Domain Adaptation for Egocentric Action Recognition

Valerio Firmano
Polytechnic of Turin
Corso Duca degli Abruzzi, 24, Turin, Italy
valerio.firmano@studenti.polito.it

Filippo Grobbo
Polytechnic of Turin
Corso Duca degli Abruzzi, 24, Turin, Italy
filippo.grobbo@studenti.polito.it

Matteo Bianco
Polytechnic of Turin
Corso Duca degli Abruzzi, 24, Turin, Italy
matteo.bianco@studenti.polito.it

Abstract

Egocentric action recognition has gained significant attention due to its potential applications in various fields. One of the biggest challenges is improving its generalization across different domains. Working on RGB features extracted from the EPIC KITCHEN DATASET with a 3D-inflated network to handle the temporal dimension of videos, we investigate different DA methods to address the generalization problem on different domain shifts. We then propose an approach that combines three adversarial learning modules applied to different temporal aggregation levels (clip, relation and video levels) with an attentive mechanism and with the Minimum Class Confusion loss, gaining (on average over shifts) $\sim +8\%$ *on 'Source Only' of EPIC-KITCHEN. The code is available at: https://github.com/ValerioFirmanoo/mldl23-ego-1/tree/dev_TA3N_mcc*

Keywords - Egocentric Action Recognition, Domain Adaptation, Minimum Class Confusion, Attention.

1. Introduction

Egocentric Action Recognition refers to the process of identifying and understanding human actions from a first-person perspective. It involves analyzing visual data captured by a wearable camera or a device worn by a person performing actions. Unlike traditional action recognition, which relies on third-person viewpoints, Egocentric Action Recognition focuses on the wearer's own field of view, providing a more immersive and natural perspective. Nowadays, its potential applications in fields such as robotics, augmented reality, health care, sports analytics and more have led it to become a topic of common interest and development. With the rise of deep convolutional neural networks (CNNs) this problem was largely addressed initially from an individual frame perspective and only later from a video point of view, incorporating the temporal dimension [1].

Two fondamental aspects that must be taken into consideration when tackling this problem are Domain Shift and Domain Adaptation. The first one refers to the difference in the data distributions between the Source Domain (labeled training data) and the Target Domain (unlabeled test data). It occurs when the statistical properties of the data, such as the feature distributions or the label distributions, vary significantly between the two domains. This shift can pose a challenge to machine learning models, as they are typically designed to perform well when the training and testing data come from the same distribution. Possible answers to this problem are Domain Adaptation strategies, the purpose of which is to align Source and Target Domains to reduce the distribution discrepancy. This enables the model to generalize well in the target domain by learning domain-invariant representations or adapting its parameters to account for the differences between the domains.

Although the use of these techniques enables the achievement of state of the art performaces on different datasets, it is fair to highlight that they can have some limitations. In particular, the availability of labeled data for both Source and Target domains poses a bottleneck for its applicability in supervised techniques. Acquiring labeled data in general can be infeasible in certain scenarios or expensive and time-consuming, as for the EPIC KITCHEN DATASET [3]. In addition, the representativeness of the Source Domain is not guaranteed at all, and this may affect the limited generalization of these techniques in the case of novel scenarios or very different domains in the distribution. Addressing these limitations is an active area of research to develop more robust and effective Domain Adaptation tech-

niques.

This paper aims to address the challenge of Domain Shift by exploring Domain Adaptation techniques for egocentric action recognition on the EPIC KITCHEN DATASET [3]. This dataset is a large-scale egocentric video benchmark recorded by 32 people in their native kitchen environments. People were asked to capture all kitchen visits for three consecutive days while being alone in the kitchen, thus capturing only one-person activities. Data was captured using a head-mounted GoPro. The paper presents an ablation study conducted on the so called Temporal Attentive Adversarial Adaptation Network (TA^{3}N) [2], with features ectracted by means of an I3D CNN [1] from 3 kitchens of the EPIC KITCHEN DATASET [3].

Our main contribution are summarized as follows:

- We propose our approach of sampling clips from a video to feed to the I3D feature extractor [1].
- We show with our ablation study the effectiveness of Adversarial Domain Adaptation using spatio-temporal features, as proposed in [2].
- We observe a decay in performances using the attentive mechanism of [2]. Looking at experiment logs we notice a particular spike in the predictions for the largest classes, so we decide to introduce a Minimum Class Confusion loss [5] to cope with this problem.

In the next sections we first report some related works in this field and than present our method in details, giving also experimental settings and final results obtained.

2. Related works

2.1. Video Classification

Video data have been studied a lot in recent years. After the revolution of the CNNs with milestones like AlexNet [8], researchers have done many steps forward. The main additional complexity when dealing with videos instead of pictures lies in the temporal dimension. The first approach used consisted in extracting features from each single frame of a video using a CNN and then pooling prediction across the whole video [6]. This strategy has the crucial drawback of ignoring the temporal structure (for example if we reversed a video it would have the same feature representation of its real version). Thus in following years this approach started being combined with recurrent layers such as LSTM [11] in order to include the concept of temporal sequence of events. This strategy gave promising results but the main revolution in video modeling was brought by 3D CNNs. As their name may suggest, these networks adopt 3D convolutional filters to give a spatio-temporal representation of data. While ordinary 2D CNNs only exploit the spatial dimensions of frames, 3D filters are applied to a bunch of subsequent frames together. As a consequence they are able to extract features from data which take into account also the temporal aspect. Despite the promising idea, these architectures have a crucial drawback: the hugely increased number of parameters.

Different solutions were given in the recent years regarding this problem. Some works focused more on multimodal learning, demonstrating for example the importance of audio in egocentric action recognition [7]. Multiple modalities ususally adopted are RGB, Optical Flow and Audio. Optical flow refers to the pattern of apparent motion in a sequence of images, depicting how objects move and change position between consecutive frames. The integration of optical flow in convolutional neural networks (CNNs) can provide crucial temporal information and enhance the network's ability to capture motion patterns, enabling more accurate action recognition and understanding of dynamic visual scenes. Two important aspects regard the training and aggregation of multiple modalities. Temporal aggregation can be done within each modality independently and before they are fused, performing only at the end the so called *late fusion* of modalities [12]. Alternatively, inputs can be threated in an asynchronous way by the definition of a Temporal Binding Window where modalities are fused with a mid-level fusion and trained jointly [7].

Another solution to the issue of 3D CNNs is given in [1] with the I3D architecture. This network exploits a 2D CNN pretrained on ImageNet and *inflates* its filters in order to make them 3-Dimensional. This results in a great improvement of performances while keeping the number of parameters reasonably small. Multiple modalities can be included for this architecture as well, despite the increase in the number of model parameters. For this reason we choose to use only RGB with this network as feature extractor.

2.2. Domain Adaptation

In recent years, the challenge of domain adaptation have been studied from different point of views, including Supervised and Unsupervised Domain Adaptation techniques. Supervised Domain Adaptation utilizes a limited amount of labeled data in the target domain to align the distributions. On the other hand, Unsupervised Domain Adaptation focuses on aligning the distributions using only unlabeled data in the Target Domain. This is the scenario where we focus. Some approaches involve learning domain-invariant representations by minimizing some specific statistical distance between the two domains, such as Maximum Mean Discrepancy (MMD) [10]. Other techniques like deep correlation alignment [9] propose to match the mean and covariance of the two distributions.

A topic extensively studied in the literature and further explored by us is Adversarial Domain Adaptation. The concept of Adversarial Learning is mainly related with Generative Adversarial Networks (GANs) [4]. These networks

are developed to address the challenge of generating realistic samples by training a generator and discriminator in an *adversarial setting*. The generator network aims to generate synthetic samples that are indistinguishable from real data, while the discriminator network tries to differentiate between real and generated samples. By similar arguments, in Adversarial Domain Adaptation a feature extractor is trained in parallel with a domain classifier. The feature extractor aims to generate features that are indistinguishable between the source and target domains, while the domain classifier tries to correctly classify the domain. Before back-propagating the gradients, a gradient reversal layer (GRL) is applied to the domain classifier in order to optimize feature extractor to learn features that are indistinguishable across domains, enabling the model to generalize well also in the target domain.

Finally, another field of study is Partial Domain Adaptation, where the target domain may contain labels that are a subset of the source domain. In such scenarios, the aim is to construct a network capable of learning transferable features across Source and Target Domains while minimizing the target classification risk, as done in [13].

3. Proposed Methods

3.1. Overview

Our model is based on the TA^{3}N model proposed in [2]. Our net is thus composed by different layers extracting information about two different characteristics of video data: spatial and temporal. Each single data consist in a video from which we subsample K short clips. Features are extracted from each single clip using I3D CNN [1], which is able to apply convolution also in the temporal dimension. Next, the features associated to every single clip pass through the *spatial module*: G_{sf}. This consist of an MLP having the role of adapting and trasforming the general features to the scope of our net: action classification. After this, features are passed to the *temporal module*: G_{tf}. This part of the net has the role of extracting patterns linked to the sequentiality of clips in time. We propose two ways of doing the temporal aggregation: a simple average pooling of the clips or a TRN model [14].
The net is provided with two action classifiers:

- *GSY*: which classifies actions basing on features of the G_{sf}, thus before the temporal aggregation

- *GVY*: which classifies actions basing on features of the G_{tf}, thus after the temporal aggregation

Each of them makes a contribution to the loss function of the net:

$$\mathcal{L}^i_{sy} = \frac{1}{K}\sum_{i=1}^{K} L_y(G_{sy}(G_{sf}(x^j_i)), y_i)$$
$$\mathcal{L}^i_{vy} = L_y(G_{ty}(G_{tf}(G_{sf}(X_i))), y_i)$$

where L_y is a cross-entropy loss, x^j_i are the features of 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 label corresponding to the i-th video. A detailed representation about the final network structure can be seen looking at Fig 1. In the next parts all the aspects about each single component are explained in detail.

3.2. Feature Extraction

Every video of the dataset, corresponding to one action, is composed by RGB frames. We decide to subsample 5 clips (so $K = 5$) of 16 frames from every video and to extract them in such a way that they are equally spaced along each video. Then from every clip we extract features using a CNN with 3D filters. This process is performed with a I3D CNN with backbone BNInception, proposed in [1]. Is is important to notice that, differently from what is done in [2], we do not extract features from single frames of videos, but from short clips. This is why we use a 3D CNN instead of a simple 2D CNN.

3.3. Temporal Aggregation

Once the features are extracted there are two natural ways of interpreting them. One one side, by aggregating single frames with I3D CNN [1], we obtained clip features that encode the temporal dimension. On the other side instead there is also temporal reasoning, that is the ability to link or aggregate meaningful informations over time. For this purpose we propose two approaches in order to create temporal features:

- **Average Pooling**: simple pooling operator on the temporal dimension where the value of each feature is obtained by taking the mean of that feature between the different clips. The final result can be intepreted as a video feature that incorporates both spatial and temporal aggregation.

- **Temporal Relation Network**: multi-scale aggregation method designed by Zhou *et al.* [14] to learn temporal dependencies between clip features. The main goal of this method is to capture both long and short term time relation between features to improve activity recognition. Dependecies are represented by combination of clip features at different time level. For example, the so called 2-frames temporal relation, is

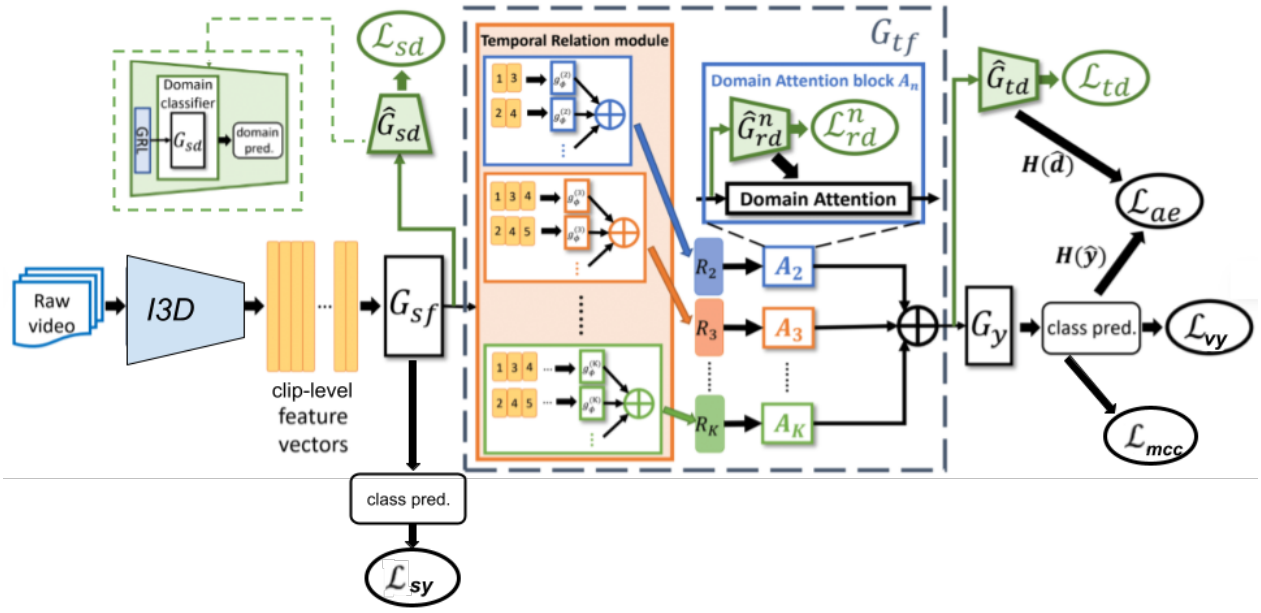

Figure 1. Final Architecture of the proposed confusion attentive Adversarial Learning Network. We can see the three different DA modules applied at different time aggregations. Each of theese modules is composed by a gradient reverse layer (*GRL*) and a domain classifier. In the temporal relation module we generate 4 relation features representations $R_n = \{R_2, R_3, R_4, R_5\}$ where R_n corresponds to the n-frame relation. After the relation module the features are avaraged to give the final video-level representation which gives us both a domain prediction and a prediction about the class, from which we compute the attentive loss $\mathcal{L}_{ae}$ and the minimum confusion loss $\mathcal{L}_{mcc}$.

given by the function:

$$R_2(X_i) = h_\phi(\sum_{j<k} g_\theta(x_i^j, x_i^k))$$

where h_ϕ and g_θ are two MLP used to fuse features. Multi-scale aggregation is made possible by extending the previous function to n-frames temporal relation. In our case n $\in$ [2,3,4,5]. The final aggregated feature is obtained by the accumulation of frames relations at different scales and can be written as:

$$R_f(X_i) = R_2(X_i) + R_3(X_i) + R_4(X_i) + R_5(X_i)$$

An example of this procedure is illustrated in Fig 2. Notice that in our model we consider relations between clips and not between frames as in [14].

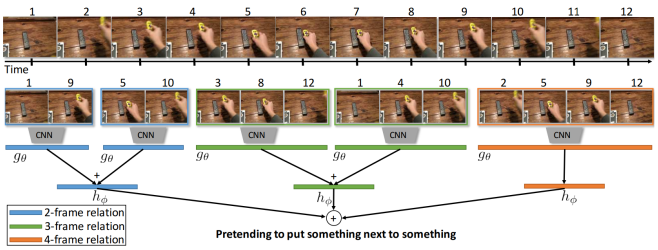

Figure 2. Example of TRN aggregation from Zhou *et al.* [14].

3.4. Domain Adaptation

The core of our model consist in the **Domain Adaptation** strategy. As in [2], we decide to implement an *Adversarial-based DA*. This means that we are providing our architecture with domain discriminators. During training, while the discriminators are optimized to distinguish domains, the G_{sf} is optimized in the opposite way. Thus we are instructing our feature extractor to ignore attributes of the data that allow to recognize their domain. Therefore we are representing data in such a way that only characteristics linked with the label are considered, while features dealing with the domain are ignored.

In practice we follow the implementation given in [2] of the TA^{2}N network. This means that we provide our net with three different domain discriminators:

- *GSD* which classifies domains using the clip-level features given by the G_{sf}, before temporal aggregation
- *GRD* which classifies domains using features given by different temporal combinations of clips of the same video. It exploits the TRN definition of dependencies and it is done before the final aggregation. For this reason it is possible to define n GRD classificators, where G_{rd}^n stands for the relation discriminator associated to the n-frame relation R_n.

- *GVD* which uses video-level features, the final output given by the G_{tf}

We use three different domain discriminators to exploit the more information possible contained in the data. In fact each one of these modules uses a different representation of the data that we are manipulating. Since each of these discriminators basically is a classifier of the domain, we associate them with a term in the loss of the whole network. Thus we will have three new loss terms:

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

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

$$\mathcal{L}^i_{td} = L_d(G_{vd}(G_{tf}(G_{sf}(X_i))), d_i)$$

where d_i is the domain label of each individual data and L_d is again a cross-entropy loss. Finally, when backpropagating errors, we introduce a *Gradient Reversal Layer* inside each of these discriminators. This has the role to flip the gradient whenever we are backpropagating errors outside DA modules towards the main model. In this way we are istructing the G_{sf} to align feature distribution between the two domains.

3.5. Attentive mechanism

We introduce then the attention mechanism proposed in [2].This mechanism consists of inserting multiplicative weights for features at relation level in order to give more importance to features that are more discriminative for the domain so that the DA mechanism can better focus on aligning those features. First of all we generate the domain attention weight for each n-frame relation feature using Entropy as below:

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

where $\hat{d}_i^n$ is the output of G^n_{rd} for the *i-th* video. These weights increase when the entropy $H(\hat{d}_i^n)$ decreases, which means the domain is well distinguished. Therefore we multiply each relation-level *i-th* feature, which is the output of the n-th temporal module of the TRN, for the attention weight:

$$h_i^n = (1 + w_i^n) \cdot G^{(n)}_{tf}(G_{sf}(X_i))$$

and aggregate along the different n-relations dimension (n) using an AvgPooling.

We want to attend to the videos which are more similar across domains, so the net can focus better on aligning the domains. In order to do that we add the additional *attentive entropy loss*:

$$\mathcal{L}^i_{ae} = (1 + H(\hat{d}_i))H(\hat{y}_i)$$

proportional to the domain and class Entropy.

3.6. MCC Loss

Since we saw that for several shifts the predictions tended to focus on a few classes, particularly the larger classes within the sample, and tended not to predict the smaller classes well, we thought we should include the Minimum Confusion Class loss, proposed in [5], in the learning process. This loss improves the accuracy of the predictor in case there are easily confounded classes, that is, in case the vector of predictions $\hat{y}$ has probability peaks for more than one class. We think the addition of this loss can work because it helps in the recognition of those smaller classes, which are easily mistaken for something else since the larger classes are better represented and thus better recognized by the network. As described in [5] we must first construct the "Confusion Matrix", whose entries $\hat{\boldsymbol{C}}_{j,j'}$ indicate the probability (frequency) of predicting class j' when the real class is j. Since we don't have the real labels of the target data we use instead the Class Correlation matrix

$$\boldsymbol{C}_{j,j'} = \hat{\boldsymbol{y}}^T_{.,j}\hat{\boldsymbol{y}}_{.,j'}$$

as a proxy of the real confusion matrix, where $\hat{\boldsymbol{y}}$ is the softmaxed prediction vector. Clearly we want this loss to focus on the most informative predictions thus those with peaked distributions over the classes, while focusing the least on the less informative uniform distributions. We then introduce a *weighting mechanism* based on *entropy* $H(\hat{\boldsymbol{y}}_i) = -\sum_{j=1}^{|C|}\hat{y}_{i,j}log\hat{y}_{i,j}$ as a measure of uncertainty in the distribution of predictions. So, the transformation matrix is the diagonal matrix **W** such that

$$\boldsymbol{W}_{ii} = \frac{B(1 + exp(-H(\hat{\boldsymbol{y}}_i)))}{\sum_{i'=1}^{B}(1 + exp(-H(\hat{\boldsymbol{y}}_i)))}$$

where B is the batch size and W_{ii} models the importance of the i-th example of the batch. Note that we take the opposite value of entropy to reflect *certainty*.

Finally, we define **Class Confusion** as

$$\boldsymbol{C}_{j,j'} = \hat{\boldsymbol{y}}^T_{.,j}\boldsymbol{W}\hat{\boldsymbol{y}}_{.,j}$$

and after a *category normalization* (that means normalizing on the rows of the confusion matrix) we can define the **Class Confusion Loss** as

$$\mathcal{L}_{MCC} = \frac{1}{|C|}\sum_{j=1}^{|C|}\sum_{j'\neq j}^{|C|}\tilde{\boldsymbol{C}}_{jj'}$$

The whole process is summarized in Fig 3.

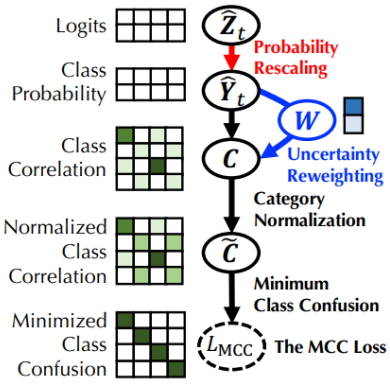

Figure 3. Brief description of the steps required to compute the MCC

3.7. Loss

The finally loss function is given by:

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

where $\lambda^s, \lambda^r, \lambda^v, \lambda^h$ and λ^m are hyperparameters that we tune manually to obtain the best accuracies.

Actually we never exploit together the attention mechanism and the MCC loss, thus we keep in the general loss only the terms we are practically using.

4. Experimental Results

4.1. Dataset and Experimental setting

We test our network on the EPIC-KITCHEN DATASET [3]. The data are divided into train and test sets for each kitchen, and data are provided for 3 different kitchens, thus 3 different domains: D1,D2,D3. We implement all network methods in Pytorch by running the code on Google Colab to have access to GPU's.

As a first step, we extract the RGB features from the videos with the pretreated I3D network, obtaining 5 feature vectors (one for each clip in the sample) of size 1024. Each classifier (both domain and classes) within the network has the structure we can see in Fig 4 where num classes is 8 in case of action prediction, 2 in case of domain prediction. The optimizer of the network is a SGD with momentum 0.9, decay $1e-7$, learning rate 0.01 that after 3000/5000 iterations changes to 0.001. The chosen batch size is 32. The best combination of λ multiplier for the loss is $[\lambda^s, \lambda^r, \lambda^v, \lambda^h] = [0.75, 0.5, 0.75, 0.3]$ when we use All GD + Att and $[\lambda^s, \lambda^r, \lambda^v, \lambda^m] = [1, 1, 1, 1]$ when we use All Gd + MCC loss.

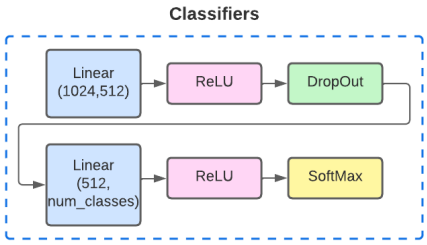

Figure 4. Structure of the lable and domain Classifiers within the net: two Linear layers followed by ReLU activation function, a Dropout operator and a Softmax at the end. Num classes is 8 for the lables, 2 for the domain.

Temporal aggregation	D1→D1	D2→D2	D3→ D3
Avg Pool	55.86	63.42	68.10
TRN	57.70	65.73	70.94

Table 1. Results for training and testing on the same Domain without any additional module, using AvgPool or TRN as temporal aggregation methods.

4.2. Results

In this section we report and discuss the experimental results obtained with the different Domain Adaptation approches of our model. First of all we show in Table 1 the results of the classification made only with the two label classifiers (at clip-level and video-level), without any additional DA module because we train and test on the same Domain. We can use theese results as a sort of upper bound for the case in which we shift between domains.

The main results of our work are summarized in Table 2. Each $D_i \rightarrow D_j$ shift represents an experiment in which we train on D_i domain data and test on D_j. To better understand the effects of each DA module, we have reported an ablation study in the table in which we can observe the change in accuracy when the modules are activated individually and when they are aggregated. Notice that results for some DA methods (as GRD and all modules combinations that include GRD) are impossible with AvgPool Temporal Aggregation so the results are left blank.

We can see an increase in general by a few percentage points between AvgPool and TRN as methods of aggregating temporal features,especially when we activate the DA modules. Beyond that, we can see that all DA modules bring a benefit to the final accuracy, particularly with $\sim +4.5\%$ for GSD, $\sim +2\%$ for GRD and $\sim +5\%$ for GVD (notice that the gain is calculated reffering to the same temporal aggregation method in *'Source Only'*).

Setting	Temporal aggregation	D1→D2	D1→D3	D2→ D1	D2→ D3	D3→ D1	D3→D2	Mean	Gain
Source Only	**Avg Pool**	44.53	52.05	44.83	47.74	45.52	48.13	47.13	0
	TRN	48.40	55.13	45.52	48.36	44.37	49.60	48.56	0
GSD only	**Avg Pool**	50.67	56.78	48.51	55.65	43.45	55.73	51.80	+4.67
	TRN	50.40	59.55	48.05	56.16	48.74	56.67	53.26	+4.70
GRD only	**Avg Pool**	-	-	-	-	-	-	-	-
	TRN	47.07	56.26	46.21	52.16	49.43	52.67	50.63	+2.07
GVD only	**Avg Pool**	49.33	56.88	49.43	57.49	44.60	54.80	52.09	+4.96
	TRN	49.33	60.27	48.97	58.10	48.28	56.53	53.58	+5.02
All GD	**Avg Pool**	-	-	-	-	-	-	-	-
	TRN	50.93	61.60	51.26	**59.65**	51.26	57.47	55.36	+6.80
All GD + Domain Attention	**Avg Pool**	-	-	-	-	-	-	-	-
	TRN	48.40	**62.01**	45.75	54.72	51.72	58.53	53.52	+4.96
All GD + MCC Loss	**Avg Pool**	-	-	-	-	-	-	-	-
	TRN	**53.60**	60.57	**54.48**	56.16	**55.86**	**60.40**	**56.79**	**+8.2**

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

Regarding the attention mechanism, we find a deterioration in performance in most shifts., even trying different combinations of hyperparameters. We thought that this may be due to the fact that in first place attention, in order to calculate the weights, relies on the predictions made by GRD, which is the module that performs worst in our case. We also think that our model has already managed to achieve good gains in accuracy, leaving little room for improvement, and it could therefore be that the attention weights 'confound' the network predictions rather than help them.

Therefore, we decide to implement an alternative mechanism that takes advantage of Minimum Class Confusion loss. Our intuition started from the fact that we saw the predictions sometimes concentrated too much on the classes with largest number of samples, and this could be due to class confusion, as explained in the Methods section. As can be seen in table 3, we observe a better accuracy on smaller classes when applying the MCC Loss. In fact we find a gain in accuracy in almost all shifts and in general better balanced predictions on the different classes, reaching a mean gain on all classes of $\sim +8\%$ with respect to *'Source Only'* with TRN.

Class	Num of samples	Without MCC	With MCC
Class 0	124	51.61	45.97
Class 1	104	67.31	74.04
Class 2	52	61.54	42.31
Class 3	34	8.82	41.18
Class 4	66	75.76	78.79
Class 5	13	30.77	46.15
Class 6	20	0.00	45.00
Class 7	22	0.00	27.27
Total	435	51.26	55.86

Table 3. Accuracy (%) obtained in each of the eigth label classes without and with MCC Loss. Here we take as an example the shift $D3 \rightarrow D1$

.

5. Conclusion

In this paper we use EPIC-KITCHEN Dataset to investigate the domain shift problem across videos for action recognition task. We start with features extracted from a I3D-inflated network so that both spatial and temporal information of the video are preserved. Then we propose a Domain Adaptation method, in order to align the domains, based on three Domain classifiers at different time levels, to which we assign the task of performing Adversarial Learning. Since we notice that there could easily be confusion in the prediction of similar classes, we add Minimum Class Confusion Loss to the network which allows us to obtain more balanced and accurate predictions, gaining $\sim +2/3\%$ in several shifts respect to 'All GD', with an overall average accuracy gain of $\sim 8\%$ on 'Source Only'.

The attention method, on the other hand, does not bring positive results, returning lower accuracy in different shifts.

References

[1] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[2] Min-Hung Chen, Zsolt Kira, Ghassan AlRegib, Jaekwon Yoo, Ruxin Chen, and Jian Zheng. Temporal attentive align-

ment for large-scale video domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6321–6330, 2019.

[3] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset, 2018.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[5] Ying Jin, Ximei Wang, Mingsheng Long, and Jianmin Wang. Minimum class confusion for versatile domain adaptation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16*, pages 464–480. Springer, 2020.

[6] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[7] Evangelos Kazakos, Arsha Nagrani, Andrew Zisserman, and Dima Damen. Epic-fusion: Audio-visual temporal binding for egocentric action recognition, 2019.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[9] Aochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European Conference on Computer Vision*, pages 443–450. Springer, 2016.

[10] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance, 2014.

[11] Amin Ullah, Jamil Ahmad, Khan Muhammad, Muhammad Sajjad, and Sung Wook Baik. Action recognition in video sequences using deep bi-directional lstm with cnn features. *IEEE access*, 6:1155–1166, 2017.

[12] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition, 2016.

[13] Yuecong Xu, Jianfei Yang, Haozhi Cao, Qi Li, Kezhi Mao, and Zhenghua Chen. Partial video domain adaptation with partial adversarial temporal attentive network, 2021.

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