Domain Adaptation for Egocentric Action Recognition

Fantasia Claudio
Polytechnic of Turin
Corso Duca degli Abruzzi, 24, Turin, Italy
claudio.fantasia@studenti.polito.it

Gatto Raul
Polytechnic of Turin
Corso Duca degli Abruzzi, 24, Turin, Italy
raul.gatto@studenti.polito.it

Lanzillotti Donato
Polytechnic of Turin
Corso Duca degli Abruzzi, 24, Turin, Italy
donato.lanzillotti@studenti.polito.it

Abstract

Egocentric action recognition is one of the most dynamic research areas within computer vision and artificial intelligence; its increasing interest arises from its wide-ranging potential applications. The generalization across different domains therefore represents one of the main obstacles. In this paper we address the problem of Video Domain Adaptation and investigate the Temporal Attentive Adversarial Adaptation Network (TA3N) performance, taking as input RGB features extracted with an Inflated 3D ConvNet (I3D). The problems of Domain Shift will be faced trying to capture the temporal dependencies and align features across the time dimension. We also explore the improvements obtained adding the Minimum Class Confusion Loss (MCC). For this study, the Epic Kitchen Dataset was used, evaluating the performances of our tests through the top1-accuracy. The results show small improvements depending on domain adaptation techniques used, and thanks to the ablation study we can delve deeper into the functionality of each one. The code can be found at this link.

1. Introduction

Egocentric Action Recognition has increasingly gained importance in the last years in the landscape of computer vision. It aims at recognizing first-person actions recorded by any sort of wearable gadget, from the person's point of view. This first-person viewpoint creates new advantages and challenges compared to the usual third-person perspective: it captures all of the nuances of the interaction between the wearer and the immediate environment, usually lost with the traditional method. These details can vary from the obvious hand movement of a person who is cooking to the subtle gaze shift when scrolling through the phone.

The application of Egocentric Action Recognition are numerous, such as augmented reality, human-computer interaction, healthcare and many more, hence the growing interest in the field and the necessity of new and better solutions to solve the problems related to these fields. Initially, they were tackled with neural networks that worked frame-by-frame, but it is now the norm to consider the time as an integral part of the problem, making it a video-based approach.

In this area of interest, **Domain adaptation (DA)** tech-

niques have been studied extensively in recent years to address the *Domain Shift* problem. It refers to a scenario where the statistical properties of the data change between different domains and consequently the models trained on a source labeled dataset do not generalize well on the target dataset [3]. Figure 1 depicts domain shift and DA with a simple example.

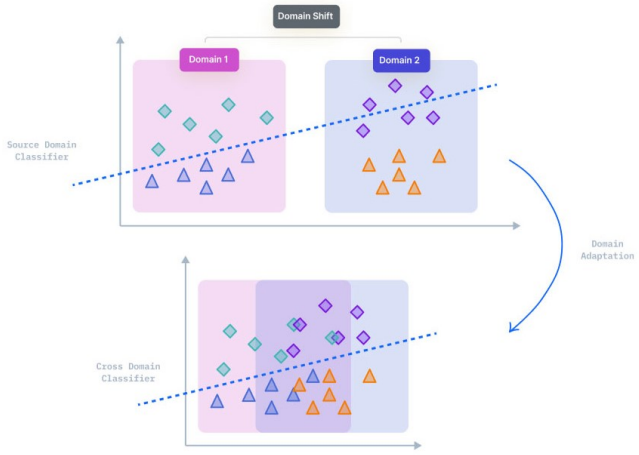

Figure 1. Domain Shift and Domain Adaptation example.

The analysis of the effectiveness of different DA approaches is performed on the **Epic Kitchens Dataset**. It is a large-scale egocentric video benchmark recorded by 32 participants in their native kitchen environments. Participants from 10 different nationalities contributed to the recordings, which were conducted in 4 cities across North America and Europe. This diverse composition led to a rich variety of cooking styles captured in the dataset [4].

In this paper we want to face the problem of domain shift previously introduced, with techniques of Domain Adaptation for Egocentric Action Recognition considering only 3 kitchens out of the whole Epic Kitchens Dataset.
Our contributions will be deeply examined in the following sections and can be summarized as follows:

- A *dense sampling* approach to feed the I3D feature extractor [2];
- Through the ablation study, showing the positive impact of *Adversarial Modules* [3];
- Demonstrating the improvement led by the introduction of *MCC loss*, justified by an accurate analysis of results' logs [7].

2. Related works

2.1. Video Classification

In recent years, video classification tasks have received notable attention in the field of Deep Learning, leading to many improvements in their results.
Deep **Convolutional Neural Networks (CNNs)** have emerged as a dominant architecture due to their ability to automatically learn hierarchical features directly from raw pixel data. CNNs excel at capturing spatial hierarchies present in video frames. In their early layers, they acquire basic features like edges and textures, gradually advancing to learn complex patterns such as object parts and shapes. At the beginning CNNs were able to capture only spatial features (*2D CNNs*) making them not adequate for video data, since the temporal information was completely not considered. In other words, we could process the original video or the same video with switched frames and we will end up with the same results. For this reason a new version of CNNs was introduced: *3D CNNs*, which, unlike the previous architecture, were able to exploit the temporal features [2]. This led to having high number of parameters, which increased the computational cost as well. The introduction of 3D CNNs raised problems that required to be solved.
At the same time another type of architecture gained popularity, the so called **Recurrent Neural Networks (RNNs)**. RNNs excel at modeling sequential data and capturing temporal dependencies in videos more effectively than CNNs. They are computationally more demanding than CNNs due to their sequential nature and it can lead to slower training and inference times, impacting real-time applications [5].
Shifting the attention from the architecture to the input format led to **multi-modal learning** which is a method where multiple types of data (e.g. images, text, sound) are integrated and processed simultaneously. In this context, the importance of audio features is investigated in [8].

2.2. Video Domain Adaptation

In the world of Machine Learning and Deep Learning, having a model capable of adapting to different scenarios and with a high level of generalization has always been a fundamental need, especially in real-world cases. Therefore, the problem of Domain Shift and consequently Domain Adaptation (DA) techniques have been deeply investigated in the last years.
With 'Domain Adaptation' we refer to all the processes whose goal is mitigating the effects of the differences among data distributions between the source domain and the target domain, in order to have a more general model. It is possible to distinguish Domain Adaptation techniques in Supervised and Unsupervised.
Supervised Domain Adaptation (SDA) involves using

class-labeled data from the target domain to align the distributions, while **Unsupervised Domain Adaptation (UDA)** primarily concentrates on distributions alignment using exclusively unlabeled data from the target domain. Notice how both the techniques are aware of the domain from which the data come from. Later in this paper the UDA approach will be investigated more. Furthermore, in the unsupervised setting different methods tackle the challenge in different ways. Some of them try to bridge the domain gap, minimizing the distributional distance computed using different metrics such as the *Maximum Mean Discrepancy (MMD)* [11] or the *Kullback-Leibler divergence (KL divergence)* [9].

Other techniques instead are based on the **Adversarial framework**, with a mechanism similar to the **Generative Adversarial Networks (GANs)** [6]. In this case, a feature extractor competes with a domain classifier trying to fool it in its task. Progressively, this competition will lead the feature extractor to generate features indistinguishable between the source and the target domain.

Another method is the *Transfer Learning* which instead leverages pre-trained models on the source domain and fine-tunes them on the target domain [1]. All the previous methods can be used both in the case where source label space and target label space fully overlap or not. Regarding the latter, *Partial Video Domain Adaptation (PVDA)* has been studied specifically for the case where source label space subsumes the target label space. The key challenge of PVDA is the issue of negative transfer caused by source-only classes [11].

3. Technical Approach

The model examined in this paper is based on the **Temporal Attentive Adversarial Adaptation Network** (**TA3N**) (Figure 2) proposed in [3] and to make the analysis more clear in the next sections, an overview of the entire architecture will be provided. It can be partitioned into different sections accordingly to the different level of data aggregation, each one with a different purpose and able to exploit both spatial and temporal information. In the whole net, there are two *action classifiers* (G_{sy}, G_{vy}) and three *domain classifiers* (G_{sd}, G_{rd}, G_{vd}) that contribute with different weights to the overall loss function of the network. It's worth noting that the feature extractor doesn't converge to a constant solution, which would deceive the discriminator. Instead, it will converge to a non-trivial solution thanks to the losses from the action classifiers.

3.1. I3D

The starting point are the raw videos and the initial phase is to extract the features by using the **I3D network** [2]. Every video of the dataset corresponds to an action and consists of RGB frames. Through our sampling method, 5 short clips of 16 frames are sampled and features are extracted from each clip. The features extraction process is performed by I3D, using a CNN with 3D filters.
It is important to underline that differently from what is proposed in [2], the features are not extracted from a single frame but from a set of them and this explain why 3D filters are used instead of 2D ones: to consider the temporal dimension as well.

The output of I3D represents the input for the Spatial Module G_{sf} which consists of a *Multilayer Perceptron (MLP)* that converts the general-purpose feature vectors into task-driven feature vectors, where the task considered is video classification [3].

3.2. Temporal Aggregation

Once the features are extracted, they enter the *Temporal Module* G_{tf} that aggregates the clip-level feature vectors to form a single video-level feature vector for each video. In this paper the temporal aggregation is investigated using two different techniques:

- **Average Pooling:** A basic pooling operator is applied to the temporal dimension, computing the average value of each feature across various clips. The outcome can be seen as video features that include both spatial and temporal aggregation.

- **Temporal Relation Network (TRN):** Thanks to this module, it is possible to capture both long and short dependencies between different clips, improving the video classification performance. A simple representation can be seen in Figure 3.
  TRN can be applied at different time scales, obtaining a *multi-scale temporal relation* as showed in [13]. For instance, the pairwise temporal relation is defined as follows:

$$T_2(V) = h_\Phi(\sum_{i<j} g_\Theta(c_i, c_j)) \tag{1}$$

  where c_i and c_j represent the clips. The functions h_Φ and g_Θ are multilayer perceptrons used to fuse features of different ordered clips. This concept can be extended considering different number of clips.
  In our analysis, $2-$clips, $3-$clips, 4-clips, and $5-$clips temporal relations are considered obtaining the following aggregated result:

$$MT_5(V) = T_2(V) + T_3(V) + T_4(V) + T_5(V) \tag{2}$$

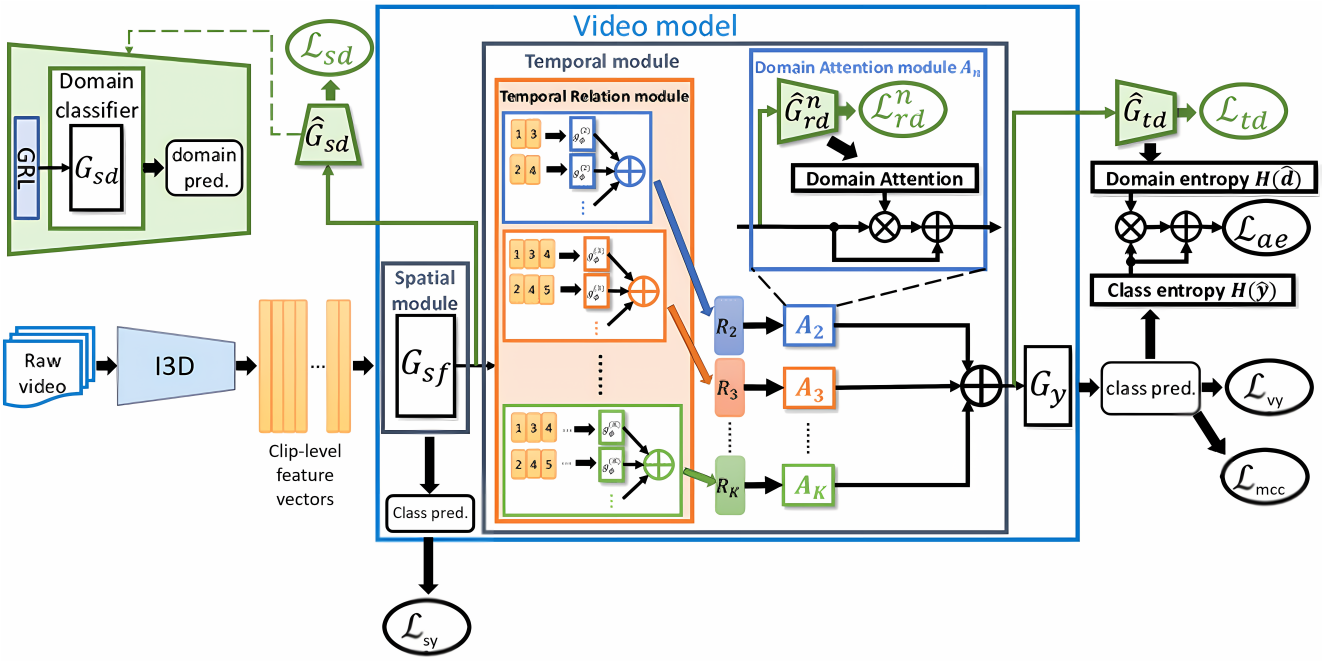

Figure 2. Temporal Attentive Adversarial Adaptation Network Architecture.

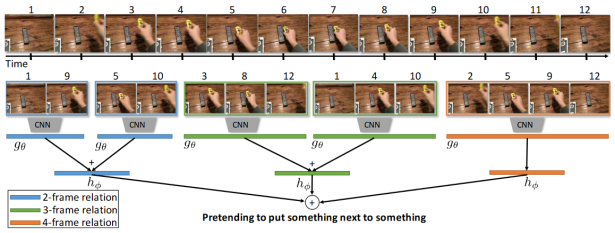

Figure 3. Illustration of Temporal Relation Networks. Representative frames of a video are sampled and fed into different frame relation modules.

To better understand the final results it is fundamental to highlight that the TRN module involves more complex operations than the Average Pooling leading to a richer set of information that can be exploited from the entire architecture. For this reason, better performance is expected when this second approach is used.

3.3. Domain Adaptation

Domain Adaptation represents the main point of our research study and it is applied indiscriminately at each level of aggregation. Among the different techniques presented in the paragraph Video Domain Adaptation (2.2), the *Adversarial framework* has been chosen [12]. It implies the introduction of three domain classifiers (G_{sd}, G_{rd}, G_{vd}) that compete in an adversarial manner with the G_{sf}. In other words, during the training phase while the discriminators' task is to distinguish whether the video is from the source or target domain, the G_{sf} acts trying to fool them. This competition leads to ignoring features that are connected with a specific domain.

The three domain classifiers that act as domain discriminators are:

- G_{sd} : it operates at clip level, using the output features from the G_{sf} and before the temporal aggregation step.
- G_{rd} : it operates at relational level, using the output features from the TRN module and before the final temporal aggregation step seen in equation Eq. (2). There is one single domain discriminator for each $n-$clips temporal relation which implies having four G_{rd} in our case, where G^n_{rd} corresponds to the discriminator associated with the $n-$clips relation $T_n(V)$.
- G_{vd} : it operates at video level, using the output features from the G_{tf} and after the final temporal aggregation step.

The three domain discriminators operate in the same way but at different level of aggregation.
To train the discriminators it is necessary to introduce three loss terms in the overall loss function of the network:

$$\mathcal{L}^i_{sd} = \frac{1}{K}\sum_{j=1}^{K} L_d(G_{sd}(G_{sf}(c^j_i)), d_i) \quad (3)$$

$$\mathcal{L}^i_{rd} = \frac{1}{K-1}\sum_{n=2}^{K} L_d(G^n_{rd}(T_n(G_{sf}(X_i))), d_i) \quad (4)$$

$$\mathcal{L}_{td}^{i} = L_d(G_{td}(G_{tf}(G_{sf}(X_i))), d_i) \tag{5}$$

where K is the number of clips from each video, L_d is the cross entropy loss function and d_i is the domain label associated to a specific video [3].

3.3.1 Gradient Reverse Layer (GRL)

In each discriminator previously presented, during the back-propagation phase a Gradient Reversal Layer (GRL) is introduced.
The GRL is a component whose primary purpose is encouraging a neural network to learn *domain-invariant* features during training. It modifies the gradients by multiplying them by a constant factor of $-\lambda$. In this way, the domain classifier is faced with features that are intentionally made less informative about the domain, encouraging the feature extractor to generate domain-invariant features.

3.4. Action Classification

The performance of the network proposed measures its capability to correctly classify actions, for this reason two action classifiers are added. As in the case of domain classifiers, they act at different level of aggregation:

- G_{sy}: its classification task takes place before the temporal aggregation, thus at clip level.
- G_{vy}: its classification task takes place after the temporal aggregation, thus at video level.

Two new terms are added to the overall loss function of the network:

$$\mathcal{L}_{sy}^{i} = \frac{1}{K}\sum_{i=1}^{K} L_y(G_{sy}(G_{sf}(c_i^j)), y_i) \tag{6}$$

$$\mathcal{L}_{vy}^{i} = L_y(G_{ty}(G_{tf}(G_{sf}(X_i))), y_i) \tag{7}$$

where L_y is a *cross-entropy loss*, c_i^j are the features corresponding to the $j-$th clip of the $i-$th video, X_i are the features corresponding to all the K clips of the $i-$th video and y_i is the action label associated with the $i-$th video [3].

3.5. Attentive Mechanism

The objective of domain adaptation techniques is to mitigate the effect of distributional differences among different domains, but not all the temporal features are equally important in the alignment process. This leads to introducing a *temporal attentive mechanism* whose goal is to emphasize more the temporal features with larger domain discrepancy.

As proposed in [3] the attention mechanism operates at the relational level and the entropy criterion is used to generate the weights for each $n-$clips relation:

$$w_i^n = 1 - H(\hat{d}_i^n) \tag{8}$$

where $\hat{d}_i^n$ is the output of G_{rd}^n for the $i-$th video and $H(p) = -\sum_k p_k \cdot log(p_k)$ is the entropy function to measure uncertainty. The weight w_i^n increases when $H(\hat{d}_i^n)$ decreases, which means that the domains can be well distinguished.
At this stage, the weights are used to generate the attended versions of the $n-$clips relational level features as below:

$$h_i^n = (w_i^n) \cdot G_{tf}^{(n)}(G_{sf}(X_i)) \tag{9}$$

Therefore, the previous outcomes are aggregated obtaining the final video feature representation h_i:

$$h_i = \sum_{n=2}^{K} h_i^n \tag{10}$$

Finally, the objective is to minimize the entropy for the videos that are similar across domains. Therefore, focusing more on minimizing the entropy for videos which have low domain discrepancy, a new loss term is introduced:

$$\mathcal{L}_{ae}^{i} = (1 + H(\hat{d}_i)) \cdot H(\hat{y}_i) \tag{11}$$

where $\hat{d}_i$ and $\hat{y}_i$ is the output of G_{td} and G_{vy}, respectively.

3.6. Minimum Class Confusion (MCC)

The *Minimum Class Confusion* (*MCC*) was chosen as a possible method to improve the model's performance.
In the paper of Wang X. *et al*. [7], the *Versatile Domain Adaptation* (*VDA*) techniques have been investigated and the *class confusion* is presented as a more general inductive bias other than the domain alignment. The objective of VDA is to is to develop a model capable of addressing multiple Domain Adaptation (DA) scenarios, without requiring any modifications.

The term class confusion refers to the classifier's tendency at confusing the predictions between the correct and the ambiguous classes for target samples. In [7], it was uncovered that less class confusion leads to more transfer gains (i.e. knowledge learned from one domain to another domain) for all domain adaptation scenarios. Differently from the other DA techniques that focus more on the features, the class predictions have been investigated.
The first step consists of computing the *confusion matrix* which cannot be computed if the target labels are not provided, as in our case. In order to solve this issue, the *class correlation matrix* is used. Therefore a coarse estimation of class confusion is defined as follow:

$$C_{jj'} = \hat{y}_{.j}^T \hat{y}_{.j'} \tag{12}$$

where $\hat{y}_{.j'}$ denotes the probability that the $batch\_size$ examples in each batch come from the $j^{'}-$th class.
In quantifying the class confusion not all the samples are equally important, for this reason in [7] is introduced a weighting method. Its objective is to give more relevance to videos whose predictions show several peaks, indicating the presence of various ambiguous classes.
To do that, the *entropy* $H(\hat{y}_i) = -\sum_{j=1}^{|C|} \hat{Y}_{ij} log \hat{Y}_{ij}$ is introduced, where $\hat{Y}_{ij}$ represents the probability that the $i-$th video belongs to the $j-$th class.

The examples with higher certainty in class predictions should contribute more to the class confusion. Since the entropy represents the level of uncertainty, its opposite value is used in computing weights, obtaining the following formula for the $i-$th video:

$$W_{ii} = \frac{B(1 + exp(-H(\hat{y}_{i.})))}{\sum_{i'=1}^{B}(1 + exp(-H(\hat{y}_{i.})))} \tag{13}$$

where W is the corresponding diagonal matrix.
At this stage, it is possible to define the *weighted class confusion* as:

$$C_{jj'} = \hat{y}_{.j}^T W \hat{y}_{.j'} \tag{14}$$

Before defining the **Minimum Class Confusion Loss**, since the number of classes is high and there could be severe class imbalances in each batch, a category normalization step is required.

Finally, it is possible to define the *Minimum Class Confusion Loss* as follows:

$$L_{MCC}(\hat{Y}_t) = \frac{1}{|C|} \sum_{j=1}^{|C|} \sum_{j' \neq j}^{|C|} |\tilde{C}_{jj'}| \tag{15}$$

Each of the previous steps is explained in more details in [7], and a representation of the flow from one step to another is seen in Fig. 4. The improvements that this implementation has brought to our code will be discussed in the next section.

3.7. Loss

The overall loss function of TA3N, considering the contributions of both class and domain classifiers, is defined as follows:

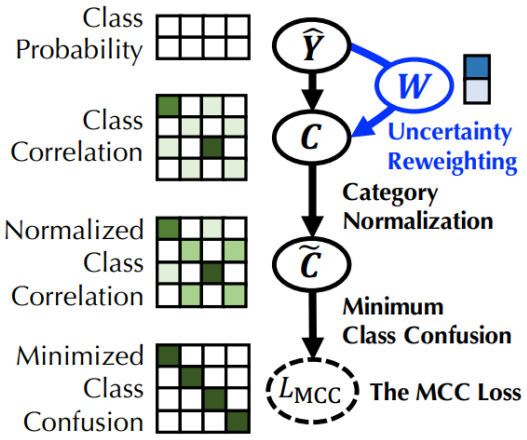

Figure 4. Minimum Class Confusion steps.

$$\mathcal{L} = \frac{1}{N_S} \sum_{i=1}^{N_S} (\mathcal{L}_{sy}^i + \mathcal{L}_{vy}^i) + \frac{1}{N_{S \cup T}} \sum_{i=1}^{N_{S \cup T}} \lambda^a \mathcal{L}_{ae}^i + \\ - \frac{1}{N_{S \cup T}} \sum_{i=1}^{N_{S \cup T}} (\lambda^s \mathcal{L}_{sd}^i + \lambda^r \mathcal{L}_{rd}^i + \lambda^v \mathcal{L}_{vd}^i) + \lambda^m \mathcal{L}_{MCC} \tag{16}$$

where λ^s, λ^r, λ^v, λ^a, and λ^m are the multipliers of the corresponding loss terms, which help tuning the weights of the corresponding loss.

4. Experiments

4.1. Experimental Setting

To test our algorithm the Epic Kitchen Dataset has been used, which consists of actions performed in the kitchen environment. For the training and the testing, we are provided with 3 kitchens corresponding to 3 different domains: D1, D2, D3 [4].

During the training phase the source train split and target train split are used. Instead, during the testing phase only the target test split is used. The training and testing data in both the domains are separated to avoid potential overfitting while aligning different domains distributions.

As explained in one of the previous sections, the initial phase is dedicated to the features extraction process using I3D (Sec. 3.1). It is important to underline that I3D is pretrained on *Kinetics dataset* as explained in [2].

The classifiers used within the architecture have structures composed by two linear layers followed by ReLU activation function, a dropout operator and a Softmax function. Notice that the output of the action classifiers is a probability vector with shape [1, num_classes]. The domain classifiers instead have as output the probability to belong to one

domain or the other, having as ground truth the vector [0, 1] for data coming from the source domain and [1, 0] for data coming from the target domain.
As optimizer the *Stochastic Gradient Descent (SGD)* is used with momentum and weight decay as 0.9 and 1×10^{-7}, respectively. The initial learning rate is 0.01, decreasing this value after 3000 iterations out of 5000 for a better convergence and the batch size is 32.
The best combination for the λ multipliers has been found through a manual grid-search method that has led to the following configuration [λ^s, λ^r, λ^v, λ^a] = [0.9,0.5,1,0.3] using all the G_d modules and the attention mechanism and [λ^s, λ^r, λ^v, λ^m] = [1,1,1,1] using the MCC loss.

The attention mechanism and the MCC loss are not used simultaneously.

4.2. Experimental Results

In this section, the results of our DA approaches are investigated. Through an ablation study, it has been possible to understand the contribution of each DA module in the proposed architecture.
The analysis starts with the evaluation of the performances obtained using only two action classifiers (at clip and video level), training and testing on the same domain as shown in Tab. 1. The results quantify how the model performs in absence of a Domain Shift scenario and represent an upper bound for the DA techniques' performance.

Temporal Aggregation	$D_1 \rightarrow D_1$	$D_2 \rightarrow D_2$	$D_3 \rightarrow D_3$
Pooling	54.02	61.20	68.69
TRN	59.77	69.20	72.18

Table 1. Model's accuracy training and testing on the same domain, without using any DA module.

The core results of our research study are summarized in Tab. 2 thanks to which is possible to both discuss the impact of each DA module and how its behaviour change across the different domain shifts. In the table, $D_i \longrightarrow D_j$ refers to the experimental setting in which the model is trained on the $i-$th domain and tested on the $j-$th.

Before analysing the influence of each DA module, the focus is on the temporal aggregation techniques. As mentioned in one of the previous sections, we expected that the performance in case of TRN would be better compared to Pooling due to the fact that the former exploits the temporal dependencies. Indeed, it is possible to appreciate in Tab. 2 that in each experimental setting the accuracy of the model using the TRN module in the temporal aggregation step is higher. Applying both pooling and G_{rd} does not result in any change of performances because of the structure of the network, therefore the corresponding row is empty. For the same reason the attention mechanism and pooling aggregation are blank.

Moving on to the discussion of the effectiveness of the DA modules, the first relevant observation is that each of them provide an improvement, although with different extent. In particular using TRN, there is a gain of about $\sim 1.01\%$ for G_{sd}, $\sim 0.13\%$ for G_{rd}, $\sim 1.22\%$ for G_{vd} with respect to the *SourceOnly* performance in case of the same temporal aggregation method. The better results have been achieved in the experimental setting in which all the G_d modules and the corresponding losses have been combined, with about $\sim 1.68\%$ improvement.

About the application of the Attentive Mechanism, we can notice a slight decrease in the results for each domain shift compared to the *All* G_d ones. The weights computed in the attentive mechanism are based on the G_{rd} predictions which is the DA module that seems to perform worse, which can be found as one of the causes of the worse performance.

Regarding the application of the *Minimum Class Confusion* there are different observations we can point out. First, the usage of the MCC loss has led to the best results for almost all domain shifts, outperforming the *SourceOnly* performance by more than 3%, see Tab. 2.
Focusing on the accuracy of each action class, the introduction of the MCC loss has carried out outcomes which are completely different from the previous ones. After an accurate analysis of the results' logs, it has been noted that in absence of MCC loss the model tended to correctly classify only few classes, corresponding to the ones with higher number of samples. The situation is completely different using MCC loss, since also the less-represented classes achieve an accuracy that is comparable with the well-represented ones.

In the Tab. 3, it is possible to notice how the MCC loss has led to better-balanced predictions on the different classes.

Class	**Num of Samples**	**Without MCC**	**With MCC**
Class 0	124	42.74%	55.65%
Class 1	104	75.96%	61.54%
Class 2	52	53.85%	53.85%
Class 3	34	17.65%	20.59%
Class 4	66	78.79%	83.33%
Class 5	13	15.38%	23.08%
Class 6	20	0%	85%
Class 7	22	0%	36.36%
Overall	435	50.57%	57.70%

Table 3. Model's accuracy among the classes, with or without the MCC loss. The domain shift taken as an example is $D_3 \rightarrow D_1$.

Setting	Temporal Aggregation	$D_1 \rightarrow D_2$	$D_1 \rightarrow D_3$	$D_2 \rightarrow D_1$	$D_2 \rightarrow D_3$	$D_3 \rightarrow D_1$	$D_3 \rightarrow D_2$	Mean	Gain
SourceOnly	**Pooling**	39.60	51.75	47.59	52.87	45.98	40.40	46.36	-
	TRN	42.27	55.54	54.02	59.03	49.93	41.60	50.31	-
Gsd Only	**Pooling**	40.53	54.32	47.82	55.65	48.51	40.13	47.86	1.50
	TRN	43.47	56.37	53.10	61.81	51.72	41.47	51.32	1.01
Grd Only	**Pooling**	-	-	-	-	-	-	-	-
	TRN	42.27	54.52	55.63	58.73	50.34	41.20	50.44	0.13
Gvd Only	**Pooling**	40.80	53.59	47.82	55.24	48.74	40.93	47.85	1.49
	TRN	41.07	57.80	54.02	61.19	50.34	44.80	51.53	1.22
All Gd	**Pooling**	41.33	54.31	46.44	54.83	49.20	40.13	47.70	1.34
	TRN	44.40	57.39	54.94	61.19	50.57	43.47	51.99	1.68
All Gd + Att	**Pooling**	-	-	-	-	-	-	-	-
	TRN	43.20	56.88	**56.55**	58.32	52.18	43.07	51.70	1.39
All Gd + MCC	**Pooling**	42.13	54.93	47.82	55.85	51.03	40.53	48.72	2.36
	TRN	**45.60**	**58.52**	55.17	**62.63**	**57.70**	**45.60**	54.2	3.89

Table 2. Ablation Study on EPIC-KITCHEN of our Model. Accuracies (%) are Top-1.

5. Conclusion and Future Work

In recent years, video classification methods have seen significant improvements thanks to the incorporation of the temporal dimension in their computations, leading to higher than ever performances. However, taking into account this added dimension, makes applying hyperparameter tuning both more challenging and resource-intensive, stressing the growing need for better-performing computers in this field.

In the paper we face a problem that pervades many areas of deep learning, especially when dealing with real-life tasks: Unsupervised Domain Adaptation.
Using the Epic Kitchen Dataset as base for our tests, we analyzed how to obtain domain-invariant features, even though our target data is unlabeled. One of the possible future works is to explore Domain Generalization, where we have neither the target labels nor the features [10].

The features have been analyzed with different aggregation levels. Through the ablation study we can see which level provided the most information regarding our action classification and domain adaptation task. We observed that G_{rd} did not bring significant improvement. One of the potential reasons for its poor performances might be the limited number of clips used, only 5 in our case. This is also the reason of the suboptimal results of the attentive module.

Despite the class imbalances in the target domain, by implementing the MCC Loss we were able to correctly classify the less-populated classes as well. This led to more than 3% increase in the results compared to the source. The improvements can also be seen thanks to the visualization in Fig. 5, where it is clearly noticeable how before the application of any modules the two distributions are clearly separated, while after applying TA3N with MCC they overlap. This is a clear statement of how Domain Alignment has been successfully reached.

Another potential direction for future implementations would be to face problems where the classes in the target domain space do not fully overlap those in the source domain space. This would introduce the challenges associated with PVDA [11].

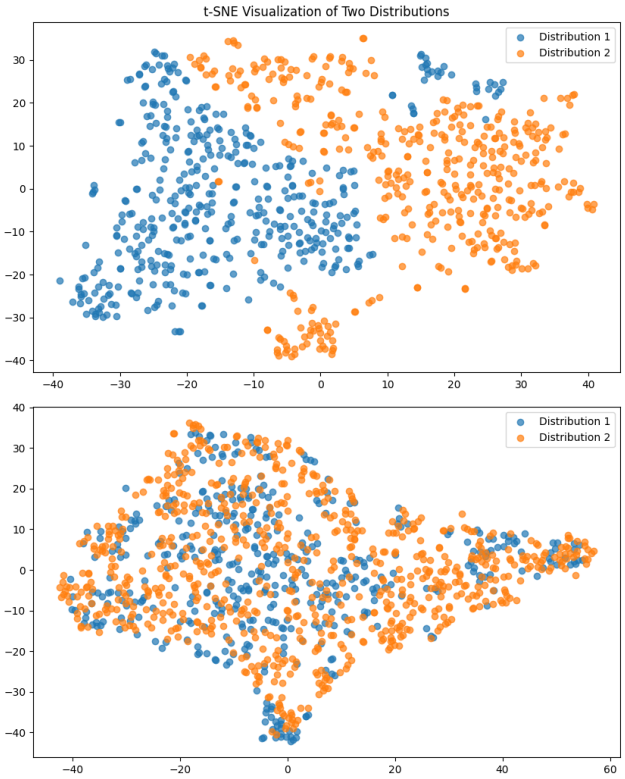

Figure 5. Distributions D1 and D2 before and after TA3N + MCC.

References

[1] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Michael I. Jordan. Partial transfer learning with selective adversarial networks, 2017. 3

[2] Zisserman A. Carreira, J. Quo vadis, action recognition? a new model and the kinetics dataset., 2017. In proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 6299-6308). 2, 3, 6

[3] Kira Z. AlRegib G. Yoo J. Chen R. Zheng J. Chen, M. H. Temporal attentive alignment for large-scale video domain adaptation, 2019. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 6321-6330). 2, 3, 5

[4] Doughty H. Farinella G. M. Fidler S. Furnari A. Kazakos E. ... Wray M. Damen, D. Scaling egocentric vision: The epic-kitchens dataset., 2018. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 720-736). 2, 6

[5] Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description, 2016. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 6321-6330). 2

[6] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 3

[7] Wang X. Long M. Wang J. Jin, Y. Minimum class confusion for versatile domain adaptation., 2020. In European Conference on Computer Vision (pp. 464-480). Springer, Cham. 2, 5, 6

[8] Nagrani A. Zisserman A. Damen D. Kazakos, E. Epic-fusion: Audio-visual temporal binding for egocentric action recognition., 2019. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 5492-5501). 2

[9] A. Tuan Nguyen, Toan Tran, Yarin Gal, Philip H. S. Torr, and Atılım Güneş Baydin. Kl guided domain adaptation, 2022. 3

[10] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip S. Yu. Generalizing to unseen domains: A survey on domain generalization, 2022. 8

[11] Yang J. Cao H. Chen Z. Li Q. Mao K. Xu, Y. Partial video domain adaptation with partial adversarial temporal attentive network., 2021. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 9332-9341). 3, 8

[12] Hana Ajakan Pascal Germain Hugo Larochelle Franc¸ois Laviolette Mario Marchand Yaroslav Ganin, Evgeniya Ustinova and Victor Lempitsky. Domain-adversarial training of neural networks., 2016. The Journal of Machine Learning Research, 17(1):2096–2030, 2016. 4

[13] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos, 2018. In Proceedings of the European conference on computer vision (ECCV) (pp. 803-818). 3