J. M. Rehg. Delving into egocentric actions. In *Proc. CVPR*, 2015.