First Person Action Recognition from ConvLSTM to Transformer

Andrea Parolin
Politecnico di Torino
Torino, Italy

Gennaro Petito
Politecnico di Torino
Torino, Italy

Gioele Scaletta
Politecnico di Torino
Torino, Italy

Abstract

In this report, we tackle the challenge of first-person action recognition which is recently emerging thanks to the now widespread popularity of wearable devices. As opposed to conventional action classification, first-person action recognition presents multiple additional issues such as dealing with egocentric motions, frequent changes of viewpoints, and diverse global motion patterns which often characterize this kind of videos. And this adds up to the lack of information on the subject performing the action. Firstly, we explored Ego-RNN, a two-stream architecture that separately encodes (temporal) motion and (spatial) appearance information. We then enhanced this architecture by implementing a self-supervised motion prediction pretext task to jointly learn spatial and temporal information so as not to lose the correlations between them. Finally, we proposed our variation [1] . We embraced the current research trend of introducing the transformer in computer vision by replacing the ConvLSTM module with a video transformer network in the first-person action recognition architecture. This to model the temporal relationships between frames with a multi-head self-attention mechanism. The variation led to similar accuracies while keeping the training time lower.

1. Introduction

Action recognition is a very trendy topic and has been widely studied in the last 15 years. The first studies were based on using sensors and accelerometers. Nowadays the spread of wearable devices and cameras like GoPro or smart glasses has brought a big improvement and an increasing interest in the field of first-person action recognition: the increasing popularity of social networks like Youtube, Instagram and Facebook has made available a large number of videos to be analyzed. The real question is what can we learn from these?

The goal we set was to apply labels on segments of first-person videos and recognize the corresponding fine-grained activity. A big problem in this kind of analysis is the enormous amount of variability: thinking about hands' location within the scene they can be visible entirely or partially and their motion can be crucial in a labeling task. A not fixed camera position entails noise, the environment and light condition affect the quality during labeling so we need to exploit as much as possible each available frame. The work is mainly focused on computer vision in particular image recognition, optimizing and introducing our proposal in a CNN image recognition task.

Moreover, the point of this work is fine-grained activity recognition, and this is a real challenge compared to action recognition: typically the label involves an action (verb) and some object (noun). Usually, the verb identifies the motion pattern while the noun relates to the manipulated object. Hence two different pieces of information emerge: visual appearance of the object of interest and motion information.

Ego-RNN [6], the first architecture we explored and implemented, handles separately the spatial stream, that processes RGB images, and the temporal stream, that exploits the optical flow extracted from adjacent frames to model the motion information.

One of the drawbacks of EGO-RNN is that by modeling temporal and spatial information separately, the spatio-temporal relationships may be overlooked. To address this we followed [5], joint modeling was achieved by using a motion prediction self-supervised pre-text task to make the backbone learn an image embedding more prone to motion. In this way, this additional information is directly encoded in the backbone throughout the training process.

Following the great success achieved in natural language processing tasks, the transformer is now becoming more and more employed in the computer vision domain. Thus, we decided to employ it in first-person action recognition too. We consider the frames in a video as words in a sen-

[1] Our implentation of the architectures is available at `https://github.com/gioele-scaletta/ML-DL-FPAR`

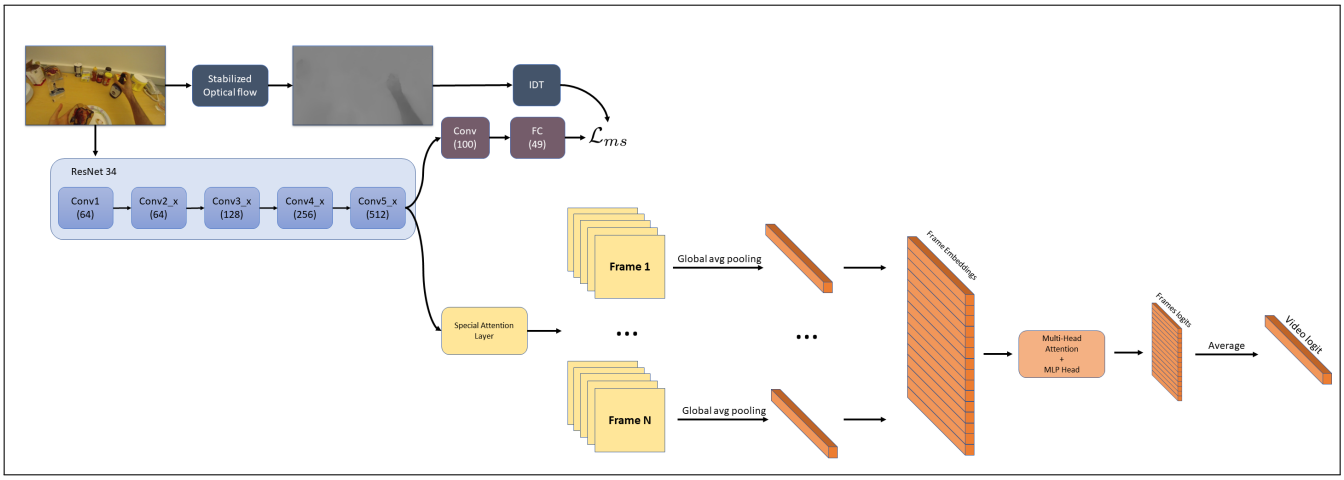

Figure 1: Architecture of this project's proposed variation.

tence. For our variation of the previously mentioned architectures, we replaced the ConvLSTM module with a transformer. In this case, the features extracted with ResNet-34 (spatial backbone) are processed by a transformer to infer global temporal dependencies in sequential data. Finally, an MLP head handles the classification of the output embedding by providing a final predicted category.

2. Related Works

As already briefly mentioned previously, the first two steps of our work consisted in replicating the results obtained by the two papers [6] and [5] while for the last part, in which we included the transformer, our work is based on [4] and [3].

2.1. EGO-RNN

As already explained EGO-RNN presents two streams. The spatial stream uses Class Activation Maps to identify the discriminant regions in the image. This produces feature maps that are then processed through the convLSTM module to obtain the temporal encoding. The architecture is trained in two stages: in the first only the convLSTM and the final classifier are trained, in the second also the last convolutional layer and the fully connected ResNet-34 layer. This is done since the ConvLSTM is trained from scratch while ResNet is pre-trained on ImageNet. Moreover, a temporal network that uses stacked optical flow images as input is added for motion changes. To merge the two networks, two possible approaches can be followed: one consists in averaging the classifications scores and the other in concatenating the outputs of the two networks and adding a fully-connected layer on top to get a final class-category score.

2.2. Self-supervised Motion

The paper [5] emphasizes the importance of jointly modeling temporal and spatial features together and suggests adding three different motion prediction pretext tasks to transform the original architecture into a multi-task network: (1) Motion segmentation: minimize differences between a ground-truth motion map and one predicted by a network observing a single static RGB frame. The ground-truth motion maps are obtained through the Improved Dense Trajectories method [8] that compensates for the strong camera motion typical of first-person videos. (2) Estimate the stabilized flow from a static RGB image as a classification problem. (3) Estimate the stabilized flow from a static RGB image as a regression problem.

2.3. Transformer

In the literature different implementations have been proposed for using transformers for the temporal encoding of video frames. Among these [4] presents a transformer-based framework VTN for video recognition with a generic approach that can be used on top of any 2D spatial network, both transformer or convolution-based. While VTN is optimized for any video length and especially to process long videos, in GTEA61, actions take place in a short window of time, so in our variation, we opted for obtaining all features first and then feeding them to the temporal encoder. Our architecture is similar to the one proposed in [3], where features extracted from the input frames are mapped into feature embeddings and fed into a transformer. However, we thought that augmenting the feature extractor through class activation maps and self-supervised motion prediction tasks would lead to better results.

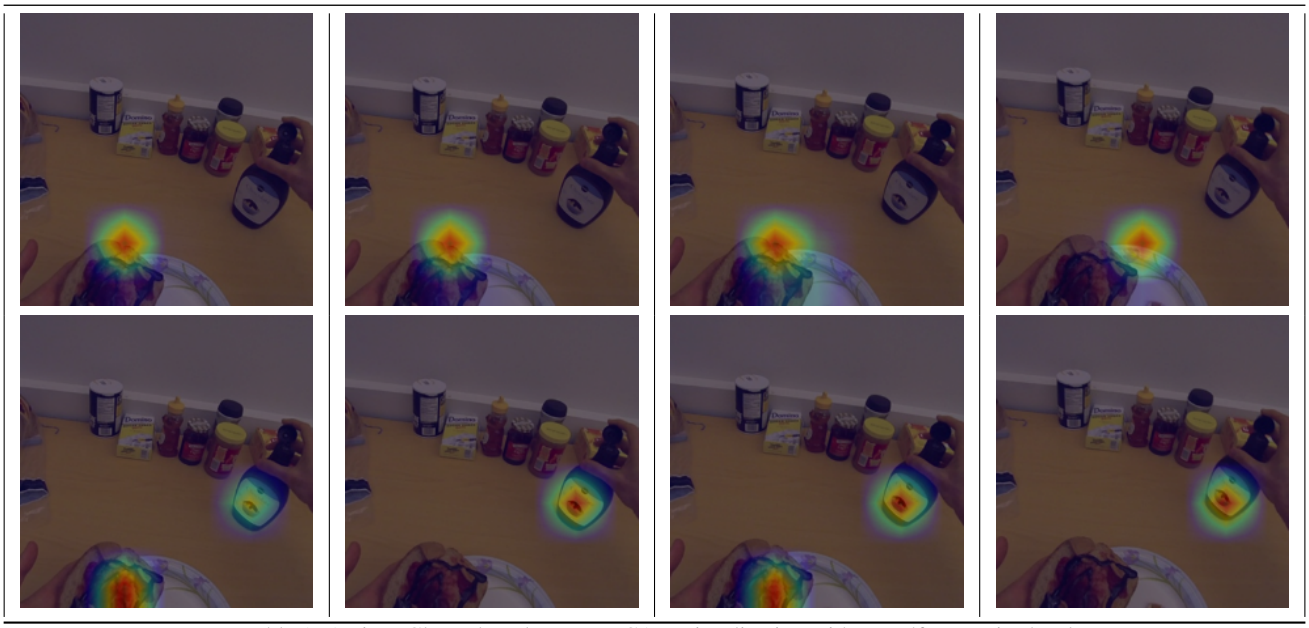

Table 1: Action: Close chocolate. Top: CAM visualization without self-supervised task.
Bottom: CAM visualization with self-supervised task.

3. Proposed Variation

We propose to replace the ConvLSTM with a transformer. Consider the feature maps gathered from a feature extractor (ResNet-34 in our case), suppose these feature maps have size (n_{batch} x n_{frames} x n_{channels} x h x w), that in our case is 32 x 7 x 512 x 7 x 7. First, we have to address an important difference, while the ConvLSTM accepts as input all the feature maps each of size 7x7, in the Vaswani et al. [7] transformer we have to create a vector embedding, so we start from a global average pooling of each channel for each feature map: in this way, we obtain an embedding vector of size nchannels. At this point, we can apply the transformer as described in [7] treating each frame as a word and each video as a sentence. The position of the frame in the video is of the utmost importance, so we had to apply positional encoding: we opted for the same implementation of Vaswani et al. [7]:

$$PE_{(pos,2i)} = \sin\left(pos/10000^{2i/d_{\text{model}}}\right) \quad (1)$$

$$PE_{(pos,2i+1)} = \cos\left(\text{ pos }/10000^{2i/d_{\text{model}}}\right) \quad (2)$$

where d_{model} is the size of the input embedding, pos is the position in the embedding and i the frame. While we opted for multi-head attention, we achieved the best accuracy using a single multi-head layer which was also the best from a computational perspective. Empirically, we observed that, for classification purposes, taking the average of the output embeddings obtained from the frames of the same video was a better option than introducing a [CLS] token as in the Video Transformer Network [4] and Attention Is All You Need [7]. Eventually, we process the output embedding with an MLP head: a linear layer, followed by a GELU activation function, then a dropout layer, and finally another linear layer.

4. Implementation

4.1. Dataset

Our models have been trained on the GTEA-61 dataset, which is made up of 7 different types of daily activities, performed by 4 different subjects. So we decided to use subjects 1, 3, and 4 as training sets and subject 2 as validation set. Along with the RGB frames, we were also provided with the warp flow frames (optical flow frames compensated for camera motion), obtained with the Gunnar-Farneback method [2].

4.2. EGO-RNN

For this first part, we adopted the parameters provided by [6]. The ConvLSTM module was set to have 512 hidden units for temporal encoding. Moreover, the model was trained in two stages: in the first stage, the network was trained for 200 epochs with an initial learning rate of 10^{-3} decayed by a factor of 0.1 after 25, 75 and 150 epochs. We also apply dropout at a rate of 0.7 at the fully-connected classifier layer. In the second stage, we train the network for 150 epochs with a learning rate of 10^{-4} and decayed after 25 and 75 epochs by a factor of 0.1. We use ADAM optimization algorithm with a batch size of 32 during train-

Table 2: Accuracies on GTEA61

Configurations	Accuracy% 7 Frames	Accuracy% 16 Frames
ConvLSTM	50	57.75
ConvLSTM-attention	61.2	67.24
Temporal-warp flow	46.55*	-
two-stream (joint train)	62.86	**69.8**
SS_classification**	65.5	-
SS_regression**	**68.1**	-

* for Temporal-warp flow the number of frames is equal to 5.
** for the self-supervised classification and regression only experiments with 7 frames were conducted.

ing.

For the temporal network, we used the dense optical flow and trained for 750 epochs with a batch size of 32 using stochastic gradient descent algorithm with a momentum of 0.9. The learning rate is initially fixed to 10^{-2} and is then reduced by a factor of 0.5 after 150, 300 and 500 epochs.

4.3. SparNet

For SparNet we did not follow the original paper's [5] implementation, but instead opted also in this case for a two stages implementation: in the first stage, the network was trained for 200 epochs with an initial learning rate of 10^{-3} decayed by a factor of 0.1 after 25, 75 and 150 epochs. We also apply dropout at a rate of 0.7 at the fully-connected classifier layer. In the second stage, we train the network for 150 epochs with a learning rate of 10^{-4} and decayed after 25 and 75 epochs by a factor of 0.1. We use ADAM optimization algorithm with a batch size of 32 during training.

Some of these parameters will be adjusted in the hyperparameter optimization mentioned in section 5.3. As ground truth for the self-supervised motion segmentation task, we used the motion maps obtained through Improved Dense Trajectories [8]. The model for the self-supervised is inserted after the last convolutional layer and takes as input feature maps of sizes 7x7, different activation functions and losses were tested. So we had to downsample the output of the IDT to match the 7x7 size.

The losses of the self-supervised task and of the action recognition were summed. We tried setting a weighted sum but the best results were obtained when the ratio was 1:1.

4.4. Proposed Variation

Also in this case we trained the network in two stages: in the first stage for 200 epochs and the second for 150 epochs. We use the Stochastic Gradient Descent optimization algorithm with a learning rate of 10^{-3}, decayed through cosine annealing with a maximum number of iterations equal to 10. The batch size is set to 32. Regarding the transformer's parameters, we set the number of heads to 8, hence the dimensions of query, key, value and the output of each head, are equal to 64. The latter will then be concatenated to form

Table 3: Accuracies Self-Supervised Regression Task

Loss	Activation Function	Accuracy% 7 Frames
MSE	ReLU	64.65
MSE	sigmoid	63.79
L1	ReLU	61.20
L1	sigmoid	58.62
Smooth L1	ReLU	**68.10**
Smooth L1	sigmoid	67.24
Smooth L1	LeakyReLU(0.01)	65.51
Mod Smooth L1	ReLU	65.51

an output of dimension 512. For the MLP head, the first fully connected layer is set to 2048, which is followed by the fully connected layer of size 512 with a dropout rate of 0.1. Finally, since our dataset is composed of 61 classes, we apply a final linear layer of size 61.

5. Results

5.1. EGO-RNN

We start from various implementations of EGO-RNN, analyzing how the different elements contribute to the accuracy. In Table 2 we can see that the experiments were carried with 7 and 16 frames, apart from "temporal-warp flow" where only 5 frames were used. In all the cases more frames resulted in higher accuracy. First of all, we notice that the addition to the model of the class activation map, which enhances attention by localizing the object being handled in the action, greatly improves the accuracy. At this point, we add a second stream, temporal, that has as input the warp optical flow. In this way, we obtain a further improvement, despite the very low accuracy that is achieved when only using warp optical flow in a temporal stream network.

5.2. Self-Supervised Task

Now following [5] we add a motion segmentation task. We employ both a classification and a regression task. For the classification we had to binarize the motion maps, among the various thresholds empirically we found 10^{-2} to be the best option. From the results, we notice that the accuracy is slightly higher when treating the task as a regression problem. This is in line with our intuition: while with classification it is verified only whether the motion happens or not in a certain area of the frame, with regression also a hint on the entity of the motion is obtained.

Regarding the regression task, as shown in Table 3, we use different types of losses and activation functions to assert the best combination. The highest accuracy is reached with Smooth L1 Loss and ReLU: in regressions of this kind, normally, a better result would be expected with MSE loss. However, this discrepancy is explained by the regression not being straight-on, but instead much more complicated.

In Figure 1 we also show the difference between the class activation maps when the self-supervised task is present or not. These show that the attention mechanism is improved

Table 4: Hyperparameter Optimization Classification

LearnRate	LearnRate Decay	StepSize	WeightDecay	BatchSize	Accuracy% 7 Frames
1e-3/1e-4	0.1/0.1	[25,100,150]/[25,75]	4e-5/4e-5	32	62.93
1e-3/5e-5	0.05/0.25	[40,90,150,220]/[35,85,120]	1e-4/5e-3	32	57.77
5e-4/5e-5	0.1/0.1	[40,90,150,220]/[25,75,120]	4e-5/5e-4	32	49.13
2e-3/2e-4	0.2/0.3	[30,90,140]/[25,75,100]	8e-5/1e-4	64	**65.50**

providing both a more accurate localization of the object and richer motion information in the feature extractor.

5.3. Hyperparameters Optimization

In table 4 we have the results of hyperparameter optimization for the classification MS task: the parameters for which we tested were the learning rate, learning rate decay, step size, weight decay and batch size. The same parameters were tested for the regression task, where we used the Smooth L1 Loss and ReLU activation function since it previously returned the best accuracy.

5.4. Transformer Variation

The results are reported in Table 5. At first, we only used the feature maps from the ResNet-34, then the class activation map was added and this entailed a great improvement of 8.62%.

At this point following [5], we decided to add a motion segmentation task for self-supervision we have shown that this task can help the feature extractor to gather information about the motion in each frame and improve the attention mechanism. This however returns a lower accuracy than when only using the class activation map for attention. This is explained by the fact that the transformer, as opposed to the convLSTM, does not preserve the spatial information of the feature maps. However, we would have expected this handicap to be balanced by the improvement in the class activation map for attention, as we had seen previously in our SparNet implementation.

Therefore, we concatenated the embedding of length 512 with the vector of size 49 that comes as the output of the motion segmentation task, to provide to the transformer a hint on where the motion is located. However, this did not improve the results that are in line with the model without neither class activation map nor motion segmentation as self-supervised task. So we decided to also concatenate the class activation map to provide to the transformer information on where the handled object is, together with where the motion is happening. This does improve the accuracy which is however still lower than when we only use the class activation map to boost the attention and unchanged embedding. Note that, when concatenating, we used inputs for the transformer of size equal to 610, so we had to increase the number of heads to 10 and the size of the key, value and query were equal to 61.

Table 5: Accuracies on GTEA61 with our variation

Configurations	Accuracy% 7 Frames
Transformer	56.89
Transformer CAM attention	**65.51**
Transformer CAM attention and Self Supervised	62.93
Transformer concat Self Supervised*	56.89
Transformer concat Self Supervised and CAM*	59.48
Transformer concat CAM*	56.89

* with concatenation the embedding dimension increases from 512 to 610 and the number of heads from 8 to 10.

Table 6: Comparison 7 Frames

Configurations	MParams	GMACS	Accuracy%
ConvLSTM SS-regression*	40.73	32.17	68.10
Transformer CAM attention	51.23	25.75	65.51
Transformer concat SS and CAM	57.15	25.79	59.48

* for ConvLSTM we used the architecture with the best result after the hyperparameter optimization.

During training, we noticed that the time to complete a run was considerably lower when dealing with the transformer. We decided to calculate the number of parameters and MACs, to better evaluate the number of operations needed. The results are reported in table 6. We notice that although the accuracy is slightly in favor of the ConvLSTM architecture, with the transformer we have an increment of the 24.93% of the number of GMACs. This shows that the transformer could be more suitable for lighter models.

6. Conclusion

In this paper, we analyzed various first-person action recognition architectures and proposed our variation. Taking inspiration from the advances in Natural Language Processing, we adopted a transformer to encode the relationship among frames of a video. While the accuracy was in line with the architecture containing the ConvLSTM, the number of GMACs was considerably lower. We are confident that a more suitable self-supervised task could bring better results, for instance, through contrastive representation learning. Also, it should be acknowledged that the dataset GTEA61 is limited in size, this leads to overfitting and not enough variability, so in the future it would be interesting to apply the architecture on wider datasets. Moreover, with GTEA61 we were not able to fully exploit the ability of the transformer to capture long term dependencies with respect to the ConvLSTM.

Moreover, we would like to point out that, using ViT [1] as a spatial network for features extraction together with the transformer, could lead to even more interesting results. Unfortunately, due to our limited time and resources, we couldn't explore all these possibilities, but we would suggest those just mentioned approaches as future works, that could significantly improve the results we were able to obtain. All in all, we believe that the transformer is the way forward for this kind of task.

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[2] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In Josef Bigun and Tomas Gustavsson, editors, *Image Analysis*, pages 363–370, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[3] Alexander Kozlov, Vadim Andronov, and Yana Gritsenko. Lightweight network architecture for real-time action recognition, 2019.

[4] Daniel Neimark, Omri Bar, Maya Zohar, and Dotan Asselmann. Video transformer network, 2021.

[5] Mirco Planamente, Andrea Bottino, and Barbara Caputo. Self-supervised joint encoding of motion and appearance for first person action recognition, 2020.

[6] Swathikiran Sudhakaran and Oswald Lanz. Attention is all we need: Nailing down object-centric attention for egocentric activity recognition, 2018.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[8] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *2013 IEEE International Conference on Computer Vision*, pages 3551–3558, 2013.