# Spatial attention motion segmentation order prediction
## A fine-grained action recognition network

Michele Apicella
Politecnico di Torino
s273657@studenti.polito.it

Giulio Alfarano
Politecnico di Torino
s267589@studenti.polito.it

Gabriele Cuni
Politecnico di Torino
s277957@studenti.polito.it

## Abstract

*The purpose of this paper is to tackle the video recognition task, in terms of first person egocentric action recognition, a field which has become more popular in recent times, after reaching good results in the third person action recognition field. In particular, the core idea of this work is the building of a Recurrent Neural Network with the jointly implementation of two different Self-Supervised Tasks. This is not done from scratch, but the classification experiments are replicated on the base of two main works in which the neural networks proposed have reached brilliant results : the Ego-RNN [1] and the Spar-Net [2]. In the first it is explained how to do a spatio-temporal video encoding into a convLSTM, working on highly specialized spatial attention maps generated for each frame using class-specific activations from a CNN pre-trained for generic image recognition. In the second a different, single-stream network architecture is proposed, focusing on the implementation a self-supervised block that uses a motion segmentation (MS) pretext task to handle the information capturing of frames motion and appearance to make a better the spatio-temporal video enconding into the convLSTM, improving the fine-grained first person actions classification task. Basing on these high accuracy level different experiments, after adding the MS task block to a single-stream Ego-RNN and implementing this self-supervised task as a regression problem, a personal own variation of this approach is proposed, consisting in the jointly adding of a shuffled frames sequences prediction order self-supervised task (PO) [3].*

## 1. Introduction

Action video recognition is a central topic in computer vision. While encouraging results are reached in the third person action recognition field, the first person branch has been explored relatively in recent times and it is open to big improvements. The reason is that mobile phones and wearable devices endowed with cameras have become more and more popular and affordable, focusing the attention computer vision scientific community to this topic. The idea of this classification task is to detect fine-grained first person actions instead of simple egocentric activities: this is more challenging because actions are more complex and there are much more factors to be considered. This video ego-perspective brings along several problems that complicate the classification task. First, the ego motion caused by the camera wearer is a huge difficulty to tackle; in this sense warp optical flow data processing [1] and the self-supervised motion segmentation [2] attempt to resolve this problem. Another hitch is how and so, where, to focus the spatial attention [1]: hands motion patterns, objects locations and manipulation are all very important factors the networks needs to menage and handle in order to carry out the classification task. The main analogy between the two approaches, on which the experiments of this paper are based, is the use of a convLSTM for the spatio-temporal encoding of the video frames in order to eventually select the inner memory state to conduct the classification, but they differ in many other aspects (this point will be discussed in Section 2). Different kinds of data are processed for different purposes (Section 2) : RGB frames [1][2], warp optical flows generated from adjacent frames [1] and MMAPS [2], which are images that show of pixel-per-pixel frame motion capturing. In the following sections are described into details the Ego-RNN [1] and the Spar-Net[1] architectures, because several experiments of these two models are replicated in order to show the model proposed which is a single-stream architecture, trained in two stages, with self-supervised implementations. Then some Class Activations Maps (CAM) [1] are showed in order to have a graphic projection of what is the purpose of the spatial attention focusing, once when the self-supervised block [2] is added. After that, the MS Task [2] is implemented as a regression problem and a proper tuning optimization of the two self-supervised methods is carried out. Eventually, after several experiments and different implementations an own personal variation of the self-supervised task is proposed and results are reported and discussed. On the base of the core idea be-

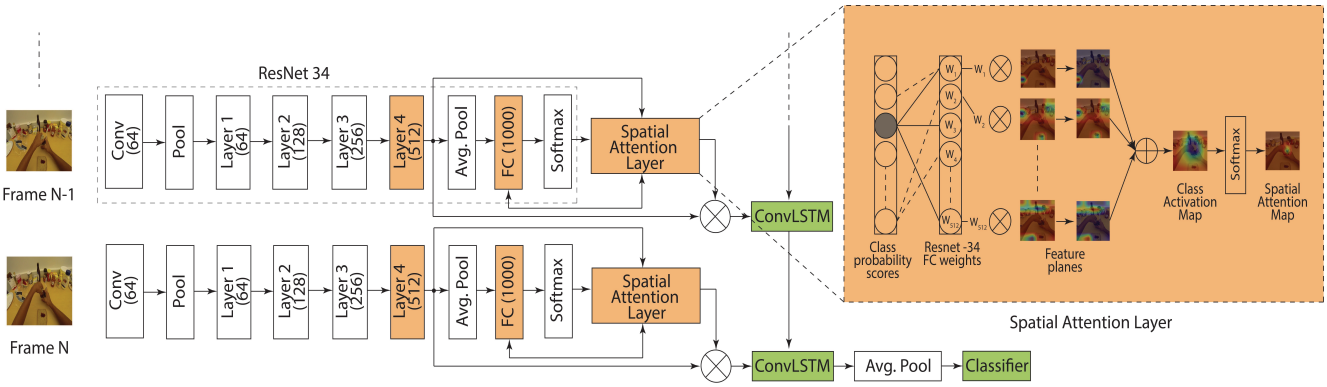


Figure 1. Ego rnn architecture

hind the Order Prediction Network (OPN) [3], we decide to melt together the three different approaches adding the MS [2] and the OP [3] tasks to a single-stream Ego-RNN focusing on spatial attention: we call the resulting architecture SAMSOP-Net, which stands of Spatial Attention Motion Segmentation Order Prediction Network.

## 2. Related works

First person action recognition requires great efforts in terms of methods used to extract knowledge from video frames and in terms of understanding which special kinds of knowledge are really useful to conduct properly the classification task. The Ego-RNN, the Spar-Net and the OPN address this issue in different ways. While other studies concentrate on gaze location and hand segmentation, which may lead to very complex and heavy operations, in [1], using a Res-Net34 network as backbone architecture, trained in two different stages, a great focus is made on spatial attention, especially in appearance, trying to find relevant regions of the image where there are the most important objects that characterize the activity of the video under consideration. On this purpose, class activation maps (CAM) are used for spatially selective feature extraction, which develop their own representation classes implicitly, while training the video-level classification, making the network, which, in this sense, is wisely pre-trained on imagenet dataset, able to identify the discriminant regions of the frame which can be used to recognize the activity class. The CAM is converted in a spatial attention map and it is multiplied with the output of the final convolutional layer of the Res-Net34: the resulting 3D tensor in encoded to a convLSTM to use eventually the memory state to complete the action recognition. The other great issue of this kind of task is how to find a good solution to detect and to handle motion over time. The approach proposed in [1] is a two-stream architecture in which the second one processes warp flow data of adjacent frames stacked together. The information about appearance and motion, which are learned separately, are eventually merged with a weighted sum for ultimate the classification. The approach suggested in [2] is different, though analogies can be encountered by means of convLSTM for temporal encoding and of the Res-Net34 pretrained on imagenet dataset as backbone. The Spar-Net, differently from classical two-streams architectures who learn appearance and motion features separately, addresses this issue extracting jointly knowledge of appearance, through an action recognition block consisting in a series convolutional layers, whose output goes through a standard convLSTM, and of temporal motion through an implemented MS task. This added branch looks for generating a map in which pixels are either labeled ”static” or ”moving”, in a completely unsupervised environment, giving a good overview of the motion trend in the input video considered. The estimated motion map, which is function of both MS head parameters and image embedding because the MS head receives the output of the final convolutional layer of the appearance block, is compared to the ground-truth map generated by the Improved Dense Trajectory method, which tries to minimize the camera motion effect. With this implementation, the use of warp flow data for motion tracking, and so the adoption of a two-stream architecture, becomes useless. Differently to the two approaches discussed above, which have analogies in terms of convLSTM presence but a lot of differences in terms of algorithm choices, architecture and implementation ideas, the OPN [3] is not a recurrent neural network and does not use a convLSTM. It uses as inputs shuffled frames sequences taken from a video, and implements a self-supervised task with the purpose of predicting the sequence order, learning features about frames which provide temporal coherence knowledge. It does not consist of a binary classifier telling if the sequence is temporally correct or incorrect, but the model tries to predict which permutation has been made. Sequence sorting becomes a surrogate task for training a CNN: successfully solving the sequence sorting makes learning visual representation to recover the temporal coherence of video frames, under-

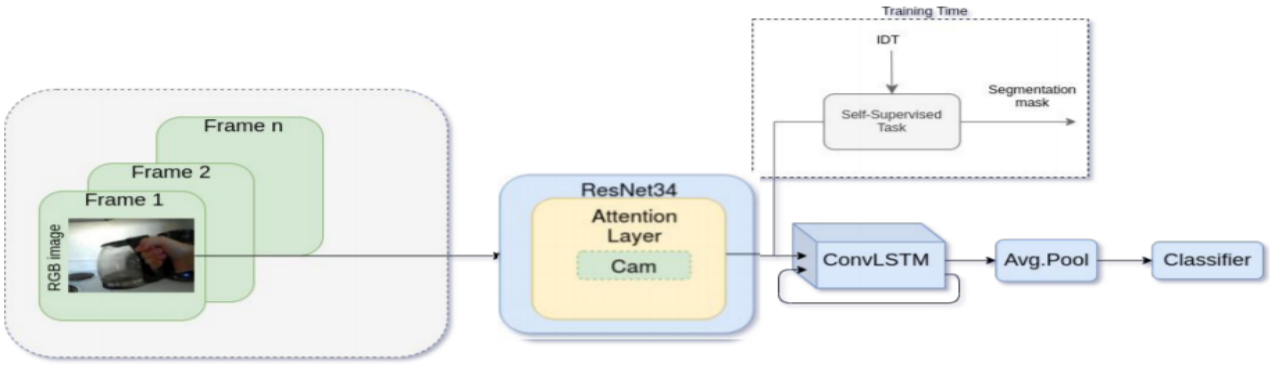


Figure 2. Network structure with the MS task implementation

standing temporal structure of image sequences. In order to do this, a small number of frames are extracted from the original video, for example 4. These frames are shuffled, but only 4!/2 permutations are used because, conceptually similar to the horizontal flipping technique, backward and forward permutations are grouped into same class, because such actions are coherent in both temporal directions (for example close/open ketchup). Great attention is focused on the train data sampling phase. In particular three techniques are carried-out together in [3] : motion aware tuple selection, which uses the magnitude of optical flow data to select patches of frames with larger motion regions, spatial jittering and channel splitting in order to learn higher-level features. Instead of extracting knowledge from a discrete number of frames like in [1], the number of frames selected for the permutation is lower, and after the merged train data sampling techniques, features are extracted. The core idea of the approach proposed is to use features extracted from tuples of all possible pairs of frames of the selected permuted sample used as input and to fuse this information to gain useful temporal features and order knowledge making the classifier able to discriminate which permutation has been made on the sample. Looking with attention at these three different approaches and the clever different ideas behind the models used to carry out a fine grained action recognition, it is proposed a sort of merging of these three solutions, whose architecture and implementations details are discussed in the following sections.

## 3. Architecture overview

In this section we describe how our proposed model is built, in terms of analogies across networks on which this neural net and the experiments made are based on. There are three sub-sections focusing respectively to three main components of the network proposed highlighting the three tasks merged to carry out the fine-grained first persons action recognition. The first, which is the core of the model, is basically a single stream Ego-RNN [1] focusing on spatial attention, extracting useful features from RGB frames. The second and the third are two different, parallel branches that carry out self-supervised task: motion segmentation from Spar-net [2] and order sequence prediction from OPN [3]. The whole process follows two different training stages. Below there are the details.

**Spatial attention: Ego-RNN.** This section of the neural network is the network used in paper [1], which uses the Resnet34 convolutional network to extract features from RGB frames, class activation maps (CAM) are used for spatially selective feature extraction and a convLSTM network, which is a variant of long short term memory with convolutional gates instead of fully-connected gates, for encoding spatio-temporal changes. The network is used for egocentric activity recognition which is capable of a deep feature encoding with spatial attention. The system is based on the use of CAMs: they allow to identify the image regions that the more important to carry out frames classification. Focusing the attention of the network on the area selected by the CAM is important because it allows to identify the image region in which the object being handled is located, this technique improves the classification and recognition of activity. The block diagram of the proposed idea by Sudhakaran and Lanz [1] for encoding the RGB frames is shown in Figure 2. The Resnet34 core network is pretrained with the weights obtained from the well known Imagenet dataset. RGB frames are inserted into the forward of the network and analyzed as normal images from the convolutional layers of the model up to layer 4. Now the CAMs are calculated starting from the product of the activation value in the final convolutional layer for the weights corresponding to the classified class returned by the fully connected layer of Resnet. The CAMs thus obtained is then converted to a probability map by applying the softmax operation along the spatial dimensions. These spatial attention maps thus obtained are now multiplied by the Hadamard product with the same weights of the last convolutional layer of the resnet. At the exit of the spatial at-

tention layer the image features with spatial attention are obtained, then the temporal encoding action carried out by the module called convLSTM is incorporated. The convLSTM module designed by Sudhakaran and Lanz [1] works in a similar way to a traditional LSTM, but differently from this it receives 3D tensor input to adapt to the spatial attention features generated by the convolutional network together with the spatial attention layer. This method they developed allows to combine the information from temporal and spatial variations at the same time. Once all the RGB frames have been submitted to the network, the features that describe the entire video are obtained by taking them from the internal memory of the module described above. Finally the features are sent to a pooling layer and a fully-connected layer for classification. This network conceived by [1] is trained in two stages: the first step is to train only the convLSTM module and the final classifier, in the second step in addition to the modules just mentioned, the last convolutional layer, the Resnet fully connected and the spatial attention module are also trained.

**Motion segmentation: Sparnet.** This part of the network architecture is similar to the shallow head used by the [2] paper network. Considering the Spar-Net, we implemented the last part of the shallow head elaborating the features and the motion maps, since we take these from the provided GTEA-61 dataset. Basically, it is a motion segmentation task (MS), which is a formalization of a self-supervised task, which aim to embed into the network the temporal clues of the video without using optical flow images, helping the appearance stream learning motion features. In this case the pseudo labels are Improved Dense Trajectories images (IDT), obtained by an elaboration of the images that defines the movement of a certain subject in the image, but avoiding the camera motion, usually always present in ego recorded videos. This elaboration allows us to leverage a binary map which codify every pixel as moving or static with a value included in an interval from 0 to 1. In our experiments we implemented two versions of the same MS Task: the first is implemented as a classification problem, the second one as a regression task. The main differences are obviously in the elaboration of the binary map and on the loss computing. For what concerns the first implementation: the spatial transformation, applied to the input motion maps images, includes a function called Binary, which take an image as input and it approximates every pixel value to 1, if it is greater than a given threshold (in this case 0.4), or to 0 if it is lower. This elaboration is fundamental since this prepare the motion maps to be processed for a classification task. For what concern the structure, the branch is connected at the end of the spatial attention layer, getting the CAM features as inputs. From an architectural point of view, the shallow head that performs the motion segmentation is similar to the paper [2] one: it is composed by a

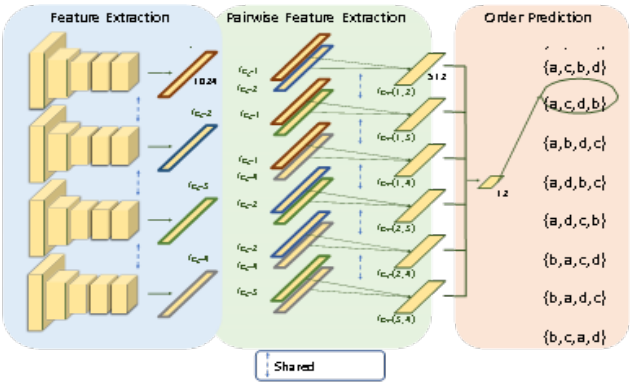


Figure 3. Architecture of Order classifier branch

convolutional layer with 100 outputs and a kernel size set to 1, and a fully connected layer that performs the feature extraction in order to have 2 outputs for every pixel, one for the motion state and the other one for the static state. The results are passed to a Softmax layer. These represents the state of a certain pixel, they will be compared to the ground truth motion maps in order to calculate the loss. Since it is a classification task, the used function is a cross entropy loss, which performs a multiplication between the ground truth of a pixel with the log of the outcome of a pixel.

$$\mathscr{L}_{ms}(x,m) = -\sum_{i=1}^{n}\sum_{k=1}^{N}\sum_{j=1}^{s^2} m_i^k(j)\cdot log(l_i^k(j))$$

For implementing the problem as a regression task, we had to perform a slightly change in the model. In particular, we replaced the fully connected layer with $2 \cdot s^2$ outputs (where s stands for the size of the image) with a similar layer but with an outcome of only $s^2$ outputs: this is because for every pixel there is only one neuron that should give an approximate value of motion in the interval [0,1]. Following the same reasons, the motion maps are elaborated without the Binary function, but they are passed directly to the loss in order to perform the cost. Since we must tackle a regression problem, also the loss function is changed, replaced with an MSE loss function, based on the L2 equation.

$$\mathscr{L}_{ms}(x,m) = \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{N}\sum_{j=1}^{s^2}(m_i^k(j) - l_i^k(j))^2$$

In the two different implementations the chosen loss is summed to the loss related to the ego rnn backbone and the result is the subject to minimize.

**Order prediction: OPN.** For this part of the network we implemented a side of the Order Prediction Network proposed in [3]. In the paper it is described a complex and useful preprocessing of the images which we could not implement because we would not have any connection to the

backbone of our net, so there would not be any way to embed the learning. Our hypothesis is that the task of ordering a sequence of frames can be useful for embed the learning of more general temporal clues, since the MS Task is more focused on the motion part of the video. For this reason, we decided to implement only the last part of the network: we used our ResNet34 as feature extractor, these features are then passed to the implemented shallow head. The inputs are so four randomly shuffled frames sampled from a video, like in [3], since it is the number of frames that gives the best outcomes; the branch must solve the sorting problem as a multiclass classification. Since we used 4 frames, we expect to have 4! = 24 possible permutation, but we should consider that the actions are coherent forward and backward, so the specular classes can be grouped, obtaining 24/2 = 12 classes. As we said, our backbone is used as feature extractor, so the branch is connected at the end of the ResNet34; after the extraction of the features, the results are passed to two concatenated fully connected layers: in the first one (fc6) all the frame features are elaborated, in order to reduce the dimensions; then, through a list of combination, all the pairwise of features are elaborated by the second layer (fc7) and the results are passed to the final classifier which gives the most probable outcome. Since it is a classification problem, to calculate the loss we used the cross entropy formulation.

## 4. Experiments

In this section first are described the dataset used to carry out this task and the particular kinds of data. Then, implementation details of the different architecture combination experiments, also in terms of training parameters choices and optimization, and the results obtained are discussed and compared. First, some experiments of both single and dual stream architectures proposed in [1] are replicated, with different parameters configurations, then at the chosen single stream model the MS task described in [2] is added. This is done in two different ways: at first, it is replicated as in [2], then it is implemented as regression task as showed in section 3.2. After that, the OP task, differently implemented as in [3], is added to the network. Different combinations of these approaches are discussed in the Ablation study, section 5.

### 4.1. Dataset

The dataset used to carry out this fine-grained first person action recognition task is the GTEA61. It has first person data from four different users. For the sake of simplicity, three of them are used in the training phase and one for testing phase. There are basically four different kinds of data: RGB frames, used for detect spatial attention and for the OP inputs, warp optical flow frames, both in x and y directions, used to handle motion information in the first

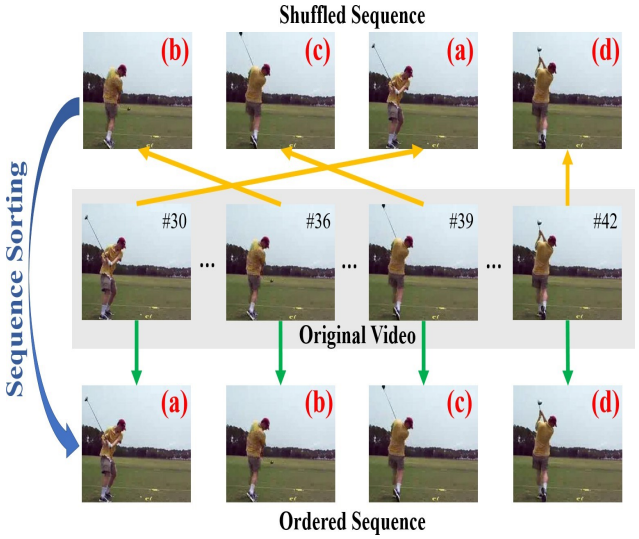


Figure 4. Implementation of Order classifier branch

| Configuration | 7 Frames | 16 Frames |
|---|---|---|
| ConvLSTM | 52.59 | 57.76 |
| ConvLSTM-attention | 56.9 | 64.66 |
| Temporal-warp flow | 42.24 | 42.24 |
| Two stream (joint train) | 64.66 | 68.10 |

Table 1.

architecture configurations, and the motions maps, that are images showing where is detected motion pixel per pixel, used during the MS task phase, as pseudo grand-truth labels. The final model proposed does not make use of warp optical flow data.

### 4.2. Implemetations details

As starting point, some experiments proposed in in [1] replicated, using different parameters. We can subdivide them in 4 different categories: ConvLSTM, ConvLSTM attention, Temporal warp flow and two-stream (joint-train). The analogy between them is that they all use convLSTM of temporal encoding information, but they act differently to extract spatial knowledge. Every neural network performances are carried in two different stages in which different part of the net are pre-trained on Imagenet: in stage 1 only the convLSTM and the final output classifier, in stage 1 in addiction to these two parts also the last convolutional layer (Layer 4), the fully connected later of the ResNet-34, which is the core of the achitecture, and the spatial attention layer which is the part of the network that generates the class activation maps used for focusing on spatial attention. The stage 2 training phase starts loading weights and biases from the best model of the stage 1. The stage 1 is trained for 300 epochs, the stage 2 for 150 epochs. For every model implementation different parameters sets are experimented, in particular the learning rate parameters are

1e-3, 1e-4, 1e-5 and the input frames data that are 7 and 16, in order to discover how aggressive and low learning rates and the number of input frames affect the action recognition performance. The first experiments category, called ConvLSTM, does not make use of Spatial Attention Later, but instead, detect spatial attention simply from the features extracted by the convolutional layers of the ResNet-34: this is done for underlining the great importance of the removed layers, as seen in 5th section (ablation study and results). The data used are just the RBG frames. The second category, ConvLSTM, makes use of the class activation maps generated by the spatial attention layer, to extract greater spatial knowledge in addition to the features extracted by the convolutional layers of ResNet-34. Also in this case only RBG data frames are used as input. The third category, Temporal warp flow, uses as inputs only the warp flow data pre-processed in x and y directions. The spatial attention layer is not activated because the spatial attention requested is linked to the motion information detection, already present in the warp flow input data, and so CAMs are not generated, relegating them only if RGB video frames are used as input. The features extracted to gain information are the result of convolutional layers of the ResNet-34. The last category, called the two-stream (joint-train), jointly encodes the knowledge learned through the spatial attention, focused on the RBG frames with the activation of the spatial attention layer and the generation of the CAMs, and through the motion attention, using the features extracted from the warp flow data. This is the final two-stream architecture proposed in [1]. After these experiments the MS task branch is implemented on the single stream model with spatial attention. There are two different models regarding the MS task: the one suggested in [2], making use of a binary classifier for motion/no motion, and the one formulated as a regression problem. The training phase again consists of the two stages as described before. The activation of both the MS task formulations take place only in the second stage, making the stage 1 only used to gain spatial attention. Again, learning rate parameters tried are 1e-3, 1e-4, 1e-5 and the input RGB frames data selected are 7 and 16. After that, the OP task branch is implemented in different combinations with the spatial attention and MS architecture that will be described in section 5. Regarding the OP task, similarly to the MS task, it is activated only in stage 2 and always 4 frames are selected for the permutation. The experiments conducted with this implementation are carried out with the usual learning rate and input RGB frames sets.

## 5. Ablation study and results

In this section we evaluate the effect of the various design choices on the provided GTEA-61 dataset. As we previously wrote, after a pre training on ImageNet, we first perform a first training phase where the output classifier and the convLSTM are trained, then we perform a second phase of training which covers most of the remaining net. We do not have any specific baseline for our model, but we aim to obtain similar results from the papers we studied (not the same since all the papers uses a very high number of frames as input), or at least to have a slightly accuracy increment through our variation.

7 frames

| **Configurations/LR** | 1e-3 | 1e-4 | 1e-5 |
|---|---|---|---|
| Ego-rnn + MS Task | 7% | 56% | 44% |
| Ego-rnn + regression MS Task | 12% | 51% | 39% |
| Ego-rnn + MS Task + Order Task | 8% | **57%** | 42% |

Table 2.

16 frames

| **Configurations/LR** | 1e-3 | 1e-4 | 1e-5 |
|---|---|---|---|
| Ego-rnn + MS Task | 11% | 65% | 54% |
| Ego-rnn + regression MS Task | 10% | 62% | 49% |
| Ego-rnn + MS Task + Order Task | 10% | **66%** | 53% |

Table 3.

**Hyperparameter optimization.** We grouped all our results and accuracies in Table 2 and Table 3, where we show how the tasks we implemented influence the validation accuracies for different values of hyperparameters. Our analysis is focused principally on Learning Rate value and on the proposed number of frames. We clearly see that the default LR gives the best results, since weaker or higher values make the accuracy decrease. On the other hand, the number of frames is somehow directly proportional to the accuracy, probably because this situation allows the network to elaborate more features, especially for the motion segmentation task and for the order sequence task, which can influence the differences between the 4 frames it extracts.

**Motion segmentation task.** The motivation in inserting the motion segmentation task is to make the network embed a form of motion clues from the images in order to help the appearance side of the network to better recognize the action, and the contribution of the task is partly increasing the accuracy: we have an increment of about the 1% at 16 frames respect to the convLSTM + attention configuration. We know that the influence of the shallow head performing this task is highly influenced by the input features it receives; during our experiments it is connected to the exit of the spatial attention network, but if it is connected to the end of the ResNet34 it could give different results. Despite this, we also tried to visualize the CAMs produced by the appearance part of the Ego-RNN network with the Motion Segmentation branch implemented (Figure 5) and we can notice that they are similar to the ones produced by [1], so the visual outcome is quite positive, even though the increment on the accuracy is minimal. For what concern the MS task in a regression form, it brings the network to a slightly

Figure 5. CAMs visualization

decreasing accuracy of about 3%. This is probably because the input features are more suitable for a classification problem than for a regression one.

**Order sequence task.** About the second task we implemented, we firstly tried to implement the network without the MS Task but the results were quite lower than expected: this is probably because even though it manages to make the network learn some general temporal clues regarding the different frames, it is not enough efficient in the motion clues recognition. So, the next experiment we tried was to connect the Motion Segmentation task to the end of the spatial attention layer and the Order Sequence task at the end of the ResNet34, in order to consider the backbone network as a features extractor. Through this design choice we manage to have an increment of about 1%, which is a quite positive result for our objectives we have set at the beginning.

## 6. Conclusions

In this paper it's showed how the task of ordering an unordered sequence of frames, in combination with other implementations, allows a neural network to learn important features for fine grained first person action recognition. Our work proposes a network capable of ordering the previously disordered RGB frames: this self supervised method is sufficient to obtain an improvement in classification accuracy only if combined with the self supervised task inspired by [2]. Our method lays the foundation on the work [1], which is the core of the model proposed, using a pre-trained network for image recognition and exploits the possibility of combining static and dynamic information, having foundations in high spatial attention. The greatest improvement is achieved through the [2] self supervised model which incorporates motion features, in addition to it our variation learns useful visual representations of the frames. Network training with this method allows us to learn rich and generalized features useful for the action recognition task.

## References


[1] Swathikiran Sudhakaran and Oswald Lanz, *Attention is all we need: Nailing down object-centric attention for egocentric activity recognition object-centric attention*. British Machine Vision Conference, 2018.

[2] Mirco Planamente, Andrea Bottino and Barbara Caputo, *Joint Encoding of Appearance and Motion Features with Self-supervision for First Person Action Recognition*. 2020.

[3] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, Ming-Hsuan Yang, *Unsupervised Representation Learning by Sorting Sequences*, 2017

[4] Code related to this report: https://github.com/GabrieleCuni/MLDL_Group_Project