Imprinting the Motion: Self-Supervised Approach to Instilling Motion Information since the Beginning for First Person Action Recognition.

Simone Papicchio
Politecnico di Torino
s281811@studenti.polito.it

Eleonora Poeta
Politecnico di Torino
s292481@studenti.polito.it

Federico Lorenzo Pes
Politecnico di Torino
s287822@studenti.polito.it

Abstract

First person action recognition (FPAR) task is one of the most challenging in action recognition field. Most of the existing works address this issue with two-stream architectures, where the visual appearance and the motion information of the object of interest, are exploited. In this paper, we use as starting point the Ego-RNN architecture with the addition of the Motion Segmentation (MS) auxiliary task. We propose the injection of a new branch in the architecture, in order to employ the motion information more effectively. This leads to have better predictions.

1. Introduction

In the last decade there was a large diffusion of mainstream wearable cameras, caused by the progresses in: sensor miniaturization, low-power computing and battery life. Every year, millions of hours of videos are captured by these devices, creating the opportunity to develop new capabilities and applications for computer vision. In this context, First person action recognition (FPAR) field of research, has attracted increasing interest changing the previous point-of-view. It has passed from a third-person perspective, based on the usage of sensors into the environment, to the possibility of following the user, in-mobility, by means of wearable devices. FPAR has an enormous potential in societal issues and in many of everyday activities, passing from security and surveillance, to sports, health and well-being.
The main challenge of FPAR, in contrast to what happens in third-person action recognition, is the lack of information about the pose of the main actor, and so, it becomes pivotal to extract from the video frames as much information as possible on the objects being manipulated, their position and the motion data encoded in the video. Second problem is represented by egomotions present in the frames, that derives from the usage of wearable cameras placed on the actor body.

A well-known approach to address this task is to combine two type of information: the visual appearance of the object of interest and the motion information. The visual appearance is obtained by means of the spatial stream that processes RGB images. Instead, a second stream provides the motion information, taking as input the optical flow that represents the pattern of apparent motion of objects, surfaces, and edges caused by the relative motion between an observer and a scene [10].

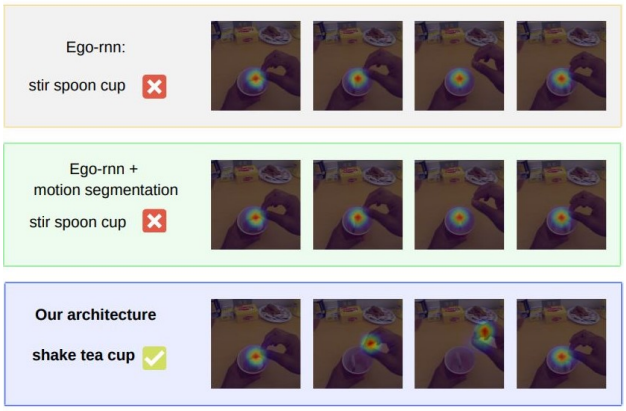


Figure 1. Our architecture is able to further exploit the motion information provided by the motion segmentation by merging them with the appearance features in the first layers of the backbone. The result is a model that better focuses on the relevant elements for action recognition and this lead to the correct prediction (shake tea cup instead of stir spoon cup)

These two streams can be combined either through a simple weighted sums [6][5] or using an additional fully connected layer [7]. The main disadvantage of this solution is that the visual appearances and the motion features are learned separately, triggering, not only the growth of the number of parameters of the overall architecture, but also the non-exploitation of the existing spatio-temporal relationships.
In this paper, we address this duty by proposing a different approach that tries to use an end-to-end architecture, composed of a single RGB stream influenced by the motion-

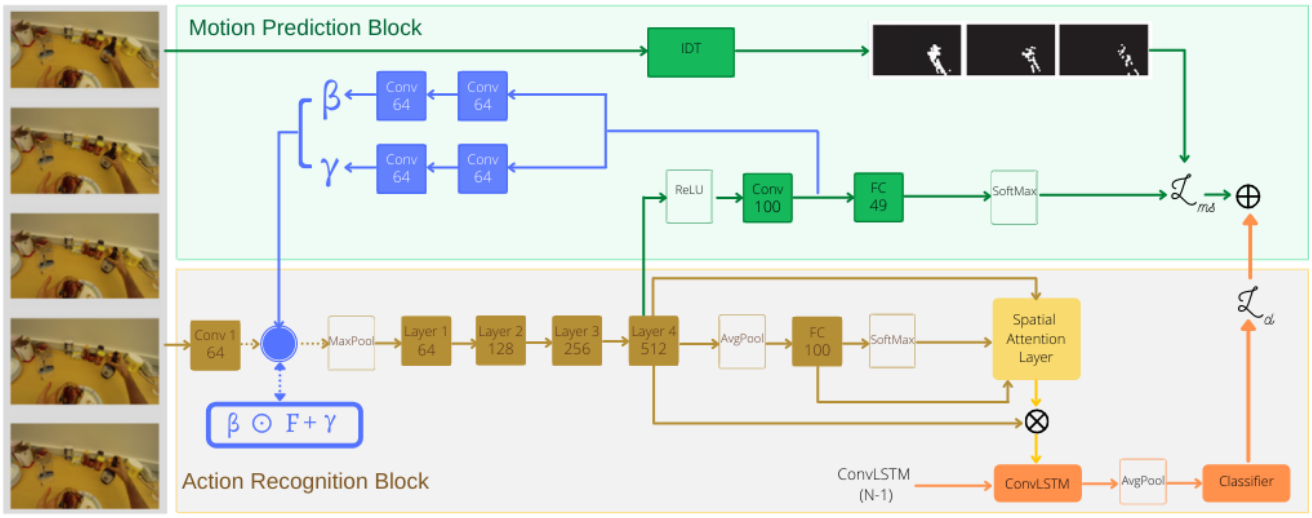


Figure 2. Our proposed architecture. The Action Recognition Block extracts important spatial and temporal information from the video with the exploitation of the ResNet-34 (mustard), Spatial Attention Layer (yellow) and ConvLSTM (orange). Moreover, it takes advantage from the auxiliary task of the Motion Prediction Block, by embedding its knowledge inside the first layers of the backbone. This is performed with a feedback branch (blue) that takes as input the features of the motion segmentation (MS) task (green) [4]. The Motion Prediction Block takes, as input, the appearance features from the layer 4 of the ResNet and tries to identifies which parts of the image are going to move.

prediction (MP) self-supervised task [4]. In addition, we include an auxiliary task to inject the idea of movement directly in the first layers of the RGB stream, in order to guide the model to pay more attention on what really moves, rather than considering the static background.

2. Related Works

First Person Action Recognition. The literature on FPAR is very extensive and has moved from using hand-crafted features to the deep learning application. Egocentric action recognition requires a fine understanding of the interactions between hands and objects. Most papers in the literature focus their attention on object detection, in order to offer a better object-context recognition.
In this regard, in 2016, Minghuang *et al.* [2] proposed an architecture that learns features capturing objects' attributes exploiting hands segmentation technique.

Others papers, instead, propose solutions encompassing the spatio-temporal attention, or the gaze annotation, for the purpose of localizing the interacting objects.

In 2018, Sudhakaran *et al.* [7] proposed an architecture to address the first person recognition task introducing a spatio-temporal encoding mechanism, the ConvLSTM, that allows to focus on the point where the object involved in the action is located. This architecture, called Ego-RNN, proposes a multi-stream approach. One stream is dedicated to the analysis of RGB frames for spatial-temporal encoding. The other stream is specialized in processing optical flow features. These two flows are merged by means of a fully-connected layer at the end of the architecture to provide the predictions. This is the starting point for our analysis. There are two main problems of the Ego-RNN. Firstly, the multi-stream approach requires high expense of resources that cannot be provided by wearable devices. Furthermore, optical flow features are learned in a separate branch and the two branches are late joined.

Single stream model.There are alternatives that try to overcome the ahead limits. The architecture proposed by Planamente *et al.*[4] is composed by a single main-stream, devoted to the action recognition, with the addition of a self-supervised block implementing an auxiliary task that exploits motion predictions to intertwine motion and appearance knowledge.
Even if the optical flow features can provide meaningful contributions, experiments demonstrate that, the usage of the motion condition to modulate RGB features, improves the detection accuracy. This was shown by Jiaojiao Zhao and Cees G. M. Snoek [9] with an architecture that embeds RGB and optical-flow into a single stream for spatio-temporal detection. However, the pre-computation of the optical flow is still computationally expensive.

3D CNN. Another way to analyze a video is using 3D tensors with two spatial and one temporal dimension. Hence, this leads to the usage of 3D CNNs as processing units. In [1] it is proposed a model that extracts features from both spatial and temporal dimensions by performing

3D convolutions, thereby capturing the motion information encoded in multiple adjacent frames. Although, C3D are hard to optimize because they need both large scale datasets, and long time period to converge during the training.

3. Method

3.1. Architecture Overview

The starting point of our architecture (Figure 2, action recognition block) is the RGB stream defined by Sudhakaran *et al.* [7]. Firstly, we extract uniformly N sparse representative RGB frames, from each input video segment. These frames are used as input to a pretrained CNN backbone in order to extract the embedded appearances from each frame. The spatial attention layer is used to make the network focused on the regions consisting of the objects. The output is then passed to a ConvLSTM network. The last layers are used for the classification (avgpool, dropout(0.70), fully connected). This stream is used to extract important spatial and temporal information from the video. However, the resulting features are still lacking the motion information which is crucial for FPAR task. This issue is solved in the two-stream approaches relying on explicit optical flow data. In contrary, we decide to follow the idea of Planamente *et al.* [4], i.e. by extending the basic architecture into a single multi-task network. During the training, the network has to solve two different tasks concurrently: the action recognition task and a motion-segmentation (MS) auxiliary task. The second one is formalized as a self-supervised problem which takes as input a single RGB frame and tries to identify which parts of the image are going to move.

This identification task is treated as a labelling problem which minimizes the differences between the motion map, in which pixels are labeled as moving or not, and the object movements predicted by the network for a single static RGB frame. The unsupervised motion maps are extracted from the videos using the approach in [3] and leveraging the Improved Dense Trajectories IDT [8], for extracting "stabilized" motion information. The self-supervised motion prediction task is used by the architecture in order to benefit from both motion and appearance information, using a single RGB stream. The resulting architecture is lighter than the two stream model with less parameters and it is end-to-end trainable.

Our contribution We decide to further exploit the auxiliary task by embedding its knowledge inside the first layers of the backbone. The main idea is that the low-level RGB features, from the appearance network, are adjusted by two factors β and γ in order to give more importance to the zones that are classified as "in movement". This is performed by means of the feedback branch, which takes, as input, the features of the motion segmentation task, and outputs β and γ. In order to modulate the appearance network, we apply a transform function $\theta(\cdot)$, with the learned β and γ, to the RGB features F^{rgb} computed in the first layers of the backbone:

$$\theta(F^{rgb}) = \beta \bigodot F^{rgb} + \gamma \tag{1}$$

$\bigodot$ is an element-wise multiplication operation and β, γ and F^{rgb} have the same dimensions. The motion information represented by β and γ influences the appearance network by both feature-wise and spatial-wise manipulations. The complete network is shown in Figure 2. In order to visualize what changes when $\theta(\cdot)$ is applied, and to conceptually understand the proposed implementation, we plot the generated feature maps from the appearance network before and after $\theta(\cdot)$ [Fig. 3].

We can also visualize the effect of our contribution on the architecture by plotting the class activation maps (*CAM*) used in Ego-RNN [7] before and after the insertion of the feedback branch. As we can see from the Figure 1, both Ego-RNN and Ego-RNN with motion segmentation fail to predict the action of the video. In fact, they are predicting the "stir spoon cup" action rather than the "shake tea cup" action. Our architecture, in contrary, correctly predicts the action. This can be explained by looking to the CAMs where is clear that the focus of the action does not remain on the cup (as in the other architectures), but it follows the action of the hand (the hand goes up and down to shake the tea).

3.2. Architecture Details

Mathematical Details. Let T be a training set consisting of $T_i = \{H_i, y_i\}_{i=1}^{n}$, where H_i is a set of N timestamped images uniformly sampled from the video segment and y_i is the action performed in that video. Let also be x the output of the model M, depending both on the back bone until and after the Layer-4. Let $g(x)$ be a class probability estimator on the embedding x. We define the categorical cross-entropy classification loss is defined as:

$$L_{cl}(x,y) = -\sum_{i=1}^{N} y_i \cdot \log(g(x_i)) \tag{2}$$

The MS branch ends with a fully connected layer of size s^2 followed by a Softmax, and it is trained with a loss L_{ms} based on the per-pixel cross entropy loss between the computed label image l_{ms} and the Ground-Truth motion map m. The estimated l_{ms} is obtained as a function of both image embedding z derived from the backbone until the Layer-4 and MS head parameters. Thus, the L_{ms} loss can be defined as:

$$L_{ms}(z,m) = -\sum_{i=1}^{n}\sum_{k=1}^{N}\sum_{j=1}^{s^2} m_i^k(j)\cdot\log\left(l_{ms,i}^k(j)\right) \quad (3)$$

The model solves jointly two separate optimization problems: minimize both L_{cl} and L_{ms} (2) (3). L_{cl} affects by back-propagation the Action Recognition head (blocks following the Layer4 of the ResNet-34). On the other hand the Motion-Segmentation branch is updated only by its loss. Both losses update the backbone and also the feedback branch of the architecture by a weighted sum of the two through the coefficient α, obtaining :

$$L_{tot} = L_{cl} + \alpha * L_{ms} \quad (4)$$

The value of α is derived by an hyperparameter optimization.

Implementation. The backbone chosen for this architecture is a ResNet-34 pretrained on ImageNet. The output features of the Layer-4 of the backbone, which size is 7*7*512, are forwarded also to the Motion Prediction block. The Conv100 block of the Motion Segmentation branch (green in Fig 2.) reduces the feature channels to 100 and, after a flattening operation, the size provided to the fully-connected layer is 49. Meanwhile the output of the Conv100, afore mentioned, is also subjected to a UpSampling operation. From this we obtain features of size 64*64*100. This is the input of the two branches by which we calculate β and γ (blue part in Fig 2.). Both of them reduce the channel multiplier to 64 and then are "joined" with the output of the Conv1 with the (1) formula.

4. Experiment

4.1. Dataset

All experiments presented in this paper were trained and validated on the GTEA61 dataset that contains a collection of videos, corresponding to 61 actions, performed by 4 different users, recorded with a camera mounted on a cap worn by the subject. In the dataset version provided to us, each video is already divided into frames, scaled at 256. We also were provided with the extracted warp flow and the ground-truth motion maps, in order to implement the self-supervised task [4].

Our model's backbone is the ResNet-34 that was pretrained on the ImageNet dataset.

4.2. Implementation Details

As mentioned previously, ResNet-34 pretrained on ImageNet dataset is used as backbone for our architecture to extract frame level features and generating the object specific spatial attention map. We use a ConvLSTM module

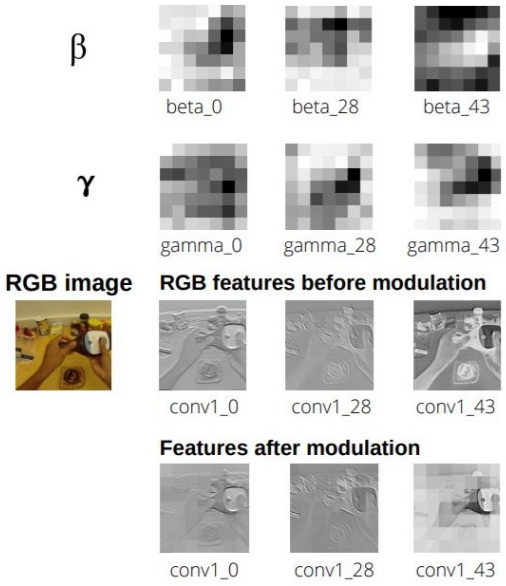


Figure 3. The first two lines represent β and γ calculated using as input the features from the motion segmentation part at time *(t-1)*. The learned factors are then applied to the appearance features at time *(t)*. The numbers after the name of the features represent the channel, i.e. conv1_43 is the appearance feature from the ResNet conv1 at channel 43.

with 512 hidden units for temporal encoding, initialized as in [7]. Our approach includes three training stages that are outlined below.

Stage 1. The first stage of the training is dedicated to the ConvLSTM and the Classifier. It is used to provide prior knowledge to these two blocks, by feeding them the features extracted by the pretrained backbone.In the first stage, the network is trained for 300 epochs with an initial learning rate of 10^{-3} and the learning rate is decayed by a factor of 0.1 after 25, 75 and 150 epochs. Batch size in this stage and in the next step is 32. We also apply dropout at a rate of 0.7 at the fully-connected classifier layer.

Stage 2. In addition to the afore mentioned blocks, in this stage, also the Layer-4 and the fully-connected layer of the backbone are trained. This greatly increases the architecture's capability of localizing the relevant features for discriminating the activity. Furthermore, the motion segmentation MS is concurrently trained, in order to give a hint about the motion features to the appearance stream. In this stage the blocks involved are trained for 150 epochs with an initial learning rate of 10^{-4} that is then decayed of 0.1 after 25, 75 epochs.

Stage 3. The third stage serves to implement the feedback branch with the addition of β and γ. This branch helps the architecture on focusing more on the frame's section involved with the motion, since the first layers of the backbone [9]. In the final stage the network is trained for 200 epochs with learning rate 10^{-4} that is then decayed of 0.1

	Stage 1	Stage 2	Stage 3
Batch Size	32	32	4
Number of Epochs	300	150	200
Learning Rate	1e-3	1e-4	1e-4
Step Size	25 75 100	25 75	50 100 150

Table 1. Hyperparameters used in training stages

after 50, 100 and 150 epochs.
Here we use a batch size of 4. An overview of all these parameters is shown in Table 1.

4.3. Ablation Study

τ and α. Losses from the classifier L_{cl} (2) and from the motion segmentation task L_{ms} (3) are weighted through the coefficient α (4). Our results Fig.4.3 showed that, for values of α different than 1, the accuracy of the model is lower.
The classification task predicts if a certain area of the frame is moving or not, according to the threshold τ. Initially, each frame of the ground-truth motion maps is composed only by the pixels with values either 0 or 255 . Then we scale each frame, obtaining that there are different units whose values represents the percentage of motion in any area that originated the units. The threshold, aforementioned, is used to map every unit above τ as 1 and to 0 the others. This kind of interpretation enables us to implement the classification task. We performed a hyperparameter optimization on τ obtaining that the best accuracy values of the model were found for: $\tau = \mathbf{0.2}$. Table 2

Classification or Regression. The motion segmentation task can be implemented either as a classification task or a regression task. The regression task predicts the percentage of movement in a certain area of the picture. We select two losses for the regression, the MSE and the L1 and so the L_{ms} changes as follow:

$$L_{ms,MSE} = \sum_{i=1}^{n} \sum_{k=1}^{N} \frac{1}{s^2} \sum_{j=1}^{s^2} \left\| f_i^k(j) - y_i^k(j) \right\|_2^2 \quad (5)$$

$$L_{ms,L1} = \sum_{i=1}^{n} \sum_{k=1}^{N} \sum_{j=1}^{s^2} \left\| f_i^k(j) - y_i^k(j) \right\| \quad (6)$$

Unfortunately, our experiments show that the model that uses the classification task greatly exceeds the one with the regression task for both losses, even with lower α. Table 3.

Where to add β and γ. The role of β and γ is to embed motion features into the main stream of the architecture and, to do so, it is necessary to understand at what architecture level is more convenient to insert this features. We tried different settings for our model finding out that, applying the features after the *Conv1* of the backbone ResNet-34, the model increases in accuracy. This result is probably obtained because, if we apply the motion features β and γ to the last layers of the ResNet-34, we are going to modify some high level features that are relevant to identify the action and the object involved.

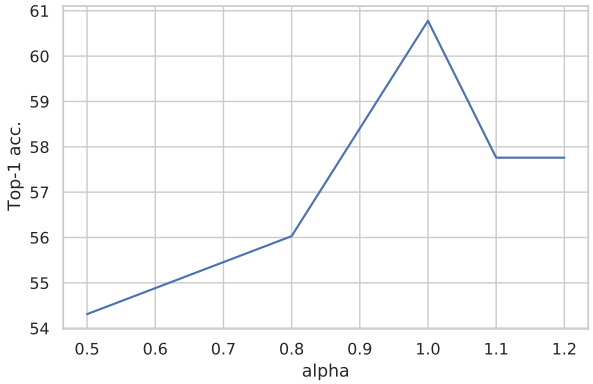


Figure 4. Values of Top-1 accuracy obtained w.r.t the paratemer α.

Number of training stages. Since, theoretically, this architecture is end-to-end trainable we initially decide to have two training stages as performed in [7], i.e. we jointly train the action recognition block and the motion prediction block, but we obtained poor results. This is probably caused by the small size of the dataset. More precisely, when we train the entire model, for implementation reasons, the feedback branch will use the feature of the motion segmentation at time $(t-1)$ rather than the feature at time t. Therefore, since at the beginning the motion prediction block does not have any kind of prior knowledge, the calculated outputs, β and γ, wrongly modify the RGB main stream. This may be solved with a longer training and a larger dataset. Our solution, (forced by the poor resources available from colab) is to train in the second stage only the MS and then add a third training stage that leverages the learned weights as prior knowledge.

What should be trained. To obtain better performances from this architecture is necessary, not only to

τ	Top 1 Accuracy %
0.00	58.62
0.15	60.34
0.18	59.84
0.20	**62.06**
0.22	56.89
0.25	53.44
0.35	58.62

Table 2. Accuracy values obtained by varying the parameter τ

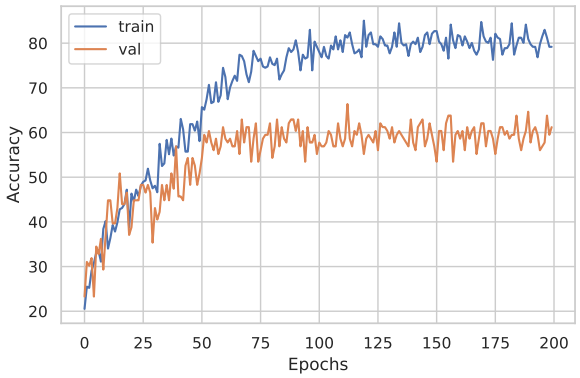


Figure 5. Training and Validation of the variation

choose where to do the grafting of the feedback branch into the network, but also, to select which blocks to train along with this branch. The intuition is that also the backbone should be partly trained in order to understand this new type of features. Our studies showed that, to achieve higher results, we need to jointly train the Conv1 and Layer-4 of the ResNet-34, and to freeze the blocks of the Motion-Segmentation MS branch, during the third stage. The results we obtained are probably limited by the size of our dataset. In fact, if we train the entire backbone in the third stage, the outcomes show over-fitting on the training set. Following this, we deduce that the architecture has probably too many parameters to deal with and it is not able to generalize correctly. We think that this could be an intuition for further studies.

4.4. Results

The proposed variation does not bring the highest accuracy among the state-of-the-art architectures. However, in order to understand if what we did really improved Ego-RNN [7], we performed different training with different parts of the architecture. In Table 3 are reported the average results of the obtained accuracy scores. In the upper part of the table are displayed the scores obtained by the Ego-RNN's part, while, in the lower part, are reported the accuracy values of our architecture. All the results are obtained with *number of frames* = 7. Even if our variation does not outperform the Two-joint Stream, our proposed model can be theoretically trained end-to-end, given that we have a lower number of parameters and this brings the model to converge faster. In addition, as it is possible to see in Fig. 5, our variation manages to reach about the 80% of accuracy in the training. Even if there is a small overfitting, we believe that with more powerful resources we could overcome this problem. Interesting are also the confusion matrices present in the appendix where is evident as our variation makes more stable prediction than the MS

Configuration	Accuracy
Temporal-warp Flow	42.24%
ConvLSTM	52.56%
ConvLSTM with Attention	57.75%
Two-joint Stream	67.10%
MS Regression task with MSE	48.27%
MS Regression task with L1	50.86%
MS Classification task	60.77%
Our Variation	**65.09%**

Table 3. Results of the experiments on GTEA61. First block Ego-RNN's parts. Second block our architecture's parts

5. Conclusion

In this paper, we presented our architecture that is based on the Ego-RNN architecture and MS auxiliary task for FPAR. Its improvement is given by the implementation of the feedback branch that helps the architecture to focus more on the motion features, since the beginning.

From the confusion matrices in the appendix (A B) we notice that both Ego-RNN + MS and our architecture are not able to correctly classify the "take action". We think that this is caused because in these videos the object of the action usually is shown in the last frames and, as a consequence, the architectures focus only on the motion of the hand or the objects in the background resulting in wrong predictions. At the end, we obtained promising results and we think that the path of motion segmentation could still lead the FPAR to major improvements.

References

[1] Shuiwang Ji et al. "3D Convolutional Neural Networks for Human Action Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.1 (2013), pp. 221–231. DOI: 10.1109/TPAMI.2012.59.

[2] Minghuang Ma, Haoqi Fan, and Kris M. Kitani. "Going Deeper into First-Person Activity Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[3] Deepak Pathak et al. "Learning Features by Watching Objects Move". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[4] Mirco Planamente, Andrea Bottino, and Barbara Caputo. "Self-Supervised Joint Encoding of Motion and Appearance for First Person Action Recognition". In: *2020 25th International Conference on Pattern Recognition (ICPR)*. 2021, pp. 8751–8758. DOI: 10.1109/ICPR48806.2021.9411972.

[5] Karen Simonyan and Andrew Zisserman. "Two-Stream Convolutional Networks for Action Recognition in Videos". In: *CoRR* abs/1406.2199 (2014). arXiv: 1406.2199. URL: http://arxiv.org/abs/1406.2199.

[6] Swathikiran Sudhakaran, Sergio Escalera, and Oswald Lanz. "LSTA: Long Short-Term Attention for Egocentric Action Recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

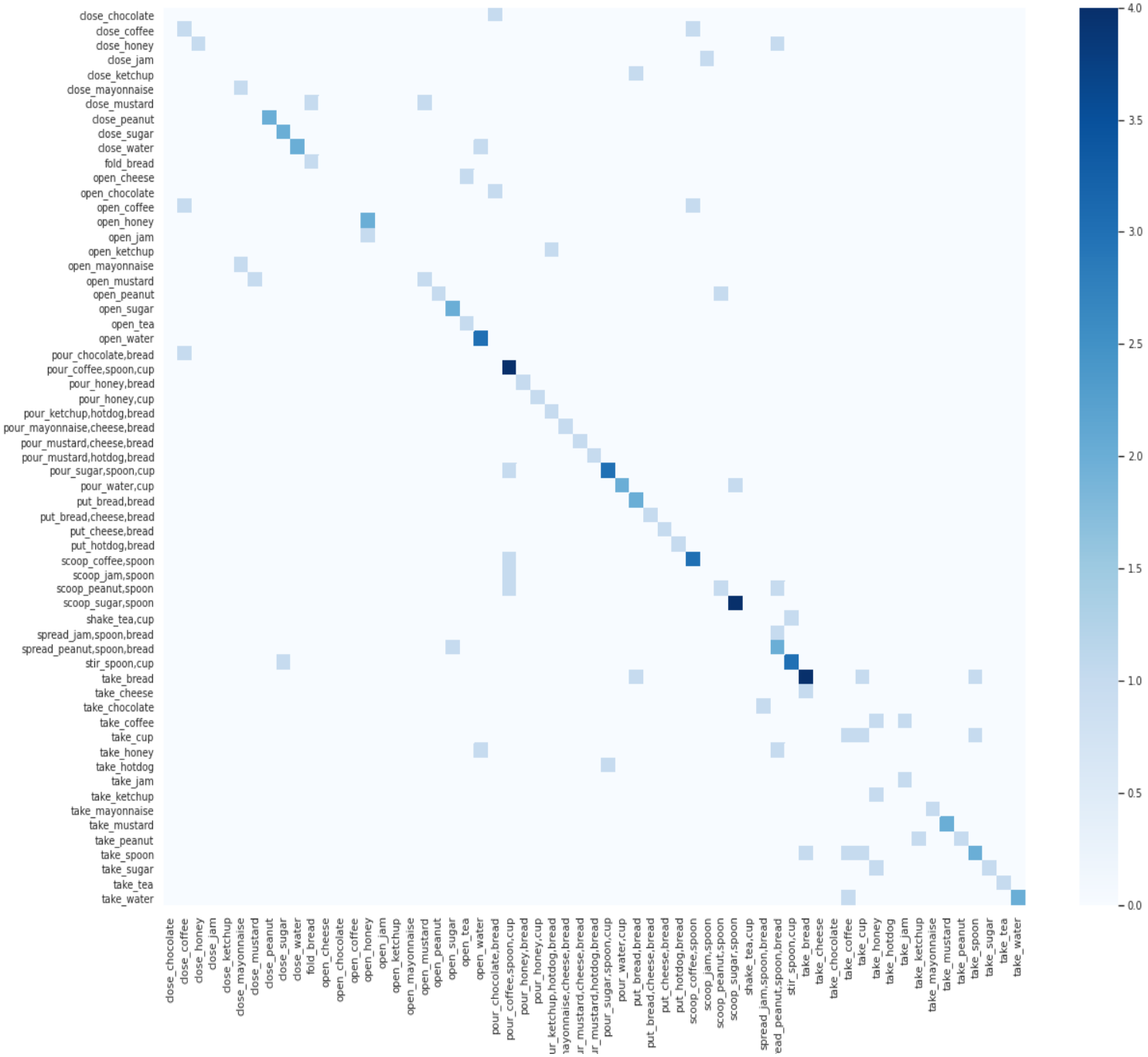
[7] Swathikiran Sudhakaran and Oswald Lanz. *Attention is All We Need: Nailing Down Object-centric Attention for Egocentric Activity Recognition*. 2018. arXiv: 1807.11794.

[8] Heng Wang and Cordelia Schmid. "Action Recognition with Improved Trajectories". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Dec. 2013.

[9] Jiaojiao Zhao and Cees G. M. Snoek. "Dance With Flow: Two-In-One Stream Action Detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[10] Yi Zhu et al. *A Comprehensive Study of Deep Video Action Recognition*. 2020. arXiv: 2012.06567 [cs.CV].

A. Confusion Matrix Motion Segmentation



B. Confusion Matrix Variation