Action Recognition with Knowledge Transfer

Jinwoo Choi

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

Jia-Bin Huang, Chair
A. Lynn Abbott
Harpreet. S. Dhillon
Bert Huang
Gaurav Sharma

December 10, 2020
Blacksburg, Virginia

Keywords: Computer Vision, Machine Learning, Deep Learning, Convolutional Neural Networks, Representation Learning, Action Recognition, Domain Adaptation, Bias Reduction, Semi-Supervised Learning, Unsupervised Learning

Action Recognition with Knowledge Transfer

Jinwoo Choi

(ABSTRACT)

Recent progress on deep neural networks has shown remarkable action recognition performance from videos. The remarkable performance is often achieved by transfer learning: training a model on a *large-scale labeled* dataset (source) and then fine-tuning the model on the small-scale labeled datasets (targets). However, existing action recognition models do not always generalize well on new tasks or datasets because of the following two reasons. i) Current action recognition datasets have a spurious correlation between action types and background scene types. The models trained on these datasets are biased towards the scene instead of focusing on the actual action. This scene bias leads to poor generalization performance. ii) Directly testing the model trained on the source data on the target data leads to poor performance as the source, and target distributions are different. Fine-tuning the model on the target data can mitigate this issue. However, manual labeling small-scale target videos is labor-intensive. In this dissertation, I propose solutions to these two problems. For the first problem, I propose to learn *scene-invariant* action representations to mitigate the scene bias in action recognition models. Specifically, I augment the standard cross-entropy loss for action classification with 1) an adversarial loss for the scene types and 2) a human mask confusion loss for videos where the human actors are invisible. These two losses encourage learning representations unsuitable for predicting 1) the correct scene types and 2) the correct action types when there is no evidence. I validate the efficacy of the proposed method by transfer learning experiments. I transfer the pre-trained model to three different tasks, including action classification, temporal action localization, and spatio-temporal action detection. The results show consistent improvement over the baselines for every task and dataset. I formulate human action recognition as an unsupervised

domain adaptation (UDA) problem to handle the second problem. In the UDA setting, we have many labeled videos as source data and *unlabeled* videos as target data. We can use already existing labeled video datasets as source data in this setting. The task is to align the source and target feature distributions so that the learned model can generalize well on the target data. I propose 1) aligning the more important temporal part of each video and 2) encouraging the model to focus on action, not the background scene, to learn domain-invariant action representations. The proposed method is simple and intuitive while achieving state-of-the-art performance without training on a lot of labeled target videos. I relax the unsupervised target data setting to a *sparsely labeled* target data setting. Then I explore the semi-supervised video action recognition, where we have a lot of labeled videos as source data and sparsely labeled videos as target data. The semi-supervised setting is practical as sometimes we can afford a little bit of cost for labeling target data. I propose multiple video data augmentation methods to inject photometric, geometric, temporal, and scene invariances to the action recognition model in this setting. The resulting method shows favorable performance on the public benchmarks.

Action Recognition with Knowledge Transfer

Jinwoo Choi

(GENERAL AUDIENCE ABSTRACT)

Recent progress on deep learning has shown remarkable action recognition performance. The remarkable performance is often achieved by transferring the knowledge learned from existing large-scale data to the small-scale data specific to applications. However, existing action recognition models do not always work well on new tasks and datasets because of the following two problems. i) Current action recognition datasets have a spurious correlation between action types and background scene types. The models trained on these datasets are biased towards the scene instead of focusing on the actual action. This scene bias leads to poor performance on the new datasets and tasks. ii) Directly testing the model trained on the source data on the target data leads to poor performance as the source, and target distributions are different. Fine-tuning the model on the target data can mitigate this issue. However, manual labeling small-scale target videos is labor-intensive. In this dissertation, I propose solutions to these two problems. To tackle the first problem, I propose to learn scene-invariant action representations to mitigate background scene-biased human action recognition models for the first problem. Specifically, the proposed method learns representations that cannot predict the scene types and the correct actions when there is no evidence. I validate the proposed method's effectiveness by transferring the pre-trained model to multiple action understanding tasks. The results show consistent improvement over the baselines for every task and dataset. To handle the second problem, I formulate human action recognition as an unsupervised learning problem on the target data. In this setting, we have many labeled videos as source data and *unlabeled* videos as target data. We can use already existing labeled video datasets as source data in this setting. The task is to align the source and target feature distributions

so that the learned model can generalize well on the target data. I propose 1) aligning the more important temporal part of each video and 2) encouraging the model to focus on action, not the background scene. The proposed method is simple and intuitive while achieving state-of-the-art performance without training on a lot of labeled target videos. I relax the unsupervised target data setting to a *sparsely labeled* target data setting. Here, we have many labeled videos as source data and sparsely labeled videos as target data. The setting is practical as sometimes we can afford a little bit of cost for labeling target data. I propose multiple video data augmentation methods to inject color, spatial, temporal, and scene invariances to the action recognition model in this setting. The resulting method shows favorable performance on the public benchmarks.

Dedication

To love of my life, Sophie, and my lovely daughter Ella.

Acknowledgments

I am immensely grateful to my advisor Jia-Bin Huang who has been an excellent mentor and teacher for me. Without his encouragement, guidance, and mentorship, none of this work would be possible. I am also very grateful to Gaurav Sharma, my internship mentor, and a Ph.D. dissertation committee member. Without his constructive feedback and discussion, the majority of my work would not be possible. I enjoyed collaborating with him. I also appreciate my Ph.D. dissertation committee - A. Lynn Abbott, Harpreet S. Dhillon, Bert Huang, for their valuable feedback and advice.

I have been fortunate to have many great collaborators: Yuliang Zou, Chen Gao, Joseph Messou, and Qitong Wang. I also thank all of our current and previous group members: Shih-Yang Su, Badour AlBahar, Esther Robb, Lowell Weissman, Sanket Lokegaonkar, Adithya Nallabolu, Subhashree Radhakrishnan, Ting-I Hsieh, Meng-Li Shih, Jin-Dong Dong, Yun-Chun Chen, Chieh Hubert Lin, Wei-Yu Chen, Yen-Chen Lin, and Hao-Wei Yeh. I have also been fortunate to collaborate with Samuel Schulter, Manmohan Chandraker during my internship at NEC Labs America for many insightful discussions, help, and advice.

I also thank my life-long friends who have been supported, loved, and encouraged me along the long journey of my Ph.D.: Sungyong Seo, Jaewon Shin, Sungyeon Kim, Sunghoon Kim, Jaesik Ahn, Geunyoung Oh, Sinyoung Hwang, Kijin An, Eunkyung Shin, Yoonchang Sung, Jongwan Kim, Sungjae Ohn, Hubert Kim, Byung-Jun Kim, Changkoo Kang, and Yegyu Han.

I would like to thank my family, especially - my parents, parents-in-law, sisters-in-law, brothers-in-law, for their love, support, and encouragement. Last but not least, I dedicate this dissertation to my wife, Soongha Hwang, and our lovely child, Ella Sia Choi, for their tireless love, support, and sacrifices.

Contents

List of Figures

List of Tables

List of Abbreviations

CNN Convolutional neural networks

DA Domain adaptation

GRU Gated recurrent unit

LSTM Long short-term memory

NN Neural networks

RNN Recurrent neural networks

SSL Semi-supervised learning

UDA Unsupervised domain adaptation

Chapter 1

Introduction

Understanding human action from videos has been a fundamental and longstanding problem in computer vision. Developing an effective video action recognition approach can lead to many interesting real-world applications such as autonomous driving, health care, surveillance, sports analytics, content-based video search, video recommendation system, augmented/virtual reality, entertainment, etc.

Recently, convolutional neural networks (CNN) have shown impressive performance on the video action recognition task. For example, state-of-the-art 3D CNNs can achieve $\sim 80\%$ top-1 accuracy on the Kinetics-400 [80] dataset which consists of diverse 400 action classes [18, 43, 145, 170]. The reason for the current success is the availability of a massive amount of *labeled* training data. The Kinetics-400 dataset consists of $240K$ labeled training videos with the ~ 10 seconds average length. With this massive amount of labeled videos, we can learn good action representations. However, in the real world, we could have new video data of interest according to the applications. We desire models to recognize what is going on in the new videos. Therefore, we need to train a model on the new dataset. However, it is prohibitively expensive to manually label a massive amount of training videos of every new video dataset due to the high labeling cost for videos.

A widely used solution in computer vision is *transfer learning*. i.e., We transfer the knowledge learned from the already existing and labeled training videos, which we call the source dataset, to the new dataset, which we call the target dataset, of our interest. We train a model on the source video dataset. Then we use the model weights trained on source data as an initialization of the

target fine-tuning. By transfer learning, we can leverage large-scale source data to boost target performance even if the target data is relatively smaller than the source data. For example, ResNet-101 [62] trained on the large-scale, labeled source dataset ImageNet [35], can be transfered to Pascal VOC object detection [40], which is smaller scale. The transfer learned ResNet-101 shows high mean average precision of 76.4% on the VOC 07 split and 73.8% on the VOC 12 test split. Similarly, in video domain, Kinetics-400 pre-trained models [18, 43, 145, 170] are transfered to UCF-101 [140], HMDB-51 [85] which are small-scale target datasets. The transfer learned video recognition models achieve high accuracy on the target, e.g., R(2+1)D-RGB achieves 96.8% top-1 accuracy on the UCF-101, 74.5% top-1 accuracy on the HMDB-51.

However, transfer learning existing action recognition models may not always generalize well on the new tasks and datasets because of two problems. i) The source dataset, e.g., Kinetics, is biased towards the scene, objects, and human appearance [95]. The models trained on these datasets are biased towards the scene instead of focusing on the actual action. This scene bias leads to poor generalization performance. ii) Directly testing the model trained on the source data on the target data leads to poor performance as the source, and target distributions are different. Fine-tuning the model on the target data can mitigate this issue. However, labeling a relatively small-scale target dataset is still labor-intensive, as humans need to watch videos thoroughly. In this dissertation, I tackle these two problems hampering video action recognition models' generalization performance. To tackle the problems, I address the following three major topics.

Mitigating scene bias in action recognition [28]. Human activities often occur in specific scene contexts, e.g., playing basketball on a basketball court. Training a model using existing video datasets thus inevitably captures and leverages such bias (instead of using the actual discriminative cues). The learned representation may not generalize well to new action classes or different tasks. In this work, we propose to mitigate scene bias for video representation learning. Specifically, we augment the standard cross-entropy loss for action classification with 1) an adversarial loss for

scene types and 2) a human mask confusion loss for videos where the human actors are masked out. These two losses encourage learning representations that are unable to predict the scene types and the correct actions when there is no evidence. We validate the effectiveness of our method by transferring our pre-trained model to three different tasks, including action classification, temporal localization, and spatio-temporal action detection. Our results show consistent improvement over the baseline model without debiasing.

Unsupervised domain adaptation for video action recognition [29, 30]. We address the problem of domain adaptation in videos for the task of human action recognition. Inspired by image-based domain adaptation, we can perform video adaptation by aligning the features of frames or clips of source and target videos. However, equally aligning all clips is sub-optimal as not all clips are informative for the task. As the first novelty, we propose an attention mechanism which focuses on more discriminative clips and directly optimizes for video-level (cf. clip-level) alignment. As the backgrounds are often very different between source and target, the source background-corrupted model adapts poorly to target domain videos. To alleviate this, as a second novelty, we propose to use the clip order prediction as an auxiliary task. The clip order prediction loss, when combined with domain adversarial loss, encourages learning of representations which focus on the humans and objects involved in the actions, rather than the uninformative and widely differing (between source and target) backgrounds. We empirically show that both components contribute positively towards adaptation performance. We report state-of-the-art performances on two out of three challenging public benchmarks, two based on the UCF and HMDB datasets, and one on Kinetics to NEC-Drone datasets. We also support the intuitions and the results with qualitative results.

Semi-supervised learning for video action recognition. We tackle the video action recognition task in a low-data regime, where only a small amount of labeled examples are available during training. Recent semi-supervised learning methods show improved accuracy by enforcing the prediction consistency on a large amount of unlabeled data. One of the critical factors is to design

strong, diverse data augmentation strategies that capture the various visual invariances. Compared to the image domain, data augmentations for videos are under-explored. Applying data augmentation strategies for images to each video frame individually without considering the temporal structure of videos leads to sub-optimal performance. In this work, we investigate data augmentation strategies for the video domain from several different perspectives, including photometric, geometric, temporal, and actor/scene. We show that our proposed data augmentation strategy fits into the state-of-the-art SSL framework [139] and leads to promising performance on the UCF-101 and HMDB-51 datasets in the low labeled data regime.

The rest of the dissertation is organized as follows. Chapter 2 starts with the problem of scene bias in action recognition, which is a problem of source datasets in transfer learning. In Chapters 3 and 4, I tackle the problem of mitigating the target data's high labeling cost by formulating the problem as unsupervised learning on the target data. Then I relax the unsupervised setting to a semi-supervised setting in Chapter 5. I conclude remarks in Chapter 6.

The relevant publication list for this dissertation is as follows:

- (Chapter 2) Why Cant I Dance in the Mall? Learning to Mitigate Scene Bias in Action Recognition [28], NeurIPS 2019. (Poster)
- (Chapter 3) Unsupervised and Semi-Supervised Domain Adaptation for Action Recognition from Drones [29], WACV 2020. (Video)
- (Chapter 4) Shuffle and Attend: Video Domain Adaptation [30], ECCV 2020. (Video)
- (Chapter 5) What Makes A Good Data Augmentation for Semi-Supervised Video Action Recognition?, under submission to CVPR 2021.

Chapter 2

Mitigating Unwanted Biases in Source Data

Figure 2.1: **Quiz time!** Can you guess what action the (blocked) person is doing in the four videos? Even though we cannot see a human actor, we can easily predict the action by considering where the scene is. Training a CNN model from these examples may lead to a strong bias toward recognizing the scene or the objects present in the video as opposed to paying attention to the actual action the person is doing. In this work, we show that learning video representation with debiasing leads to improved generalization to novel classes and tasks.

2.1 Introduction

Convolutional neural networks (CNNs) [18, 145, 162, 170] have demonstrated impressive performance on action recognition datasets such as the Kinetics [80], UCF-101 [140], HMDB-51 [86], and others. These CNN models, however, may sometimes make correct predictions for wrong reasons, such as leveraging scene context or object information instead of focusing on actual human actions in videos. For example, CNN models may recognize a classroom or a whiteboard in an input video and predict the action in the video as *giving a lecture*, as opposed to paying attention to the actual activity in the scene, which could be, for example, *eating*, *jumping* or even *dancing*. Such biases are known as representation bias [95]. In this chapter, I focus on mitigating the effect of *scene* representation bias [95]. Following Li et al. [95], I define the scene representation bias of

a dataset D as

$$B_{\text{scene}} = \log[M(D, \phi_{\text{scene}})/M_{\text{rand}}]. \tag{2.1}$$

Here, ϕ_{scene} is a scene representation; $M(D, \phi_{\text{scene}})$ is an action classification accuracy with a scene representation ϕ_{scene} on the dataset D; M_{rand} is a random chance action classification accuracy on the dataset D. With this definition, I can now measure the scene representation bias of the UCF-101 [140] dataset by computing the log ratio between two accuracies: 1) the action classification accuracy of ResNet-50 backbone pre-trained on the Places365 dataset [188] on UCF-101: 59.7%. 2) the random chance accuracy on UCF-101: 1.0%. To get the first accuracy, a linear classifier is trained on UCF-101 on top of the Places365 pre-trained ResNet-50 feature backbone. As an input to the linear classifier, I use a $2,048$ dimensional feature vector extracted from the penultimate layer of the ResNet-50 feature backbone. The scene representation bias of UCF-101 is $\log(59.7/1.0) = 4.09$. A completely unbiased dataset would have $\log(1.0/1.0) = 0$ scene representation bias. Thus, the UCF-101 dataset has quite a large scene representation bias.

The reason for a scene representation bias is that human activities often occur in specific scene contexts (e.g., playing basketball in a basketball court). Figure 2.1 provides several examples. Even though the actors in these videos are masked-out, I can still easily infer the actions of the masked-out actors by reasoning about where the scene is. As a result, training CNN models on these examples may capture the biases towards recognizing scene contexts. Such strong scene bias could make CNN models unable to generalize to unseen action classes in the same scene context and novel tasks.

In this chapter, I propose a debiasing algorithm to mitigate scene bias of CNNs for action understanding tasks. Specifically, I pre-train a CNN on an action classification dataset (Mini-Kinetics-200 dataset [170] in the experiment) using the standard cross-entropy loss for the action labels. To

Figure 2.2: **Motivation of the proposed debiasing algorithm.** (*Left*): A man is ***singing*** in a baseball field. However, representations with certain bias toward the scene may predict the incorrect action e.g., , *playing baseball*. (*Right*): A person (masked-out) is ***swimming*** in a swimming pool. A model is capable of predicting the correct action i.e., *swimming* even without looking at the evidence. Video representations that make correct (or incorrect) predictions by leveraging the scene bias may not generalize well to unseen action classes and tasks.

mitigate scene representation bias, I introduce two additional losses: (1) *scene adversarial loss* that encourages a network to learn scene-invariant feature representations and (2) *human mask confusion loss* that prevents a model from predicting an action if humans are not visible in the video. I validate the proposed debiasing method by showing transfer learning results on three different activity understanding tasks: action classification, temporal action localization, and spatio-temporal action detection. The debiased model shows the consistent performance improvement of transfer learning over the baseline model without debiasing across various tasks.

I make the following three contributions in this chapter.

- I tackle a relatively under-explored problem of mitigating scene biases of CNNs for better generalization to various action understanding tasks.
- I propose two novel losses for mitigating scene biases when pre-training a CNN. I use a scene-adversarial loss to obtain scene-invariant feature representation. I use a human mask confusion loss to encourage a network to be unable to predict correct actions when there is no visual evidence.
- I demonstrate the effectiveness of the method by transferring the pre-trained model to three

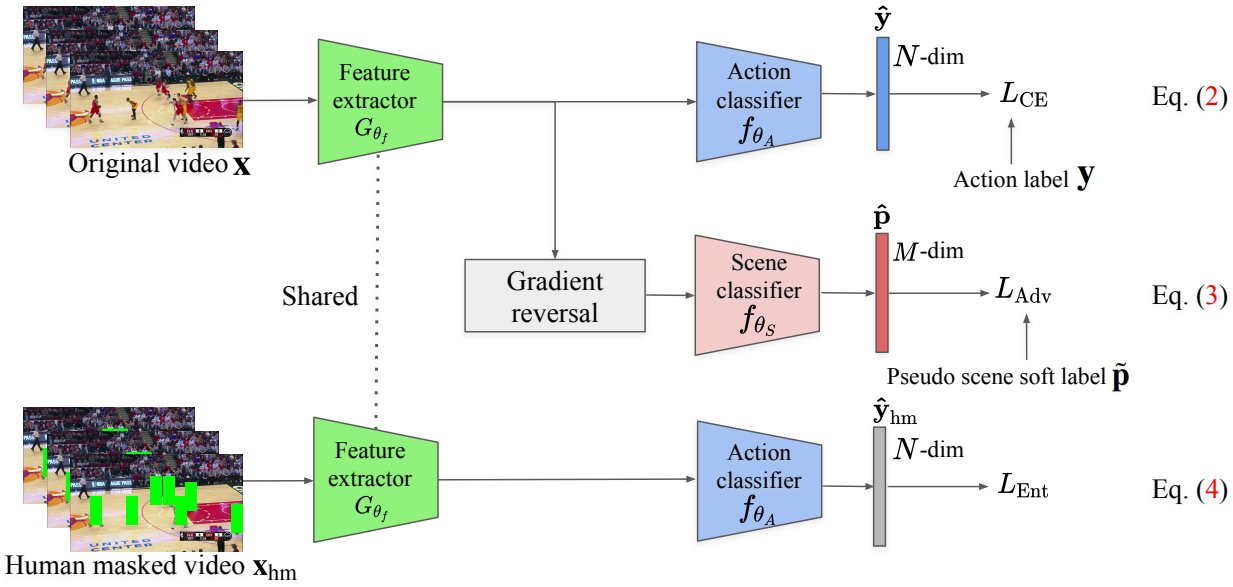


Figure 2.3: **Overview of the proposed approach for learning debiased video representation.** Here, our goal is to learn the parameters θ_f for the feature extractor ϕ by pre-training it on a large-scale video classification task. Our training involves three types of losses. First, we use a standard cross-entropy loss L_{CE} for training action classification. Second, we impose a scene adversarial loss L_{Adv} so that one cannot infer the scene types based on the learned representation. Third, we prepare additional videos by detecting and masking out the humans using an off-the-shelf human detector. We apply an entropy loss L_{Ent} on the predicted action class distributions of the human-masked-out videos. Our intuition here is that the model should not be able to infer the correct action without seeing the evidence.

action understanding tasks and show consistent improvements over the baseline.

2.2 Related Work

Mitigating biases. Mitigating unintended biases is a critical challenge in machine learning. Examples include reducing gender or age biases for fairness [2, 13, 83, 160], easy-example biases [137, 163], and texture bias of image CNNs for improved generalization [49]. The most closely related work to ours is on mitigating scene/object/people biases by resampling the original dataset [95] to generate less biased datasets for action recognition. In contrast, I learn scene-

invariant video representations from an original biased dataset *without* resampling.

Using scene context. Leveraging scene context is useful for object detection [17, 31, 37, 105], semantic segmentation [90, 100, 105, 183], predicting invisible things [82], and action recognition without looking at the human [63, 152]. Some work have shown that explicitly factoring human action out of context leads to improved performance in action recognition [164, 184]. In contrast to prior work that uses scene contexts to facilitate recognition, my method aims to learn representations that are invariant to scene bias. I show that the debiased model generalizes better to new datasets and tasks.

Action recognition in video. State-of-the-art action recognition models either use two-stream (RGB and flow) networks [44, 133] or 3D CNNs [18, 58, 145, 162, 170]. Recent advances in this field focus on capturing longer-range temporal dependencies [157, 158, 167]. Instead, my work focuses on mitigating scene bias when training these models on existing video datasets. My debiasing approach is *model-agnostic*. I show the proposed debiasing losses improve the performance on different backbone architectures: 3D-ResNet-18 [58] and VGG-16 network [135].

Video representation transfer for action understanding tasks. Many recent CNNs for action classification [18, 58, 145, 162, 170], temporal action localization [34, 127, 173, 176, 185], and spatio-temporal action detection [8, 53, 59, 68, 77, 112, 118, 119, 120, 136, 166, 191] rely on pre-training a model on a large-scale dataset such as ImageNet or Kinetics and finetuning the pre-trained model on the target datasets for different tasks. While Kinetics is a large-scale video dataset, it still contains a significant scene representation bias [95]. In light of this, I propose to mitigate scene bias for pre-training a more generalizable video representation.

Adversarial training. Domain adversarial training introduces a discriminator to determine where the data is coming from. It has been successfully applied to unsupervised domain adaptation [47, 149] and later extended to pixel-wise adaptation [14, 26, 66] and multi-source adaptation [65].

Building upon adversarial training, recent work aims to mitigate unintended biases by using the adversary to predict the protected variables [160, 180] or quantify the statistical dependency between the inputs and the protected variables [1]. My work uses a similar strategy for mitigating scene bias from video representation. Unlike existing work where the protected variables are given (demographic information such as gender, age, race), the ground truth scene labels of my training videos are not available. To address this, I propose to use the output of a pre-trained scene classifier as a proxy.

Artificial occlusion. Applying occlusion masks to input images (or features) and monitoring the changes at the output of a model has been used to visualizing whether a classifier can localize objects in an image [178], learning object detectors from weak image-level labels [7, 93, 94, 137], improving robustness of object detectors [163], and regularizing model training [50]. I adopt a similar approach by masking out humans detected by an off-the-shelf object detector. My focus, however, differs from existing work in that I aim to train the model so that it is *unable* to predict the correct class label.

2.3 Method

2.3.1 Overview of the method

Pre-training. I show an overview of the proposed approach for learning debiased video representations in Figure 2.3. My goal is to learn parameters θ_f of a feature extractor G by pre-training it on a large-scale video classification dataset. Given a video and the corresponding action label $(\mathbf{x}, \mathbf{y}) \in \mathbf{X} \times \mathbf{Y}$, where $\mathbf{X}$ is a video dataset and $\mathbf{Y}$ is the action label set with N classes, I extract features using a CNN denoted as G_{θ_f} with parameters θ_f. Note that my method is *model-agnostic* in that I can use any 3D CNN or 2D CNN as my feature extractor. I feed the extracted features

into an action classifier f_{θ_A} with parameters θ_A. I use a standard cross-entropy loss for the action classifier for penalizing the incorrect action predictions as follows.

$$L_{CE} = -\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim(\mathbf{X},\mathbf{Y})} \sum_{k=1}^{N} y_k \log f_{\theta_A}(G_{\theta_f}(\mathbf{x})). \tag{2.2}$$

In addition to the cross-entropy loss in (2.2), I propose two losses for debiasing purpose: 1) scene adversarial loss L_{Adv}, and 2) human mask confusion loss L_{Ent}. I impose a scene adversarial loss to encourage learning scene-invariant representations. The scene adversarial loss penalizes the model if it could infer the scene types based on the learned representation. The human mask confusion loss penalizes the model if it could predict the actions when the humans in the video are masked-out. For this loss, I first detect humans in the input video and mask-out the detected humans. I then extract features of the human-masked-out video. I feed the features into the same action classifier. As shown in Figure 2.3 (dotted line), the weights of the feature extractor θ_f and action classifiers θ_A for the human mask confusion loss are shared with those for the action classification cross-entropy loss. I maximize the entropy of the predicted action class distributions when the input is a human-masked-out video. I provide more details on each of the two losses and the optimization process in the following subsections.

Transfer learning. After pre-training a CNN with the proposed debiasing method, I initialize the weights of the feature extractor, θ_f for downstream target tasks: action classification, temporal action localization, and spatio-temporal action detection. I remove the scene prediction and the human-masked action prediction heads of the network i.e., θ_A, θ_S. Then I finetune θ_f and the task-specific classification and regression parameters on the target datasets.

2.3.2 Scene adversarial loss

The motivation behind the scene adversarial loss is that I want to learn feature representations suitable for action classification but invariant to scene types. For example, on Figure 2.2 left, I aim to encourage a network to focus on a singer in the video and to predict the action as *singing*. I do not want a network to classify the video as *playing baseball* because the network recognizes that the scene is a baseball field. To enforce the scene invariance criterion to the network, I learn a scene classifier f_{θ_S} with parameters θ_S on top of the feature extractor G_{θ_f} in an adversarial fashion. I define the scene adversarial loss as,

$$L_{\text{Adv}} = -\mathbb{E}_{(\mathbf{x},\mathbf{p})\sim(\mathbf{X},\mathbf{P})} \sum_{m=1}^{M} p_m \log f_{\theta_S}(G_{\theta_f}(\mathbf{x})). \tag{2.3}$$

Here I denote a scene label as $\mathbf{p} \in \mathbf{P}$, and the number of scene types by M. The loss (2.3) is adversarial in that the parameters of the feature extractor θ_f maximize the loss while the parameters of the scene classifier θ_S minimize the loss.

Pseudo scene label. Most of the action recognition datasets such as Kinetics and UCF-101 do not have scene annotations. In this work, I obtain a pseudo scene label $\tilde{\mathbf{p}} \in \tilde{\mathbf{P}}$ by running Places365 dataset pre-trained ResNet-50 [188] on the Kinetics dataset.

2.3.3 Human mask confusion loss

I show the motivation for using the human mask confusion loss on the right of Figure 2.2. If every human, (who is *swimming* in this example), in the input video is masked out, I aim to make the network unable to infer the true action label (*swimming*). I denote a human-masked-out video as $\mathbf{x}_{\text{hm}} \in \mathbf{X}_{\text{hm}}$. I define the human mask confusion loss as an entropy function of the action label

distributions as follows.

$$L_{\text{Ent}} = -\mathbb{E}_{\mathbf{x}_{\text{hm}} \sim \mathbf{X}_{\text{hm}}} \sum_{k=1}^{N} f_{\theta_A}(G_{\theta_f}(\mathbf{x}_{\text{hm}})) \log f_{\theta_A}(G_{\theta_f}(\mathbf{x}_{\text{hm}})). \tag{2.4}$$

Both of the parameters of the feature extractor θ_f and the action classifier θ_A maximize (2.4) when an input is a human-masked-out video $\mathbf{x}_{\text{hm}}$. My intuition here is that models should not be able to predict correct action without seeing the evidence.

Human mask. To mask-out humans in videos, I run an off-the-shelf human detector [61] on the Kinetics dataset offline and store the detection results. During training, I load the cached human detection results. For every human bounding box in a frame, I fill in the human-mask regions with the average pixel value of the video frame, following the setting from Hendricks et al.[2].

2.3.4 Optimization

Using all three losses, I define the optimization problem as follows. When an input video is an original video $\mathbf{x}$ without human masking, the optimization is

$$\begin{aligned} L(\theta_f, \theta_S, \theta_A) &= L_{CE}(\theta_f, \theta_A) - \lambda L_{\text{Adv}}(\theta_f, \theta_S), \\ (\theta_f^*, \theta_A^*) &= \underset{\theta_f, \theta_A}{\operatorname{argmin}} L(\theta_f, \theta_S^*, \theta_A), \\ \theta_S^* &= \underset{\theta_S}{\operatorname{argmax}} L(\theta_f^*, \theta_S, \theta_A^*). \end{aligned} \tag{2.5}$$

Here λ is a hyperparameter for controlling the strength of the scene adversarial loss. I use the gradient reversal layer for adversarial training [46]. When an input video is a human-masked-out

video $\mathbf{x}_{\text{hm}}$, the optimization is

$$(\theta_f^*, \theta_A^*) = \underset{\theta_f, \theta_A}{\operatorname{argmax}} L_{\text{Ent}}(\theta_f, \theta_A). \tag{2.6}$$

For every iteration, I alternate between the optimization (2.5) and (2.6).

2.4 Experimental Results

I start with describing the datasets used in my experiments (Section 2.4.1) and implementation details (Section 2.4.2). I then address the following questions: i) Does the proposed debiasing method mitigate scene representation bias? (Section 2.4.3) ii) Can debiasing improve generalization to other tasks? (Section 2.4.4, Section 2.4.5) iii) What is the effect of the two proposed losses designed for mitigating scene bias? What is the effect of using different types of pseudo scene labels? (Section 2.4.6)

2.4.1 Datasets

Pre-training. I pre-train my model on the Mini-Kinetics-200 dataset [170]. Mini-Kinetics-200 is a subset of the full Kinetics-400 dataset [80]. Since the full Kinetics is very large and I do not have sufficient computational resources, I resort to using Mini-Kinetics-200 for pre-training a model. The training set consists of 80K videos and the validation set consists of 5K videos.[1] To validate whether my proposed debiasing method improves generalization, I pre-train models with debiasing and another model without debiasing. I then compare the transfer learning performances of the two models on three target tasks: 1) action classification, 2) temporal action localization, 3) spatio-temporal action detection.

[1] As the sources of Kinetics dataset are from YouTube videos, some of the videos are no longer available. The exact number of training videos I used is 76,103, and the number of validation videos is 4,839.

Action classification. For the action classification task, I evaluate the transfer learning performance on the UCF-101 [140], HMDB-51 [86], and Diving48 [95] datasets. UCF-101 consists of $13,320$ videos with 101 action classes. HMDB-51 consists of $6,766$ videos with 51 action classes. Diving48 [95] is an interesting dataset as it contains no significant biases towards the scene, object, and human. Diving48 consists of 18K videos with 48 fine-grained diving action classes. I use the train/test split provided by Li et al.[95]. Videos in all three datasets were temporally trimmed. I report top-1 accuracy on all thee splits of the UCF-101 and the HMDB-51 datasets. The Diving48 dataset provides only one split. Thus I report the top-1 accuracy on this split.

Temporal action localization. Temporal action localization is a task to not only predict action labels but also localize the start and end time of the actions. I use the THUMOS-14 [75] dataset as my testbed. THUMOS-14 contains 20 action classes. I follow Xu et al. [173]'s setting for training and testing. I train my model on the temporally annotated validation set with 200 videos. I test my model on the test set consisting of 213 videos. I report the video mean average precision at various IoU threshold values.

Spatio-temporal action detection. Given an untrimmed video, the task of spatio-temporal action detection aims to predict action label(s) and also localize the person(s) performing the action in both space and time (e.g., an action tube). I use the standard JHMDB [73] dataset for this task. JHMDB is a subset of HMDB-51. It consists of 928 videos with 21 action categories with frame-level bounding box annotations. I evaluate models on all three splits of JHMDB. I report the frame mean average precision at the IoU threshold 0.5 as my evaluation metric.

2.4.2 Implementation details

I implement my method with PyTorch (version 0.4.1). I choose 3D-ResNet-18 [58] as my feature backbone for Mini-Kinetics-200 $\rightarrow$ UCF-101/HMDB51/Diving48, and Mini-Kinetics-200 $\rightarrow$

THUMOS-14 experiments because open-source implementations for action classification [58] and temporal action localization [159] are available online. I use a 3 channels $\times$ 16 frames $\times$ 112 pixels $\times$ 112 pixels clip as my input to the 3D-ResNet-18 model. I use the last activations of the `Conv5` block of the 3D-ResNet-18 as my feature representation $G_{\theta_f}(\mathbf{x})$.

For the spatio-temporal action detection task, I adopt the frame-level action detection code [136] based on the VGG-16 network [135]. I use a 3 channels $\times$ 1 frame $\times$ 300 pixels $\times$ 300 pixels frame as my input to the VGG-16 network. I use the `fc7` activations of the VGG-16 network as my feature representation $G_{\theta_f}(\mathbf{x})$.

I use a four-layer MLP as my scene classifier, where the hidden fully connected layers have 512 units each. I choose $\lambda = 0.5$ for the gradient reversal layer [46] using cross-validation. I set the batch size as 32 with two P100 GPUs. I use SGD with a momentum of 0.9 as my optimizer. I set the weight decay as 0.00001. The learning rate starts from 0.001, and I divide it by 10 whenever the validation loss saturates. I train my network for 100 epochs. I use the validation loss on Mini-Kinetics-200 for model selection and hyperparameter tuning. When I conduct the transfer learning on the target datasets of the target tasks, I follow the same hyperparameter settings of Hara et al. [58], Wang and Cheng [159] and Singh et al.[136] for action classification, temporal action localization, and spatio-temporal action detection, respectively.

2.4.3 Scene classification accuracy

On the Mini-Kinetics-200 validation set, my scene classifier achieves an accuracy of 29.7% when training the action classification without debiasing (i.e., , with standard cross-entropy loss only).[2] With debiasing, the scene classification accuracy drops to 2.9% (the accuracy of random guess is 0.3%.) The proposed debiasing method significantly reduces scene-dependent features.

[2]Since there are no ground truth scene labels in the Mini-Kinetics-200 dataset, I use pseudo labels to measure the scene classification accuracy.

Table 2.1: Transfer learning results on action classification. The video representation trained using the proposed debiasing techniques consistently improves the accuracy on new datasets. For HMDB-51 and UCF-101, we show the average accuracy of all three test splits. All methods but TSN use a clip length of 16. In the first block, we list the accuracies of the other methods.

Method	Backbone	HMDB-51	UCF-101	Diving48
C3D [164]	C3D [144]	-	82.3	-
Factor-C3D [164]	C3D [144]	-	84.5	-
RESOUND-C3D [95]	C3D [74]	-	-	16.4
TSN [157]	BN-Inception	51.0	85.1	16.8
3D-ResNet-18 [58]	3D-ResNet-18	53.6	83.5	18.0
3D-ResNet-18 [58] + debiased (ours)	3D-ResNet-18	56.7	84.5	20.5

2.4.4 Transfer learning for action classification

Table 2.1 shows the results on transfer learning for action classification on other datasets. As shown in the last two rows of Table 2.1, my debiased model consistently outperforms the baseline without debiasing on all three datasets. The results validate that mitigating scene bias can improve generalization to the target action classification datasets. As shown in the first block of Table 2.1, action-context factorized C3D (referred to as Factor-C3D) [164] also improves the baseline C3D [144] on UCF-101. The accuracy of Factor-C3D is on par with my debiased 3D-ResNet-18. Note that the model used in Factor-C3D is 2.4$\times$ larger than ours. I also compare my method with RESOUND-C3D [95] on the Diving48 dataset. All the videos in the Diving48 dataset share similar scenes. My proposed debiasing method shows a favorable result compared to [95]. I show a large relative improvement (14.1%) on the Diving48 dataset since it has a small scene representation bias of 1.26 [95].

Relative performance improvement vs. scene representation bias. Figure 2.4 illustrates the relationship between relative performance improvement from the proposed debiasing method and the scene representation bias (defined in [95]). I measure the scene representation bias defined as (2.1) and the relative improvement of each split of the HMDB-51, UCF-101, Diving48 datasets. The

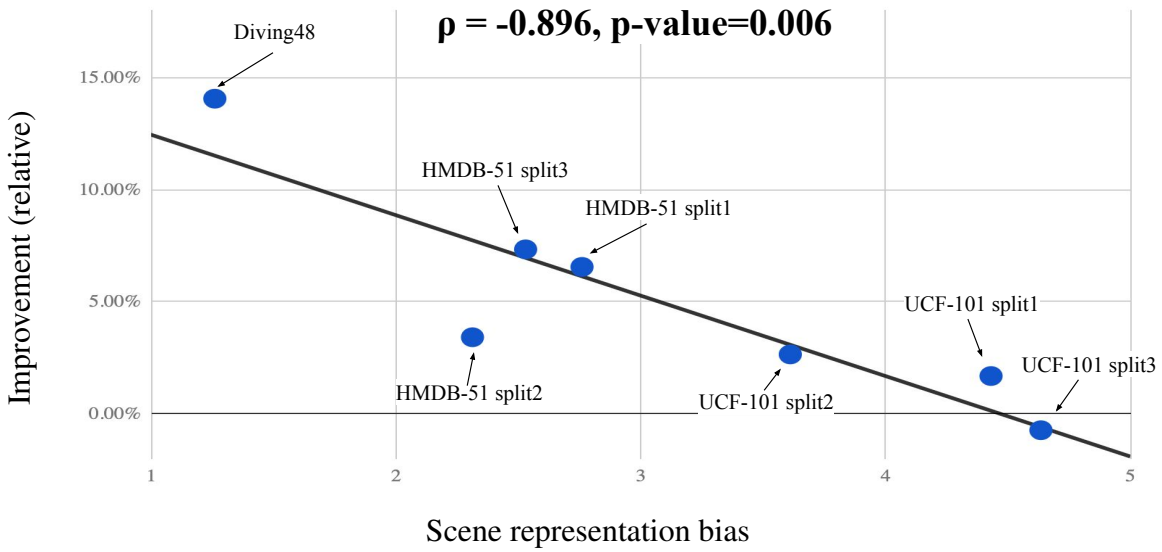


Figure 2.4: **A strong negative correlation between relative performance improvement and scene representation bias.** We measure the scene representation bias defined as (2.1) and the relative improvement of each split of the HMDB-51, UCF-101, Diving48 datasets between models trained without and with the proposed debiasing method. The Pearson correlation is $\rho = -0.896$ with a p-value 0.006, highlighting a *strong negative correlation*.

Pearson correlation is $\rho = -0.896$ with a p-value 0.006, highlighting a *strong negative correlation* between the relative performance improvement and the scene representation bias. My results show that if a model is pre-trained with debiasing, the model generalizes better to the datasets with *less* scene bias, as the model pays attention to the actual action. In contrast, if a model is pre-trained on a dataset with a significant scene bias e.g., Kinetics without any debiasing, the model would be biased towards certain scene context. Such a model may still work well on target dataset with strong scene biases (e.g., UCF-101), but does not generalize well to other *less* biased target datasets (e.g., Diving48 and HMDB-51).

2.4.5 Transfer learning for other activity understanding tasks.

Temporal action localization Table 2.2 shows the results of temporal action localization on the THUMOS-14 dataset. Using the pre-trained model with the proposed debiasing method consis-

Table 2.2: Transfer learning results on temporal action localization task: THUMOS-14 dataset. The evaluation metric is the video mAP at various IoU threshold values. The video representation trained using the proposed debiasing techniques consistently improves the performance on the new task. In the first block, we list the mAPs of the state-of-the-arts.

			IoU threshold							
Method	Inputs	Backbone	0.1	0.2	0.3	0.4	0.5	0.6	0.7	avg.
TAL-Net [19]	RGB+Flow	I3D	59.8	57.1	53.2	48.5	42.8	33.8	20.8	45.1
SSN [185]	RGB+Flow	InceptionV3	66.0	59.4	51.9	41.0	29.8	19.6	10.7	39.8
R-C3D [173]	RGB	C3D	54.5	51.5	44.8	35.6	28.9	-	-	-
CDC [127]	RGB	C3D	-	-	40.1	29.4	23.3	13.1	7.9	-
R-C3D [159]	RGB	3D-ResNet-18	48.6	48.6	45.6	40.8	32.5	25.5	15.5	36.7
R-C3D [159] + debiased (ours)	RGB	3D-ResNet-18	50.2	50.5	47.9	42.3	33.4	26.3	16.8	38.2

Table 2.3: Transfer learning results on spatio-temporal action detection. The evaluation metric is the frame mAP at the IoU threshold of 0.5. We list the frame mAP of the state-of-the-arts for reference.

Method	Backbone	Inputs	Pre-train	JHMDB (all splits)
ACT [77]	VGG	RGB+Flow	ImageNet	65.7
S3D-G [170]	Inception (2D+1D)	RGB+Flow	ImageNet+FullKinetics	75.2
ROAD [136]	VGG	RGB	ImageNet+MiniKinetics	32.5
ROAD [136] + debiased (ours)	VGG	RGB	ImageNet+MiniKinetics	34.5

tently outperforms the baseline (pre-training without debiasing) on all the IoU threshold values. My result suggests that a model focusing on the actual discriminative cues from the actor(s) helps localize the action.

Spatio-temporal action detection Table 2.3 shows spatio-temporal action detection results. With debiasing, my model outperforms the baseline without debiasing on the JHMDB dataset. The results in Table 2.2 and 2.3 validate that mitigating scene bias effectively improves the generalization of pre-trained video representation to other activity understanding tasks.

Table 2.4: Effect of debiasing using different pseudo scene labels generated from the off-the-shelf classifier.

Pseudo label used	HMDB-51 split-1	split-2	split-3	avg.
None (w/o debiasing)	52.9	55.4	52.6	53.6
Hard label	54.8	54.2	54.6	54.5
Soft label (ours)	**56.4**	**55.9**	**56.4**	**56.2**

Table 2.5: Effect of using different losses for reducing scene bias. Both losses improve the performance in transfer learning.

Loss L_{Adv}	Loss L_{Ent}	HMDB-51 split-1	split-2	split-3	avg.
×	×	52.9	55.4	52.6	53.6
×	✓	55.0	55.3	55.1	55.1
✓	×	**56.4**	55.9	56.4	56.2
✓	✓	**56.4**	**57.3**	**56.5**	**56.7**

2.4.6 Ablation study

I conduct ablation studies to justify the design choices of the proposed debiasing technique. Here I use the Mini-Kinetics-200 → HMDB-51 setting for the ablation study.

Effect of the different pseudo scene labels. First, I study the effect of pseudo scene labels for debiasing in Table 2.4. Compared to the model *without* using scene adversarial debiasing, using hard pseudo labels improves transfer learning performance. Using soft pseudo labels for debiasing further enhances the performance. I attribute the performance improvement to many semantically similar scene categories in the Places365 dataset. Using soft scene labels alleviates the issues of committing to one particular scene class. In all the remaining experiments, I use pseudo scene soft labels for scene adversarial debiasing.

Effect of the two different losses. Next, I ablate each of the two debiasing loss terms: scene adversarial loss (L_{Adv}) and human mask confusion loss (L_{Ent}) in Table 2.5. I observe that both losses improve the performance individually. Using both debiasing losses gives the best results and suggests that the two losses are regularizing the network in a complementary fashion.

2.4.7 Class activation map visualization

To further demonstrate the efficacy of my debiasing algorithm, I show class activation maps (CAM) [187] from models with and without debiasing in Figure 2.5. I show the CAM overlayed

Figure 2.5: **Class activation maps (CAM) on the HMDB-51 (first row) and UCF-101 (second row) datasets.** The words underlined in blue are correct predictions, and those in red with no underline are incorrect predictions. The video representation using the proposed debiasing algorithm focuses more on the direct visual cues (i.e., the main actors) rather than the surrounding scene contexts.

over the center frame (the eighth frame out of the sixteen frames) of each input clip. I present the results on the HMDB-51 and UCF-101 datasets. I observe that without debiasing, a model predicts the incorrect classes because the model focuses on the scene instead of human actors. However, with debiasing, a model focuses on human actions and predicts the correct action classes.

2.5 Conclusions

I address the problem of learning video representation while mitigating scene bias. I augment the standard cross-entropy loss for action classification with two additional losses. The first one is an adversarial loss for scene class. The second one is an entropy loss for videos where humans are masked out. Training with the two proposed losses encourages a network to focus on the actual action. I demonstrate the effectiveness of my method by transferring the pre-trained model to three target tasks.

As I build my model upon relatively weak baseline models, my model's final performance still falls behind other state-of-the-art models. In this work, I only addressed one type of representation

bias, i.e., scene bias. Extending the proposed debiasing method to mitigate other kinds of biases e.g., objects and persons for human action understanding is an interesting and important future work.

Chapter 3

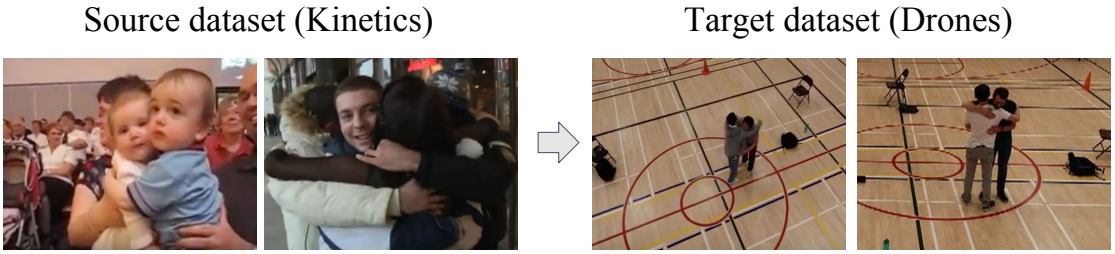
Unsupervised Learning with Target Data, Part I

3.1 Introduction

People create large amounts of digital video data recently. Such data comes from many sources e.g., surveillance videos, personal videos, commercial videos, and etc. Many of videos are human-centered. Automatic analysis of videos, e.g., for indexing and searching, is thus an interesting and critical problem. It is also very challenging due to its unconstrained nature and sheer scale. Human action recognition is one of the tasks, in this genre, which has gained substantial attention in recent years [18, 129, 133, 145]. Most of such works have addressed third-person videos while there are some works on egocentric videos as well [42, 138, 175].

Drones are becoming more popular and readily available for purchase in the consumer market. Similar to the existing human-borne camera videos, it is desirable to automatically analyze drone-captured videos. However, drone-captured videos present distinct challenges due to continuous and typical motions, perspectives, and distortions. Thus they are very different from human-borne camera videos (Fig. 3.1a).

In this chapter, I focus on an unsupervised video domain adaptation setting. I aim to leverage the



(a) Domain differences due to viewpoint, appearance, and background

(b) Label set differences due to different classes in the two domains

Figure 3.1: **Action recognition from drone videos**. Transferring knowledge learned from existing action recognition datasets is challenging as they contain mostly third-person videos. We address two challenges, i.e., domain difference (a) due to visual variation as well as (b) due to different label sets, in the two domains.

existing large-scale annotated datasets of third-person videos[1], to help perform action recognition on challenging drone-captured videos. Since acquiring and annotating videos in any new domain is an expensive and time-consuming task, under such domain adaptation settings, I aim to minimize the annotation efforts.

The large domain differences between the *source* domain of third-person videos and the *target* domain of drone-captured videos (Fig. 3.1a), motivate us to also investigate the case of *semi-supervised* domain adaptation [54]. In the semi-supervised domain adaptation setting, I assume that a limited amount of annotated target data is available during training in contrast to the unsupervised domain adaptation setting.

In addition to the case where both source and target have the same label sets, I also address the

[1] I refer to existing action recognition datasets such as Kinetics and UCF-101 as third-person datasets while noting that they may contain some other perspective videos, e.g., first person, as well.

challenging setting where the label sets are *different* (Fig. 3.1b). To reduce the domain gap between source and target data, I employ a domain classifier and adversarial loss in the both problem settings, i.e., same and different label sets. I use standard cross-entropy loss in the same label set setting, while I use an embedding-based framework in the different label sets case. The input in the latter case is agnostic of the specific class annotations of the training examples. I care only about dis-/similarities between examples, i.e., if they belong to different/same classes irrespective of the particular classes. By employing an embedding-based method, my classifier can generalize to new categories in the target domain.

I also propose to do both full video-based as well as instance-based adaptation. The full video-based method has the merit that exploits correlated context while the instance-based approach is motivated by the argument that focusing on the actor itself is more critical for better performance.

To evaluate the presented methods, I also propose a novel dataset of human actions captured by drones: NEC-DRONE. The NEC-DRONE dataset consists of 5250 videos. I evaluate the proposed method on this challenging dataset and show that I can successfully perform domain adaptation from mostly third-person videos to drone-captured videos. I further evaluate the proposed method on a publicly available Charades-Ego dataset [130]. I show qualitative results on the NEC-DRONE dataset to better understand the behaviors of the methods.

To summarize, I make the following three contributions of this work.

- I introduce a new problem of unsupervised and semi-supervised domain adaptation for action recognition from drones with two settings, i.e., same and different source and target label sets.
- I propose a new dataset, NEC-DRONE, containing 5250 videos for action recognition from drones.
- I explore the problem with thorough experiments and show significant improvements with the proposed method.

3.2 Related Work

Drone-based video datasets. A few drone-based video datasets have been proposed [6, 106, 117, 189]. However, there is only one dataset for drone-based human action recognition that I are aware of – the OKUTAMA-ACTION dataset [6]. The OKUTAMA-ACTION dataset is an outdoor dataset, and it is 43 minutes total while ours (NEC-DRONE) is 256 minutes. The number of actors is 9 vs. 19 actors (ours), and actions are 12 vs. 16 actions (ours). To the best of my knowledge, the proposed dataset is the largest drone-captured dataset for human action recognition.

Action recognition. After the success of deep networks in the image domain, many works have addressed action recognition in videos [5, 12, 18, 38, 44, 52, 74, 78, 79, 129, 132, 133, 144, 145, 157, 161, 162, 170]. This is in contrast to the earlier handcrafted features [153, 154].

Most of these methods use third-person videos to train their models. In this work, I show that such third-person models do not accurately transfer to novel domains. I propose methods to make models to generalize better using domain adaptation, utilizing a large amount of annotated third-person data.

Cross-view modeling. Understanding object, scene, and action across different views has drawn attention in computer vision. There have been works on aerial and ground view matching [98, 115], albeit the tasks are not human action recognition. For human actions, recent approaches use multi-stream networks to model first and third person videos jointly [3, 41, 128]. However, most of them require a dataset of *paired* videos across views.

I also want to learn view-invariant representations. However, collecting paired videos across different views such as a drone view, a third-person view, and a first-person view is expensive. Thus, I aim to leverage the existing *labeled* third-person videos while using only *unlabeled* target videos

(from drones), for learning representations.

Domain adaptation. Many works have addressed the problem of domain adaptation for the case of image classification [46, 54, 101, 104, 121, 125, 148, 150, 181] and object detection [25, 114, 174]. However, not much work has been done on domain adaptation for video-related tasks. A few approaches deal with an image to video domain adaptation [101, 142]. My work is different as I are interested in a video to video domain adaptation with the target videos being captured by drones.

There are a few works on video domain adaptation [21, 71]. Similar to them, I also use the basic adversarial learning framework. However, I are also dealing with more challenging problem setting where I have different source and target label sets. I are also different in that I propose to use instance-based domain adaptation as my NEC-DRONE dataset has more significant domain gap.

Open set domain adaptation. Open set domain adaptation is the setting where both source and target datasets have 'unknown' classes, and unseen class examples are all classified together into one 'unknown' category [15, 110, 122]. However, I are interested in classifying the unknown examples in different novel classes in the target domain (e.g., 'exchanging backpack').

3.3 Approach

My aim is to do domain adaptation from a source domain where I have class annotated training data $(\mathbf{x}_s, \mathbf{y}_s) \in \mathbf{X}^s \times \mathbf{Y}^s$, where $\mathbf{Y}^s$ is the source label set, and unannotated data or a very limited amount of annotated data from the target domain $(\mathbf{x}_t, \mathbf{y}_t) \in \mathbf{X}^t \times \mathbf{Y}^t$ with $\mathbf{Y}^t$ being the target label set. I address two cases of domain adaptation: (i) when the source and target label sets are the same i.e., $\mathbf{Y}^s = \mathbf{Y}^t$, and (ii) when they are different i.e., $\mathbf{Y}^s \neq \mathbf{Y}^t$. I partition the target annotated

data into three parts, the usual train and test sets and a third *support set* $(\mathbf{X}_N^t, \mathbf{Y}_N^t)$. I use the support set only in the case of unsupervised domain adaptation with different source and target label sets, to do k-NN classification in the target domain. I report the target performances on the target test set, which I again stress, has no overlap with the support set.

3.3.1 Overview of the architecture

My overall architecture (Figure 3.2) leverages the advances made in both video representations as well as domain adaptation. The system takes a video with T frames, denoted as $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2 \dots \mathbf{v}_T\}$ where $\mathbf{v}_i \in \mathbb{R}^{h \times w \times c}$ are the height h, width w, and c channel frames, as an input and splits it into small, potentially overlapping, clips $\mathbf{x} = [\mathbf{v}_j, \mathbf{v}_{j+1} \dots \mathbf{v}_{j+L-1}]$ where L is the clip length. Then I pass the clips through a state-of-the-art video CNN, denoted as $\psi(\cdot)$ to obtain feature representations, $\psi(\mathbf{x})$ of the clips. I pass the clip features to a softmax with classification loss or an embedding-based metric learning loss, as well as to a discriminator network with domain adversarial loss. I describe the different cases in the following.

3.3.2 Same source and target label set

The first case is when the K classes are the same in the source and target domains i.e., $\mathbf{Y}^s = \mathbf{Y}^t$ (Figure 3.2a). Even in this case, the domain differences are substantial due to the various challenges such as variations in appearance, perspective, motion, etc. In this case, the system learns representations with a combination of cross-entropy loss for classification in the source domain along with the domain adversarial loss, i.e., binary cross-entropy loss, between examples of source and target domains. Formally, denoting the classifier by $f_C(\cdot)$ with parameters θ_c, and

the discriminator by $f_D(\cdot)$ with parameters θ_d, I define the losses as,

$$\mathscr{L}_{CE} = -\mathbb{E}_{(\mathbf{x}_s,\mathbf{y}_s)\sim(\mathbf{X}^s,\mathbf{Y}^s)} \sum_{k=1}^{K} y_{s,k} \log f_C(\psi(\mathbf{x}_s)), \tag{3.1}$$

$$\begin{aligned}\mathscr{L}_{ADV} = &-\mathbb{E}_{\mathbf{x}_s\sim\mathbf{X}^s} \log f_D(\psi(\mathbf{x}_s)) \\ &-\mathbb{E}_{\mathbf{x}_t\sim\mathbf{X}^t} \log(1 - f_D(\psi(\mathbf{x}_t))).\end{aligned} \tag{3.2}$$

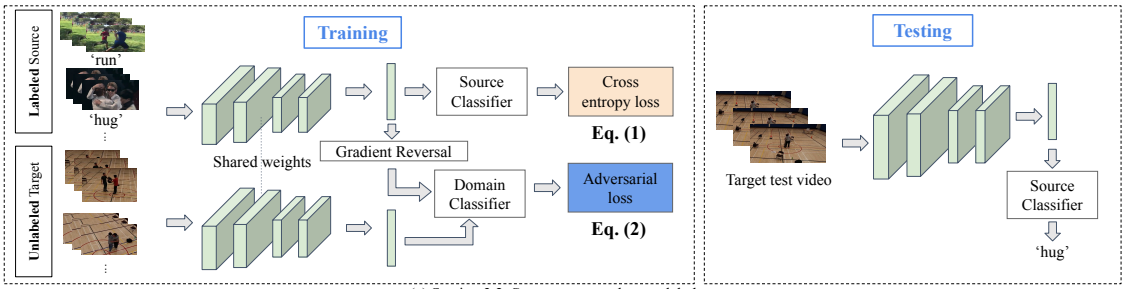
The optimization problem is then given by,

$$\begin{aligned}&\mathscr{L}(\theta_f,\theta_c,\theta_d) = \mathscr{L}_{CE}(\theta_f,\theta_c) - \lambda\mathscr{L}_{ADV}(\theta_f,\theta_d), \\ &(\theta_f^*,\theta_c^*) = \arg\min_{\theta_f,\theta_c} \mathscr{L}(\theta_d^*),\ \theta_d^* = \arg\max_{\theta_d} \mathscr{L}(\theta_f^*,\theta_c^*).\end{aligned} \tag{3.3}$$

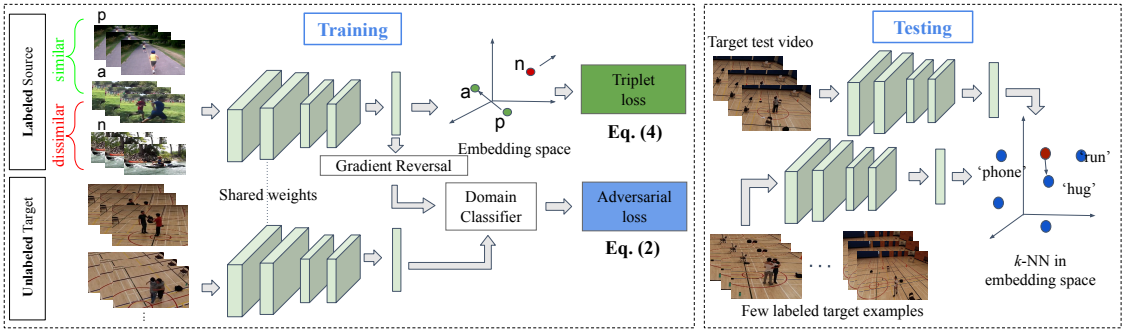
where, θ_f are the feature extractor parameters of ψ, and λ is a hyper-parameter for the trade-off between the cross-entropy and the domain adversarial losses. I mark optimal parameters θ with a symbol $*$ in a superscript.

The optimization learns a classifier by minimizing the classification loss, a discriminator by minimizing the adversarial loss and a feature extractor by minimizing the classification loss and maximizing adversarial loss, to learn domain invariant and discriminative representations. I use the gradient reversal layer [46] for adversarial training.

Semi-supervised adaptation. I also evaluate *semi-supervised* domain adaptation, where, in addition to the unlabeled target examples, some annotated target examples are available for training as well. I use the target *annotated* examples with cross-entropy loss, and the target *unannotated* examples with domain adversarial loss only.



(a) Section 3.2. **Same** source and target label sets



(b) Section 3.3. **Different** source and target label sets

Figure 3.2: **Overview of the proposed domain adaptation method.** Our system takes a video as an input and splits it into small clips. We pass these clips through a video CNN. (a) In the same source and target label set setting, the clips features are input to a softmax with classification loss as well as to a discriminator network with domain adversarial loss. At testing time, the system takes a video as an input, split into multiple clips, pass the clips into the trained CNN to extract features. The system then predicts labels with the source classification layer. (b) In a different source and label sets setting, the clip features are input to an embedding based metric learning loss, as well as to a discriminator network with domain adversarial loss. At testing time, the system takes a video as an input, split into multiple clips, pass the clips into the trained CNN to extract features. The system requires few labeled target examples at test time (a support set) to perform k-NN classification.

3.3.3 Different source and target label sets

In the second case, the domain differences are due to the difference in labels sets i.e., $\mathbf{Y}^s \neq \mathbf{Y}^t$ (Figure 3.2b) as well as the variations such as appearance, perspective, motion, etc. The source and target label sets could be different with some or potentially no overlap. In this case, I propose to learn embeddings of the videos which are agnostic of the specific classes but are aware of being similar (when examples come from the same class) or dissimilar (when they come from different classes). To do this I use a standard metric learning loss, i.e., the triplet loss [124], which takes a

triplet of examples $(\mathbf{x}_a, \mathbf{x}_p, \mathbf{x}_n)$ with $\mathbf{x}_a$ being the anchor and $\mathbf{x}_p, \mathbf{x}_n$ being the positive (same class as anchor) and negative (different class than the anchor) examples respectively. In the embedding space, the triplet loss forces the smaller distance between the anchor and the positive example by a margin of δ, than the distance between the anchor and the negative example. Formally the loss and optimization problem are given as,

$$\begin{aligned}\mathscr{L}_{TRI} = -\mathbb{E}_{(\mathbf{x}_a,\mathbf{x}_p,\mathbf{x}_n)} \max(0, \delta + \\ \|\psi(\mathbf{x}_a) - \psi(\mathbf{x}_p)\|^2 - \|\psi(\mathbf{x}_a) - \psi(\mathbf{x}_n)\|^2), \end{aligned} \tag{3.4}$$

$$\begin{aligned}\mathscr{L}(\theta_f, \theta_d) = \mathscr{L}_{TRI}(\theta_f, \theta_d) - \lambda \mathscr{L}_{ADV}(\theta_f, \theta_d), \\ \theta_f^* = \arg\min_{\theta_f} \mathscr{L}(\theta_d^*), \;\; \theta_d^* = \arg\max_{\theta_d} \mathscr{L}(\theta_f^*). \end{aligned} \tag{3.5}$$

In a minibatch, I sample examples from both the source as well as the target domain. All samples contribute to minimizing the adversarial loss, while the samples from the source domain construct the triplet examples and contribute to minimizing the triplet loss.

Once I have trained the network, the system classifies query examples, by first, obtaining the embeddings from a forward pass of the base CNN, and then performing k-NN-based classification in the embedding space, using the target support set $(\mathbf{X}_N^t, \mathbf{Y}_N^t)$.

Semi-supervised adaptation. I also evaluate semi-supervised setting in a different source, and target label sets setting, similar to the same label sets setting. The two differences are, (i) I use the cross-entropy loss for target classes as well, and (ii) I do not use the support set, as now the system can directly do target class classification.

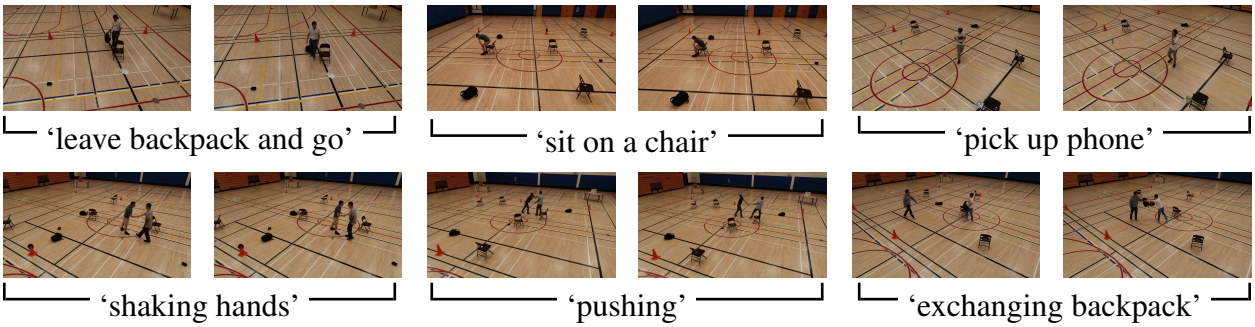


Figure 3.3: **Sample frames from the NEC-DRONE dataset**. We show two close-by frames per video. The first row shows single person actions, while the second row shows two person actions. Best viewed on screen, with zoom and color.

3.3.4 Video-based and instance-based adaptation

Since I are interested in human actions, the discriminative visual regions in the frames are expected to be around humans. I could expect that focusing on the humans in frames would give better performance by eliminating noise from the background. On the other hand, the background might contain correlated elements which could potentially contribute to better recognition. Since both the human foreground as well as the background have potential merits, I propose to do both 'video-based' and 'instance-based' adaptation. In the video-based adaptation I give the full clip as the input to the system, while for the instance-based case, I first perform human detection using a state-of-the-art pre-trained human detector [61] and then feed only the human spatio-temporal tube (i.e., a clip made by cropping out human from every frame) as an input the the system.

I independently train video-based and instance-based domain adaptation models. During testing, I perform late-fusion of the two predictions from video-based and instance-based models. I empirically show that both have advantages, especially when some amount of target annotated data is available (semi-supervised setting), and their combination consistently improves over either of them alone.

3.4 NEC-DRONE Dataset

I propose a new dataset, NEC-DRONE, of videos taken from drones for the task of domain adaptation from third-person videos to drone videos. Figure 3.3 shows some examples. I collected the dataset inside a school gym with 19 actors acting out their interpretations of 16 pre-defined actions multiple times. The actions performed by the actors are in an unconstrained manner without any close supervision.

The actions are both single as well as two-person actions. The partial motivation of defining the actions was to keep surveillance scenarios in mind, e.g., two people getting together and exchanging a backpack could be an interesting event to tag, or retrieve. There are 10 single person actions, i.e., walk, run, jump, pick up a backpack and go, leave a backpack and go, sit on a chair, talk on a mobile phone, drink water from a bottle, throw something, pick up a small object, and 6 two-person actions, i.e., shake hands, push a person, hug, exchange a backpack, walk toward each other and stay, stand together leave.

I recorded each action instance by two drones simultaneously flown in an unconstrained manner by relatively new pilots. The videos in the dataset are from varied perspectives with the drones flying at varying distances and heights from the actors. The drones used were 'DJI Phantom 4.0 pro v2' and the videos were recorded at 30 fps at a resolution of 1920×1080 pixels. I manually annotated the actions of all videos.

Finally I have a total of 5250 videos with a total of more than $460k$ frames. I split the videos into 1188 *train*, 437 *val*, and 454 *test* sets with labels, and 3171 videos without labels. I make sure that the actors in the *train*, *val* and *test* sets are disjoint. I evaluate the performance on the dataset as the mean class accuracy.

The proposed dataset is challenging for the following main reasons. First, *view point* is different from typical action datasets, and it changes heavily over time due to the flying drone. Second,

Table 3.1: Nearest neighbor *test* results (without learning any parameters) on the UCF-101 and the NEC-DRONE dataset with pre-trained I3D features.

Dataset	UCF-101 (vid.)	NEC-DRONE (vid.)	NEC-DRONE (inst.)
Acc(%)	72.3	8.2	10.8

due to the *continuous* and often *erratic drone motion*, the videos have jitters and motion blur. Third, often the person(s) of interest are *not centered* and are relatively *small*. To the best of my knowledge, the NEC-DRONE is the largest drone dataset for human action recognition. I plan to release it publicly upon acceptance of the paper.

In Table 3.1, I show a significantly larger *domain gap* between Kinetics and the NEC-DRONE dataset, compared to the domain gap between Kinetics and UCF-101. I perform a nearest neighbor classification on the UCF-101 [140] and NEC-DRONE datasets. Note that I do not learn any parameters. I use ℓ_2 normalized `mixed5c` activations of the I3D network [18] pre-trained on Kinetics as my feature.

Video-based nearest neighbor classifier can achieve 72.3% accuracy on UCF-101 dataset (split 1). However, the video-based nearest neighbor can achieve only 8.2% accuracy on the NEC-DRONE dataset. Using instance-based nearest neighbor, I can achieve 10.8% accuracy. This significant difference is due to the large domain gap between the NEC-DRONE dataset and typical third-person video datasets. Furthermore, the existing third-person video datasets such as UCF-101 and Kinetics have correlated backgrounds for different actions. However, the NEC-DRONE dataset has a similar background for all the actions. Thus the dataset is very challenging, as without capturing the human motion, it is difficult to recognize the different human actions in the NEC-DRONE dataset.

3.5 Experimental Results

Abbreviations. I use the following abbreviations. DA: domain adaptation, UDA: unsupervised domain adaptation, SSDA: semi-supervised domain adaptation, src.: source, tgt.: target, vid.: video-based, inst.: instance-based, sup.: supervised finetuning.

Same label set for source and target. I use the Kinetics [80] dataset as the source dataset and the NEC-DRONE dataset as the target dataset. Since the two datasets do not share the same classes, I subsample the two datasets to obtain similar classes. I choose 13 classes from Kinetics [80] dataset and 7 classes from the NEC-DRONE dataset which have similar actions to construct the source and target datasets. See supplementary for the details.

Different label sets for source and target. When I work with different label sets settings, I use the UCF-101 [140] as my source dataset. UCF-101 dataset is mainly a third-person dataset and contains $13,320$ videos from 101 action classes. The domain gap between the UCF-101 and NEC-DRONE datasets is significant (as I also show quantitatively below) and the label sets of the UCF-101 dataset and NEC-DRONE datasets are entirely disjoint. Hence this makes a challenging and practical domain adaptation setting from third-person videos to drone-captured videos.

For both settings, I use labels for m target examples per class, in addition to the unannotated target and annotated source examples, for semi-supervised adaptation.

Implementation details. I use state-of-the-art I3D network [18] as my base network for feature extraction with $L = 16$ frame clip inputs for drone experiments and $L = 32$ frame clip inputs for Charades-Ego experiments. I attach the domain discriminator to `mixed5c` layer of the I3D network. I use a 4 layer MLP for domain classifier where the hidden fully connected layers have 4096 units each.

When aligning features at the instance-based, I extract the human tubes by running per frame

detectors and making tracks based on the overlaps of the detections in the successive frames. I use a Mask R-CNN [61] pre-trained on MS-COCO dataset [99] for person detection.

I set $\lambda = 1.0$ for the gradient reversal layer [46], $\delta = 0.5$ for the margin parameter of the triplet loss and the embeddings. I use a batch size of 10 and sample mini-batches as follows. In the case of triplet loss only, 7 out of 10 examples are from anchor class, and the rest 3 are from different classes. In the case of triplet loss with unsupervised domain adaptation, 5 (3 same class, 2 different classes) out of 10 examples are source examples and rest 5 are target examples. I use SGD optimizer with the momentum of 0.9. For the source pre-training and semi-supervised finetuning, I use an initial learning rate of 10^{-4}, and for the unsupervised domain adaptation training, I use an initial learning rate of 10^{-6}. I reduce the learning rate by $1/10$ after 5 epochs.

Table 3.2: Correspondences between classes of the NEC-DRONE dataset and the Kinetics dataset for the same label set for source and target setting.

Drone dataset class	Kinetics dataset classes
walking	marching
running	jogging, running on treadmill
jumping	high jump, jumping into pool
drinking water from a bottle	drinking beer
throwing an object	throwing axe, throwing ball, throwing discuss, shot put, javelin throw
shaking hands	shaking hands
hugging	hugging

I use the Kinetics dataset as source dataset and the proposed drone dataset as target dataset. Since the two datasets do not share exactly the same classes, I subsample the two datasets to obtain similar classes. I manually choose 13 classes from Kinetics dataset and 7 classes from my drone dataset which have similar or closely related actions to construct the source and target datasets. Table 3.2 gives the class correspondences between the classes of drone dataset and the Kinetics dataset.

Table 3.3: Action recognition accuracies (%) on the NEC-DRONE dataset (*val* set) in the **same source and target label sets** case. m is the number of target annotated examples per class used while training. As a reference, the full target supervised I3D performance is 76.7%.

Method	$m=0$	$m=3$	$m=5$	$m=10$	$m=20$
Inst. no DA	12.6	31.1	35.4	43.2	49.5
Inst. DA	16.5	31.6	39.3	41.3	52.4
Vid. DA	13.6	24.3	35.4	53.9	52.9
Vid. & inst. DA	15.1	32.0	41.8	54.9	58.3

3.5.1 Quantitative evaluation on NEC-Drone

Same source and target label sets. I first perform an ablation study of video-based DA and instance-based DA and their combination with different number m of target annotated examples per class used during training. I also include the results without any domain adaptation. Since it is an ablation study, I perform experiments on the *val* set. The column where $m=0$ is the unsupervised domain adaptation setting while the columns where $m>0$ are semi-supervised domain adaptation settings.

The results show the contribution of different adaptation components. The video-based adaptation achieves 13.6% in the unsupervised case, the instance-based achieves 16.5%, while the combination of the two gives 15.1%. The performances rise rapidly as even a small number of annotated examples from the target domain are provided during training. With only $m=3$ examples the performance of the combined method increases to 32.0% which further increases to $41.8\%, 54.9\%, 58.3\%$ on $m=5,10,20$. The $m=20$ performance of 58.3% is still far from the full target supervised performance of 76.7%; in the latter case, the average number of examples per class is 80. I also note that the combination of the video-based adaptation with the instance-based adaptation is always greater than either of them indicating complementary information in the two methods.

Table 3.4: Comparison of methods on the NEC-DRONE dataset (*test* set) in the **same source and target label sets** setting, with $m = 5$ target annotated examples per class used in semi-supervised adaptation. The classifier here is the multi-class source classifier.

Method	Training data	Acc (%)	Gain(%)
Fully sup.	*labeled* drone	69.3	N/A
Src. only	Kinetics	13.6	0.0
Vid. DA	Kinetics + *unlabeled* drone	27.2	100.0
Inst. DA	Kinetics + *unlabeled* drone	29.4	116.1
Vid. & inst. DA	Kinetics + *unlabeled* drone	32.0	135.2

Table 3.5: Comparison of methods on the NEC-DRONE dataset (*test* set) in the **different source and target label sets** setting, with, $m = 0$ i.e., unsupervised domain adaptation, and $n = 3$ target examples per class used as a support set at testing. The classifier is nearest neighbor in embedding space.

Method	Training data	Acc (%)	Gain (%)
Fully sup.	*labeled* drone	68.3	N/A
Src. only	UCF101	8.2	0.0
Vid. DA	UCF101 + *unlabeled* drone	10.6	29.2
Inst. DA	UCF101 + *unlabeled* drone	14.3	74.3
Vid. & inst. DA	UCF101 + *unlabeled* drone	14.5	76.8

Table 3.4, third column, gives the final test performances of the same label set setting for the different methods on the NEC-DRONE dataset for $m = 5$. I see that the video-based adaptation improves the source only classifier from 13.6% to 27.2%, while the instance-based adaptation achieves 29.4%. The combination of both gets the best performance of 32.0%. This is still quite far from the target fully supervised value of 69.3% indicating that still, a large domain gap exists even after semi-supervised adaptation.

Table 3.6: Accuracies (%) on the NEC-DRONE dataset (*test* set) in the **different source and target label sets** case. m is the number of target annotated examples per class used for training. In testing time, we do not use any target examples as a support set. i.e., $n = 0$ setting.

Method	$m = 3$	$m = 5$	$m = 10$	$m = 20$
Inst. DA	15.9	21.6	31.3	34.6
Vid. DA	12.8	18.7	29.1	34.4
Vid. & inst. DA	18.1	22.5	36.1	39.7

Different source and target label sets. Table 3.5 shows the results of the different methods for the case of different source and target label sets. Here, I are using no target annotated examples for training ($m = 0$). But I are using $n = 3$ target examples per class at testing as a support set. The task is harder as I use a larger number of classes (all 16 classes present in the NEC-DRONE dataset) compared to the same source and target label sets case, while I use only 7 classes due to the constraint of finding similar classes. The full target supervised accuracy, in this case, is 68.3% compared to 69.3% of the former.

The trends among the methods are similar to the previous case of the same source and target label sets. The source only classifier performs very poorly at 8.2%, cf. 6.25% for a random chance for this 16 class case. The contrast is much higher in this case compared to the previous as (i) it is a harder setting where completely new classes are predicted, and (ii) in general embedding-based methods perform lower than cross entropy-based 1-of-C class classifiers. Compared to the source only classifier, the video-based method improves performance by 29.2% relatively, while the instance-based method improves by 74.3% relatively. The combination of the two further improves 76.8% relatively.

Table 3.6 gives the semi-supervised domain adaptation results for the setting when the source and target label sets are different. I show the results for the different number m of target annotated examples per class, used during training. The trend is similar to the Table 3.3. With a small

number of annotated examples used during training, I can improve the performance compared to the unsupervised domain adaptation (14.5% for $m = 0$ vs. 39.7% for $m = 20$).

3.5.2 Ablation study on the losses

Table 3.7 gives the performances with the instance only adaptation for different configurations of the embedding-based framework in the different source and target label sets setting. If I use neither the adversarial loss nor the triplet loss and use only the cross entropy loss, the performance is 10.8%. If I add adversarial loss, the performance improves by 25.9%, relatively. If I use triplet loss only, I get 33.3% relative improvement over using cross entropy loss only. When both the triplet and adversarial losses are taken into account the performance is improved by 74.0% compared to cross entropy loss only. These results highlight two things. First, both the adversarial and triplet losses are important for good performance. Second, triplet loss performs better than cross entropy loss in the setting where the source and target label sets are different.

Table 3.7: Ablation study on the losses (*val* set) in the **different source and target label sets** setting. Our instance-based domain adaptation method is used for the ablation study.

Adversarial	triplet	Cross entropy	Acc (%)	Gain (%)
×	×	✓	10.8	0.0
✓	×	✓	13.6	25.9
×	✓	×	14.4	33.3
✓	✓	×	18.8	74.0

3.5.3 Quantitative evaluation on Charades-Ego

I compare the performance with other methods on a publicly available Charades-Ego dataset [130] in Table 3.8. Please note that Charades-Ego is a *paired* dataset. Therefore every first person and third person video is paired with its counterpart. My method does not require paired dataset, thus

Table 3.8: Comparison of methods on the Charades-Ego dataset (first person *test* set). Note that for the semi-supervised domain adaptation, we use x% of the target training data with labels and use the rest of the target training data without labels for training.

Method	Back-bone	Pair sup.	Train	Test	% of anno. tgt	mAP (%)
[31]	ResNet-152	✓	3rd + 1st	1st	pair sup.	20.0
Src. only	I3D	×	3rd	1st	0	16.6
UDA	I3D	×	3rd + 1st	1st	0	17.9
SSDA	I3D	×	3rd + 1st	1st	10	20.4
SSDA	I3D	×	3rd + 1st	1st	20	21.9
SSDA	I3D	×	3rd + 1st	1st	30	22.8
SSDA	I3D	×	3rd + 1st	1st	40	23.1
Fully sup.	I3D	×	1st	1st	100	25.8

Figure 3.4: **Effect of DA and inst. DA.** (Top row) Examples misclassified by the method without DA but are correctly classified with DA. (Bottom row) Examples that are misclassified with vid. DA method but are correctly classified with the vid. & inst. DA. Ground truth/incorrect predictions in green/*red*.

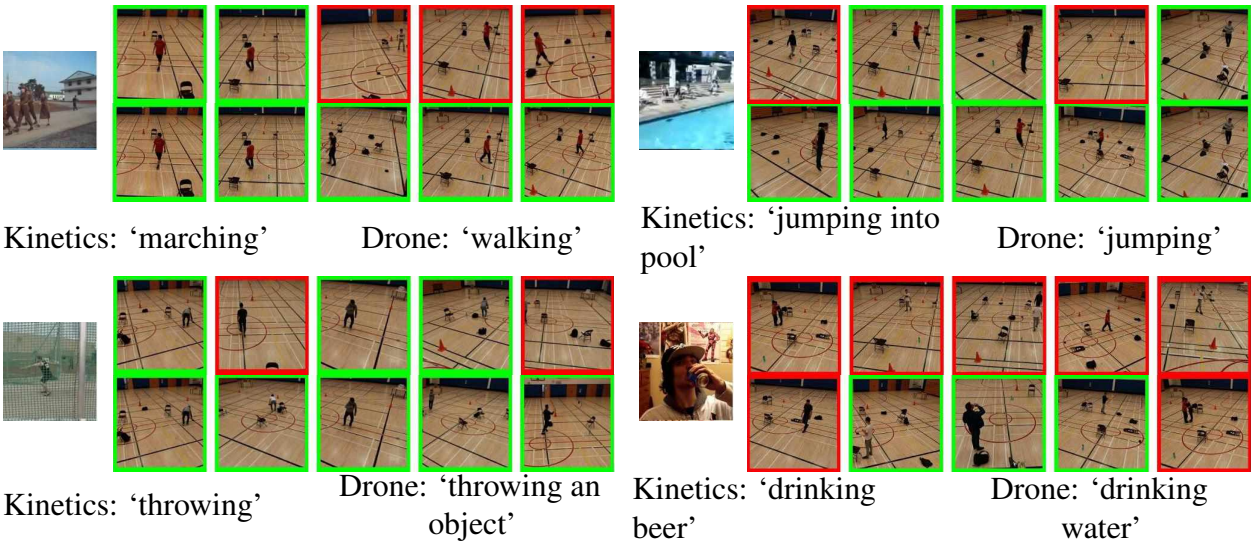


Figure 3.5: **Cross-domain retrieval results.** For each of the 2×2 blocks, the first column shows a frame of a query video from the Kinetics dataset. The rest of the columns show the top five retrieved videos from the NEC-DRONE dataset. The correct/incorrect category level retrievals are highlighted in green/red. Top row is with the video only model *without* domain adaptation and the bottom row is the same model, but trained *with* domain adaptation. We show the class labels in Kinetics and the corresponding classes from the NEC-DRONE dataset. Best viewed on screen, with zoom and color.

more general than Actor and Observer [128]. Also note that the reported performance in [128] is invalid because the authors evaluated on the wrong split[2]. Thus, I run the authors' code and report mAP on the valid test set, which is 20.0%. I obtain 17.9% mAP with my video-based *unsupervised* domain adaptation. Using annotated target data improves performance. With only 10% of the labeled target data, my semi-supervised domain adaptation achieves 20.4% mAP, outperforming the Actor and Observer. Both Actor and Observer and my semi-supervised domain adaptation method use some level of supervision: Actor and Observer uses paired videos for supervision, while my method uses x% of the target label for supervision. My semi-supervised domain adaptation method is more general than requiring paired third-person and first-person videos. Therefore, my method is more suitable for action recognition from novel domains where getting paired video dataset is difficult e.g., drone-captured videos.

[2] https://github.com/gsig/actor-observer/issues/7

3.5.4 Qualitative evaluation on NEC-Drone

I qualitatively demonstrate the contribution of domain adaptation in Figure 3.4, showing (i) misclassified examples when not using domain adaptation but are correctly classified with domain adaptation (top row), (ii) examples where video only adaptation method fails but combined instance and video-based adaptation method succeeds. I can observe the large domain gaps in terms of the perspective and the area occupied by the actor in the example frames. While the typical third-person videos have a direct perspective and almost centered actor as the major content of the video, the videos in the proposed dataset have challenging perspectives and can also be taken from far. The proposed method addresses these domain gaps and improves performance.

With my embedding-based method, I also obtain a common space where I can compare videos from the source and target domains. To demonstrate it, I show the cross-domain action category level retrieval results in Figure 3.5. Given a query from 'marching' action class of the Kinetics (first column of the top-left block), I show the top nearest neighbors from the NEC-DRONE dataset. In each block, the first and second row show the retrieval results without and with domain adaptation respectively. I can observe that the domain adpated model can successfully retrieve 'walking' class videos from the NEC-DRONE dataset despite a huge domain gap. Without domain adaptation, the retrievals contain more irrelevant videos from other classes.

3.6 Conclusions

I addressed the task of human action recognition from drones in the setting where I do not have any labeled examples of drone dataset, or I have only a few labeled examples. I further explored a more challenging setting where the source and the target label sets are different. To deal with this challenging setting, I proposed to use metric learning loss and unsupervised domain adaptation

along with instance-level action recognition.

Since a challenging large dataset of drone videos for human action recognition did not exist, I collected 5250 high-resolution videos from two drones with 16 predefined single person and two-person actions. I empirically showed that a large domain gap exists between third-person video datasets and the NEC-DRONE dataset. I will release the dataset upon acceptance to the community.

My work is among the first to show encouraging domain adaptation results on challenging video domains. However, I also show that I are still far from the fully supervised classifier performances in the target domain of drone videos, and hence, there is much room for improvement.

Chapter 4

Unsupervised Learning with Target Data, Part II

4.1 Introduction

Recent computer vision-based methods have reached very high performances in supervised tasks [18, 61, 62, 70, 162] and many real-world applications have been made possible such as image search, face recognition, automatic video tagging etc. The two main ingredients for success are (i) high capacity network design with an associated practical learning method, and (ii) large amounts of *annotated* data. While the first aspect is scalable, in terms of deployment to multiple novel scenarios, the second aspect becomes the limiting factor. The annotation issue is even more complicated in video-related tasks, as I need temporal annotation, i.e., I need to specify the start and end of actions in long videos. Domain adaptation has emerged as an important and popular problem in the community to address this issue. The applications of domain adaptation have ranged from simple classification [46, 101, 121, 148, 150, 181] to more complex tasks like semantic segmentation [20, 26, 67, 146, 151, 182] and object detection [16, 25, 64, 69, 81, 190]. However, the application on video tasks e.g., action recognition is still limited [21, 29, 71].

I address this less studied but challenging and practically important task of video domain adaptation for human action recognition. I work in an unsupervised domain adaptation setting. That is, I have annotated data for the source domain and only *unannotated* data for the target domain. Examples

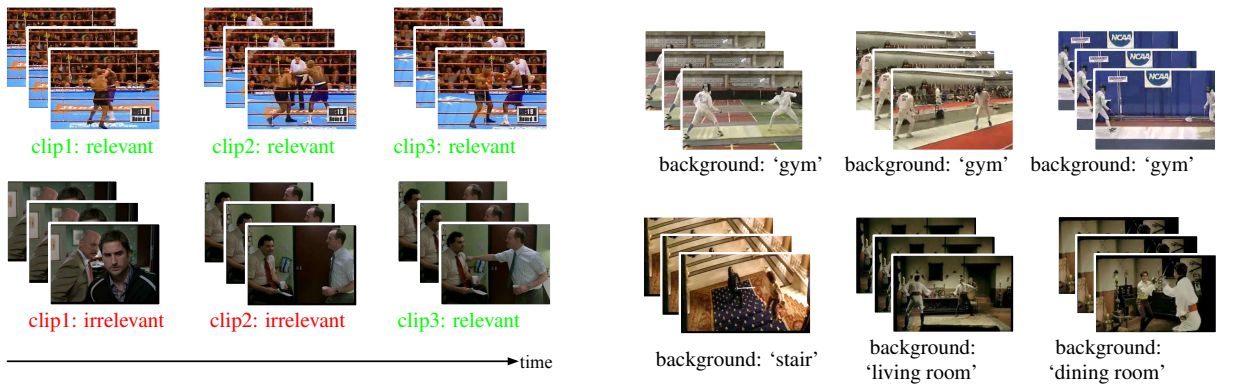


Figure 4.1: **Motivation.** We do video domain adaptation and introduce the following two key components: (*Left*): Clip attention. The top video and the lower video have the same action *punching*. However, the lower video has only one relevant punching clip, while the top video has three relevant punching clips. Our proposed attention suppresses features from irrelevant clips, improving the feature alignment across domains. (*Right*): Clip order prediction. The top and bottom videos are from different domains, but all capture the action *fencing*. However, the backgrounds are different: the top domain has a gym as a background, and the lower domain has a dining room or a living room or a stair as a background. Predicting the order of clip encourages the model to focus more on the humans, not the background, as the background is uninformative for predicting temporal order. Best viewed with zoom and color.

domains that I use in experiments include (human) actions from movies, unconstrained actions from sports videos, YouTube videos, and even videos taken from drones.

I exploit two insights related to the problem and propose two novel adaptation components inspired by them. First, I note that the existing domain adaptation methods, when applied directly to the video adaptation task, sample frames or clips [21, 71], depending on whether the video encoding is based on a 2D network, e.g., temporal relation network [186] or a 3D network, e.g., C3D [144]. I sample clips (or frames) and then average the final outputs from multiple clips at test time, following the video classification networks they are built upon. Performing domain adaptation by aligning features for all sampled clips is suboptimal, as a lot of network capacity is wasted on aligning clips that are not crucial for the task. In the worst case, it can even be detrimental if a large number of unimportant clips dominate the learning loss and adversely affect the alignment of important clips. For example, in Figure 4.1 left, both the top video from one domain and the bottom video from another domain have the same action, *punching*. However, the bottom video

contains a lot of clips irrelevant to *punching*. Aligning features from those irrelevant clips would not improve the target performance much.

Second, this clip-wise training method is likely to exploit correlations in the scene context for discriminating the action classes [28, 95, 96], e.g., in a formal sports-oriented dataset fencing might happen in a gym only as shown in the top right three videos of Figure 4.1. However, in the domain adaptation setting, the target domain might have vastly different scene contexts, e.g., the same fencing might happen in a living room or dining room, as shown in the bottom right three videos of Figure 4.1. When the source model uses the correlated gym information to predict a fencing action, it may perform poorly on the same class in the target domain, which does not have a gym scene. Similar scene context corruption issues have been identified for transfer learning, and few works have addressed the problem of *debiasing* the representations explicitly [28, 164].

Based on the above insights, I propose **Shuffle and Attend: Video domain Adaptation** (SAVA) with two novel components. First, I propose to identify and align *important* (which I define as *discriminative*) clips in source and target videos via an attention mechanism. The attention mechanism leads to the suppression of temporal background clips, which helps us focus on aligning only the important clips. Such attention is learned jointly for video-level adaptation and classification. I estimate the clip's importance by employing an auxiliary network and derive the video feature as the weighted combination of the identified important clip features.

Second, I propose to learn *spatial-background invariant human action representations* by employing a self-supervised clip order prediction task. While there could be some correlation between the scene context/background and the action class, e.g., soccer field for 'kicking the ball' action, the scene context is not sufficient for predicting the temporal clip order. In contrast, the actual human actions are indicative of the temporal order, e.g., for 'kicking the ball' action the clip order follows roughly the semantics of 'approaching the ball', 'swinging the leg' and 'kicking'; if I shuffle the clips, the actual human action representation would be able to recover the correct order, but the

scene context based representation would be likely to fail.

Thus using the clip order prediction based loss helps us counter the scene context corruption in the action representations and improves adaptation performance. I employ the self-supervised clip order prediction task for both source and target. As this auxiliary task is self-supervised, it does not require any annotation (which I do not have for target videos).

I provide extensive empirical evaluations to demonstrate the benefits of the proposed method on three challenging video domain adaptation benchmark settings. I also give qualitative results to highlight the benefits of my system.

In summary, my contributions are as follows.

- I propose to learn to align important (discriminative) clips to achieve improved representation for the target domain.
- I propose to employ a self-supervised task which encourages a model to focus more on actual action and suppresses the scene context information, to learn representations more robust to domain shifts. The self-supervised task does not require extra annotations.
- I obtain state-of-the-art results on the HMDB to UCF adaptation benchmark, and Kinetics to NEC-Drone benchmarks.

4.2 Related Work

Action recognition. Action recognition using deep neural networks has shown quick progress recently, starting from two-stream networks [133] to 3D [18, 144, 162] or 2D and 1D separable CNNs [145, 170] have performed very well on the task. More recent advances in action recognition model long-term temporal contexts [43, 162]. However, most models still rely on target

supervised data when finetuning on target datasets. In contrast, I are interested in unsupervised domain adaptation, where I do not have access to target labels during training.

Unsupervised domain adaptation for images. Based on adversarial learning, domain adaptation methods have been proposed for image classification [46, 101, 121, 148, 150, 181], object detection [16, 25, 64, 69, 81, 190], semantic segmentation [20, 26, 67, 146, 151, 182], and low-level vision tasks [116]. I also build upon adversarial learning. However, I work with videos and not still images.

Unsupervised domain adaptation for videos. Unlike image-related tasks, there are only a few works on video domain adaptation [21, 29, 71]. I also use the basic adversarial learning framework but improve upon it by adding auxiliary tasks that depend on the temporal order in videos, (i) to encourage suppression of spatial-background, and (ii) to focus on important clips in the videos to align.

Self-supervision. Image based self-supervised methods work with spatial context, e.g., by solving jigsaw puzzle [108], image inpainting [111], image colorization [89], and image rotation [51] to learn more generalizable image representation. In contrast, video based self-supervised methods exploit temporal context, e.g., by order verification [102], frame sorting [92], and clip sorting [171]. Recent video domain adaptation methods employ self-supervised domain sequence prediction [22], or self-supervised RGB/flow modality correspondence prediction [107].

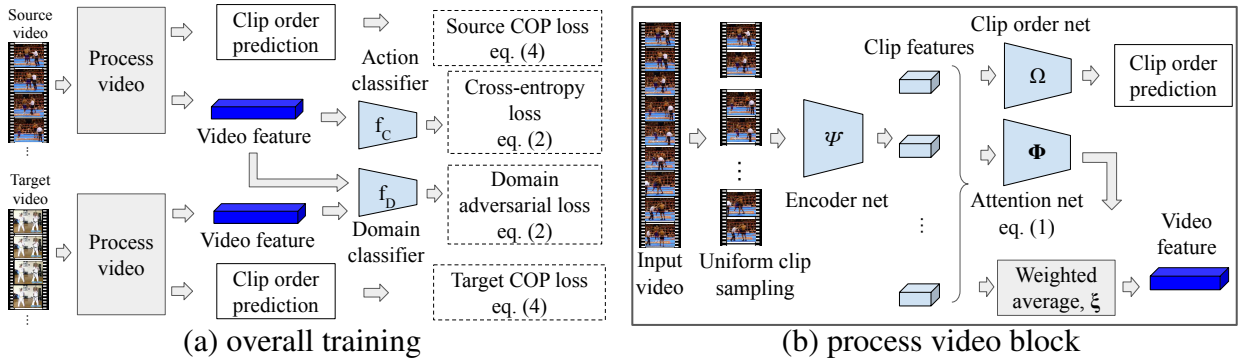
I make a connection between the self-supervised task of clip order prediction [171] and learning a robust spatial-background decoupled representation for action recognition. I hypothesize (see Section 4.1) that, in combination with adversarial domain adaptation loss, this leads to suppression of domain correlated background, and simultaneous enhancement of the task correlated human part in the final representation leading to better domain adaptation performance.

Attention. There are numerous methods employing attention model for image [48, 72] and

video tasks [45, 52, 78, 84, 109, 131, 156, 172]. The most closely related work is the that by Chen et al. [21]. While both the proposed method and Chen et al. [21] are based on attention, the main difference is in *what* they attend to. Chen et al. [21] attends to temporal relation features (proposed by Zhou et al. [186]) with *larger domain gaps*. In contrast, my proposed method attends to *discriminative* clip features. The clips in the same video may have different discriminative content, e.g., leg swinging (more discriminative) vs. background clips (less so) in a video of 'kicking a ball' class. The proposed method attends to more discriminative clips and focuses on aligning them. Chen et al. [21] samples $2 \sim 5$ frames relation features and attends to the ones with a larger domain gap measured by the entropy of the domain classifiers. However, the relation feature with a larger domain gap might come from frames irrelevant to the action, aligning them would be suboptimal. The proposed method addresses this problem. In another closely related work Pan et al. [109] temporally align the source and target features using temporal co-attention and match their distributions. In contrast, the proposed method argues that human-focused representation is more robust to domain shifts, and captures it via self-supervised clip order prediction.

4.3 Method

I work in an unsupervised domain adaptation setting, where (i) I have *annotated source data* $(\mathbf{x}_s, \mathbf{y}_s) \in \mathbf{X}^s \times \mathbf{Y}^s$, where $\mathbf{X}^s$ is the set of videos containing human-centered videos and $\mathbf{Y}^s$ is the actions label set, and (ii) *unannotated target data* $\mathbf{x}_t \in \mathbf{X}^t$. The task is to train a model using all the data, which performs well on the target data. Since the source data distribution, e.g., actions in movies, is expected to be very different from the target data distribution, e.g., actions in sports videos, the model trained on the source data only does not work well on target videos. The challenge is to design methods that can adapt a model to work on the target data, using both annotated source data and unannotated target data. The method proposed here has, at a high level, three main



(a) overall training (b) process video block

Figure 4.2: **Overview of SAVA.** We employ standard domain adversarial loss along with two novel components. The first component is the self-supervised clip order prediction loss. The second is a clip attention based feature alignment mechanism. We predict attention weights for the uniformly sampled clips from the videos and construct the video feature as a weighted average of the clip features. Then we align the source and target video features. Best viewed with zoom and color.

components for adaptation: domain adversarial loss, clip order prediction losses, and an attention module for generating video features.

Figure 4.2 gives an overview of the proposed method, which I call Shuffle and Attend Video domain Adaptation (SAVA). I start with uniformly sampling N clips, with L frames, from an arbitrary length input video, as shown in the 'process video block' in the figure. I encode source and target clips into clip features by an encoder network $\Psi(\cdot)$; which can be either the same for both or different. Here I assume it is the same for the brevity of notation. Then I use the clip features for (i) the clip order prediction network $\Omega(\cdot)$, and (ii) constructing the video-level features using the attention network $\Phi(\cdot)$. The video-level features obtained after the attention network, are then used with (i) linear action classifier, for source videos only, and (ii) domain classifier, for both source and target videos, as shown in the left of the figure.

In total, there are three types of losses that I optimize, (i) domain adversarial loss, (ii) clip order prediction loss for both source and target, and (iii) classification loss for source only. The clip order prediction loss works with clip level features, while the other two work on video-level features. As discussed in Section 4.1, the clip order prediction loss helps a model to learn a representation

that is less reliant on correlated source data background. The attention network gives us the final video feature by focusing on important clips. The domain adversarial loss helps a model to align video-level features between source and target videos. All these are jointly learned and hence lead to a trained system that gives aligned representations and achieves higher action classification performance than the baselines. I now describe each of my proposed components individually in detail in the following subsections.

4.3.1 Clip order prediction

As shown on Figure 4.1 (right), the source videos of the same class may have correlations with similar background context [95], and the target videos of the same class might have a background which is vastly different from the source background. While the source model might benefit from learning representation, which is partially dependent on the correlated background, this would lead to poor target classification. To address this problem, I propose to employ clip order prediction (COP) to enable better generalization of the representation. COP would not be very accurate if a model focuses on the background as the background might not change significantly over time. However, the temporal evolution of the clip depends more on the humans performing actions, and possibly the objects. Thus, if I employ the COP, the representation would focus more on the relevant humans and objects, while relying less on the background.

I build my COP module upon the work by Xu et al. [171]. I show the illustration of the COP network Ω in Figure 4.3. I incorporate an auxiliary network, taking clip features as input, to predict the correct order of shuffled clips of an input video. I sample M clips, with L frames each, from an input video and shuffle them. The task of the module is to predict the order of the shuffled clips. I formulate the COP task as a classification task with $M!$ classes, corresponding to all permutation tuples of the clips, and consider the correct order tuple as the ground truth class. I

concatenate clip features pairwise and pass them to a fully connected layer with ReLU activation followed by a dropout layer. Then I concatenate all of the output features and use a final linear classifier to predict the order of the input clips. Since this is a self-supervised task and requires no extra annotation, I can use the task for the videos from source, target, or both; I evaluate this empirically in Section 4.4.3.

4.3.2 Clip-attention based video-level features

As shown in the left side of Figure 4.1, all clips are not equally *important* (*discriminative* or *relevant*) for predicting the action. Aligning the irrelevant clip features is suboptimal, and it might even degrade performance if they dominate the loss cf. the important clips. Focusing on and aligning the important clips would lead to better adaptation and classification performance. To achieve such focus on important clips, I propose a clip attention module. The attention module takes N number of clip features as inputs, and outputs N softmax scores indicating the importance of each of them. The final video-level feature is obtained by the weighted average of the clip features. Formally, given $\mathbf{x}_1, \ldots, \mathbf{x}_N$ as the N clips from an input video $\mathbf{x}$, I obtain the video-level feature $\mathbf{x}_v$ as

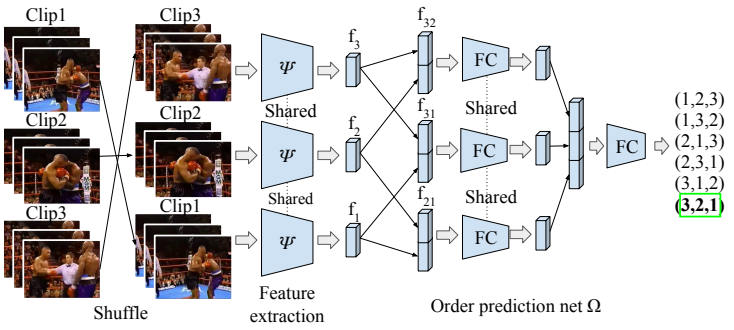


Figure 4.3: **Clip order prediction network Ω (the layers after Ψ).**

$$\mathbf{w} = \Phi(\Psi(\mathbf{x}_1), \ldots, \Psi(\mathbf{x}_N)), \qquad \mathbf{x}^v = \mathbf{x}_i(\mathbf{w}, \Psi(\mathbf{x}_1), \ldots, \Psi(\mathbf{x}_N)) = \sum_{i=1}^{N} w_i \Psi(\mathbf{x}_i), \tag{4.1}$$

where, $\mathbf{x}_i(\cdot)$ is the weighted average function.

The attention module $\Phi(\cdot)$ is a network that takes N clip features with D dimension as an input. It

outputs the importance vector $\mathbf{w} \in R^N$, which is used for weighted averaging to obtain the video-level feature. Thus, I can train the model end-to-end with the full domain adaptation system.

There can be multiple valid choices for the architecture of the attention module, e.g., a standard feed-forward network which takes concatenation of the clip features as input, or a recurrent network that consumes the clip features one by one. I explore two specific choices in an ablation experiment in Section 4.4.3, (i) Multi Layer Perceptron (MLP) similar to Kar et al. [78], and (ii) Gated Recurrent Units (GRU).

4.3.3 Training

I pre-train the attention module with standard binary cross-entropy loss, where I get the ground truth attention vector as follows. The ground truth label is 1 if the clip is correctly classified by the baseline clip-based classification network, and has confidence higher than a threshold c_{th}, and 0 otherwise. The pre-training makes the attention module to start from good local optima, mimicking the baseline classifier. Once pre-trained, the attention module can then either be fixed or can be trained end-to-end with the rest of the network. Please note that I train the attention module only on the source dataset as the training requires the ground truth action labels.

For the feature distribution alignment, I follow the well-known adversarial domain adaptation framework of ADDA [150]. I define my losses as,

$$
\begin{aligned}
L_{\text{CE}} &= -\mathbb{E}_{(\mathbf{x}_s,\mathbf{y}_s)\sim(\mathbf{X}^s,\mathbf{Y}^s)} \sum_{k=1}^{K} [y_{s,k} \log f_C(\mathbf{x}_s^v)], \\
L_{\text{ADV}_{f_D}} &= -\mathbb{E}_{\mathbf{x}_s\sim\mathbf{X}^s}[\log f_D(\mathbf{x}_s^v)] - \mathbb{E}_{\mathbf{x}_t\sim\mathbf{X}^t}[\log(1 - f_D(\mathbf{x}_t^v)], \\
L_{\text{ADV}_{\psi_t}} &= -\mathbb{E}_{\mathbf{x}_t\sim\mathbf{X}^t}[\log f_D(\mathbf{x}_t^v)],
\end{aligned} \tag{4.2}
$$

where f_C is the linear source classifier and f_D is the domain classifier. The video feature $\mathbf{x}^v =$

$\mathbf{x}_i(\mathbf{w}, \Psi(\mathbf{x}_1)\ldots,\Psi(\mathbf{x}_N))$ is the weighted average of clip level features, with weights $\mathbf{w} = \Phi(\Psi(\mathbf{x}_1),\ldots,\Psi(\mathbf{x}_N))$ obtained from the attention module. Then my optimization objective is as follows,

$$\theta_s^*, \theta_{f_C}^*, \theta_\Phi^* = \underset{\theta_s,\theta_{f_C}}{\operatorname{argmin}} L_{\mathrm{CE},\theta_\Phi}, \theta_{f_D}^* = \underset{\theta_{f_D}}{\operatorname{argmin}} L_{\mathrm{ADV}_{f_D}}, \theta_t^* = \underset{\theta_t}{\operatorname{argmin}} L_{\mathrm{ADV}_{\psi_t}}, \tag{4.3}$$

where θ_s is the parameter of the source encoder $\Psi_s(\cdot)$, θ_{f_C} is the parameter of the source classifier $f_C(\cdot)$, θ_t is the parameter of the target encoder $\Psi_t(\cdot)$, and θ_{f_D} is the parameter of the domain classifier $f_D(\cdot)$.

I optimize this objective function in a stage-wise fashion [150]. I first optimize the source cross-entropy loss L_{CE} over the source parameters θ_s and θ_{f_C} with the annotated source data. Then I freeze source model parameters θ_s and θ_{f_C}, and optimize the domain classification loss $L_{\mathrm{ADV}_{f_D}}$ over the domain classifier parameter θ_{f_D}, and the inverted GAN loss $L_{\mathrm{ADV}_{\psi_t}}$ over the target encoder parameter θ_t with both the labeled source and the unlabeled target data.

Clip order prediction. I define the COP loss as follows.

$$L_{\mathrm{COP}} = -\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim(\mathbf{X},\mathbf{Y})} \sum_{k=1}^{M!} [y_k \log f_O(\phi)]. \tag{4.4}$$

Here, f_O is the linear classification function for the COP, $\phi = \Omega(\Phi(\mathbf{x}_1),...,\Phi(\mathbf{x}_M))$ is the ReLU activation of the MLP which takes M clip features as input. I can employ the L_{COP} for both source and target. I optimize the loss L_{COP} over the source encoder parameter θ_s, target encoder parameter θ_t, COP MLP parameter θ_Ω, and clip order cliassifier parameter θ_{f_O}.

4.3.4 Inference

At inference time, I remove the domain discriminator and clip order prediction network. I divide the input video into N clips and extract clip features. These features are then weight averaged with

weights obtained using the attention network. The action classifier predicts the action using the video-level feature.

4.4 Experimental Results

4.4.1 Datasets

I show results on the publicly available benchmark based on the UCF [140] and HMDB [86] datasets. I further show the result in a more challenging setting where the source dataset is part of the Kinetics dataset [18], and the target dataset is drone-captured action dataset [29]. In the following, the direction of the arrow indicates the source (arrow start) to target (arrowhead).

UCF↔HMDB. Chen et al. [21] released the UCF-HMDB dataset for studying video domain adaptation. This dataset has $3,209$ videos with 12 action classes. All the videos come from the original UCF [140] and HMDB [86] datasets. They subsampled overlapping 12 classes out of 101/51 classes from the UCF/HMDB, respectively. There are two settings of interest, UCF $\rightarrow$ HMDB, and the other is HMDB $\rightarrow$ UCF. I show the performance of my method in both of the two settings. I use the official split provided by the authors [21].

Kinetics→NEC-Drone. I also test my method on a more challenging target dataset captured by drones [29]. The dataset contains $5K$ videos with 16 classes in total, while the domain adaptation subset used contains 994 videos from 7 classes, which overlap with Kinetics dataset. I use the official train/val/test split provided by Choi et al.[29]. I conduct domain adaptation experiments with Kinetics→NEC-Drone setting, which is more challenging than UCF↔HMDB as there is a more significant domain gap between source and target domains.

Summary of the datasets I present the summary of the datasets used in this work in Table 4.1.

Table 4.1: Video domain adaptation datasets summary.

	UCF	HMDB	Kinetics	NEC-Drone
Length (sec.)	1-33	1-33	1-10	1-22
Spatial resolution	320×240	varies × 240	varies	1920×1080
Frame rate	25	30	varies	30
# of classes	12	12	7	7
# of training videos	1,438	840	9,955	560
# of validation videos	571	360	742	206
# of test videos	-	-	-	228
# of training frames	276,148	84,883	2,415,462	75,901
# of validation frames	107,223	34,023	181,878	29,224
# of test frames	-	-	-	29,742
Domain gap	UCF→HMDB: 14.7%p HMDB→UCF: 8.0%p		Kinetics→Drone: 64.5%p	

In addition to the other information, I add a domain gap row between datasets by measuring classification performance difference between the supervised source only I3D (lower bound) and the supervised target I3D (upper bound) in the last row of Table 4.1. Kinetics→Drone has a domain gap of 64.5% while UCF→HMDB has 14.7%p and HMDB→UCF has 8.0%p. The domain gap difference suggests that Kinetics→Drone is a more challenging setting than UCF→HMDB and HMDB→UCF.

In all three settings, I report top-1 accuracy on the target dataset and compare it to other methods.

4.4.2 Implementation details

I implement my method with the PyTorch library. I use the I3D [18] network as my clip feature encoder architecture for both source and target. The source and target encoders are different from each other and do not share parameters. Both are initialized with the Kinetics pre-trained model weights and then trained further as appropriate. Such pre-training on a large dataset is common

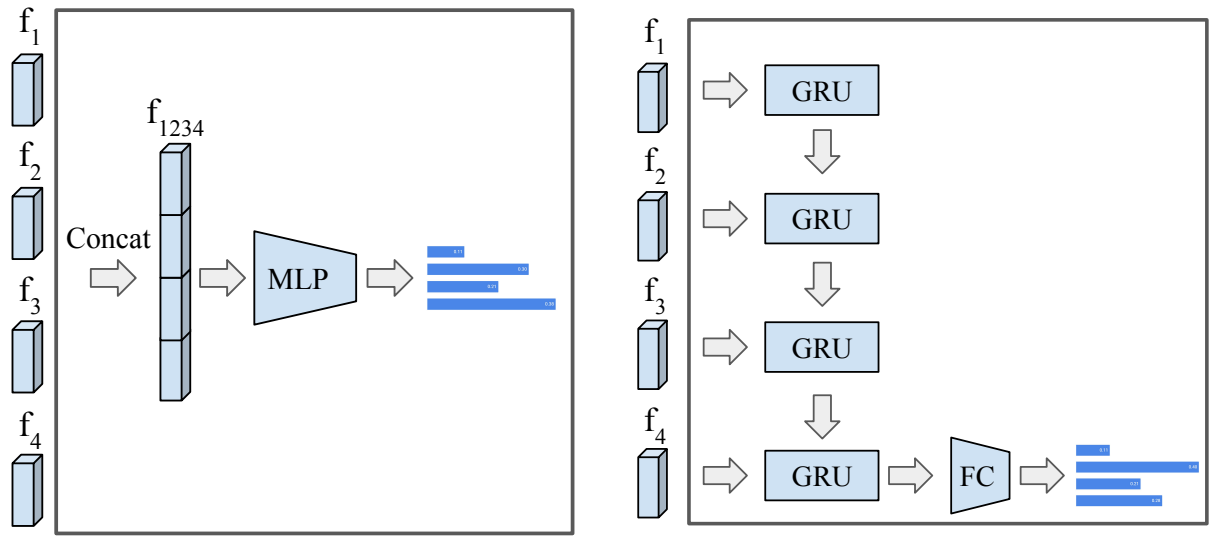
in domain adaptation, e.g. for images (ImageNet) [25, 46, 146, 150] and videos (Sports-1M [71], Kinetics [22, 107]). The input to the clip feature encoder is a 3 channels $\times$ 16 frames $\times$ 224 $\times$ 224 pixels clip. I set the number of clips per video to $N = 4$ via validation. During testing, I sample the same $N = 4$ number of clips. COP module is a 2-layer MLP with 512 hidden units. I sample $M = 3$ clips per video for the COP task by following Xu [171].

By using attention, I compute the weighted average of the clip-level softmax score as my final video-level softmax score. I evaluate two types of networks for the attention module. One is 4-layer MLP with 1024 hidden units in each layer, and the other is a GRU [27] with 1024 hidden units. I found GRU to be better in two out of the three cases (Section 4.4.3), so I report all results with GRU. I set the attention module's confidence threshold c_{th} as 0.96 for the UCF and HMDB and 0.8 for Kinetics by validation on the source dataset. I use 4-layer MLP with 4096 hidden units in each layer as my domain classifier.

I set the batch size to 72. The learning rate starts from 0.01, and I divide the learning rate by 10 after two epochs and ten epochs. I train models for 40 epochs. I set the weight decay to 10^{-7}. I use stochastic gradient descent with momentum 0.9 as my optimizer.

I follow the 'pre-train then adapt' training procedure similar to previous work [150]. (i) I train the feature extractor $\Psi(\cdot)$ with the COP loss (4.4). I train my feature extractor $\Psi(\cdot)$ on both source and target datasets as I do not require any labels. (ii) Given the trained feature extractor $\Psi(\cdot)$, I further train it on the labeled source and unlabeled target datasets with a domain classifier $f_D(\cdot)$ attached. I also train the attention module $\Phi(\cdot)$ on the labeled source dataset, given the trained feature extractor $\Psi(\cdot)$ from step 1. (iii) Given the feature extractor $\Psi(\cdot)$ and the attention module $\Phi(\cdot)$, I train my full model with the labeled source dataset and unlabeled target dataset.

Attention module. There can be multiple valid choices for the architecture of the attention module $\Phi(\cdot)$, e.g., a standard feed-forward network which takes concatenation of the clip features as input,



(a) MLP implementation (b) GRU implementation

Figure 4.4: **Implementations of clip attention network Φ.** Best viewed with zoom and color.

or a recurrent network that consumes the clip features one by one. I explore two specific choices: (1) MLP and (2) gated recurrent unit (GRU) network, as shown in Figure 4.4. Let us assume the number of clips per video $N = 4$ for brevity. *The MLP-based attention module* takes four clips as input. Then I concatenate the four clips (temporally ordered as shown in Figure 4.4 (a)) and pass it through 4 fully connected layers. The four fully connected layers consist of 3 layers with 1024 hidden units each, and four units in the final fully connected layer. The output of the MLP is the length-4 vector indicating that which clip is more important (discriminative) and which clip is less important. *The GRU-based attention module* also takes four clips as input. Then I pass each clip in temporal order to the GRU with 1024 hidden units, as shown in Figure 4.4 (b). I pass the output feature of GRU to a fully connected layer to get the length-4 vector indicating that which clip is more important and which clip is less important.

Baselines. I implement competitive baseline video domain adaptation methods by extending the two state-of-the-art image-based domain adaptation methods DANN [46], and ADDA [150] as shown in Figure 4.5. The major differences between DANN and ADDA are two-fold: (1) The

DANN method shares the feature backbone parameters between source and target encoders. The ADDA method, on the other hand, does not share the network parameters. (2) The DANN method uses a gradient reversal layer or inverted GAN loss for adversarial training of domain classifiers. I extend DANN and ADDA by replacing the feature vector input to the domain classifier from image feature to clip feature. I sample a $L = 16$-frames long clip from each video and pass it through a clip feature encoder (e.g., I3D [18] in this work). I feed the clip features from source and target videos to the domain classifier.

I3D-based TA^{3}N. The original TA^{3}N builds upon the TRN using the 2D ResNet-101 [62] feature backbone. I replace the 2D ResNet-101 backbone with the I3D backbone. Given a video, I densely slide a temporal window with a temporal stride of 1 to sample 16 frames long clips. For a frame k in the video, the temporal window consists of frames from $k-7$ to $k+8$. I zero-pad the beginning and the end of the input video. I then extract I3D features by feeding the sliding windows to the I3D feature backbone. As a result, I obtain a $1,024$ dimensional feature vector for every frame. I then follow the remaining steps as in the original TA^{3}N method.

4.4.3 Ablation study

I perform several ablation experiments to analyze the proposed domain adaptation method. I conduct the experiments on more challenging Kinetics→NEC-Drones setting except the attention module design choice experiment in Table 4.4, which I performed on the UCF, HMDB, and Kinetics datasets.

Effect of clip order prediction. Table 4.2 gives the results showing the effect of COP. Here, the source only is the I3D network trained on the source dataset, which I directly test on the target dataset without any adaptation. Clip-level domain adaptation (Clip DA) is the baseline where I randomly sample clips and align features of the clips without any attention. On top of the clip DA,

Table 4.2: Ablation experiments on the COP loss, on Kinetics→NEC-Drone.

Method	COP on Source	COP on Target	Top-1 acc (%)	Δ
Clip DA + COP	✓	✓	28.5	+ 11.3
Clip DA + COP	✓	×	25.9	+ 8.7
Clip DA + COP	×	✓	22.4	+ 5.2
Clip DA only	×	×	23.7	+ 6.5
Supervised source only	×	×	17.2	reference

Table 4.3: Ablation experiments on the clip attention on Kinetics→NEC-Drone.

Method	Align	Clip attention	Top-1 acc (%)
SAVA (ours)	video-level	✓	31.6
SAVA (ours) w/o. clip attention	video-level	×	30.3
Clip-level align	clip-level	×	28.5

I can optionally use the COP losses for either source or target or both.

The clip DA only (without COP) improves performance over the supervised source only baseline by 6.5%p. More interestingly, the results show that using both source and target COP improves performance significantly compared to the clip DA only baseline by 4.8%p. I also observe that the source COP is more crucial compared to the target COP. This is because the target NEC-Drone dataset (i) contains similar background appearance across all videos (a high school gym), (ii) has a limited number of training videos ($\sim 1K$), and (iii) has the main activities occurring with small spatial footprint (as the actors are small given the videos were captured by drones). Thus, applying COP on the NEC-Drone dataset does not lead to improved results. However, applying COP on the source or both source/target produces large improvements over the baseline.

Table 4.4: Effect of using different attention implementation. We show the attention module accuracy (%) on the Kinetics, UCF, and HMDB datasets.

Method	No. params	Kinetics	UCF	HMDB
MLP	$6.3M$	72.2	86.1	75.4
GRU	$6.3M$	78.0	78.9	76.6

Clip attention performance. I evaluate the two different design choices, MLP and GRU, for my attention module in this experiment. I show the clip attention accuracy on the three source datasets in Table 4.4. I get the attention accuracy by comparing the ground truth importance label (see Section 4.3.3 for the details) and the predicted importance. I compute the clip attention performance on the three source datasets Kinetics, UCF, and HMDB. Using such curated ground truth ensures that the attention module starts from good local minima, which is in tune with the base I3D encoder network.

The GRU shows a higher attention performance in two out of the three cases, while it has a similar number of parameters to the MLP-based attention. Thus, I employ the GRU-based attention module on all experiments in this chapter.

Effect of attention module. I show the effect of the attention module in the overall method in Table 4.3. Here, all the methods are pre-trained using source and target COP losses turned on. I train my domain adaptation network with three settings, (i) video-level alignment with clip attention (my full model), (ii) video-level alignment without clip attention (using temporal average pooling instead), and finally (iii) clip-level alignment.

The results show that video-level alignment gives an improvement over random clip sampling alignment, 30.3% vs. 28.5%. My full model with clip attention alignment further improves the performance to 31.6%, over video-level alignment without attention. The video-level alignment without attention treats every clip equally. Hence, if there are some non informative clips, e.g., temporal background, equally aligning those clips is a waste of the network capacity. My *discriminative* clip attention alignment is more effective in determining more discriminative clips and doing alignment based on those.

Table 4.5: Results on UCF↔HMDB.

Method	Encoder	UCF→HMDB	HMDB→UCF
Supervised source only [21]	ResNet-101-based TRN	73.1 (71.7)	73.9 (73.9)
TA^{3}N [21]	ResNet-101-based TRN	75.3 (78.3)	79.3 (81.8)
Supervised target only [21]	ResNet-101-based TRN	90.8 (82.8)	95.6 (94.9)
Supervised source only [21]	I3D-based TRN	80.6	88.8
TA^{3}N [21]	I3D-based TRN	81.4	90.5
Supervised target only [21]	I3D-based TRN	93.1	97.0
TCoN [109]	ResNet-101-based TRN	87.2	89.1
Supervised source only	I3D	80.3	88.8
SAVA (ours)	I3D	82.2	91.2
Supervised target only	I3D	95.0	96.8

4.4.4 Comparison with other methods

Methods compared. The methods reported are (i) 'supervised source only': the network trained with supervised source data (a lower bound for adaptation methods), (ii) 'supervised target only': the network trained with supervised target data (an upper bound for the adaptation methods), and (iii) different unsupervised domain adaptation methods. For the TA^{3}N, I compare with the latest results obtained by running the public code[1] provided by the authors [21] and not the results in the paper (given in brackets for reference, in Table 4.5). While the original TA^{3}N [21] works with 2D features based temporal relation network (TRN) [186], I go beyond and integrate the TA^{3}N with stronger I3D [18] based TRN features. This allows a fair comparison with my method when all other factors (backbone, computational complexity, etc) are similar. For TCoN, I report the numbers from the paper [109] as code is not publicly available.

For the Kinetics → NEC-Drone setting, I implement video versions of the DANN [46] and ADDA [150], which align the clip-level I3D [18] features and show the results. I also compare with both unsupervised and semi-supervised methods of Choi et al. [29].

[1] https://github.com/cmhungsteve/TA3N

UCF → HMDB. I compare my method with existing methods in Table 4.5. The first three blocks contain the results of the method with TRN-based encoder [186]. The fourth block shows the results of my SAVA with domain adaptation as well as the source only I3D [18] baseline. I also show the result of fully supervised finetuning of the I3D network on the target dataset as an upper bound.

SAVA with the I3D-based encoder shows 82.2% top-1 accuracy on the HMDB dataset, in this setting. SAVA improves the performance of the strong I3D encoder, 80.3%, which in itself obtains better results than the TRN-based adaptation results, 75.3% with TA^{3}N. My SAVA is closer to the upper bound (82.2% vs. 95.0%), than the gap between TA^{3}N and its upper bound (75.3% vs. 90.8%). Furthermore, SAVA outperforms TA^{3}N with I3D-based TRN features, 81.4%.

HMDB → UCF. Table 4.5 gives the comparison of my method with existing methods in this setting. I achieve state-of-the-art results in this setting while the other trend is similar to the UCF→HMDB setting. SAVA achieves 91.2% accuracy on the target dataset with domain adaptation and without using any target labels. The baseline source only accuracy of the I3D network is already quite strong cf. the existing best adaptation method, i.e., 88.8% vs. 90.5% for TA^{3}N with I3D-based TRN features. I improve this to 91.2%. SAVA is quite close to the upper bound of 96.8%, which strongly supports the proposed method. In contrast, TA^{3}N is still far behind its upper bound (79.3% vs. 95.6%).

Kinetics → NEC-Drone. This setting is more challenging, as the domain gap is larger, i.e., the gap between the source only and target finetuned classifiers is 64.5% cf. 14.7% for UCF→HMDB.

Table 4.6 gives the results. The first block uses the TRN with ResNet-101 features, and the second block uses the TRN with I3D features while the others use I3D features. I observe that similar to previous cases, SAVA outperforms all methods, e.g., DANN (42% relative), ADDA (33% relative), TA^{3}N (26.4% relative), TA^{3}N with I3D features (12.4% relative), and Choi et al.(the unsupervised

Table 4.6: Results on the Kinetics→NEC-Drone.

Method	Encoder	Target labels used (%)	Top-1 acc (%)
Supervised source only [21]	ResNet-101-based TRN	None	15.8
TA^{3}N [21]	ResNet-101-based TRN	None	25.0
Supervised source only [21]	I3D-based TRN	None	15.8
TA^{3}N [21]	I3D-based TRN	None	28.1
Supervised source only	I3D	None	17.2
DANN [46]	I3D	None	22.3
ADDA [150]	I3D	None	23.7
Choi et al. [29] (on val set)	I3D	None	15.1
SAVA (ours)	I3D	None	31.6
Choi et al. [29]	I3D	6	32.0
Supervised target only	I3D	100	81.7

domain adaptation case). It is very close to the semi-supervised result of Choi et al.(31.6 vs. 32.0), where they use 5 target labeled examples per class.

While the improvements achieved by SAVA are encouraging in this challenging setting, the gap is still significant, 31.6% with adaptation vs. 81.7% with the model finetuned with the target labels. The gap highlights the challenging nature of the dataset, and the large margin for improvement in the future, for video-based domain adaptation methods.

4.4.5 Qualitative evaluation

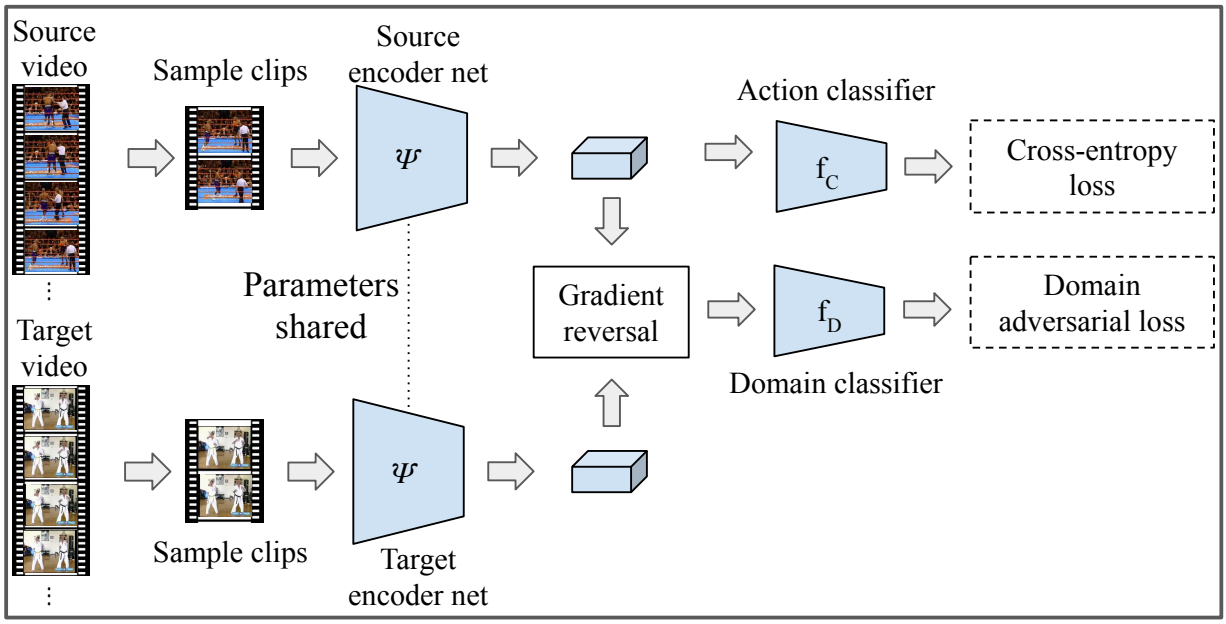
Clip order prediction. To better understand the effect of the proposed COP module, I show class activation maps (CAM) [187] of target videos in Figure 4.6. I compute the CAM of the center (8th) frame of a 16 frames long clip. I show CAMs from models with and without COP (baseline/ours). The baseline without COP tends to focus more on the scene context. However, the proposed model with COP focuses more on the actual human action (typically around the actors). As the model with COP focuses more on the actual action, it generalizes better to a new domain with a completely

different scene cf. the model without COP, which is heavily biased by the scene context.

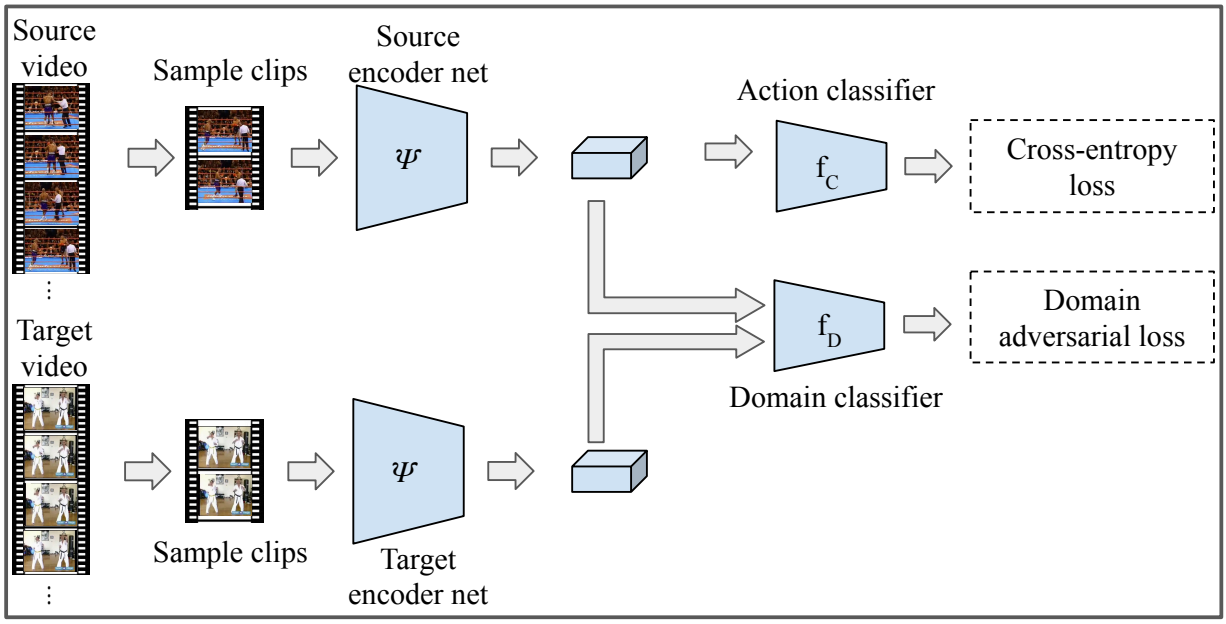
Clip attention. I show the center frames of 4 clips per video with the clip attention module based selection. The videos demonstrate how the proposed clip attention module focuses more on the action class relevant clips and less on the irrelevant clips with either highly occluded actors or mainly background. E.g., in the fencing video in the second row, first and the fourth clips are not informative as the actor, or the object (sword), is highly occluded or cropped. Thus, aligning the features from the relevant second and the third clips is encouraged. Similarly, in the golf video of the first row, the last clip (green background) is irrelevant to the golf action, and my attention module does not attend to it. However, a model without attention treats all the clips equally.

4.5 Conclusions

I proposed **Shuffle and Attend: Video domain Adaptation** (SAVA), a novel video domain adaptation method with self-supervised clip order prediction and clip attention based feature alignment. I showed that both of the two components contribute to the performance. I achieved state-of-the-art performance on the publicly available HMDB→UCF and Kinetics→Drone datasets. I showed extensive ablation studies to show the impact of different aspects of the method. I also validated the intuitions for designing the method with qualitative results for both the contributions.



(a) Overview of DANN for video



(b) Overview of ADDA for video

Figure 4.5: **Overview of the DANN and ADDA extended for video.** Best viewed with zoom and color.

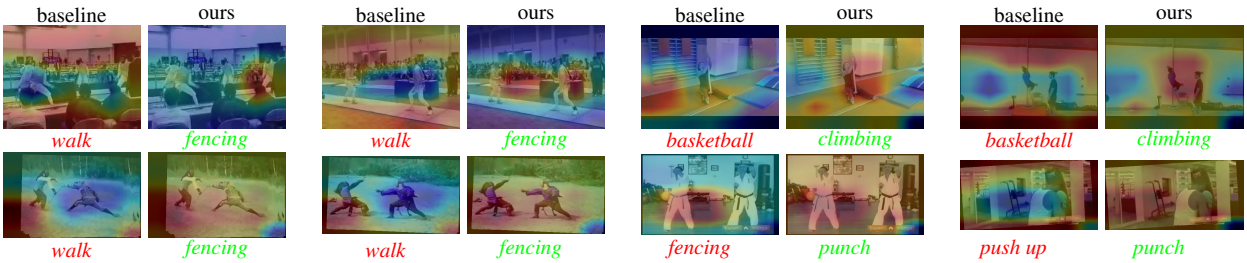


Figure 4.6: **Class activation maps (CAM) on the UCF (first row) and HMDB (second row) datasets.** The actions green are correct predictions, and those in red are incorrect predictions. Here the baseline is ADDA without COP, and ours is ADDA with COP. Note how the COP encourages the model to focus more on human action instead of scene context. Best viewed with zoom and color.

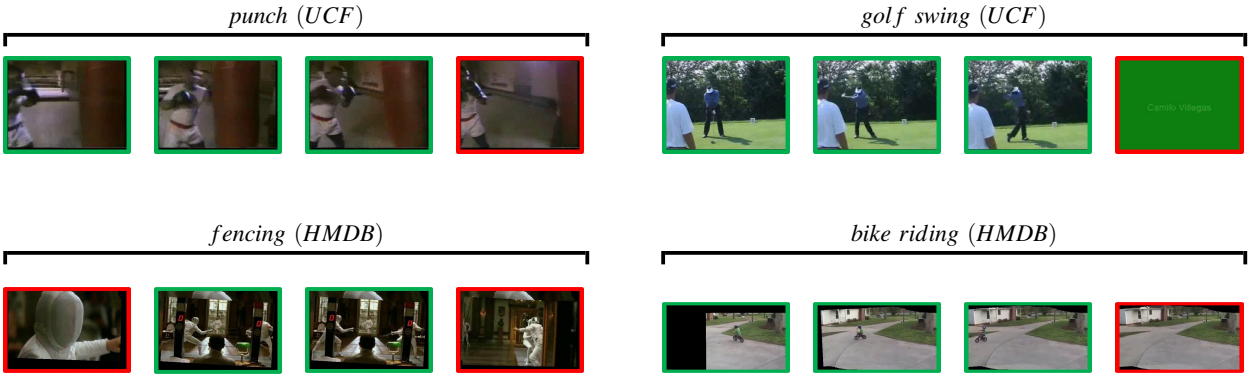


Figure 4.7: **Attention visualization on center frames of 4 clips from 4 videos.** The frames with green borders are given more importance by our attention module cf. those with red borders. Note that our attention module can attend to relevant clips where the action is clearly visible, while the baseline without attention would align all clips equally, even those where the actor is missing or highly occluded. Best viewed with zoom and color.

Chapter 5

Semi-Supervised Learning with Target Data

5.1 Introduction

Deep neural networks have shown rapid progress in video action recognition [18, 43, 134, 145, 170]. However, these approaches rely on training a model on a massive amount of *labeled* videos. For example, the SlowFast networks [43], R(2+1)D [145], I3D [18] are pre-trained on the Kinetics dataset [80], which contains $\sim 300K$ manually labeled and temporally trimmed videos. The dependency on large-scale annotated video datasets is not scalable because manual labeling of videos is expensive, time-consuming, and error-prone. Hence, it is of great interest to develop video action recognition models that can learn from limited labeled data.

The state-of-the-art video recognition models suffer from a significant performance drop when only a limited amount of labeled training data is available. For example, an R(2+1)D model achieves only 19.24% accuracy on the UCF-101 dataset using 5% of the labeled training data, compared to the 55.67% using the full training set when it is trained from scratch. To improve the performance in a *low labeled data regime*, I need to devise methods that can capitalize on both labeled and unlabeled data during training.

I build upon the state-of-the-art semi-supervised learning algorithm, FixMatch [139], to tackle the video action recognition problem in the low labeled data regime. FixMatch shows that generating a one-hot pseudo label from the weakly-augmented example to supervise the strongly-augmented

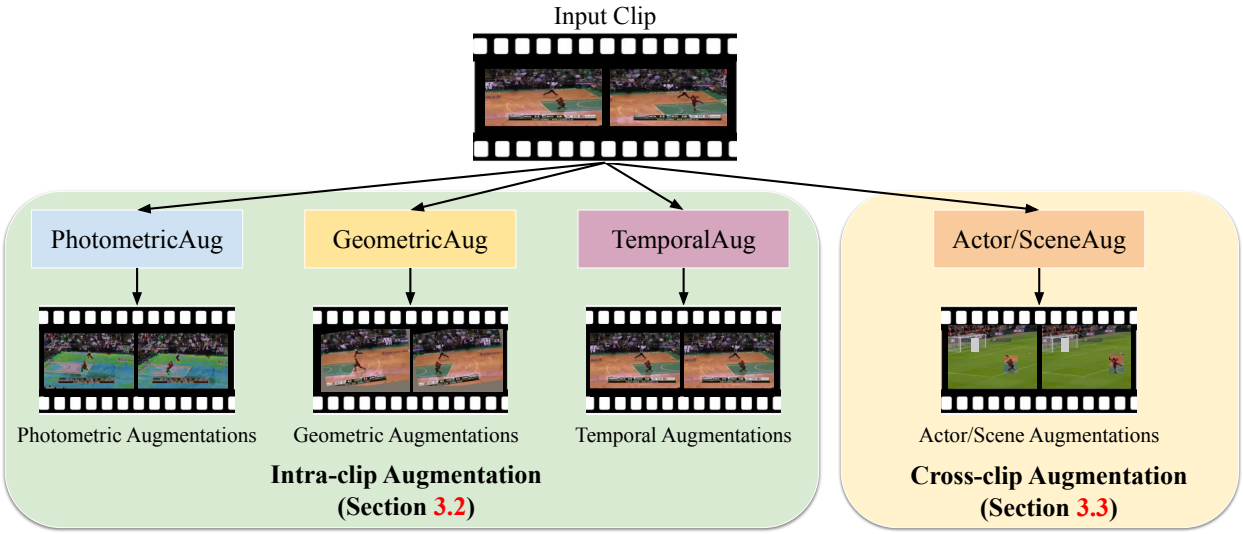


Figure 5.1: **Strong data augmentation for video data.** We extensively explore data augmentation strategy for semi-supervised video representation learning from different perspectives: photometric, geometric, temporal, and actor/scene.

counterpart is critical to its success. Here, the weak augmentation refers to standard augmentation methods (e.g., , random scaling, cropping, etc.), while the strong augmentation (e.g., , CTAugment [10], RandAugment [33]) provides stronger and more diverse transformation for the input data. Enforcing the consistency between the weakly and strongly-augmented example helps the model learn to capture the invariance better and mitigate the risk of overfitting.

While strong data augmentation in the image domain has been extensively studied [32, 33, 36, 97, 177], data augmentation techniques for videos are relatively under-explored. In this work, I aim to address a question: "what kind of *invariance* do I need for semi-supervised video action recognition?" The answer to this question directly informs us how to design a good (strong) data augmentation for consistency regularization in the semi-supervised learning framework. In light of this, I investigate different augmentation strategies to capture various types of visual invariances.

Photometric (color) and geometric (spatial) invariance. Photometric and geometric based augmentations are commonly used in image recognition [139, 169]. I empirically find that applying photo-

metric and geometric augmentation in a *temporally-coherent manner* is crucial for semi-supervised video recognition.

Temporal invariance. I introduce temporal data augmentation operations specifically for video data, including (1) T-Half for temporal occlusion and (2) T-Drop for speed and temporal occlusion invariance, (3) T-Reverse for temporal order invariance, and study the effect of the proposed temporal augmentations. More specifically, I randomly discard the first (or last half) of the frames in a clip in T-Half, while I replace every frame with its previous frame by a random chance in T-Drop. I reverse the order of the frames in T-Reverse.

Scene invariance. As actions often occur in context (e.g., in a specific scene), training a model without diverse data leads to an unwanted bias toward a spurious association with the background scene. I propose ActorCutMix to mitigate such scene biases [28, 95, 96] by copying all humans in an input video and then pasting them into another input video. With ActorCutMix, I can effectively augment various background scenes for each action and therefore mitigate scene bias.

I empirically show that all these augmentation techniques individually help improve performance. With all the augmentations (as shown in Figure 5.1), i.e., , photometric, geometric, temporal augmentations, ActorCutMix, I achieve favorable results in the low labeled data regime on the UCF-101 [141] and HMDB-51 [85] datasets.

To summarize, I make the following contributions.

- I extensively study the video data augmentation strategies, a relatively under-explored area, in semi-supervised video action recognition.
- I introduce several temporal data augmentations and a scene augmentation strategy for video action recognition.
- My introduced strong data augmentation strategies naturally fit into the state-of-the-art holis-

tic semi-supervised classification framework for video action recognition, leading to a promising performance in the low-data regime. My source code and pre-trained models will be made publicly available.

5.2 Related Work

Semi-supervised learning. Semi-supervised learning (SSL) aims to improve the performance using the abundant unlabeled data, alleviating the need for manual annotations. Most of recent SSL approaches adopt either one of the following two strategies: (1) consistency regularization [87, 88, 103, 123, 143, 169], and (2) entropy minimization [55, 91]. The critical insight of consistency regularization is that a model should generate *consistent* predictions for the same (unlabeled) data undergone different transformations/perturbations. As argued by Grandvalet & Bengio [55], unlabeled data can also be used to encourage a model to have a good separation between classes. To achieve this goal, one can encourage a model to output confident (low-entropy) predictions for the unlabeled data, which is the core idea of entropy minimization. Pseudo-labeling [91] is one of the efficient ways to minimize entropy implicitly.

Recently, holistic approaches [10, 11, 139] that combine both SSL strategies have been proposed to tackle the semi-supervised image classification task effectively. Inspired by the recent success in semi-supervised image classification, I leverage the state-of-the-art FixMatch framework [139] and apply it to the video action recognition problem. My focus, however, is not on improving the algorithmic framework for SSL. Instead, I investigate different types of video data augmentations for consistency regularization.

The most relevant work to ours is VideoSSL [76], whose focus is to design a semi-supervised framework for video recognition. The VideoSSL method differs from my work in two perspectives. First,

VideoSSL does not apply strong/weak dual augmentations to capture the video invariance. Second, VideoSSL uses an additional ImageNet pre-trained model for supervised knowledge distillation.

Data augmentation. A key factor to a successful consistency-based SSL approach is to apply a set of diverse yet reasonable augmentations for the same unlabeled example. Early approaches [88, 123] only apply weak augmentations such as random translation and cropping. In addition to simple geometric transformations, random Gaussian or Dropout noise [4] and adversarial noise [103] have also been proposed for semi-supervised learning, leading to improved performance. Recently, learning-based approaches [32, 97] have been proposed to alleviate the requirement of manually designing the data transformations. Such a network learns to adjust the data augmentation policy according to the feedback on a held-out (labeled) validation set. Leveraging the state-of-the-art data augmentation strategies in supervised classification, recent methods [139, 169] apply strong image space augmentation (e.g., , RandAugment [33]) by cascading standard color jittering, geometric transformations, and regional dropout [36, 177], achieving state-of-the-art results. In addition to perturbing the unlabeled image in the pixel space, several approaches [87, 165] propose to augment the example in the feature space. Most existing works design the data augmentation strategy specifically for *images*. My focuses on video data augmentation have been less explored in the literature. This work aims to study data augmentation in semi-supervised video action recognition from multiple perspectives: photometric, geometric, temporal, and the actor/scene.

Self-supervised learning. Self-supervised methods learn representations by solving *pretext tasks* that do not require manual human annotation. For video related tasks, frame sorting [92], clip order verification [102, 171], speed prediction [9, 39, 155], video-induced visual invariance [147], future prediction [56, 57] have shown promising results.

More recently, contrastive learning has emerged as a powerful tool for learning visual representations [23, 24, 60, 113, 126, 168]. In contrastive learning, a model learns representations by

discriminating instances. Given transformations of input data, these methods encourage the same instance's embeddings similar and all the different instances' embedding dissimilar. Since the self-supervised learning method can learn representation without human annotations, I could apply it to the low labeled data scenario, i.e., , semi-supervised learning. In this work, I compare the proposed method with a few self-supervised video representation methods.

Video recognition models. Recent advances in video recognition (e.g., , two-stream networks [44, 134], 3D CNNs [18, 58, 144, 162], 2D and 1D separable CNNs [145, 170], incorporating long-term temporal contexts [43, 162, 167]) focus on the fully supervised learning setting, where a large amount of human-annotated video data, e.g., , Kinetics-400 [80] with $300K$ labeled videos, is available. In this work, I focus on the relatively under-explored semi-supervised video recognition setting, where I have limited labeled videos and a large amount of unlabeled data.

5.3 Method

I organize this section as follows: First, I formulate the semi-supervised classification task in Section 5.3.1. I then present the *intra-clip* data augmentation strategies (photometric, geometric, temporal) in Section 5.3.2. Next, I propose and study ActorCutMix, a *cross-clip* human-centric data augmentation operation in Section 5.3.3. Lastly, I show how I combine all these data augmentation operations to construct the final data augmentation strategy in Section 5.3.4.

5.3.1 Background: semi-supervised classification

Considering a multi-class classification problem, I define $\mathscr{X} = \{(x_i, y_i)\}_{i=1}^{N_l}$ as the *labeled* set, where x_i is the i-th input data, y_i is the corresponding ground truth label, and N_l is the number of data points within the labeled set. Similarly, I define $\mathscr{U} = \{x_j\}_{j=1}^{N_u}$ as the *unlabeled* set, where N_u

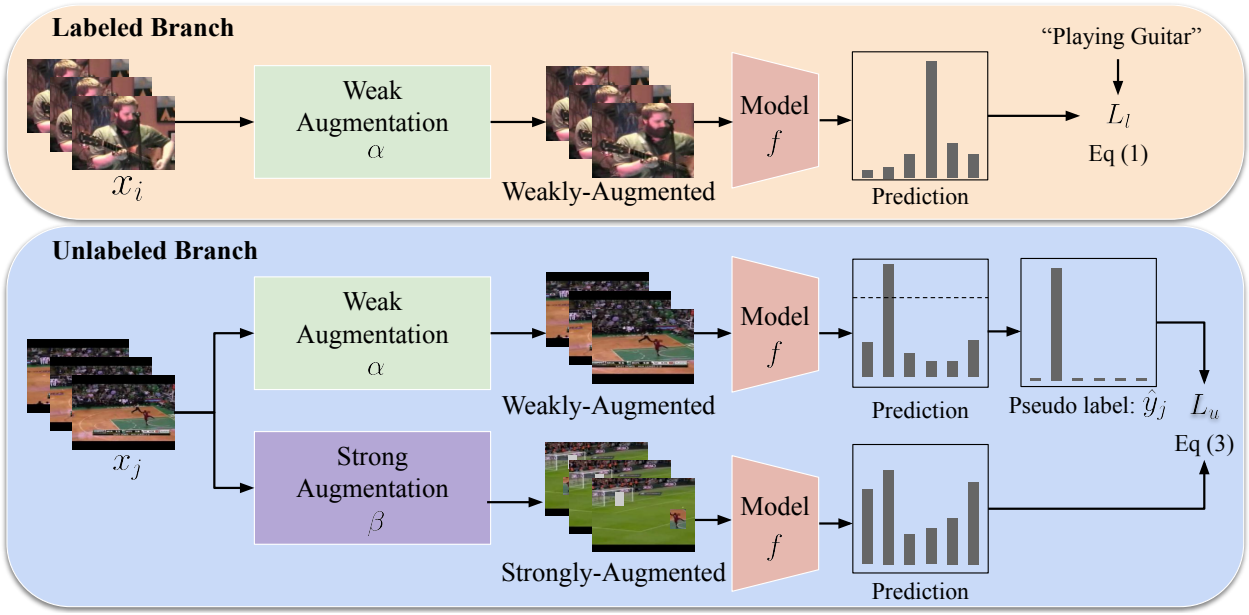


Figure 5.2: **Overview.** The overall training pipeline consists of a supervised branch and an unsupervised branch. We use ground truth labels to supervise the supervised branch. We use pseudo labels generated from weakly-augmented video clips to supervise the strong-augmented counterpart in the unsupervised branch. Best viewed with zoom and color.

is the number of data in the unlabeled set. Usually, I have a small number of labeled examples and a large amount of unlabeled data (i.e., , $N_l << N_u$). I use f_θ to denote a classification model with trainable parameters θ. Lastly, I use $\alpha(\cdot)$ to represent the weak (standard) augmentation (e.g., , random horizontal flip, random scaling, random crop), and $\beta(\cdot)$ to represent the strong data augmentation strategies (my focus in this chapter).

I show an overview of the semi-supervised classification pipeline in Figure 5.2. I denote an input video clip consists of L frames as x_i from now throughout the chapter. Given a mini-batch of labeled data $\{(x_i, y_i)\}_{i=1}^{B_l}$, I minimize the standard cross-entropy loss L_l defined as

$$L_l = -\frac{1}{B_l}\sum_{i=1}^{B_l} y_i \log f_\theta(\alpha(x_i)). \tag{5.1}$$

For a mini-batch of unlabeled data $\{x_j\}_{j=1}^{B_u}$, I enforce the prediction consistency as suggested by

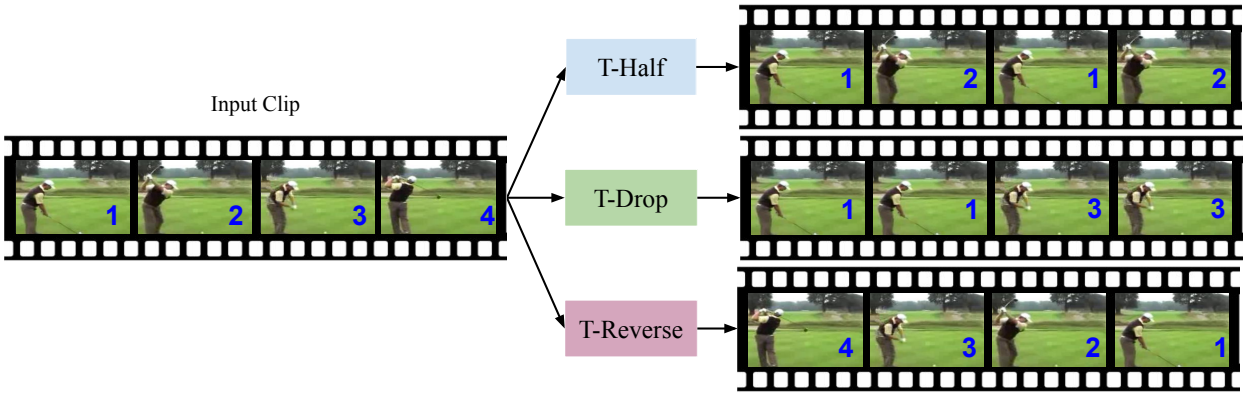


Figure 5.3: **Temporal augmentations.** We introduce three different temporal transformations: T-Half, T-Drop, and T-Reverse to achieve *temporal occlusion*, *speed*, and *order invariances*. Note that the **blue number** of each frame indicates the frame index within the corresponding video clip. We construct the final temporal augmentation by randomly selecting one of these transformed clips (including the original input) for each input video clip. Best viewed with zoom and color.

Sohn et al. [139]. More specifically, I generate pseudo-label $\hat{y}$ for the unlabeled data by confidence thresholding

$$C = \{x_j | \max f_\theta(\alpha(x_j)) \geq \tau\}, \tag{5.2}$$

where τ denotes a pre-defined threshold, and C is the confident example set for the current mini-batch. I then convert the confident model predictions $f_\theta(\alpha(x_j))$ to one-hot labels $\hat{y}_j$ by taking *argmax* operation. I optimize a cross-entropy loss L_u on the confident set of unlabeled examples.

$$L_u = -\frac{1}{B_u} \sum_{x_j \in C} \hat{y}_j \log f_\theta(\beta(x_j)). \tag{5.3}$$

My overall training objective is the summation of the two loss functions.

$$L = L_l + \lambda_u L_u. \tag{5.4}$$

I set $\lambda_u = 1$ as I empirically find it leads to good results.

5.3.2 Intra-clip data augmentation

Temporally-coherent photometric and geometric augmentation. Photometric (color) and geometric (spatial) augmentation strategies are widely used in supervised [33], self-supervised [23, 60], and semi-supervised [139] image classification tasks. Similar to image recognition, it is natural to apply photometric and geometric transformations to a video. However, as I validated with an ablation study in Section 5.4.3, individually applying state-of-the-art photometric and geometric augmentation (e.g., , RandAugment [33]) on each video frame leads to sub-optimal performance. I conjecture that the random transformation for each frame breaks the temporal coherency of a video clip. As a result, the video recognition model may not extract motion cues from these augmented video clips. I instead apply exactly the same photometric and geometric transformations for every frame to maintain the temporal consistency within a sampled video clip. More specifically, I sample two basic operations from a pool of photometric and geometric transformations (as in FixMatch [139]) and then apply them sequentially for each video clip.

Temporal augmentation. In addition to color space and spatial dimension, the temporal dimension is also essential for video data. I introduce three different types of temporal data augmentation operations: (1) T-Half, (2) T-Drop, and (3) T-Reverse, and study the effect of them. I illustrate the three temporal augmentations in Figure 5.3. First, to avoid a video recognition model focusing too much on particular frames instead of understanding the temporal context, I randomly drop some frames within a video clip. Randomly dropping frames is conceptually similar to the Cutout [36] operations in the spatial dimension. I implement this temporal version of Cutout in two ways: 1) I randomly discard the second half of a video clip and fill in the empty slots with the first half, which I refer to as T-Half; 2) For each frame in a video clip, I randomly replace it with its previous frame with a probability of $p = 0.5$, which I refer to as T-Drop. In addition to dropping part of the information, the latter also simulates *speeding-up* (frame indexes: $23 \rightarrow 13$) and *slowing-down*

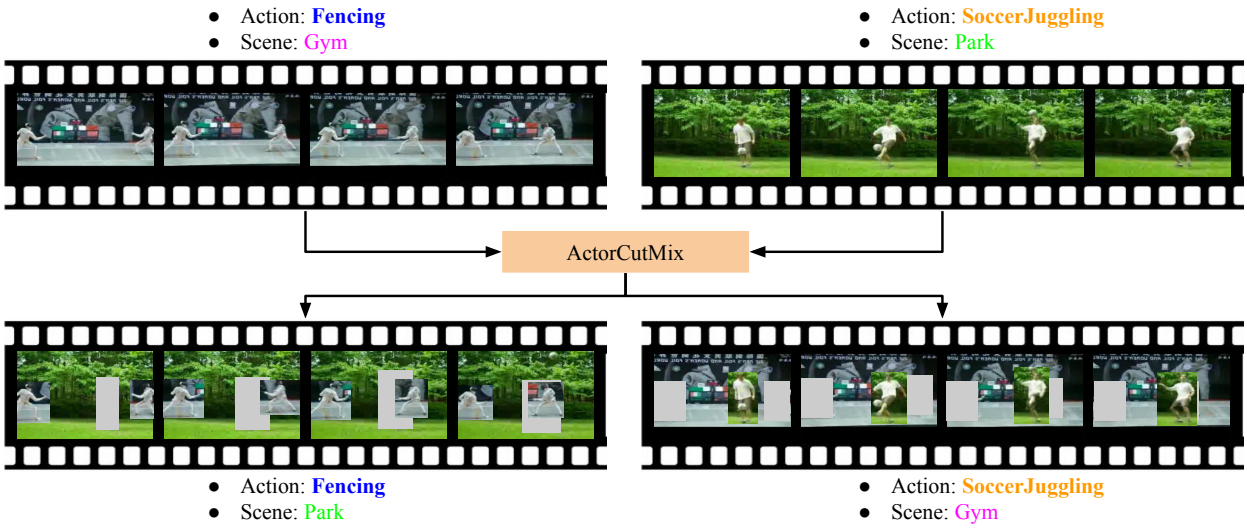


Figure 5.4: **ActorCutMix scene augmentation.** We introduce an actor/scene augmentation method to achieve *scene invariance* in the semi-supervised action recognition task. We transform two video clips by swapping the background regions of the video clips. We generate human/background masks by running an off-the-shelf human detector. We smooth labels in order to mitigate data corruption due to the missing/false-positive human detections. Best viewed with zoom and color.

(frame indexes: $12 \rightarrow 11$) within a video clip, encoding the speed invariance. Second, I argue that many actions have temporal order invariance. Thus, a video recognition model should classify these video clips no matter in the original order or reversely. Inspired by this, I introduce the third temporal augmentation operation, transforming a video clip by reversing its temporal order, referred to as T-Reverse. As validated in the ablation study (Section 5.4.3), all three operations are beneficial for SSL video action recognition. Thus, I include all these three operations to form my final temporal augmentation. Since these operations are exclusive to each other, I put them in an operation pool (including the *identity* operation) and randomly sample one for each input video clip.

5.3.3 Cross-clip data augmentation

ActorCutMix. It is shown that there are severe scene representation biases in the popular action recognition datasets such as UCF-101, HMDB-51, Kinetics-400, Charades, ActivityNet [28, 95, 96]. Models trained on a biased dataset may capture unwanted bias. A biased model is likely to fail when tested for the new data with a different distribution than the training distribution. In this work, to address the scene bias problem, I propose a new human-centric video data augmentation method, ActorCutMix. The operation of the proposed ActorCutMix is similar to that of CutMix [177]. However, my method differs in motivation. Specifically, the goal of CutMix [177] is to achieve to *occlusion robustness*. In contrast, my ActorCutMix aims to improve *scene invariance* (scene debiasing).

As shown in Figure 5.4, ActorCutMix generates new training examples $(\tilde{x}_A, \tilde{y}_A)$, $(\tilde{x}_B, \tilde{y}_B)$ by swapping the background regions in the two training examples $(x_A, \hat{y}_A)$, and $(x_B, \hat{y}_B)$ in a mini-batch. I defined the swapping operation as follows:

$$\begin{aligned} \tilde{x}_A &= m_A \odot x_A + (\mathbf{1} - m_A) \odot (\mathbf{1} - m_B) \odot x_B, \\ \tilde{y}_A &= \lambda \hat{y}_A + (1 - \lambda) \hat{y}_B. \end{aligned} \tag{5.5}$$

Here, m is the binary masks for a video clip, with a value of 1 for human regions and 0 for the background regions, $\odot$ represents the element-wise multiplication, and λ is a combination ratio for label smoothing. $(\tilde{x}_B, \tilde{y}_B)$ can be generated similarly. I generate the human mask m by running an off-the-shelf human detection algorithm [61], without fine-tuning on the dataset used in this work. I run the human detector on the datasets offline and store the detection results. I load the cached human bounding boxes during training. I demonstrate that the proposed ActorCutMix significantly improves the semi-supervised action recognition performance in Section 5.4.

Label smoothing. A straightforward way to select the combination ratio λ in Eq. (5.5), is to set it according to the ratio of the foreground mask. Assume $x_A \in \mathscr{R}^{T\times H\times W\times 3}$, then I can compute the foreground ratio

$$\gamma = \frac{\sum m_A}{THW}, \tag{5.6}$$

and then use γ as the combination ratio λ as in CutMix. I emphasize that the ActorCutMix and CutMix have different purposes of label smoothing. The purpose of label smoothing in CutMix is to provide multiple labels for a single training image, as a training image consists of information from two different classes (e.g., , a dog and a cat). In contrast, I smooth labels in ActorCutMix in order to mitigate data corruption due to the missing/false-positive human detections. Ideally, suppose I have a perfect human detector. In this case, the pseudo label of the video clip x_A is $\tilde{y}_A = \hat{y}_A$. In other words, I aim to recognize the *human action* instead of the background scene. However, due to missing/false-positive human detections, an augmented video clip could potentially contain humans performing different actions from different clips. Therefore, to prevent model confusion, I apply label smoothing with a higher weight for the label of the (potentially corrupted) human action $\hat{y}_A$ and a lower weight for the label of the (potentially corrupted) background scene $\hat{y}_B$, via the following scaling function:

$$\lambda = -(\gamma-1)^{\alpha}+1, \gamma \in [0,1] \tag{5.7}$$

where I empirically find $\alpha = 4$ yields good results.

5.3.4 Combining different data augmentations

Algorithm 1 outlines the proposed (strong) data augmentation strategy.

Combining photometric-geometric and temporal augmentations. I combine the photometric-

Algorithm 1: Strong video data augmentation

Input : A mini-batch of unlabeled video clips X and the corresponding human mask M

Draw a sample p from uniform distribution $U(0,1)$
If $p > 0.5$
  Reverse the order of the batch dimension to get X' and M'
  $\tilde{X}, \tilde{X}' = \textbf{ActorCutMix}(X, X', M, M')$
else
  for each video clip x in X
    Sample op_1, op_2 from photometric-geometric op pool, op_3 from temporal op pool
    $\tilde{x} = \textbf{PhotometricGeometricAug}(x, op_1, op_2)$
    $\tilde{x} = \textbf{TemporalAug}(\tilde{x}, op_3)$
  end for
end if

Return : Strongly-augmented video clips $\tilde{X}$

geometric and temporal augmentations by cascading both of them. This can be regarded as a spatial-temporal counterpart of RandAug [33]: Randomly sample two operations from photometric and geometric augmentation strategies, and then sample one operation from the temporal augmentation strategies. I refer this combined augmentation as *intra-clip augmentation*.

Combining intra-clip and cross-clip augmentations. Now I have both intra-clip augmentation (photometric-geometric-temporal) and cross-clip augmentation (ActorCutMix). To combine these two very different augmentation strategies, I propose randomly applying either one of them for each mini-batch. I validate the effectiveness of this combination in Section 5.4.3.

5.4 Experimental Results

5.4.1 Experimental setup

Dataset. I evaluate the proposed method on the public action recognition benchmarks: UCF-101 [141] and HMDB-51 [85]. UCF-101 consists of 13,320 videos with 101 action classes.

HMDB-51 consists of 6,766 videos with 51 action classes. For semi-supervised learning evaluation, I split the datasets following Jing et al. [76].

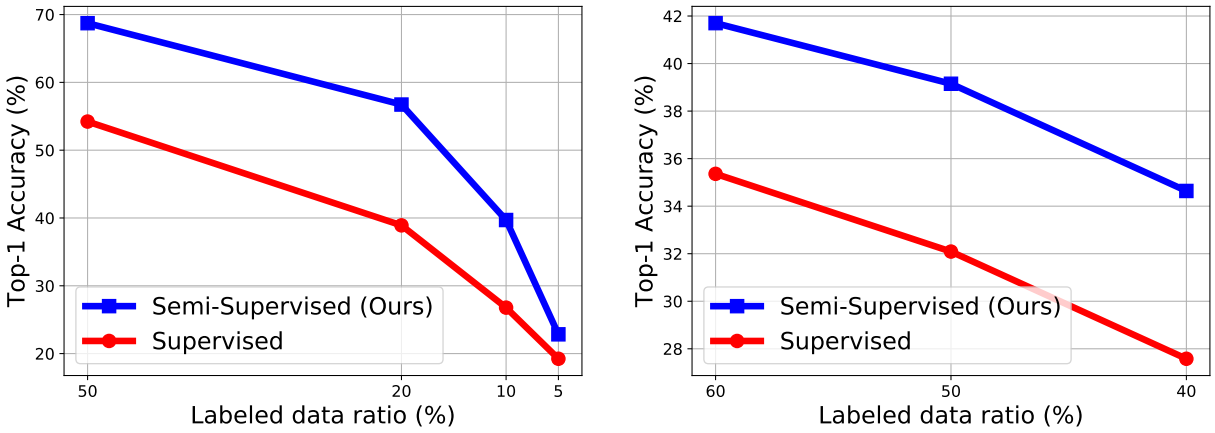
Evaluation metrics. For all the datasets I evaluate, I report top-1 accuracy and compare it to other methods.

Compared methods. As a first baseline, I train a model with only the labeled data. I call it a *supervised baseline*. I compare my method with VideoSSL [76] and several semi-supervised methods adapted to video (i.e., , PseudoLabel [91], MeanTeacher [143], S4L [179]). With the same amount of data, one can also pre-train the model using all the available data (using self-supervised learning) and then fine-tune the model with the small amount of labeled data. Thus, I establish another type of baselines by fine-tuning the self-supervised video pre-trained models on the labeled data. I choose two recent state-of-the-art self-supervised methods for video representation, i.e., , VCOP [171] and DPC [56].

Implementation details. I implement my method on top of the publicly available mmaction2 codebase. [1] Unless specified, I use the R(2+1)D model [145] with ResNet-34 [62] as the feature extraction backbone. To better understand the effect of my augmentation techniques, I initialize the model with random weights (as opposed to using models with supervised pre-training on ImageNet or Kinetics). I randomly sample eight frames with eight-frame intervals from a video to construct a clip. For the supervised learning baseline, I use a batch size of 16 clips for each GPU. I use a mini-batch of five clips from labeled data and five clips from unlabeled data for each GPU for my semi-supervised learning method. I train my models using 8 RTX 2080 Ti GPUs.

I use SGD with momentum as my optimizer, with an initial learning rate of 0.2, a momentum value of 0.9, and a weight decay value of $1e-4$. I use the cosine annealing policy for learning rate decay. I also adopt the synchronous batch normalization across 8 GPUs. For the UCF-101 dataset [141],

[1] https://github.com/open-mmlab/mmaction2



(a) UCF-101 (b) HMDB-51

Figure 5.5: **Improvement over the supervised baseline.**

I train my method for 360 epochs (w.r.t unlabeled data). For the HMDB-51 dataset [85], I train my method for 600 epochs (w.r.t unlabeled data). I list the values of the other hyper-parameters as below.

Table 5.1: **Hyper-parameters.**

Symbol	Description	Value
τ	Pseudo label threshold (Eq. (2))	0.95
λ_u	Unlabeled loss weight (Eq. (4))	1.0
α	Scaling factor for label smoothing (Eq. (7))	4.0

5.4.2 Improvement over supervised baseline

I validate my method's effectiveness by comparing it with the supervised baseline trained only on the limited labeled video data. As shown in Figure 5.5, my method consistently outperforms the supervised baseline by a large margin across different label ratios in both datasets. The performance improvement highlights the effectiveness of leveraging unlabeled video data.

Table 5.2: **Ablation study.** Please refer to the main text for detailed descriptions.

(a) Temporally-coherent color-spatial aug.

Strategy	Top-1 acc.
Supervised baseline	38.91
Per-frame	44.17
Temporally-coherent	**53.37**

(b) Temporal augmentation

Strategy	Top-1 acc.
T-Half	42.77
T-Drop	43.14
T-Reverse	43.40
TemporalAll	**44.07**

(c) ActorCutMix augmentation

Strategy	Top-1 acc.
Supervised baseline	38.91
w/o label smoothing	42.82
w/ label smoothing	**45.28**

(d) Combining photo.-geo. and temporal augmentations

Strategy	Top-1 acc.
Supervised baseline	38.91
Randomly sample one	53.37
Cascaded	**54.48**

(e) Combining intra- and inter-clip augmentations

Strategy	Top-1 acc.
Supervised baseline	38.91
Cascaded	50.89
Randomly sample one	**56.73**

5.4.3 Ablation study

To answer the question, "what kind of *invariance* do I need in the semi-supervised video action recognition problem?", I conduct ablation experiments on the 20% labeled data split of the UCF-101 dataset [141]. In the following, I validate each design choice of the proposed framework.

Temporally-coherent photometric-geometric augmentation. I first conduct an ablation experiment to study the necessity of applying *temporally-coherent* photometric-geometric augmentation. As shown in Table 5.2(a), although applying photometric-geometric augmentation individually in a frame-by-frame manner improves upon the supervised baseline (38.91% $\rightarrow$ 44.17%), the proposed temporally-coherent augmentation further enhances the performance by a large margin (44.17% $\rightarrow$ 53.37%). The results validate my hypothesis that preserving motion cues in the augmented videos leads to improved results.

Temporal augmentation. Next, I study the effectiveness of the proposed temporal augmentation and its atomic operations. As shown in Table 5.2(b), all of the atomic operations are beneficial to

the recognition accuracy. The results validate my motivation: Temporal occlusion and order invariances are beneficial for semi-supervised video action recognition. With all the atomic temporal operations combined, the proposed temporal augmentation further improves the accuracy.

ActorCutMix augmentation. I validate the proposed human-centric data augmentation, ActorCutMix in Table 5.2(c). ActorCutMix shows moderate improvement over the baseline without label smoothing. With label smoothing, ActorCutMix shows even more significant performance improvement. The results validate that the capturing scene invariance improves recognition accuracy. My results suggest that label smoothing can mitigate data corruption due to the missing/false-positive human detections.

Combining photometric-geometric and temporal augmentations. In this ablation experiment, I study how to combine photometric-geometric augmentation and temporal augmentation. I explore an alternative combination strategy: sample only one of them and apply it for a video clip. As shown in Table 5.2(d), my default cascaded strategy leads to better performance, indicating that combining photometric-geometric and temporal augmentations in a cascaded manner is more effective.

Combining intra-clip and cross-clip augmentations. Similarly, I study how to combine both intra-clip (photometric-geometric-temporal) and cross-clip (ActorCutMix) data augmentations. A straightforward approach is to combine these two types of augmentations in a cascaded fashion, the same as intra-clip transformation combinations. Surprisingly, as shown in Table 5.2(e), I find that the cascaded data augmentation is even *worse* than applying the intra-clip augmentation alone. My intuition is that cascading both intra-clip, and cross-clip augmentation produces severely distorted video clips that no longer resemble natural videos. As a result, cascading the two augmentation hurts pseudo labels' performance to supervised the strongly-augmented branch. Hence, I randomly apply only one data augmentation selected from intra-clip *or* cross-clip for each input video clip.

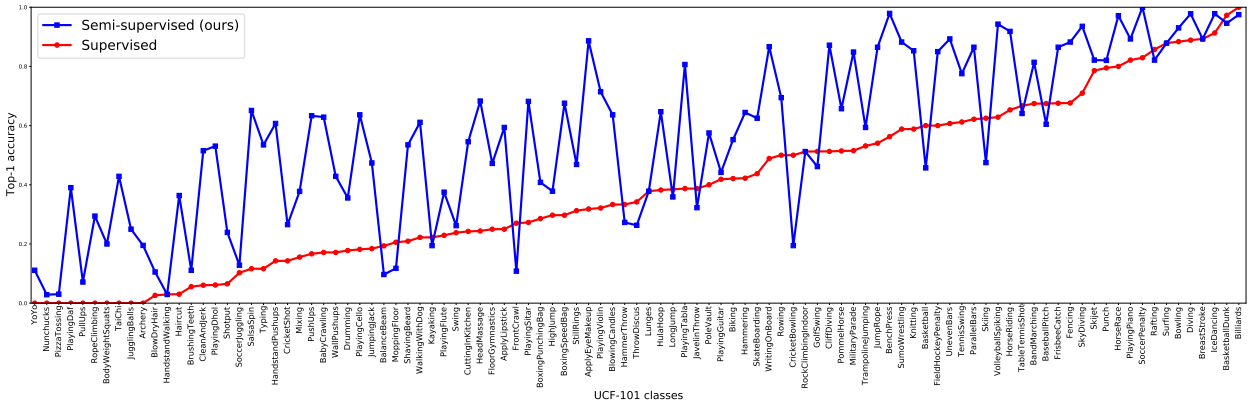


Figure 5.6: **Class accuracy.** We compare the class accuracies of the supervised baseline and our semi-supervised model on the UCF-101 dataset. We sort the classes in ascending order of the baseline accuracy. Best viewed with zoom and color.

Different Initialization. In this study, I aim to test if the proposed data augmentation strategy, together with the semi-supervised consistency regularization framework, can still improve over the *supervised baseline*, given a strong pre-trained model as initialization. I here choose an off-the-shelf R(2+1)D-34 model trained on the full Kinetics-400 dataset [80] as the initialization and train it on the 20% label ratio split of the UCF-101 dataset [141]. As shown in Table 5.3, although the improvement over the supervised baseline is not as large as the boost in the train-from-scratch setting, the proposed method can still achieve a sizable improvement (72.40% $\rightarrow$ 77.37%). The results validate the general applicability of the proposed method in practical scenarios.

Table 5.3: **Effect of using different initialization.**

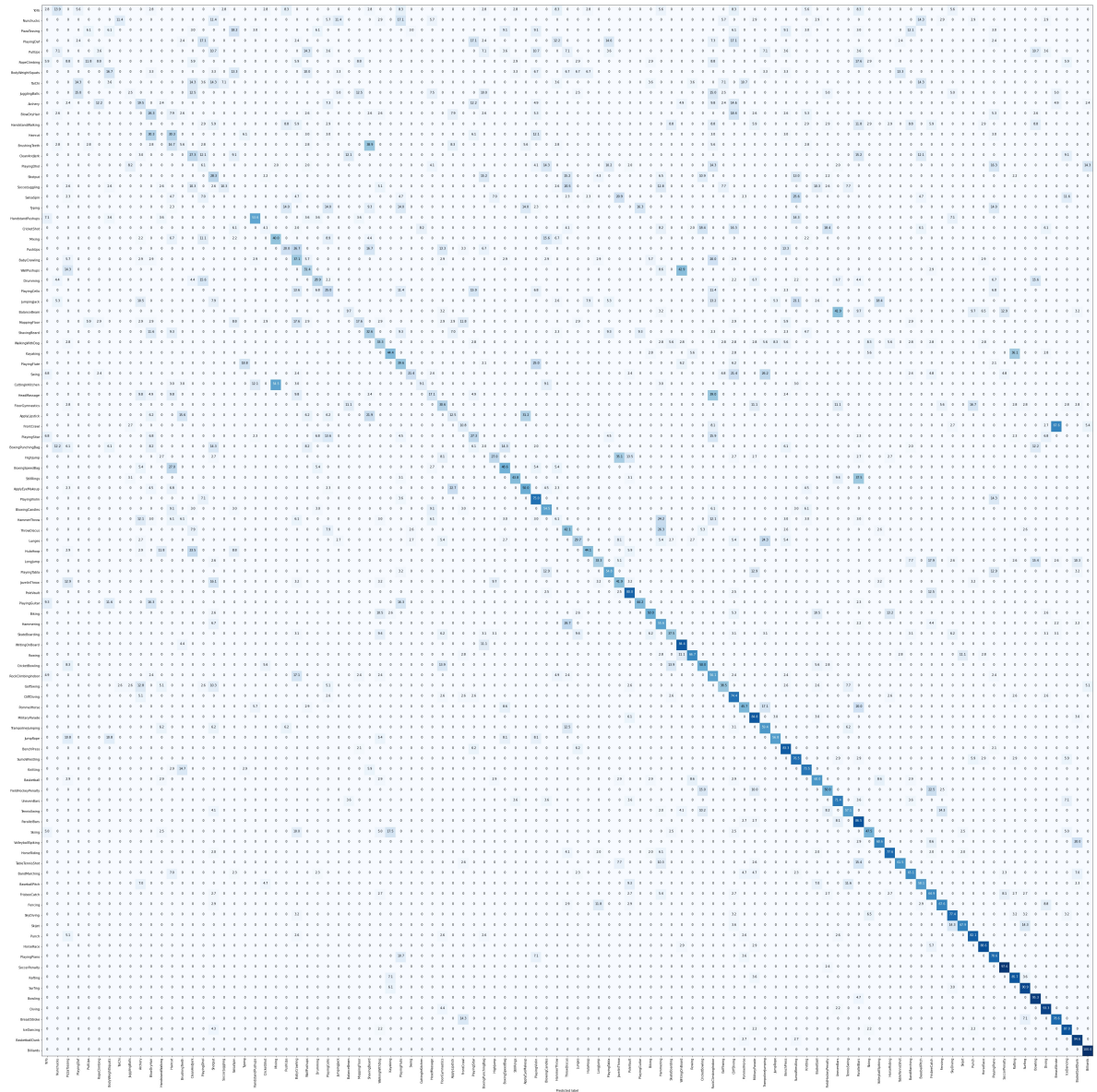
	Initialization	
	Random	Kinetics-400
Supervised	38.91	72.40
Ours	**56.73**	**77.37**

5.4.4 Error Analysis

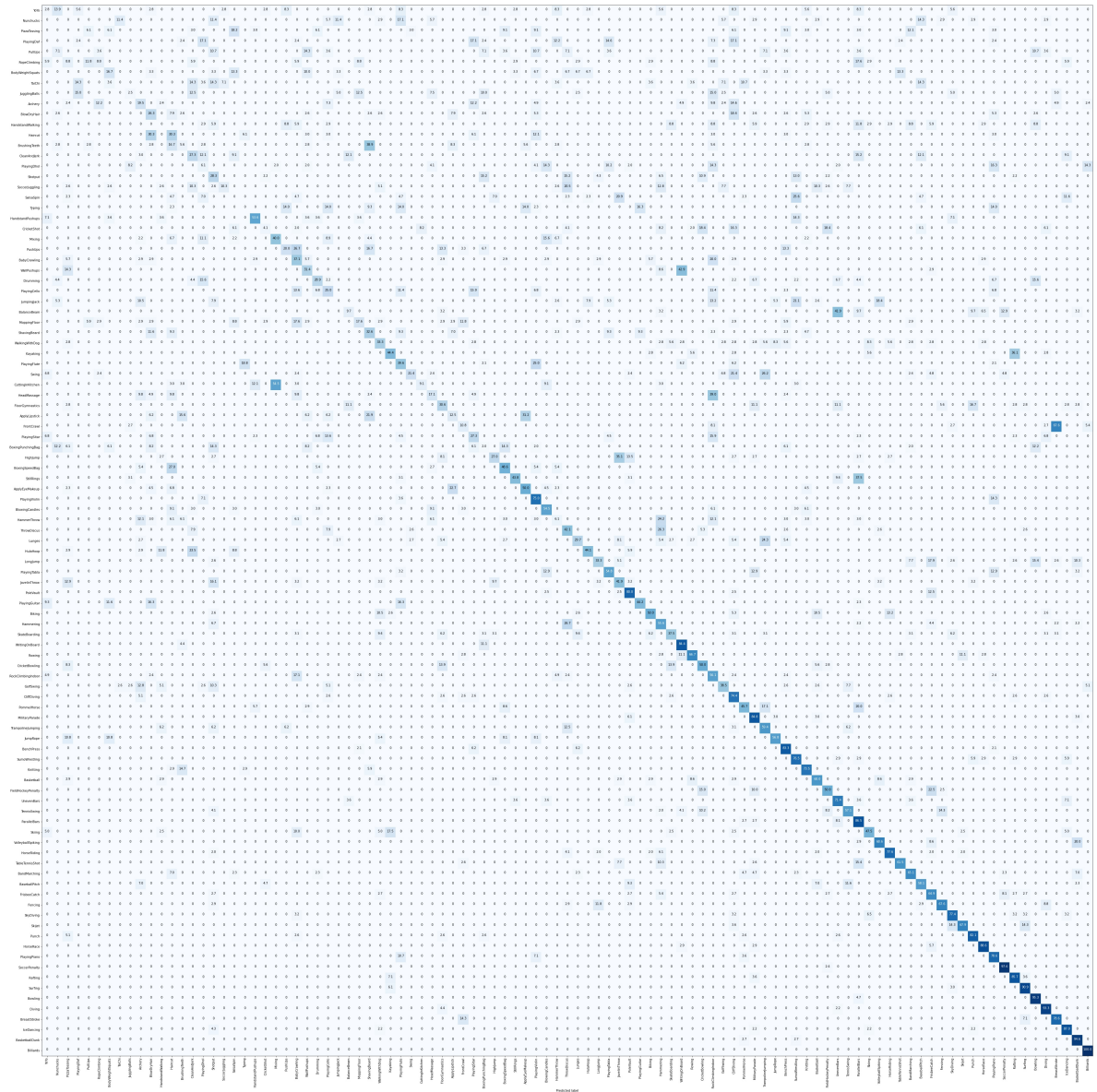
In Figure 5.6, I visualize the class accuracies of the supervised baseline (red curve) and my semi-supervised model with all augmentations (blue curve) on the UCF-101 datasets. I sort the classes in ascending order of the baseline accuracy. Temporal augmentations could reduce the baseline error rate for the classes such as “BodyWeightSquats" and “Haircut".

In Figure 5.7, I visualize UCF-101 (20% label ratio) confusion matrices of the (i) supervised baseline, and my semi-supervised models with (ii) temporal augmentation only, (iii) ActorCutMix only, and (iv) all augmentations (i.e., photometric, geometric, temporal, and scene).

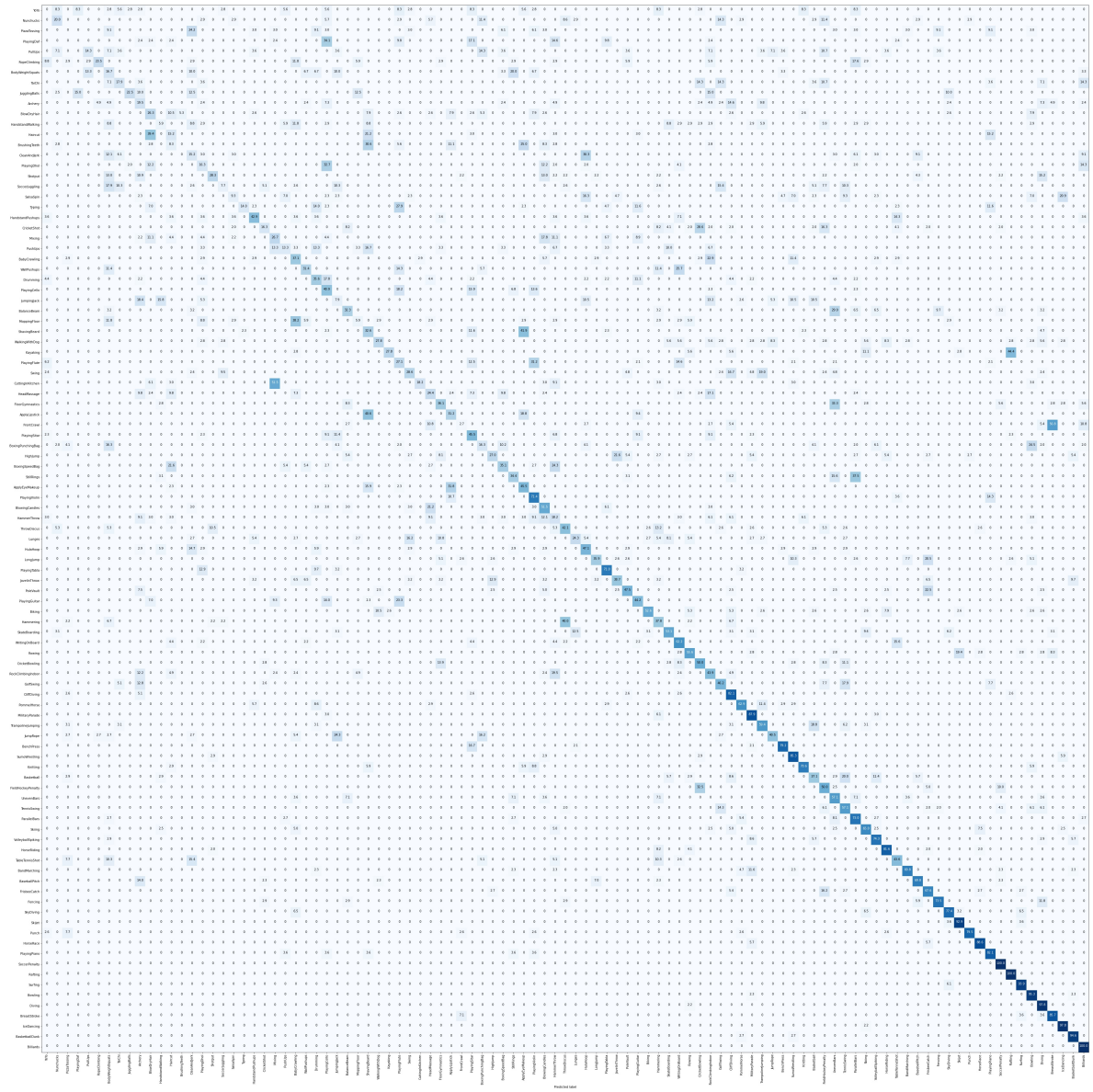
The baseline model confused “BodyWeightSquats" with “TableTennisShot". However, my model with temporal augmentation reduces this type of error significantly (26.7% $\rightarrow$ 13.3%). Temporal augmentations such as T-Half and T-Drop are essentially providing a model less information. Therefore, they could encourage the model to focus more on the fine-grained details related to human actions such as human pose. Compared to the baseline, ActorCutMix reduces the error rate of the classes such as “BlowDryHair" and “ShotPut". The baseline is confused “BlowDryHair" with “ApplyLipstick," which has similar scenes, e.g., , bedroom and restroom. In contrast, ActorCutMix significantly reduces this type of error, (18.4% $\rightarrow$ 7.9%). The results imply that ActorCutMix provides various scenes and results in scene invariance of my model. Additional invariances to photometric/geometric transformations help my model not to use shortcuts. Using all augmentations significantly reduces the error rate of the baseline for the class “PlayingDaf" with “PlayingFlute" (26.8% $\rightarrow$ 4.9%).



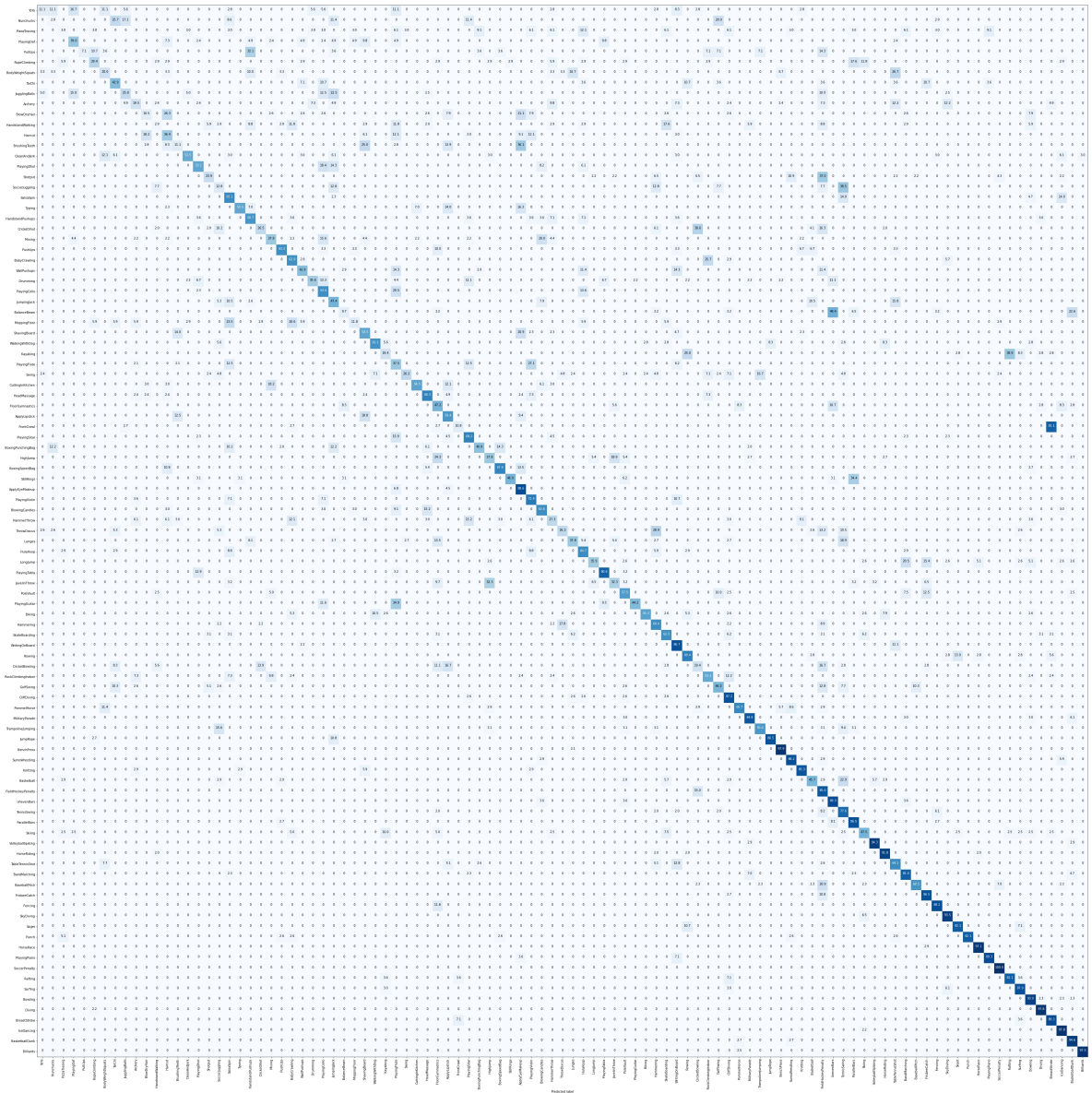
(a) Supervised baseline



(b) Temporal augmentations only



(c) ActorCutMix augmentation only



(d) All augmentations

Figure 5.7: **UCF-101 confusion matrix visualization.** We compare the confusion matrices of the i) supervised baseline, and our semi-supervised models with ii) temporal augmentation only, iii) ActorCutMix only, and iv) all augmentations. We sort the classes in ascending order of the baseline accuracy. Best viewed with zoom and color.

Table 5.4: **Results on UCF-101.** Note that VideoSSL [76] uses an additional ImageNet pre-trained model for knowledge distillation. The best performance is in **bold** and the second best is underlined.

Method	Backbone	w/ ImageNet pre-trained	Label ratio (%)			
			50	20	10	5
PL [91]	3D ResNet-18	-	47.5	37.0	24.7	17.6
MT [143]	3D ResNet-18	-	45.8	36.3	25.6	17.5
S4L [179]	3D ResNet-18	-	47.9	37.7	29.1	22.7
VideoSSL [76]	3D ResNet-18	✓	54.3	48.7	**42.0**	**32.4**
VCOP [171]	R(2+1)D ResNet-10	-	67.9	53.1	31.1	14.4
DPC [56]	R(2+3)D ResNet-18	-	53.1	38.9	25.3	19.2
Ours	R(2+1)D ResNet-18	-	64.2	52.5	39.9	26.1
Ours	R(2+1)D ResNet-34	-	**68.7**	**56.7**	39.7	22.8

5.4.5 Comparing with the state of the arts

Lastly, I compare my method with several existing semi-supervised learning approaches and two recent video self-supervised learning methods. I first pre-train the models on the entire training set for the self-supervised learning methods and then fine-tune them with the labeled data. As shown in Table 5.4 and Table 5.5, my method consistently achieves promising results compared with other approaches with the same amount of supervision. I observe that VideoSSL [76] achieves state-of-the-art performance in the extremely low label ratios (10% and 5%) in the UCF-101 dataset while being worse in other label ratios. I conjecture that the knowledge distillation from ImageNet pre-trained model plays a crucial role in the extremely low-data regime. At the same time, its improvement becomes marginal quickly as the number of labeled data grows. I also observe that a deeper model could perform worse in the low-data regime, which might be caused by overfitting. Other regularization techniques or better pseudo-labeling might mitigate this issue.

Table 5.5: **Results on HMDB-51.** Note that VideoSSL [76] uses an additional ImageNet pre-trained model for knowledge distillation. The best performance is in **bold** and the second best is underlined.

Method	Backbone	w/ ImageNet pre-trained	Label ratio (%) 60	50	40
PL [91]	3D ResNet-18	-	33.5	32.4	27.3
MT [143]	3D ResNet-18	-	32.2	30.4	27.2
S4L [179]	3D ResNet-18	-	35.6	31.0	29.8
VideoSSL [76]	3D ResNet-18	✓	37.0	36.2	32.7
VCOP [171]	R(2+1)D ResNet-10	-	27.8	26.5	25.7
DPC [56]	R(2+3)D ResNet-18	-	37.2	33.2	31.6
Ours	R(2+1)D ResNet-18	-	**42.5**	**39.5**	33.0
Ours	R(2+1)D ResNet-34	-	41.7	39.2	**34.6**

5.4.6 ImageNet Knowledge Distillation

As discussed in Section 5.4.5, I conjecture that the ImageNet knowledge distillation in VideoSSL [76] may be the key factor to its good performance in the extremely low label ratios (10% and 5%) in the UCF-101 dataset. In this study, I aim to explore whether an additional knowledge distillation loss can be integrated into my training framework and improve performance. Following VideoSSL [76], I adopt an ImageNet pre-trained ResNet-18 to compute a 1000D ImageNet class probability vector for each frame of all the training videos offline. Accordingly, I add an additional classification head to predict the ImageNet probability of each video clip, on top of the feature extraction backbone. I randomly select one frame from each (weakly-augmented) video clip during training time and then use its corresponding ImageNet probability as a soft pseudo label for a cross-entropy loss. The soft pseudo label provides an additional supervisory signal to the feature backbone and the new classification head of the weakly-augmented branch.

Table 5.6: **Effect of ImageNet knowledge distillation on the UCF-101.**

Method	Backbone	#Parameters	w/ ImageNet pre-trained	Label ratio (%)	
				10	5
VideoSSL [76]	3D ResNet-18	33M	✓	42.0	32.4
Ours	R(2+1)D ResNet-18	33M	-	39.9	26.1
Ours (w/ distillation)	R(2+1)D ResNet-18	33M	✓	**52.7**	**49.5**

As shown in Table 5.6, the performance of my method significantly improves with ImageNet distillation compared to my method without ImageNet distillation (26.1% $\rightarrow$ 49.5% for 5% label ratio and 39.9% $\rightarrow$ 52.7% for 10% label ratio). My method with ImageNet distillation shows a favorable performance when compared to the VideoSSL [76].

5.5 Conclusions

In this chapter, I investigate different types of data augmentation strategies for semi-supervised video action recognition. My study shows the importance of (1) temporally-coherent photometric and geometric augmentations, (2) temporal augmentations, and (3) actor/scene augmentation. With all the augmentations, I show promising semi-supervised action recognition performance on the public benchmarks. I hope my insights help facilitate future semi-supervised video action recognition research.

Chapter 6

Conclusions

In this dissertation, I have investigated video action recognition in the context of transfer learning. First, I addressed the problem of learning to mitigate scene bias in video action recognition. I proposed two losses for scene debiasing, i) scene adversarial loss and ii) human-mask confusion loss. The scene debiased model consistently generalizes better on the new tasks and datasets. Second, I addressed the problem of the high labeling cost of target video data. I formulated the problem as i) unsupervised domain adaptation for video action recognition and ii) semi-supervised video action recognition. To tackle the unsupervised domain adaptation problem, I collected a new action recognition dataset captured by drones. Then I proposed a video domain adaptation method with clip attention based feature alignment and self-supervised clip order prediction. The proposed method shows state-of-the-art performance on the public benchmarks. To tackle the semi-supervised video action recognition problem, I explore multiple types of video data augmentations that inject photometric, geometric, temporal, and scene invariances into our model. The proposed method shows favorable results on the public benchmarks.

The main contribution of this dissertation is addressing two crucial problems in transfer learning for video action recognition. i.e., i) Scene bias problem in the source datasets, and ii) high labeling cost for the target datasets. I demonstrated that the proposed solutions could improve action recognition models generalization performance by addressing the two problems in transfer learning for videos. I hope that the dissertation work sheds light on addressing action recognition models' generalization with a limited amount of labeled data.

Bibliography

[1] Ehsan Adeli, Qingyu Zhao, Adolf Pfefferbaum, Edith V Sullivan, Li Fei-Fei, Juan Carlos Niebles, and Kilian M Pohl. Bias-resilient neural network. *arXiv preprint arXiv:1910.03676*, 2019.

[2] Lisa Anne Hendricks, Kaylee Burns, Kate Saenko, Trevor Darrell, and Anna Rohrbach. Women also snowboard: Overcoming bias in captioning models. In *ECCV*, 2018.

[3] Shervin Ardeshir and Ali Borji. Integrating egocentric videos in top-view surveillance videos: Joint identification and temporal alignment. In *ECCV*, 2018.

[4] Philip Bachman, Ouais Alsharif, and Doina Precup. Learning with pseudo-ensembles. In *NeurIPS*, 2014.

[5] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. In *ICLR*, 2016.

[6] Mohammadamin Barekatain, Miquel Martí, Hsueh-Fu Shih, Samuel Murray, Kotaro Nakayama, Yutaka Matsuo, and Helmut Prendinger. Okutama-action: An aerial view video dataset for concurrent human action detection. In *1st Joint BMTT-PETS Workshop on Tracking and Surveillance, CVPR*, 2017.

[7] Loris Bazzani, Alessandra Bergamo, Dragomir Anguelov, and Lorenzo Torresani. Self-taught object localization with deep networks. In *WACV*, 2016.

[8] Harkirat S. Behl, Michael Sapienza, Gurkirt Singh, Suman Saha, Fabio Cuzzolin, and Philip H. S. Torr. Incremental tube construction for human action detection. In *BMVC*, 2018.

[9] Sagie Benaim, Ariel Ephrat, Oran Lang, Inbar Mosseri, William T Freeman, Michael Rubinstein, Michal Irani, and Tali Dekel. Speednet: Learning the speediness in videos. In *CVPR*, 2020.

[10] David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang,

and Colin Raffel. Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring. In *ICLR*, 2019.

[11] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *NeurIPS*, 2019.

[12] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *CVPR*, 2016.

[13] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *NeurIPS*, 2016.

[14] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *CVPR*, 2017.

[15] Pau Panareda Busto, Ahsan Iqbal, and Juergen Gall. Open set domain adaptation for image and action recognition. *TPAMI*, 2018.

[16] Qi Cai, Yingwei Pan, Chong-Wah Ngo, Xinmei Tian, Lingyu Duan, and Ting Yao. Exploring object relation in mean teacher for cross-domain detection. In *CVPR*, 2019.

[17] Peter Carbonetto, Nando De Freitas, and Kobus Barnard. A statistical model for general contextual object recognition. In *ECCV*, 2004.

[18] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

[19] Yu-Wei Chao, Sudheendra Vijayanarasimhan, Bryan Seybold, David A. Ross, Jia Deng, and Rahul Sukthankar. Rethinking the faster r-cnn architecture for temporal action localization. In *CVPR*, 2018.

[20] Minghao Chen, Hongyang Xue, and Deng Cai. Domain adaptation for semantic segmentation with maximum squares loss. In *ICCV*, 2019.

[21] Min-Hung Chen, Zsolt Kira, Ghassan AlRegib, Jaekwon Woo, Ruxin Chen, and Jian Zheng. Temporal attentive alignment for large-scale video domain adaptation. In *ICCV*, 2019.

[22] Min-Hung Chen, Baopu Li, Yingze Bao, Ghassan AlRegib, and Zsolt Kira. Action segmentation with joint self-supervised temporal domain adaptation. In *CVPR*, 2020.

[23] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

[24] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[25] Yuhua Chen, Wen Li, Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Domain adaptive faster r-cnn for object detection in the wild. In *CVPR*, 2018.

[26] Yun-Chun Chen, Yen-Yu Lin, Ming-Hsuan Yang, and Jia-Bin Huang. CrDoCo: Pixel-level domain transfer with cross-domain consistency. In *CVPR*, 2019.

[27] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

[28] Jinwoo Choi, Chen Gao, Joseph CE Messou, and Jia-Bin Huang. Why can't i dance in the mall? learning to mitigate scene bias in action recognition. In *NeurIPS*, 2019.

[29] Jinwoo Choi, Gaurav Sharma, Manmohan Chandraker, and Jia-Bin Huang. Unsupervised and semi-supervised domain adaptation for action recognition from drones. In *WACV*, 2020.

[30] Jinwoo Choi, Gaurav Sharma, Samuel Schulter, and Jia-Bin Huang. Shuffle and attend: Video domain adaptation. In *ECCV*, 2020.

[31] Myung Jin Choi, Joseph J Lim, Antonio Torralba, and Alan S Willsky. Exploiting hierarchical context on a large database of object categories. In *CVPR*, 2010.

[32] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. In *CVPR*, 2019.

[33] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical

automated data augmentation with a reduced search space. In *CVPR Workshops*, 2020.

[34] Achal Dave, Olga Russakovsky, and Deva Ramanan. Predictivecorrective networks for action detection. In *CVPR*, 2017.

[35] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[36] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[37] Santosh K Divvala, Derek Hoiem, James H Hays, Alexei A Efros, and Martial Hebert. An empirical study of context in object detection. In *CVPR*. IEEE, 2009.

[38] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015.

[39] Dave Epstein, Boyuan Chen, and Carl. Vondrick. Oops! predicting unintentional action in video. In *CVPR*, 2020.

[40] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

[41] Chenyou Fan, Jangwon Lee, Mingze Xu, Krishna Kumar Singh, Yong Jae Lee, David J Crandall, and Michael S Ryoo. Identifying first-person camera wearers in third-person videos. In *CVPR*, 2017.

[42] Alireza Fathi, Ali Farhadi, and James M Rehg. Understanding egocentric activities. In *ICCV*, 2011.

[43] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, 2019.

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

[45] Adrien Gaidon, Zaid Harchaoui, and Cordelia Schmid. Temporal localization of actions

with actoms. *TPAMI*, 35(11):2782–2795, 2013.

[46] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015.

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

[48] Chen Gao, Yuliang Zou, and Jia-Bin Huang. ican: Instance-centric attention network for human-object interaction detection. In *BMVC*, 2018.

[49] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Felix Wichmann, Wieland Brendel, and Matthias Bethge. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *ICLR*, 2019.

[50] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. In *NeurIPS*, 2018.

[51] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018.

[52] Rohit Girdhar and Deva Ramanan. Attentional pooling for action recognition. In *NeurIPS*, 2017.

[53] Georgia Gkioxari and Jitendra Malik. Finding action tubes. In *CVPR*, 2015.

[54] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, 2012.

[55] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *NeurIPS*, 2005.

[56] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. In *ICCV Workshops*, 2019.

[57] Tengda Han, Weidi Xie, and Andrew Zisserman. Memory-augmented dense predictive coding for video representation learning. In *ECCV*, 2020.

[58] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *CVPR*, 2018.

[59] Jiawei He, Mostafa S Ibrahim, Zhiwei Deng, and Greg Mori. Generic tubelet proposals for action localization. *WACV*, 2018.

[60] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[61] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017.

[62] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[63] Yun He, Soma Shirakabe, Yutaka Satoh, and Hirokatsu Kataoka. Human action recognition without human. In *ECCV Workshop*, 2016.

[64] Zhenwei He and Lei Zhang. Multi-adversarial faster-rcnn for unrestricted object detection. In *ICCV*, 2019.

[65] Judy Hoffman, Mehryar Mohri, and Ningshan Zhang. Algorithms and theory for multiple-source adaptation. In *NeurIPS*, 2018.

[66] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. CyCADA: Cycle-consistent adversarial domain adaptation. In *ICML*, 2017.

[67] Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. *arXiv preprint arXiv:1612.02649*, 2016.

[68] Rui Hou, Chen Chen, and Mubarak Shah. Tube convolutional neural network (T-CNN) for action detection in videos. In *ICCV*, 2017.

[69] Han-Kai Hsu, Chun-Han Yao, Yi-Hsuan Tsai, Wei-Chih Hung, Hung-Yu Tseng, Maneesh Singh, and Ming-Hsuan Yang. Progressive domain adaptation for object detection. In *WACV*, 2020.

[70] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[71] Arshad Jamal, Vinay P Namboodiri, Dipti Deodhare, and KS Venkatesh. Deep domain adaptation in action space. In *BMVC*, 2018.

[72] Saumya Jetley, Nicholas A Lord, Namhoon Lee, and Philip HS Torr. Learn to pay attention. In *ICLR*, 2018.

[73] Hueihan Jhuang, Juergen Gall, Silvia Zuffi, Cordelia Schmid, and Michael J Black. Towards understanding action recognition. In *ICCV*, 2013.

[74] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *TPAMI*, 35(1):221–231, 2013.

[75] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes. http://crcv.ucf.edu/THUMOS14/, 2014.

[76] Longlong Jing, Toufiq Parag, Zhe Wu, Yingli Tian, and Hongcheng Wang. Videossl: Semi-supervised learning for video classification. *arXiv preprint arXiv:2003.00197*, 2020.

[77] Vicky Kalogeiton, Philippe Weinzaepfel, Vittorio Ferrari, and Cordelia Schmid. Action tubelet detector for spatio-temporal action localization. In *ICCV*, 2017.

[78] Amlan Kar, Nishant Rai, Karan Sikka, and Gaurav Sharma. Adascan: Adaptive scan pooling in deep convolutional neural networks for human action recognition in videos. In *CVPR*, 2017.

[79] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[80] 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 preprint arXiv:1705.06950*, 2017.

[81] Mehran Khodabandeh, Arash Vahdat, Mani Ranjbar, and William G Macready. A robust learning approach to domain adaptive object detection. In *ICCV*, 2019.

[82] Aditya Khosla, Byoungkwon An An, Joseph J Lim, and Antonio Torralba. Looking beyond the visible scene. In *CVPR*, 2014.

[83] Byungju Kim, Hyunwoo Kim, Kyungsu Kim, Sungjin Kim, and Junmo Kim. Learning not to learn: Training deep neural networks with biased data. In *CVPR*, 2019.

[84] Bruno Korbar, Du Tran, and Lorenzo Torresani. Scsampler: Sampling salient clips from video for efficient action recognition. In *ICCV*, 2019.

[85] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. HMDB: A large video database for human motion recognition. In *ICCV*, 2011.

[86] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. HMDB: A large video database for human motion recognition. In *ICCV*, 2011.

[87] Chia-Wen Kuo, Chih-Yao Ma, Jia-Bin Huang, and Zsolt Kira. Featmatch: Feature-based augmentation for semi-supervised learning. In *ECCV*, 2020.

[88] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *ICLR*, 2016.

[89] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Colorization as a proxy task for visual understanding. In *CVPR*, 2017.

[90] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[91] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML Workshop*, 2013.

[92] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequences. In *ICCV*, 2017.

[93] Dong Li, Jia-Bin Huang, Yali Li, Shengjin Wang, and Ming-Hsuan Yang. Weakly supervised object localization with progressive domain adaptation. In *CVPR*, 2016.

[94] Dong Li, Jia-Bin Huang, Yali Li, Shengjin Wang, and Ming-Hsuan Yang. Progressive representation adaptation for weakly supervised object localization. *TPAMI*, 2019.

[95] Yingwei Li, Yi Li, and Nuno Vasconcelos. Resound: Towards action recognition without representation bias. In *ECCV*, 2018.

[96] Yi Li and Nuno Vasconcelos. Repair: Removing representation bias by dataset resampling. *arXiv preprint arXiv:1904.07911*, 2019.

[97] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. In *NeurIPS*, 2019.

[98] Tsung-Yi Lin, Yin Cui, Serge Belongie, and James Hays. Learning deep representations for ground-to-aerial geolocalization. In *CVPR*, 2015.

[99] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

[100] Aurelien Lucchi, Yunpeng Li, Xavier Boix, Kevin Smith, and Pascal Fua. Are spatial and global constraints really necessary for segmentation? In *ICCV*, 2011.

[101] Zelun Luo, Yuliang Zou, Judy Hoffman, and Li F Fei-Fei. Label efficient learning of transferable representations acrosss domains and tasks. In *NeurIPS*, 2017.

[102] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *ECCV*, 2016.

[103] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *TPAMI*, 41(8):1979–1993, 2018.

[104] Saeid Motiian, Quinn Jones, Seyed Iranmanesh, and Gianfranco Doretto. Few-shot adversarial domain adaptation. In *NeurIPS*, 2017.

[105] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic

segmentation in the wild. In *CVPR*, 2014.

[106] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. In *ECCV*, 2016.

[107] Jonathan Munro and Dima Damen. Multi-modal domain adaptation for fine-grained action recognition. In *CVPR*, 2020.

[108] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.

[109] Boxiao Pan, Zhangjie Cao, Ehsan Adeli, and Juan Carlos Niebles. Adversarial cross-domain action recognition with co-attention. In *AAAI*, 2020.

[110] Pau Panareda Busto and Juergen Gall. Open set domain adaptation. In *ICCV*, 2017.

[111] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

[112] Xiaojiang Peng and Cordelia Schmid. Multi-region two-stream r-cnn for action detection. In *ECCV*, 2016.

[113] Senthil Purushwalkam and Abhinav Gupta. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. *arXiv preprint arXiv:2007.13916*, 2020.

[114] Anant Raj, Vinay Namboodiri, and Tinne Tuytelaars. Subspace alignment based domain adaptation for rcnn detector. In *BMVC*, 2015.

[115] Krishna Regmi and Ali Borji. Cross-view image synthesis using conditional gans. In *CVPR*, 2018.

[116] Zhongzheng Ren and Yong Jae Lee. Cross-domain self-supervised multi-task feature learning using synthetic imagery. In *CVPR*, 2018.

[117] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *ECCV*, 2016.

[118] Suman Saha, Gurkirt Singh, and Fabio Cuzzolin. Amtnet: Action-micro-tube regression by end-to-end trainable deep architecture. In *ICCV*, 2017.

[119] Suman Saha, Gurkirt Singh, Michael Sapienza, Philip H.S. Torr, and Fabio Cuzzolin. Deep learning for detecting multiple space-time action tubes in videos. In *BMVC*, 2016.

[120] Suman Saha, Gurkirt Singh, Michael Sapienza, Philip HS Torr, and Fabio Cuzzolin. Spatio-temporal human action localisation and instance segmentation in temporally untrimmed videos. *arXiv preprint arXiv:1707.07213*, 2017.

[121] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*, 2018.

[122] Kuniaki Saito, Shohei Yamamoto, Yoshitaka Ushiku, and Tatsuya Harada. Open set domain adaptation by backpropagation. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[123] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *NeurIPS*, 2016.

[124] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015.

[125] Ozan Sener, Hyun Oh Song, Ashutosh Saxena, and Silvio Savarese. Learning transferrable representations for unsupervised domain adaptation. In *NeurIPS*, 2016.

[126] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *ICRA*, 2018.

[127] Zheng Shou, Jonathan Chan, Alireza Zareian, Kazuyuki Miyazawa, and Shih-Fu Chang. Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In *CVPR*, 2017.

[128] Gunnar Sigurdsson, Abhinav Gupta, Cordelia Schmid, Ali Farhadi, and Karteek Alahari. Actor and observer: Joint modeling of first and third-person videos. In *CVPR*, 2018.

[129] Gunnar A Sigurdsson, Santosh Divvala, Ali Farhadi, and Abhinav Gupta. Asynchronous temporal fields for action recognition. In *CVPR*, 2017.

[130] Gunnar A Sigurdsson, Abhinav Gupta, Cordelia Schmid, Ali Farhadi, and Karteek Alahari. Charades-ego: A large-scale dataset of paired third and first person videos. *arXiv preprint arXiv:1804.09626*, 2018.

[131] Karan Sikka and Gaurav Sharma. Discriminatively trained latent ordinal model for video classification. *TPAMI*, 40(8):1829–1844, 2017.

[132] Karan Sikka and Gaurav Sharma. Discriminatively trained latent ordinal model for video classification. *TPAMI*, 40(8):1829–1844, 2018.

[133] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NeurIPS*, 2014.

[134] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NeurIPS*, 2014.

[135] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[136] Gurkirt Singh, Suman Saha, and Fabio Cuzzolin. Online real time multiple spatiotemporal action localisation and prediction on a single platform. In *ICCV*, 2017.

[137] Krishna Kumar Singh and Yong Jae Lee. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. In *ICCV*, 2017.

[138] Suriya Singh, Chetan Arora, and C. V. Jawahar. First person action recognition using deep learned descriptors. In *CVPR*, 2016.

[139] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *NeurIPS*, 2020.

[140] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[141] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[142] Kevin Tang, Vignesh Ramanathan, Li Fei-Fei, and Daphne Koller. Shifting weights: Adapting object detectors from image to video. In *NeurIPS*, 2012.

[143] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*, 2017.

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

[145] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 2017.

[146] Yi-Hsuan Tsai, Kihyuk Sohn, Samuel Schulter, and Manmohan Chandraker. Domain adaptation for structured output via discriminative representations. In *ICCV*, 2019.

[147] Michael Tschannen, Josip Djolonga, Marvin Ritter, Aravindh Mahendran, Neil Houlsby, Sylvain Gelly, and Mario Lucic. Self-supervised learning of video-induced visual invariances. *arXiv preprint arXiv:1912.02783*, 2019.

[148] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *ICCV*, 2015.

[149] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, 2017.

[150] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, 2017.

[151] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez. Dada: Depth-aware domain adaptation in semantic segmentation. In *ICCV*, 2019.

[152] Tuan-Hung Vu, Catherine Olsson, Ivan Laptev, Aude Oliva, and Josef Sivic. Predicting actions from static scenes. In *ECCV*, 2014.

[153] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *CVPR*, 2011.

[154] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *ICCV*,

2013.

[155] Jiangliu Wang, Jianbo Jiao, and Yun-Hui Liu. Self-supervised video representation learning by pace prediction. In *ECCV*, 2020.

[156] Junbo Wang, Wei Wang, Yan Huang, Liang Wang, and Tieniu Tan. M3: Multimodal memory modelling for video captioning. In *CVPR*, 2018.

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

[158] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks for action recognition in videos. *TPAMI*, 2018.

[159] Shiguang Wang and Jian Cheng. A faster pytorch implementation of r-c3d. *https://github.com/sunnyxiaohu/R-C3D.pytorch.git*, 2018.

[160] Tianlu Wang, Jieyu Zhao, Kai-Wei Chang, Mark Yatskar, and Vicente Ordonez. Adversarial removal of gender from deep image representations. In *ICCV*, 2019.

[161] Xiaolong Wang, Ali Farhadi, and Abhinav Gupta. Actions˜ transformations. In *CVPR*, 2016.

[162] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.

[163] Xiaolong Wang, Abhinav Shrivastava, and Abhinav Gupta. A-fast-rcnn: Hard positive generation via adversary for object detection. In *CVPR*, 2017.

[164] Yang Wang and Minh Hoai. Pulling actions out of context: Explicit separation for effective combination. In *CVPR*, 2018.

[165] Yulin Wang, Gao Huang, Shiji Song, Xuran Pan, Yitong Xia, and Cheng Wu. Regularizing deep networks with semantic data augmentation. *arXiv preprint arXiv:2007.10538*, 2020.

[166] Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Learning to track for spatio-temporal action localization. In *ICCV*, 2015.

[167] Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krahenbuhl, and Ross Girshick. Long-term feature banks for detailed video understanding. In *CVPR*, 2019.

[168] Tete Xiao, Xiaolong Wang, Alexei A Efros, and Trevor Darrell. What should not be contrastive in contrastive learning. *arXiv preprint arXiv:2008.05659*, 2020.

[169] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Unsupervised data augmentation for consistency training. In *NeurIPS*, 2020.

[170] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning for video understanding. In *ECCV*, 2018.

[171] Dejing Xu, Jun Xiao, Zhou Zhao, Jian Shao, Di Xie, and Yueting Zhuang. Self-supervised spatiotemporal learning via video clip order prediction. In *CVPR*, 2019.

[172] Dejing Xu, Zhou Zhao, Jun Xiao, Fei Wu, Hanwang Zhang, Xiangnan He, and Yueting Zhuang. Video question answering via gradually refined attention over appearance and motion. In *ACM MM*, 2017.

[173] Huijuan Xu, Abir Das, and Kate Saenko. R-C3D: Region convolutional 3d network for temporal activity detection. In *ICCV*, 2017.

[174] Jiaolong Xu, Sebastian Ramos, David Vázquez, and Antonio M López. Domain adaptation of deformable part-based models. *TPAMI*, 36(12):2367–2380, 2014.

[175] Takuma Yagi, Karttikeya Mangalam, Ryo Yonetani, and Yoichi Sato. Future person localization in first-person videos. In *CVPR*, 2018.

[176] Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *CVPR*, 2016.

[177] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019.

[178] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[179] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4l: Self-supervised semi-supervised learning. In *ICCV*, 2019.

[180] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating unwanted biases with adversarial learning. In *Proceedings of AAAI/ACM Conference on AI, Ethics, and Society*, 2018.

[181] Jing Zhang, Wanqing Li, and Philip Ogunbona. Joint geometrical and statistical alignment for visual domain adaptation. In *CVPR*, 2017.

[182] Qiming Zhang, Jing Zhang, Wei Liu, and Dacheng Tao. Category anchor-guided unsupervised domain adaptation for semantic segmentation. In *NeurIPS*, 2019.

[183] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[184] Yue Zhao, Yuanjun Xiong, and Dahua Lin. Recognize actions by disentangling components of dynamics. In *CVPR*, 2018.

[185] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. Temporal action detection with structured segment networks. In *ICCV*, 2017.

[186] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *ECCV*, 2018.

[187] B. Zhou, A. Khosla, Lapedriza. A., A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. *CVPR*, 2016.

[188] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *TPAMI*, 2017.

[189] Pengfei Zhu, Longyin Wen, Xiao Bian, Ling Haibin, and Qinghua Hu. Vision meets drones: A challenge. *arXiv preprint arXiv:1804.07437*, 2018.

[190] Xinge Zhu, Jiangmiao Pang, Ceyuan Yang, Jianping Shi, and Dahua Lin. Adapting object detectors via selective cross-domain alignment. In *CVPR*, 2019.

[191] Mohammadreza Zolfaghari, Gabriel L. Oliveira, Nima Sedaghat, and Thomas Brox.

Chained multi-stream networks exploiting pose, motion, and appearance for action classification and detection. In *ICCV*, 2017.