THE UNIVERSITY *of* EDINBURGH

Towards Efficient and Robust Action Recognition

Kiyoon Kim

Doctor of Philosophy

Institute of Perception, Action and Behaviour
School of Informatics
The University of Edinburgh

2023

Abstract

Human action recognition is a crucial computer vision task that plays an important role in various video understanding applications. Due to the increasing popularity of video content on the internet, it is vital to research better action recognition to replace excessive human labor to analyze videos. In this thesis, mainly three different problems related to efficiency and robustness that prohibit action recognition technology from being accessible to more people are identified, and solutions to each are proposed.

Firstly, the problem of capturing temporal information for video classification in 2D networks, without increasing their computational cost, is addressed. Existing approaches focus on modifying the architecture of 2D networks (*e.g.*, by including filters in the temporal dimension to turn them into 3D networks, or using optical flow), which increases computation cost. Instead, we propose a novel sampling strategy, where we re-order the channels of the input video, to capture short-term frame-to-frame changes. We observe that even without extensions, the proposed sampling strategy improves performance on multiple architectures (*e.g.*, TSN, TRN, TSM, and MVFNet) and datasets (CATER, Something-Something-V1 and V2), up to 24% over the baseline of using the standard video input. In addition, our sampling strategies do not require training from scratch and do not increase the computational cost of training and testing. Given the generality of the results and the flexibility of the approach, we hope this can be widely useful to the video understanding community.

Next, precisely naming the action depicted in a video can be a challenging and oftentimes ambiguous task. In contrast to object instances represented as nouns (*e.g.*, dog, cat, chair), in the case of actions, human annotators typically lack a consensus as to what constitutes a specific action (*e.g.*, jogging versus running). In practice, a given video can contain multiple valid positive annotations for the same action. As a result, video datasets often contain significant levels of label noise and overlap between the atomic action classes. In this work, we address the challenge of training multi-label action recognition models from only single positive training labels. We propose two approaches that are based on generating pseudo training examples sampled from similar instances within the train set. Unlike other approaches that use model-derived pseudo-labels, our pseudo-labels come from human annotations and are selected based on feature similarity. To validate our approaches, we create a new evaluation benchmark by manually annotating a subset of EPIC-Kitchens-100's validation set with multiple verb labels. We present results on this new test set along with additional results

on a new version of HMDB-51, called Confusing-HMDB-102, where we outperform existing methods in both cases. Data and code are publicly available.

Finally, despite recent advances in video action recognition achieving strong performance on existing benchmarks, these models often lack robustness when faced with natural distribution shifts between training and test data. We propose two novel evaluation methods to assess model resilience to such distribution disparity. One method uses two different datasets collected from different sources and uses one for training and validation, and the other for testing. More precisely, we created dataset splits of HMDB-51 or UCF-101 for training, and Kinetics-400 for testing, using the subset of the classes that are overlapping in both train and test datasets. The other proposed method extracts the feature mean of each class using the target evaluation dataset's training data (*i.e.*, class prototype), and estimates test video prediction as a cosine similarity score between each sample to the class prototypes of all classes. This procedure does not alter model weights using the target dataset and it does not require aligning overlapping classes of two different datasets, thus is a very efficient method to test the model robustness to distribution shifts, without prior knowledge of the target distribution. We address the robustness problem by adversarial augmentation training – generating augmented views of videos that are "hard" for the classification model by applying gradient ascent on the augmentation parameters – as well as "curriculum" scheduling the strength of the video augmentations. We experimentally demonstrate the superior performance of the proposed adversarial augmentation approach over baselines across three state-of-the-art action recognition models - TSM, Video Swin Transformer, and Uniformer. Curated datasets and code are publicly released. The presented work provides critical insight into model robustness to distribution shifts and presents effective techniques to enhance video action recognition performance in a real-world deployment.

Lay summary

We address the problem of automatic human action recognition, where an AI system sees a video, usually involving humans, and understands their behaviors. This process is typically done with machine learning, where the AI system gets trained by looking at a large amount of video data with their annotated labels, for example, “opening a door”, or “throwing something”. The system remembers the pattern of each action, and when it sees a new video that it is not trained with, it should still understand that it resembles the pattern of the other videos with the same action it has seen before.

More specifically, we address the issues in existing recognition technologies. Firstly, although current action recognition systems are improving their recognition accuracy, they are usually getting extremely expensive to use. In other words, they require expensive computers and a tremendous amount of time to train the system and use them with new data. To address this issue, we propose to use lightly designed systems, but the only modification is the looks of the videos so that the system can understand the motion. This way, it was possible to improve the accuracy of the inexpensive systems without incurring additional speed and computational costs.

As our second contribution, we identify that following the current video data labeling (annotating) convention introduces confusing meanings, which can lead to problems in training and using the systems. For example, an action of “peeling a carrot” can be explained differently as “cutting a carrot’s peel”, or “removing a peel of a carrot”. Furthermore, they are not necessarily synonyms, because “removing” can be used for various contexts such as “removing a tray from an oven”, which is equivalent to “taking”. We address this by allowing the action recognition system to predict multiple actions, but because it is generally not feasible to annotate in such a way, we still train the system with only a single action label per video. We annotated a large-scale action dataset with multiple labels to test that our system works much better than previous methods.

Finally, we show that using a video that looks drastically different from the ones that are used for training will fail the system. We show this by creating a dataset where the training and testing (that is used for prediction) data look very different, but they contain similar actions. We improve the accuracy in this scenario by changing the looks of the videos so that it is “hard” for the existing system, to improve for difficult (in terms of looks) videos.

Acknowledgements

I would like to thank my supervisor, Prof. Robert Fisher, for being inspiring as well as supportive. He is really kind-hearted and enjoyable to work with. Thanks to him, my PhD memories are filled with joy.

I thank Prof. Laura Sevilla for all the help in the early stages of the PhD.

I am grateful to Prof. Oisin Mac Aodha for dedicating his time to my projects as my second supervisor.

I would also like to express my sincere gratitude to my collaborators, Shreyank Gowda, Davide Moltisanti, Panagiotis Eustratiadis, and Antreas Antoniou for inspiring me as a researcher as well as for giving me motivation and emotional support.

I would like to thank the School of Informatics at The University of Edinburgh for providing the PhD scholarship, and thank EPSRC and the Alan Turing Institute for funding to collect the dataset.

Finally, I would like to thank my family. A huge part of my PhD was conducted online due to the COVID-19 pandemic, so I got to spend a significant time with them. Without their mental support, it would have been more difficult for me.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

The work published within this thesis has been published in the following peer-reviewed articles with attribution and contribution as follows:

> Kiyoon Kim, Shreyank N Gowda, Oisin Mac Aodha, and Laura Sevilla-Lara. Capturing Temporal Information in a Single Frame: Channel Sampling Strategies for Action Recognition. In *33rd British Machine Vision Conference*, 2022.

> Kiyoon Kim, Davide Moltisanti, Oisin Mac Aodha, and Laura Sevilla-Lara. An Action Is Worth Multiple Words: Handling Ambiguity in Action Recognition. In *33rd British Machine Vision Conference*, 2022.

The final work is not yet published at the time of writing.

> Kiyoon Kim, Shreyank N Gowda, Panagiotis Eustratiadis, Antreas Antoniou, and Robert B Fisher. Adversarial Augmentation Training Makes Action Recognition Model More Robust to Realistic Video Distribution Shift.

Contribution:

- Kiyoon: idea proposal, implementation, data collection and annotation, experiments, and writing
- Shreyank, Panagiotis, Antreas: help with building the core idea and experiments, and writing
- Davide: help with implementation, analyzing results, dataset, and writing
- Oisin, Laura: supervision, verification of ideas and help with writing
- Robert: supervision, verification of the ideas and theories, and help with writing

(*Kiyoon Kim)*

List of Acronyms

AN Assume Negative

AT Adversarial Training

BCE Binary Cross-Entropy

CNN Convolutional Neural Networks

EM Entropy Maximization

GrayST Grayscale Short-Term Stacking

GCE Global Context Encoder

GCN Graph Convolution Network

GST Grouped Spatial-Temporal Aggregation

RN Relation Network

RPN Region Proposal Network

LS Label Smoothing

P+S BCE Pseudo+Single label Binary Cross-Entropy

SRM Spatial Relation Module

SPML Single Positive Multi-label Learning

TC Time-Color Reordering

TDN Temporal Difference Network

TSN Temporal Segment Network

TRN Temporal Relation Network

TSM Temporal Shift Module

MTRN Multi-Scale Temporal Relation Network

MVFNet Multi-View Fusion Network

NLN Non-local Neural Network

N-LS Label Smoothing only for assumed negatives

WAN Weak Assume Negative

Table of Contents

Chapter 1

Introduction

1.1 Problem

Human action recognition is an important computer vision task that involves video category classification, and it serves as a fundamental technology for various video understanding applications including video searching, anomaly detection, and gesture recognition. Recent advances in mobile cameras that created a plethora of public video data (e.g. social media), and even private video such as CCTV footage and police body cameras, can require a tremendous amount of human labor to review. Furthermore, there are likely to be many robots with vision in the near future. However, the performance of action recognition is still not as reliable as that of many image-based recognition tasks. Moreover, video-based action recognition requires a significant amount of computational power to train and deploy. As a consequence, it is not yet common to see products that utilize video understanding technology, yet there is an increasing demand for this technology.

1.2 Claims

Recent advances in computer vision have shown remarkable progress. Today's image classification and detection performance is excellent, for instance. In contrast, video research progress is further behind. One may think image methods can directly be applied to a video pipeline, by processing each frame, but in many cases, this is simply not true due to the nature of video having an extra temporal dimension that complicates the problem.

The first hypothesis as to why video classification is harder than image classifica-

tion is that today's action models lack temporal understanding. There have been many attempts to enhance temporal understanding in action recognition models [1, 2], however, it is still unclear what method is the best for generic action recognition problems.

But how important is temporal understanding for action recognition? There are many papers that claim that simple 2D CNN-based models that ignore temporal information can outperform the 3D counterparts [3, 4]. 2D CNNs are widely popular due to their effectiveness, and thus it is crucial to study improving their temporal capability. In addition, the importance of temporal understanding and ways of modeling such capabilities varies according to the data, so it is vital to understand the problem being solved.

Another problem is the lack of suitable datasets. There are quite a few action recognition datasets, however, it is not enough to understand all of the different kinds of challenges in action recognition. A lot of the approaches work well only on a couple of datasets, and unfortunately they need to be tuned differently on others. Because of this difficulty, many machine learning approaches are being tested on image datasets (*e.g.*, Single Positive Multi-label Learning [5, 6], Progressive Self Label Correction [7]).

It is extremely challenging to create an action recognition dataset due to the many uncertainties in labeling. Video data is fundamentally a lot more complex than images. In image datasets, most of the objects of interest are center-framed, whereas it is hard to keep the framing consistent in videos due to moving subjects and cameras, resulting in inconsistent framing. Similarly, temporal boundaries of actions can be hard to define [8]. As a consequence, it is very expensive to create a video dataset, and furthermore, they are noisy.

Much research on designing video-based action recognition models has been done [3, 9, 10, 1, 11, 2, 12]. Instead of designing another network architecture for action recognition, we claim that sampling videos in a way that encodes multiple frames in a single frame representation will significantly improve action recognition performance. In this thesis. we propose an extremely general video channel sampling strategy that enhances the input representation, especially for efficient 2D networks. We also claim that there exists a video-specific annotation ambiguity problem, and thus treating action recognition as a multi-label classification task learned from single positive labels is a better approach. We show how image-based methods [5, 6] fail in this scenario, and suggest a novel solution using pseudo labels. Finally, we claim that there exists a severe distribution disparity between collected training videos and real-life videos, making it extremely difficult to create video-based applications. This distribution shift problem

can be addressed with a better augmentation strategy, and we claim that a novel adversarial augmentation strategy makes the models more robust under such disparities. Furthermore, we create a dataset to demonstrate the significant performance drop with distribution shifts and suggest adversarial augmentation training as a solution to compensate for this, with two different non-training-based measures to assess the benefits of the adversarial augmentation methods.

To summarize, below are the factors that make action recognition difficult:

- Lack of temporal understanding in action models
- Lack of datasets
- Ambiguous, or single action annotations
- Poor problem understanding although video data is fundamentally more complex than images

Thus, we claim that the methods given below will advance action recognition:

- Sampling videos such that the models have a bigger temporal receptive field
- Creating datasets for multiple verb learning
- Multi-label learning with single positive labels to enhance action understanding
- Creating datasets containing realistic distribution shifts
- A novel augmentation pipeline that will tackle the distribution shift issues that is especially severe in video.

1.3 Contributions and Thesis Outline

Our main contributions are as follows. Firstly, we improve efficient 2D action recognition performance by a significant margin without adding computational cost, by sampling video channels in novel ways. We show this by evaluating on many popular datasets and architectures, showing a meaningful performance boost compared to normal RGB sampling. See Chapter 3.

Next, we identify complexity in labeling action recognition datasets, due to ambiguity in defining actions. This leads to issues in both training and testing the action

recognition models. As a solution, we propose two novel ways to train multi-label action classification given only single positive labels. The approaches try to disambiguate the label space by analyzing similarities in training data and generating pseudo-labels from similar instances that have different labels. We further annotate a large-scale dataset in a multi-label fashion to test the multi-label performance, as well as introduce a dataset with synthetically added label noise. We show that our methods outperform existing single positive multi-label learning baselines. See Chapter 4.

As our final contribution, we empirically demonstrate that the proposed adversarial augmentation and curriculum adversarial training frameworks enhance robustness to realistic distribution shifts between the training and test datasets, through extensive experiments across multiple state-of-the-art architectures. We propose two novel evaluation protocols to assess model resilience to distribution disparity using naturally sourced datasets, as opposed to solely artificially corrupted data. For our first approach, we construct new cross-dataset benchmarks by identifying overlapping classes between HMDB-51, UCF-101, and Kinetics-400. Models are trained on either HMDB or UCF, and evaluated on the Kinetics data. We then introduce a similarity-based evaluation approach that estimates predictions using cosine similarity between embedded training and test features, without requiring class alignment. Experiments reveal substantial performance degradation on our cross-dataset benchmarks, quantitatively demonstrating the challenge posed by real-world distribution shift. We have publicly released the constructed subsets of HMDB-51, UCF-101, and Kinetics-400 to enable further research on this important problem.

Finally, we summarize the work and discuss successful and failed approaches, suggesting future directions in Chapter 6.

To summarize,

- We improve 2D action recognition models by resampling frames. This does not add any additional computational cost.
- We propose novel ways to train the single positive multi-label learning task, that analyzes confusing label space that can overlap in meaning based on the context of the videos.
- We annotated a large portion of the EPIC-Kitchens-100 dataset in a multi-label fashion. This data subset is public at URL: `https://github.com/kiyoon/verb_ambiguity`

- We propose a novel adversarial augmentation training recipe that improves generalization in video action recognition.
- We propose new datasets to evaluate video action recognition performance with a huge distribution shift. These datasets will be made public at URL: `https://github.com/kiyoon/video-adversarial-augmentation`
- We propose a novel evaluation method given a test dataset whose categories do not align with the training dataset.

Chapter 2

Background

2.1 Action Recognition Overview

In this section, common and popular action recognition approaches using videos as input are addressed.

2.1.1 Action Understanding Tasks

There are three main tasks in action understanding. Firstly, the *action recognition (classification)* task refers to predicting what action is present usually in short video segments. Secondly, the *temporal action detection (localization)* task requires answering when the action is present [13, 14, 15, 16]. Lastly, predicting when and where the action is happening is called the *spatial temporal action detection* task [17, 18, 19].

Additionally, each task can be done online or offline, the former uses real-time data and the prediction has to be made without the future data, and the latter uses stored data, and the predictions can be in labels or natural language.

In the later sections, we will focus on the offline action recognition task, where each video has one label and trimmed[1] videos are used.

2.1.2 Video Representations

The most common representation of video frames for deep learning is 8-bit raw RGB, where each pixel has three color channels – red, green, and blue, each represented by 8 bits (*i.e.*, 256 levels). This representation does not encode any temporal information in a frame. The raw format gives the deep learning networks the maximum potential

[1] short, usually within 10-second long video segments that contain one action.

for analysis, however, it is extremely memory inefficient and thus limits the model's capacity to accept higher resolution or frame rates. In practice, most of the images and videos are stored in a compressed manner. For instance, the chroma subsampling method samples chrominance (*i.e.*, color) information less frequently than luminance (*i.e.*, brightness) information, due to the fact that human eyes are less sensitive to color than brightness changes. This approach is rarely used in deep learning because of the complex encoding (*i.e.*, inconsistent resolution throughout the channels) and lack of pre-trained networks shared publicly to begin with. In order to load compressed videos for deep networks that accept raw videos, decoding (*i.e.*, decompressing) is required which can potentially be a bottleneck in the training process if done inefficiently (*e.g.*, using an inefficient processor such as CPU rather than GPU). On the other hand, [20, 21] used compressed images or videos as inputs for efficiency in storage, memory, and speed.

Another approach is to compute differential images by simply subtracting each pair of consecutive frames [3]. This gives the network a more explicit representation of temporal high-frequency changes. Unfortunately, this is not robust to spatial shifting, so as the camera pans, all pixel values change, which defeats the purpose of identifying the moving objects by subtracting the frames.

Optical flow is another common representation in action recognition [11, 3, 22]. Given two consecutive video frames, the optical flow is the direction and magnitude of the motion at each pixel. Using optical flow as a separate stream [11] generally improves performance, although it is often computationally expensive. Recent work has used the motion vectors from compressed video (*e.g.*, H.264) to avoid the computational cost of optical flow [20, 23]. Given any of these representations, one of the standard approaches is to simply feed them as input to a 2D convolutional network, similar to those used in the image domain, where frames are processed independently, and the predictions are aggregated in the end. This family of approaches is called 2D, or frame-based networks. Despite their simplicity, they often work well on many datasets, and are fast to train and test.

Dynamic images [24] are a compact video representation that encodes spatial and the entire motion information in a single image, so any image classification network can perform action recognition using the generated images. In contrast, our methods in Chapter 3 encode short-term temporal information per frame, resulting in a space-time sequential representation (*i.e.*, they do not compress an entire video into a single frame) and can thus use off-the-shelf 2D action recognition networks to better capture

temporal information compared to only using a single image classification model as in [24].

Lastly, videos can be represented as graphs which are more straightforward high-level representations than others. For example, [25] used object detection networks to detect objects in videos to be represented as nodes, and connected the nodes (*i.e.*, graph edges) of visually similar and spatial-temporally overlapping objects. This is followed by graph convolution networks (GCNs) to learn to predict action categories from the representation.

2.2 Datasets

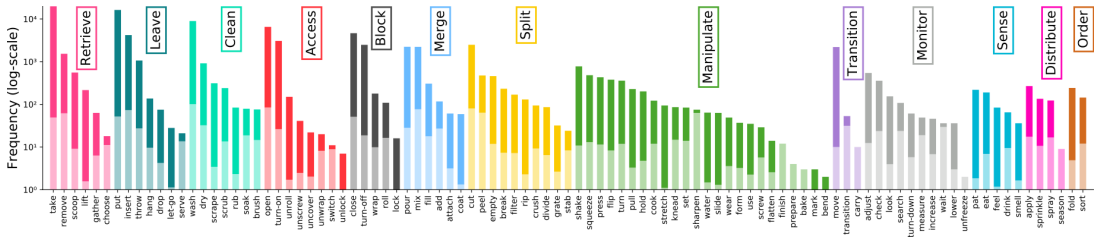
Kinetics [26] is one of the most popular large-scale action recognition datasets gathered from public YouTube videos. The latest release (Kinetics-700) consists of approximately 650,000 video clips of 700 action classes, and each video clip is around 10 seconds long. This dataset has often been used when training from scratch, producing good pre-trained models for fine-tuning for different datasets. Kinetics-400 has been also widely used due to many existing benchmarks, and there are over 300,000 videos in 400 classes.

HMDB-51 [27] is relatively a small-scale action dataset that is popular due to its small size while containing key actions such as facial actions, body movements, and object interaction. It is composed of 7,000 video clips in 51 categories.

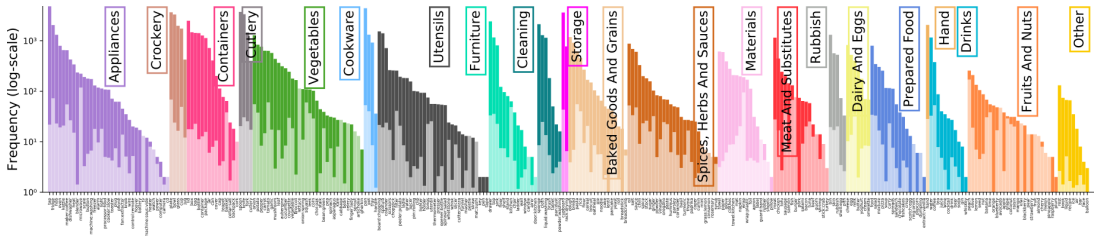
UCF-101 [28] dataset consists of 13,320 videos in 101 action classes collected from YouTube.

Something-Something-V2 [29] is an interesting large-scale dataset in which the categories are labeled without mentioning the object present in the scene (e.g., Putting [something] onto [something]) so that the model is forced to focus on the motion features of the data over the appearance features. This is an important dataset for many action recognition benchmarks because of the previous reports [3, 30, 4] claiming that other datasets (especially HMDB, UCF, and Kinetics) have an extreme background and object bias that enables even image classification networks (*i.e.*, 2D action models) to classify action categories, sometimes even better than 3D counterparts. It consists of 220,847 videos distributed in 174 classes.

EPIC-Kitchens-100 [31] is the second largest of first-person action datasets, containing roughly 90,000 action segments in 97 verb classes and 300 noun classes. There are 45 actors in 45 different kitchens so the evaluation can be done using unseen



(a) Verb categories



(b) Noun categories

Figure 2.1: Verb and noun categories (x-axis) and sub-categories (differentiated by color) that the EPIC-Kitchens-100 dataset has. The dataset is collected in an unscripted way, and naturally there exists an extreme long-tailed distribution. Figure taken from [32].

kitchens as well as seen kitchens. The dataset is non-scripted, meaning the actors and actresses are acting as naturally as possible and the labels are annotated in post, simulating the real-world scenarios but giving extremely biased data (e.g., "put" and "take" actions occur significantly more often than "throw"). See Figure 2.1 for the list of verb and noun categories and their distribution.

Ego4D [33] is another large-scale egocentric video dataset that contains over 3,000 hours of videos in 9 different countries collected by 931 participants. In contrast to EPIC-Kitchens having only cooking-related actions in kitchens, Ego4D consists of various scenarios as shown in Figure 2.2.

2.3 Literature Review

2.3.1 3D Networks

A snippet refers to one or a few frames of images. In this section, a deep representation of videos at a snippet level using 3D CNNs to recognize actions will be introduced.

3D CNNs were explored several times to extract spatial-temporal features [34, 35]. Instead of using 2D spatial filters for CNNs, they use 3D spatial-temporal filters to

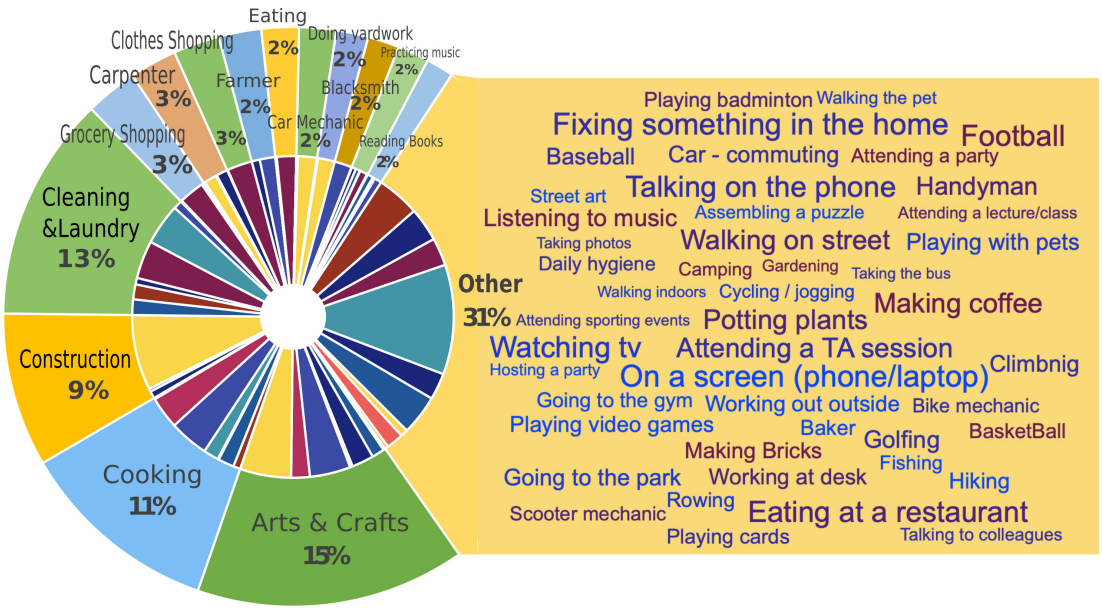


Figure 2.2: Various scenarios described in the Ego4D dataset. Figure taken from [33].

learn both spatial and temporal features simultaneously. They are trained to extract better feature representations. However, it is hard to train these models due to too many parameters. In order to make use of pre-trained 2D CNNs models for image classification, I3D networks [22] have been proposed to inflate 2D CNNs by copying image frames to make stationary videos, and also copying 2D filters to make 3D filters. This network is commonly used as a baseline or a backbone model for action recognition tasks.

Due to having an abundant number of parameters in the 3D CNNs, R(2+1)D (*i.e.*, P3D) [36] and Grouped Spatial-Temporal Aggregation (GST) [37] are proposed to decouple the spatial and temporal capturing paths. For the R(2+1)D, the 3D convolutions are completely decomposed to 2D spatial and 1D temporal convolutions as shown in Figure 2.3, while GST tried to decompose into 2D spatial and 3D spatial-temporal paths in parallel, as described in Figure 2.4.

SlowFast model [38] has slow and fast pathways in 3D CNNs that run at different frame rates to capture both appearance-oriented and motion-oriented features. As a result, the architecture captures temporal information at two different temporal scales (i.e., short-range and long-range), which means the features have richer temporal information.

Lastly, SmallBigNet [39] tackled the limitation of temporal convolution having a small spatial receptive field. For example, when an object in a frame moves far away, the temporal convolution does not track the object but it just focuses on the area

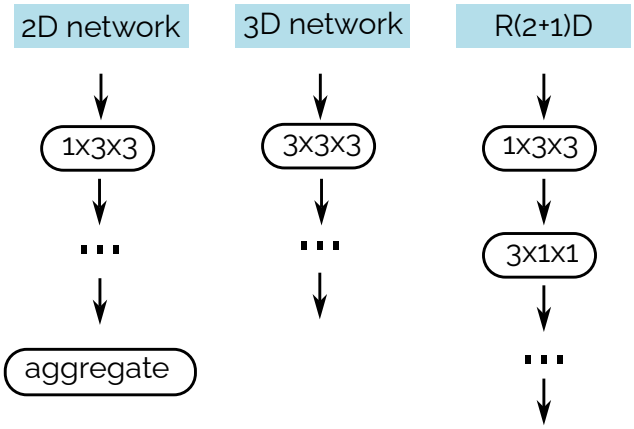


Figure 2.3: R(2+1)D network in comparison to traditional 2D and 3D networks. 2D networks process per image using spatial convolutions only, and at the end they aggregate the features or predictions. TSN, TRN, TSM, and MVFNet fall into this category. 3D networks, such as C3D, I3D, and SlowFast, operate on video, using spatial-temporal filters. Finally, the R(2+1)D network decouples the 3D convolution into spatial (2D) and temporal (1D) convolutions.

where the object is no longer present. As a solution, a parallel branch called Big View provides contextual information to the Small View by convolving over the 3D-max-pooled feature.

Lastly, Temporally-Adaptive Convolutions [40] calibrates spatial convolutional kernels based on the local and global context, instead of operating the same spatial convolution for every frame. Since the calibration is done on the kernel and not on the feature maps, it efficiently models the temporal information.

2.3.2 Relation Networks and Attention

The Relation Networks (RNs) were first suggested to be used for visual question answering (i.e., Visual QA) [41]. RNs are designed to solve tasks that require relational reasoning between a pair of features (*e.g.*, pixels, objects). An example of a relational question (*i.e.*, question that requires relational reasoning) can be found in Figure 2.5. The RN is applied to the final CNN feature maps, pairing each object representation and inputting the pairs as a batch to two multi-layer perceptrons (MLPs). For example, if the final CNN feature maps have the size of $H \times W \times f$ where H and W are the spatial coordinates and f is the number of filters, then the number of objects is $H \times W$ of $1 \times f$ size, and there will be $(H \times W)^2$ object pairs. Furthermore, the paper also claims

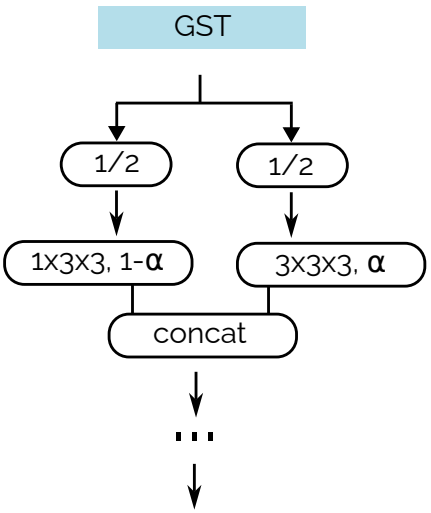


Figure 2.4: GST network consists of GST modules that involve splitting the feature maps and applying both spatial and 3D convolutions.

that the RN can be applied not only to the CNN feature maps but also to any form of object representations such as object bounding boxes and physical state matrices.

A similar approach was used for video question answering (*i.e.*, Video QA) [42]. In order to capture the spatial-temporal relationships, a Spatial Relation Module (SRM) as [41], Global Context Encoder LSTM (GCE), and a Temporal Relation Module are used. The GCE module takes the C3D motion feature and generates long-term temporal information. The SRM and GCE features are then concatenated and passed to the Temporal Relation Network (TRN) [9], which will be explained in the next paragraph. The generated spatial-temporal feature is then used in the answer decoder module.

There have been several attempts at using RNs for action recognition. Non-local Neural Networks (NLNs) [43] is a neural network module that can be inserted into any 2D, 3D, or N-D CNN network. Since it outputs the same size as the input, it can be inserted into any pre-trained networks without changing their shape and weights. A position x_i's response is computed by the weighted average of the features of all positions x_j, learning the spatial-temporal relation for video inputs. Unlike TRN which uses the last convolutional feature, NLNs can be inserted a number of times in between convolutional layers in a network, however this can introduce a significant computational overhead as described in Figure 2.6. There are four variants of NLNs: Gaussian, embedded Gaussian, dot product, and concatenation. The authors reported that they all perform similarly and embedded Gaussian is used for easy visualization but not for performance reasons. Using embedded Gaussian can be seen as a self-attention

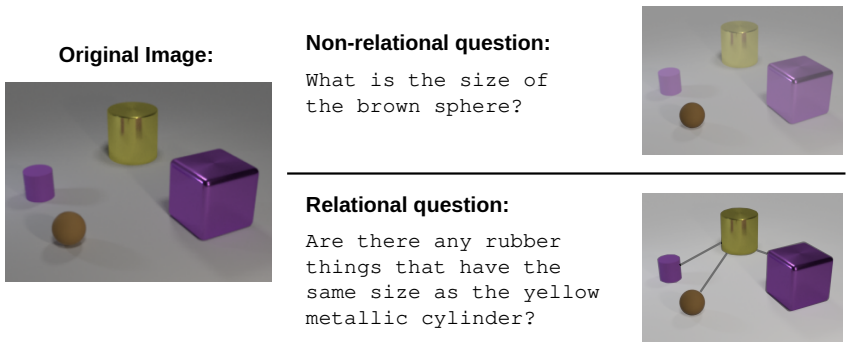


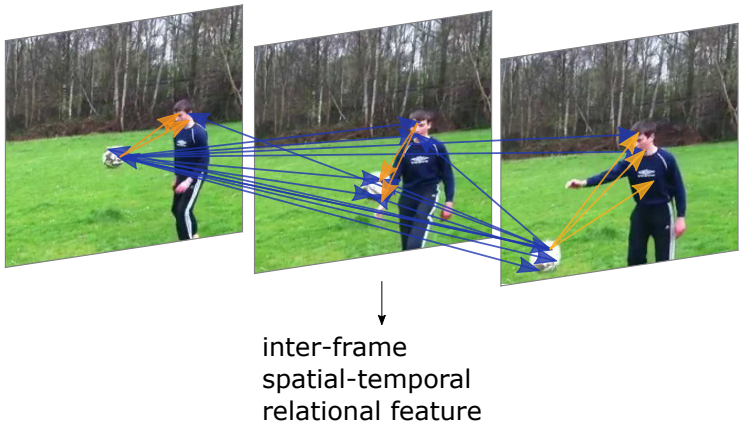
Figure 2.5: An example of a relational question. A typical CNN network may not be able to answer such questions, and adding relation networks enables this reasoning. Figure taken from [41].

module [44] being used to learn space-time relationships for computer vision.

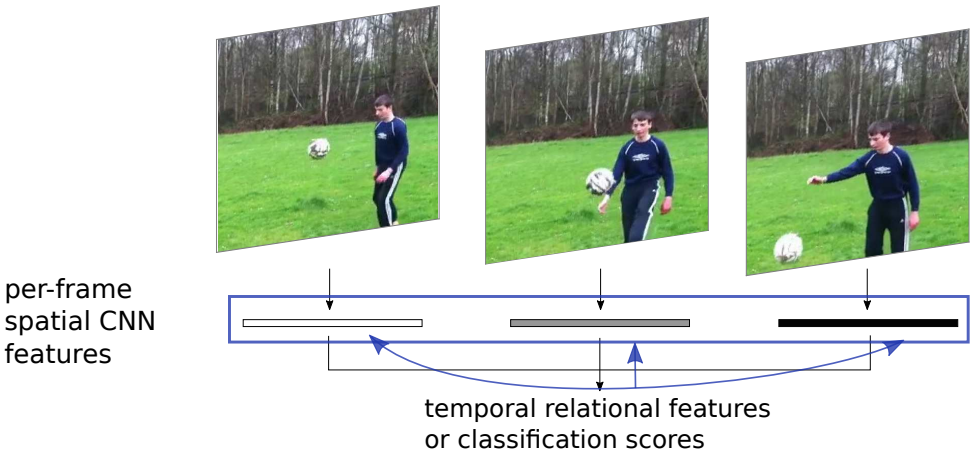
Due to the popularity of non-local operations, there have been several studies improving or modifying the NLNs. [45] claims that in many scenarios attention can be dispersed. For example, in boxing, the network should attend to two distant boxers. The bilinear transform supports a wide range of transforms which allows the network to capture local and global attention.

Graph Convolution Networks (GCNs) [25] represent a video as space-time graphs: appearance-similarity relations and spatial-temporal relations. Video inputs are passed to an I3D network [22] and to a Region Proposal Network (RPN) to generate the object bounding boxes and their feature representation. Then, RoIAlign is used to generate $7 \times 7 \times d$ output features which are then max-pooled to $1 \times 1 \times d$. The extracted features are used to construct the similarity graph using the self-attention module [44] and the bounding boxes are used to generate the spatial-temporal graph where objects in one frame are connected to spatially-close objects in the next frame (and vice versa for bidirectional information). The graphs are convolved and features are generated, which are then concatenated to the I3D feature to perform classification. This approach performs 2.7% better than Non-local Networks on the Charades dataset. However, the limitation of the graph convolution is that it works for structured data such as bounding boxes, whereas Non-local blocks or relation networks can be applied to unstructured data such as CNN features. In other words, between NLNs and GCNs, there is a flexibility and performance trade-off. Additionally, the spatial-temporal relation graph does not have any learnable parameters and is constructed using a naive measure.

Instead of adding relational reasoning in CNN architectures or graph representations, a more recent trend is to use transformer-based architectures, getting rid of



(a) Non-local Neural Networks



(b) Temporal Relation Networks

Figure 2.6: Comparison between the Non-local Neural Networks (a) and the Temporal Relation Networks (b). The relational reasoning happens on a per-pixel level for (a), and only on final features for (b). This is why (b) is much more efficient than (a). Figure inspired and modified from [43].

convolutional layers [46, 2, 12]. This can add a significant computational overhead especially when the data gets larger, but it is known to perform better given large-scale training data.

2.3.3 2D Architectures for Temporal Modeling

While 3D action recognition networks can perform better than their 2D counterparts, they have larger computational requirements for training and testing. This in turn,

makes them challenging to deploy in resource-constrained settings, *e.g.*, mobile and online scenarios. As a result, 2D networks for action recognition tasks are still common practice for certain application domains.

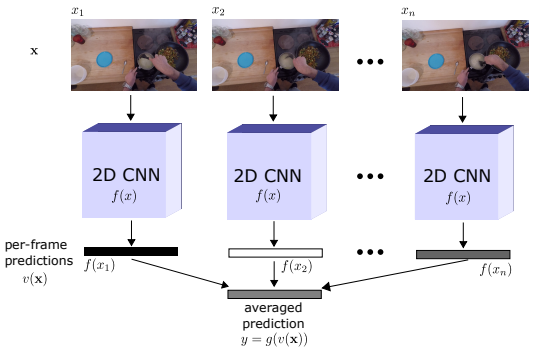
Temporal Segment Networks (TSN) [3] is an early and widely used 2D method. TSN simply extracts predictions for each frame sampled from the input video using a 2D backbone network and then averages these predictions for the final output. Frames are sparsely sampled from longer videos so that the model can reason over long time spans. With an added optical flow stream, TSN achieves competitive performance on action datasets, even defeating some 3D networks at the time of release. This high performance is more obvious in datasets where there is a strong correlation between actions and static objects and scenes [47]. In contrast, TSN performs worse on datasets that require explicit temporal reasoning [29, 30].

Temporal Relation Networks (TRN) [9] aim to capture relational information across frames. For this, it also uses a 2D backbone to extract features from sparsely sampled frames, but then aggregates the frame-level features using a relational module [48], as described in Figure 2.7. The authors proposed a multi-scale relational module that learns 2-frame, 3-frame, and up to T-frame relationships (TRN-multiscale, or MTRN in short). This slightly improves performance over the single-scale TRN. This is an improvement over TSN, which is agnostic to the temporal ordering of the input frames. As a result, TRN improves performance on temporal datasets, *e.g.*, [29]. However, their model is not very memory-efficient, as it requires $T-1$ relational modules (from 2 to T frame) using fully-connected layers, making it challenging to process many frames.

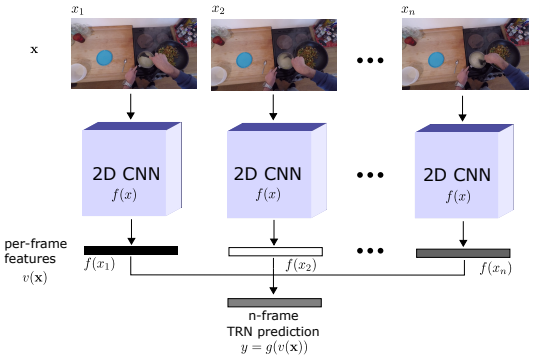
The Temporal Shift Module (TSM) [10] follows the same strategy as the TSN but modifies the ResNet [49] backbone so that the network can partially access information from one past and one future frame (*i.e.*, temporal shift). In order to maintain the spatial feature learning capability, the temporal shift happens inside a residual branch, so the backbone choice is limited to those with residual connections. TSM achieved improved performance on the Something-Something-V2 dataset [29], showing that 2D models are not obsolete and are still competent in action recognition.

Multi-View Fusion Network (MVFNet) [50] adds channel-wise convolutions to model height-width, height-time, and width-time views instead of treating height-width-time video frames as a space-time signal. This approach performs better than TSM while still remaining computationally efficient.

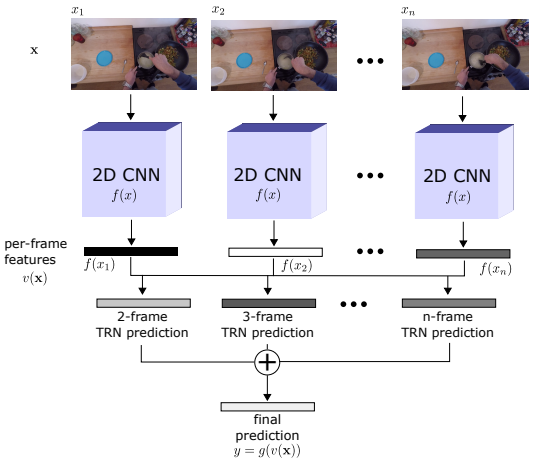
Finally, Temporal Difference Networks (TDN) [51] explicitly compute motion in-



(a) TSN



(b) TRN



(c) TRN multi-scale

Figure 2.7: Comparison between the Temporal Segment Networks (a) which is the simplest of the 2D networks, Temporal Relation Networks (b) and its multi-scale variant (c).

formation by sampling five times as many input frames (*e.g.*, 8-frame TDN uses 40 frames) and computes RGB differences for capturing short-term information and uses multi-scale attention for capturing long-term temporal information. TDN presented

a new state-of-the-art on the Something-Something datasets, but at the cost of using significantly more input frames.

In Chapter 3, we take advantage of the efficiency and simplicity of 2D video models by proposing image channel sampling strategies that improve a model's ability to capture temporal information. This increases accuracy, all without introducing any additional computation during training or testing.

2.3.4 Modeling Longer Timespans

Conventional 3D CNNs are not able to capture long-term information. There is much research tackling this problem by modeling longer timespan.

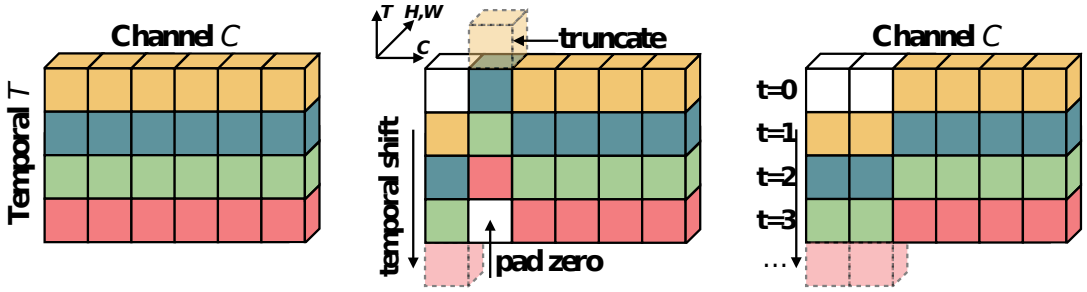
Long-Term Feature Banks [16] take short-term features generated by 3D CNNs and long-term features which is a collection of short-term features at all time stamps, and computes updated versions of short-term features for better classification. The feature bank operator is flexible, and the authors have demonstrated using attention and pooling operations.

Instead of stacking a large number of local operations, [52] proposed a multiple temporal aggregation (MTA) module that divides the feature's channel dimension by 4 and applies sub-convolutions successively to each fragment and adding the previous fragment's convolution output so that the message from distant frames will be preserved. Along with that, they also proposed a motion excitation (ME) module that calculated learnable motion information using two consecutive frames. It is similar to optical flow, but it is learnable and calculates feature-level motion instead of using raw pixels.

MeMViT [53] utilizes memory to model features at arbitrary layers, as opposed to the Long-Term Feature Banks which only model the output feature. As a result, their transformer backbone can refer to the prior context efficiently without much increase in cost.

2.3.5 Efficient Training

In order to reduce the trade-off between the fast training convergence time in low-resolution and more accurate predictions in high-resolution, Multigrid training [54] attempts to train in small input sizes at the beginning of training and in large input sizes at the later steps. It is reported that it is not only 4.5x faster, but also increases the accuracy by 0.8% on the Kinetics-400 dataset.



(a) The original tensor without shift.

(b) Offline temporal shift (bi-direction).

(c) Online temporal shift (uni-direction).

Figure 2.8: TSM efficiently adds temporal reasoning abilities to the existing 2D networks by shifting the feature maps temporally. This introduces almost zero computational overhead while giving a significant performance boost on some of the datasets, especially those that require extensive temporal reasoning. Figure taken from [10].

Another study [55] suggested eliminating redundancy in the 3D CNN filter's time domain by converting the filters into the frequency domain and discarding rows with small frequency values. Intuitively, this can be seen as an edge detection process in visualized features.

X3D [56] is an expanding architecture that starts from a small 2D model and expands one single axis such as temporal duration, spatial resolution, width, depth, etc. at a time until it finds the best computation and accuracy trade-off by training and validating the model.

[57] proposed a regularization method for action recognition by multiplying noise by smoothed (*i.e.*, low-frequency components) features. Their analysis showed that multiplying low-frequency parts of the feature by noise degrades accuracy less than multiplying the high-frequency parts. This suggests that high-frequency features are more important in action recognition, and note that this observation matches that from [55]. By simply multiplying the low-frequency components of the feature by Gaussian noise, they managed to generalize the network better.

For efficiency, it has been also popular to use 2D backbone-based networks like TSN. The problem with the 2D backbone is that it usually lacks reasoning about temporal information, making it vulnerable to tasks that require temporal understanding (*e.g.*, distinguishing between opening and closing a door). The Temporal Shift Module (TSM) [10] brought a significant accuracy improvement over TSN without introducing computational complexity, by shifting the CNN features of the backbones temporally so that the feature maps contain multi-time information. See Figure 2.8.

Calculating accurate optical flow is computationally expensive, prohibiting its use in many online scenarios. There has been a lot of work addressing this issue. Representation Flow [58] adds optical flow layers in CNN architectures which will learn flow-like features using the CNN feature maps. The layer is inspired by optical flow algorithms but is not limited to RGB frames as inputs. Therefore, the layer can even compute "flow of flow" representations. Due to simpler computation yet being more flexible, they managed to maintain the performance of two-stream models efficiently. In addition, Motion Augmented RGB Stream (MARS) [59] networks distil information from the flow stream that has motion information to the RGB stream. As a result, they made standard 3D CNN models operating on RGB frames mimic the motion stream, which improved the accuracy of the single-stream networks at test time.

A recent trend is data-efficient training with masked autoencoders (MAE), which trains with reconstruction loss given a sparsely-masked input, that enables self-supervision without labels. The authors of VideoMAE [60] demonstrated that using an extremely high masking ratio (around 90%), even small datasets performed well if they were of good quality. The high masking ratio is possible due to the fact that video data has high temporal consistency. The VideoMAE can outperform all baselines without requiring any additional data for pre-training, unlike other methods, in many datasets and tasks. Hiera [61] optimizes the vision transformer architecture by removing unnecessary components without losing accuracy. For example, they showed that changing the max pooling kernel stride so they do not overlap between separate masks improves speed and performance. Masked Video Distilation [62] uses a reconstruction of high-level features instead of low-level RGB pixels. Furthermore, the model uses a knowledge distillation method with separate spatial and temporal teachers. It achieved state-of-the-art performance on Something-Something-V2. However, the efficient MAEs still use expensive transformer models that are reported to require almost 200 times higher GFLOPs, and in Chapter 3 we deal with an extremely efficient 2D models.

2.3.6 Noisy Labels and Videos

Video datasets are generally noisy. [63] proposed a way to learn from noisy web videos using a Q-learning method. It tries to learn a data labeling policy from a small training dataset, and automatically labels new data using the policy. However, the concept of noisy labels in this research is coming from web data, and is different from the ambiguous annotation problem and multi-label learning we discuss in Chapter 4. [64]

introduced a video dataset where noise in the labels comes from inaccurate parsing of web tags. They showed that using multiple labels per video, pre-training the network with their dataset and fine-tuning on the UCF-101 improves performance compared to using only single labels. However, in their dataset, they already have multiple labels per video although they are noisy, but we tackle the problem of only having a single label to train multi-class classification. [65] explored pre-training methods using large-scale video datasets with label noise (*i.e.*, missing or incorrect labels) and temporal segmentation noise. Their study is based on the fact that pre-training with a large-scale action dataset improves action recognition performance even with the presence of label noise and low-quality videos. [8] studied temporal ambiguity in action labeling showing that the perception of temporal bounds is highly subjective, which in turns leads to recognition robustness issues. They proposed a new strategical annotation strategy that involves dividing the actions into a pre-actional phase and an actional phase, achieving higher consistency in temporal labeling. Multiple instance learning has also been used to tackle noisy video datasets, where noise comes from less reliable annotations or sparse labeling [66, 67]. The idea is that it is more likely to have false positive annotations than false negatives. Therefore, they group multiple positive-labeled examples together, which will smooth out the noise, while keeping the negative examples as is (*i.e.*, group of 1) for training.

However, none of these methods specifically address the noise generated by the ambiguous nature of verbs. This type of noise exhibits different distributions compared to noise stemming from the other problems discussed above. This is because the meaning of a verb varies depending on the action context, and as a consequence there is only a partial overlap across different labels, *i.e.*, labels are not exactly interchangeable across all videos.

2.3.7 Corruption Robustness Analysis

[68] provided benchmarks for measuring a neural network's robustness to corruptions and perturbations, by evaluating with 15 algorithmically-generated corruptions (*e.g.*, noise, blur, pixelate, compression artifacts). [69] extended this to video classification tasks and video corruptions (*e.g.*, video compression artifacts, frame rate conversion, bit error, packet loss). [70] reported a large-scale robustness analysis of deep action recognition models again using pre-defined perturbations.

However, these approaches were evaluated using simulated data, while we propose to use real data for testing. Evaluating robustness with augmented data prohibits the same augmentations from being used for training. Chapter 5 focuses on a more realistic scenario where a known set of data augmentation strategies is used for training and evaluating is done with unprocessed real data.

2.4 Conclusion

In conclusion, we saw that action recognition networks with temporal reasoning, datasets that require temporal reasoning, spatio-temporal relational reasoning models, and using different representations (*e.g.*, optical flow) as separate streams in the networks have been investigated. We did not see efficient sampling methods for efficient models, action recognition as a single positive multi-label learning problem, and adversarial augmentation training with realistic distribution shifts, which we will explore in chapters 3 to 5.

Chapter 3

Channel Sampling Strategies for Action Recognition

3.1 Introduction

Understanding temporal information is crucial in order to understand video. While 2D convolutional filters are usually the standard approach to capture spatial information, there is a wider variety of methods for capturing temporal information. These include, for example, using the transformer architecture [71], 3D networks [72], computing optical flow and feeding it to a 2D convolutional network [11], using relational networks [9], or using Recurrent Neural Networks (RNNs) [73], among others. All of these methods require changes in the underlying network architectures, additional computational cost compared to simple 2D networks, as well as the time-consuming process of pre-training from scratch.

In this chapter, we propose two simple and novel channel sampling strategies that improve the ability of a given 2D network to capture temporal information without changing the architecture. In particular, we re-order the channels of the input video in two ways: in the first, we re-order the channels of the video so that each frame is composed of three channels: one that belongs to the frame before, one that belongs to the current frame, and one that belongs the frame after. These three channels are concatenated in the channel dimension as if it were a single frame. This procedure incorporates temporal information from neighboring frames, while keeping the dimensions of the input frame.

In the second strategy, we compute a grayscale version of three neighboring frames at a time, and again concatenate them in the channel dimension, as if they were a single

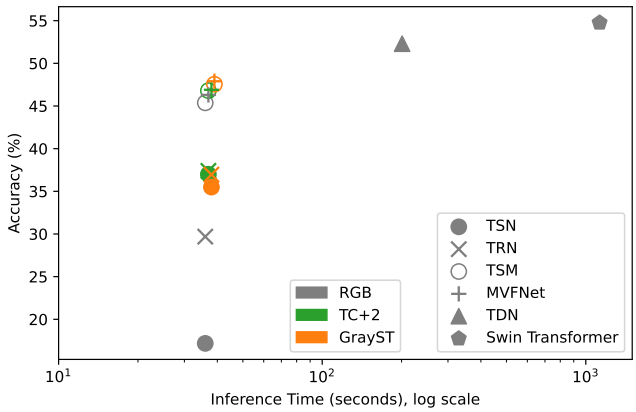


Figure 3.1: We show that it is possible to significantly increase the performance of lightweight action recognition networks on the challenging Something-Something-V1 dataset by simply adapting how the individual image channels are sampled. Our approaches, *TC+2* (in green) and *GrayST* (in orange), improve accuracy across several networks without increasing inference time. This enables us to narrow the gap between these efficient baselines and much more computationally demanding methods such as TDN [51] and Video Swin Transformers [2]. Note that we use a log scaling on the horizontal axis.

RGB image.

We test these extremely simple re-ordering strategies on five widely used 2D networks (TSN [3], TRN/MTRN [9], TSM [10], and MVFNet [50]). We observe that without any additional engineering these re-ordering strategies improve results up to 24% compared to the standard RGB channel ordering, across multiple challenging video datasets and different networks. We also observe that the improvement is particularly large in the datasets where temporal information is more important, in particular on CATER [74] and Something-Something [29]. Figure 3.1 illustrates the performance improvement which does not add additional significant computational overhead.

Our main contributions are:

- We improve the performance of simple 2D CNN-based action recognition models while incurring no additional computational complexity and with no modification to the underlying models. Our solutions can be used with most existing network architectures.

- Through extensive experimental evaluation on several common models and datasets, we show the efficiency of our proposed sampling methods and report superior

performance compared to standard image sampling.

This chapter has been published in The 33rd British Machine Vision Conference (BMVC), 2022.

3.2 Method

3.2.1 Overview

Given an input video v, the task of action recognition is to predict the action class label y of the video. Typically, the input video v can be sampled sparsely, which means that each video is evenly divided into T segments. Then, in the standard 2D paradigm, a frame $\mathbf{x}_t$ is chosen at random for each segment at training time. We call the set of sampled frames $\mathbf{X}$, *i.e.*, $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$.

We let $f(\mathbf{x}_i)$ denote the 2D backbone model that takes a single image as input, and outputs a feature map. We compute $f(\mathbf{X}) = \{f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_T)\}$. Finally, $g(\cdot)$ is a temporal aggregation module that takes the T feature maps and returns the output category prediction $\hat{y}$. The aggregation function $g(\cdot)$ can simply be an average of the individual predictions as in TSN [3, 10],

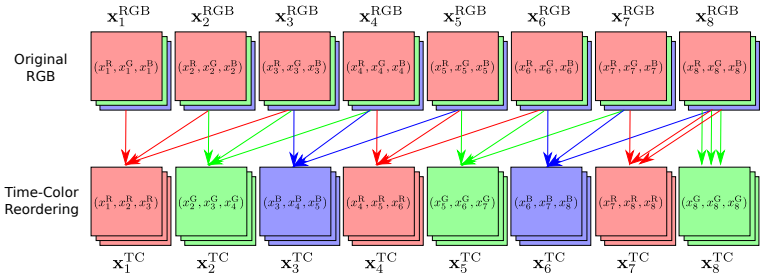
$$g(f(\mathbf{X})) = \frac{\sum_{t=1}^{T} h_\theta(f(\mathbf{x}_t))}{T}, \tag{3.1}$$

where h_θ is a per-frame classifier. In other cases, like TRN [9], the aggregation function reasons about the relationship among multiple frames by concatenating the features and applying a multi-layer perceptron (MLP) h_ϕ,

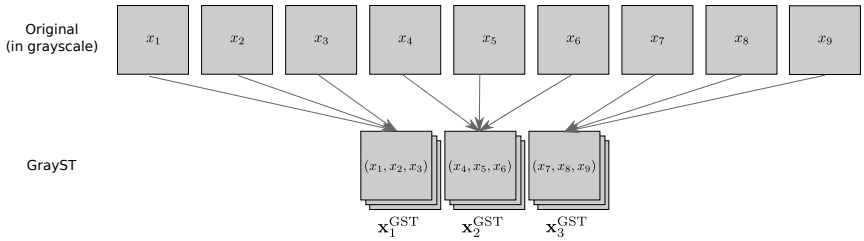
$$g(f(\mathbf{X})) = h_\phi(\text{concat}(f(\mathbf{X}))). \tag{3.2}$$

The multi-scale version of TRN (MTRN) works similarly but has several aggregation modules, each accounting for reasoning between 2-frame, 3-frame, ..., T-frame relationships.

Across variants of the 2D paradigm, the frames of a video are typically sampled sequentially with all color channels intact. Instead, we propose two alternative sampling strategies, *Time-Color Reordering* and *Grayscale Short-Term Stacking* that enables 2D backbone-based models to exploit temporal information by means of reordering the input channels. Despite their conceptual simplicity, these two sampling strategies significantly improve action recognition performance without requiring any structural changes to the backbone model.



(a) 8-frame TC Reordering



(b) 3-frame GrayST. The input video frames are converted to grayscale and the output is simply the concatenation of these frames into groups of three channels

Figure 3.2: Visualization of our TC Reordering (a) and GrayST (b) methods. In each case, the top row represents the original input frames, *i.e.*, eight RGB frames in (a), and nine grayscale frames in (b). Note also that, GrayST gets to use three times as many input frames, *e.g.*, to generate 8 output frames, one needs to sample 24 inputs.

3.2.2 Time-Color Reordering

We propose a simple yet powerful sampling technique for video frames called *Time-Color (TC) Reordering*. We allow the 2D backbone model $f(\mathbf{x}_i)$ to see temporal information by re-sampling the three color channels of the model. We do this by taking one color channel from the input clip (*i.e.*, red) of 3 consecutive frames, and concatenating in the channel dimension to form an input "image". Then we repeat the process with the next color channel (*i.e.*, green), and so on. Note that this is merely a different representation of the input video by means of changing the channel order, and does not require any modification to the backbone model or include additional information or processing. More specifically, the process is as follows. Let $\mathbf{X}^{\mathrm{TC}} = \{\mathbf{x}_1^{\mathrm{TC}}, \mathbf{x}_2^{\mathrm{TC}}, \ldots, \mathbf{x}_T^{\mathrm{TC}}\}$ denote a video clip sampled following the proposed TC Reordering sampling strategy,

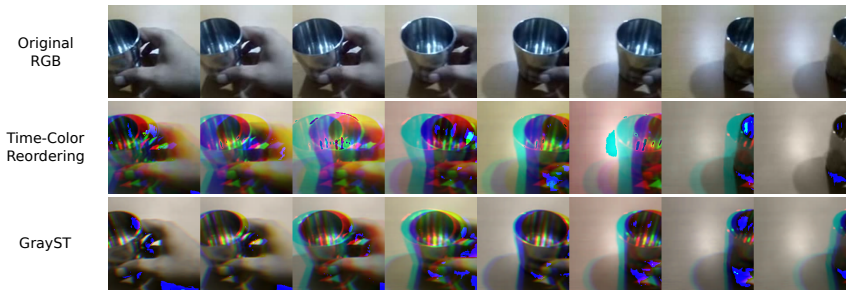


Figure 3.3: Visualization of the different sampling strategies. All frames are interpreted as if they were RGB. The video depicted is an instance of "Pulling something from left to right" from the Something-Something-V2 dataset. Our two sampling strategies make the motion and its direction clearly visible by utilizing three color channels.

where

$$\mathbf{x}_i^{\mathrm{TC}} = \begin{cases} (x_i^{\mathrm{R}}, x_{i+1}^{\mathrm{R}}, x_{i+2}^{\mathrm{R}}), & \text{if } i \bmod 3 = 1. \\ (x_i^{\mathrm{G}}, x_{i+1}^{\mathrm{G}}, x_{i+2}^{\mathrm{G}}), & \text{if } i \bmod 3 = 2. \\ (x_i^{\mathrm{B}}, x_{i+1}^{\mathrm{B}}, x_{i+2}^{\mathrm{B}}), & \text{otherwise.} \end{cases} \tag{3.3}$$

Here, x_i^{R} is the red channel from the i-th frame, x_i^{G} for green, and x_i^{B} for blue. Since the last two TC frames try to access future frames (*i.e.*, x_{T+1} and x_{T+2}) that are not available, we just use x_T with the corresponding color channel for this case. This simple process yields frames that contain information about the neighboring frames, and it is the core reason for the large advantage we observe in temporal tasks. Figure 3.2(a) outlines the procedure in detail.

We alternate color channels to expose the network to more varied data. That is, different channels depict a particular object with different brightness and contrast, while keeping the original shape of the object. We observe that this color alternation produces better results than a single one, and speculate that it may work as a form of data augmentation.

We also propose an alternative method *TC+2* where we sample just two more frames to avoid the duplication of the last frames. The cost of sampling two more frames is almost negligible in many practical scenarios even with long-input networks, and we observe significant improvement of performance as the way the input data is formed is consistent throughout the frames.

3.2.3 Grayscale Short-Term Stacking

Here we introduce another sampling technique that we call *Grayscale Short-Term Stacking* (*GrayST*). This approach is designed to use more source frames with the same compact 2D networks by using grayscale images instead of RGB. The motivation is that in semantic action/behavior understanding tasks, color information can be redundant, and we can use that capacity to include temporal information instead, by inputting more frames. Previous work [75] showed that there is only a 0.5% drop in accuracy on ImageNet [76], when training on grayscale images compared to RGB images. Thus, we replace the three color channels that are normally fed into a 2D backbone network with three grayscale frames, from three different sequential time steps. In effect, this allows the backbone model to see short temporal information at the expense of forgoing the ability to reason about color. This sampling strategy is visualized in Figure 3.2(b).

When the input sequence length to a network is intended to be T frames, we simply sample $3 \times T$ frames in grayscale, and stack the three consecutive frames into one image, containing three grayscale channels. In offline scenarios, we can utilize higher temporal resolution per video clip without introducing any latency for training or testing. Some online scenarios may have a limited number of frames, so we also experiment with matching the number of input RGB frames, *i.e.*, 8-frame GrayST (that sees 24 frames) vs 24-frame RGB.

We let $\mathbf{X}^{\mathrm{GST}} = \left\{\mathbf{x}_1^{\mathrm{GST}}, \mathbf{x}_2^{\mathrm{GST}}, \ldots, \mathbf{x}_T^{\mathrm{GST}}\right\}$ denote a GrayST video clip. We first sample $3 \times T$ grayscale frames following the same sparse sampling strategy as before,

$$\mathbf{X}^g = \left\{x_1^g, x_2^g, \ldots, x_{3T}^g\right\}, \tag{3.4}$$

where x_i^g is a grayscale image. Then, a GrayST frame is made of three neighboring temporal frames in $\mathbf{X}^g$ which is defined as

$$\mathbf{x}_i^{\mathrm{GST}} = (x_{3i-2}^g, x_{3i-1}^g, x_{3i}^g). \tag{3.5}$$

A comparison between the two sampling techniques is visualized in Figure 3.3.

3.3 Evaluation

In this section we evaluate our two channel sampling strategies on multiple different action recognition datasets using several different network architectures.

3.3.1 Experiment Details

Datasets. We experiment with challenging datasets that require extensive temporal reasoning. The datasets are chosen to span a wide range of situations, *e.g.*, short and long range temporal reasoning, static and moving cameras. CATER [74] is a synthetic action recognition dataset involving long-term temporal reasoning. CATER has two versions of the videos, with camera motion and without, and we experiment with both of them. CATER also consists of three different tasks: primitive multi-label action recognition (task 1), compositional multi-label action recognition (task 2), and localization of object of interest (task 3). We evaluate task 2, compositional action recognition, as it requires the longest temporal reasoning (from the beginning to the end of the videos consisting of 301 frames). This provides the biggest challenge for action recognition methods that only focus on short-term clips.

Specifically, the CATER task 2 has 301 classes. Each class represents the ordering of two action pairs: an object performing an action before/during/after another object performing another action, for example, "cone slides before cylinder rotates". Furthermore, CATER allows containment of objects, and even recursion of such containment. That is, the action models have to remember which object is contained by which carrier, during which period, to successfully keep track of all actions happening in the scene.

We also evaluate on datasets that require temporal understanding such as Something-Something-V1 and Something-Something-V2 [29]. The Something-Something datasets consist of videos of pre-defined actions being performed using everyday objects. To focus on the action, and not the objects being used, these datasets removed object and scene bias by grouping the same action using various objects. The labels include "pushing something from right to left", "putting something on a surface", *etc*., which ensures the focus is on the action. Something-Something-V1 contains 108,499 videos with 174 class labels, while Something-Something-V2 consists of 220,847 videos of the same 174 classes.

Networks. We use a ResNet50 [49], pre-trained on ImageNet [76], as our 2D backbone model for all experiments. Both sampling strategies would likely benefit further from the backbone being pre-trained for their specific channel sampling strategies, but we leave this for future work. We mainly present results for five popular 2D models: TSN [3], TRN/MTRN [9], TSM [10], and MVFNet [50] with some additional exper-

iments using I3D [22] pre-trained on Kinetics-400 [47] for completeness. Note, that we were not able to perform 32-frame MTRN experiments due to GPU VRAM limitations, as it has 31 temporal relational modules consisting of dense fully-connected layers.

Training Details. On CATER task 2, CATER Camera Motion task 2, and Something-Something datasets, we used 2 NVIDIA RTX 3090 GPUs to train and test. We used 2 NVIDIA RTX 2080 Ti GPUs for the other datasets. For CATER, we used 32 frames due to the need for long-term temporal understanding. We set the total batch size to 24 and the initial learning rate to 0.0024. For Something-Something, we used 8 frames, a total batch size of 64, and an initial learning rate of 0.0064. As an exception, TRN and MTRN models are trained with one RTX 3090 GPU with half the total batch size and quarter the learning rate.

In all of the experiments, we kept the following protocols. The videos were resized so that the shorter side becomes 224 to 336 pixels, and we performed random cropping during training. For testing, we resized the input to have the shorter spatial side resolution of 256 and we used one center crop for CATER and Something-Something, and five crops (center and corners) and their horizontal flips for other datasets. For I3D experiments, we followed the common practice from [43] of densely sampling the video with a sampling stride of eight for RGB, three for GrayST because it samples more frames, and tested using ten evenly sampled clips throughout the video with three spatial crops of each, totaling 30 clips. The learning rate was decayed by 0.1 when validation metrics saturated for ten epochs. We stopped the experiments when the validation metrics saturated for 20 epochs. We used 16-bit precision to save memory, increase the batch size, and to train faster.

3.3.2 Ablation Studies

TC variants. We explore some variants for ablation experiments: TC-Red, TC-RGB, and TC-ShortLong. TC-Red uses only red channels with the same frame ordering, *i.e.*, $\mathbf{x}_i^{\mathrm{TC}} = (x_i^{\mathrm{R}}, x_{i+1}^{\mathrm{R}}, x_{i+2}^{\mathrm{R}})$. This baseline allows us to measure the effect of the diversity of color information using the TC Reordering. TC-RGB uses traditional RGB-like representation with the same temporal frame ordering as the TC Reordering: $\mathbf{x}_i^{\mathrm{TC}} = (x_i^{\mathrm{R}}, x_{i+1}^{\mathrm{G}}, x_{i+2}^{\mathrm{B}})$. Intuitively, this may seem to be the best representation as this is the closest representation to the RGB representation that the model is pre-trained on.

Model	Sampling	Top-1	Top-5
TSN	RGB	17.2	42.7
	TC	**36.8**	**65.3**
	TC-RGB	33.6	61.4
	TC-Red	36.0	65.2
	TC-ShortLong	35.2	64.4
TSM	RGB	45.4	74.5
	TC	**45.8**	**74.7**
	TC-RGB	44.9	74.1
	TC-Red	44.7	73.8
	TC-ShortLong	44.8	73.6

Table 3.1: Ablation experiments of different TC Reordering strategies on Something-Something-V1. RGB refers to standard RGB channel sampling and TC is our proposed sampling strategy.

However, in our experiments we observe that our TC Reordering actually outperforms TC-RGB. Finally, TC-ShortLong replaces the last two frames that consists of a lot of duplicates, with frames having longer sampling stride instead, *i.e.*, $\mathbf{x}_7^{\mathrm{TC}} = (x_3^{\mathrm{R}}, x_5^{\mathrm{R}}, x_7^{\mathrm{R}})$ and $\mathbf{x}_8^{\mathrm{TC}} = (x_4^{\mathrm{G}}, x_6^{\mathrm{G}}, x_8^{\mathrm{G}})$ for the $T = 8$ case.

In Table 3.1 we contrast the different variants of TC Reordering. In the case of TSN, all variants of TC Reordering result in a significant performance increase compared to standard RGB. However, for TSM, only our proposed TC variant is superior. Perhaps counter-intuitively, TC-RGB, which maintains the RGB channel order but temporally shifts them, actually performs worse than our TC Reordering. TC frames consist of information from the same input color channel, which perhaps better enables it to capture local temporal changes, especially in cases where color is not informative for the task at hand.

Grayscale-only. We report a grayscale-only experiment result on Something-Something-V1 in Table 3.2. We let $\mathbf{X}^{\mathrm{GO}} = \left\{\mathbf{x}_1^{\mathrm{GO}}, \mathbf{x}_2^{\mathrm{GO}}, \ldots, \mathbf{x}_T^{\mathrm{GO}}\right\}$ denote a GrayOnly video clip. We sample T grayscale frames following the sparse sampling strategy,

$$\mathbf{X}^g = \left\{x_1^g, x_2^g, \ldots, x_T^g\right\}, \tag{3.6}$$

where x_i^g is a grayscale image. Then, a GrayOnly frame is made by duplicating the

Model	Sampling	Top-1	Top-5
TSN	GrayOnly	16.1	41.2
	RGB	17.2	42.7
	TC	36.8	65.3
	TC+2	**37.0**	**65.6**
	GrayST	35.5	65.4

Table 3.2: Grayscale-only result on Something-Something-V1. Surprisingly, the performance is very close to that of RGB, and our methods utilize this fact to make the models further capture temporal information.

same channel three times.

$$\mathbf{x}_i^{\mathrm{GO}} = (x_i^g, x_i^g, x_i^g). \tag{3.7}$$

We see very little difference in performance compared to RGB, and our proposed channel sampling strategies (TC, TC+2, and GrayST) take advantage of the fact that the color information does not play a significant role, and maximize the action understanding capabilities of the networks, showing a significant improvement in performance compared to the RGB sampling.

3.3.3 Results

We illustrate results for CATER Static/Camera Motion task 2 in Table 3.3 and Something-Something-V1/V2 in Table 3.4. Note that the GrayST method gets to use three times more frames and TC+2 uses just two more frames than RGB or TC, while maintaining the same computational cost of the model. We also report I3D results on Something-Something-V1, to show the impact of our sampling strategies when applied to 3D networks, see Table 3.4(a).

TC Reordering Performance. Using our TC Reordering, we observe that even the simple TSN model obtains much better performance compared to using conventional RGB frames with TRN and MTRN on all datasets. On TRN and MTRN, we observe significant improvement on most datasets, except on CATER Camera Motion task 2. However, TC Reordering has less impact when combined with TSM and MVFNet, and sometimes hurts performance.

TC Reordering shifts the input video through its color channels in order to provide

Model	Sampling	mAP	
		Static	Camera Motion
TSN	RGB	49.6	51.6
	TC	**73.7**	56.6
	TC+2	73.5	60.5
	GrayST	71.9	**61.9**
TRN	RGB	54.9	54.7
	TC	**72.4**	52.9
	TC+2	72.3	52.9
	GrayST	69.8	**57.6**
TSM	RGB	79.9	65.8
	TC	81.2	63.3
	TC+2	82.0	64.0
	GrayST	**82.2**	**74.7**
MVFNet	RGB	80.2	63.5
	TC	82.1	62.7
	TC+2	82.8	65.5
	GrayST	**83.4**	**67.8**

Table 3.3: Performance on CATER task 2 using 32-frame models (T = 32). We report last epoch's validation mAP using one clip. Note, GrayST uses three times as many frames as RGB or TC, which are then converted to grayscale, and TC+2 uses just two frames more. The different sampling strategies do not impact the size of the network, *i.e.*, in the end, they all get the same number of input frames. Due to the GPU VRAM constraints (as we are using the 32-frame model), MTRN could not be used.

temporal information to 2D backbones. As a result, if a model already captures temporal information (*e.g.*, TSM and MVFNet), TC Reordering is less effective, but still helps in many cases.

GrayST Performance. We observe consistent improvement on GrayST over RGB on most datasets and all models. The first obvious reason may be that it gets to see more frames, *i.e.*, three times as many as the other strategies. By combining grayscale information from different points in time it enables the 2D backbones to see temporal information. Despite utilizing more frames, this method does not increase training and inference times, with the exception of the time associated with loading the extra frames and converting them to grayscale. In comparison, using more RGB frames would make training and testing slower.

8-frame GrayST vs 24-frame RGB. One might wonder if GrayST is better just

(a) Something-Something-V1

Model	Sampling	Top1	Top5
TSN	RGB	17.2	42.7
	TC	36.8	65.3
	TC+2	**37.0**	**65.6**
	GrayST	35.5	65.4
TRN	RGB	29.7	57.6
	TC	35.9	65.2
	TC+2	**37.4**	**66.8**
	GrayST	37.0	66.1
MTRN	RGB	31.0	59.4
	TC	36.6	65.8
	TC+2	38.1	67.5
	GrayST	**38.4**	**67.8**
TSM	RGB	45.4	74.5
	TC	45.8	74.7
	TC+2	46.8	75.8
	GrayST	**47.6**	**76.7**
MVFNet	RGB	46.3	75.3
	TC	45.7	74.6
	TC+2	46.9	75.8
	GrayST	**47.9**	**76.6**
I3D*	RGB	40.5	68.5
	TC	39.1	68.6
	TC+2	40.8	69.2
	GrayST	**41.1**	**70.2**

(b) Something-Something-V2

Model	Sampling	Top1	Top5
TSN	RGB	30.2	60.5
	TC	**51.9**	**79.5**
	TC+2	51.3	79.4
	GrayST	50.9	79.4
TRN	RGB	45.7	73.6
	TC	51.3	78.9
	TC+2	**52.3**	**80.5**
	GrayST	52.1	79.9
MTRN	RGB	46.7	75.3
	TC	52.0	79.9
	TC+2	52.9	80.9
	GrayST	**53.0**	**81.1**
TSM	RGB	59.1	85.6
	TC	59.2	85.5
	TC+2	59.7	86.0
	GrayST	**59.8**	**86.2**
MVFNet	RGB	59.7	85.9
	TC	59.6	85.9
	TC+2	59.7	86.1
	GrayST	**60.8**	**86.6**

Table 3.4: Validation accuracy on Something-Something with 8-frame models ($T = 8$). Models marked with $*$ use 30 clips for testing (3 spatial $\times$ 10 temporal), otherwise we report 1-clip accuracy. Our focus is on making simple 2D networks better, and thus we only report the performance of the expensive I3D on Something-Something-V1 for comparison.

because it 'sees' more frames. However, it has been reported that simply increasing the number of RGB frames does not necessarily improve performance or it is very marginal [9], and can even decrease performance [77] depending on the dataset. For example, [43] showed a baseline I3D-ResNet50 network with a 32-frame input obtained 73.3% on the Kinetics-400 dataset [47]. [1] later conducted an experiment with an 8-frame input following the same recipe for training and obtained 73.4%. We also

Model	Sampling	Model Size (T)	Top-1
TSN	RGB	8	17.2
	RGB	24	18.3
	GrayST	8	**35.5**
TRN	RGB	8	29.7
	RGB	24	31.7
	GrayST	8	**37.0**
TSM	RGB	8	45.4
	RGB	24	**51.1**
	GrayST	8	47.6

Table 3.5: Validation accuracy on Something-Something-V1 for 8 vs 24 frames. T refers to the temporal size of the model, and note that the GrayST gets to access three times as many frames without increasing the model size.

conducted an ablation experiment in Table 3.5 to show that in many cases, the GrayST 8-frame (that looks at 24 frames and generates 8-frame 3-channel representation) is better than the RGB 24-frame. A GrayST frame not only increases the number of frames without increasing training and inference cost, but also allows 2D models to see more temporal information by discarding redundant color information. Note that TSM can make use of the information contained in the extra frames (see 8 versus 24 frames), but this benefit comes with increasing the training/testing cost by a factor of three.

Only Time Can Tell dataset. We show results on OnlyTimeCanTell-Temporal dataset [30] in Table 3.6. The dataset consists of 50 classes which require extensive temporal reasoning, taken from the Kinetics-400 and Something-Something-V1 dataset. Our methods outperformed the baselines by a significant margin on this dataset.

3.3.4 Limitations

Despite the positive results, we also find some limitations of the proposed approaches, which we hope can lead to interesting future work.

Datasets Requiring Less Temporal Reasoning. Numerous datasets have significant object and scene bias making even TSN perform very similar to powerful 3D networks. In such cases, we found that the sampling strategies do not result in improved

Model	Sampling	Top-1	Top-5
TSN	RGB	50.7	84.6
	TC+2	65.9	92.0
	GrayST	**66.4**	**92.7**
TSM	RGB	71.7	94.2
	TC+2	73.3	94.0
	GrayST	**73.7**	**95.0**

Table 3.6: Evaluation of our methods on the OnlyTimeCanTell-Temporal dataset.

Model	Backbone	#frame	Sampling	Top-1
TSN	ResNet50	8	RGB	**75.4**
			TC	75.0
			GrayST	73.5
SlowFast*	ResNet101	16	RGB	77.6

Table 3.7: Evaluation on Diving-48-V2. The result marked with $*$ is from [79], which uses 30 clips for testing (3 spatial $\times$ 10 temporal).

performance. Table 3.7 and 3.8 show such cases on Diving-48 [78] and UCF-101 [28] datasets.

Interestingly, Diving-48 V2 shows a decrease in performance with the GrayST input. The implication of this is that color information is important on this dataset, and with grayscale images it is difficult to distinguish divers of interest from the background. For example, the foreground (divers) is usually in an orange tone, and the background (water) is a teal color. The two colors are complementary, so it creates the maximum separation between the two only by assessing with colors. Despite the fact that the dataset is said to be "temporally-heavy", our experiment showed that the gap between 2D network and the state-of-the-art 3D network with double the number of frames and backbone depth is marginal.

The UCF-101 labels are biased towards object and scene information. As a result the performance of TSM is almost identical to that of TSN. The result did not show a strong pattern but in general RGB seems to be preferable in this case.

Camera Motion. We found that TC Reordering combined with TRN and TSM negatively impacts performance on the CATER Camera Motion dataset. Note that the

Model	Backbone	Sampling	Top-1	Top-5
TSN	ResNet50	RGB	83.6	**96.1**
		TC	83.6	96.0
		GrayST	**84.7**	95.9
TRN	ResNet50	RGB	**84.6**	**96.6**
		TC	81.6	95.3
		GrayST	82.2	95.5
TSM	ResNet50	RGB	**83.7**	96.0
		TC	81.4	95.5
		GrayST	83.6	**96.1**

Table 3.8: 8-frame evaluation on UCF-101 split 1.

models make use of the temporal ordering of frames while TSN does not. Again, we still saw improvement with the GrayST method on this dataset. Judging from the fact that TC Reordering only hurts the temporal models, we think that this ordering plays a critical role in heavy camera motion scenarios.

Additionally, this dataset requires 3D-geometric understanding as the camera orbits around the objects substantially, making stationary objects look like they are sliding. The strength of TC Reordering is in cases where the model can analyze the motion information, but the large camera motion likely confuses the model when trying to understand what is the action of interest and what is the camera motion.

3.4 Conclusion

Complex action recognition models can sometimes be too expensive, and thus it is vital to improve performance on the efficient 2D action models. We presented three efficient and cost-free channel sampling strategies for action recognition, primarily for efficient 2D backbone-based networks. We saw a significant improvement in performance on various challenging action recognition datasets.

TC Reordering. This method re-orders the color channels over time to increase temporal information in each frame. This method does not require sampling more frames compared to the standard RGB representation, however, it duplicates some channels on the last two frames, making the representation slightly inconsistent. This

method often improves performance, while the result can be unstable depending on the situation such as heavy camera movement. We tried many variations of TC Reordering, and concluded that reordering color across frames (*e.g.*, the first frame uses red, the second uses green, and the third uses blue) gives the best performance.

TC+2. In order to tackle the minor inconsistent encoding issue of the TC Reordering, we also proposed TC+2, which samples just two more frames compared to the RGB representation, but achieves a very stable performance boost.

GrayST. This method uses grayscale images to increase the temporal receptive field. It samples three times as many frames compared to the RGB representation, and thus produces much more detailed temporal images in return. This method generally achieves the best performance among all the sampling strategies proposed.

Discussions. It is an interesting discovery that just shuffling video channels using TC Reordering can improve action recognition performance significantly. Furthermore, the method improves performance, even though it is marginal, on more recent models, despite the fact that they are known to capture temporal information effectively. This seems to be because the representation not only increases the temporal information per frame, but also alters the color channel over time behaving like a data augmentation. The GrayST approach uses more frames, and this is probably why it works best. It may be interesting to see if we can apply a similar data augmentation on GrayST by altering color channels instead of converting them to grayscale. We leave this as a future work. The implementation of all strategies is extremely simple, so we believe that the methods can be widely used in any deep learning framework.

Limitations. The proposed methods may not perform the best with datasets that do not require much temporal reasoning or require excessive color analysis, such as HMDB, UCF, and Diving-48. Furthermore, when used with models with high temporal understanding capabilities, we did not see a huge improvement using the methods.

Chapter 4

Handling Ambiguity in Action Recognition

4.1 Introduction

Actions are often fundamentally ambiguous. Barker and Wright [80] explained that the hierarchical nature of an action is described differently depending on the viewpoint. In their example of "children going to school", one could consider the atomic body movement (*e.g.*, "going down the stairs") to the goal and intention of the action (*e.g.*, "walking to the school", "getting an education"). This fundamental ambiguity and complexity in actions seems to influence the way humans learn verbs. For many languages, it has been observed that infants tend to learn nouns before verbs [81]. Furthermore, according to WordNet [82], the number of unique nouns in the English vocabulary is more than ten times that of verbs, allowing for a more precise categorization. Similarly, researchers also face difficulties in the computational setting when naming, categorizing and learning actions in videos [83].

Designing an action label space for a given dataset where each video is assigned a single unique label is an extremely difficult task. For example, consider the verbs "stir" and "mix" as labels for a cooking video. You can stir to mix the ingredients, but stirring is not the only way to mix them (*e.g.*, you can also "shake"). Since each verb can often have multiple meanings, the label space can partially overlap. Specifically, one label can represent multiple types of actions, and many labels can represent a single type of action, as in Figure 4.1. This is particularly difficult when datasets are large, and so is the label space, making it a combinatorial problem of finding overlapping classes for every instance. Labeling datasets in a thorough multi-label fashion can

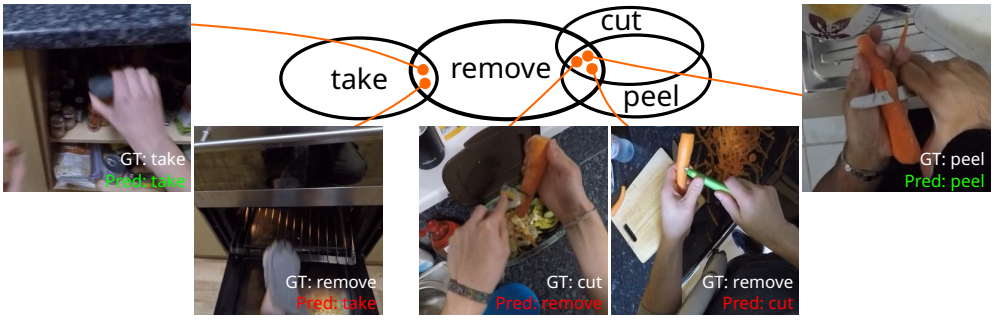


Figure 4.1: Ambiguity caused by single-verb labels. The same action can be described in different ways, *e.g.*, the act of peeling a carrot can be annotated as "cut", "peel" or "remove" (skin). Large action recognition datasets gather several instances with different but equivalent verb labels. Here we show some examples from the EPIC-Kitchens dataset [31]. Recognition models can be confused by this semantic ambiguity and are usually not evaluated taking this verb overlap into account. Reasonable and semantically correct predictions can be counted as incorrect (*e.g.*, red text, middle frames) by standard evaluation protocols as the predicted label does not match the noisy ground truth.

be prohibitively expensive. As a result, we typically only get partially-labeled video datasets, which negatively impacts both training and testing.

To address this, we propose a new training procedure to handle the problem of missing and ambiguous labels in action recognition. In particular, we generate pseudo-labels by searching neighboring video instances in the feature space of the training set and analyze how similar videos are labeled differently. We take the commonly-agreed labels to be pseudo-labels (*i.e.*, labels that are highly likely to be positive and thus relevant for the query video) and train action recognition models to either not penalize the potentially-positive classes or boost the learning signal for them. In order to evaluate our proposed methods we manually annotated a large subset of the validation set of the EPIC-Kitchens-100 dataset [32] with multiple verbs. We also evaluate our methods on a modified version of the popular HMDB-51 dataset [27], where we added synthetic ambiguity. Overall, our methods improve multi-label accuracy up to 18% over several recent baselines. To summarize:

- We show that the ambiguity in current action recognition labeling leads to issues in training and testing models. We show "failure cases" on challenging datasets where the model predicts reasonable labels but the accuracy is penalized because of label ambiguity.

- To mitigate the issue, we introduce two novel approaches for training multi-label action recognition models when only single positive labels are available. Our approaches attempt to disambiguate the label space by generating pseudo-labels from similar instances that are labeled differently.
- We introduce a multi-label subset of the EPIC-Kitchens-100 validation set for the purpose of multi-label evaluation. On this dataset, and another with synthetically generated label ambiguity, we report results that outperform existing methods.
- The dataset and code are publicly released[1].

This chapter has been published in The 33rd British Machine Vision Conference (BMVC), 2022.

4.2 Related Work

Noisy labels and images. [84] provides a summary of methods that learn from noisy labels. Label smoothing [85] is one popular and simple method to prevent models from being overly confident by converting hard labels to soft ones. This method may be effective when there are only a few labeling mistakes, however, this method does not deal with the problem of having interchangeable labels. Progressive Self Label Correction [7] is a label correction technique that mixes model predictions with labels. This method assigns a larger weight to the prediction of a given class if the model still outputs high confidence for a class after some initial training. Notice that this method tries to "correct" the label, not allowing that having multiple positive labels can be an option. Another method represents uncertainty with multiple hypotheses [86], in other words, have multiple outputs with multiple losses so that it can disambiguate an uncertain problem. [87] uses interpolated (mixup) images to detect and degrade the presence of label noise. A noisy sample is either interpolated with a clean sample that degrades the effect of the noise, or with another noisy sample which makes the model difficult to memorize the pattern. It also optimizes jointly with contrastive learning along with a classification loss. [88] corrects the loss by estimating probabilities of one class being flipped to another. However, in our problem, we have multiple classes that are all valid, instead of having commonly mistaken classes. [89] optimizes network parameters and labels alternatively. This is another label correction approach that again does not align with our problem.

[1] https://github.com/kiyoon/verb_ambiguity

These methods are commonly evaluated on image datasets with synthetic noise or on noisy image datasets collected from the web. In the latter case, noise derives from incorrect annotations or imperfect data collection. In our setting, noise stems from semantic ambiguity, *i.e.*, multiple equivalent labels can apply to the same action, which involves arguably more complicated dynamics compared to noise caused by labeling omissions.

Visual and semantic ambiguity. [90] proposed a method to learn actions from visually similar instances with different semantic labels. Their framework builds a graph where nodes are connected if they are visually similar or semantically related. A Markov walk through this graph estimates the action label of a new video. This work is very close in spirit to ours, however experiments showed that it requires fine-grained verb meaning labels in order to achieve good results. On the contrary, we do not require further annotations and only use the available single-label ground truth. Furthermore, we release multi-label annotations for better testing of our methods. [83] identified the issues arising from single-verb labels due to semantic ambiguity. They proposed training with multi-verb representations, which were obtained by asking annotators to label one representative sample per action with all verbs they deemed relevant. Verbs collected for a given action sample were then attached to the other instances of the action. This work is also closely related to ours, however we stress again that we only require single-label ground truth for training. We also show in our experiments that instance-based multi-verb representations better alleviate semantic ambiguity compared to global dataset-based representations. Visual Sense Disambiguation [91] aims to identify the context of an action given an image and a verb. Understanding visual and semantic relationships in the label space has also been shown to improve multi-label image classification [92]. [93] use language priors in the form of word embeddings to find relationships between verbs in human-object interaction data. However, our preliminary experiments with word embeddings failed to produce sensible similarity estimates for the finer-grained datasets we use in this work. [94] address the problem of missing label imputation for scene graph generation proposing an iterative pseudo label approach.

Single positive multi-label learning. There has been recent interest in the problem of multi-label learning from only single positive labels. The problem is outlined in [5], where the authors propose a loss that encourages models to predict both known and

unlabeled positives. The loss also adds a regularization term to match the expected number of positives per image. [6] learns from the annotated positive labels using an entropy maximization approach and decreases the loss penalty for unknown labels. Their method also treats samples with the lowest confidence as pseudo negatives. [95] outlined an image-specific spatial consistency loss that measures consistency in model predictions over multiple random crops from the input image. A similar problem of multi-label learning from missing labels has been tackled by leveraging the estimated distribution of missing and known labels [96, 97].

Semi-supervised learning. Semi-supervised learning approaches tackle the problem of having partially annotated labels, which translates to having noisy label estimates. This problem considers partially annotated instances in single-label classification tasks and does not entail multi-label classification. In the context of image datasets, [98] showed that self-supervised contrastive learning can result in representations that are more robust to label noise. Contrastive learning has also been explored for action recognition by using only a small subset of the labels [99]. Recently, an uncertainty-aware pseudo-label selection framework was proposed for both image and video [100]. Pseudo-labeling was also used to group potentially similar classes in [101]. The major benefit of pseudo-labeling over contrastive learning is that it does not rely on domain-specific data augmentation.

4.3 Problem and Methodology

As outlined above, in contrast to nouns, verb annotations are prone to semantic ambiguity. Such ambiguity stems from two issues: i) verbs have different meanings, thus a single verb can be attached to visually different actions and ii) individuals can often use different verbs to annotate the same action. Figure 4.1 shows an example of semantic ambiguity, where the act of peeling a carrot in EPIC Kitchens [31] is labeled as “peel”, “cut”, and “remove” (skin). This is a common issue in large datasets where multiple annotators label videos and thus it is hard to enforce consistent classes. While noun labels alleviate semantic ambiguity, they add to the annotation burden and make learning actions a fragmented process. In fact, as noted in [90, 83], nouns are mostly used to facilitate learning, however they unnaturally split the same action into different classes. For example, “open-door” and “open-drawer” are the same action, but belong to separate classes when categories are indicated by both verbs and nouns. Annotating

actions with multiple verbs [83] also mitigates ambiguity, however it comes with the cost of choosing multiple labels for each video. Models are typically not designed, nor evaluated, taking this semantic ambiguity into account. We address the semantic ambiguity introduced by single-verb labels with two methods which generate multi-verb pseudo-labels from single-verb annotations during training. We also evaluate our methods across several metrics better suited for datasets with semantic ambiguity. We next formalize the problem and review common baselines, before introducing our methods.

Formalization. We consider our problem to be related to that of single positive multi-label learning (SPML) [5, 6]. Here, we have a training and validation set where instances have only one positive class label. Other class labels are unknown (unannotated) and could be either positive or negative (*i.e.*, present or absent). For evaluation, instances in the test set have multiple class labels. Formally, let $\Gamma = \{1,\ldots,C\}$ be the set of class labels in a given dataset, where C is the number of classes. Each training/validation video is annotated with a single positive class label $y_i \in \Gamma$. The remaining classes in $\Gamma \setminus \{y_i\}$ are not annotated and cannot be assumed to be negative. A test video is annotated with a positive class label set $\mathcal{Y}_i \subset \Gamma$ which can contain multiple classes, and remaining the classes $\Gamma \setminus \mathcal{Y}_i$ are treated as confirmed negatives.

SPML. One of the most common approaches in SPML is the "assume negative" loss (AN) [5]. This treats all unknown labels as negative ones and uses the binary cross-entropy (BCE) loss for multi-label learning. Given a training video-label pair $(\mathbf{x}_i, y_i)$:

$$\mathcal{L}_{\text{AN}}(\mathbf{x}_i, y_i) = -\frac{1}{C}\sum_{c=1}^{C}\left[\mathbb{1}_{[y_i=c]}\log\left(f^{(c)}(\mathbf{x}_i)\right) + \mathbb{1}_{[y_i\neq c]}\log\left(1-f^{(c)}(\mathbf{x}_i)\right)\right] \tag{4.1}$$

where $\mathbb{1}_{[.]}$ is the indicator function and $f^{(c)}(\mathbf{x}_i)$ is the model prediction for class c. To address the imbalance between the known positive labels and the unknown assumed-negative labels, an alternative version of AN is proposed: the "weak assume negative" loss (WAN) [5]. This improves over AN in that it gives equal weight for all assumed negatives and the single positive, however all unknown labels are still treated as negatives. Label smoothing (LS) [85] is another popular approach to prevent models from being overly confident. A variant of LS only for assumed negative labels (N-LS) was also proposed in [6]. This improves over LS in that it only alters unknown (assumed

negative) labels. The focal loss [102], originally designed for object detection, has been used for SPML as well. This works well in the presence of imbalanced labels, however it requires additional parameter tuning to achieve optimal performance. Finally, the entropy maximization (EM) loss [6] allows the unknown labels to be unknown rather than assuming them to be negatives. EM learns mainly from annotated labels and attains state-of-the-art results in SPML on *image* datasets.

For convenience, $f_i^{(c)} = f^{(c)}(\mathbf{x}_i)$ denotes the model prediction confidence for class c. Firstly, Weak Assume Negative (WAN) loss [5] has a negative balancing weight $\frac{1}{C}$ from the Assume Negative (AN) loss:

$$\mathcal{L}_{\text{WAN}}(\mathbf{x}_i, y_i) = -\frac{1}{C}\sum_{c=1}^{C}\left[\mathbb{1}_{[y_i=c]}\log\left(f^{(c)}(\mathbf{x}_i)\right) + \mathbb{1}_{[y_i\neq c]}\frac{1}{C-1}\log\left(1-f^{(c)}(\mathbf{x}_i)\right)\right] \tag{4.2}$$

Label Smoothing (LS) loss [85] is defined as:

$$\mathcal{L}_{\text{LS}}(\mathbf{x}_i, y_i) = -\frac{1}{C}\sum_{c=1}^{C}\left[\mathbb{1}^{\frac{\varepsilon}{2}}_{[y_i=c]}\log\left(f^{(c)}(\mathbf{x}_i)\right) + \mathbb{1}^{\frac{\varepsilon}{2}}_{[y_i\neq c]}\log\left(1-f^{(c)}(\mathbf{x}_i)\right)\right] \tag{4.3}$$

where $\mathbb{1}^{\alpha}_{[Q]} = (1-\alpha)\mathbb{1}_{[Q]} + \alpha\mathbb{1}_{[\neg Q]}$ for any logical preposition Q. The label smoothing only for assumed negatives (N-LS) [6] would be the same only for the assumed negatives:

$$\begin{aligned}\mathcal{L}_{\text{N-LS}}(\mathbf{x}_i, y_i) = -\frac{1}{C}\sum_{c=1}^{C}\Bigg[&\mathbb{1}_{[y_i=c]}\log\left(f^{(c)}(\mathbf{x}_i)\right) \\ &+ \mathbb{1}_{[y_i\neq c]}\left[(1-\varepsilon)\log\left(1-f^{(c)}(\mathbf{x}_i)\right) + \varepsilon\log\left(f^{(c)}(\mathbf{x}_i)\right)\right]\Bigg]\end{aligned} \tag{4.4}$$

We used $\varepsilon = 0.1$ throughout the chapter. Focal loss [102] has a balancing parameter α and a focusing parameter γ:

$$\begin{aligned}\mathcal{L}_{\text{Focal}}(\mathbf{x}_i, y_i) = -\frac{1}{C}\sum_{c=1}^{C}\Bigg[&\mathbb{1}_{[y_i=c]}\left[\alpha(1-f^{(c)}(\mathbf{x}_i))^{\gamma}\log\left(f^{(c)}(\mathbf{x}_i)\right)\right] \\ &+ \mathbb{1}_{[y_i\neq c]}\left[(1-\alpha)(f^{(c)}(\mathbf{x}_i))^{\gamma}\log(1-f^{(c)}(\mathbf{x}_i))\right]\Bigg]\end{aligned} \tag{4.5}$$

We set $\alpha = 0.25$ and $\gamma = 2$. Finally, the Entropy Maximisation loss [6] is defined as follows:

$$\mathcal{L}_{\text{EM}}(\mathbf{x}_i, y_i) = -\frac{1}{C}\sum_{c=1}^{C}\left[\mathbb{1}_{[y_i=c]}\log\left(f^{(c)}(\mathbf{x}_i)\right) + \mathbb{1}_{[y_i\neq c]}\alpha H(f^{(c)}(\mathbf{x}_i))\right] \tag{4.6}$$

$$H(f^{(c)}(\mathbf{x}_i)) = -\left[f^{(c)}(\mathbf{x}_i)\log\left(f^{(c)}(\mathbf{x}_i)\right) + (1-f^{(c)}(\mathbf{x}_i))\log\left(1-f^{(c)}(\mathbf{x}_i)\right)\right] \tag{4.7}$$

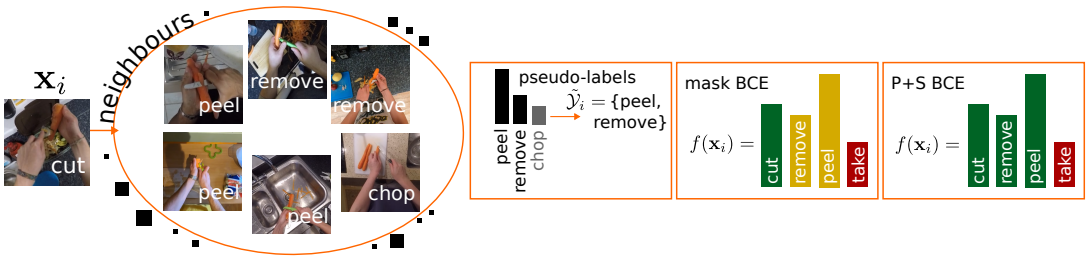


Figure 4.2: Summary of our approach. Given a training video $\mathbf{x}_i$ we first retrieve its nearest neighbors in the feature space. Looking at the frequency of the neighbors' classes we then obtain the pseudo-labels set $\tilde{\mathcal{Y}}_i$. Pseudo-labels are used to optimize the model given its predictions $f(\mathbf{x}_i)$. The model is forced to predict the ground truth class (green bar) and penalized when predicting negative classes with high confidence (red bar). When using our Mask BCE loss, the model predictions for the pseudo-labels (yellow bars) are masked out during back-propagation, thus the model is neither penalized nor rewarded for guessing reasonable classes. When using our P+S BCE loss, pseudo-labels are assumed to be positive classes, thus the model is forced to predict both the ground truth and the pseudo-labels.

We compare our methods described in Section 4.3.1 against these losses, in Section 4.5.1.

4.3.1 Mask and Pseudo+Single-label BCE

The above methods were mostly designed for partial labeled setting, where instances potentially contain multiple classes but *classes do not semantically overlap*. In this setting there are multiple "things" present in an instance and the task is to label all of them, using only the available single-label annotations at training time. In our case, however, instances can belong to multiple classes *because classes can semantically overlap*. We thus treat the unknown labels such that the model does not get penalized if it predicts an unknown label with high likelihood. Specifically, we pick classes that are likely to be positive based on feature similarity. We treat these as pseudo-labels. The model learns from single-verb ground truth annotations, however it is not penalized if it predicts a pseudo-label with confidence. Classes that are neither ground truth nor pseudo-labels are assumed negative.

To obtain pseudo-labels we extract features for each sample in the training set using a pre-trained action recognition backbone. Given a training pair $(\mathbf{x}_i, y_i)$ we take K neighboring videos in the feature space. These videos are likely to share the same

label as $\mathbf{x}_i$, however we also expect these videos to belong to different classes given the semantically ambiguous setting we work in. We build pseudo-labels for $\mathbf{x}_i$ by looking at the label frequency of its K neighbors. Formally, let Ω be the sequence of labels found among the K neighbors, and let Υ be the set of unique labels in Ω. For each $y_j \in \Upsilon$ we count its frequency in Ω, *i.e.*, $\omega(y_j) = |(\eta \in \Omega : \eta = y_j)|/K$. The pseudo-label set $\tilde{\mathcal{Y}}_i$ is finally obtained from Υ by applying a threshold τ to $\omega(y_j)$, *i.e.*, $\tilde{\mathcal{Y}}_i = \{y_j \in \Upsilon : \omega(y_j) > \tau, y_j \neq y_i\}$.

Note that we do not add the original label to the pseudo-labels. Unlike other methods [6, 7] which generate pseudo-labels from model predictions, we use actual human annotations to produce pseudo-labels. With this approach, the ambiguous label space is realistically well-represented. Clearly we rely on selecting informative neighbors, but we find empirically that the backbone we use is sufficient for this task.

With the above defined, we now propose our **Mask BCE** loss. This loss treats the single-label ground truth as the only positive, pseudo-labels as unknown classes, and assumes the remaining classes are negative:

$$\mathcal{L}_{\text{mask}}(\mathbf{x}_i, y_i) = -\frac{1}{C - |\tilde{\mathcal{Y}}_i|} \sum_{\substack{c=1 \\ c \notin \tilde{\mathcal{Y}}_i}}^{C} \left[\mathbb{1}_{[y_i = c]} \log\left(f^{(c)}(\mathbf{x}_i)\right) + \mathbb{1}_{[y_i \neq c]} \log\left(1 - f^{(c)}(\mathbf{x}_i)\right) \right] \tag{4.8}$$

Here, the output of the model $f^{(c)}(\mathbf{x}_i)$ for classes $c \in \tilde{\mathcal{Y}}_i$ is detached from the gradient computation and back-propagation of the loss, *i.e.*, pseudo-label classes are frozen and do not provide any learning signal. This does not penalize the model if it produces reasonable predictions for relevant classes, *i.e.*, if it outputs a high likelihood for a class that is potentially an equivalent label for the video. Importantly, this loss does not send incorrect positive learning signals when the pseudo-labels are not accurate. Instead, when pseudo-labels can be assumed to be correct, treating pseudo-labels as positive would be a better strategy.

With this in mind, we propose an alternative loss named **Pseudo+Single-label BCE**:

$$\begin{aligned} \mathcal{L}_{\text{P+S}}(\mathbf{x}_i, y_i) = -\frac{1}{C} \sum_{c=1}^{C} \Bigg[& \mathbb{1}_{[y_i = c]} \log\left(f^{(c)}(\mathbf{x}_i)\right) + \mathbb{1}_{[c \in \tilde{\mathcal{Y}}_i]} \log\left(f^{(c)}(\mathbf{x}_i)\right) \\ & + \mathbb{1}_{[y_i \neq c]} \mathbb{1}_{[c \notin \tilde{\mathcal{Y}}_i]} \log\left(1 - f^{(c)}(\mathbf{x}_i)\right) \Bigg] \end{aligned} \tag{4.9}$$

Intuitively, $\mathcal{L}_{\text{P+S}}$ will work better than $\mathcal{L}_{\text{mask}}$ when the pseudo-labels are accurate, but

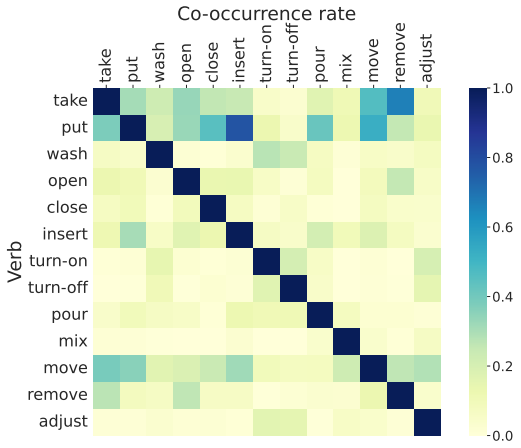


Figure 4.3: Co-occurrence ratio between verbs in our final multi-label annotations. The final annotations refer to verbs that at least half of the annotators agreed to be valid, filtering out noisy labels that only one annotator used. Rows are normalized by the number of classes which makes it asymmetric, *e.g.*, 1 indicates two verbs always co-occur (diagonal), 0.5 means they occur half of the time together. "Take" and "remove" frequently co-occur, and so do "put" and "insert". Only head (top 13 most frequently occurring) classes are visualized.

it will hurt performance when they are not. Figure 4.2 summarizes our pseudo-label generation and the two losses described above.

4.4 EPIC-Kitchens-100-SPMV annotations

Here we describe the annotation protocol we followed to compile our EPIC-Kitchens-100-SPMV dataset. Figure 4.3 shows the co-occurrence frequency of head (top 13 most frequently occurring) classes. Figure 4.4 depicts our annotation interface.

Annotating difficult samples. We first trained a TSM [10] RGB model on EPIC-Kitchens-100 [32]. This model achieved 60.9% verb accuracy on the original validation set. We then annotated failure cases where the top-1 prediction was incorrect, which amounted to 39.1% of EPIC-Kitchens-100's validation set (3,782 videos). Note that our annotation is multi-label.

Verb candidates. Annotators were presented with a video containing an action and were asked to choose any verb they saw fit to describe the action. Annotators could

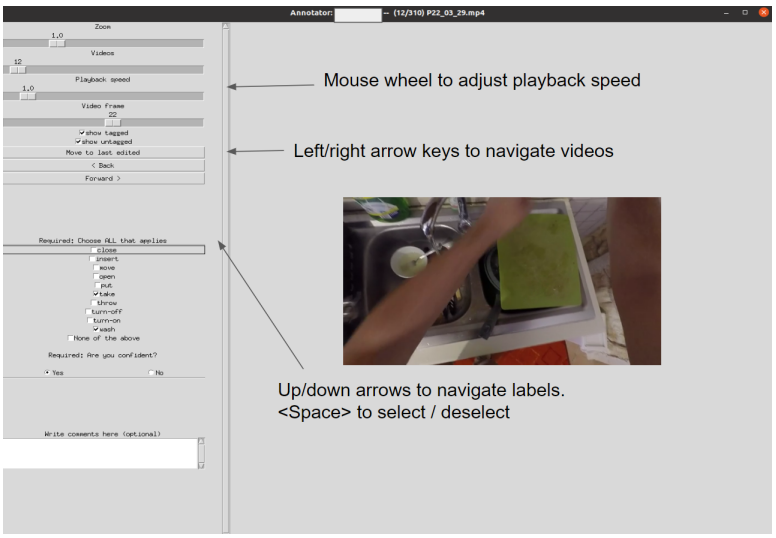


Figure 4.4: Our annotation interface.

also choose no labels in case the video was noisy and did not show any action. Annotators could not see the original ground truth and could only choose from a limited list of verbs. We exploited the TSM model predictions to compile a list of verb candidates for each video. Specifically, we observed that in many cases the model was predicting reasonable verbs within its top predictions. To facilitate labeling we thus show annotators only the 10 verbs corresponding to the top 10 predictions of the model for a given video. We also included the ground truth single-verb label among the verb list (without telling annotators that this was the ground truth). This was to analyze the performance of the annotators: if they did not choose the ground truth verb in most cases then they likely did not pay enough attention to the video. Regardless of the chosen verbs we always included the original ground truth in the final multi-labels.

Multiple annotators. For robustness each video was annotated by at least three different annotators. In some cases annotators did not meet our quality requirements, *e.g.*, they chose too few verbs on average per video or failed to choose the ground truth most of the time. In these cases we asked other annotators to label the same videos. In total we employed 26 annotators. People were students mostly from the School of Informatics.

Filtering noisy samples. For each video annotators were asked whether they were confident about their verb selection or not. We discarded samples where more than half annotators did not feel confident. Many of the discarded videos include actions

happening outside the viewpoint, segments with bad temporal boundaries or where no action was happening. Altogether, we filtered out 289 videos from the initial 3,782 samples, compiling a set of 3,493 clean annotations.

Finalizing labels. For a given video we kept labels that were chosen by at least half of the annotators. For example, for a video annotated by three people a verb would have to be chosen at least twice to be kept in the video multi-verb annotations. We also included the original single-label ground truth regardless of the annotators' choice.

Held-out validation set. Our 3,493 multi-verb annotations constitute our test set. The remaining $9,668 - 3,493 = 6,175$ videos from EPIC-Kitchens-100's validation set constitute our own validation set, where videos are annotated with only one verb.

Fleiss' kappa agreement score. We computed Fleiss' kappa to measure agreement between the three different annotators. We report an average κ of 0.52, which is typically translated as moderate agreement. Labeling actions is highly subjective, thus we believe annotators showed a satisfactory agreement.

4.5 Experiments

Datasets. We manually annotated 40% of the EPIC-Kitchens-100 [32] validation set with multiple verbs, using the procedure described in Section 4.4. This subset is our test set and consists of 3,493 multi-verb videos with 2.4 labels per video on average. The remaining videos in the original validation set form our own validation set. The training set is the same as the original one. Figure 4.5 illustrates a few examples from our dataset, while Figure 4.6 shows the verb distribution from our multi-label annotations and the original labels. Our annotations share a very similar distribution compared to the original ones, thus no bias towards specific classes is introduced with our labels. We call this dataset *EPIC-Kitchens-100-SPMV* (Single Positive Multi-Verb learning). We also constructed a dataset from the popular HMDB-51 dataset [27] with synthetic labels in order to simulate ambiguous annotations. We doubled the original 51 classes to 102 classes, where each class c is split into c^0 and c^1. Given a training video $\mathbf{x}_i^{train}$ originally belonging to c, we randomly assign it to *either* c^0 or c^1. Given a test video $\mathbf{x}_i^{test}$ originally belonging to c, we assign it to *both* c^0 and c^1. This simulates verb ambiguity, where actions are represented with multiple verbs. We name this

Figure 4.5: Samples from our EPIC-Kitchens-100-SPMV dataset. The yellow text indicates the original label from EPIC-Kitchens-100 [32], which is also included in our annotations. The white text shows the additional labels we gathered from multiple annotators.

dataset *Confusing-HMDB-102*.

Metrics. We evaluate *Top-Set Multi-label Accuracy* following [83]. This measures how many top predictions match the ground truth: $A_{\text{Top-set\_ML}} = \frac{1}{N}\sum_{i=1}^{N} |\mathcal{Y}_i \cap \hat{\mathcal{Y}}_i|/|\mathcal{Y}_i|$, where N is the total number of videos, $\mathcal{Y}_i$ is the multi-label ground truth, and $\hat{\mathcal{Y}}_i$ is the top-K predicted classes, where K is the number of ground-truth labels. We also report another metric called *Top-1 Multi-label Accuracy*: $A_{\text{Top1\_ML}} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{[\hat{y}_i \in \mathcal{Y}_i]}$, where $\hat{y}_i$ is the top-1 predicted class. This metric is more relaxed and does not penalize the model when it predicts a relevant verb, regardless of the specific choice. For example, when the ground-truth is ("stir", "mix"), the model is free to choose either "stir" or "mix", and not predicting the other one will not lower this accuracy. This is different from Top-Set Multi-label Accuracy, where predicting only one label in this case would give 50% accuracy, and the model would have to predict both "stir" and "mix" to obtain 100%. We also evaluate *IOU Accuracy* [103], which is the intersection over union between the ground truth and predicted labels, and *F_1-Measure*, which is the harmonic mean of precision and recall. We set the confidence threshold to 0.5, so any predictions over 0.5 will be treated as positive labels.

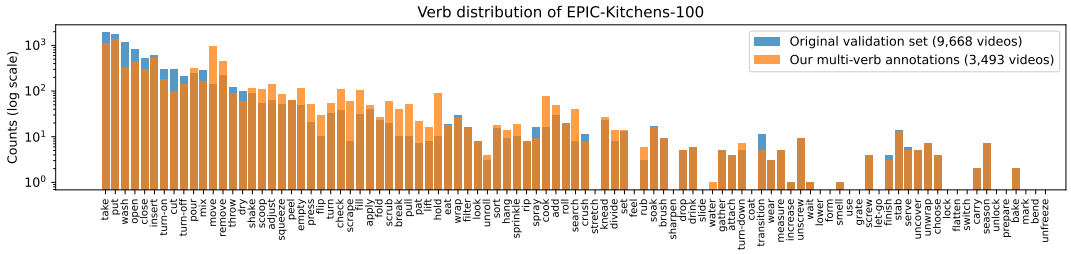


Figure 4.6: EPIC-Kitchens-100 verb distribution from our multi-label annotations (orange) compared to the original labels (blue). Although we only annotated 40% of the validation dataset, we obtained similar or even more labels per class. Note that some generic verbs (*e.g.*, move, hold, search) gained a significant number of annotations compared to the original.

Let N be the number of videos in the test set, $\mathcal{Y}_i$ be the set of ground truth, and $\hat{\mathcal{Y}}_i$ be the set of predicted classes with over 50% prediction confidence. The IOU accuracy and F_1-Measure are defined as follow: **IOU accuracy:**

$$A_{\text{IOU}} = \frac{1}{N}\sum_{i=1}^{N}\frac{|\mathcal{Y}_i \cap \hat{\mathcal{Y}}_i|}{|\mathcal{Y}_i \cup \hat{\mathcal{Y}}_i|} \tag{4.10}$$

F_1-Measure:

$$F_1 = \frac{1}{N}\sum_{i=1}^{N}\frac{2|\mathcal{Y}_i \cap \hat{\mathcal{Y}}_i|}{|\mathcal{Y}_i| + |\hat{\mathcal{Y}}_i|} \tag{4.11}$$

Finally, we report mean average precision *mAP*. This is a popular metric for detection tasks. However, mAP is not very suitable for long-tail datasets such as EPIC-Kitchens, since class scores are averaged with equal weight. For EPIC-Kitchens-100-SPMV, we choose the best model by looking at the standard top-1 accuracy on our single-label validation set, reporting results obtained on five different runs with different seeds. For Confusing-HMDB-102 we take the best score per metric and take the average over the official three splits.

Implementation details. We pre-extracted RGB and optical flow features using TSM [10] pre-trained on each dataset's training set using a standard cross entropy loss with the single positive labels. TSM shares information across frames and has been shown to perform well on multiple action-orientated datasets. The concatenated RGB and optical flow features form the input to our model, which is a multi-layer

(a) Mask BCE loss

τ/K	3	5	10	15	20
0.1	27.2	30.5	31.0	36.4	36.2
0.2	27.2	23.8	26.3	27.5	28.6
0.3	27.2	23.8	23.9	25.8	25.1

(b) P+S BCE loss

τ/K	3	5	10	15	20
0.1	37.9	42.7	41.6	44.5	43.7
0.2	37.9	30.5	34.3	34.7	37.6
0.3	37.9	30.5	29.7	33.5	31.9

Table 4.1: F_1-Measure for our Mask and P+S BCE loss on EPIC-Kitchens-100-SPMV with different values of τ and K.

perceptron (MLP) with three layers and hidden dimension of 1024. We train the MLP using the Adam optimizer [104] with a batch size of 64 and learning rate of 5e-6. We stopped training whenever the validation accuracy did not improve for 20 epochs. For EPIC-Kitchens-100-SPMV all experiments took less than 130 epochs to saturate. For Confusing-HMDB-102 it took longer, less than 390 epochs, given the much smaller size. Pseudo-labels are generated only once before training. We set the number of neighbors $K = 15$ and use a threshold of $\tau = 0.1$ to obtain pseudo-labels.

4.5.1 Results

Impact of K and τ. There are two hyperparameters involved in our methods: the number of neighbors (K) and the pseudo-label threshold (τ). Table 4.1 shows the impact of the different choices of K and τ. Results improve with a larger number of neighbors and a smaller τ, which entail a larger set of pseudo-labels. This shows the benefits of using more pseudo-labels to better alleviate semantic ambiguity.

Main results. Table 4.2 compares our methods to multiple recent SPML baselines introduced in Section 4.3. We observe that our methods outperform all baselines across all metrics except mAP on EPIC-Kitchens-100-SPMV. As mentioned above, this metric is not well suited for imbalanced datasets. We note that in most cases the P+S loss outperforms the Mask loss. Given that the P+S loss treats pseudo-labels as positives, this demonstrates that our method for obtaining pseudo-labels is able to find good matches. We observe a clear boost especially in IOU and F_1-measure, which are well suited metrics for our multi-label setting. This indicates that the SPML baselines suffer from being penalized for correct (*i.e.*, related) classes. This also possibly suggests that SPML methods designed for partial-label settings struggle to work well when the source of the multi-label noise originates from semantic ambiguity.

Interestingly, the EM [6] loss, despite working well on static images, shows the worst IOU accuracy and F_1-measure. We hypothesize this is because the loss does not treat unknown labels as negative. This results in having no negative signal, which in turns gives overly confident predictions for most classes. Looking deeper into the model's output on EPIC-Kitchens-100-SPMV, the EM loss predicts 51.5 positive labels on average per video, whereas our mask method predicts on average 1.2 and P+S method predicts on average 1.8, with a confidence threshold of 0.5. Considering that our test videos have 2.4 positive labels on average, it is evident that the EM loss predicts *too many* false positives. Our methods instead effectively push confidence up only for the relevant labels, keeping confidence for irrelevant labels low.

Table 4.3 (bottom) reports an ablation study on Confusing-HMDB-102 with an ideal label search, *i.e.*, the retrieved pseudo-labels are intentionally chosen as the correct classes. Results show that it is theoretically possible to improve performance with perfect pseudo-labels, however we do not observe a dramatic improvement compared to the labels obtained through our visual neighbor search. Table 4.3 (top) illustrates another ablation experiment, where we obtain pseudo-labels from class co-occurrences retrieved on our EPIC-Kitchens-100-SPMV test set. In this case, pseudo-labels are formed globally by looking at the co-occurrence of our multi-verb annotations across the test set. Given a verb v_i we count how many times it was annotated together with any other verb v_j. If v_i and v_j occur at least half of the time together v_j will form the pseudo-label set for instances labeled with v_i. The large performance drop observed when using these co-occurrence-based global pseudo-labels suggests that the semantic ambiguity in our setting requires *instance-level* disambiguation rather than *class-level* pseudo-labels. This is because verbs can have subtly different meaning which can change according to the context. Clustering verbs using co-occurrences ignores this signal, and thus hurts performance.

End-to-end experiments. We additionally performed end-to-end training on the Confusing-HMDB-102 dataset, using optical flow images. Optical flow was extracted using the TV-L1 [105] algorithm. In this case the training set features and the pseudo labels were updated every 5 epochs. Table 4.4 reports results obtained with this setting. Performance with end-to-end training naturally improves compared to results reported in Table 4.2, where pre-extracted features were used throughout the training. We observe that our methods consistently outperform all baselines.

Dataset	Loss	Top-set ML	Top-1 ML	IOU Acc.	F_1	mAP
EPIC-Kitchens-100-SPMV	AN	43.7 ± 0.5	51.0 ± 0.4	11.2 ± 0.4	15.0 ± 0.6	22.8 ± 1.8
	WAN	44.8 ± 0.3	52.5 ± 0.6	15.2 ± 5.0	24.6 ± 6.6	**26.3 ± 1.3**
	LS	43.7 ± 0.9	51.8 ± 0.7	9.6 ± 0.4	12.9 ± 0.6	24.4 ± 1.5
	N-LS	43.4 ± 0.6	50.7 ± 0.4	11.0 ± 0.4	14.8 ± 0.5	22.1 ± 0.8
	Focal	43.3 ± 0.6	51.5 ± 0.3	5.7 ± 0.2	7.8 ± 0.2	23.9 ± 0.6
	EM	44.7 ± 0.4	52.9 ± 0.3	4.1 ± 0.0	7.8 ± 0.0	25.1 ± 1.0
	Mask (ours)	46.6 ± 0.2	55.2 ± 0.4	27.8 ± 0.4	36.9 ± 0.4	25.9 ± 0.8
	P+S (ours)	**46.9 ± 0.1**	**56.0 ± 0.6**	**33.5 ± 0.2**	**44.9 ± 0.3**	25.8 ± 0.8
Confusing-HMDB-102	AN	32.0 ± 1.6	38.5 ± 2.0	18.9 ± 1.6	24.8 ± 1.6	20.6 ± 3.8
	WAN	36.8 ± 0.7	40.7 ± 0.6	4.1 ± 0.1	7.9 ± 0.1	32.0 ± 1.4
	LS	32.4 ± 2.3	38.8 ± 1.8	19.3 ± 2.3	24.9 ± 2.3	19.7 ± 3.8
	N-LS	32.2 ± 1.3	38.8 ± 1.7	19.3 ± 1.7	25.3 ± 1.8	20.1 ± 3.7
	Focal	31.6 ± 1.9	38.0 ± 2.0	13.0 ± 0.8	17.6 ± 0.7	14.8 ± 3.8
	EM	31.9 ± 0.6	37.4 ± 0.9	3.2 ± 0.0	6.2 ± 0.1	18.6 ± 3.1
	Mask (ours)	41.8 ± 1.1	43.3 ± 0.9	**30.8 ± 2.5**	**36.3 ± 2.7**	40.3 ± 2.2
	P+S (ours)	**41.9 ± 0.9**	**43.4 ± 0.5**	29.9 ± 2.2	35.9 ± 2.0	**40.7 ± 2.0**

Table 4.2: Results with ± standard deviation calculated over five runs for EPIC-Kitchens-100-SPMV and three splits for Confusing-HMDB-102. For EPIC-Kitchens-100-SPMV the mAP metric shows unstable results due to the very long-tailed distribution of the verbs. In other words, many classes have only a few samples and mAP is a non-weighted average of AP computed per class, encouraging correct predictions on tail classes. Best and second-best results are bolded and underlined respectively.

Dataset	Loss	Top-set ML	Top-1 ML	IOU Acc.	F_1	mAP
EPIC-Kitchens-100-SPMV	Mask	46.6 ± 0.2	55.2 ± 0.4	27.8 ± 0.4	36.9 ± 0.4	**25.9 ± 0.8**
	Mask†	37.8 ± 1.3	48.4 ± 1.0	18.1 ± 0.4	23.7 ± 0.4	21.7 ± 0.9
	P+S	**46.9 ± 0.1**	**56.0 ± 0.6**	**33.5 ± 0.2**	**44.9 ± 0.3**	25.8 ± 0.8
	P+S†	23.0 ± 0.4	28.0 ± 0.5	12.4 ± 0.4	18.0 ± 0.6	20.8 ± 1.3
Confusing-HMDB-102	Mask	41.8 ± 1.1	43.3 ± 0.9	30.8 ± 2.5	**36.3 ± 2.7**	40.3 ± 2.2
	Mask*	42.6 ± 2.2	43.8 ± 2.3	30.4 ± 2.3	34.2 ± 2.4	37.4 ± 3.6
	P+S	41.9 ± 0.9	43.4 ± 0.5	29.9 ± 2.2	35.9 ± 2.0	**40.7 ± 2.0**
	P+S*	**43.2 ± 1.8**	**44.3 ± 2.1**	**31.4 ± 2.5**	35.9 ± 2.1	39.1 ± 3.1

Table 4.3: Ablation experiments using class-level pseudo labeling (†) or ideal pseudo-label search ($*$). We report results with ± standard deviation calculated over five runs for EPIC-Kitchens-100-SPMV and three splits for Confusing-HMDB-102. Best and second-best results are bolded and underlined.

Loss	Top-set ML	Top-1 ML	IOU Acc.	F_1	mAP
AN	46.7± 1.5	49.5± 1.7	11.5± 0.8	13.7± 1.1	47.8± 1.2
WAN	38.2± 2.4	40.9± 2.7	13.7± 0.9	23.0± 1.3	37.4± 2.2
LS	47.8± 0.8	51.0± 0.5	9.5± 1.2	11.4± 1.5	49.8± 0.8
N-LS	48.6± 1.3	51.1± 1.2	12.0± 2.2	14.2± 2.8	50.0± 2.1
EM	47.7± 1.7	48.5± 1.7	3.4± 0.1	6.5± 0.1	49.6± 2.8
Mask	50.4± 0.1	**53.2± 0.2**	25.6± 1.8	27.9± 2.0	51.9± 1.2
P+S	**50.5± 0.3**	51.8± 0.3	**30.9± 2.3**	**33.4± 2.6**	**52.4± 1.3**
Mask*	53.4± 0.9	54.7± 1.5	32.5± 0.8	34.3± 0.9	55.4± 1.1
P+S*	57.2± 0.6	57.3± 0.8	41.7± 2.1	42.7± 1.9	59.9± 1.3

Table 4.4: Results obtained training the model end-to-end with optical flow images. Results in percent (%) ± standard deviation on the three splits of Confusing-HMDB-102. Losses marked with $*$ assume the case when the pseudo label search is ideal, and gets to use actual ground truth labels during training. Mask and P+S (without $*$) are trained with $K = 10$ and $\tau = 0.2$.

Qualitative results. Figure 4.7 illustrates a few qualitative examples where we compare the predictions obtained with our methods and the other SPML baselines. Verbs shown here were selected by thresholding the model confidence (sigmoid) at 50%. A "-" indicates no predictions, which happens when all classes have low confidence. We note that WAN and EM tend to over-predict numerous verbs, whereas AN, LS and N-LS tend to predict only one verb. Our methods instead correctly output relevant verbs without over-predicting a large number of labels.

4.6 Conclusion

We proposed to treat action recognition as a single positive multi-label learning task, as labels can often be ambiguous and have overlapping semantics. We annotated a part of the large-scale EPIC-Kitchen-100 dataset to be used as a multi-label evaluation, and furthermore, we created a synthetically generated ambiguously-labeled dataset called Confusing-HMDB-102.

EPIC-Kitchens-100-SPMV. This is a multi-verb annotated version of the original EPIC-Kitchens-100 dataset for efficiently evaluating the proposed problem. We anno-

Figure 4.7: Qualitative examples from our methods and the other SPML baselines. White indicates the ground truth, yellow denotes a partial match, while red and green denote incorrect and successful total matches, respectively. We cap the model predictions shown here to five. A dash "–" denotes the case where the are no predictions due to the model having low confidence for all classes.

tated around 3,500 videos and each video was annotated by three annotators to obtain clean labels. There are 2.4 labels per video on average.

Confusing-HMDB-102. This is a dataset with increased label confusion by synthetically increasing the number of classes in HMDB-51 to make confusingly interchangeable labels. This is a simple yet efficient method to simulate our problem setting.

Metrics. Five different popular metrics for multi-label classification tasks were used for evaluating the methods: Top-Set ML, Top-1 ML, IOU accuracy, F1-Measure, and mAP. Some metrics focus on top predictions while others treat all positive predictions equally.

Performance. Extracting pseudo labels from within the training set annotations correctly models this type of ambiguity, showing a significant improvement in performance measured using the five different multi-label metrics. Two different loss functions were proposed that utilize the extracted pseudo-labels in different ways. The

Mask BCE loss treats pseudo labels to be ambiguous, thus prohibiting the classes from passing any kind of learning signal. The P+S BCE loss considers pseudo labels to be accurate and passes positive learning signals. It is a trade-off to decide whether the improved performance is worth the risk of instability of the learning.

Discussions. Without our multi-label dataset and only using the original EPIC-Kitchens annotations, we observed that many of the failure cases of the action model actually predict valid categories. That signifies faulty evaluation in current action recognition practices. Indeed, the top-1 accuracy on the original EPIC-Kitchens-100's validation set is: AN 64.7, WAN 65.2, LS 65.0, N-LS 64.6, Focal 64.8, EM 65.5, Mask 65.4, P+S 65.4. Using this metric, all baselines are close in performance. However using appropriate multi-label metrics, as in the chapter, we observe much wider gaps. Based on the fact that our multi-label dataset has 2.4 labels per video on average, it became clear that a lot of the labels overlap in meaning, however, they depend heavily on the context and they cannot be simply grouped as one class.

Based on the baseline and initial ablation results, we found out that the standard SPML losses or label correction approaches primarily designed for images do not work well and there is a big room for improvement. It was a crucial step to understand how they are designed. They either try to "correct" the label between commonly mistaken classes, or they try to boost classes based on model predictions. What worked well was to model the confusing label space by looking at other video annotations. Interestingly, with our losses, it was not even necessary to obtain perfect modeling of this space, and sometimes the perfect pseudo labels can even hurt the performance. It is potentially possible to improve the quality of the pseudo labels as a future work, but new loss functions or optimization frameworks need to be introduced along with that.

We found out that the top 10 predictions of the pre-trained model cover most of the valid labels, and our annotations have no huge bias towards the model failures in the end based on the Figure 4.6. It is highly recommended to keep the procedure efficient like this for any future multi-verb annotation tasks.

Limitations. The chapter lacks full end-to-end experiments and experiments with different action models. We leave those as future work.

Chapter 5

Adversarial Augmentation Training for Video Distribution Shifts

5.1 Introduction

Video action recognition has become a vital computer vision task with applications in surveillance, robotics, and more. Video data exhibits greater diversity than image data, and therefore action recognition architectures are not as robust to distribution shifts [69, 70, 106]. In addition to image-level effects like viewpoint and appearance changes, video introduces effects such as camera motion, focus shifts, and background object movements. Moreover, an action class incorporates substantial intra-class variation as illustrated in Figure 5.1. For example, the class "playing basketball" may involve dribbling, running, or shooting in different contexts. Furthermore, depending on the data source, there are biased video processing artifacts. For example, videos collected from YouTube have standardized YouTube processing (VP9 compression), making the dark areas have extremely low quality. Often, the videos go through a frame rate conversion algorithm, which can create frame interlacing artifacts. As a result, the slight distribution shift in video data can dramatically reduce the action recognition performance.

Data augmentation is one potential solution to account for this fragility. It is a popular method to create synthetic variations of the existing training data that will enable classification models to generalize better to previously unseen test data. However, it is not yet clear what kind of augmentation is necessary to generalize to test data with different distribution shifts. There has been much work on automatically selecting augmentation policies given training and validation data [107, 108, 109, 110, 111].

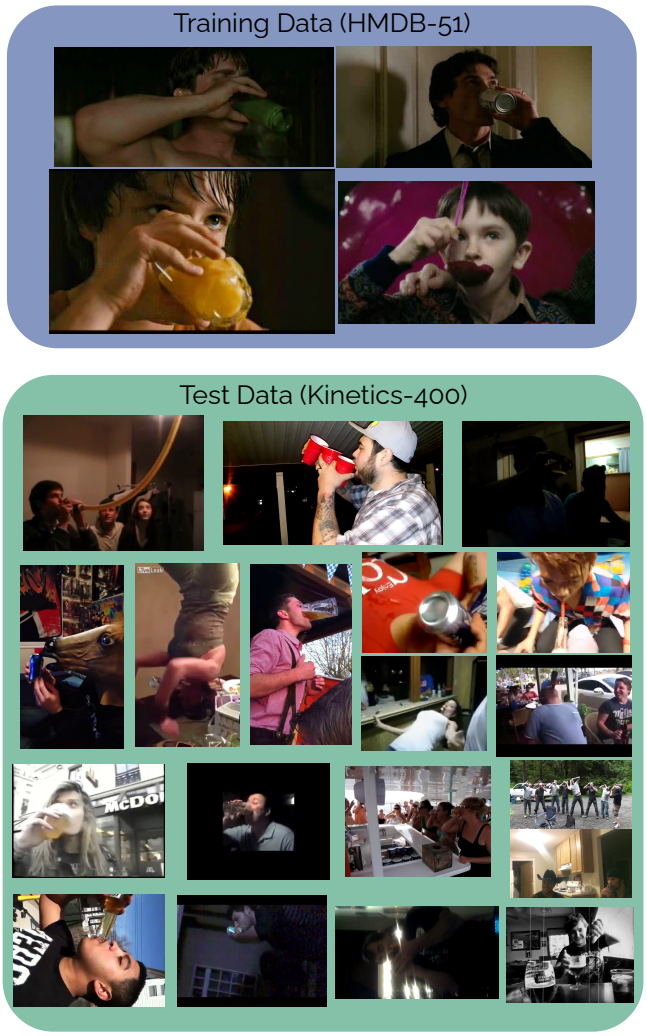


Figure 5.1: A common scenario with biased training data and testing with real-life data, with an example class of “drink”. The training data is from the HMDB-51 dataset whose video samples are usually taken from movies, and the test data is from the Kinetics-400 dataset which is from YouTube videos. There are many reasons why the test data looks so different: poor camera quality, wrong orientation, extreme camera shake, inconsistent frame rate, frame rate conversion artifacts (interlacing), poor lighting, lack of professional post-processing (*e.g.*, color grading), different ways of performing the action, poor framing, inconsistent aspect ratios, editing artifacts, various actions happening at the same time. Thus, it is common for the performance to drop significantly when the trained model is applied in more general applications.

However, such approaches optimize augmentation of the training data, and it is not clear whether the resulting generalization applies to test data with much distribution shift.

To address the video domain shift problem, we propose an adversarial augmentation scheme that generates "hard" video examples for the action recognition networks. The pipeline is simple to implement, and results in a meaningful improvement in performance on the proposed datasets with realistic distribution shifts compared to no augmentation and random augmentation baselines. The benefits are demonstrated using three popular action recognition architectures. The training and validation datasets are subsets of HMDB-51 or UCF-101, and the test data are a subset from equivalent Kinetics-400 classes, to realistically evaluate distribution shifts over the same action classes.

This approach and evaluation requires multiple datasets with common aligned action classes. The chapter also presents another simple method to evaluate robustness using a target dataset with different classes using cosine similarity of features as logits. The method requires no training (*i.e.*, fine-tuning) on the target dataset, and thus it correctly measures the transferability of the trained classifier on the target dataset.

5.2 Related Work

Data augmentation. [112] summarizes image data augmentation techniques for deep learning. AutoAugment [107] is an augmentation policy search algorithm that finds the best augmentation on a target dataset, based on the highest validation accuracy.

Due to AutoAugment's expensive policy search, Population-Based Augmentation [108], FastAutoAugment [109], and FasterAutoAugment [110] proposed more efficient searching algorithms, by learning a schedule policy over a fixed-policy, using density matching, and using differential augmentation with a generative adversarial network (GAN) [113] architecture that involves a policy generator and a discriminator, respectively.

Differentiable Automatic Data Augmentation [111] proposed differential data augmentation policy searching algorithm using Relaxed Bernoulli distribution [114] which is differentiable, similar to FasterAutoaugment, and further introduced an unbiased gradient estimator that enables joint optimization of the augmentation policies and network parameters, instead of using GAN.

RandAugment [115] showed that simple random augmentations with randomly sampled transformations achieve similar performance efficiently.

However, the policies in most work are optimized on the training set, and we focus on the scenario where test data can have severe distribution disparity which is unknown

during the training time.

Adversarial training. Different than conventional training with data augmentation, adversarial training (AT) is defined as a min-max problem whereby the trained model uses observed training samples to minimize its prediction error, while an adversary attempts to generate training samples that maximize it.

While it is well-established that AT is the most effective way to achieve adversarial robustness [116], it has further been shown to yield other types of robustness, *e.g.*, against natural corruptions [117], domain shift [118, 119, 120], and others. Note that even though the classical definition of AT uses adversarial input noise [121, 122], more adversarial image augmentations have been studied, *e.g.*, rotations [123, 124], contrast, jitter [125].

AT should be approached with care, as generating adversarial training examples that are too challenging for the trained model may actually harm downstream performance [126].

In this chapter, we employ two measures to control this trade-off: (i) We create maximally informative adversarial examples (confusing to the model, but near the classification boundaries) via maximum-entropy regularization, as per the work of [127, 128, 129]. (ii) We train with curriculum AT as per the work of [126, 130], which involves training with harder adversarial examples over time.

Domain adaptation. Domain adaptation is a transfer learning task where the source and target datasets have a significant distribution shift while sharing the same task. [131] explains types of domain adaptation tasks and approaches.

There are discrepancy-based techniques that learn transferable features from a source domain to a target domain [132, 133], reconstruction-based methods that utilize autoencoders, which aims to extract a useful feature for the target domain [134, 135], and adversarial domain adaptation approaches involving a source/target discriminator that distinguishes where the data come from and a feature extractor that aims to confuse the discriminator by trying to produce generic features regardless of the domain [136, 137, 138, 139, 140, 141, 142, 143]. More recently, analyzing frequency components of deep feature maps using attention to filter domain-general components [144] is proposed.

Domain adaptation for video action recognition is first proposed by using a feature alignment approach on online test videos [106]. This work is evaluated using com-

putationally simulated corrupted videos, while we propose to use real examples that involve more diverse types of discrepancy between the domains.

It is important to note that most domain adaptation techniques require examples from the target dataset to be present, while our work focuses on evaluating using an utterly unknown dataset.

5.3 Problem and Methodology

Action recognition predicts an action category label given a video sequence. This chapter explores how well different variations of action recognition models, training, and loss functions generalize by evaluating on a different data domain.

The main difference with transfer learning is that our approach does not tune model parameters using the target evaluation dataset, whereas transfer learning usually involves fine-tuning the model with the target dataset's training set.

To improve generalizability, the training data is augmented. Hard-to-classify adversarial examples are generated by applying gradient ascent on the augmentation parameters which are fully differentiable. We then train the classifier using AT loss, calculated using both clean and adversarial examples.

5.3.1 Adversarial Augmentation Training

Adversarial augmentation training uses a two stage training loop, as described in Figure 5.2.

Stage 1: Generate adversarial examples. "Hard" adversarial examples are found by tuning the augmentation parameters using gradient ascent.

Let $g_\theta(\mathbf{x})$ be an augmentation model with fully differentiable parameters θ, and $f_\phi(\mathbf{x})$ be a video classification model with parameters ϕ, that outputs class predictions given an input video $\mathbf{x}$. The goal of stage 1 is to find the augmentation parameters θ' that maximize categorical cross-entropy loss. Note that by *maximizing* the loss, we aim to find the augmentation strategy that is challenging for the classifier, and in Figure 5.2, this is described as *minimizing* the negative cross-entropy loss.

At each training step, the augmentation parameters θ are randomly initialized. Gradient ascent is then done only on θ, freezing the classification parameters ϕ to learn

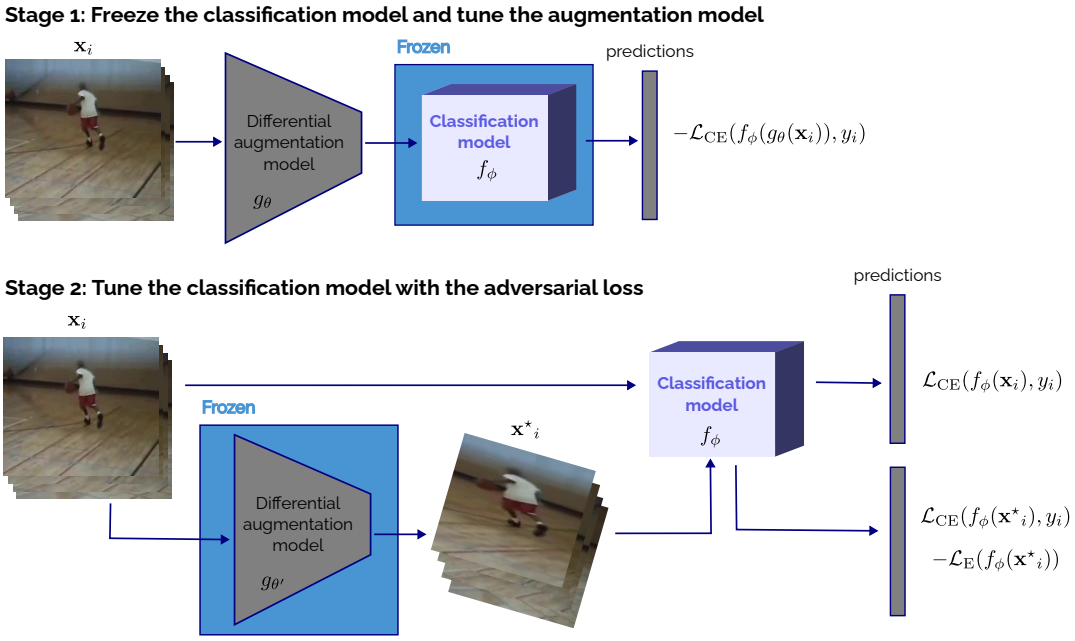


Figure 5.2: The proposed adversarial augmentation training has two separate stages. Firstly, the classification model is frozen while the differential augmentation model is trained using the negative cross-entropy loss. This is equivalent to performing gradient ascent to maximize normal cross-entropy loss. As a result, the augmentation model will generate hard augmentations for the classification model. The second stage trains the classification model using both clean and adversarial examples. The maximum entropy regularization loss is integrated by subtracting the entropy of the adversarial examples, encouraging the predictions to be evenly balanced.

adversarial augmentations, with the cross-entropy loss $\mathcal{L}_{\mathrm{CE}}(f_\phi(g_\theta(\mathbf{x}_i)), y_i)$. This optimizes augmentation parameters θ' for generating adversarial examples.

Stage 2: Optimize the classification model. Next, the classification parameters ϕ are optimized while freezing the augmentation parameters θ. For simplicity, $\mathbf{x}_i^\star = g_{\theta'}(\mathbf{x}_i)$ denotes the generated adversarial example of $\mathbf{x}_i$. The vanilla AT loss is defined as:

$$\begin{aligned}\mathcal{L}_{\mathrm{AT}}(f_\phi(\mathbf{x}_i), f_\phi(\mathbf{x}_i^\star), y_i) = &\alpha \mathcal{L}_{\mathrm{CE}}(f_\phi(\mathbf{x}_i), y_i) \\ &+ (1-\alpha)\mathcal{L}_{\mathrm{CE}}(f_\phi(\mathbf{x}^\star{}_i), y_i)\end{aligned} \tag{5.1}$$

which is a weighted average of the cross-entropy loss using the clean sample and the augmented sample.

Max-Entropy Regularization. The cross-entropy loss encourages predictions to be over-confident by pushing the examples further from the classification boundaries. However, adversarial examples are supposed to be confusing. We regularize the cross-entropy-based adversarial loss in Eq. (5.1) by maximizing the entropy, encouraging the overall predictions to be evenly balanced for adversarial examples.

$$\begin{aligned}\mathcal{L}_{\text{AT-ME}}(f_\phi(\mathbf{x}_i), f_\phi(\mathbf{x}_i^\star), y_i) =& \alpha \mathcal{L}_{\text{CE}}(f_\phi(\mathbf{x}_i), y_i) \\ &+ (1-\alpha)\mathcal{L}_{\text{CE}}(f_\phi(\mathbf{x}^\star{}_i), y_i) \\ &- \gamma \mathcal{L}_{\text{E}}(f_\phi(\mathbf{x}^\star{}_i))\end{aligned} \tag{5.2}$$

where entropy loss $\mathcal{L}_{\text{E}}$ is defined as

$$\mathcal{L}_{\text{E}}(f_\phi(\mathbf{x}_i^\star)) = -\frac{1}{C}\sum_{c=1}^{C} f_\phi^{(c)}(\mathbf{x}_i^\star)\log(f_\phi^{(c)}(\mathbf{x}_i^\star)) \tag{5.3}$$

C is the number of classes, and $f_\phi^{(c)}(\mathbf{x}_i^\star)$ is the prediction score of class c for the adversarial example $\mathbf{x}_i^\star$.

Curriculum Adversarial Training. Applying curriculum training by starting from training with "easy" samples and gradually generating "harder" samples makes the model more robust [126]. Here, the classification models are trained initially from clean data without augmentation, and gradually harder adversarial examples are added by scheduling the learning rate of the augmentation model.

5.3.2 Cross-Dataset Evaluation

We train on one dataset and test on another dataset, where there are distribution shifts between the train and the test data. We propose two different evaluation approaches.

Matched Class Evaluation (Expt. 1). The first approach creates two datasets that share the same classes, but which have a significant disparity in the train and test data distribution. To choose the common classes, visual features are extracted on two existing datasets using the I3D [22] model pre-trained on Kinetics-400 [47] and semantic cues are extracted using the sen2vec [145] model pre-trained on Wikipedia. The visual and semantic similarity features are combined and then used in the TruZe [146]

procedure (*i.e.*, include exact matches, matches that can be either superset or subset, and matches that predict the same visual and semantic matches) to obtain a set of extremely similar classes from the two source action recognition datasets. The matched classes from the two datasets are verified manually and a subset is selected that has a substantial overlap in visual and semantic cues. One dataset is used for training and validation, and the other dataset is used for testing. We describe the curated datasets in Section 5.4. Table 5.3 shows some of the matched classes. This is the most realistic method to evaluate on a distribution shift, but it requires some manual procedures as well as finding datasets that share largely similar classes.

Cosine Similarity Evaluation (Expt. 2). The second method uses two datasets that have different classes without modifying them or fine-tuning the model on the target dataset for transfer learning. It is a simpler method compared to the first one and does not require manual class matching.

We first train the classification model using the source dataset. Since the number and types of classes of the training (source) and evaluation (target) datasets are different, we detach the last classification layer of the source dataset classification model to use it as a feature extractor. We then calculate class prototypes of all the classes in the target dataset by simply averaging the features of each class in the training set, inspired by [147]. The prediction score of a sample from the target dataset's test set is estimated using the cosine similarity of their feature vector and the class prototypes, followed by softmax. In other words, the cosine similarity is used as logits, which are formed by producing high activations on classes that are closely aligned to the training set of the target dataset. Using these estimated prediction probabilities, we report accuracy. We will show that adversarial augmentation of the source dataset also produces improved classification of the target dataset, by producing a more robust feature extractor for use with this similarity measure. Note that the source dataset is never used for evaluation, and only be used for training the model, and the target dataset's training set does not contribute to fine-tuning the model. The role of all four splits in two different datasets can be found in the Table 5.1.

Formally, a class prototype $\mathbf{c}_k \in \mathbb{R}^M$ is an M-dimensional representation. The prototype for each class k in the target dataset is the mean vector of the embedded features belonging to its class in the training set. Let S_k be the set of videos in the training set

Dataset Type	# Classes	Split	Usage
Source	N	train	Training
		test	Validation
Target	K	train	Obtaining K class prototypes
		test	Testing with cosine similarity as logits

Table 5.1: Purpose of all dataset splits for the cosine similarity evaluation method (Expt. 2). Given the four splits of the data, the source dataset is used to train (with or without AT) and validate the model. The target dataset is used to evaluate the model without further fine-tuning.

in the target dataset from class k. $\mathbf{c}_k$ is computed by:

$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{(\mathbf{x}_i, y_i) \in S_k} h_\omega(\mathbf{x}_i) \tag{5.4}$$

where $h_\omega()$ is an embedding function, which is the feature extraction model that shares parameters partially with f_ϕ excluding its classification layer.

For a given test video from the target dataset $\mathbf{x}_i \in S_k^{test}$, the probability of a given class label k is estimated using the cosine similarity of the embedding to the target dataset's class prototype $\mathbf{c}_k$, followed by softmax. Given the cosine similarity function $d(\mathbf{x}, \mathbf{c}) = \frac{\mathbf{x} \cdot \mathbf{c}}{\|\mathbf{x}\| \|\mathbf{c}\|}$ we get:

$$P(y = k | \mathbf{x}_i \in S_k^{test}) = \frac{\exp(-d(h_\omega(\mathbf{x}_i), \mathbf{c}_k))}{\sum_{k'} \exp(-d(h_\omega(\mathbf{x}_i), \mathbf{c}_{k'}))} \tag{5.5}$$

where y denotes the class label.

The parameter ω is not tuned during this operation. That is, the training data from the target dataset does not contribute to fine-tuning the model. The motivation of this approach is to measure the transferability of the model from a source dataset to a target dataset without actually tuning the model parameters, showing the robustness of the model to sample distribution shift.

5.4 Experiments

HMDB/Kinetics and UCF/Kinetics Datasets (Expt. 1). Training datasets are created from subsets of HMDB-51 or UCF-101 and the subset of the Kinetics-400 test set is used for testing. The subsets share the same classes between the training and

test sets. The motivation for this approach to creating the datasets is to simulate a real-world environment where the test data comes from many unknown sources, with many variations in actions, capture conditions, aspect ratio, and so on. The Kinetics-400 dataset has many more samples in the fine-grained classes, so it was used for testing.

TruZe [146] is used to identify shared classes from the HMDB-51 and Kinetics-400, and UCF-101 and Kinetics-400 datasets, based on the visual and semantic similarity. More details on this procedure can be found in Section 5.3.2. In TruZe, normally, classes from UCF or HMDB that have overlapping context with the corresponding Kinetics class are removed, so that using the Kinetics pre-trained models would not bias the zero-shot settings. However, in our robustness problem, the train and test datasets are created with the *opposite* goal, where only overlapping classes are selected. For instance, the class "climb" in HMDB-51 is treated as the same class as "rock climbing", "ice climbing", "climbing a rope", and "climbing tree" in Kinetics-400. The final datasets are named HMDB-28/Kinetics-28 and UCF-65/Kinetics-65, where the training sets are subsets of the HMDB and UCF data, respectively, and test subsets come from Kinetics. The 28 and 65 refer to the number of shared classes. Examples of the resulting splits can be found in Table 5.3. The three official published splits of HMDB-51 and UCF-101 are used, but only the shared classes are selected. The results in Table 5.4 are the average performance over the three splits.

HMDB $\leftrightarrow$ UCF Evaluation (Expt. 2). For the experiments using the cosine similarity function, the HMDB-28 trained models are tested on UCF-101, and the UCF-65 trained models are tested on HMDB-51, using the cosine similarity measure with class prototypes as predictions. The results in Table 5.4 are the average performance over the nine splits, three runs from the source dataset and each run evaluates with three splits from the target dataset.

Table 5.2 summarizes all datasets used for the experiments (Expt. 1 and Expt. 2).

Augmentation Methods. Results are compared for four different training approaches: training without augmentation, with random augmentation, with adversarial augmentation, and with curriculum AT [126] as described in Section 5.3.1. Experiments used the popular efficient 2D TSM model [10] with a ResNet50 backbone, Video Swin Transformer [2] Tiny, and Uniformer-S [12] model. ImageNet pre-trained models were used instead of Kinetics pre-trained, so that the models never get to see the Kinetics data distribution.

Expt	Name	Original Dataset	Usage
1a	UCF-65 Kinetics-65	UCF-101 Kinetics-400	Training, validation Testing UCF-65 trained models
1b	HMDB-28 Kinetics-28	HMDB-51 Kinetics-400	Training, validation Testing HMDB-28 trained models
2a	UCF-65 HMDB-51	UCF-101 (original)	Training, validation Testing UCF-65 trained models w/ cosine similarity
2b	HMDB-28 UCF-101	HMDB-51 (original)	Training, validation Testing HMDB-28 trained models w/ cosine similarity

Table 5.2: List of datasets used for our experiments. The HMDB-51, UCF-101, and Kinetics-400 are the original datasets. The UCF-65, Kinetics-65, HMDB-28, and Kinetics-28 are the proposed subsets.

HMDB-51 Classes	Kinetics-400 Classes
brush hair	curling hair, dying hair, brushing hair
cartwheel	cartwheeling, somersaulting
catch, throw	shooting goal, juggling, catching or throwing frisbee, catching or throwing baseball, catching or throwing softball, throwing axe, throwing ball, throwing discus
clap	clapping, applauding
climb	rock climbing, ice climbing, climbing tree, climbing a rope
dive	diving cliff, bungee jumping
dribble	dribbling basketball
drink	drinking beer, drinking shots, drinking
eat	eating burger, eating cake, eating carrots, eating chips, eating doughnuts, eating hotdog, eating ice cream, eating spaghetti, eating watermelon
golf	golf driving, golf chipping, golf putting
...	...

UCF-101 Classes	Kinetics-400 Classes
Basketball	shooting basketball, dribbling basketball, playing basketball
BasketballDunk	dunking basketball
BodyWeightSquats	lunge, squat, dead lifting
BreastStroke	swimming breast stroke, swimming back stroke
CleanAndJerk	clean and jerk, snatch weight lifting
CliffDiving	diving cliff, springboard diving
Haircut	shaving head, braiding hair, getting a haircut
HorseRiding	riding or walking with horse, riding mule
PlayingPiano	playing piano, playing organ
Skiing	skiing (not slalom or crosscountry), skiing crosscountry, skiing slalom
...	...

Table 5.3: The HMDB-28/Kinetics-28 (above) and UCF-65/Kinetics-65 (below) datasets which are subsets from the original HMDB-51 UCF-101, Kinetics-400 datasets. Visually and semantically similar classes were combined which allows testing with real-life data from YouTube that has a significant distribution shift from the training data. The HMDB or UCF data are used for training and validation, and the Kinetics data are used for testing.

Augmentation used translation to a maximum of 28 pixels, 10° rotation, shear transform of 0.1, and scale from 0.9 to 1.5. For curriculum training, no augmentation was used for 20 epochs, then AT with a zero learning rate of the augmentation model was used for 20 more epochs. Then, AT with a triangular learning rate scheduling

from 0.1 to 1.0 on the augmentation model was used for the rest of the training, except on the Uniformer model. For Uniformer, the above was done for only 20 epochs, and then trained with random augmentation for 20 more epochs and fine-tuning with no augmentation for the rest to mitigate under-fitting issues.

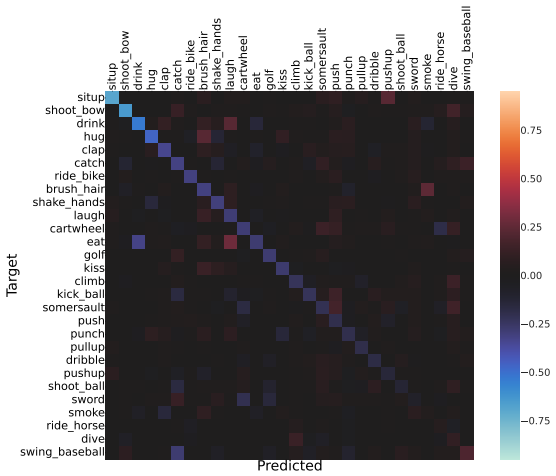
Implementation Details. Videos were resized to 224 $\times$ 224 and sampled to 8 frames sparsely similar to [3]. The classifier models were trained for 200 epochs using an SGD optimizer with an initial learning rate of 0.0001, decaying the learning rate using cosine annealing scheduling. For adversarial training (AT), the augmentation model used a learning rate of 0.1. For adversarial plus curriculum training, the $\mathcal{L}_{\text{AT-ME}}$ loss was used with $\alpha = 0.5$ and $\gamma = 0.5$. Two NVIDIA RTX 3090 GPUs were used with batch size 16 to train the TSM models, an NVIDIA A100 GPU with batch size 16 and 32 for training the Uniformer and Swin Transformer models, respectively.

5.5 Results

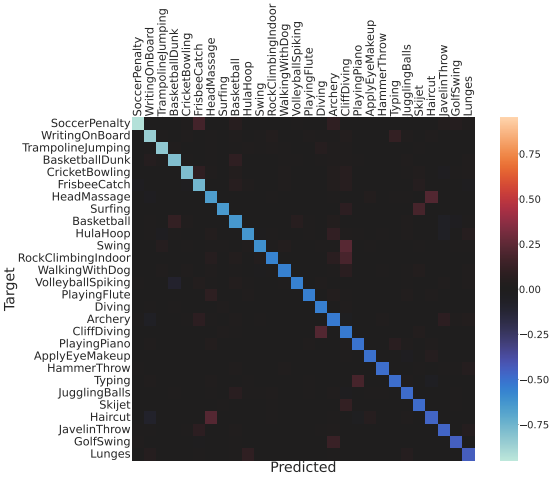
5.5.1 Shared Class Experiments 1a, 1b

Two confusion matrices are created collating the performance when training on HMDB-28 and testing on either HMDB-28 or Kinetics-28 (test sets). The former is subtracted from the latter to record how much difference there is in the performance, which is shown in Figure 5.3(a). (The figure also shows the same results when using UCF-65 and Kinetics-65.) In this figure, no augmentation strategy is used. The figure shows that there is a dramatic performance drop when the models are tested on a different dataset (Kinetics) to training (HMDB or UCF). Some categories such as "situp", "shoot_bow", and "drink" are damaged more severely. As seen in Figure 5.1, HMDB/UCF videos have objects and actors that are clearly visible and stable in the frame, while Kinetics videos tend to have lots of camera motion with different sizes of the objects. The overall accuracy drop from UCF-65 to Kinetics-65 is even more significant. These results show that the raw trained model is not robust to distribution shifts between datasets, which is not ideal for deployment in real applications.

When the adversarial training approaches presented in Section 5.3.1 are applied, target dataset performance improves. Table 5.4 summarizes the main cross-dataset evaluation results for the four augmentation strategies presented in Section 5.3.1. The different adversarial augmentation strategies gave improved accuracy for the target



(a) HMDB-28



(b) UCF-65 (top 28 classes with the highest accuracy drop)

Figure 5.3: Confusion matrix difference between evaluating on the same dataset (a:HMDB at top or b:UCF at bottom) and a different one (Kinetics), given a TSM model with no augmentation strategy. The negative (blue) values on the diagonal line indicate the class accuracy drop when evaluated on Kinetics. Almost every class has a drastic drop. Overall, there is a 25.7% and 38.37% accuracy drop for the HMDB and UCF training datasets, respectively.

datasets (see Kinetics-28 and 65 results columns).

Also, unlike what is reported in [70], the convolutional architecture performed better with the distribution shift compared to transformer models in most of the cases except for the Swin Transformer trained on UCF-65 and tested on HMDB-51. We hypothesize that this is because the Kinetics test dataset shows natural and realistic distribution shifts. In addition, we did not use the Kinetics pre-trained models, and the

Model	Train Dataset	Train Augmentation	Test Accuracy	
			Kinetics-65 Matched Class Expt 1a	HMDB-51 Cosine Expt 2a
TSM	UCF-65	None	39.13 ± 0.56	37.61 ± 1.88
		Random	40.90 ± 0.32	38.19 ± 1.39
		Adversarial	42.42 ± 0.63	38.99 ± 1.26
		Curriculum	**42.51 ± 0.62**	**39.14 ± 1.21**
Swin	UCF-65	None	37.08± 1.43	38.91 ± 1.63
		Random	40.80 ± 1.90	40.61 ± 1.40
		Adversarial	**42.27 ± 0.24**	41.48 ± 1.58
		Curriculum	41.20 ± 0.72	**41.58 ± 1.63**
Uniformer	UCF-65	None	18.78 ± 0.22	21.39 ± 1.32
		Random	22.42 ± 0.98	24.92 ± 1.66
		Adversarial	22.95 ± 0.68	**26.16 ± 2.10**
		Curriculum	**23.61 ± 0.27**	24.93 ± 1.60

Model	Train Dataset	Train Augmentation	Test Accuracy	
			Kinetics-28 Matched Class Expt 1b	UCF-101 Cosine Expt 2b
TSM	HMDB-28	None	29.75 ± 1.08	60.51 ± 0.79
		Random	30.13 ± 0.42	61.21 ± 0.57
		Adversarial	32.44 ± 0.48	62.60 ± 0.59
		Curriculum	**32.82 ± 1.41**	**63.18 ± 0.71**
Swin	HMDB-28	None	25.26± 0.80	59.88 ± 0.55
		Random	26.63 ± 0.97	60.99 ± 1.30
		Adversarial	27.31 ± 0.57	62.57 ± 0.72
		Curriculum	**27.60 ± 0.62**	**62.95 ± 0.82**
Uniformer	HMDB-28	None	14.53 ± 0.51	28.23 ± 0.94
		Random	14.33 ± 1.03	28.67 ± 1.36
		Adversarial	15.13 ± 0.79	29.88 ± 2.62
		Curriculum	**15.25 ± 0.75**	**30.83 ± 1.04**

Table 5.4: Results from three models (TSM ResNet50, Video Swin Transformer Tiny, and Uniformer-S), two training datasets (UCF-65 and HMDB-28), four augmentation strategies (no augmentation, random, adversarial, and curriculum), and two test datasets (Kinetics, and HMDB/UCF). In all cases, adversarial augmentation or curriculum adversarial augmentation training outperformed all baselines.

transformer architectures require large-scale data to reach the maximum potential.

In all cases, the adversarial augmentation or curriculum methods outperform all baselines, given a fixed network architecture, for all training and test datasets. Al-

though the "random augmentation" and "adversarial augmentation" allows an identical range of transforms, generating adversarial examples through gradient ascent produces "harder than random" augmentation which improved the overall performance. Furthermore, adding the simple curriculum mostly improved over the adversarial benchmark.

To demonstrate the improvement in transfer performance by the use of the curriculum adversarial strategy, we computed the difference of the confusion matrices for None and Curriculum augmentation. Figure 5.4 shows the results, where the reddish boxes on the diagonal indicate improved performance. It is clear that the proposed adversarial augmentation makes the model more robust on the classes with the larger distribution shifts, as presented in Figure 5.3. This supports our claim that the proposed adversarial augmentation makes the model more robust on classes with a huge distribution shift.

5.5.2 Cosine Similarity Evaluation 2a, 2b

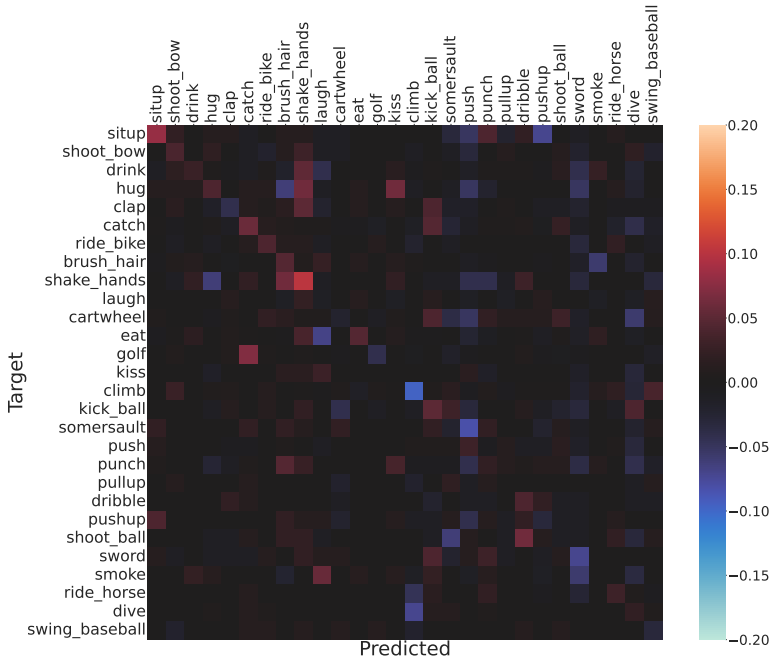
The cosine-similarity-based accuracy results on HMDB-51 and UCF-101 are shown in the Cosine Expt columns of Table 5.4. The results follow a very similar trend to the "realistic" Kinetics Expt 1. The advantage of using this accuracy measure as compared to testing on Kinetics with overlapping classes is that it requires no thorough analysis of the source and target datasets to find overlapping classes, making it simple to set up the cross-dataset experiments even using new datasets.

5.5.3 Adversarial Augmentation Examples

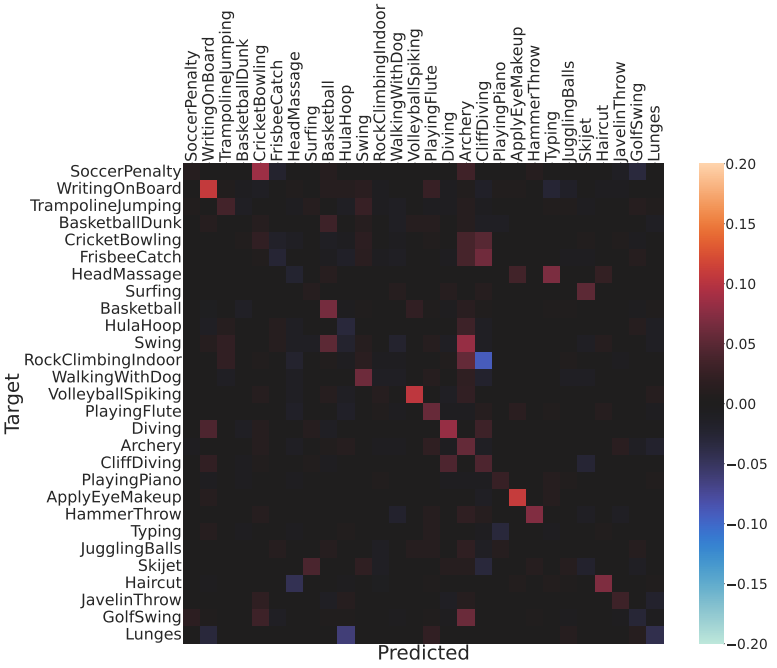
Some examples of adversarial augmentations in comparison to random augmentations are depicted in Figure 5.5. One might think that the most extreme and unrealistic augmentations will be challenging to the classifier. However, adversarial augmentation can sometimes render more realistic yet challenging examples.

5.6 Conclusion

One of the biggest reasons why it is hard to deploy action recognition models in real applications is because of the sheer amount of variation in real data that is unseen during training. We suggested new ways to evaluate action recognition performance over distribution shifts, to test the actual robustness of the model. Adversarial training and



(a) HMDB-28



(b) UCF-65 (top 28 classes with the highest accuracy drop)

Figure 5.4: Confusion matrix difference between no augmentation strategy and the proposed curriculum adversarial augmentation training with the TSM model and evaluation on Kinetics. The positive (red) values on the diagonal line indicate the added class-wise accuracy by using the proposed approach. The classes are sorted by the drop in performance using the proposed cross-dataset evaluation, as seen in Figure 5.3.

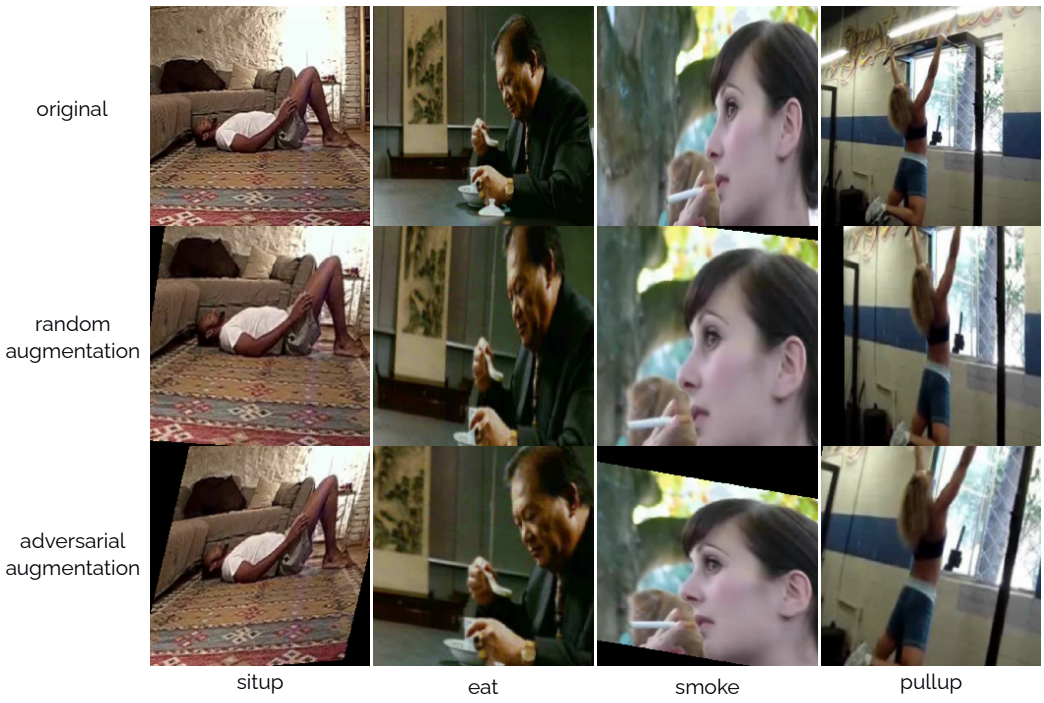


Figure 5.5: Examples of the vanilla and proposed augmentation examples in the HMDB-51 dataset while training the Video Swin Transformer model.

curriculum adversarial training benchmarks showed the robustness of three popular action recognition models. To summarize:

Cross-Dataset Evaluation Matching Classes in Two Datasets. Two new datasets were created by taking overlapping class subsets of HMDB-51 and Kinetics-400, and UCF-101 and Kinetics-400. HMDB or UCF data were used for training, and the Kinetics data was used for testing. We matched the overlapping classes by analyzing the visual and semantic similarity of the classes. This method required no training on the target Kinetics dataset. This method showed that a slight distribution shift in video data dramatically lowers the performance.

Cross-Dataset Evaluation using Cosine Similarity as Logits. Another way to evaluate a trained model on another dataset is to use cosine similarity scores between class prototypes and test video features as predictions. This method also does not require training on the target dataset, nor matching the classes manually, thus is a very effective method to evaluate model performance with data disparity. The experiments showed that results using this method align very closely with the matched-class results, suggesting that it is a very useful evaluation method. However, it is still advised to use

a dataset that contains similar actions to the original training dataset.

Adversarial Augmentation Training. We proposed a novel augmentation strategy using adversarial examples. We first tuned the differential augmentation model by applying gradient ascent, creating hard examples for the classifier model. We then trained the classifier model with both clean and adversarial examples. We also apply "curriculum training" to this procedure, creating harder examples over time. This method improves the robustness of the action recognition models to huge distribution shifts.

Discussion. Many augmentation studies about corruption analysis, as discussed in Section 2.3.7, did not propose a solution to the problem of improving generalization with the presence of severe distribution shifts during testing. This is because they all used algorithms that add known corruptions to the testing data which is an inappropriate methodology when the same corruptions have been applied to the test data. Image/video corruption augmentations are very important especially for research, however, are less practical because in most applications it is desired to use known data augmentation algorithms for training. Based on this idea, we proposed a new data augmentation strategy using real test data. The challenge was to analyze the results. As we were using real test data, it was not clear how different training strategies affected the performance. It was crucial to view the training and test examples to understand how they are different, and how much the performance on each class changed. Adversarial augmentation training can improve performance, but it has to be used with care, as discussed in the previous section. It is advised to use curriculum methods, or apply more advanced adversarial scheduling methods (which has to be investigated as a future work).

Limitations. The matching class evaluation method requires manual procedures, limiting the choice of datasets. The cosine similarity-based method does not require this, however, it still needs two datasets that share relatively similar actions. In addition, the adversarial augmentation training pipeline uses the same transformation on all the images in a video, which could be improved by using varying transforms over time.

Future Work. More adversarial training methods, more transformations (corruptions), and more augmentation strategies could be compared with the current experi-

ments. We leave those as future work.

Chapter 6

Conclusion

6.1 Summary

In the first work, we presented two novel video channel sampling strategies: TC Reordering and GrayST. The former re-orders RGB channels to increase temporal information, and the latter uses grayscale images to use more frames resulting in an increased temporal receptive field. In spite of the simplicity of our approaches, we observe a significant boost in performance on multiple challenging datasets. Importantly, these sampling strategies allow us to significantly increase the performance of existing lightweight video networks without increasing the computational cost or requiring any modifications to the underlying network architectures. Our hope is that this will pave the way for alternative sampling strategies. Future work includes developing a temporal aggregation module that is compatible with our sampling strategies for reasoning over much longer time scales.

In the next chapter, we addressed the problem of action label ambiguity at training time in video action recognition. Here, videos can contain multiple equivalent labels due to the semantic ambiguity of verbs. This is a distinct problem compared to most partial-label settings, where multiple actions are present but not all are annotated. We showed that state-of-the-art SPML approaches struggle in the presence of this semantic ambiguity. To address this, we proposed two pseudo-labeled-based losses: *Mask BCE*, which treats pseudo-labels neutrally and *Pseudo+Single-label BCE*, which trusts pseudo-labels and uses them as positive labels. Through results on one dataset with synthetically generated ambiguity and another consisting of a new set of annotations for EPIC-Kitchens, we show that our methods outperform current SPML methods. One interesting direction for future research is addressing the additional ambiguity

resulting from temporally overlapping actions that are often present in dense video datasets.

For our final work, we addressed the problem of model generalization to realistic test distribution shift. Two new datasets that are comprised of three existing datasets were created that shared the same subset of label classes. Although the same classes were used, the variety of videos in the original dataset sources meant that there was a huge distribution shift from the source to the target datasets. When using the target datasets, action recognition performance dropped significantly. This led to trying adversarial augmentation, with and without curriculum scheduling, as an approach to generating hard adversarially augmented videos. This approach gave a small but meaningful improvement in performance, even with the large distribution shift in the test data. The second cross-dataset evaluation approach, using the cosine similarity as logits, also showed a similar trend as the matching dataset experiments, providing a simpler alternative method without having to curate datasets with matching classes.

6.2 Successful Results

Re-ordering video representation. It is interesting to see that even changing the input channel ordering of the same video can drastically improve performance.

Consistent representation. TC Reordering has duplicated channels at the end of the frames. This inconsistency sometimes hurts the performance depending on the task. On the other hand, TC+2 and GrayST representations are consistent throughout the frames. This improved stability of the performance in more general network architectures and datasets.

Choice of datasets. Each dataset has different benefits and problems. Understanding the data and problem of the dataset is essential to apply any methods. Chapter 3 shows that our sampling methods work particularly well on temporal datasets. In Chapter 4, existing datasets lack multi-label annotations, and annotating further in a multi-label fashion played a key role in success. In Chapter 5, we created dataset splits to evaluate with a noticeable distribution shift, which proved our claim that distribution disparity hurts performance and adversarial augmentation training can compensate it.

Pseudo-labeling. As can be seen in Chapter 4, most of the video datasets have some

form of ambiguity. Pseudo labeling or label correction methods are highly suggested to look into when dealing with action recognition datasets or other verb-based research questions.

Augmentation. Better data augmentation strategies can improve action recognition model performances by a meaningful margin. An alternative video data augmentation strategy using adversarial loss is suggested in Chapter 5.

6.3 What Did Not Work

Spatial datasets. As discussed in Chapter 3, it is sufficient to use image-based networks on datasets that do not require temporal reasoning. The proposed methods increase the temporal understanding capabilities of the models, but the datasets simply do not require motion understanding and the action can be inferred from simply looking at objects and backgrounds. For example, in HMDB, UCF, and Kinetics, there is a class about playing basketball. This can be easily inferred from a single image that contains a basketball court and players.

Transfer learning between largely different datasets. We often observed that choosing the right pre-trained network is a very important step to action recognition. Using a large-scale dataset like Kinetics does not guarantee to give better performance than Imagenet pre-trained models in many cases. This shows how much disparity video tasks have between the datasets, although they fall into the same video classification category. We observed that HMDB, UCF, and Kinetics datasets are very similar, so using Kinetics pre-trained models improves performance drastically. However, for other datasets like Something-Something and EPIC-Kitchens, it is better to use Imagenet pre-trained models because the required task is too different from that in Kinetics.

Using image-based methods. Directly applying image-based methods to video problems can fail. For example, most of the label correction techniques assume there exist two categories that annotators often get confused about and interchange them. However, in video action recognition, we observed that many verbs overlap in meaning depending on the context of the video and 2.4 verb labels were valid per video on average.

6.4 Future Work

We explored three main problems that make it difficult to deploy action recognition models to real applications. The first problem was that the recent action recognition models require significant costs to run, and thus we proposed new sampling strategies for efficient 2D models. However, it is potentially feasible to design strategies that are more generic to more types of models. We summarize the options below.

- **Smart channel sampling.** We presented a simple sampling strategy that enhances temporal information in a frame. However, this still samples a few frames of the video among many, in a sparse-sampling strategy that samples frames with a consistent frame rate. Through thorough analysis of model activations, it should be possible to design a system that samples channels and aggregate them with the goal of maximizing the temporal cue of the intermediate features. There exists work that investigates smart frame sampling [77], but with the combination of our channel sampling technique, it has the potential to improve action recognition performance by a great margin.

- **Channel sampling for 3D or Transformer networks.** The goal of the proposed channel sampling techniques is to improve 2D-based action networks. Usually, transformer-based networks have much better temporal reasoning capabilities even without re-arranging the channels. That being said, there is no doubt that inputting the more useful frames for classification will enhance the system no matter what the architecture does. Perhaps the smart channel sampling strategy suggested above can be used in more general types of models.

- **New video representation.** There can be other types of efficient representation than channel sampling. For instance, the original RGB representation itself is not the most efficient way to represent images. A more intuitive representation includes HSL (Hue, Saturation, Luminance). There may be another way to take advantage of image components in other color representations and re-ordering them.

In Chapter 4, we showed that the verb labels can often be used interchangeably, and as a consequence cause issues in training and testing. To make the problem simpler, we focused only on verb labels. In addition, we found that even other datasets

(*e.g.*, Something-Something) suffer from this ambiguity issue even though the labels are more than just verbs. We summarize the future work below.

- **Verb ambiguity problem with noun labels.** In our problem setup, we focused on verb-only labels because we found that ambiguity comes in verbs most of the time. However, depending on the chosen verb, the valid noun label can change. For example, there is a video of the participant taking a basket so that the water is poured into the sink. That can be labeled as take "basket", turn "basket", empty "water", pour "water", throw-away "water", and so on. Thus, one cannot simply add a noun classifier in the model that predicts multiple verbs and it is necessary to verify the validity of both. A recent advance in language models can significantly benefit in this problem. Furthermore, given a single positive noun label, it should help improve verb predictions and vice versa, so it is expected to see better verb classification performance.
- **Ambiguity in different datasets.** In Chapter 4, we mostly studied the problem in EPIC-Kitchens dataset. However, other datasets like Something-Something have similar label ambiguity problems, but their distribution may be different due to being a scripted action dataset.

Finally, in Chapter 5, we discussed the disparity between training and real examples causing a significant drop in performance. Our proposed adversarial augmentation strategy is very simple to implement, however, there is room for improvement. We propose the topics below as future work.

- **Video-based augmentation.** Our work on adversarial augmentation still applies the same augmentations throughout the entire video frames. It is desirable to study a method that augments each frame differently, but not completely independently to each other, so that the overall augmentation has a realistic flow over time.
- **Adversarial augmentation pipeline with memory.** The current adversarial augmentation generates "harder than random" augmentations by initializing the augmentation model parameters randomly and taking a gradient ascent step to tune parameters once. One might be able to take advantage of tracking the augmentation quality of all the samples in the training dataset, and generate views

that explore/exploit based on the history (*e.g.*, one can generate different kinds of augmentation each epoch for one video, or find an optimal augmentation for one and use it again).

Bibliography

[1] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6202–6211, 2019.

[2] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3202–3211, 2022.

[3] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks for action recognition in videos. *IEEE transactions on pattern analysis and machine intelligence*, 41(11):2740–2755, 2018.

[4] 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. *Advances in Neural Information Processing Systems*, 32, 2019.

[5] Elijah Cole, Oisin Mac Aodha, Titouan Lorieul, Pietro Perona, Dan Morris, and Nebojsa Jojic. Multi-label learning from single positive labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 933–942, 2021.

[6] Donghao Zhou, Pengfei Chen, Qiong Wang, Guangyong Chen, and Pheng-Ann Heng. Acknowledging the unknown for multi-label learning with single positive labels. In *ECCV*, 2022.

[7] Xinshao Wang, Yang Hua, Elyor Kodirov, David A Clifton, and Neil M Robertson. Proselflc: Progressive self label correction for training robust deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 752–761, 2021.

[8] Davide Moltisanti, Michael Wray, Walterio Mayol-Cuevas, and Dima Damen. Trespassing the boundaries: Labeling temporal bounds for object interactions in egocentric video. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

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

[10] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7083–7093, 2019.

[11] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014.

[12] Kunchang Li, Yali Wang, Gao Peng, Guanglu Song, Yu Liu, Hongsheng Li, and Yu Qiao. Uniformer: Unified transformer for efficient spatial-temporal representation learning. In *International Conference on Learning Representations*, 2022.

[13] Chunhui Gu, Chen Sun, David A Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, et al. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6047–6056, 2018.

[14] Rohit Girdhar, João Carreira, Carl Doersch, and Andrew Zisserman. A better baseline for ava. *arXiv preprint arXiv:1807.10066*, 2018.

[15] Chen Sun, Abhinav Shrivastava, Carl Vondrick, Kevin Murphy, Rahul Sukthankar, and Cordelia Schmid. Actor-centric relation network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 318–334, 2018.

[16] Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krahenbuhl, and Ross Girshick. Long-term feature banks for detailed video understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 284–293, 2019.

[17] Georgia Gkioxari and Jitendra Malik. Finding action tubes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 759–768, 2015.

[18] Vicky Kalogeiton, Philippe Weinzaepfel, Vittorio Ferrari, and Cordelia Schmid. Action tubelet detector for spatio-temporal action localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4405–4413, 2017.

[19] Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Learning to track for spatio-temporal action localization. In *Proceedings of the IEEE international conference on computer vision*, pages 3164–3172, 2015.

[20] Chao-Yuan Wu, Manzil Zaheer, Hexiang Hu, R Manmatha, Alexander J Smola, and Philipp Krähenbühl. Compressed video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6026–6035, 2018.

[21] Lionel Gueguen, Alex Sergeev, Ben Kadlec, Rosanne Liu, and Jason Yosinski. Faster neural networks straight from jpeg. In *Advances in Neural Information Processing Systems*, pages 3933–3944, 2018.

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

[23] Bowen Zhang, Limin Wang, Zhe Wang, Yu Qiao, and Hanli Wang. Real-time action recognition with enhanced motion vector cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2718–2726, 2016.

[24] Hakan Bilen, Basura Fernando, Efstratios Gavves, and Andrea Vedaldi. Action recognition with dynamic image networks. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2799–2813, 2017.

[25] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *Proceedings of the European conference on computer vision (ECCV)*, pages 399–417, 2018.

[26] Joao Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. A short note on the kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987*, 2019.

[27] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *2011 International conference on computer vision*, pages 2556–2563. IEEE, 2011.

[28] 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.

[29] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision*, pages 5842–5850, 2017.

[30] Laura Sevilla-Lara, Shengxin Zha, Zhicheng Yan, Vedanuj Goswami, Matt Feiszli, and Lorenzo Torresani. Only time can tell: Discovering temporal data for temporal modeling. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 535–544, January 2021.

[31] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 720–736, 2018.

[32] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Evangelos Kazakos, Jian Ma, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Rescaling egocentric vision. *International Journal of Computer Vision*, 2021.

[33] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, Miguel Martin, Tushar Nagarajan, Ilija Radosavovic, Santhosh Kumar Ramakrishnan, Fiona Ryan, Jayant Sharma, Michael Wray, Mengmeng Xu,

Eric Zhongcong Xu, Chen Zhao, Siddhant Bansal, Dhruv Batra, Vincent Cartillier, Sean Crane, Tien Do, Morrie Doulaty, Akshay Erapalli, Christoph Feichtenhofer, Adriano Fragomeni, Qichen Fu, Christian Fuegen, Abrham Gebreselasie, Cristina Gonzalez, James Hillis, Xuhua Huang, Yifei Huang, Wenqi Jia, Weslie Khoo, Jachym Kolar, Satwik Kottur, Anurag Kumar, Federico Landini, Chao Li, Yanghao Li, Zhenqiang Li, Karttikeya Mangalam, Raghava Modhugu, Jonathan Munro, Tullie Murrell, Takumi Nishiyasu, Will Price, Paola Ruiz Puentes, Merey Ramazanova, Leda Sari, Kiran Somasundaram, Audrey Southerland, Yusuke Sugano, Ruijie Tao, Minh Vo, Yuchen Wang, Xindi Wu, Takuma Yagi, Yunyi Zhu, Pablo Arbelaez, David Crandall, Dima Damen, Giovanni Maria Farinella, Bernard Ghanem, Vamsi Krishna Ithapu, C. V. Jawahar, Hanbyul Joo, Kris Kitani, Haizhou Li, Richard Newcombe, Aude Oliva, Hyun Soo Park, James M. Rehg, Yoichi Sato, Jianbo Shi, Mike Zheng Shou, Antonio Torralba, Lorenzo Torresani, Mingfei Yan, and Jitendra Malik. Ego4d: Around the World in 3,000 Hours of Egocentric Video. In *IEEE/CVF Computer Vision and Pattern Recognition (CVPR)*, 2022.

[34] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013.

[35] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[36] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.

[37] Chenxu Luo and Alan L Yuille. Grouped spatial-temporal aggregation for efficient action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5512–5521, 2019.

[38] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 6202–6211, 2019.

[39] Xianhang Li, Yali Wang, Zhipeng Zhou, and Yu Qiao. Smallbignet: Integrating core and contextual views for video classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1092–1101, 2020.

[40] Ziyuan Huang, Shiwei Zhang, Liang Pan, Zhiwu Qing, Mingqian Tang, Ziwei Liu, and Marcelo H Ang Jr. Tada! temporally-adaptive convolutions for video understanding. *arXiv preprint arXiv:2110.06178*, 2021.

[41] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976, 2017.

[42] Gursimran Singh. *Spatio-temporal relational reasoning for video question answering*. PhD thesis, University of British Columbia, 2019.

[43] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[45] Lu Chi, Zehuan Yuan, Yadong Mu, and Changhu Wang. Non-local neural networks with grouped bilinear attentional transforms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11804–11813, 2020.

[46] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6824–6835, 2021.

[47] 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.

[48] Adam Santoro, David Raposo, David GT Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *NeurIPS*, 2017.

[49] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[50] Wenhao Wu, Dongliang He, Tianwei Lin, Fu Li, Chuang Gan, and Errui Ding. Mvfnet: Multi-view fusion network for efficient video recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2943–2951, 2021.

[51] Limin Wang, Zhan Tong, Bin Ji, and Gangshan Wu. Tdn: Temporal difference networks for efficient action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1895–1904, 2021.

[52] Yan Li, Bin Ji, Xintian Shi, Jianguo Zhang, Bin Kang, and Limin Wang. Tea: Temporal excitation and aggregation for action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 909–918, 2020.

[53] Chao-Yuan Wu, Yanghao Li, Karttikeya Mangalam, Haoqi Fan, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Memvit: Memory-augmented multiscale vision transformer for efficient long-term video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13587–13597, 2022.

[54] Chao-Yuan Wu, Ross Girshick, Kaiming He, Christoph Feichtenhofer, and Philipp Krahenbuhl. A multigrid method for efficiently training video models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 153–162, 2020.

[55] Hanting Chen, Yunhe Wang, Han Shu, Yehui Tang, Chunjing Xu, Boxin Shi, Chao Xu, Qi Tian, and Chang Xu. Frequency domain compact 3d convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1641–1650, 2020.

[56] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 203–213, 2020.

[57] Jinhyung Kim, Seunghwan Cha, Dongyoon Wee, Soonmin Bae, and Junmo Kim. Regularization on spatio-temporally smoothed feature for action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12103–12112, 2020.

[58] AJ Piergiovanni and Michael S Ryoo. Representation flow for action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9945–9953, 2019.

[59] Nieves Crasto, Philippe Weinzaepfel, Karteek Alahari, and Cordelia Schmid. Mars: Motion-augmented rgb stream for action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7882–7891, 2019.

[60] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. VideoMAE: Masked autoencoders are data-efficient learners for self-supervised video pre-training. In *Advances in Neural Information Processing Systems*, 2022.

[61] Chaitanya Ryali, Yuan-Ting Hu, Daniel Bolya, Chen Wei, Haoqi Fan, Po-Yao Huang, Vaibhav Aggarwal, Arkabandhu Chowdhury, Omid Poursaeed, Judy Hoffman, Jitendra Malik, Yanghao Li, and Christoph Feichtenhofer. Hiera: A hierarchical vision transformer without the bells-and-whistles. *ICML*, 2023.

[62] Rui Wang, Dongdong Chen, Zuxuan Wu, Yinpeng Chen, Xiyang Dai, Mengchen Liu, Lu Yuan, and Yu-Gang Jiang. Masked video distillation: Rethinking masked feature modeling for self-supervised video representation learning. In *CVPR*, 2023.

[63] Serena Yeung, Vignesh Ramanathan, Olga Russakovsky, Liyue Shen, Greg Mori, and Li Fei-Fei. Learning to learn from noisy web videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5154–5162, 2017.

[64] Mohit Sharma, Raj Aaryaman Patra, Harshal Desai, Shruti Vyas, Yogesh Rawat, and Rajiv Ratn Shah. Noisyactions2m: A multimedia dataset for video understanding from noisy labels. In *ACM Multimedia Asia*, pages 1–5, 2021.

[65] Deepti Ghadiyaram, Du Tran, and Dhruv Mahajan. Large-scale weakly-supervised pre-training for video action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12046–12055, 2019.

[66] Thomas Leung, Yang Song, and John Zhang. Handling label noise in video classification via multiple instance learning. In *2011 International Conference on Computer Vision*, pages 2056–2063. IEEE, 2011.

[67] Anurag Arnab, Chen Sun, Arsha Nagrani, and Cordelia Schmid. Uncertainty-aware weakly supervised action detection from untrimmed videos. In *European Conference on Computer Vision*, pages 751–768. Springer, 2020.

[68] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

[69] Chenyu Yi, SIYUAN YANG, Haoliang Li, Yap peng Tan, and Alex Kot. Benchmarking the robustness of spatial-temporal models against corruptions. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[70] Madeline Chantry Schiappa, Naman Biyani, Prudvi Kamtam, Shruti Vyas, Hamid Palangi, Vibhav Vineet, and Yogesh S. Rawat. A large-scale robustness analysis of video action recognition models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14698–14708, June 2023.

[71] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning (ICML)*, July 2021.

[72] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[73] Linchao Zhu, Du Tran, Laura Sevilla-Lara, Yi Yang, Matt Feiszli, and Heng Wang. Faster recurrent networks for efficient video classification. In *Proceed-*

ings of the AAAI Conference on Artificial Intelligence, volume 34, pages 13098–13105, 2020.

[74] Rohit Girdhar and Deva Ramanan. CATER: A diagnostic dataset for Compositional Actions and TEmporal Reasoning. In *ICLR*, 2020.

[75] Yiting Xie and David Richmond. Pre-training on grayscale imagenet improves medical image classification. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018.

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

[77] Shreyank N Gowda, Marcus Rohrbach, and Laura Sevilla-Lara. Smart frame selection for action recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1451–1459, 2021.

[78] Yingwei Li, Yi Li, and Nuno Vasconcelos. Resound: Towards action recognition without representation bias. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[79] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? *arXiv preprint arXiv:2102.05095*, 2021.

[80] Roger G Barker and Herbert F Wright. Midwest and its children: The psychological ecology of an american town. 1955.

[81] Sandra Waxman, Xiaolan Fu, Sudha Arunachalam, Erin Leddon, Kathleen Geraghty, and Hyun-joo Song. Are nouns learned before verbs? infants provide insight into a long-standing debate. *Child development perspectives*, 7(3):155–159, 2013.

[82] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[83] Michael Wray and Dima Damen. Learning visual actions using multiple verb-only labels. *British Machine Vision Conference*, 2019.

[84] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2013.

[85] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[86] Christian Rupprecht, Iro Laina, Robert DiPietro, Maximilian Baust, Federico Tombari, Nassir Navab, and Gregory D Hager. Learning in an uncertain world: Representing ambiguity through multiple hypotheses. In *Proceedings of the IEEE international conference on computer vision*, pages 3591–3600, 2017.

[87] Diego Ortego, Eric Arazo, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Multi-objective interpolation training for robustness to label noise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6606–6615, 2021.

[88] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1944–1952, 2017.

[89] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5552–5560, 2018.

[90] Michael Wray, Davide Moltisanti, Walterio Mayol-Cuevas, and Dima Damen. Sembed: Semantic embedding of egocentric action videos. In *European Conference on Computer Vision*, pages 532–545. Springer, 2016.

[91] Spandana Gella, Mirella Lapata, and Frank Keller. Unsupervised visual sense disambiguation for verbs using multimodal embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016.

[92] Xiang Deng, Songhe Feng, Gengyu Lyu, Tao Wang, and Congyan Lang. Beyond word embeddings: Heterogeneous prior knowledge driven multi-label image classification. *IEEE Transactions on Multimedia*, 2022.

[93] Xubin Zhong, Changxing Ding, Xian Qu, and Dacheng Tao. Polysemy deciphering network for robust human–object interaction detection. *International Journal of Computer Vision*, 129(6):1910–1929, 2021.

[94] Arushi Goel, Basura Fernando, Frank Keller, and Hakan Bilen. Not all relations are equal: Mining informative labels for scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15596–15606, 2022.

[95] Thomas Verelst, Paul K Rubenstein, Marcin Eichner, Tinne Tuytelaars, and Maxim Berman. Spatial consistency loss for training multi-label classifiers from single-label annotations. *arXiv preprint arXiv:2203.06127*, 2022.

[96] Youcai Zhang, Yuhao Cheng, Xinyu Huang, Fei Wen, Rui Feng, Yaqian Li, and Yandong Guo. Simple and robust loss design for multi-label learning with missing labels. *arXiv preprint arXiv:2112.07368*, 2021.

[97] Thibaut Durand, Nazanin Mehrasa, and Greg Mori. Learning a deep convnet for multi-label classification with partial labels. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 647–657, 2019.

[98] Aritra Ghosh and Andrew Lan. Contrastive learning improves model robustness under label noise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2703–2708, 2021.

[99] Ankit Singh, Omprakash Chakraborty, Ashutosh Varshney, Rameswar Panda, Rogerio Feris, Kate Saenko, and Abir Das. Semi-supervised action recognition with temporal contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10389–10399, 2021.

[100] Mamshad Nayeem Rizve, Kevin Duarte, Yogesh S Rawat, and Mubarak Shah. In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning. In *International Conference on Learning Representations*, 2021.

[101] Islam Nassar, Samitha Herath, Ehsan Abbasnejad, Wray Buntine, and Gholamreza Haffari. All labels are not created equal: Enhancing semi-supervision via label grouping and co-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7241–7250, 2021.

[102] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[103] Mohammad S Sorower. A literature survey on algorithms for multi-label learning. *Oregon State University, Corvallis*, 18:1–25, 2010.

[104] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[105] Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime TV-L 1 optical flow. *Pattern Recognition*, pages 214–223, 2007.

[106] Wei Lin, Muhammad Jehanzeb Mirza, Mateusz Kozinski, Horst Possegger, Hilde Kuehne, and Horst Bischof. Video test-time adaptation for action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22952–22961, June 2023.

[107] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[108] Daniel Ho, Eric Liang, Xi Chen, Ion Stoica, and Pieter Abbeel. Population based augmentation: Efficient learning of augmentation policy schedules. In *International conference on machine learning*, pages 2731–2741. PMLR, 2019.

[109] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. *Advances in Neural Information Processing Systems*, 32, 2019.

[110] Ryuichiro Hataya, Jan Zdenek, Kazuki Yoshizoe, and Hideki Nakayama. Faster autoaugment: Learning augmentation strategies using backpropagation. In

Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16, pages 1–16. Springer, 2020.

[111] Yonggang Li, Guosheng Hu, Yongtao Wang, Timothy Hospedales, Neil M Robertson, and Yongxin Yang. Differentiable automatic data augmentation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*, pages 580–595. Springer, 2020.

[112] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[113] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[114] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.

[115] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020.

[116] Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. Recent advances in adversarial training for adversarial robustness. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2021.

[117] Junhao Dong, Seyed-Mohsen Moosavi-Dezfooli, Jianhuang Lai, and Xiaohua Xie. The enemy of my enemy is my friend: Exploring inverse adversaries for improving adversarial training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24678–24687, June 2023.

[118] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. In *International Conference on Learning Representations (ICLR)*, 2018.

[119] Kaiyang Zhou, Yongxin Yang, Timothy M. Hospedales, and Tao Xiang. Deep domain-adversarial image generation for domain generalisation. In *Conference on Artificial Intelligence, (AAAI)*, 2020.

[120] Minyoung Kim, Da Li, and Timothy M. Hospedales. Domain generalisation via domain adaptation: An adversarial fourier amplitude approach. In *International Conference on Learning Representations (ICLR)*, 2023.

[121] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015.

[122] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *Symposium on Security and Privacy*, 2017.

[123] Yue Zhao, Yuwei Wu, Caihua Chen, and Andrew Lim. On isometry robustness of deep 3d point cloud models under adversarial attacks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[124] Ruibin Wang, Yibo Yang, and Dacheng Tao. Art-point: Improving rotation robustness of point cloud classifiers via adversarial rotation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[125] Arno Blaas, Xavier Suau, Jason Ramapuram, Nicholas Apostoloff, and Luca Zappella. Challenges of adversarial image augmentations. In *I (Still) Can't Believe It's Not Better! Workshop at NeurIPS 2021*, 2021.

[126] Qi-Zhi Cai, Min Du, Chang Liu, and Dawn Song. Curriculum adversarial training. *arXiv preprint arXiv:1805.04807*, 2018.

[127] Ahmadreza Jeddi, Mohammad Javad Shafiee, Michelle Karg, Christian Scharfenberger, and Alexander Wong. Learn2perturb: An end-to-end feature perturbation learning to improve adversarial robustness. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[128] Tianyuan Yu, Yongxin Yang, Da Li, Timothy M. Hospedales, and Tao Xiang. Simple and effective stochastic neural networks. In *Conference on Artificial Intelligence (AAAI)*, 2021.

[129] Panagiotis Eustratiadis, Henry Gouk, Da Li, and Timothy M. Hospedales. Weight-covariance alignment for adversarially robust neural networks. In *International Conference on Machine Learning (ICML)*, 2021.

[130] Jingfeng Zhang, Xilie Xu, Bo Han, Gang Niu, Lizhen Cui, Masashi Sugiyama, and Mohan Kankanhalli. Attacks which do not kill training make adversarial learning stronger. In *International conference on machine learning*, pages 11278–11287. PMLR, 2020.

[131] Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R Arabnia. A brief review of domain adaptation. *Advances in data science and information engineering: proceedings from ICDATA 2020 and IKE 2020*, pages 877–894, 2021.

[132] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR, 2015.

[133] Xu Zhang, Felix Xinnan Yu, Shih-Fu Chang, and Shengjin Wang. Deep transfer network: Unsupervised domain adaptation. *arXiv preprint arXiv:1503.00591*, 2015.

[134] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520, 2011.

[135] Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, and Wen Li. Deep reconstruction-classification networks for unsupervised domain adaptation. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 597–613. Springer, 2016.

[136] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015.

[137] Zhongyi Pei, Zhangjie Cao, Mingsheng Long, and Jianmin Wang. Multi-adversarial domain adaptation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[138] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE international conference on computer vision*, pages 4068–4076, 2015.

[139] Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand. Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446*, 2014.

[140] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3722–3731, 2017.

[141] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2107–2116, 2017.

[142] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016.

[143] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998. Pmlr, 2018.

[144] Shiqi Lin, Zhizheng Zhang, Zhipeng Huang, Yan Lu, Cuiling Lan, Peng Chu, Quanzeng You, Jiang Wang, Zicheng Liu, Amey Parulkar, Viraj Navkal, and Zhibo Chen. Deep frequency filtering for domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11797–11807, June 2023.

[145] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*, 2017.

[146] Shreyank N Gowda, Laura Sevilla-Lara, Kiyoon Kim, Frank Keller, and Marcus Rohrbach. A new split for evaluating true zero-shot action recognition. In

DAGM German Conference on Pattern Recognition, pages 191–205. Springer, 2021.

[147] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.