HIERARCHICAL TEMPORAL MEMORY ENHANCED ONE-SHOT DISTANCE LEARNING FOR ACTION RECOGNITION

Yixiong Zou[1], *Yemin Shi*[1], *Yaowei Wang*[2*], *Yu Shu*[1], *Qingsheng Yuan*[3], *Yonghong Tian*[1*]

[1] National Engineering Laboratory for Video Technology, School of EE&CS, Peking University, Beijing, China
[2] School of Information and Electronics, Beijing Institute of Technology, Beijing, China
[3] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

ABSTRACT

One-shot action recognition is one of the most challenging tasks due to the very limited training samples. For one-shot video action recognition, randomly selected frames from cluttered frame features may result in a poor performance. To use the most valuable frames in a better feature space, this paper proposes Hierarchical Temporal Memory Enhanced One-shot Distance Learning (HED). Firstly, we introduce temporal triplet from different frames, so that the intra-class distance will be decreased while the inter-class distance will be increased. Secondly, the Hierarchical Temporal Memory (HTM), a biological plausible unsupervised model for sequence prediction, is employed to enhance the one-shot action recognition by finding the most valuable frames in a video sequence. Finally, the selected frames together with the temporal triplet trained model are used to get the corresponding category label. Extensive experiments conducted on three benchmark datasets (i.e UCF11, UCF50 and HMDB51) demonstrate that we can achieve significant improvement than the state-of-the-art methods.

Index Terms— One-shot Action recognition, Hierarchical Temporal Memory, Distance Learning

1. INTRODUCTION

Action recognition is growing to be an important research area in computer vision. Since Two-stream ConvNets [1], deep learning has shown its effectiveness on understanding human actions. Due to the complexity of CNNs, many challenging benchmark datasets have been proposed to train the model, such as UCF101 [2] and Kinetics [3]. However, as the datasets growing larger and larger, the recognition models are getting harder and harder to train, for example, C3D [4] is trained on over 1 million videos for two months.

However, human need only several samples to recognize an action. For example, even a child can recognize a basketball shooting action after watching it once. In order to learn knowledge more efficiently, one-shot learning is proposed by Feifei Li *et al.* [5] and has drawn researchers' attention to recognize an object given only one training sample. [6] separates classes to seen classes with multiple examples and unseen classes with single examples, and claims surpassing human performance in one-shot learning. [7] and [8] also proposed to apply deep learning on one-shot learning problem by learning a Neural Turing Machine. However, it still remains one of the most challenging tasks.

Inspired by these previous works on image recognition, this paper focus on one-shot action recognition. However, seldom deep learning based one-shot action recognition methods exist in this subfield. To the best of our knowledge, [9] may be the only deep learning based work performing one-shot learning on challenging action datasets like UCF11 [10] and HMDB51 [11]. Another similar subfield is one-shot gesture recognition. Akin to action recognition, one-shot gesture recognition is to recognize a semantic gesture given only one

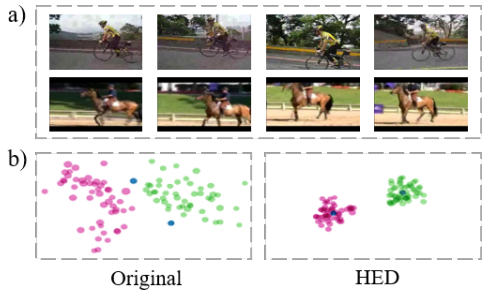

Fig. 1. (a) Actions are complicated and even frames from the same video can vary in illumination, background and view. (b) Simple visualization of frame distribution of two videos. Each dot represents a single frame and blue dot means selected frame. Left: as temporal variation exists, inter-class distance may be small and intra-class distance can be large. Sometimes random frame selection may make it hard to distinguish different actions. Right: frame distribution and selection after HED, which is obviously better than the left one.

Corresponding author: Yaowei Wang (email: yaoweiwang@bit.edu.cn), Yonghong Tian (email: yhtian@pku.edu.cn).

978-1-5386-1737-3/18/$31.00

training sample, such as [12] and [13]. However, none of these deep learning methods is effective enough.

Different from images, videos can be much more complicated with temporal complexity and variation. As shown in Fig. 1(a), a simple biking or horse riding action varies a lot between different frames within each video. The intra-class and inter-class distance is shown in Fig. 1(b), where dots in red and green represent frames in biking and horse riding respectively. The left part represents the original frame distribution and selection while the right part represents those using HED. And it can be very hard to distinguish actions by some randomly distributed features. Thus we employ distance learning method such as triplet loss [14] to decrease intra-class distance while increasing inter-class distance.

Another important part in video recognition is frame selection. In Fig. 1(b), blue dots represent the selected frames. Obviously, choosing the most discriminative frames in the right is much better than randomly choosing in the left.

Imagine that if you have watched an action multiple times, given a former part of it, you might be able to predict the latter part of it. However, if you are confronted with a never-before-seen video, you can't achieve it easily. For example, you can't correctly predict what an athlete will do after a dash when you never seen long jump before, but it is the wrongly predicted frame(jump after dash) that represents long jump the most. That is to say, the harder one frame can be predicted in a video, the more representative it is.

Follow this idea, we employ Hierarchical Temporal Memory(HTM) [15] in our work for frame selection. HTM is a promising unsupervised method for sequence prediction proposed by Hawkins, and is a biological plausible and even universal sequence learning architecture. Given a HTM model that has watched(trained on) seen-class videos for multiple times, for a frame in a never-before-seen unseen-class video, the less accurate HTM can predict it, the more valuable it is.

In this paper, we propose the Hierarchical Temporal Memory Enhanced One-shot Distance Learning(HED) for action recognition. We introduce temporal triplet so that we not only learn the classifier but also distance between categories. Then we propose to use HTM for frame selection. Finally, we use temporal triplet trained model and the selected frames to get the corresponding category label. Our contributions are mainly in two aspects. First, to the best of our knowledge, we are the first to use temporal triplet for distance learning on one-shot action recognition; second, we creatively introduce HTM to select frames for action recognition.

The rest of this paper is organized as following. In section 2, we introduce related works of this paper, then we propose our method in section 3. The experiment results are discussed in section 4. Finally, section 5 concludes this paper.

2. RELATED WORK

Basically, action recognition aims at categorizing the actions or behaviors in video sequences. Two-stream [1] is widely recognized as the baseline method and proposed to use two CNNs to capture appearance and motion feature. [16] proposed to combine spatial and temporal information with CNNs. [17] used warped optical flow image to assist original Two-Stream recognition. And [18] proposed to aggregate spatial-temporal features with LSTMs. Another group of researchers use 3-dimensional CNNs(C3D) to extract spatial-temporal features directly, such as [4] and [19].

To use training data more efficiently, one-shot learning was introduced to train models given only one training sample. [5] proposed to use Bayesian approach to explore probability distribution given only one sample. Follow this idea, [6] introduced another Bayesian approach to recognize real world's alphabet and proposed a frequently used dataset, Omniglot. Deep learning also plays an important role in this field. In [7], Deepmind first proposed to use Memory-Augmented Neural network and meta-learning to train a deep network, which treated one-shot learning problem as a Neural Turing Machine learning problem. After that, Deepmind improved its Turing Machine architecture in [8] and [20]. Based on these ideas, many deep learning based networks emerges. [21] proposed to train a network with a meta network for better transferring. [22] proposed to use GAN to augment training samples in recognition. And [23] proposed to use RNN to learn from each pair of examples repeatedly.

To recognize an action given few training data, this paper focus on one-shot action recognition. [9] proposed to train Matching Network [8] with a Neural Turing Machine on UCF11 and HMDB51. Another related subfield is one-shot gesture recognition. [12] learned from the Kinect trajectories of human and then augmented such trajectories using GMM. [24] also used Kinect trajectories to learn a HMM model and train SVMs for classification.

3. HIERARCHICAL TEMPORAL MEMORY ENHANCED ONE-SHOT DISTANCE LEARNING

For clarity, we denote classes with multiple training examples as seen classes and those with single training examples as unseen classes, and test will be conducted on unseen classes. For one-shot learning, training examples in both seen classes and unseen classes can be used. Thus our method is divided into two stages to utilize these two kinds of classes respectively.

Framework of our method is shown in Fig. 2. As you can see, in seen-class training stage, we train our model using multiple examples by Matching Network, and also train HTM model with these data. In unseen-class training stage, we train our model using single example by temporal triplets. After training, we test our model on unseen classes with HTM frame selection in testing stage. In the following subsections

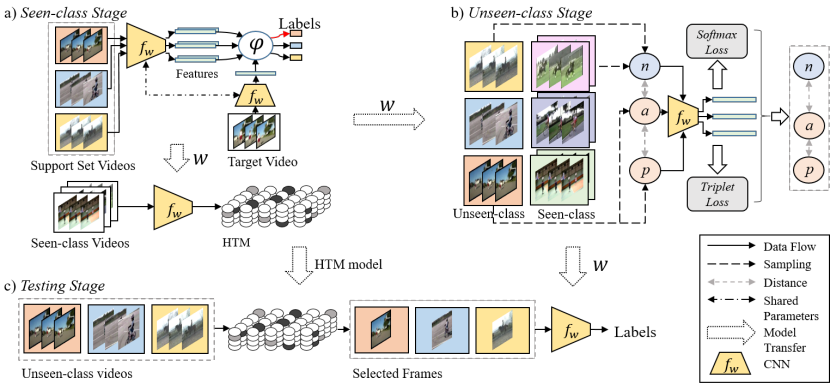

Fig. 2. Framework of HED. The training phase has two stages. (a) Seen-class stage: we first use Matching Network to train on seen classes, and then use trained model(w) to train HTM model. (b) Unseen-class stage: single sample training using temporal triplets which is fine-tuned from seen-class trained model(w). a means anchor sample, p means positive sample and n means negative sample. (c) Testing Stage: testing is performed on unseen classes **with** HTM frame selection and unseen-class trained model(w).

we will describe these in detail.

3.1. Seen-Class Stage

In order to describe every frame, a CNN is trained on the seen classes, which have been proved to achieve the state-of-the-art performance on many areas. And the Matching Network is used to learn the similarity.

As shown in Fig. 2(a), in Matching Network, we sample a support set containing K videos and one target video at each iteration, which will be referred as $S = \{(x_i, y_i)\}_{i=1}^{K}$ and $T = \{\hat{x}\}$ respectively, and x_i and $\hat{x}$ means videos in each class. y_i means one-hot label of x_i. Let $\hat{y}$ denotes the probability over all classes in support set, Matching Network tries to compute

$$P(\hat{y}|\hat{x}, S) = \sum_{i=1}^{K} \varphi(f_w(\hat{x}), f_w(x_i))y_i \tag{1}$$

where $\varphi(\cdot)$ denotes an attention mechanism and f_w denotes a feature extractor which may be a CNN with parameter w to be learned. Eq.1 is actually a linear combination over labels in support set. Attention mechanism can be represented as

$$\varphi(f_w(\hat{x}), f_w(x_i)) = softmax(c(f_w(\hat{x}), f_w(x_i))) \tag{2}$$

where $c(\cdot)$ is a similarity measure function which is typically a cosine distance. With computed probability over all classes, we can use softmax loss to train the model and use accuracy to measure performance.

Another focus of HED is transferability. Deep learning, as a method with numerous parameters to train, is easy to over-fit the training data. Compared with deep learning, some unsupervised method with less parameters can perform better when transferring across datasets. As shown in Fig. 1, frame

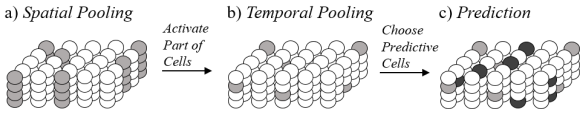

Fig. 3. HTM can be divided into three stages. a) Spatial Pooling, generate sparse binary code of input , which will activate columns; b) Temporal Pooling, activate a part of cells within active columns; c) Prediction and anomaly detection, active cells in temporal pooling stage will activate some cells in other columns, which can be viewed as a prediction.

selection is also an important part in action recognition. As illustrated in section 1, the harder one frame can be predicted, the more representative it is. Considering the success of unsupervised HTM in sequence prediction, we use it to perform frame selection in testing stage with training on seen classes.

Hierarchical Temporal Memory(HTM) is proposed by NUPIC in [15], which is shown in Fig. 3. One HTM layer is composed of multiple columns and one column contains multiple cells. Each cell has three states, inactive, active and predictive state. For training and testing, HTM can be divided into three stages. In spatial pooling stages, sparse binary code will be generated so as to activate columns in a layer, which can be described in Fig. 3(a). In temporal pooling stage, as spatial pooling activate the whole column, only a few cells will be chosen to keep active within activated columns, which makes it able to tolerate easy-to-confused elements. Cells connected to active cells will go to predictive state, for time $t-1$, predictive cells can predict active columns in time t. Thus the anomaly score can be computed as the difference between prediction and the real future as described by

$$score = \frac{|A_t - (P_{t-1} \cap A_t)|}{|A_t|} \tag{3}$$

where A_t denotes the real active columns of time t and P_{t-1} denotes predictive cells of time $t-1$.

We first train HTM on seen-class videos , then in testing stage input unseen-class videos and target video for HTM to calculate anomaly score of each frame, and finally select the most anomaly frames for Matching Network to test on, as shown in Fig.2(c).

3.2. Unseen-Class Stage

Videos are different from static images because of their temporal variation, and using temporal information has been proved to be effective in action recognition. As shown in Fig. 1, temporal variation can influent performance on a large scale, thus it is vital to increase inter-class distance between classes and decrease intra-class distance within classes.

Most works on image one-shot learning do not train their model on the one-shot classes because of easily being overfitting. They mostly train on seen classes and extract feature on unseen classes for distance comparison. Unlike traditional methods, we propose to train a distance model on both seen and unseen classes so as to take full advantage of the one unseen-class sample.

One shot learning is actually an distance learning task when using deep learning. Therefore, methods for distance learning can be used here. In this paper, we introduce the triplet loss to one-shot action recognition problem. The definition of our triplet loss is as following:

$$max\{[\sum_{i}^{N} ||f_w(z_i^a) - f_w(z_i^p)||_2^2 - ||f_w(z_i^a) - f_w(z_i^n)||_2^2 + \beta], 0\} - (ln(P(z_i^p)) + ln(1 - P(z_i^n))) \quad (4)$$

where z_i denotes the sampled clip from videos, z_i^a denotes anchor sample, z_i^p denotes positive sample, z_i^n denotes negative sample, $||_2^2$ denotes L2 normalized Euclidean distance and β is a predefined parameter.

Moreover, we also add a cross entropy loss term on it so as to make training more stable. And probability of positive and negative samples are computed as

$$P(z_i) = softmax(cos(f_w(z_i^a), f_w(z_i))) \quad (5)$$

where z_i is the positive or negative sample.

For each unseen class, which has only one video sample, we randomly sample two clips as anchor and positive sample respectively, and a clip from any other unseen classes or seen classes is used as negative sample, as shown in Fig.2(b).

4. EXPERIMENT

Implementation details and comparison with state-of-the-art on UCF11, HMDB51 and UCF50 will be introduced in the following subsections.

4.1. Implementation details

We first train on seen classes using Matching Network without HTM selection. Given a N-way classification job, for each forward and backward iteration, we need to sample a K-class support set S and a target video T from seen classes, in which K equals the number of classes to test. Repeat it r times and get the seen-class trained model. In order to train HTM, we first randomly select videos from seen classes and then extract features by the seen-class trained model, then use them to train HTM for 40 iterations. After that we need to train and test on unseen classes. We first independently sample S and T, then use samples within each unseen-class video to generate anchor and positive examples, and use samples from different unseen-class videos and seen-class videos to generate negative examples. After iteration for 10 to 20 times, we test on this S and T with HTM selecting 1 frame as the begin index for uniform frame sampling, as shown in Fig.2(c). As HTM selected frame index can also represent the most representative segment of a video, for each video we sample 5 frames with stride 10 starting with the frame selected by HTM. To get a mean performance and reduce the effect of each selection, we repeat the unseen-class training and testing procedure for thousands of times. Because sampling S and T over the whole dataset can cover enough combinations of training data and testing data, this kind of testing methods is itself a cross-validation test.

Considering limited training samples in unseen-class training, we choose to use GoogleNet [25] as f_w which has fewer parameters to train. For the same reason in Matching Network we abandoned fully conditional embeddings(FCE) which needs LSTM. We set K in support set to be the number of classes to test on unseen classes. For example, if on unseen classes we need to test a 5-way performance, we will set K to 5. For HTM, as we find that GoogleNet feature is good enough to be spatial pooling output, we set the top 20 elements in feature vector to 1 and the rest of them to 0, so as to keep sparsity (20/1024≈2%).

4.2. Experiment Results

4.2.1. Comparison with the state-of-the-art

We test HED on UCF11 and HMDB51 to compare with [9], and on UCF50 to compare with results in [26] and [27].

UCF11 is a challenging dataset containing 11 actions and 1600 videos in total. UCF50 is an extension of UCF11, which has 50 actions and 6681 videos. Compared with UCF11 and UCF50, HMDB51 may be even harder as variation in each classes is greater, and it contains 51 actions and 6766 clips.

We compare HED with [9] in table 1. HED_T means only temporal triplet is used, HED_H means only HTM is used, and HED_A means all modules are in use. In [9], they randomly split 6 classes for training and 5 classes for testing on UCF11, and randomly split 41 classes for training and 10

Table 1. 5-way one-shot performance. HED_T means only temporal triplet is used, HED_H means only HTM is used, and HED_A means all modules are in use.

Method	UCF11(%)	HMDB51(%)
Method in [9]	42.1	42.9
Matching Network	53.2	44.2
HED_T	59.7	46.9
HED_H	59.7	46.2
HED_A	**60.3**	**47.1**

classes for testing on HMDB51. 5-way one-shot experiments are performed. We follow their experiments settings and outperform them significantly by about 18% on UCF11 and 5% on HMDB51.

We also report [26] and [27] on UCF50 in Table 2. Simple Baseline means using GoogleNet to extract feature for distance comparison directly. [27] uses transfer learning to transfer knowledge from simple action datasets such as KTH to UCF50. However, this isn't a one-shot learning task, and they report classification performance with using 5% training samples. [27] performs a 10-fold cross-validation, which means using 6681*0.9*0.05=6.0129 samples per class in average. To compare with [27] fairly, we use 6-shot performance on 50-way classification task without seen-class Matching Network training. For HTM training, because of covered classes between KTH and UCF50, we do not use KTH. Instead, we choose UCF101 dataset excluding 50 classes in UCF50, which results in 51 classes, and random select only 2 samples per class, so that our training data won't exceed that of [27]. In this setting, all negative samples are selected from unseen classes because of no seen classes. As shown in Table 2, we outperform both [27] and Dense Trajectories [26] performance, which proves the effectiveness of HED.

4.2.2. HED Analysis

In both Table.1 and Table.2 we can find the performance of HED_T, which only uses HED without HTM frame selection. To show the contribution of temporal triplet, we can compare result of HED_T with Matching Network in Table.1 and HED_T with Simple Baseline in Table.2. As we can see, on all three datasets, HED_T outperforms its baselines. HMDB51 is relatively harder than UCF11, and the improvement on it is relatively smaller than that on UCF11. Among three datasets, although 50-way one-shot classification is the most difficult task, performance of HED on UCF50 promotes significantly. At the same time, 50-way and 6-shot also means that for each one-shot test, UCF50 has 300 labeled unseen videos to train, which exceeds UCF11 and HMDB51 by a large margin. Thus temporal triplet will have more data, which makes it promotes more than on the other two datasets. To show the contribution of HTM, we compare HED_H with Matching Network in Table.1 and with Simple Baseline in Table.2. As we can see, on all three datasets HTM promotes a lot, especially on UCF11, which is relatively simpler than the other two datasets.

Table 2. Performance on UCF50 with 5% training samples without Matching Network training. Simple Baseline means extract feature with Googlenet without seen-class training.

Method	UCF50(%)
Dense Trajectories [26]	43.9 ± 1.9
SCA-TLM_H [27]	48.1 ± 0.9
SCA-TLM_A [27]	50.3 ± 0.9
6-shot Simple Baseline	50.4
6-shot HED_T	57.8
6-shot HED_H	56.8
6-shot HED_A	**58.1**

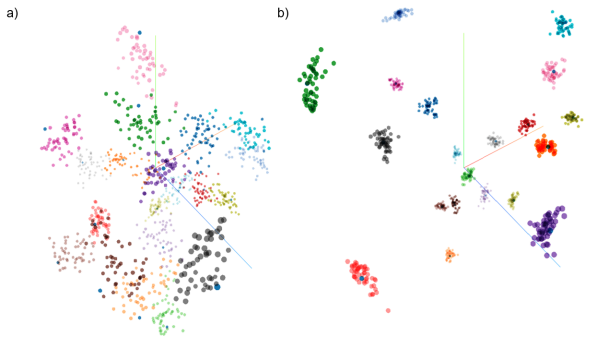

Fig. 4. Visualization of HED, each dot represents a frame and 20 videos are plotted. (a) Frame distribution before HED. (b) Frame distribution after HED. Random selections and HTM selections are represented by blue dot within each cluster in (a) and (b) respectively. As we can see, inter-class distance is larger and intra-class distance is smaller than (a), and HTM can approximately select the most representative frame .

Visualization of HED can be seen in Fig.4. Here we randomly sample 20 videos from UCF50 and plot frame distribution before and after HED by dots and this is done by TensorBoard [28]. As you can see, in Fig.4(b), inter-class distance is larger and intra-class distance is smaller than (a), thus we can find distinct clusters in Fig.4(b). Random selections and HTM selections are shown by blue dots within each cluster in Fig.4(a)(b) respectively, and it shows that HTM can approximately find the most representative frames in videos. As HTM selected frame index can also represent the most representative video segment, HED benefits from HTM a lot.

5. CONCLUSION

In this paper, we proposed an Hierarchical Temporal Memory enhanced one-shot distance learning (HED) for action recog-

nition, which consists of seen-class supervised training using Matching Network and unsupervised training using HTM, unseen-class training using temporal triplet, and testing using HTM for frame selection. We compared our method with the state-of-the-art one-shot action recognition methods and transfer learning methods with few training samples. Experiments on UCF11, UCF50 and HMDB51 show that we can achieve significant improvement than the state-of-the-art methods on all three datasets.

Acknowledgment. This work is partially supported by This work is partially supported by grants from the National Key R&D Program of China under grant 2017YFB1002401, the National Natural Science Foundation of China under contract No. U1611461, No. 61390515, No. 61425025, No. 61471042 and No. 61650202.

6. REFERENCES

[1] Karen Simonyan and Andrew Zisserman, "Two-stream convolutional networks for action recognition in videos," in *NIPS*, 2014.

[2] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv:1212.0402*, 2012.

[3] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al., "The kinetics human action video dataset," *arXiv:1705.06950*, 2017.

[4] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *ICCV*, 2015.

[5] Li Fei-Fei, Rob Fergus, and Pietro Perona, "One-shot learning of object categories," *TPAMI*, vol. 28, no. 4, 2006.

[6] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, 2015.

[7] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap, "One-shot learning with memory-augmented neural networks," *arXiv:1605.06065*, 2016.

[8] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al., "Matching networks for one shot learning," in *NIPS*, 2016.

[9] Daesik Kim, Myunggi Lee, and Nojun Kwak, "Matching video net: Memory-based embedding for video action recognition," in *IJCNN*, 2017.

[10] Jingen Liu, Jiebo Luo, and Mubarak Shah, "Recognizing realistic actions from videos in the wild," in *CVPR*. IEEE, 2009.

[11] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre, "Hmdb: a large video database for human motion recognition," in *ICCV*. IEEE, 2011.

[12] Maria Eugenia Cabrera and Juan Pablo Wachs, "A human-centered approach to one-shot gesture learning," *Front. Robot. AI 4: 8. doi: 10.3389/frobt*, 2017.

[13] Tomas Pfister, James Charles, and Andrew Zisserman, "Domain-adaptive discriminative one-shot learning of gestures," in *ECCV*. Springer, 2014.

[14] Alexander Hermans, Lucas Beyer, and Bastian Leibe, "In defense of the triplet loss for person re-identification," *arXiv:1703.07737*, 2017.

[15] J. Hawkins, S. Ahmad, S. Purdy, and A. Lavin, "Biological and machine intelligence (bami)," Initial online release 0.4, 2016.

[16] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman, "Convolutional two-stream network fusion for video action recognition," in *CVPR*, 2016.

[17] 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," in *ECCV*. Springer, 2016.

[18] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici, "Beyond short snippets: Deep networks for video classification," in *CVPR*, 2015.

[19] Joao Carreira and Andrew Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," *arXiv:1705.07750*, 2017.

[20] Danilo Rezende, Ivo Danihelka, Karol Gregor, Daan Wierstra, et al., "One-shot generalization in deep generative models," in *ICML*, 2016.

[21] Luca Bertinetto, João F Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi, "Learning feed-forward one-shot learners," in *NIPS*, 2016.

[22] Akshay Mehrotra and Ambedkar Dukkipati, "Generative adversarial residual pairwise networks for one shot learning," *arXiv:1703.08033*, 2017.

[23] Pranav Shyam, Shubham Gupta, and Ambedkar Dukkipati, "Attentive recurrent comparators," *arXiv:1703.00767*, 2017.

[24] Maria E Cabrera, Natalia Sanchez-Tamayo, Richard Voyles, and Juan P Wachs, "One-shot gesture recognition: One step towards adaptive learning," in *FG*. IEEE, 2017.

[25] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.

[26] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu, "Action recognition by dense trajectories," in *CVPR*. IEEE, 2011.

[27] Fang Liu, Xiangmin Xu, Shuoyang Qiu, Chunmei Qing, and Dacheng Tao, "Simple to complex transfer learning for action recognition," *TIP*, vol. 25, no. 2, 2016.

[28] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv:1603.04467*, 2016.