First person action recognition: different approaches to exploit optical flows

Marina D'Amato Adele de Hoffer Andrea Arcidiacono

April 5, 2021

Abstract

In this work we propose a deep learning model to tackle a first person action recognition task. We start from the contribution of previous works to better understand the challenges and the difficulties of this branch of computer vision. Then we implement a variation of the pre-existing models to experiment a new approach. Despite the powerful success of deep learning and 2D convolutions in image classification, the problem of activity recognition in videos is still open and well researched. Recently, thanks to the development of wearable cameras, there has been a growing interest in egocentric video analysis which is even more challenging. Most of the architectures developed so far consist in two-stream networks that address separately the encoding of appearance and motion information. We use this idea as a starting point and then we develop a single stream architecture with a self-supervised task. Lastly, we propose a variation of the two-stream model in which we exploit the cam block to integrate flow images and RGB images in a single architecture.

1 Introduction

Activity recognition in videos is a very challenging task and it is the subject of many researches that have been developed in the machine learning and computer vision fields. This area has a widely landscape of applications which range from surveillance and security to robotics and human-computer interaction.

The main objective of research focuses its attention on third-person videos while in this task we consider an emerging branch related to first-person action recognition problem. In this setting, all the videos taken into account are filmed by a camera on the head of the user that makes the action. Indeed, recently, the development of wearable cameras has created an increasing interest in this application, which is even more challenging with respect to the traditional setting.

First of all, in addition to the natural movement of the camera, egocentric videos are characterized by the motion of the camera wearer which most of the time is not representative of the action performed. Moreover, we have a strong occlusion of the arms and hands of the subject that could make harder recognizing the objects with which the wearer is making the actions. Another reason of difficulty is represented by the lack of information about the pose of the actor or the working environment since we don't see the subject but just his arms. Lastly, the development of this type of videos is due to the recent diffusion of wearable cameras, thus the amount of data for training is limited.

The core of this task is the ability to correctly identify the objects handled by the subject. For this reason, it is important to find a method that encodes both the appearance of the object obtained through the RGB processing and the motion information. For what concerns the motion, we exploit optical flows which tell us what pixels are moving inside consecutive frames in both coordinates. In order to find solutions for the problem of the camera movements, we use warp flow images which are obtained by subtracting from the optical flow the camera movements.

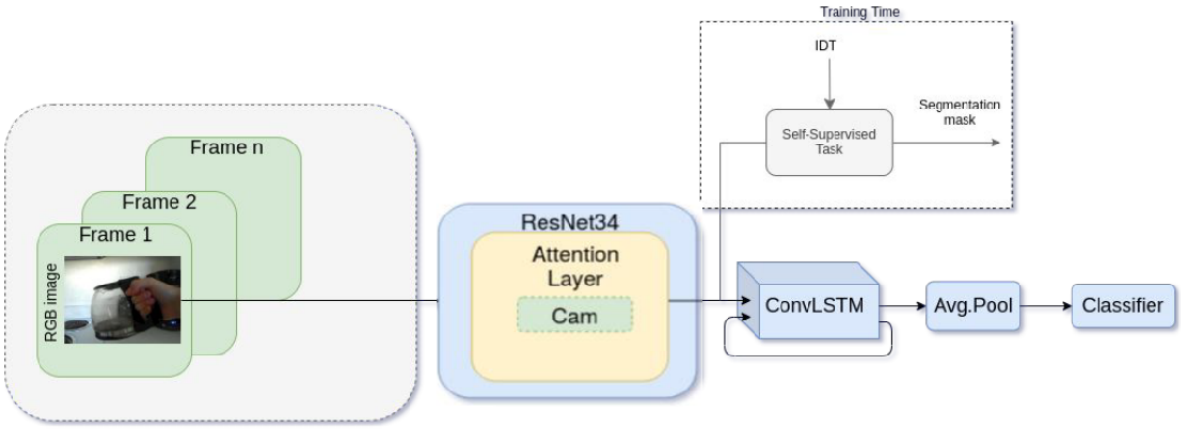


Figure 1: Architecture

The references to previous works on which we based our experiments are proposed in section 2. All the details on the algorithms and the network implementations can be found in section 3, while the experiments and the results are reported in section 4. We end with a brief conclusion and a discussion on future improvements.

The code related to the work presented in this report is available at `https://github.com/marinadamato/egornn`.

2 Related works

2.1 Ego-RNN

The main idea behind the paper [1] is the development of a deep neural network for first person action recognition that makes use of a spatial attention mechanism and a convLSTM block to better encode the spatial and temporal information.

The network proposed is a two-stream architecture. The first stream is composed by a ResNet which exploits the class activation maps mechanism. The CAM helps to improve the action recognition task, since CAM detects the regions of the environment that are mostly correlated with an object. Then the CAM are converted into Spatial Attention Maps by applying the Softmax operation. After the multiplication of these maps with the final convolutional layer of the ResNet, a ConvLSTM block is used to encode the temporal information about the frames. This variation of the standard LSTM has been proposed because it is able to detect both changes in spatial and temporal dimensions.

The train of this first stream is divided into two stages: the first stage is characterized by the training of the classifier and the convLSTM layers while in the second stage also the fourth convolutional layer and the FC layer of the ResNet are trained. For this part 25 frames for each video uniformally sampled in time are utilized.

The second stream of the network is composed by a ResNet-34 pretrained on ImageNet and trained using 5 stacked sequences of the warp optical flow images provided. The two streams are then joined by adding a new fully connected layer on top of the two individual networks. This model is able to achieve a stronger performance due to the fact that is able to identify the object location and so extract spatially selected features.

2.2 Self supervised task

The paper [2] proposes a network which tries to avoid both the two-stage training and the two-stream approach. While in the previous work the appearance and temporal flows were separated, the main idea of the paper [2] is to process simultaneously warp flow and RGB images, in this way also it is able to overcome the lack of correlation between RGB input frames and the sequence of stacked optical flow. As a result, it is obtained a simpler architecture which provides better results with lower computational time.

In particular, a SparNet model is proposed as architecture. It includes a convolutional layers backbone that ends in two branches: one is related to the action classification task, which is composed by a convLSTM followed by a fully connected layer. The other one is the motion segmentation block, which instead is composed by

a convolutional layer followed by a fully connected layer and returns as output a 1x7x7 array which is to be compared with a ground truth binary map. These maps are computed using IDT (Improved Dense Trajectories) techniques witch outputs are combined to determine if a particular pixel is moving or not. For this reason we utilized a pixel-per-pixel cross entropy loss in the motion classification task.

The innovative approach relies on the use of a self-supervised task which enables the network to learn from unlabeled data (since one problem related to ego-motion classification is the limit number of human-labeled data) how to extract useful features because, since the aim is to minimize the sum of the classification loss and the MS loss, in this way the backbone is forced to learn features that focus on object movement.

2.3 Variations

As possible variations for our project we have considered different kind of interesting self-supervised tasks, such as those referred to in [4] and [5].

The first one proposes a multi-task learning problem which aims at optimizing a convolutional neural network for action recognition and at the same time for the prediction of future frames. To do so, it exploits adaptive dynamic motion filters to learn video-specific internal motion representations. In such a way the second task of predicting new frames is used to enrich the main classification problem.

The second paper proposes a different approach to learn spatio-temporal features. In this case, optical flow images have been replaced with more robust ones in order to mitigate the camera movement: instead of warp flows, a solution based on motion boundary is presented. The network follows a traditional two branch architecture composed by the motion and the appearance branches. The self-supervised task is able to predict through partitioning patterns the spatial regions of the frames where is the largest motion and also the dominant orientation of the motion. For what concerns the appearance statistics, it is proposed a statistical method to compute the largest and smallest color diversity locations and the corresponding dominant colors.

Both the works adopt convolutional 3D networks as a backbone of the network. For this reason, we decide not to implement these self-supervised tasks because the instruments used for this work are unsuitable for such computationally demanding tasks.

2.4 Deep depth colorization

As possible variation, we exploit the concept of Deep Depth Colorization proposed in [3] to 'colorize' the warp flow. The idea is to train a 'colorization block' to colorize a greyscale image into a depth RGB map. Successfully the image is used as input for a pretrained CNN for image classification task.

In particular, a single channel black and white image is fed to a residual convolutional architecture that returns a RGB image that maps the depth of the original space into a 0-255 scale. After, the image is fed to a VGG pretrained with ImageNet, completely frozen except from the last fully connected layer. In this way the network will learn the mapping that minimizes the classification loss.

The model shows a great improvement for different datasets, also compared to previous colorization tasks like ColorJet or SurfaceNormals++, besides the colorization proposed emphasizes the borders and the salience features of the object and flatten the background, that is a useful way for improving the action recognition task.

3 Methodological overview

3.1 EGO-RNN

We exploit the architecture already provided adjusting some aspects that did not match with our goals. In particular we pass a flag to the network to activate or deactivate the attention model. To do so we connect directly the output of the ResNet to convLSTM.

3.2 Self supervised task

The paper [2] proposes a variation of the ego-RNN network which tries to avoid the two stage training. In this paper we replicate the experiment to add a self supervised task to ego-RNN architecture. In our implementation we still use a two stage approach and add the self-supervised motion task to the second stage of the training. Self-supervised task's objective is to predict whether a partic-

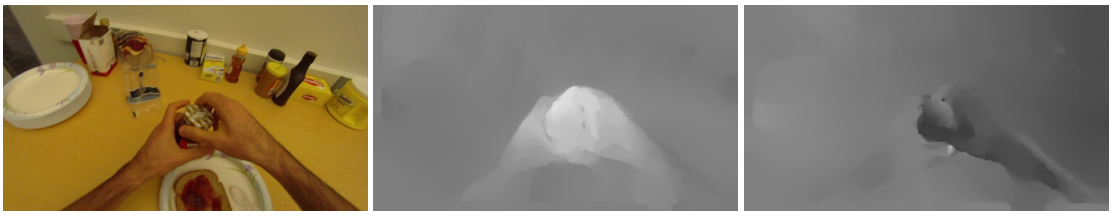


Figure 2: RBG frame and warp flows

ular pixel is moving or not. This can be addressed as a binary classification or a regression problem.

In the first one we obtain a binary output which is then compared with a black and white image computed from warp flow images as ground truth. In the second one, the model predicts how much the pixel in that particular area is moving, and we compare the networks' output with grey scale motion map re-scaled to $[0, 1]$.

While in the previous work the appearance and motion flows were separated, the idea of this paper is to process simultaneously warp flow and RGB images. In this way the final architecture is simpler and the training phase requires a lower computational time.

For what regards the architecture, we first extract N RGB frames for each input video segment whose are taken maximazing the spaces between each other. These images are fed into ego-RNN to process and we take features out from the spatial attention layer, of size $512 \times 7 \times 7$, to use them as input to the motion block. The motion segment consist of a ReLu operation and a convolutional layer with a 1×1 kernel that reduces the size from 512 to 100. The output then passes through a fully connected and softmax layer to classify each pixel of the feature if is moving or not. The result is a $s^2 = 7 \times 7$ map that is compared to the binary or regression ground truth map loaded from the dataset.

To appropriately perform the loss we reshape ground truths' dimensions to meet the feature maps'. In this way we can compute the classification loss between the two as follows:

$$L^b_{ms}(x, m) = -\sum_{i=1}^{n}\sum_{k=1}^{N}\sum_{j=1}^{s^2} m^k_i(j) \cdot log(l^k_i(j)) \quad (1)$$

Where n is the number of classes, N is the number of frames, s^2 is the number of pixel in the feature map, $m(j)$ is the ground truth and $l(y)$ is the feature map.

In the regression problem we choose a per pixel L2 loss following this formula:

$$L^r_{ms}(x, m) = -\sum_{i=1}^{n}\sum_{k=1}^{N}\sum_{j=1}^{s^2}(v^k_i(j) - e^k_i(j))^2 \quad (2)$$

Where in this case $v(j)$ is the ground truth and $e(y)$ is the estimated value in the feature map.

This loss, cross-entropy for classification and L2 for regression, is then summed with the action classification loss based on the cross-entropy from ego-RNN:

$$L_c(x, y) = -\sum_{i=1}^{n} y_i \cdot log(g_i(x)) \quad (3)$$

Where $g(x)$ is the prediction class of the model while y it's true label.

3.3 Colorization

Our first attempt of variation consists in implementing an optical flow colorization, following the ideas proposed in [3]. The goal of this task is to use the DECO colorization network to transform the two channel warp flow images into three channel ones. In this way, we can use the same architecture previously used to process the RGB frames.

Different architecture choices have been experimented in order to improve the performances of the model, for which you can find further details in section 5.1.

However, despite the different alternatives, we weren't able to implement effectively a colorization block. The network is able to highlight the regions of the warp flow images characterized by the motion, but the performances are far worse than in the previous case.

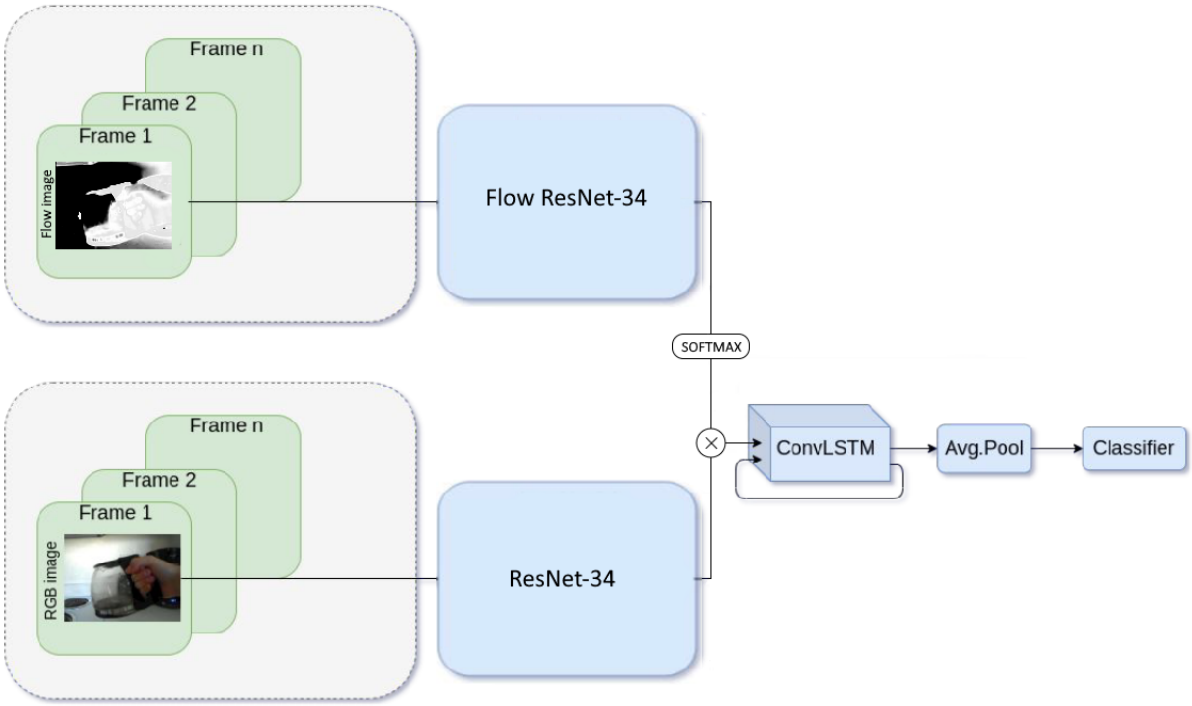


Figure 3: Architecture Cam Flow

3.4 CAM variation

Our last attempt is to prove that the previous variation is not working well not for the frame division but for the colorization techniques.

To do so we implement a new version of the architecture composed by 2 pretrained ResNet-34 and we use as output the features extracted in the fourth layer. The first one takes as input one RGB frame at the time so it is called RGB-Resnet, on the other hand the second one takes 2 warp flow images as input, so it is called resnetOF. In particular, this ResNet is modified in order to take a two channel image as input.

Both of them return a 512x7x7 array and then, before performing an element wise multiplication between them, a softmax is applied at each layer of the output of the ResnetOF before the multiplication. In this way the features obtained from the RGB frames are weighted taken into account the features from the optical flow.

The final result is then fed to the previous ConvLSTM plus fully connected layer architecture to perform the classification task.

Also the training phase follows the previous EGO-RNN two-stage approach.

4 Experiments

4.1 Dataset

The dataset used for this analysis is GTEA61, which is composed by 61 actions performed by 4 persons (S1,S2,S3,S4).

We will use S2 for both validation and test phase and the others for the training phase. In each folder the video frames and the motion maps already computed are stored, also the warp flow for x and y coordinates is provided. Before passing them to the network all images are normalized and scaled to 256x256. Moreover, since GTEA61 dataset contains only 419 elements, we perform data augmentation techniques.

On the train dataset we apply a random horizontal flip and a multiscale corner crop that also resize the image to 224x224. On the other hand, on the validation dataset we apply only a center crop tat resizes each image to 224x224. The normalization performs differently if we are dealing with RBG images or grey scale images. In the first case the normalization is over the 3 channel using the values from ImageNet, instead when we normalize the one channel grey image we utilize the mean of them.

4.2 Implementation details EGO-RNN

The appearance stream is divided into two stages. For the first stage we set the number of training epoch to 200 with an initial learning rate of 10^{-3} that decays by a factor of 0.1 after 25, 75 and 150 epochs. In this stage we train only the the classifier and convLSTM layers.

In the second stage, we set the number of epochs to 150 with a learning rate of 10^{-4} that decays by a factor of 0.1 after 25 and 75 epochs. Instead, in this stage together with the previous ones also the final convolutional layer and the fully connected layer of the ResNet are trained. We utilize for both of them Adam optimizer with a batch size of 32 and selected 7 RGB frames (uniformly sampled in time).

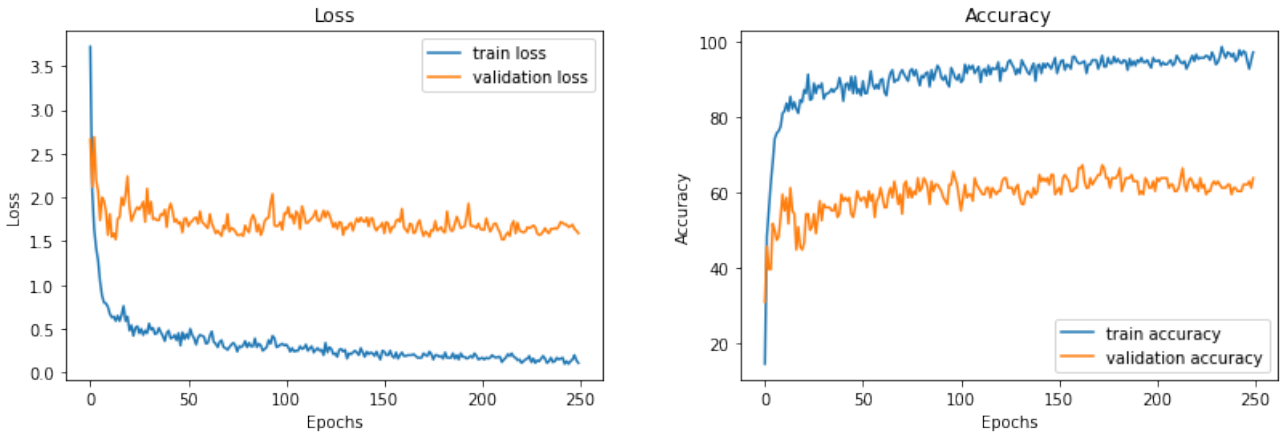
Besides, for the temporal stream the architecture is composed only by a pretrained ResNet-34, that we feed with a sequence of 5 stacked optical flow (as said before in this case we use warp flow). The learning rate is set to 10^{-2} that decays after 150, 300 epochs by a factor of 0.5. It is trained for 750 epochs using SGD as optimizer.

As last point the two streams are joined: to do so we concatenate the results and add a fully connected layers to get the score for each class. The results obtained are reported in the table 1 and show that our model is able to almost reproduce the results proposed in [1] since it achieves an accuracy of 70.6%. The drop in performance should be addressed to the different version of torch, also our version is using 7 frames instead of 25 and this decreases the number of the data for the training. Also in figure 4 the graphs show the behaviours of loss and accuracy on train and validation set during the two stream training phase. It is noticeable that there is a gap between the two loss functions that cannot be addressed due to the dataset nature.

Configurations	Accuracy% 7 Frames	Accuracy% 16 Frames
ConvLSTM	57.7	51.7
ConvLSTM-attention	58.6	62.9
Temporal-warp flow	42.2	42.2
two-stream (joint train)	67.2	70.6

Table 1: Results ego-RNN

The same process is after repeated changing the number of frames from 7 to 16. As seen in the table, increasing the amount of frames does not lead to a huge improvement in terms of accuracy without the attention layer, while the attention layer makes the difference with higher number of frames. This is to be pointed at the convLSTM layer that has an higher chance to receive for the attention model the correct attention map in input due to the higher number of frames.



(a) (b)

Figure 4: Two stream

4.3 Implementation details MS TASK

For this task we add a motion block in the second stage of the EGO-RNN. We utilize the model already provided in the previous step as a starting point and then we develop the new box as explained in 3.2. Differently from as done in paper [2] we decide not to consider the two losses as equally important, so we add a parameter α as a coefficient for the ms loss before its back-propagation to select how much importance give to the motion loss. We set the train hyper-parameters as in the previous second stage and we also try different set of learning rates and alpha, both in the binary and in the regression case.

As it is possible to observe in the two heat-maps in figure 7 and figure 8 the accuracy considering the regression problem is higher about 3-4% than the respective binary one for the 7 frames experiments, instead for the 16 frames experiments the higher result are found for the binary case. As noticeable in figure 13 and figure 14 (Appendix) the accuracy and loss from the binary case are less smooth than the ones from the regressor. This should be caused by the fact that the network is able to get more information, since in the binary classification problem the adding of a threshold removes some useful information.

We also observed that adding a activation function to force the output in the interval $[0, 1]$ does not improve the results, instead hinders it reaching at peak 66% of accuracy.

The best model found so far is related to the binary classification configuration with learning rate 10^{-4} and alpha 1,which leads to an accuracy of 74%, as seen in the figures below. It shows that the two losses should be considered as equally important. We also try different values for the learning rate but all of them lead to a dramatically decreasing of the performance of the model.

Also we analyze how the adding of the motion task influence the class activation map. As shown in figure 5 and figure 6 the spatial attention map in the second case is focused on the arms and hands region since they are moving elements, while in the previous one it depends only by the object classification.

Figure 5: CAM Ego-RNN

Figure 6: CAM Ego-RNN + motion task

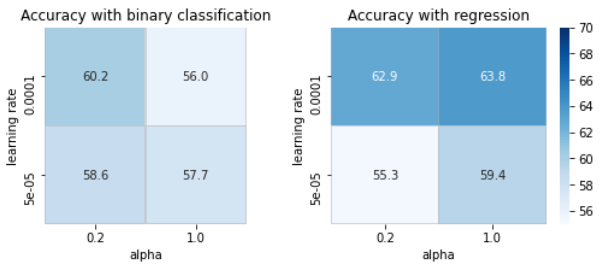


Figure 7: Accuracy motion task 7 frames

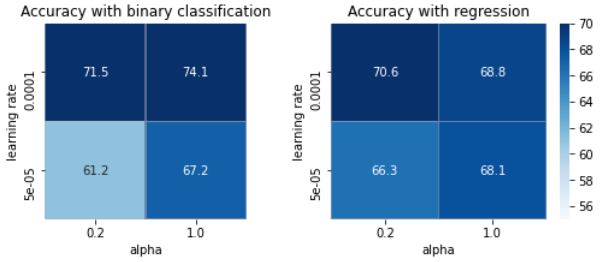


Figure 8: Accuracy motion task 16 frames

4.4 Implementation details CAM variation

Finally, we want to demonstrate that by using separated optical flow frames instead of stacked ones may lead to better results. To do so we train the model varying set of hyperparamters and changing the parts of the network we want to optimize.

We weren't able to make the model converge training all ResNetOF, so we followed the same training procedure as ego-RNN [1], ConvLSTM and FC in the first phase adding only the fourth layer of the ResnetOF in the second stage. RGB ResNet is loaded from ego-RNN with CAM model.

For this section, we utilize the same hyperparameters used 4.2 for the two phases. Our best model is able to achieve an accuracy up to 61%. That is comparable with the previous architecture and also demonstrates that the lack of results in Appendix is indeed related to the colorization techniques.

5 Conclusion

In this project we replicated the implementation of the Ego-RNN and we obtained results almost comparable to those reported in [1]. Then we integrated with the appearance stream a motion segmentation block for which two alternative problems were presented: binary and regression. In particular, we observed that the latter performs better thanks to the exploitation of motion maps with decimal values representing the movement of the pixels. Finally we experimented new possible variations of the current architecture and we proposed an alternative use of the spatial attention mechanism.

The first person action recognition task opens the possibility to further improvements, which can be performed with a larger amount of time and better computational resources. For instance, different kind of self-supervised tasks could be considered and integrated with the architecture. Moreover, also the colorization variation could be further explored taking into account all the attempts made in this project.

6 Appendix

6.1 Colorization

In this section we provide further details on the different implementations adopted for the colorization task. As said before, our goal was to implement a colorization block to convert the 2 channel warp flow into 3. This will enchant the network capabilities and let us reuse the RGB architecture.

In our first attempt, we replicate the architecture proposed in [3] for our colorization block.

In particular, initially the network was composed by a first convolutional layer with kernel size equals to 7, stride 2 and padding 3: it receives as input a two channels image and its output passes through 8 residual blocks, each of them is characterized by convolutional layers, leaky relu and batch normalization. Finally it goes to the last convolutional layer with kernel size 1, stride 1 and padding 0 that returns a 3 channel image that is successively upsampled to reshape the right dimensions.

We first tried to integrate the DECO architecture with our motion task, so we fed our final network with the colorized optical flows.

As an alternative, we also tried to use two-stream architecture without the self-supervised task in such a way that we can exploit the colorized 3 channel optical flows in place of the previous two channel ones. In practise we substitute the flow ResNet with a RGB ego-RNN, spatial attention layer included. First we train the colorization block maximizing the entropy of ResNets' output. We then concatenate the pretrained colorization block with ego-RNN, the latter pretrained with GETEA+ 61 RGB images. Freezing all the layers of ego-RNN let us train the model to colorize through loss' back propagation (3). We train also the last classifier after the convLSTM to let the model adjust itself to new flow input images. We do our experiments both with stacked optical flow frames and with single frames as input, and also in one case we select them randomly and in the other sequentially.

The resulting images are characterized by a pattern similar to that of Figure 10 . We can see that there are some points in the images which make harder for the network to learn the motion features. To fix this problem, we decide to change the convolution transpose 2D layer to an up-sampling one. In fact, we notice that the transposed

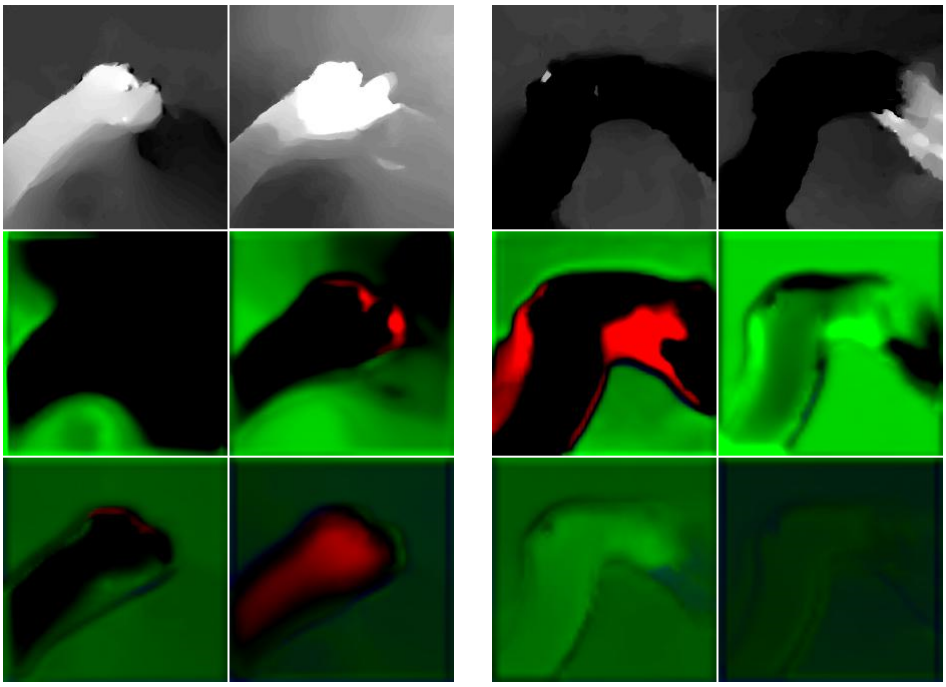


Figure 9: Colorized optical flow. In order: x channel, y channel, colorization epoch: 1,11,21,71

convolution has a problem of overlap which is the reason of the kind of texture we obtain.

As an alternative, we apply some changes to the DECO architecture, trying to make it a little bit lighter. In this sense, we create an intermediate convolutional layer. So, we have a first convolutional layer which gives as output 32 channel, the second one 64 channel and the last one the final 3 channel tensors. We also modify the number of residual blocks which was initially set to 8: we reduce them to 4, 1 and lastly we also train our network without them.

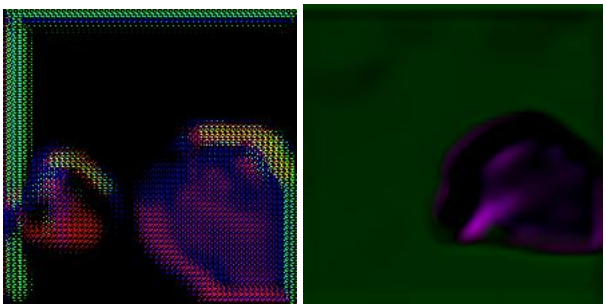


Figure 10: Colorized optical flow, before and after deconv adjustments

In order to change our training phase, we also try a two stage procedure in which we use the original network pretrained on the RGB images, then we train the DECO architecture for 100 epochs and lastly we freeze the colorization block and we train the final fully connected layer.

In this case the images obtained are the ones in figure 7. We can see that the colorization is not so satisfactory, so, even if the network is able to select the right region of the frame which characterizes the motion, it is not enough for classification. In fact, in any situation the obtained results can't exceed an accuracy of 8%, which is surprisingly low.

How it has been showed in [3], the colorization task could be very useful and many researches can be further

developed to better explore its possible uses. However, all the experiments carried out are inadequate for the proposed problem.

References

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

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

[3] F. M. Carlucci, P. Russo, and B. Caputo, *(DE)2CO: Deep Depth Colorization*,

[4] Ali Diba; Vivek Sharma, Luc Van Gool, Rainer Stiefelhagen, *DynamoNet: Dynamic Action and Motion Network*, ICCV 2019

[5] Jiangliu Wang; Jianbo Jiao; Linchao Bao; Shengfeng He; Yunhui Liu; Wei Liu, *Self-Supervised Spatio-Temporal Representation Learning for Videos by Predicting Motion and Appearance Statistics*, AAAI 2019

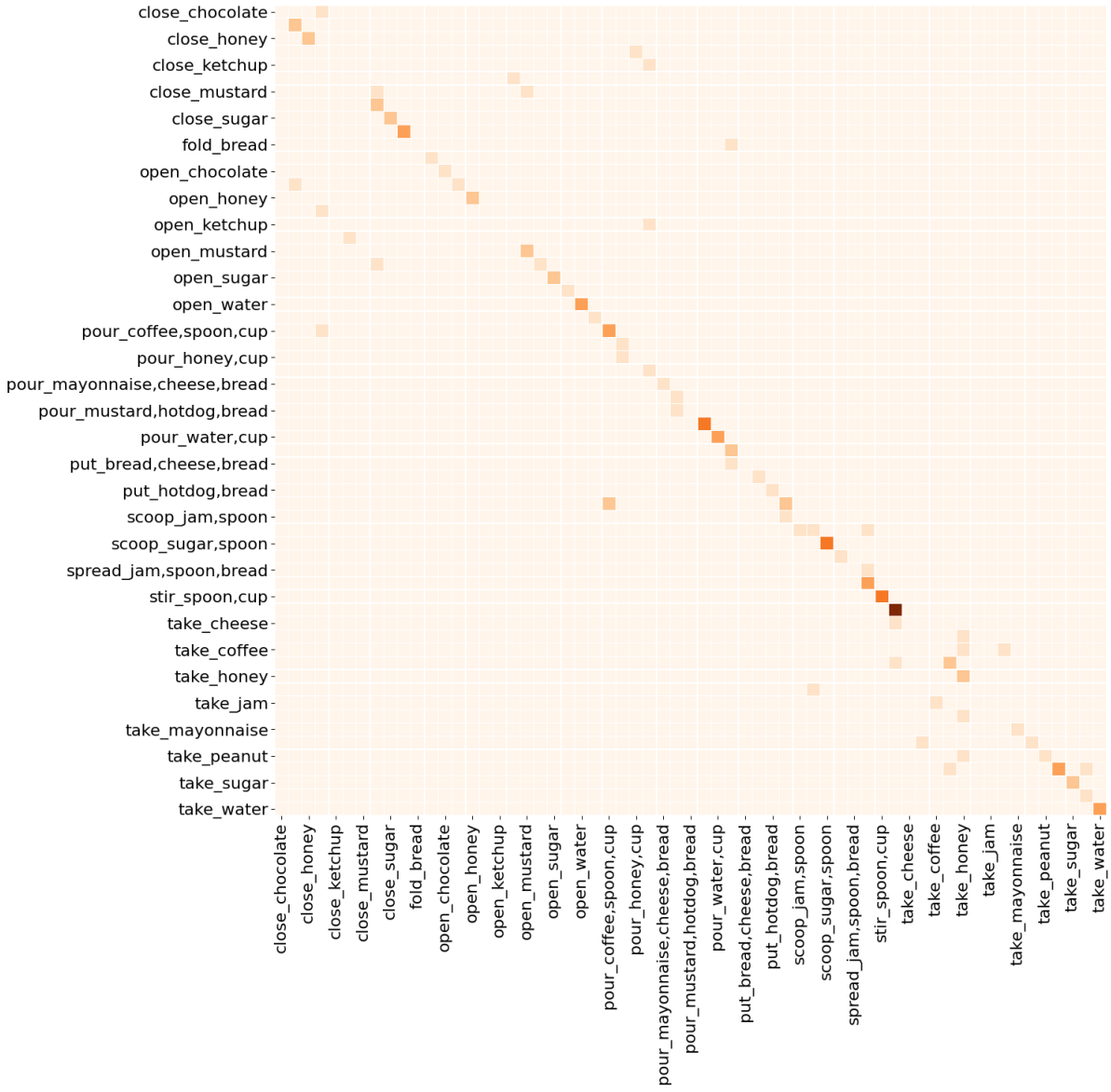


Figure 11: Confusion matrix Two Stream

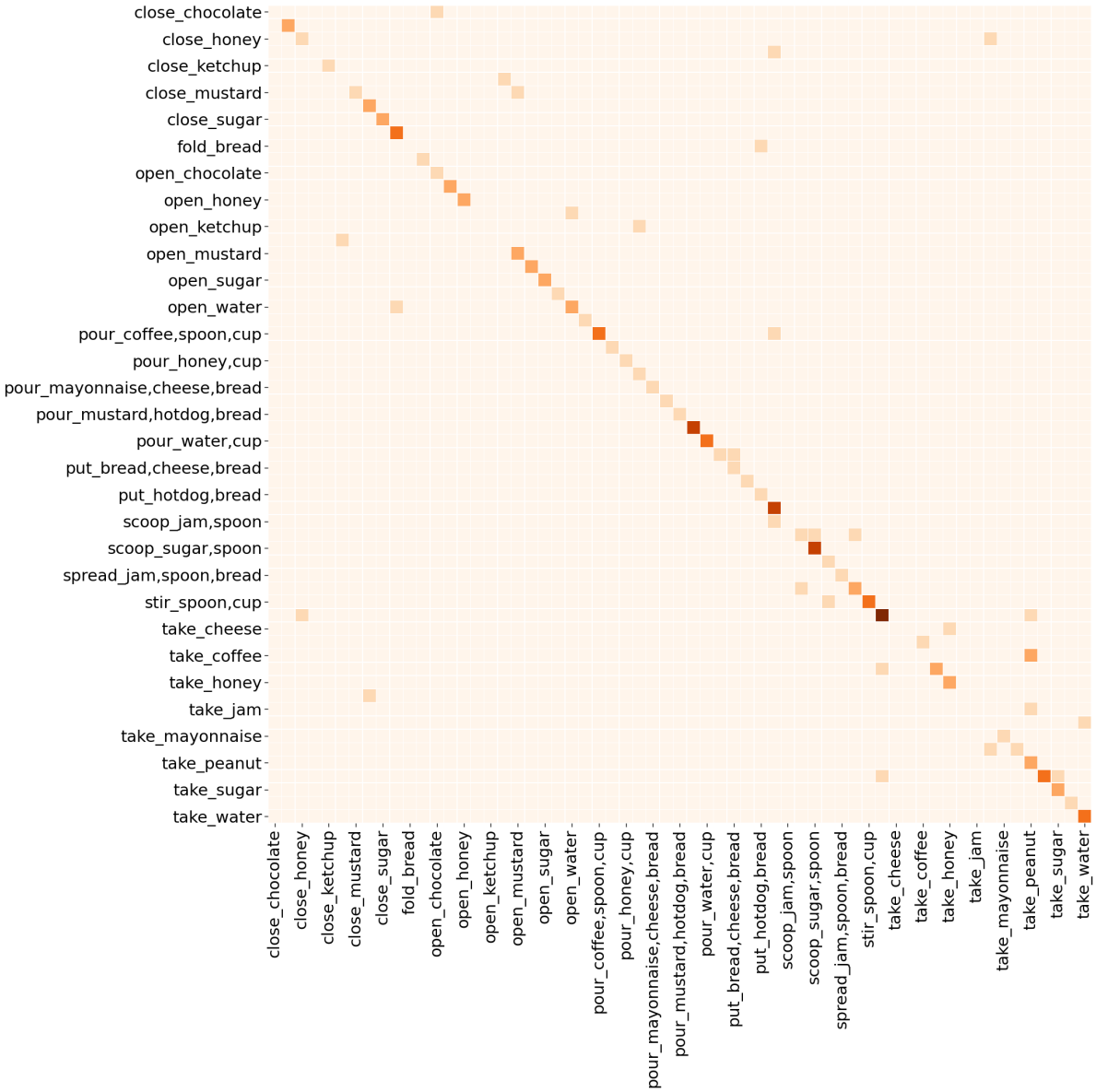


Figure 12: Confusion matrix motion task

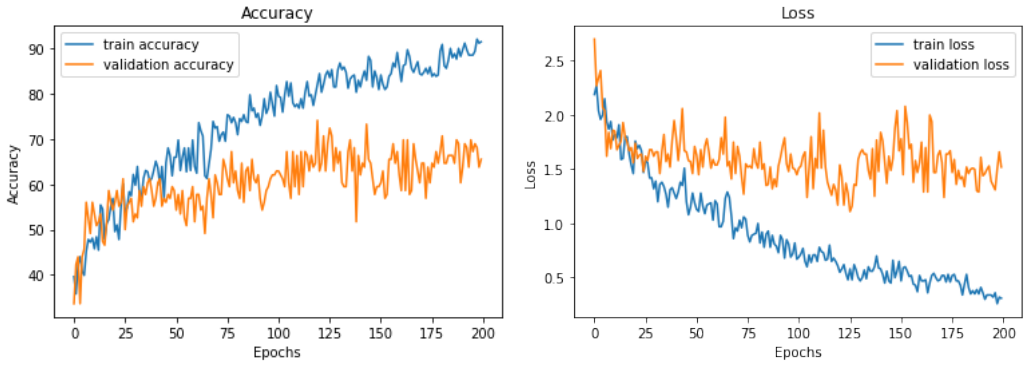


Figure 13: Accuracy and loss motion task with binary classification

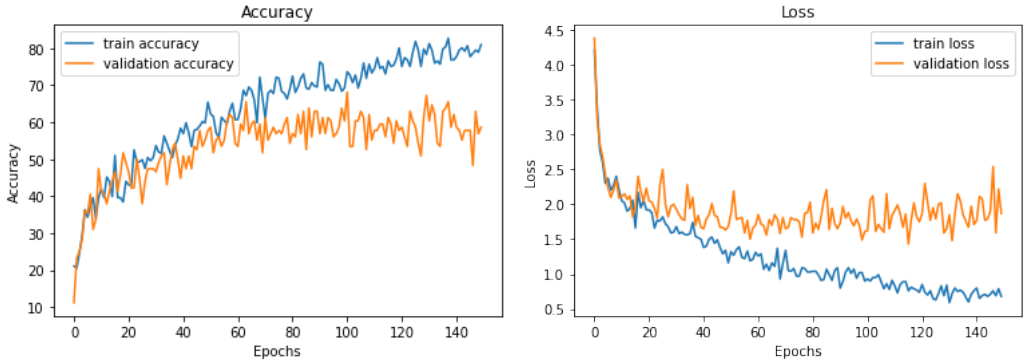


Figure 14: Accuracy and loss motion task with regression