First Person Action Recognition

Antonio Dimitris Defonte
s276033
s276033@studenti.polito.it

Filippo Cortese
s273672
s273672@studenti.polito.it

Davide Bussone
s276486
s276486@studenti.polito.it

Abstract

This report focuses on the first person action recognition problem. This kind of task is more challenging compared to the third person action recognition, since the movements caused by the wearer's camera and the absence of information about the actor's pose are critical issues. The proposed analysis embraces approaches ranging from a two-stream based architecture to a single stream network, performing a self-supervised task, passing through some variations involving different attention mechanisms, different self-supervised tasks and cross modality attention models.

1. Introduction

Being able to recognize actions and activities from videos is one of the most challenging yet well researched tasks in computer vision and its related fields. Applications range from security, behavioural analysis and surveillance to autonomous driving, human computer interaction, automatic indexing and retrieval of media content. Traditionally, researchers have been more focused on action recognition from third person view, but these techniques are less suitable for first person action recognition. In this last setting, wearable cameras are usually head mounted and, as a first issue, we need to address the egomotion that may or may not be representative of the performed action. Other relevant challenges include the lack of information about the pose of the performer, the difficulty to understand the object related to the action for the occlusion of the hand or arm and more in general the scarcity of motion information available about the action itself. These challenges are even more relevant if, instead of classifying a general motion pattern (action recognition) like ”take”, ”put”, ”eat”, ”open”, etc., we are are interested in activity recognition, that is identifying both the action and the object involved in it, such as ”open chocolate”, ”pour water”, ”put jam in bread”, etc. This kind of recognition will be performed on the GTEA61 dataset using the Ego-RNN as a starting point. Subsequently, we will move beyond this two-stream network and try another approach with a self supervised auxiliary task. We then present some possible variations related to the before mentioned approaches. As a term of comparison to the Ego-RNN two stream architecture, we implemented a further model, based on a convLSTA block. Regarding the self-supervised task instead, we proposed a variation, whose aim is to exploit warp flow information as the pseudo labels for the pretext task. Finally, we portray a cross-modality attention model, which will be compared to the two stream approaches, with the difference that the motion and spatial information are joined not only at the end of the network, but earlier in a more hierarchical and effective way.

2. Related works

The literature on action recognition in videos is wide. In order to face the main challenges, especially in the setting of first person action recognition, is crucial to take advantage from the combination between different information channels such as appearance and motion. In this section we discuss several deep learning methods to address egocentric activity recognition. Among them, the most suitable for this project can be mainly differentiated between multi-stream approach and self supervised learning.

Multi-stream approach

[14] proposed a two-stream architecture useful to combine the appearance and the warp flow. In addition, it exploits an attention mechanism with the goal of improving object recognition. A two-stream model is also described in [7], the network trained on multi-frame dense optical flow and in trajectory based descriptors [4], is able to achieve good performances. A different approach is stated by [10], TSN network models long-range temporal structures with a segment-based sampling and aggregation module. As highlighted in [16], a successful classification can be obtained by taking into account egocentric cues, such as head movement, hand pose and gaze information. Then, [3] illustrates a 4-stream CNN architecture, which has two spatial RGB-D video streams and two motion streams, originated by the extraction of optical flow from the frames. Remaining in

theme of collecting as more information as possible from videos, in view to feed the network, [13] proposes a CNN that uses improved dense trajectories, warp flow and RGB frames to capture hand pose, head motion and saliency map, respectively. While, [8] argues that RGB and flow information should be concatenated also in the middle layers of the model, by adding some cross modality attention blocks. This is called cross attention mechanism. A quite different method to classify activities is inferred by [1]; the idea is to acquire how the activities change the state of the objects in the environment.

Self supervised learning

There are also different examples of exploitation of self-supervised learning in videos. In general, self-supervised learning is a subset of unsupervised learning methods. It refers to learning method in which the networks are trained with automatic generated labels. [6] designs a self-supervised approach for video representation learning by video pace predictions, such as the recognition of the slow motion. While [12] portrays a self-supervised approach for learning representations and robotics behavior entirely from unlabelled videos, recorded from different points of view.

3. EGO-RNN Implementation details

3.1. The dataset

The proposed approach is evaluated on ”GTEA61” dataset. It contains 7 types of activities (such as ”open”,”close”), each performed by four different subjects. There are 61 actions (like ”open chocolate”,”close jam”,ecc.), after omitting the background action classes. There are 4 different subjects: the first, third and fourth are used as the training set, while the actions performed by the second subject are exploited as both validation and test set. The camera is worn on the head of each subject. Each video of the dataset is split into RGB frames. In addition, the warp flow information is already provided both for axis x and y. Finally, also the motion maps, exploited in section 4, are provided.

3.2. Architecture overview

As proposed in work [14], we implemented a ResNet-34 pretrained on Imagenet for frame appearance feature extraction. This RGB branch is the backbone of the whole architecture and it is completed by a spatial attention layer, which exploits the class activation maps (CAM). For a particular category, the CAM localizes the discriminative image regions used by the CNN to identify that category [2]. The class activation maps have a primary role since first person activities are representative of the objects handled by the subject, for instance they can distinguish between taking a bar of chocolate or of jam by making the network focusing on the image locations where the objects are. Then, each CAM is converted into a probability map, called ”spatial attention” by applying the softmax activation function and, finally, this result is multiplied by the output of the last convolutional layer of the Resnet-34. Up to this point, the architecture does not consider the spatio-temporal evolution of the activities as they progress through time. In order to encode features related to spatio-temporal changes, the network has to be extended with a convolutional long short-term memory network (conv-LSTM). It works in a similar way to the traditional LSTM, but it has as inputs 3D tensors and convolutional layers in the gates instead of fully connected ones in order to capture spatiotemporal correlations. In the final part of the network there is an average pooling, which takes the output of the conv-LSTM, and a fully connected layer for the classification task (the activity recognition).

3.3. Training specifications

The network described in 3.2 is trained following a two-stage procedure as in [14]. In the first stage, only the conv-LSTM block and the final classifier are trained for 200 epochs. The initial learning rate is set to 10^{-3}, decayed by a gamma value equal to 0.1 after 25, 75 and 150 epochs. Adam is selected as model’s optimizer. In the second stage, the last convolutional layers and the fully-connected layer of the Resnet-34 are trained as well as the layers trained in the first stage for 150 epochs. The initial learning rate is set to 10^{-4}, decayed by a factor of 0.1 after 25 and 75 epochs. The first experiment’s goal is to show the contribution brought by the attention mechanism in terms of accuracy. To do so, we trained the network with and without attention. We sampled uniformly in time 7 frames from each video as inputs for the model. A further experiment consisted in sampling 16 frames instead of 7. As tables 1 infer, the accuracy scores with the attention mechanism improve for both 7 and 16 frames.

Another crucial element of a video are the motion changes. Different deep learning techniques use stacked optical flow images as inputs. Since camera motion can degrade the performance of action recognition, we exploited warp flow as in [14]. Warp flow is obtained by subtracting the camera motion to the optical flow, whose aim is to extract the image movement, at each pixel, from spatio-temporal image brightness variation. The flow images are assembled in stacks of 5 in view to be fed to a temporal network, built on Resnet-34’s structure. During the evaluation phase, the scores achieved from the 5 stacks are averaged to obtain the final classification score. The model is trained for 750 epochs, with an initial learning rate set to 10^{-2}, decayed, after 150, 300 and 500 epochs, by a gamma factor of 0.5. Moreover, it is optimized by stochastic gradient descent

Configurations	7 Frames	16 frames
ConvLSTM	47.41%	56.03%
ConvLSTM attention	56.03%	62.06%
Warp flow	47.43%	47.43%
Two-stream joint	59.48%	64.66%

Table 1: Results on the test set

(SGD) with a momentum of 0.9. The results are shown in table 1 and they are coherent with those reached in [14].
Since the spatial and temporal networks are trained separately, we need to combine them. As stated in [14], there are different approaches for their fusion. Our choice was to follow the so called "joint-two stream" approach. We concatenate the outputs of the two, already fit, models and attach a further fully-connected layer on the top. The model is fined-tuned for 250 epochs with a starting learning rate of 10^{-2}, decayed, after each epoch, of a factor of 0.1. As for warp flow, the chosen optimizer is SGD, with a momentum's value equal to 0.9.In table 1 is possible to infer small improvements both with 7 and 16 frames from each video.

4. Self-supervised task

4.1. Classification task

The main weaknesses of the two stream method, proposed in section 3, is that the spatial and motion networks are trained separately and the final predictions of the two streams are merged only at the end of the network. Work [11] proposes to adopt a single stream architecture able to jointly learn features from both appearance and movement information. By doing so, it is possible to intertwine motion and visual features, encoding them in the layers of the backbone, instead of fusing them with a 2-stream approach. This can be realized by adding a self-supervised block that employs a pretext motion segmentation task to interlace spatial and temporal features. Self-supervised learning, as stated in [9], is a subset of unsupervised learning methods. It refers to procedures in which the networks are trained with an auxiliary pretext task in which pseudo-labels are automatically generated based on some data attributes. In our case, the ground-truth labels are the improved dense trajectories, as described in work [4]. The IDTs were already provided in GTEA61 dataset. Their main property is to remove potentially inconsistent matches due to camera motion through a robust homography estimation. These trajectories maps are sampled in 7x7 matrices and each pixel is set to 0 or 1 (black or white dot) according to a threshold, as a classical binary classification problem. Image 1 explains how the motion segmentation task is inserted in network described in subsection 3.2. This is built by a convolutional block, which reduces the number of channels from 512 to 100, a fully connected layer with size 49 followed by a softmax activation function. This block receives the output of the final convolutional layer of the Ego-RNN's model. We then compute both the loss for the backbone branch and the loss for the motion segmentation task as explained below. To optimize the action recognition head, the model has to minimize the cross-entropy loss between the predicted and true labels, as portrayed in equation 1:

$$L_c(x, y) = -\sum_{i=1}^{n} y_i * log(g(x_i)) \tag{1}$$

with $g(x_i)$ being the class probability estimator on x_i, that is the i^{th} set of N-times stacked frames (n is the number of those sets).

The auxiliary motion segmentation task, is trained with a per-pixel cross-entropy loss, following equation 2:

$$L_{ms}(x, m) = -\sum_{i=1}^{n}\sum_{k=1}^{N}\sum_{j=1}^{s^2} m_i^k(j) * log(g(l_i^k(j))) \tag{2}$$

with $m_i^k(j)$ being the ground truth motion map downsampled to a size $s \times s$, then vectorized, $l_i^k(j)$ represents instead the estimated motion map in the i^{th} set of N-times stacked frames at the k^{th} position in the video segment and finally x is the image embedding.

Thus, the overall network is trained by combining the two previous losses with an equal weight, as equation 3 describes:

$$L(x, y, m \mid \theta_M) = \\ L_c(x, y \mid \theta_f, \theta_c) + L_{ms}(x, m \mid \theta_f, \theta_{ms}) \tag{3}$$

with θ_M being equal to $\{\theta_f, \theta_{ms}, \theta_c\}$, which respectively represent the image embedding, the motion segmentation head parameters and the classification space.

The network is trained for a fixed number of epochs equal to 150, with a step size at the 25^{th} and 75^{th} epoch and Adam as optimizer. Alpha's value is set to 1 and represents the weight given to the self-supervised task. This value is not tuned since paper [11] assigns an equal weight to the 2 losses. 7 frames of each video, uniformly sampled in time, are given as input to the network. In tables 2, 3 and 4 hyperparameter tuning is portrayed. For each single value of an hyperparameter, three trials were made and their average is reported in the tables. The same approach is exploited also in all the other experiments. As benchmarks infer, the best set of hyperparameters has a learning rate

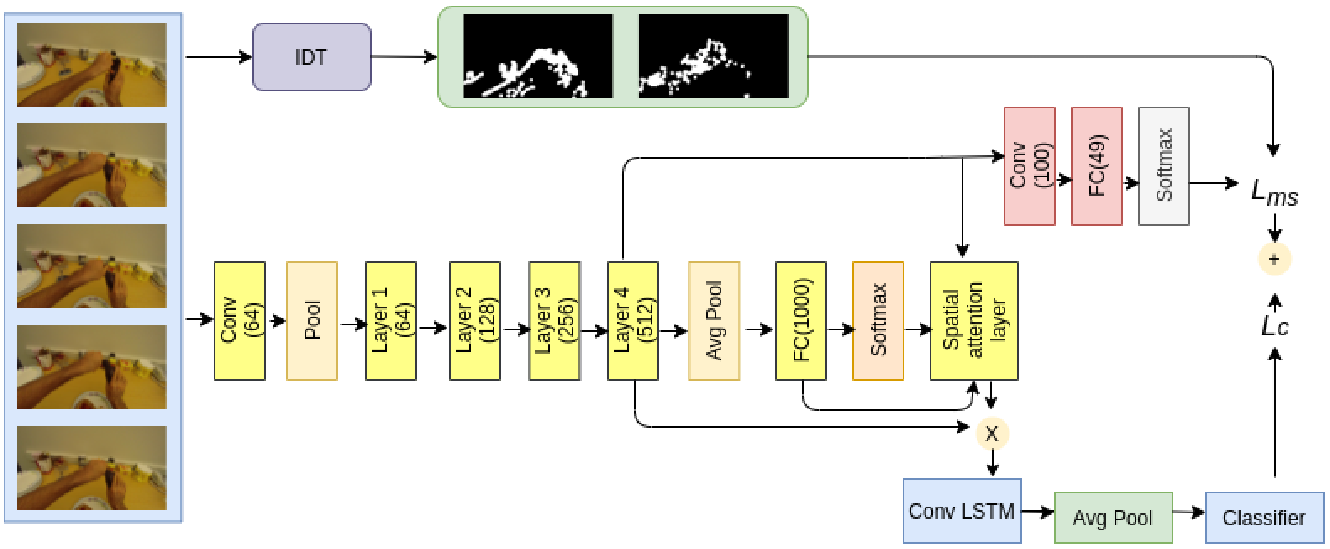

Figure 1: Self supervised task integrated into the second stage of the Ego-RNN RGB network. We can see that this pretext task is backbone agnostic and it could be integrated also in other architectures.

equal to $2 * 10^{-4}$, decayed by a factor of 0.3 and a weight decay of $4 * 10^{-5}$. This model is then evaluated on the test set. The reached accuracy is represented in board 8. As expected, there is an improvement both in terms of accuracy and in computational time compared to the two stream in subsection 3.3.

Learning rate	Validation accuracy
$1 * 10^{-4}$	55.53%
$2 * 10^{-4}$	60.35%
$5 * 10^{-5}$	40.51%

Table 2: Learning rate optimization self-supervised classification

Decay rate	Validation accuracy
0.1	60.35%
0.3	62.35%
0.5	59.32%

Table 3: Decay rate optimization self-supervised classification

4.2. Self-supervised task as a regression problem

As a variation, the self-supervised task is also set as a regression problem. The input images are not only black

Weight decay	Validation accuracy
$4 * 10^{-4}$	56.89%
$4 * 10^{-5}$	62.35%
$4 * 10^{-6}$	56.03%

Table 4: Weight decay optimization self-supervised classification

and white pixels, but also "gray scaled" coloured items. Therefore, regression technique might improve the accuracy scores. The loss function is modified, since cross entropy is not suitable.
The auxiliary motion segmentation task, in this case is trained with a per-pixel mean square error loss (MSE), following equation 4:

$$L_{msr}(x, m) = \sum_{i=1}^{n} \sum_{k=1}^{N} \frac{1}{s^2} \sum_{j=1}^{s^2} \|m_i^k(j) - l_i^k(j)\|_2^2 \quad (4)$$

The overall network is trained by combining the cross-entropy loss for the activity recognition and the previous loss for the motion segmentation task:

$$L(x, y, m \mid \theta_M) = L_c(x, y \mid \theta_f, \theta_c) + L_{msr}(x, m \mid \theta_f, \theta_{ms}) \quad (5)$$

In the code, the motion map prediction's shape is modified to 7x7, otherwise it would be incompatible with the selected loss function. The network is trained for 150 epochs, with

a decaying factor set at the 25^{th} and 75^{th} epoch and again Adam as optimizer. 7 frames of each video, uniformly sampled in time, are taken as input to the network. Then, hyperparameter tuning on the validation is performed by considering again the learning and decay rate and the weight decay. For each hyperparameter, three different training procedures were done and in tables 5,6 and 7 the average scores are portrayed.
The best set of hyperparameters involves a learning rate of $2 * 10^{-4}$, decayed by a gamma factor equal to 0.5 and a weight decay equal to $4 * 10^{-5}$. Finally, the best set is evaluated on the test set. Board 8 conveys that the regression brings a little improvement, in terms of accuracy. As a consequence, a more informative motion map helps the network to better identify the actions.

Learning rate	Validation accuracy
$1 * 10^{-4}$	49.71%
$2 * 10^{-4}$	56.32%
$5 * 10^{-5}$	47.69%

Table 5: Learning rate optimization self-supervised regression

Decay rate	Validation accuracy
0.1	56.32%
0.3	57.75%
0.5	60.05%

Table 6: Decay rate optimization self-supervised regression

Weight decay	Validation accuracy
$4 * 10^{-4}$	58.84%
$4 * 10^{-5}$	60.05%
$4 * 10^{-6}$	59.48%

Table 7: Weight decay optimization self-supervised regression

5. Conv LSTA

From the confusion matrix 9 it is possible to notice that the EGO-RNN architecture struggles to recognize activities involving multiple objects since the per frame generated attention map is independent from that of other frames. Work

Configuration	Test accuracy
Classification task	68.10%
Regression task	69.82%

Table 8: Self supervised task on the test set

[15] proposes a new architecture able to tackle this issue improving the attention mechanism that is able to track previous activated regions. The architecture of LSTA consists of extending the conv-LSTM towards two further components. The first is the recurrent attention, whose focus is on relevant spatial features, by building a weight map. While, the second element is represented by an output pooling which has a high capacity output gate. The network is a ResNet34, pre-trained on imageNet for image recognition. LSTA is fed by the output of the last convolutional layer of the final block of the ResNet. From here, LSTA generates the attention maps. The training phase consists of two stages. In the first one, only the classifier and LSTA modules are fit, while in the second one the convolutional layers in the final block and the fully connected layer of ResNet-34 are trained as well as the layers trained in the first stage. Regarding the motion stream, the network is trained in two phases. We first trained it only on action verbs (”open”, ”close”, ”pour”, ecc...) using a warp flow with stack size equal to 5. Then, the model is trained for activity recognition. Finally, the two-stream architecture is finetuned with a ”joint-two stream” approach as explained in section 3.3. Tables 9 and 10 show the obtained accuracy results. It is possible to infer an improvement in accuracy with respect to section 3. Confusion matrix 10 reports the evaluation, for two stream LSTA architecture, with 16 frames uniformly sampled in time from each video.

Configurations	Ego-RNN	LSTA
RGB attention	56.03%	54.31%
Warp flow	47.43%	42.24%
Two-stream model	59.48%	60.34%

Table 9: Comparison Ego-RNN and LSTA module on the test set with 7 frames

Configurations	EGO-RNN	LSTA
RGB attention	62.06%	63.79%
Warp flow	47.43%	42.24%
Two-stream model	64.66%	68.97%

Table 10: Comparison of EGO-RNN and LSTA networks on the test set with 16 frames

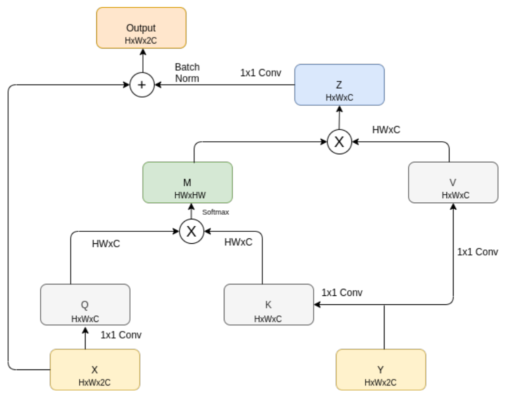

Figure 2: CMA block schema, reshaping or transposing is performed whenever needed.”$\otimes$” denotes matrix multiplication and ”$\oplus$” denotes element wise sum

6. Cross modality attention

By looking at the the confusion matrix 9 of the 2-stream jointly trained model, we can see how some actions are completely misinterpreted by our model in both the object and motion recognition. This could be due to the late fusion of information from the motion and appearance stream that the two-stream model exploit. We tried to improve this by applying a Cross-Modality attention mechanism to our 2-stream model, following the approach in work [8]. In this paper the two stream frameworks fuse their information in a more effective way in an early stage of the network through CMA blocks. The CMA block, showed in 2, receives as input the feature maps of two different modalities X and Y and tries to extract the most critical cross-modality information trough a query from one modality matched with key,value pairs from the other modality. We tried to adapt this approach and we inserted this Cross Modality Block in to our network before the 4^th layer of each modality ResNet. In this way both flow and RGB branch can communicate and embed information from the other stream before generating the spatial attention map. To address the training of this model we started from the two networks that were already trained separately for the two stream task and then we followed the same fine-tuning policy while training also the newly inserted CMA blocks. For GPU memory limitations we had to restrict our analysis to only 7 frames sequence and we fine tuned the net for 100 epochs, with a starting learning rate of 1e-3 that is decayed each epoch by a factor of 0.99. Tables 11 show the obtained accuracy results. There is a great improvement with respect to the simple two stream joint approach with 7 frames. Confusion matrix 7 reports the evaluation, for two stream CMA architecture, with 7 frames uniformly sampled in time from each video.

Two stream Architecture	Test accuracy
Two stream joint	59.48%
Two stream CMA	65.52%

Table 11: Comparison on the test set of proposed two stream architectures

7. Warp flow based self-supervised task

By observing the confusion matrices 5 and 6 obtained with the self-supervised task, sometimes, the network fails in classifying the action, despite the correct recognition of the object (the true label ”Open-honey” is predicted as ”Take-honey”) or vice versa (the true label ”Open-mayonnaise” is predicted as ”Open chocolate” in the standard self supervised and ”Take water” is classified as ”Take jam” in the regression). We still have problems when multiple objects are involved in the activity and sometimes, similar objects are wrongly classified. A way to improve the previous results, could be feeding the model with other motion pseudo-labels. To this aim, [5] uses optical flow as ground-truth of a self-supervised approach, whose goal is to identify the visual appearance of some obstacles in the environment. So, the idea is to exploit the warp flow as pseudo-label for the first person action recognition task. Since warp flow images are in grayscale, the regression seems to be the most suitable choice for network’s implementation. As a consequence, the chosen loss function is the minimum squared error (MSE). The model is fed by 7 input frames of each video, uniformly sampled in time. The network is trained for a number of epochs equal to 150, the learning rate is decayed at the 25^{th} and 75^{th} epoch and the selected optimizer is Adam. Then, the tuning is performed by considering the same hyperparameters of section 4. In tables 12, 13 and 14 the average accuracy scores are portrayed. The best set has a learning rate of $2e^{-4}$, decayed by a factor of 0.3, with a weight decay of $4 * 10^{-5}$. Finally, the best model is evaluated on the test set and board 15 underlines a small improvement in terms of accuracy. It can be concluded that warp flow seems to be a better motion map for this task, but we have to consider that the reason could be linked to the fact that GTEA61 is a small dataset.

Learning rate	Validation accuracy
$1 * 10^{-4}$	58.85%
$2 * 10^{-4}$	66.63%
$5 * 10^{-5}$	61.97%

Table 12: Learning rate optimization self-supervised flow

Decay rate	Validation accuracy
0.1	66.63%
0.3	70.30%
0.5	69.27%

Table 13: Decay rate optimization self-supervised flow

Weight decay	Validation accuracy
$4 * 10^{-4}$	69.82%
$4 * 10^{-5}$	70.30%
$4 * 10^{-6}$	64.83%

Table 14: Weight decay optimization self-supervised flow

Configuration	Test accuracy
Classification task	68.10%
Regression task	69.82%
Warp flow based	71.55 %

Table 15: Comparison on the test set of all proposed self supervised tasks

8. Results

From the previous experiments, it is possible to see that with self supervised learning we can gain in general better results rather than the two-stream approaches. In particular, tables 8 and 10 show that, even with only 7 frames, the networks with the self supervised task obtain a higher accuracy than both the EGO-RNN and LSTA (with respect to the self supervising with regression) with 16 frames. Thus, we were also able to reduce the training time by using less frames and without having to jointly fine tune two different streams. The reason behind this is that the self supervised task forces the backbone network to learn features that, in a single stream, consider both appearance and motion information by updating a single loss made up of two contributions as explained in section 4. By seeing the confusion matrices, we discovered that both the EGO-RNN and self supervised network struggle with the recognition of activities regarding multiple objects. As, work [15] states, the architecture of the LSTA should be able to tackle this issue, because of its enhanced attention mechanism able to track previously activated regions. But, except for the accuracy, we did not notice any improvement in this terms from our confusion matrices. This is an issues regarding all the architectures that we have trained, but we must need to say that this could be an issue of the dataset since it is relatively small compared to other ones that are used for this task and also because of the limited amount of frames that we used (7 and 16) instead of the 25 employed in the original papers [14] and [15]. The introduction of a Cross Modality Attention mechanism enabled our 2 stream approach to outperform its simpler joint version but this model still lacks behind the other solution proposed in terms of complexity and time requested for training. The last approach consisted in exploiting the warp flow as pseudo-labels for the motion segmentation task. In section 7 we presented the main issues we had at that point. This model does not significantly address them, but, nonetheless, provides an overall better accuracy. From these results, we can infer that the self supervised task provides a handy tool for activity recognition, enabling the network to learn the motion together with the appearance in a single stream and, at the same time, reducing the training time.

9. Conclusions

In this report are portrayed different architectures and training procedures for first-person action recognition. After a detailed analysis, it can be inferred that, in the studied setting and with the provided dataset, the best performing architectures were those including the self supervised task. Even if the cross-modality attention and LSTA were able to reach an accuracy comparable to that of the self supervised one, they are much more complex and require more training time. There are still open issues that this project were not able to tackle, therefore, further improvements are possible such as different motion segmentation self supervised tasks or different network architectures. Different backbone architectures (for the self supervised task), such as the one proposed in paper [11], are able to reach a much higher accuracy on the same dataset. Regarding the CMA, it could be interesting for future works to implement the version proposed in [8] that lacks the LSTM cell, but exploits more CMA blocks across the ResNets, and see how it performs on egocentric action recognition. Moreover, in practical applications (such as video surveillance, embedded systems, smartphone applications), the speed, memory, inference time of the models are fundamental issues. So, another challenge might be to choose the right trade-off between the portability of the models and their learning capabilities.

10. Repository

```
https://github.com/RoboTuan/FPAR.git
```

References

[1] A.Fathi and J.M.Rehg. Modeling actions through state changes.

[2] B.Zhou, A.Khosla, A.Lapedriza, A.Oliva, and A.Torralba. Learning deep features for discriminative localization https://arxiv.org/abs/1512.04150v1.

[3] D.Shirari, P.V.V.Kishore, E.K.Kumar, D.A.Kumar, M.T.K.Kumar, M.V.D.Prasad, and Ch.R.Prasad. A four-stream convnet based on spatial and depth flow for human action classification using rgb-d data.

[4] H.Wang and C.Schmid. Action recognition with improved trajectories.

[5] H.W.Ho, C. Wagter, B.D.W.Remes, and G. Croon. Optical flow based self-supervised learning of obstacle appearance applied to mav landing.

[6] J.Wang, J.Jiao, and Y.Liu. Self-supervised video representation learning by pace prediction.

[7] K.Simonyan and A.Zisserman. Two-stream convolutional networks for action recognition in videos.

[8] L.Chi, G.Tian, Y.Mu, and Q.Tian. Two-stream video classification with cross-modality attention.

[9] L.Jing and Y.Tian. Self-supervised visual feature learning with deep neural networks: a survey.

[10] L.Wang, Y.Xiong, Z.Wang, Y.Qiao, D.Lin, X.Tang, and L. Gool. Temporal segment networks for action recognition in videos.

[11] M.Planamente, B.Caputo, and A.Bottino. Joint encoding of appearance and motion features with self-supervision for first person action recognition.

[12] P.Sermanet, C.Lynch, Y.Chebotar, J.Hsu, E.Jang, S.Schaal, and S.Levine. Time-constrastive networks: self-supervised learning from video.

[13] S.Singh, C.Arora, and C.V.Jawahar. First person action recognition using deep learned descriptors.

[14] S.Sudhakaran and O.Lanz. Attention is all we need: Nailing down object-centric attention for egocentric activity recognition.

[15] O. S.Sudhakaran, S.Escalera. Lsta: Long short-term attention for egocentric action recognition.

[16] Y.Li, Z.Ye, and J. Rehg. Delving into egocentric actions.

Appendix

In this section there are the confusion matrices related to some different experiments as well as the CAMs of the models in figure 3.

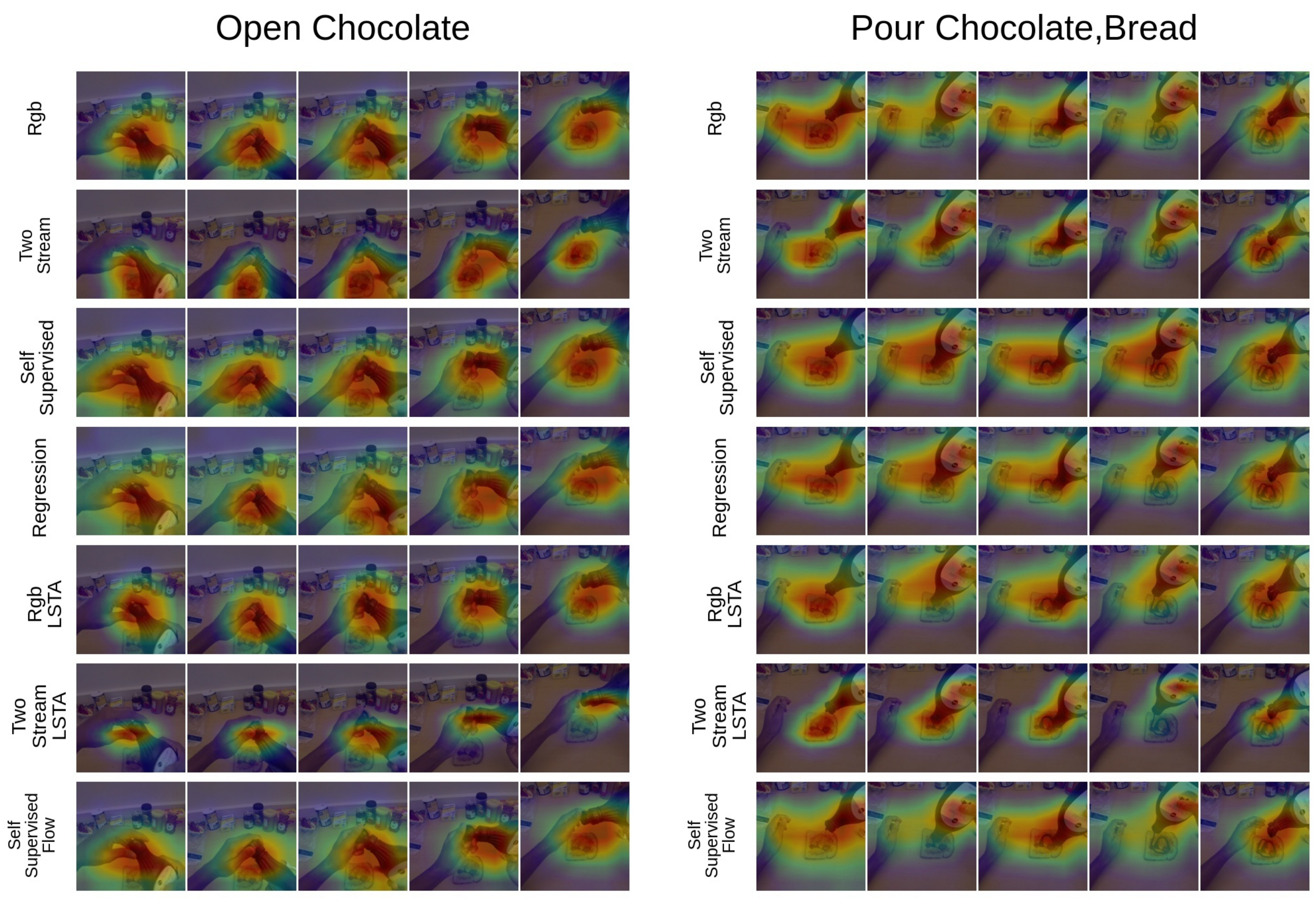

Figure 3: Visualization of the CAMs of the different networks. For each model, we can see that it focuses its ”attention” on what it thinks the most relevant regions of the image are. The activities considered are ”open-chocolate” and ”pour-chocolate,bread”, particularly interesting because the ”two-stream” model tends to confuse them.

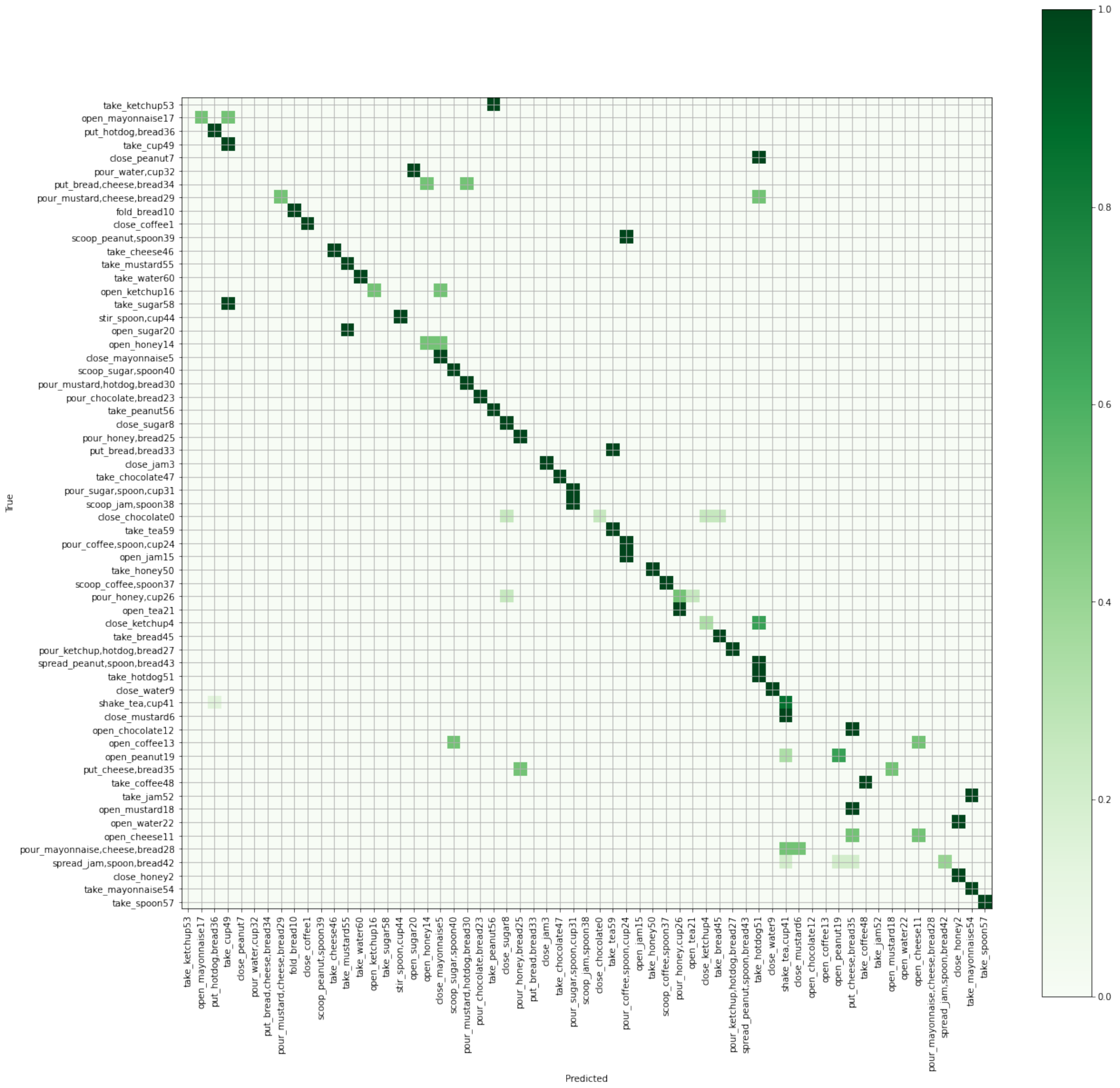

Figure 4: Joint two-stream confusion matrix with 7 frames

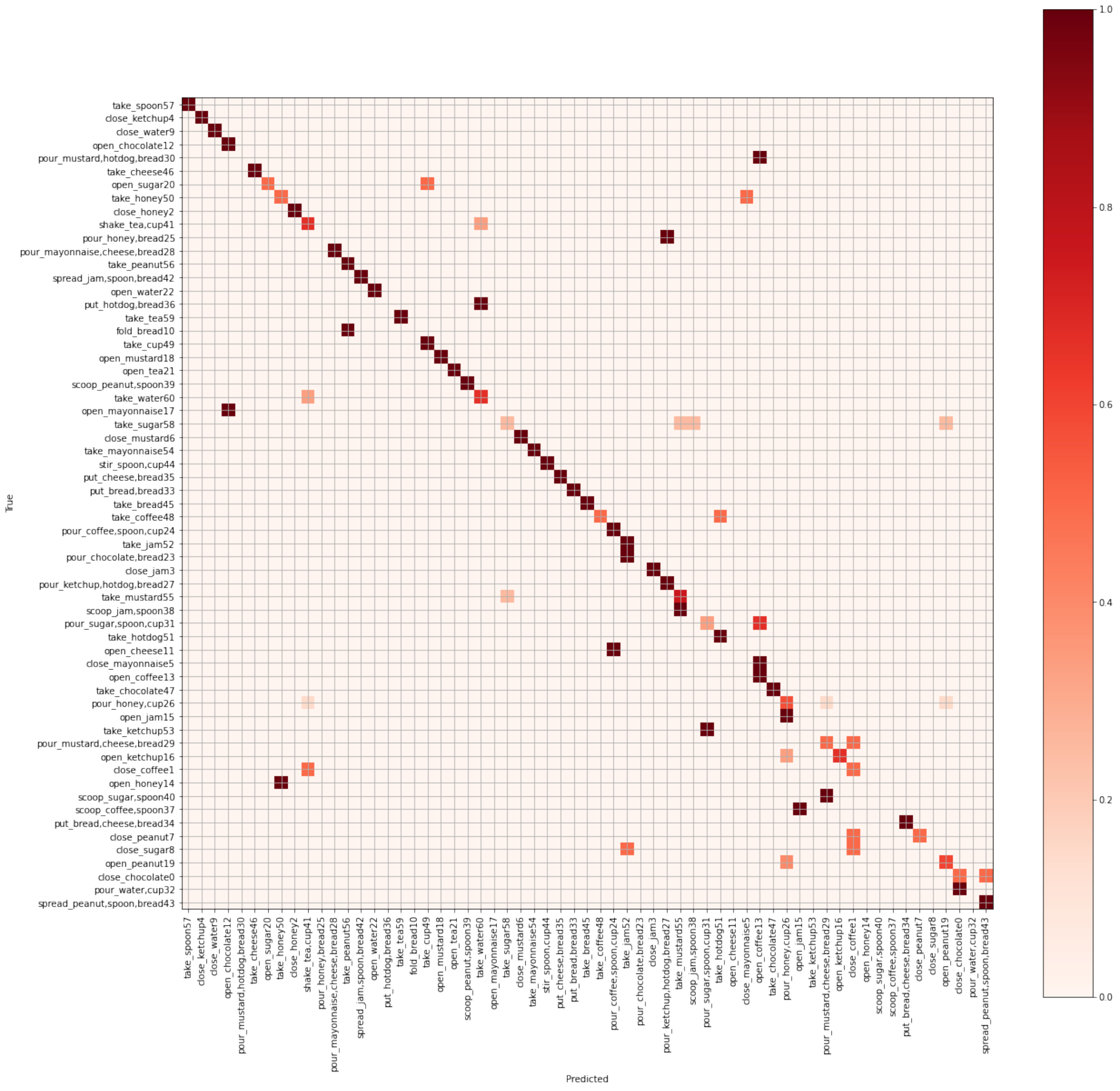

Figure 5: Self-supervised as a classification task confusion matrix

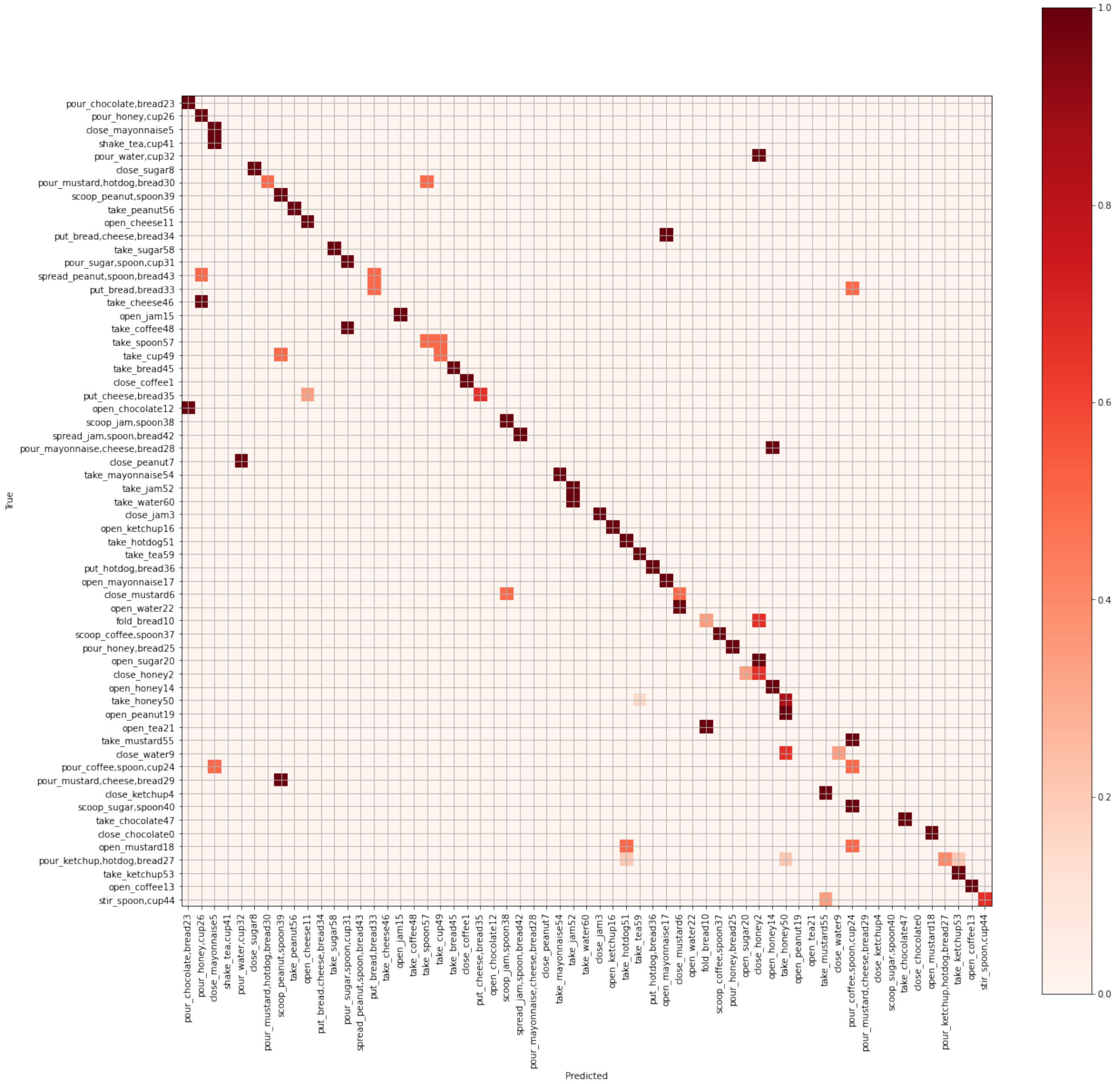

Figure 6: Self-supervised as a regression problem confusion matrix

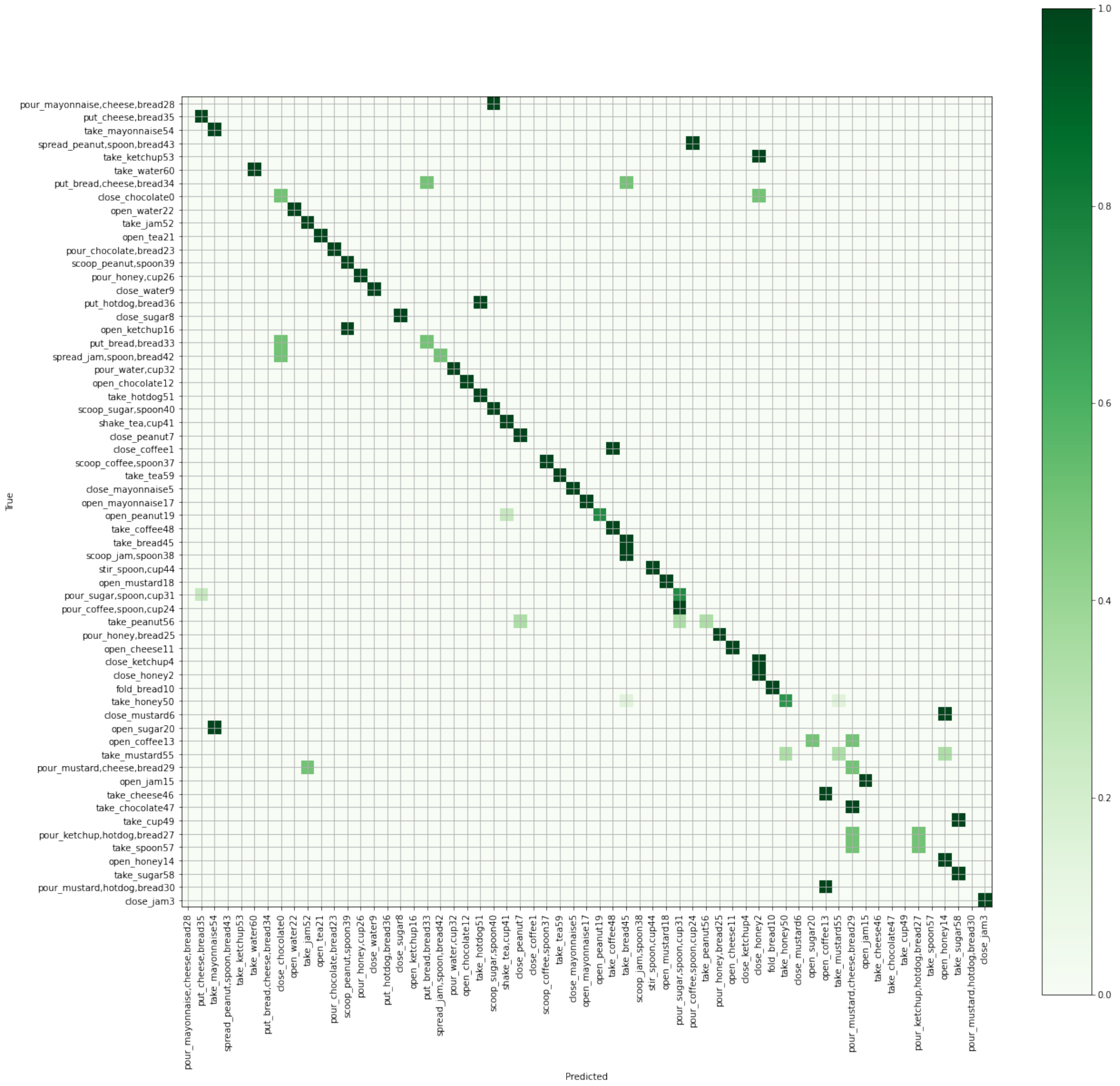

Figure 7: Cross modality attention confusion matrix for 7 frames

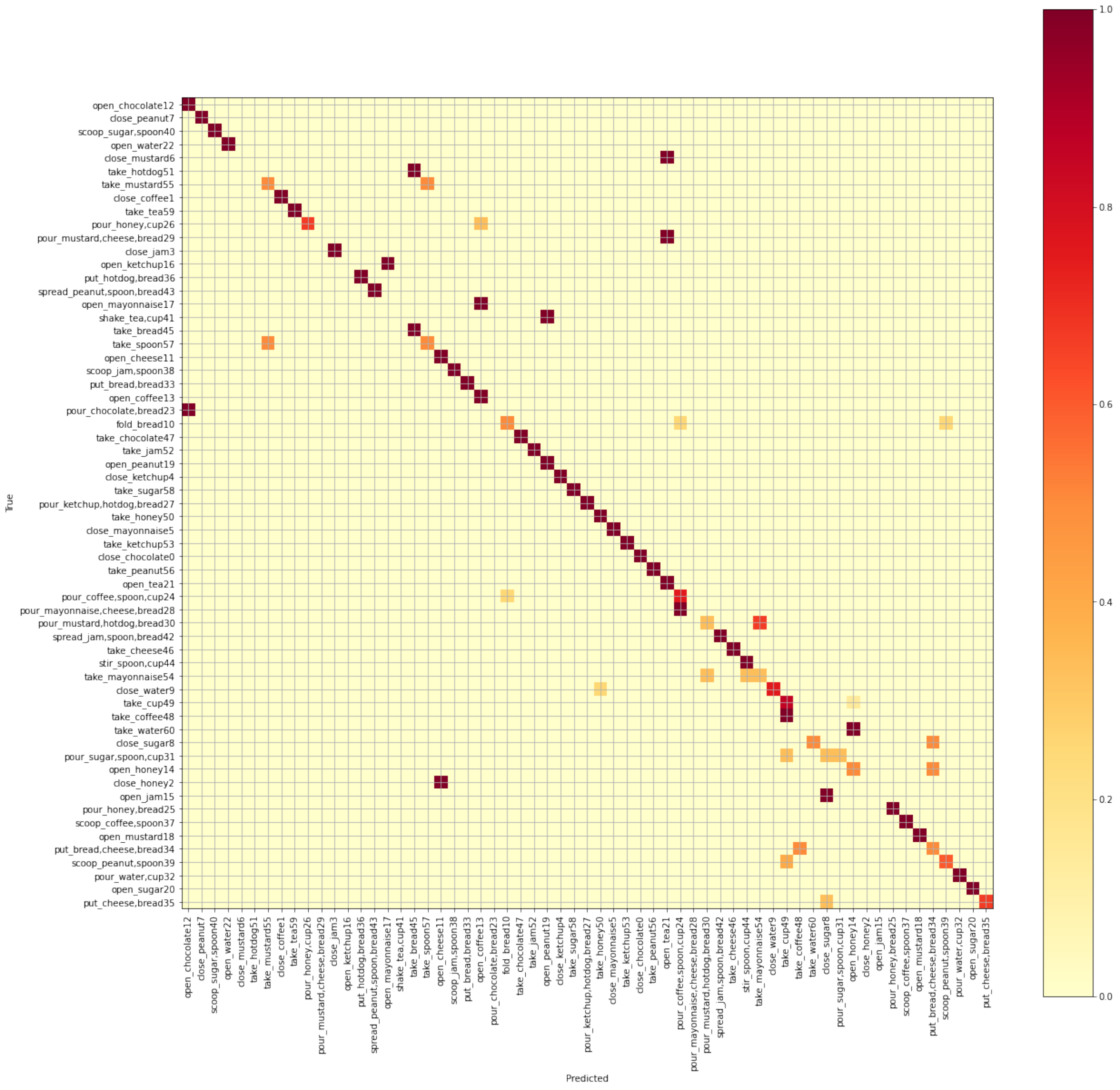

Figure 8: Warp flow based self-supervised task confusion matrix

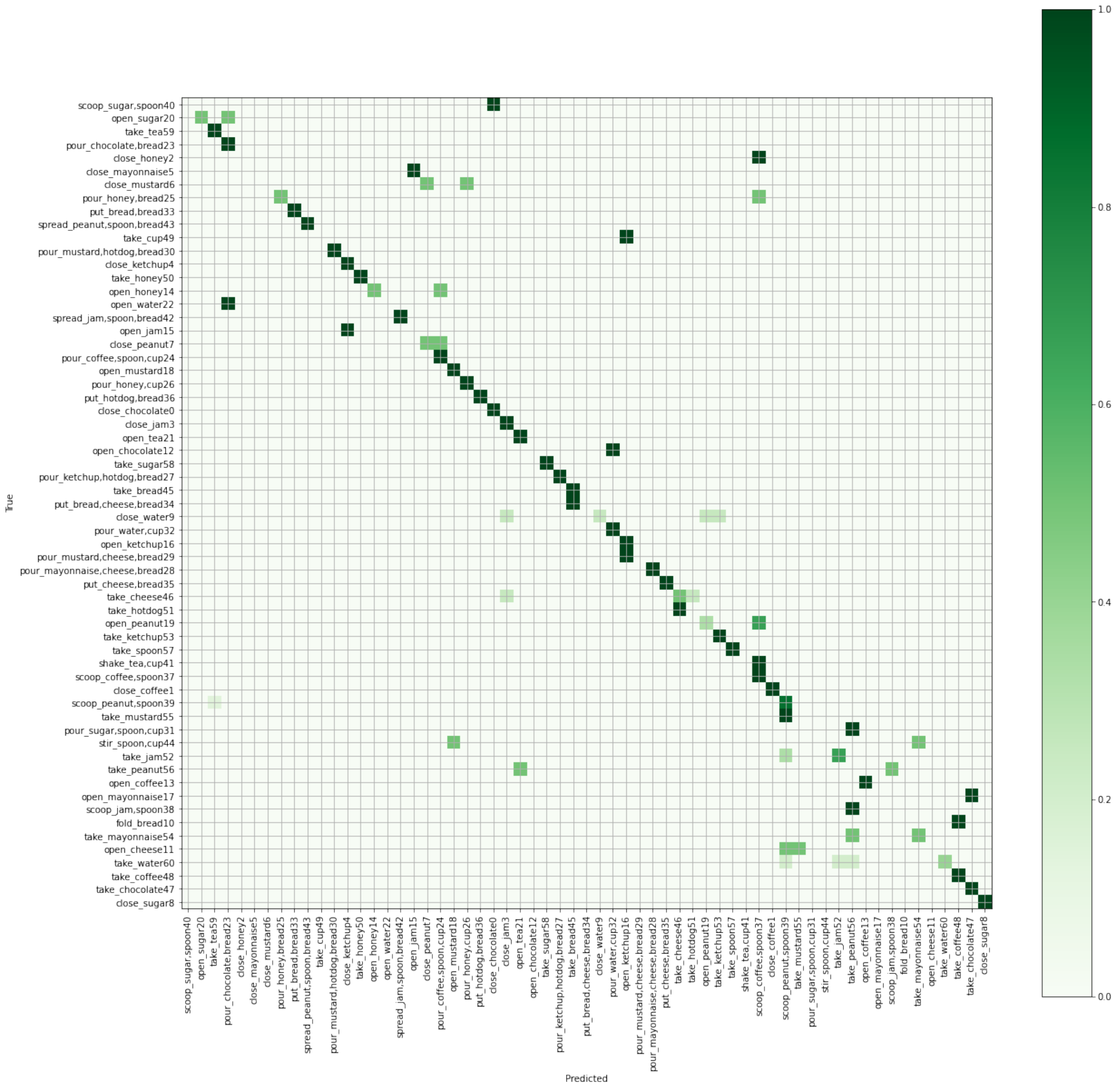

Figure 9: Joint two-stream confusion matrix with 16 frames

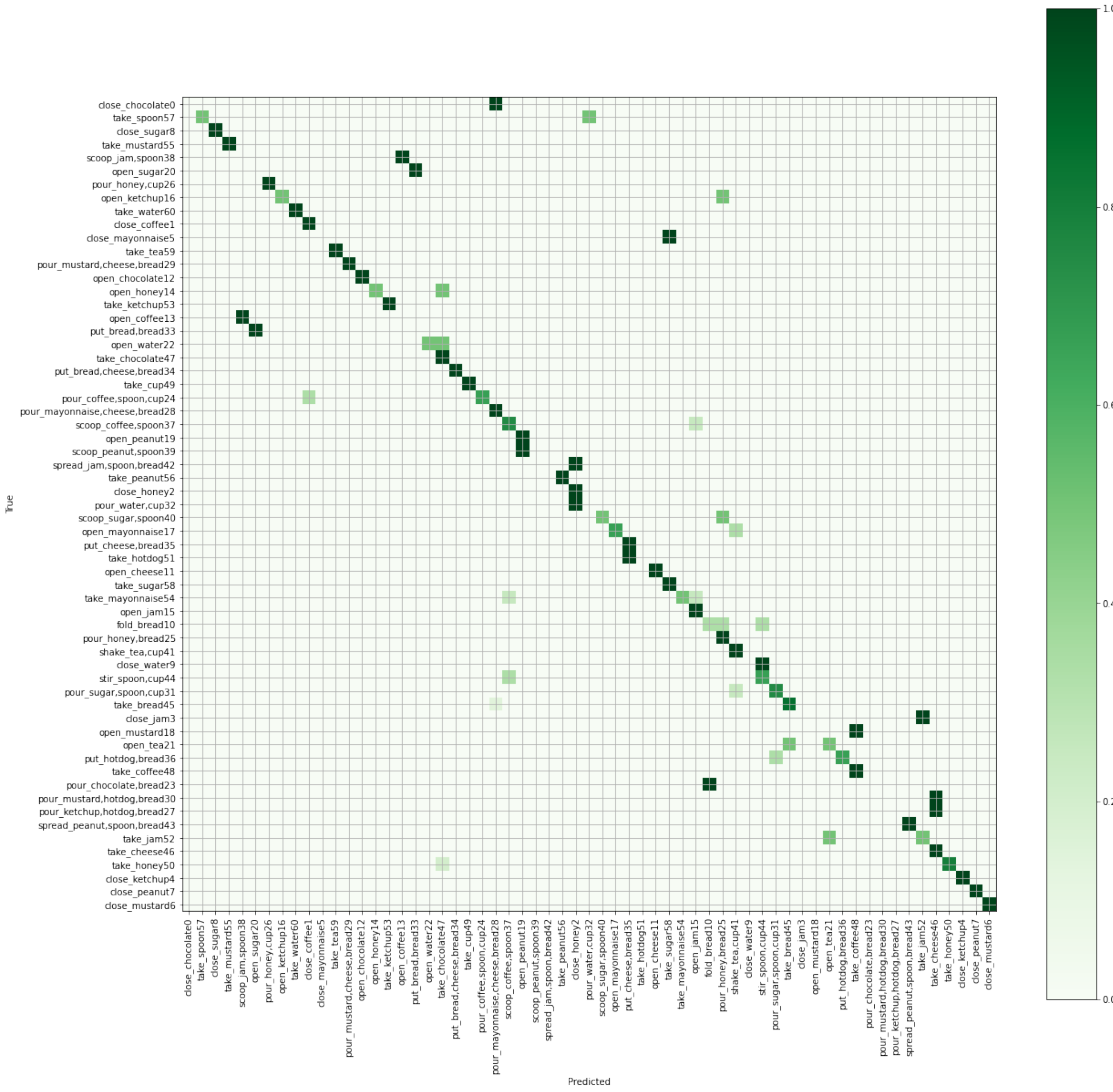

Figure 10: LSTA model confusion matrix for 16 frames