Investigating the OTAM Generalizability Problem

Alexander Swaters — S4284917

I. Abstract

Ordered temporal alignment [2] (OTAM) has been shown to outperform meta-learning baselines. Surprisingly, a simplistic classifier-based baseline with no temporal alignment outperforms metric learning with OTAM, largely attributed to the classifier's ability to exploit dropout [18]. In this paper, I propose and investigate potential improvements to the OTAM method to combine the success of classifier-based methods with the success of OTAM relative to meta-baselines by attempting to improve the generalization of OTAM. I find that temporal order shuffling augmentation and all-pair similarity loss contribution do not provide any significant improvement over OTAM and that reducing the number of learnable parameters in the OTAM embedder decreases performance.

II. Introduction

Video classification has been a topic of interest in the field of computer vision and machine learning for several decades. Over the years, significant advancements have been made in this area with the development of deep learning algorithms and convolutional neural networks [1] (CNNs). These advancements have led to state-of-the-art results in action recognition.

However, despite these successes, the challenge of limited labeled data remains a hindrance in the progress of video classification [18]. In comparison to image classification, video classification requires a large amount of annotated data for training deep learning models, which is often scarce and expensive to obtain. This problem is particularly severe in the video domain, where labeling every frame of a video is a time-consuming and resource-intensive process.

Few-shot video classification aims at reducing the amount of data required to train an effective model. Specifically, it presumes ample labeled data on its training set, but only a few labeled samples on the test set. In an n-way, k-shot problem, there is a support set consisting of k-samples (usually < 10), per unseen class, of which there are n. In addition, there are a certain number of query videos to classify.

Meta-learning methods train in episodes of this few-shot problem, repeatedly training on subsets of the training set, simulating the way the model will eventually be used. Meta-learning methods usually compare the similarity of the query to the members of the support set, evaluating feature space distance to determine which support class the query belongs to.

Ordered temporal alignment or OTAM [2], is such a method. It uses a convolutional backbone to extract embeddings for snippets taken from a video, producing a series of embeddings. In order to compare two videos, this series is reduced to a single vector representing the entire video. The simplest way of doing so is averaging along the temporal dimension, but OTAM proposes an improvement based on sequence alignment. In simple terms, it matches each in a video to a frame in the video it is compared with and vice versa. The idea is that an action consists of a series of steps in order, that two videos capturing the same action will have frames corresponding to each action in a similar order, and that the similarity along these ordered steps is the key to recognizing the action. OTAM was a convincing improvement upon the meta-learning baseline at the time, with OTAM achieving 42.8% accuracy on few-shot Something-something v2 (SSv2) [6] (5 way, 1 shot) compared to the baseline at 33.6%.

Later Zhu et al. [18] conducted experiments examining the state of few-shot learning, including OTAM. In their experiments, the difference between the meta-baseline and OTAM was smaller, with the meta-baseline achieving 37.3% (improved with a smaller learning rate and temporal jittering augmentation) and OTAM achieving 41.55% (slightly lower due to a different optimizer for a fair comparison). They also proposed a simple classifier-based method with no temporal alignment, which achieved a competitive 40.78%, outperforming the meta-baseline. Surprisingly, after adding dropout to this method, they achieved 46.04% accuracy, outperforming OTAM. They attribute this to their use of metric learning while preserving a linear classifier and dropout, concluding that a cosine classifier is not necessarily better than a linear classifier and that dropout improves the generalizability of the model. They named their model Baseline+.

Baseline+ performs well because it can exploit a linear classifier and dropout, unlike OTAM. However, OTAM does still outperform the meta-baseline, suggesting that its temporal alignment is valuable. In this paper, I investigate methods to improve the generalizability of OTAM to bridge the gap between it and Baseline+ while continuing to benefit from temporal alignment. I propose various approaches to this end. I evaluate the performance of each method on the Something something v2 dataset [6] in a 5-way 1 shot setting.

III. Related Works

A. Few-shot learning

The most simple example of few-shot learning is using a model pretrained on a large dataset like Imagenet [4], and then fine-tuning it on your different but similar problem. The pretrained model can leverage its prior knowledge to learn to recognize novel classes despite a lack of training samples; but when these samples are truly scarce (for example, numbering less than 10), traditional methods fail to produce adequate results [2] and are not seen in modern few shot learning literature.

Metric learning is a paradigm commonly used [2][17][9] to overcome this issue. Rather than classifying a sample, models trained using metric learning aim to predict the similarity of two samples, or phrased differently, the distance between their positions in feature space. Then, provided that novel classes are sufficiently consistent with the training data in terms of their mapping to the feature space, the few provided samples of these classes can be used as prototypes. Queries are classified according to their closest prototype, under the assumption that the prototype of their own class will be most similar to them.

In combination with metric learning, many few-shot approaches also use meta-learning [2][17][9]. In traditional learning, samples are fed batch-wise, and the error on the batch is used to compute loss which is then backpropagated to update the model weights. In contrast, meta-learning consists of episodes, where each episode simulates the few-shot-learning problem. Episodes consist of a support set with k samples from n unseen classes. Using the metric learning method described above, the model determines the similarity between the query and each support prototype. Loss is computed at the end of the episode based on the difference between the predicted similarities and actual similarities between the query and the support prototypes. The purpose of this approach is to make the training process similar to the actual usage of the model with the purpose of improving performance.

B. *Video classification*

The history of video classification in deep learning can be traced back to the early 2010s when convolutional neural networks (CNNs) started to gain traction in computer vision tasks [1]. At the time, video classification was a challenging problem due to the high dimensional nature of videos and the need for capturing both spatial and temporal information. Early approaches to video classification relied on hand-crafted features and traditional machine learning algorithms [8][12][14]. With the advent of deep learning and the success of CNNs in image classification, researchers started to explore their application to video classification.

Video classification remains inherently more challenging than image classification due to its additional dimension. Effective video classification models must take temporal and spatial information into account to achieve state-of-the-art performance. C3D [13] uses 3D spatiotemporal convolution to extract features from sequences of RGB frames. TSN [15] uses two-stream 2D or 3D CNNs on both RGB and optical flow modalities. A weakness of these approaches is that they rely upon large datasets to achieve their performance, and annotated video data remain sparse, especially in niche applications. Their weakness with limited data can be attributed to their large number of parameters, naturally making them more challenging to optimize.

The approaches above focus on short-term temporal relationships, like the movement of an object from frame to frame, but are less suited to modeling the relationship between picking up an object and putting it down involved in a certain action. TRN [16] proposes a temporal relational module to achieve superior performance in this regard. However, its pooling/fusing features from different frames in its final layers to extract a single feature vector limits its ability to extract long-term temporal information.

OTAM [2] proposes a means of long-term temporal alignment based on sequence alignment techniques. First, snippets are sampled from the video using a sparse sampling method introduced by the creators of the TSN. A video in SSv2 [6] typically has 30-100 frames divided into 8 segments, and a snippet is extracted from each segment, resulting in a sequence of 8 snippets regardless of the video length. A snippet refers to one or more frames in a particular modality (RGB, RGB difference, optical flow). Thus, for a video x, sparse sampling will produce a sequence of snippets $x^1, x^2..., x^T$, where T is the number of segments. The motivation for sampling frames instead of feeding the whole video is that adjacent frames are highly similar to each other, capturing only very short-term relationships like the presence of movement. Even nearby frames tend to contain highly similar content, making them redundant. Sparse sampling makes a tradeoff of eliminating frames highly likely to contain redundant information at the cost of potentially losing valuable information in the discarded frames.

After sampling snippets, an embedder is used to extract features from each snippet, producing a sequence of feature vectors representing the characteristics of the snippet. For the sequence above, the sequence of embeddings will have the form $f_\theta(x^1), f_\theta(x^2)..., f_\theta(x^T)$, where f is the forward function of the embedder and θ is the weights of the embedder. Snippets occupy a very short timespan by nature, so their feature representations essentially encode an instant in time, possibly including information about the movement of the displayed objects depending on the snippet modality. The embedder consists of a convolutional neural network, with the authors of OTAM using a ResNet50 [7]. A key advantage of using a 2D CNN backbone is that pretrained weights can be used. The image domain is better studied, has more data available, and models with state-of-the-art performance are readily available.

OTAM, being a metric learning algorithm, must produce a similarity score for the two videos using sequences of snippet embeddings. The simplest way of computing the similarity of the two sequences is to average both along the temporal dimension and compute cosine similarity. However, this sacrifices all ordering and temporal relationship information. Instead, OTAM proposes a sequence alignment-based technique called soft dynamic time warping. The first step is to compute the frame-level distance matrix D, where

$$D(l,m) = 1 - \frac{f_\theta(x_1^l) \cdot f_\theta(x_2^m)}{||f_\theta(x_1^l)|| \cdot ||f_\theta(x_2^m)||}$$

where x_1, x_2 are the l^{th} frame of video x_1 is compared to the m^{th} frame of x_2. Next, the distance matrix is used to compute an optimal alignment path, representing the degree of overlap of the specific series of steps that compose an action. After finding the optimal alignment path, only the similarities along the alignment path are considered. The authors also pad the distance matrices with an additional column on each side, allowing the alignment to begin and end freely, in contrast to forcing the alignment to begin at $D(0,0)$. The score along the path is computed as follows:

$$\gamma(l,m) = D(l,m) + \begin{cases} min\{\gamma(l-1,m-1), \gamma(l-1,m), \gamma(l,m-1)\}, \\ \qquad\qquad\qquad\qquad \text{if } m = 1 \vee m = T+1 \\ min\{\gamma(l-1,m-1), \gamma(l,m-1)\}, \text{ otherwise} \end{cases}$$

Note that the OTAM authors use continuous relaxation in their implementation of min. They use log-sum-exp with smoothing to approximate the non-differentiable minimum operator in the above equation.

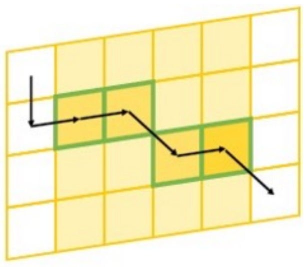

Fig. 1. Diagram from [2] of how OTAM finds the cheapest path through the distance matrix. Note that darker squares indicate higher similarity and only squares outlined in green contribute to loss.

C. Baseline+

A 2021 paper [18] conducted a comparative study of few-shot-video methods including OTAM, aiming to evaluate the methods it considered fairly. It also contributed a new approach to few-shot learning in the form of a simple baseline named Baseline+. Despite using no temporal alignment whatsoever, Baseline+ outperformed all meta-learning methods of the time, achieving 46.04% accuracy in a 1-shot 5-way setting (compared to OTAM, which they found to achieve 41.55% accuracy).

Like OTAM, Baseline+ works by embedding a series of snippets, in this case, an RGB image from each video segment with embedder f_θ. Unlike OTAM, Baseline+ uses a classifier-based method and no meta-learning or temporal alignment, instead averaging its sequence of embeddings along the temporal dimension. Baseline+ trains its linear classifier $C(.|W_b)$ by minimizing cross-entropy classification loss on the base class data X_b (in the case of SSv2 [6] this corresponds to the 64 classes in the training set). During testing, f_θ is frozen and a new classifier $C(.|W_n)$ is trained using only data from the support set.

During the model training, the authors focus entirely on a feature extractor f_θ with strong generalization, employing dropout before its linear classifier. Dropout is an effective way to improve the generalization ability of the model, which is of great importance for few-shot learning. Meta-learning methods do not benefit from dropout because intuitively, dropout forces the network to update a sub-network during training, and the average of a series of sub-networks during testing (with dropout set to 0) can generalize better. However, for metric-based methods, 0 dropout is different from the average of sub-networks due to normalization.

During testing, f_θ is frozen and the classifier is trained using only the support set. The authors of Baseline+ use a technique inspired by weight imprinting [5][11] and cosine similarity [3] to train a new classifier. $C(.|W_b)$ is retained and a new linear classifier $C(.|W_n)$ is appended. $C(.|W_n)$ takes the logits as input, which are clustered better than features. For a sample (x,y), define C_{train} training classes, C_{test} testing classes, logits $z(x;y) \in \mathbb{R}^{C_{train}}$, $C(.|W_n)$'s weights $w \in \mathbb{R}^{C_{train} \times C_{test}}$, and $C(.|W_n)$'s biases $b \in \mathbb{R}^{C_{test}}$, and the k^{th} column of w by w_k. The authors imprint w_k with $z(x_k;k)/||z(x_k;k)||$ and b with 0 where x_k denotes the k^{th} sample in the support set for one shot tasks. For multiple shot tasks, they average the logits of each class. Intuitively, the imprinted scores can be viewed as templates of the support samples. When a query passes through the classifier, a higher score denotes more similarity to the corresponding template.

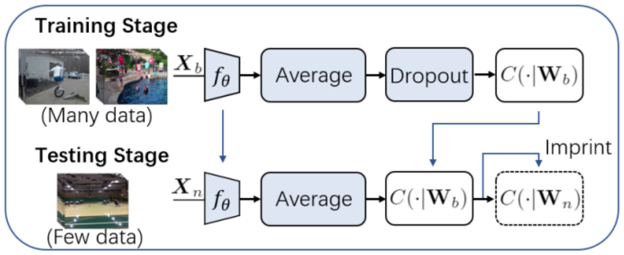

Fig. 2. Diagram from [18] of Baseline+, visualizing how the process described above appends a new classifier.

IV. Methods

A. Order shuffle augmentation

OTAM finds the optimal alignment path according to the two embedding sequences it takes as input. The produced alignment depends on the order of the embedding in the sequences, so even if a sequence of embeddings is compared to itself, but in a different order, the alignment score will not be very high. However, the precise ordering of steps that compose an action may not be fixed. For example, when getting dressed, putting on a shirt can be done before or after putting on socks. OTAM would fail to model this when comparing two dressing actions.

Furthermore, considering only similarity along the optimal alignment path teaches the model to optimize to that path, even if only indirectly. To that end, it is more likely to emphasize predictions near the diagonal, where alignment is most likely to occur (two actions of the same type will tend to proceed at a similar pace). The model may also learn orderings specific to the training classes, placing specific emphasis on key steps in training actions. For instance, the model could learn to place extreme emphasis on tea bags appearing in early segments to recognize the tea-making action, but this is unlikely to generalize well on the test set.

To mitigate these disadvantages, I propose an order shuffle augmentation. Positions in the embedding sequences are randomly swapped, with the total swap distance (potentially over multiple swaps) being a random number in the range $[0, \lambda T]$, where T is the number of segments in the sequence and λ is the shuffle strength parameter. Randomly changing the order of the embeddings will force the model to depend less on a specific embedding order. It effectively simulates the possibility of steps occurring out of order, and to some extent forces the model not to depend too heavily on specific orders witnessed in the training set.

B. All pair loss contribution

OTAM's performance relative to the meta-baseline demonstrates that it represents temporal relationships better than simple averaging across the temporal dimension. However, OTAM is limited in only being able to compare along its alignment path. It cannot change the order of its input sequences, and it cannot consider potentially relative comparisons that are not along its path. Therefore, I propose a modification to the OTAM similarity function:
$similarity = \gamma(T, T+1) + \rho \Sigma_{i=1}^{T} \Sigma_{j=1}^{T} D(i,j)/T^2$
Allowing all pairs to contribute to the loss function, where ρ is the all pair similarity weight.

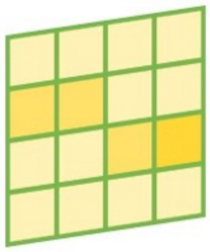

Fig. 3. Diagram of all-pair loss. Note that darker squares indicate higher similarity and all squares are outlined in green, indicating that they contribute to loss.

It may seem counterintuitive to include the all-pair weight which has been shown to be outperformed by OTAM, but the all-pair similarity does still indicate the similarity of two videos to some extent. Similar actions often appear in similar settings, have similar items, or have similar brightnesses, and all of these characteristics will be reflected in the all-pair weight. Furthermore, out-of-order actions will still contribute to the overall similarity. Meanwhile, the all-pair weight is much simpler than OTAM, which should allow it to generalize better on the test set. I use a low ρ, placing the majority of the emphasis on the path similarity, but hope that the all-pair will contribute to capturing the non-temporal video similarities.

C. Partial embedder freezing and lighter embedder

A common cause of overfitting is an overly large model. With more parameters, the model has more capacity to fit more complex patterns, even to the point that it simply learns the training data's labels and not the mapping from the input space to the label space. Reducing the number of parameters correspondingly enforces a simpler, more rigid mapping, but this mapping is in turn more likely to generalize well. I propose two ways of limiting the model's parameter count, namely using a smaller embedder and freezing part of the embedder. In particular, I try a ResNet34 [7] rather than the ResNet50 used in OTAM, and I try freezing the first three convolutional phases of ResNet50, allowing the training process to update only the later weights. In this sense, the model cannot change the lower-level features learned by the model, it can only adjust its interpretation of those features.

V. Results

I used the same experimental setup as seen in Baseline+, implemented in PyTorch [10]. Specifically, all training and testing were performed in a 5-way 1-shot setting. All experiments were performed on a few-shot subset of SSv2 [6] with 100 classes. All models were trained with a learning rate of 10^{-5} for 400 epochs with 100 episodes per epoch with a ResNet50 [7] backbone. For pretrained ResNets, ImageNet [4] weights are used. The preprocessing and augmentations used were the same as those used in Baseline+ and with 8 segments per video. Each validation step was 1,000 episodes, while every model test involved 10,000 episodes (again, all same as in Baseline+). The following are 1-shot accuracies on the test split for each method:

Method	Accuracy (%)
Meta baseline	34.97
OTAM	38.72
Temporal shuffle ($\lambda = 1$)	38.33
All pair weight ($\rho = 0.2$)	38.63
Partial freeze	36.73
ResNet34	37.95

The results achieved in my experiments were below those achieved in Baseline+, despite copying the methods almost exactly. It is possible there is a mistake in the implementation somewhere or that the model needs to be trained for longer. However, I found both my meta-baseline and OTAM results were 2.5% below the results reported in Baseline+, suggesting that the difference is consistent.

None of the proposed methods outperformed the original OTAM design. Partial freezing and using a ResNet34 lowered the performance, suggesting that the additional learnable parameters contribute to performance. Temporal shuffling and the contribution of all pair similarity loss do not seem to have a significant impact on the model's performance.

It is possible that to some extent, the ordering of steps in action is already varied within classes in the training set, which will already induce the effect intended by the temporal order shuffling augmentation. It should also be noted that due to the nature of random sampling, the same segment can produce a very different snippet, which can also produce a similar effect (forcing the model to not rely too heavily on a particular embedding in a particular position).

All pair weight places additional value on the temporally invariant similarities between two videos because its additional term discards ordering information. However, it is possible that this information is already sufficiently represented in the path. It is also possible that a high ρ waters down the meaning of the loss, and that all-pair weight could be more effective at a lower ρ, but it does not seem likely that this would result in a significant improvement in performance.

VI. Conclusion

In this paper, I proposed and investigated several ways to improve the generalizability of OTAM while preserving its valuable ability to model long-term temporal relationships. I found that neither temporal shuffling augmentation nor all-pair similarity loss contribution lead to significant improvement over the original OTAM design. Furthermore, I found that attempting to decrease the number of learnable parameters in OTAM's embedder decreases performance.

References

[1] Md. Zahangir Alom, Tarek M. Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C. Van Essen, Abdul A. S. Awwal, and Vijayan K. Asari. The history began from alexnet: A comprehensive survey on deep learning approaches. *CoRR*, abs/1803.01164, 2018.

[2] Kaidi Cao, Jingwei Ji, Zhangjie Cao, Chien-Yi Chang, and Juan Carlos Niebles. Few-shot video classification via temporal alignment. *CoRR*, abs/1906.11415, 2019.

[3] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *CoRR*, abs/1904.04232, 2019.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[5] Guneet S. Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. *CoRR*, abs/1909.02729, 2019.

[6] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fründ, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "something something" video database for learning and evaluating visual common sense. *CoRR*, abs/1706.04261, 2017.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[8] Orit Kliper-Gross, Tal Hassner, and Lior Wolf. One shot similarity metric learning for action recognition. *Similarity-Based Pattern Recognition*, page 31–45, 2011.

[9] Rex Liu, Huanle Zhang, Hamed Pirsiavash, and Xin Liu. STAF: A spatio-temporal attention fusion network for few-shot video classification. *CoRR*, abs/2112.04585, 2021.

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[11] Hang Qi, Matthew Brown, and David G. Lowe. Learning with imprinted weights. *CoRR*, abs/1712.07136, 2017.

[12] Paul Scovanner, Saad Ali, and Mubarak Shah. A 3-dimensional sift descriptor and its application to action recognition. *Proceedings of the 15th ACM international conference on Multimedia*, 2007.

[13] Du Tran, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. C3D: generic features for video analysis. *CoRR*, abs/1412.0767, 2014.

[14] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. *2013 IEEE International Conference on Computer Vision*, 2013.

[15] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks for action recognition in videos. *CoRR*, abs/1705.02953, 2017.

[16] Bolei Zhou, Alex Andonian, and Antonio Torralba. Temporal relational reasoning in videos. *CoRR*, abs/1711.08496, 2017.

[17] Linchao Zhu and Yi Yang. Compound memory networks for few-shot video classification. *Computer Vision – ECCV 2018*, page 782–797, 2018.

[18] Zhenxi Zhu, Limin Wang, Sheng Guo, and Gangshan Wu. A closer look at few-shot video classification: A new baseline and benchmark. *CoRR*, abs/2110.12358, 2021.