First Person Action Recognition Project Report

Eduard Ciprian Ilas
Politenico di Torino

Carlo Peluso
Politenico di Torino

Maurizio Vassallo
Politenico di Torino

Abstract

The aim of the project was to implement a trainable deep neural network model for egocentric activity recognition.
Firstly we tried to re-implement the EgoRNN model and reproduce the original experiments, involving a multiple stage training. Such model adopts a two-stream architecture, relying on spatial attention mechanisms that stimulate the network to attend to regions containing important objects related to the activity under consideration, and on optical flow in order to extract motion features.
In order to better capture spatio-temporal correlations between the two streams, we then transformed the model in a multiple task single-stream neural network, forcing it to learn joint features between the two domains thanks to a self-supervised auxiliary motion segmentation block.
We then further enhance the network with an additional self-supervised pretext task that drives the network to better learn temporal correlations between frames. Multiple experiments are carried out in order to assess the effectiveness of the adopted techniques.

1. Approaches

1.1. Two-Stream architecture: EgoRNN

Two-stream architectures usually address the issue by combining two sources of information: the visual appearance of the target objective, sourced from the sequence of RGB frames sampled from the video, and the motion information, sourced from a sequence of adjacent optical flow frames. The EgoRNN ([3]) model does the same, and tries to combine the two sources of information by joining 2 different models, each trained on the two domains. The model uses a ResNet-34 pre-trained on ImageNet as the backbone architecture.
First of all, RGB frames are sampled in an uniformly spaced way and then forwarded to the network. For each frame, a class activation map is computed using the winning class for each frame, and then multiplied with the output of the final convolutional layer of the backbone. The purpose of this Class Activation Map (CAM) is to help the network to focus on the relevant regions of each frame, consisting of the objects handled by the observer. Then, the resulting features are forwarded to the next block in order to perform temporal encoding of the frame features. For this reason, a convolutional long short-term memory (ConvLSTM) module is employed: LSTMs are a special kind of Recurrent Neural Network (RNN), capable of learning long-term dependencies by managing the information through its gates. Four gates, involving nonlinearities such as the sigmoid or the tanh, decide how much information should be learned, forgot, be kept hidden or exposed at each step. Since convolutions are established as very good at extracting good quality high level spatial features and LSTM are good at learning temporal relationships, by replacing the internal matrix multiplications of a LSTM model with convolution operations over 3D tensors it's possible to leverage both of these tools in a combined approach.

1.2. Motion segmentation

The two-stream architecture presents two main disadvantages:

- the visual appearance and the motion feature are learned separately and the final prediction is a join of the two streams. This lack of correlation between the spatial and temporal information processed limits the network capabilities of extracting complex motion dynamics in long video segments.
- these approaches usually require complicated architectures that should be trained in multiple stages.

A proposed alternative that aims at solving this problems consists in enhancing the main convolutional backbone with an auxiliary self-supervised task that forces the model to learn information from both spatial and temporal domains in a jointly fashion, modeling the motion information through a motion segmentation (MS) task [2]. This task allows the model to focus on a crucial information for the purpose of action recognition: object movements.

The MS task can be described as a self-supervised label-

ing problem whose purpose is to minimize the differences between a binary map labeling pixels as either moving or static and the object movement predicted by the network when observing a RGB frame.
The MS task was then integrated in the EgoRNN training process, sharing its backbone including the CAM feature and the ConvLSTM-based classifier. CAM-embedding features from the convolutional backbone are fed trough a dedicated MS convolutional block whose purpose is to reduce the number of activation maps from 512 to 100, and then passed through a fully connected layer to further adjust the number of parameters.
At this point, the problem can be designed as both a classification and regression task (Figure 1). In the first case, the provided ground truth motion maps are scaled down to 7×7, converted to a binary map encoding static and moving pixels and then confronted with the features from the MS FC layer using a pixel-wise cross-entropy loss. In the second case, rather than converting each downscaled motion frame to a binary map, they are instead considered as a 7^2-dimensional value where each of the 7^2 pixels' shade of grey is interpreted as a real number between 0 and 1. A mean squared error loss shall be employed in this second case.

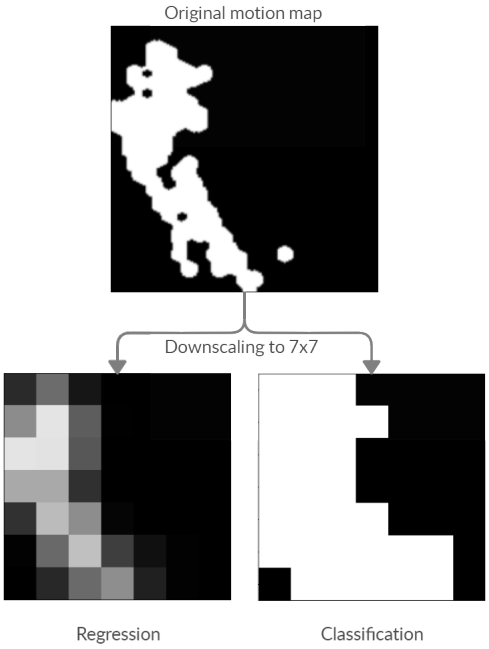

Figure 1. The motion segmentation task can be designed as both a classification and regression problem.

1.3. Frame temporal ordering

When dealing with videos, sequential data provides an important source of information. As mentioned in [2], a promising way to perform better in the egocentric action recognition task is the identification of the correct temporal order of a sequence of images (to learn better temporal correlations between frames). Learning from the observation of sequential data is a natural and implicit process for humans since it informs both low level cognitive tasks and high level abilities, like decision making and problem solving [1]. Furthermore, given the advantage that self-supervised pretext tasks have of not requiring additional annotated data, we decided to enhance the model with an additional self-supervised frame temporal-ordering task, designed to enforce a finer learning of temporal correlations between frames.
The goal of the task is to figure out, given a sequence of frames, whether these are in the correct temporal order or not, and eventually learn to predict the relative correct order. This encourages the model to reason about the motion and the appearance of the objects, and learn to verify whether a sequence of frames are temporally valid with respect to a certain action, and to interpret the temporal proximity of frames in a meaningful way in order to support the main task of action recognition.
A critical challenge when training a network on the ordering task is posed by the way frames are sampled for ordering. Training on every possible permutation is computationally very expensive, and simply sampling the same tuples of frames uniformly from a video might quickly lead to overfitting or poor training given the low amount of instances available. So among all the possible k-permutations of n frames, only some of them are chosen for the ordering task at each iteration. In particular, of all the possible k-tuples ($\frac{n!}{(n-k)!}$), $2m$ are chosen randomly and then using a modified version of the *Hamming distance*, the m most different tuples are chosen for the ordering task. This assures data diversification during the training and avoids making the task trivial. To increase efficiency, the permutations choice is made on the tensors coming out from the ResNet backbone. Regarding the criteria used to asses the diversity of the tuples, a score is calculated for each pair of tuples such that if the element in the position i is different in the two tuples or if an element i of the first tuple is not present in the second tuple the score is increased by 1. Given the final scores the best tuples are chosen.
Having chosen the best tuples for the task at each iteration, their CAM-embedded features are forwarded to a second ConvLSTM module that, in a many-to-many approach, makes a prediction for each frame regarding its relative order in the sequence. Multiple architecture configurations are tried out in order to find the best performing one which maintains the spatial average pooling layer after the Con-

vLSTM module in order to exploit the CAM and a fully connected layer that is trained to make the final prediction.

2. Experiments

2.1. Dataset

The dataset employed throughout the experiments is the provided GTEA61 dataset containing over 400 egocentric videos across 61 activities, each performed by 4 different subjects. The warp flow images provided are computed by subtracting the camera motion from the optical flow, which is a recurrent aspect in ego-centric videos and often can degrade the performance in action recognition tasks. The provided ground truth motion segmentation maps are instead computed using a Improved Dense Trajectory method, which also tries to compensate for the effect of camera motion.
All the results are obtained on the fixed split that designates subject S2 for evaluation and the other subjects for training.

2.2. EgoRNN Experiments

The first set of experiments are meant to reproduce the multiple stage training of the EgoRNN model and to better grasp the function of each of the modules and techniques employed.
In the first stage, at each time step, an input RGB frame is forwarded to the network. The spatial attention layer is computed and embedded together with the convolutional features and then forwarded to the convLSTM module. After all the sampled frames of a video are passed to the module, its memory state is passed trough an average pooling layer and then fed to a fully connected layer in order to obtain the class scores. Only the ConvLSTM and FC layer are trained in the first stage. The last convolutional layer of the ResNet is instead trained in the second stage, along with the ConvLSTM and FC layers.
Regarding the hyperparameters, we stuck to the original ones which trains the first stage network for 200 epochs with an initial learning rate of 10^{-3} decayed by a factor of 0.1 after 25, 75 and 150 epochs. During the second stage, the network is trained for 150 epochs, with a learning rate of 10^{-4}, decayed after 25 and 75 epochs by a factor of 0.1. Both stages used ADAM as optimization algorithm and a batch size of 32.
As usual with two-streams architectures, another network is dedicated to encoding motion changes during the videos. To this purpose, a ResNet-34 pre-trained on ImageNet is trained on optical flow images for 750 epochs, with an initial learning rate of 10^{-2} decayed by a factor of 0.5 after 150, 300 and 500 epochs. The two networks are merged in a final fine-tuning stage where the outputs of the two networks are concatenated and then fed trough an additional fully connected layer to obtained the final scores. The results of our experiments are visible in Table 1. All scores are computed as an average of 3 runs. Our scores closely resemble the one from the original EgoRNN paper. The table shows in particular how there is a considerable improvement in using 16 frames for the RGB training rather than 7 frames. The accuracy gap in jumping from 16 to 25 frames is lower, and furthermore the training time is considerably increased. Moreover, the spatial attention layer does indeed help at improving the performance.

Configurations	**7f** (%)	**16f** (%)	**25f** (%)
ConvLSTM	54.31	59.12	59.92
ConvLSTM + CAM	57.18	62.93	65.52
Temporal warp flow		44.60	
Two stream (joint train)	64.66	73.14	74.41

Table 1. Accuracy results from the first set of experiments with the EgoRNN architecture, obtain by training with 7, 16 and 25 RGB frames per video. Note that the temporal warp flow was instead always trained using 5 stacked warp flow frames, as per the original process.

2.3. EgoRNN + MS Task Experiments

As described in section 2.2, by adding the MS task to the network we can hopefully better encode the temporal motion between frames, which can spare us multiple stages of training involving the optical flow frames. We integrated the motion segmentation task on the model during the second stage training process and kept the spatial attention layer in order to exploit both of them. The resulting loss function was the sum of the previously mentioned pixel-wise loss used for the MS task and the classification one. This way both the classification related parameters and the MS related ones could be jointly optimized at each time step. We then launched multiple hyperparameter optimizations, whose object was to find the optimal hyperparameters for the learning rates (both for the main classification task and the auxiliary one), batch size, decay steps and decay factor, number of epochs and weight decay. Optimal values found were a learning rate of 10^{-4} for the classification parameters and 10^{-5} for the motion segmentation ones, a batch size of 8, 170 training epochs decayed by a factor of 0.1 after 100 and 150 epochs and a weight decay of 5×10^{-4}. Table 2 shows the result of our experiments. The same parameters were employed for both the classification and regression implementations. Although in the first case we were able to obtain slightly better results than the original EgoRNN model, the regression task is clearly lagging behind. Figure 3 shows a visual representation of the class activation map in action. The few samples shown would suggest that the MS task does indeed help at sharpening the network's focus on the important parts of the video frame.

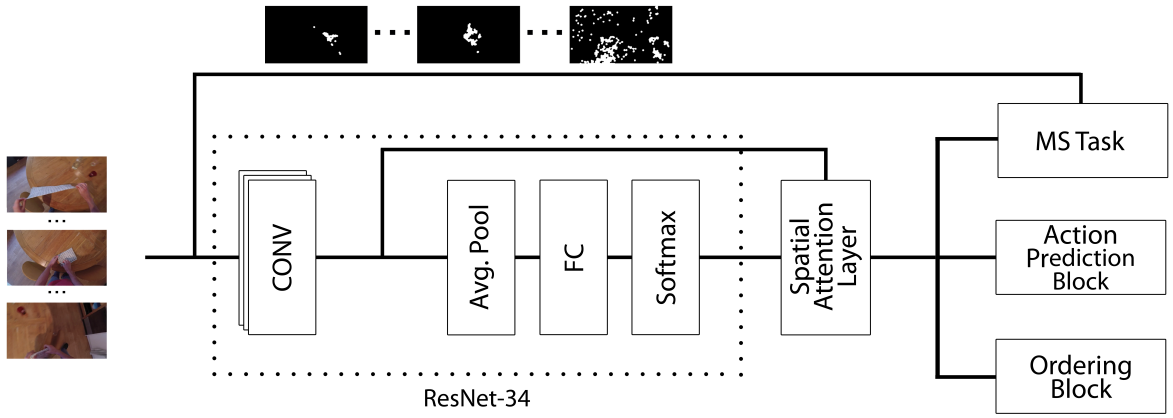

Figure 2. The final model which preserves the Spatial Attention Layer and the MS module, and adds a dedicated branch that performs the ordering detection task as described in section 2.3

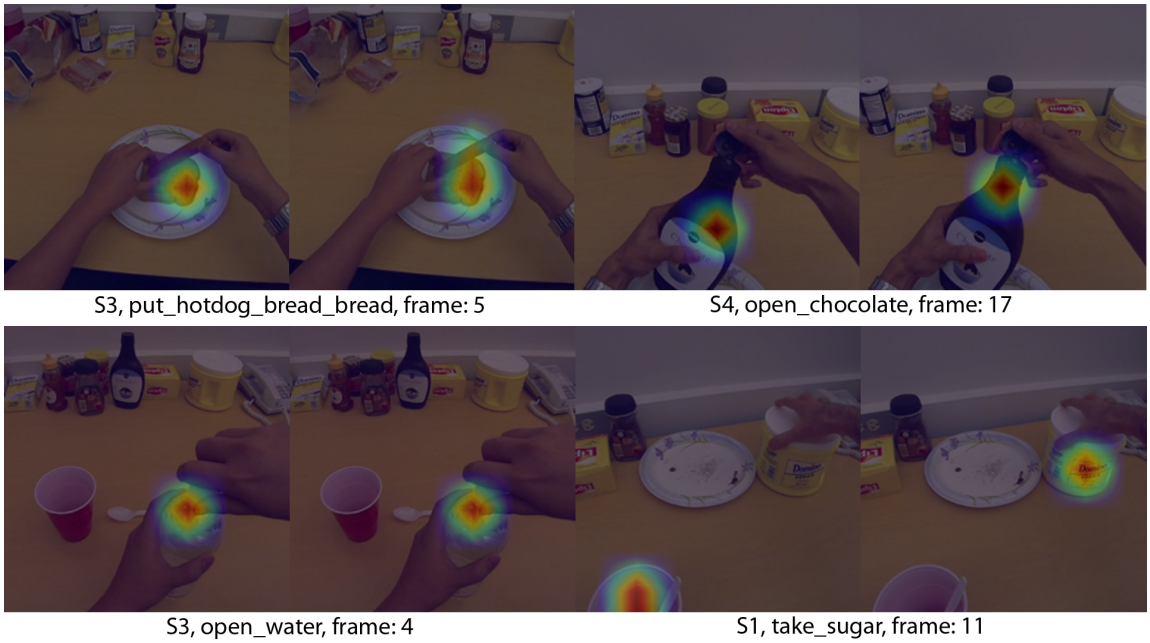

Figure 3. Differences between the class activation maps of the model trained with two joint stream (left image) and the model trained with ms (right image).

Configurations	**7f** (%)	**16f** (%)	**25f** (%)
Classification MS	69.04	75.01	76.91
Regression MS	64.14	67.90	69.92

Table 2. Accuracy results from the integration of the MS task on the EgoRNN model.

2.4. Temporal Ordering of frame integration

The experiments performed so far would suggest that the network is handling the appearance stream pretty well. To further enforce the encoding of temporal relationships during training an additional self-supervised task was added as described in section 2.3. Temporal frame ordering is usually approached in literature by using multiple Siamese CNNs,

each taking one frame as input and then joining their last convolutional layer features with a fully connected layer in order to make the final temporal prediction. Given the performance the ConvLSTM has proven so far, we would expect it to perform well in a similar task as well, and to possible improve the overall performance when integrating it together with the other self-supervised task. An additional hyperparameter optimization phase would establish as optimal a learning rate of 10^{-5} for the temporal ordering parameters and leave the other hypermarameters unchanged. Regarding the ordering hyperparameters, selecting 5 tuples of size 3 from each video was found to be optimal. Table 3 shows that the contribution of the order prediction task was negligible when integrating it with the other self-supervised task, but still relevant when alone. This might suggest that the architecture is close to its learning capacity and further improvements might require deeper adjustments to the model.

Configurations	**7f** (%)	**16f** (%)	**25f** (%)
Ordering + MS	68.55	75.69	75.73
Ordering	68.06	73.51	73.90

Table 3. Accuracy results from the integration of the ordering prediction task.

2.5. Entropy based frame retrieval

We also decided to explore a smarter way to sample RGB frames to use during training: instead of constantly sample the same frames from every video, we exploit an entropy-based frame retrieval technique. Firstly, we computed (offline) for each training video the entropy by means of the following formula:

$$H = -\sum_{k=0}^{K} p_k \log_2(p_k) \tag{1}$$

where K is the number of gray levels and
p_k is the probability associated with gray level k.

Then, in order to shuffle the frames retrieved for the training, we extracted a random value γ between 0 and 1. The sampling method used was

$$\text{Frames} = \begin{cases} \text{N highest entropy frames} & 0 \leq \gamma < 0.33 \\ \text{Uniform sampling} & 0.33 \leq \gamma < 0.66 \\ \text{N lowest entropy frames} & 0.66 \leq \gamma \leq 1 \end{cases} \tag{2}$$

Where N is the number of frames should be retrieved in a specific training process

3. Conclusions

We were able to implement the models and methodologies described in the source papers, and to successfully reproduce the experiments and obtain comparable results.
We experimented with different convolutional and recurrent modules and became familiar with the importance of spatial attention mechanisms. In addition, by being able to pass from a multiple-stage two-stream training process to a simpler and quicker one we also became familiar with self-supervised approaches and their potential in computer vision related tasks, as well as the importance of learning both spatial and temporal features in a combined approach when dealing with action recognition tasks.

References

[1] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: Unsupervised learning using temporal order verification. In *ECCV*, 2016.

[2] M. Planamente, A. Bottino, and B. Caputo. Joint encoding of appearance and motion features with self-supervision for first person action recognition. *ArXiv*, abs/2002.03982, 2020.

[3] S. Sudhakaran and O. Lanz. Attention is all we need: Nailing down object-centric attention for egocentric activity recognition. *CoRR*, abs/1807.11794, 2018.