ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CVL Computer Vision Lab

Domain-Adaptation in Action-Detection using Video Clip Order Prediction

Semester Thesis

Rishabh Singh
Department of Computer Science

Supervisor: Dr. Suman Saha and Dr. Yuhua Chen
Professor: Prof. Dr. Luc van Gool

April 8, 2021

Abstract

Recent years have witnessed rapid progress in unsupervised domain adaptation (UDA) techniques tailored to image-based problems such as semantic segmentation and object detection. However, for video-based problems like human action detection, robust domain adaptation techniques are still missing. This work tries to address such a gap in the literature by proposing a novel deep learning framework to tackle the domain shift in action detection. The proposed approach leverages the recent advancements in self-supervised video representation learning to aid domain alignment. More specifically, we use video clip order prediction as a pretext task to learn spatiotemporal representation from videos. We hypothesize that learning the chronological order of clips might help the network to understand the temporal patterns of actions more effectively. Since we do not need any ground truth annotations to compute the clip order prediction loss, it can be applied to either source or target or both the domains. We further conjecture that the features learned during the clip order prediction are helpful to mitigate the domain shift problem. We experimentally observe improvements in the target domain's action detection performance while applying the self-supervised clip order prediction loss. Specifically, we show that: (1) self-supervision improves the cross-domain and within-domain performances by 7.3% and 6.5%, respectively, on the challenging AVA and Kinetics-700 action detection benchmarks; (2) the features learned by our method have better inter-class variations and intra-class affinities while improving the mAP for each class. Finally, a qualitative analysis of the action detection results under different scenarios attests to the efficacy of the proposed method.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A video is a sequence of natural images ordered according to the time they were captured by a camera device. Videos play an important role in present scenarios for media applications as they store a vast amount of multi-modal information in a single format. This also makes video content understanding a challenging task under Artificial Intelligence (AI). Given the inherent complexity inherited from the video capturing process and the multi-modal information, extracting relevant insights from a video input is at times not feasible. Complementary modes can be useful sometimes, for eg. subtitles in a movie. However, majority of the times these complementary signals are either not available or confusing, e.g. music playing in a sports video. In the context of this thesis, we will only concern ourselves with the visual content understanding of video samples.

1.1 Why human action detection?

Human action understanding has been integral to video understanding, with applications ranging from surveillance, security, sports, assisted living, gaming, entertainment, education etc. A huge proportion of online video content is created around humans and their actions. Moreover, action understanding implicitly requires to concurrently solve several other problems such as human interaction with objects, localisation, tracking, motion-understanding etc. Human action understanding can be classified into various subcategories such as:

- Action recognition is one of the fundamental problems in human action understanding. Here, given a video clip of fixed duration, we label it with one of the categories in an action vocabulary. For instance Kinetics dataset [5] contains video clips of 10s duration with labels assigned per frame of the video among 700 possible classes.

- Temporal action segmentation deals with labelling every frame in the video with one or more labels from a list of action categories. Examples include datasets like Charades [59]

- Temporal action detection identifies the temporal boundaries (start and end timestamps) of each action instance along with its label. For a video sample, we detect all action instances with their category labels, start time, and end time. Example datasets include Charades[59]

- Spatio-temporal action detection aims to identify the label, temporal boundaries and spatial bounding box in each frame associated with an action instance. Also known as *action tube* [22], each action instance is a set of bounding boxes linked across time. Some example datasets include AVA dataset

containing 15-mins video clips where certain frames contain bounding box and action label annotations [23]

In this work we focus on spatio-temporal human action detection as the main task.

1.2 Why domain adaptation for action detection?

Convolutional Neural Networks (CNNs) have drastically improved the performance of action detection tasks and have been raging as the state-of-the-art models recently. A typical CNN model is trained on benchmark datasets for action detection and deployed for applications in real-time. However, it is often observed that the performance of these models tend to deteriorate when tested on data not used for training. In reality, any acquired visual data is subject to color imbalances, perspective distortions, variable background, or even motion blur. More specifically, the test data follows a different *distribution or domain*. In the domain adaptation (DA) terminology, these are referred to as the *source* (data used for training the model) and *target* (data used for testing the model) domains. The final aim will be to learn a classifier that is able to generalize well to datasets different from the source domain. There has been a lack of works tackling domain shift occurring between the source and target domains in the case of action detection, and this work attempts to bridge that gap.

1.3 Why self-supervised learning for domain adaptation?

One possible option to overcome the domain shift would be to fine-tune the classifier using target domain labels. Acquiring and annotating videos in every new target domain, however, is an expensive, time-consuming, and often infeasible task. The goal thus is to minimize annotation efforts and use an unsupervised domain adaptation (UDA) technique. Although many methods have been proposed in the literature, in this work we would leverage self-supervised learning (SSL) [15] to achieve UDA. The primary motivation to use self-supervised learning comes from the fact that no additional data or ground truth annotations are needed to train a self-supervised task. Also, training and optimizing simple objectives as pretext tasks in SSL is relatively an easier alternative to complex ones such as adversarial training [8]. The main idea is to achieve an alignment between the source and target domains by training a model on the same task in both domains simultaneously. Indeed, if we had labels in both domains, we could simply use our original classification task for this. However, since we lack labels in the target domain, we propose to use a self-supervised auxiliary task, which creates its own labels directly from the data. In fact, we can use multiple self-supervised tasks, each one aligning the two domains along a direction of variation relevant to that task. Jointly training all the self-supervised tasks on both domains together with the original task on the source domain produces well-aligned representations [64].

1.4 Why clip-order prediction as the self-supervised task?

Self-supervised learning is one kind of technique where the supervisory signal can be obtained easily from the data itself. Research done in this area has exploited both images and videos, although self-supervision for video understanding has still been under-explored. For image data, there exist self-supervised tasks such as predicting relative positions of image patches [12], solve jigsaw puzzles [50], image inpainting [54] and image color channel prediction [39]. Since the particular property of the video is temporal information, recent works also attempt to leverage the temporal relations among frames, such as order verification [48]

and order prediction [41] of frames. The existing self-supervised works that utilize video inputs have the framework as follows: first use 2D CNNs to extract features from the frames, then concatenate these features and predict the verification result or the actual order of the input frames. The whole framework is trained end-to-end. After the training, the learned 2D CNNs can be used as an image feature extractor or fine-tuned to other tasks such as image classification and detection. Compared to order verification, order prediction contains much richer supervisory signals and shows better performance in several validation experiments. However, the order is not uniquely determined when referring to frames merely. For actions which are bi-directional such as gymnist on a pole, given the shuffled frames, it is hard to tell the correct order of frames since both directions of the gymnast in the balance beam seems possible. To mitigate the defect of this, [41] groups both the forward and backward orders as the same class. It is a compromise under the circumstances that only frames and 2D CNNs are used. In contrast, we propose to directly use clips and 3D CNNs to make the task more well-defined. It is rational to assume that if the shuffled clips are provided, the order will be more specific because of the inner dynamics contained in each clip and that predicting this order will inherently train the model to focus on the spatio-temporal coherence in a video. Besides, the 3D CNNs are always believed to be hard to optimize due to the lack of labeled videos [24]. With the assistance of the clip order prediction task, our action detection backbone composed of 3D CNNs can leverage numerous videos without any labels. The 3D CNNs thus can be easily pre-trained to adapt to different video distributions in new application domains, which is a pre-requisite to gain good performance in UDA. Lastly, since our action detection backbone inherently focuses on varying temporal resolution as different pathways, we believe a self-supervision of clip order prediction, which is a task focused on temporal properties of videos, can aid more in the learning of the main task of action detection.

1.5 Overview

This work presents a self-supervision task based on video clip order prediction and claims that it helps improve domain adaptation for the objective of action detection. Two methods- a transfer learning method and a multi-task method are explored in this work. The encoder architecture is based on SlowFast, a large scale CNN model proposed by by Facebook research in [14]. This two pathway architecture has been shown to achieve impressive performance for both action recognition and action detection on benchmark datasets such as Kinetics and AVA respectively. AVA-Kinetics dataset [42] has been used throughout this work as it facilitates the analysis of cross-domain performance given the two datasets involved- AVA and Kinetics following the same annotation protocol. The rest of this report is organised as: we first give an extensive literature review on human action analysis, domain adaptation and self-supervision. We then discuss in detail our method and materials used to curate it such as architecture, datasets and the optimisation pipeline. We then present an elaborate experimental analysis and showcase the results achieved. This is followed by a cogent discussion section where the results obtained are first organised, presented in a more comprehensive form and then meaningful conclusions are drawn from them. Towards the end, we summarize the contribution of this work and outline a path for further improvement as future works.

Chapter 2

Related Work

2.1 Human Action Analysis

The term *Human Action* studied in computer vision research ranges from the simple limb movement to joint complex movement of multiple limbs and the human body. This process is dynamic, and thus is usually conveyed in a video lasting a few seconds. Though it might be difficult to give a formal definition of human action studied in the computer vision community, we provide some examples-

- Individual action, which contains simple daily actions such as “clapping” and “running”
- Human interaction, which consists of interactions between humans such as a “handshake”, human-object interaction such as “sports” or both for eg. “push”.
- Group Action, where a large number of humans perform a common activity such as “clapping” or “cheering”.

These actions are analysed under two objectives- Action Recognition and Action Detection. We shall discuss each in the following sections.

2.1.1 Human Action Recognition

Action recognition is a fundamental task in the computer vision community that recognizes human actions based on the complete action execution in a video. The recent advent of CNNs and their promising performance in image recognition has motivated quite a few works to explore them for action recognition in videos too. We will categorise the existing works on video action recognition into few broad categories. First set of works rely on optical flow images for motion information, and a single RGB frame for appearance information, which is limiting when reasoning about the temporal context that is required for video understanding. They consist of two separate CNNs - one for RGB frame and another for optical flow image, followed by a late fusion of prediction scores of both the streams [60]. The second class of works consider video as a sequence of frames. These methods sample multiple frames from a video input and use 2D CNNs to extract frame level representations and finally classifying among the given set of classes [62]. The third set of works model short-term temporal dependencies, thus working as a middle ground between late aggregation of frame features and early temporal processing of optical flow for eg. DMC-Nets [58]. The fourth set of works consider videos as a space-time volume. These works rely on using 3D CNNs instead of 2D

ones. However, given the huge number of parameters involved, these works focus on addressing the computational aspect of training more. [6] addresses this limitation by inflating video 3D kernels with the 2D weights of a CNN trained for image recognition. R(2+1)D [68] also show that a 2D convolution followed by 1D convolution is enough to learn discriminative features for action recognition. The fifth and the final set of works are being explored more in recent times where the focus is on modeling spatial-temporal features. These works focus on learning representations from 3D-CNNs that are efficeint from a computational point of view and effective from a performance point of view [18].

2.1.2 Human Action Detection

Compared to action recognition, action detection is a more challenging problem. A lot of works have been proposed in this regard. [66] uses spatio-temporal deformable parts model to detect actions in videos. Deep Learning has been extensively helpful in achieving superior performance for action detection. In [74], frame-level action proposals are obtained by EdgeBox and linked by a tracking algorithm. [55] proposes a two-stream R-CNN model for action detection where a spatial Region Proposal Network (RPN) and a motion RPN are used to generate frame-level action proposals. These earlier works were limited to exploiting spatial cues from videos. More recent works focus on 'action tubes' that contain both spatial and temporal information about the input [29]. Recently proposed work in [18] introduces the Action Transformer model for recognizing and localizing human actions in video clips.

2.2 Domain Adaptation

Domain Adaptation (DA) has recently become important, especially with the integration of deep learning and the necessity to deploy a model trained on one dataset for a variety of real-time applications in different scenarios. With the two branch (source and target) framework for most DA works, finding a common feature space between source and target domains is the ultimate goal, and the key is to design the domain loss to achieve this goal [9]. Domain adaptation has been extensively explored for image-based objectives. We can combine the existing set of literature into a few broad categories.

Discrepancy-based DA [45] is one of the major class of methods where the main goal is to reduce the distribution distance between the two domains. One popular measurement is the maximum mean discrepancy (MMD) – the distance between the mean of the two domains in some reproducing kernel Hilbert space, where the kernel is chosen to maximize the distance and hence induce alignment between the source and the target domains in some feature space [2]. Adversarial-based DA [16] is also popular by using domain discriminators. With carefully curated adversarial objectives, the domain discriminator and the feature extractor are optimized through a min-max training. Since the optimization landscape is much more complex than in standard supervised learning, training often does not converge or converges to a bad local minimum [43]. It also requires carefully balancing the two sets of parameters (for minimization and maximization) so one does not dominate the other [49]. Some works directly transform the source images to resemble the target images with generative models to achieve domain alignment [28]. These methods operate on image pixels directly instead of an intermediate representation space, and therefore can benefit from an additional round of adaptation in some representation space [65]. Some works consider assigning pseudo-labels to target data [76] using a model trained on the labeled source data to estimate labels on the target data, then trains on some of those estimated pseudo-labels (e.g. the most confident ones), therefore bootstrapping through the unlabeled target data. Furthermore, Ensemble-based DA [40] includes multiple target branches to construct an ensemble model. Recent works have also explored Attention-based DA [73] which assigns

attention weights to different regions of images for more effective DA.

Unlike images, unsupervised domain adaptation for videos is still not explored to its full. Some recent works focus on small-scale datasets for DA in video [32]. Few works have proposed Temporal attentive alignment for domain adaptation in videos [7]. Some works have utilized auxiliary data for other video tasks, including object detection [38] and action localization [79]. [8] is one of the most recent works that have used domain adaptation for video-based objectives. They argue for domain difference due to visual variation (appearance, perspective and motion, etc.) and due to different label sets between the source and target domains. This work considers both a semi-supervised domain adaptation setting with limited labels in the target domain and an unsupervised domain adaptation without any labelled data in the target domain. However, we will explore self-supervised learning for domain adaptation in this work. SSL is growing popular as a method for addressing DA where some works such as [15] have used SSL for learning robust video feature representations. SSL has advantage over adversarial learning as it does not require additional data, circumvent the need for any bounding box annotations thereby cutting manual labor and also an easier and more interpretive training of the network.

2.3 Self-supervised Learning in videos

SSL is an active topic of research in the computer vision community. Recent methods [1, 27] have produced state of the art results in learning representations which can distinguish between the 1000 categories of the ImageNet dataset [57]. The existing approaches can be broadly classified into three groups based on the type of information from videos that the self-supervision exploits, namely spatial, temporal and spatio-temporal.

2.3.1 Spatial Self-Supervision

This class of methods primarily extracts the spatial information present in the video frames to learn a self-supervised task. Previous works have explored various self-supervision cues such as predicting the spatial context [12], colorization [78], equivariance to transformations such as image rotation [17] and counting visual primitives [51]. Gidaris et al. [17] use rotation as a self-supervision task to learn semantic features for images. They emphasise the problem to be well-posed as objects have geometric constraints and tend to follow a particular physical state in a particular situation eg. humans while standing.Thus rotating images into multiples of 90 degress don not incur much ambiguity. They obtain an mAP of 54.4% on unsupervised ImageNet post fine-tuning done on PASCAL VOC 2007, thus outperforming existing unsupervised methods and reaching closer to the supervised ImageNet result of 56:8% mAP. Prior works have also utilised unsupervised techniques such as clustering [4, 80], generative modelling [37], and exemplar learning [13].

2.3.2 Temporal Self-Supervision

More relevant to our work is the section of SSL literature that exploits the temporal information of video frames. The temporal context of frames in video data has been widely utilised. For example, [77] propose a self-supervised temporal learning technique which leverages the chronological order of videos by predicting the order of shuffled clips from the video. There exist related works which use frames level order prediction [3, 15], while compared to frames, clips are more consistent with the video dynamics. Clips can help to reduce the uncertainty of orders and are more appropriate to learn a video representation. 3D convolutional neural networks are utilized to extract features for clips, and these features are processed to predict the actual order. The learned representations are evaluated via nearest neighbor retrieval experiments. They also use the learned networks as the pre-trained models and finetune them on the action recognition task achieving

significant improvements in performance. Other forms of temporal context include its combination with spatial context [72], and the use of spatio-temporal co-occurrence statistics [31].

Existing works on slow feature analysis [75] use videos for learning invariant representations. Slow and steady feature analysis [34] aims at learning representations which show temporal coherence at higher orders. Previous works based on temporal consistency used future frame prediction [61] as a pretext task. Single-frame prediction has proven to be a challenging task and many works in the past have been proposed to tackle it via time-agnostic prediction [33], motion segmentation [53], and cross-pixel matching [47]. Given that the videos available online are inherently multi-modal, self-supervision based on multi-modal learning methods that combine vision and audio [52], and vision and text [63] are shown to achieve better performance than their respective unimodal baselines. [69] propose a general framework for self-supervised learning of transferable visual representations based on Video-Induced Visual Invariances. They consider the implicit hierarchy present in the videos and make use of 1. frame-level invariances (e.g. stability to color and contrast perturbations), 2. shot/clip-level invariances (e.g. robustness to changes in object orientation and lighting conditions), and 3. video-level invariances (semantic relationships of scenes across shots/clips), to define a holistic self-supervised loss. Training models using different variants of the proposed framework on videos from the YouTube-8M (YT8M) data set, they obtain state-of-the-art self-supervised transfer learning results on the 19 diverse downstream tasks of the Visual Task Adaptation Benchmark, using only 1000 labels per task. The learned features are shown to improve performance for downstream tasks.

2.3.3 Spatio-temporal Self-supervision

Although there have been a lot of works for learning robust, enriched features from video samples, most of the works have focussed on using 2D CNNs and working at frame level. to essentially learn image representations by using either spatial or temporal information in videos as supervision but not both. Pathak et al. use a 2D-CNN to segment moving objects that are unsupervised segmented from videos [53]. Misra et al. verify if a sequence of frames follow the correct temporal order [48]. [71] proposes a Siamese-triplet network to rank patches from a video sequence. [15] uses odd-one-out networks to induce supervision. [41] sorts a sequence of shuffled video frames. All of these works have been based on 2D CNNs which limit their ability to focus on either spatial of temporal supervision at one time. 3D-CNN has been widely used to simultaneously model both spatial and temporal information in videos [67]. However, very few works have used them for self-supervision [70]. The performance of these methods have been limited too. Compared to the image based self-supervised learning, the spatio-temporal feature learning using 3D-CNN has not been explored to the best. Existing works in this direction can be classified into two categories- whether the pretext task is designed to focus more on the spatial or the temporal characteristics while learning the spatio-temporal features. An example of former an be classification of randomly rotated clips into different classes based on angles [35]. Here, although the features learnt are spatio-temporal, the pretext task has an objective that concerns with the spatial orientation of inputs. Our work on video clip order prediction falls in the latter case, where the focus is more on learning the temporal coherence of clips in a particular video sample.

Chapter 3

Materials and Methods

3.1 Overview

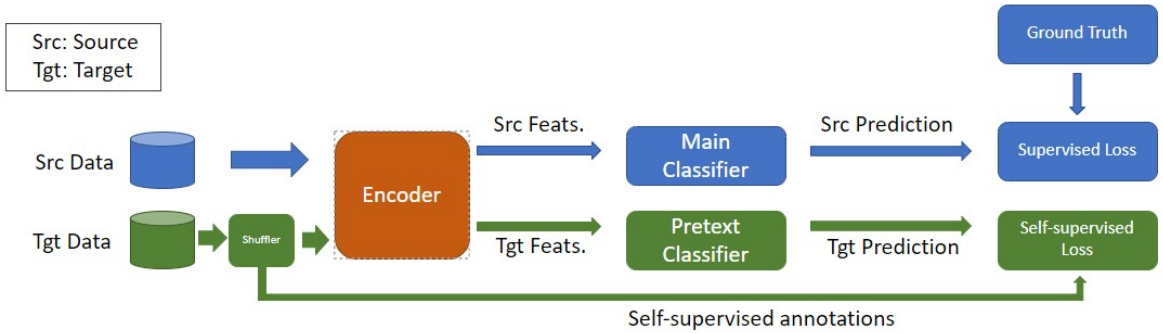


Figure 3.1: An overview of our training pipeline consisting of supervised (blue), self-supervised (green) paths and a common encoder network (orange)

We give an overview of the overall pipeline in this section. As seen in Fig. 3.1 our method broadly consists of two parallel streams of flow of information- the blue path refers to the main objective aka action detection, while the green path corresponds to a pretext task. Note here that the main task corresponds to minimizing the supervised loss function while the pretext task optimizes over the self-supervised loss. We typically analyse the performance of any model between two domains- Source and Target. The data from the source domain is transformed by an encoder network into source features which are given as input to a source classifier to get the predictions of the model on the source data. These predictions along with the already available annotations are used to minimize a supervised loss function and hence train the main classifier. Target domain usually refers to the dataset on which we want our model to generalize well at the test time. In an unsupervised domain adaptation setting, we are not allowed to access the labels from the target domain however, we can learn the characteristic properties of the distribution of the dataset by analysing the samples alone. On similar grounds, we define our self-supervised pathway to learn a simple classification task on the target dataset without relying on the actual labels. In fact, we define a pretext task, called shuffler here, which randomly shuffles the order of clips in a given video sample and simultaneously record the correct order of the clips to be used as self-supervised annotations later. Thus, the annotations used by the pretext task are

generated online and are not related to the actual ground truth labels provided for the target dataset. These shuffled clips are again transformed by the same encoder structure to generate the target features given as input to a pretext classifier. The predicted output of this classifier along with the self-supervised annotations are used to optimize a self-supervised loss function which helps the pretext classifier to train better. The whole discussion until now has been about training the pipeline with the data available from the source and the target domain. We hypothesize that training a pretext classifier with the same encoder structure helps the proposed method to learn characteristic information about the target domain, thus enabling the encoder to generate better features and the main classifier to achieve better performance at test time when data from the target domain is used to get predictions for the main task aka action detection. Note here that the self-supervision pathway is discarded during the test time and we only generate the predictions for the main task using data from the target domain and analyse the performance of main classifier in contrast with the available ground truth annotations from the target domain.

Some key aspects of this pipeline are discussed in details in this section. Although the standard approach is to use the unlabelled data from the target domain for self-supervision, it is also possible to use only source or both source and target domains to get the supervision needed. We discuss these under Types of Supervision. Next important step in the pipeline is to transfer the knowledge learnt from the self-supervised pathway to the main task on the supervised pathway. In this work, we discuss two such approaches of knowledge transfer between self-supervised and supervised tasks viz. Transfer Learning and Multi-task Learning. Irrespective of the knowledge transfer approach used, it is important to note here that the main and the pretext tasks use the same encoder to transform the input to corresponding features. We look into the details of our Encoder in the subsequent sections. Finally, we conclude our proposal by discussing the action detection and video clip order prediction tasks towards the end.

3.2 Types of Supervision

We refer to Fig 3.2 for the discussion in this section. In the case of Source Supervision (SS), we train our main task using data (X_tr) and annotations (Y_tr) from the Source domain. The pretext or auxiliary task is also trained on data (X_tr) from the source domain however, note that we don't need annotations from the dataset in this case rather the pipeline generate the annotations on its own. At test time, we test the performance of our main task on the test data from the target domain (X_val) and compare the predictions with the ground truth label available from the target domain (Y_val). On similar arguments, we define Target Supervision (TS) when the auxiliary task is trained using data (X_tr) from the target domain, rest all remaining the same as the setup for SS. An interesting conjunction of these two is the case for Joint Supervision (JS). In this setup, the main task is trained as usual using the data (X_tr) and annotations (Y_tr) from the Source domain. However, the auxiliary task is trained with the data (X_tr) from both the source and target domains. The test time evaluation on the target domain remains the same as in all other cases.

3.3 Knowledge Transfer

The information learnt from the pretext task needs to be shared with the main task in order to boost its ability to generalize better outside the source domain. In this work we propose two methods to achieve this- 1. Transfer Learning and 2. Multi-task Learning.

As seen in Fig. 3.3, we train the main and the pretext tasks separately during the transfer learning setup. The pretext task is trained first. We load the data using either type of supervision as discussed in the previous

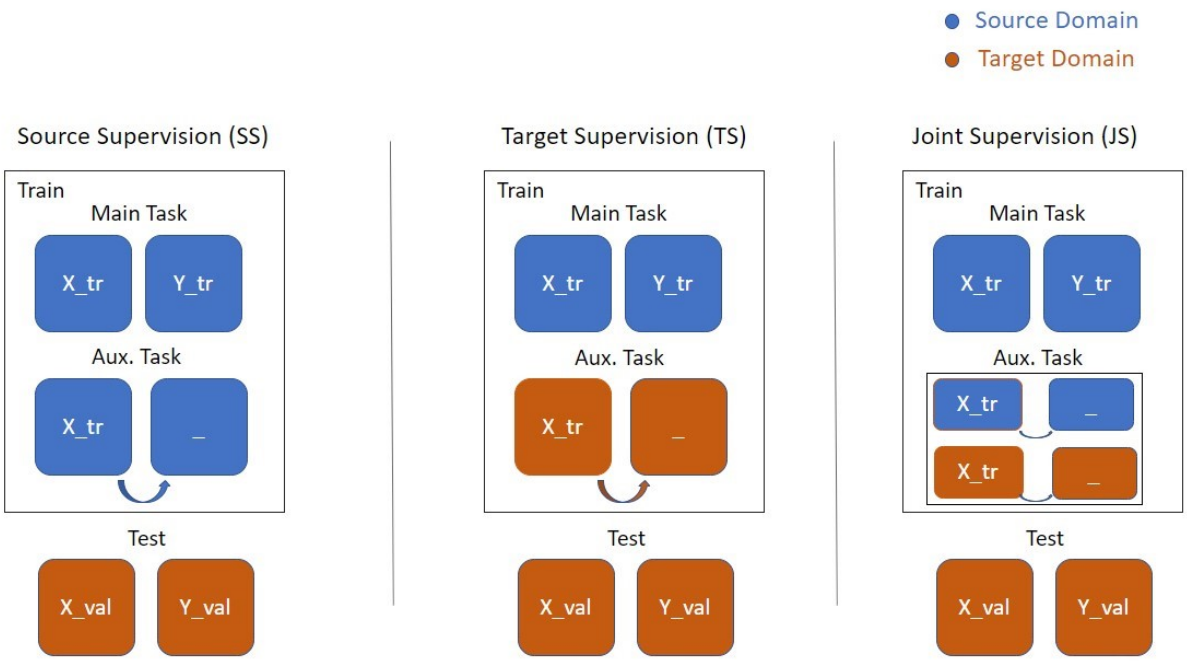


Figure 3.2: Types of supervision. Blue refers to data from source while red refers to data from target domain

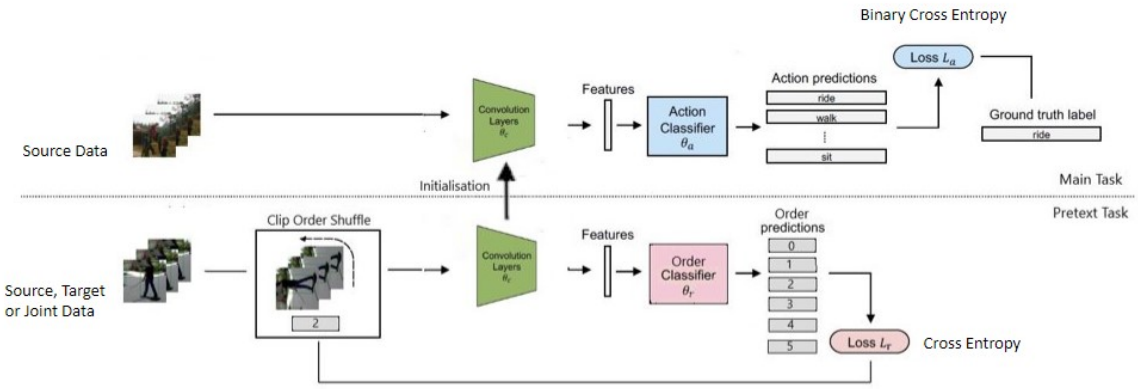


Figure 3.3: Knowledge transfer based on Transfer learning

section. The input is then shuffled using a module named clip order shuffle which also generates the ground-truth label. The shuffled clips are given to an encoder composed of convolution layers. The encoder encodes the input into feature vectors used as input by a classifier. This classifier gives order predictions for the given sample among the class definition of possible ordering of the clips. We utilize cross-entropy loss for optimizing the performance of the order classifier. Once the training is completed successfully, we use the exact same architecture for encoder for the main task, and initialise it with the weights of the encoder of the pretext task. We hypothesize that the weight initialisation done in this way is better than pre-trained weights on benchmark datasets such as ImageNet etc. After initializing the weights of the encoder for the main task, we train it with roughly the same approach. An input is taken from source domain and processed by the encoder to generate features. These are converted into prediction among classes of action by an action classifier. These predictions along with the available ground-truth from the source domain are used to optimize the supevised loss function, binary cross entropy in our case. Once the training for the main task is finished, we evaluate its performance on the test sample from the target domain.

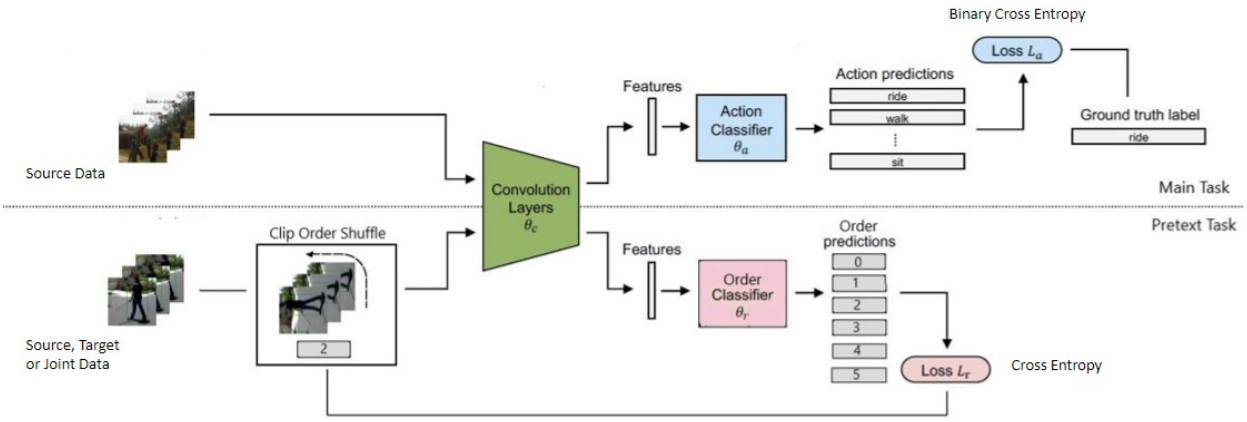


Figure 3.4: Knowledge transfer based on Multi-task learning

The training procedure for multi-task learning is almost the same as transfer learning with one key difference. As can be seen in Fig 3.4, both the pretext task and the main task share a common encoder θ_c for transforming the inputs to their respective feature representations. Also, the two tasks are trained concurrently i.e. the supervised and self-supervised losses are optimized one after the other in a given iteration. The rest of the processing in each pathways of main and pretext tasks are similar to what we have discussed for the transfer learning case.

3.4 Encoder: SlowFast Architecture

Int this section, we describe in detail the architecture of the encoder network which is based on the SlowFast [14] architecture. We briefly cover the biological inspiration behind the network design, then we discuss the rationale behind different different components of the network and conclude the section with the nature of input and output of the encoder.

3.4.1 Motivation

The SlowFast architecture builds upon the biological studies on the retinal ganglion cells in the primate visual system [11], though admittedly the analogy is rough and premature. These studies found that in these cells, about 80% are Parvocellular (P-cells) and roughly 15-20% are Magnocellular (M-cells). The M-cells operate at high temporal frequency and are responsive to fast temporal changes, but not sensitive to spatial detail or color. P-cells provide fine spatial detail and color, but lower temporal resolution, responding slowly to stimuli. The encoder structure is analogous in that: (i) it has two pathways separately working at low and high temporal resolutions; (ii) the Fast pathway is designed to capture fast changing motion but fewer spatial details, analogous to M-cells; and (iii) the Fast pathway is lightweight, similar to the small ratio of M-cells. In terms of visual recognition, all spatiotemporal orientations are not equally likely, and hence then there is no reason to treat space and time symmetrically, as is implicit in approaches to video recognition based on spatiotemporal convolutions [6]. The architecture should rather be factored into spatial structures and temporal events separately. Let us consider an example in the context of visual recognition. The categorical spatial semantics of the visual content often evolve slowly. For instance, waving hands do not change their identity as "hands" over the span of the waving action, and a person is always in the "person" category even though he/she can transit from walking to running. So the recognition of the categorical semantics (as well as their colors, textures, lighting etc.) can be refreshed relatively slowly. On the other hand, the motion being performed can evolve much faster than their subject identities, such as clapping, waving, shaking, walking, or jumping. It can be desired to use fast refreshing frames (high temporal resolution) to effectively model the potentially fast changing motion.

3.4.2 Network design

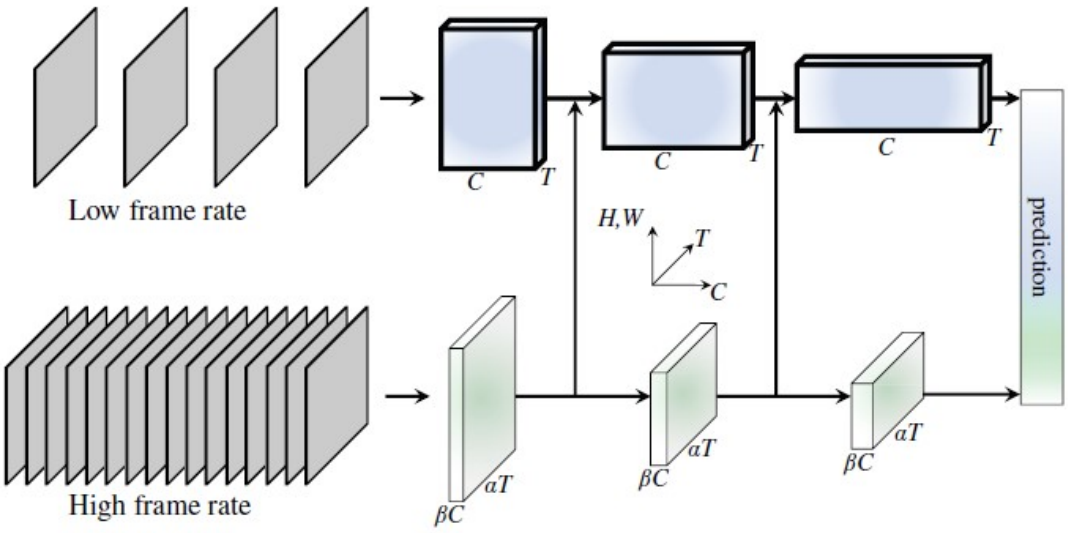


Figure 3.5: Encoder Architecture as mentioned in SlowFast [14]

The encoder architecture in Fig 3.5 is designed to process frame sequences of length $T = 64$. The number of frames processed by one path however differs. The slow pathway samples frames with a large temporal stride of $\tau = 8$, while the fast pathway samples with a relatively smaller temporal stride $\tau/\alpha = 2$,

with $\alpha = 4(\alpha > 1)$ being a tunable parameter. Therefore, the slow pathway operates on $T/\tau = 8$ frames while the fast pathway gets a factor of α more frames, i.e. 32 and therefore more detailed and faster temporal information. This imitates the faster conduction velocity of the M- type cells. The high temporal resolution in the fast pathway is obtained throughout the entire network. There is no temporal pooling until after the last convolution layers. Additionally, the pathways differ in terms of capacity. This is achieved by introducing different channel numbers for the two pathways. It is steered by the channel ratio parameter $\beta(\beta < 1)$. The number of channels in the fast pathway is $\beta = 8$ times the number of channels in the slow pathway throughout all convolution layers. This relatively higher channel capacity in the slow pathway results in more FLOPs and therefore goes in hand with the prevalent amount of P-type cells in the primate visual system. PySlowFast delivers an implementation of several different backbone architectures as C3D, I3D, X3D and ResNet. The backbone used in this project in both pathways is a ResNet-101.

Three different fusion techniques have been suggested between the two pathways by Feichtenhofer et al. [14]. In this work, we only use the time-strided convolution approach. The time-strided convolution removes the frame number multiplicity α of the fast stage and leads to the same temporal reach at the fusion output as in the slow pathway. Before the fusion the fast channel has $1/\beta$ less channels than the slow pathway. To reduce this large channel discrepancy, the fusion layer maps from C/β to $2 \times C/\beta$ channels. This convolved feature map from the previous layers fast pathway is then appended to the feature map of the next slow stage along the channel dimension. The fusion is kept unidirectional throughout the entire network.

An input sample to the encoder is a clip of 64-frames image sequence along with one bounding box annotation for the central frame which is interpolated to the rest 63 frames in its neighborhood. These frames are sampled at a frame-rate of 30 fps. Each sample can have multiple annotations since one person can perform several actions at the same time. The middle frame of the 64 frame sequence is referred to as the key-frame and it is the frame for which the human bounding boxes were created. Note that features are extracted together for all samples in one key-frame.

3.5 Action Detection

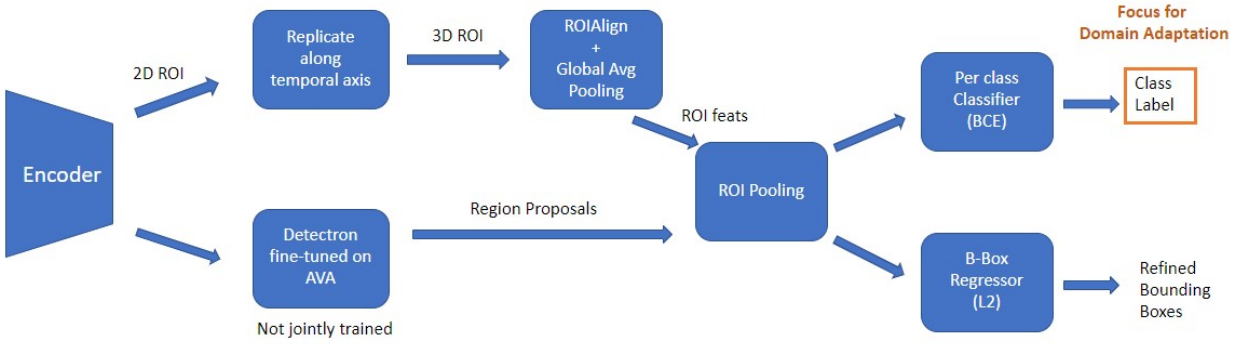


Figure 3.6: Action Detection Pathway

The output of an encoder is followed by an action detection pipeline (Fig. 3.6) to convert the given feature representation of the input clips into corresponding predictions for labels for action class and coor-

dinates for the predicted bounding box.

The action detector architecture is similar to Faster R-CNN [56] with minimal modifications adapted for video. As explained in the previous section, the encoder is based on SlowFast architecture. The spatial stride of res5 is set to 1 instead of 2. A dilation of 2 is used for its filters. This step results in an increase in the spatial resolution of res5 by a factor of 2. The Region-of-Interest (RoI) features [19] are extracted at the last feature map of res5. Each 2D RoI at a frame is first extended into a 3D RoI by replicating it along the temporal axis, based on the method discussed in [23]. Then the RoI features are calculated by RoIAlign [25] spatially, and global average pooling temporally. The RoI features are then max-pooled and fed to a per-class, sigmoid-based classifier for multi-label prediction for action classes. Regarding detection of bounding boxes, previous works have used pre-computed proposals [23]. The region proposals are computed by a person detector that is not jointly trained with the action detection models. The person detection model used in this work is trained with Detectron [46]. It is pre-trained on ImageNet and the COCO human keypoint images [44]. This detector is then fine-tuned on AVA dataset for person detection. The person detector produces 93.9 AP at an IOU level of 50 on the AVA validation set. Then, the region proposals for action detection are detected person boxes with a confidence of > 0.8, which has a recall of 91.1% and a precision of 90.7% for the person class [14].

3.6 Video Clip Order Prediction

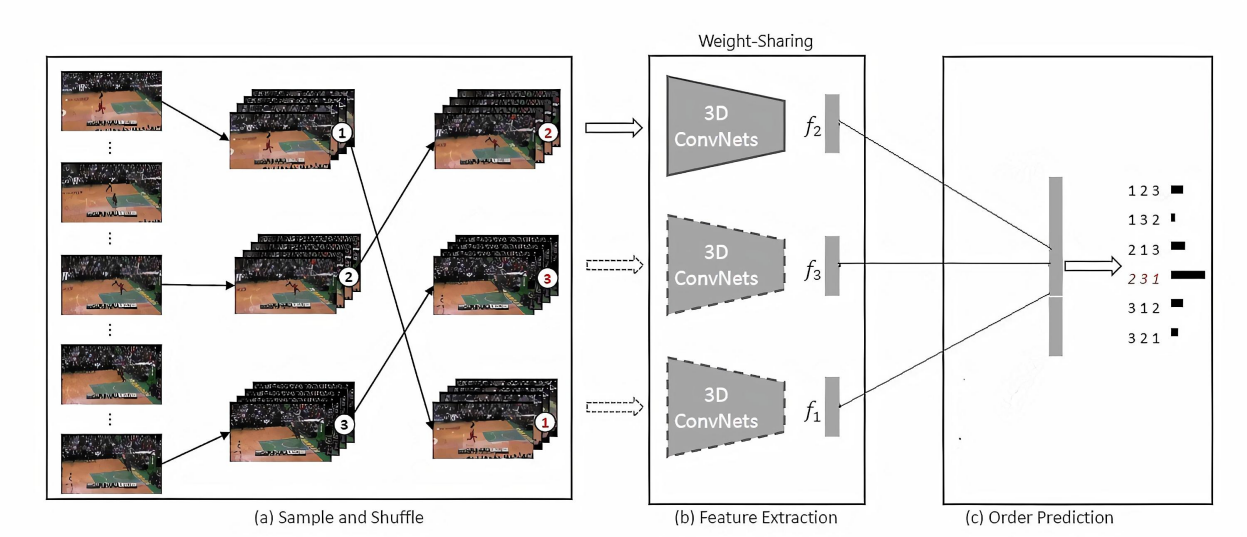


Figure 3.7: Overview of Clip Order Prediction Framework: (a) Sample and Shuffle: Sample non-overlapping clips and shuffle them to a random order. (b) Feature Extraction: Use the 3D ConvNets to extract the feature of all clips. (c) Order Prediction: The extracted features are pairwise concatenated, and fully connected layers are placed on top to predict the actual order. The dashed lines mean that the corresponding weights are shared among clips. The work is inspired from [77]

Video clip order prediction is utilised as a self-supervision to help learn the weights of the encoder that can generalise better to the target domain. Figure 3.7 presents the overall framework, which mainly comprises of three procedures. In sample and shuffle, several clips are uniformly sampled and shuffled; in

feature extraction, 3D CNNs are used to extract features for the clips, and all 3D CNNs used here shared same weights; in order prediction, we resolve the task via classification. A clip is defined as as combination of continuous frames sampled from the video with the size $c \times l \times h \times w$, where c is the number of frame channels, l refers to number of frames, h and w refer to the height and width of frames. A 3D convolution kernel has the size $t \times d \times d$, where t is the temporal depth and d is the spatial size. A tuple of ordered clips is defined as $C = (c_1, c_2, ..., c_n)$, and the features extracted by 3D CNNs are represented as $F = (f_1, f_2, ..., f_n)$. Elements in a tuple are arranged in chronological order.

3.6.1 Sample and Shuffle

Starting with N different clips, there are $N!$ possible orders of arranging them. This number grows rapidly as we increase the number of clips N. For instance, $7! = 5040$, and the classification tasks already becomes incomprehensible. Some prior works have proposed ways to select particular orders from all possibilities, either in a deterministic or adaptive way [50, 3]. However, it is important to remember here that clip order prediction is just a proxy task and if we want to focus on the learning of the encoder, the task should be solvable. Otherwise, if the whole task is too hard to solve, almost nothing can be learned. We therefore restrict the number of clips between 3, which makes the number of order classes 6 which is easy to analyse. For tuple length, 3 clips per tuple are reasonable since 2 clips order prediction is more like an order verification, while more than 3 tuples become a relatively hard task. This decision is also based on the conclusion from [50] that a good self-supervised task is neither too simple nor highly ambiguous. The clips are then sampled uniformly from the video with an interval of m frames. It is also ensured that the clips are non-overlapping, in order to avoid the situation that the whole framework tackles the task by comparing lower characteristics such as texture and color. For an extreme case, if the clips are overlapped by 1 frame, a simple comparison of frames pixels can solve the task. After the clips are sampled, they are shuffled to form the input data while the actual order is served as the target. The shuffle step is random, and no particular permutations are preferred. During the training step, the number of generated samples belonging to each order class is roughly the same as seen in Fig 3.8.

3.6.2 Feature Extraction

Once the clips are shuffled, we use the encoder discussed previously to extract features for each clip. The same encoder is used for all clips in one tuple, as shown in Fig 3.7. We refer the reader to the the discussion on Encoder for further details.

3.6.3 Order Prediction

We formulate the order prediction as a classification problem. The input is a tuple of clip features, and the output is a probability distribution over different orders. Given the extracted features, we perform the following operations:

$$a = W(f_1||f_2||...||f_N) + b$$

$$p_i = \frac{exp(a_i)}{\sum_{j=1}^{C} exp(a_j)}$$

where $||$ means the concatenation of vectors, W and b are the parameters of the fully connected layer, a refers to logits and p_i is the probability that the order belongs to class i. For instance, tuple containing 3 clips, after shuffling, we get $C = (c_1, c_2, c_3)$ and the corresponding extracted features $F = (f_1, f_2, f_3)$. These vectors

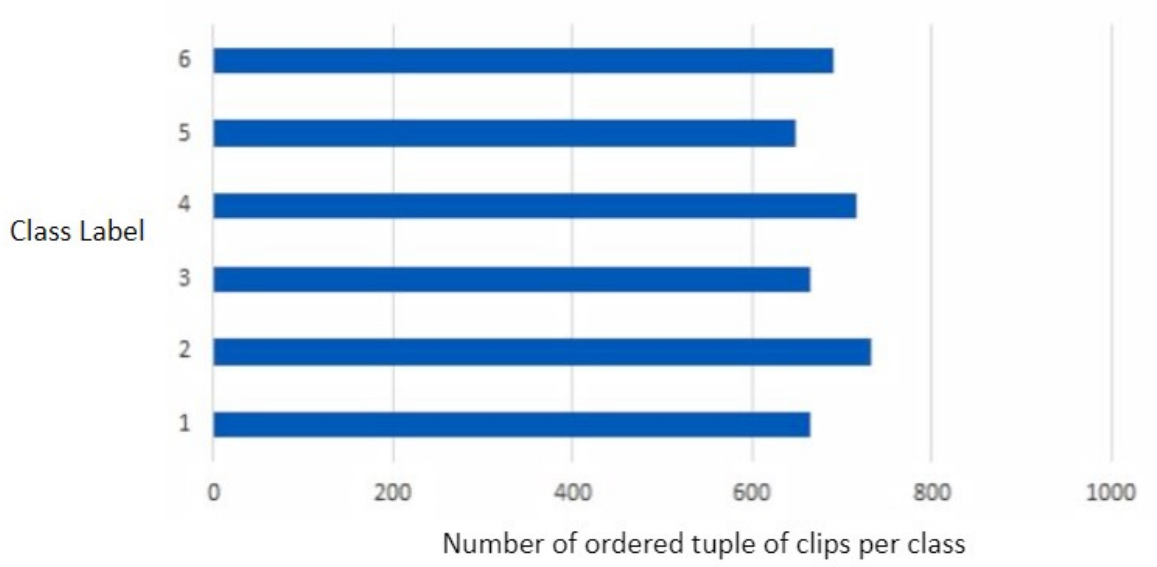


Figure 3.8: Distribution of ordered tuple of clips per class. Not that due ti randomly shuffling, the distribution of generated samples belonging to each order class is roughly uniform

are concatenated and a fully-connected layer with softmax is applied to output the final prediction. The target classes are permutations of $(1, 2, 3)$, one of which is the actual order, say $(2, 3, 1)$. The correctness of the prediction is measured using cross-entropy loss function as:

$$L = -\sum_{i=1}^{C} y_i \times log(p_i)$$

where y_i and p_i are the probability of the sample belonging to order class i in ground-truth and prediction, and C is the number of all possible orders. The loss L is then back-propagated to optimize the whole framework. When the framework is trained to predict the order of clips, the encoder gets trained to extract the meaningful features of clips meanwhile.

Chapter 4

Experiments and Results

4.1 Datasets

The aim of this work is to propose a method for domain adaptation using self-supervision. In order to assess the UDA setting, we require two datasets. We leverage AVA and Kinetics datasets interchangeably as source and target domains to evaluate our proposed method. We will have a closer look at the propoerties of the benchmark datasets followed by an understanding of the need to use a reduced version of them.

4.1.1 AVA 2.2

AVA stands for atomic visual action and consists of 430 densely annotated 15-minute clips with 80 visual actions. In total there are roughly $1.62M$ action labels. Each person can perform multiple actions at the same time. Actions are localized both in space and time. We refer the reader to [23] for a detailed discussion on dataset curation and annotations. We use version 2.2 of the released annotations in this work. We follow the same steps for dataset preparation as followed in the work on SlowFast [14]. First, 30min videos are downloaded from an Amazon server. Then, a 15min clip along with annotations are extracted from these. The videos need to be converted into frames since this is the pipeline input. Given a dense per second annotation, extraction of frames can be done efficiently. At a frame rate of 30fps, approximately 27k frames per video are extracted.

4.1.2 Kinetics-700

Kinetics700 is a huge dataset consisting of more than 650k videos among 700 action classes. Each sample is a video clip that lasts for 10s and contain a single action annotation for the entire clip. Due to absence of any spatial localisation, this dataset has thus been limited to applications in action recognition [5]. Recently, Kinetics700 dataset has been annotated for action detection i.e. the spatial localisation aspect is included. Li et al. [9] explain the annotation process in detail and provide detailed statistics of both AVA and Kinetics. For one key-frame of each of more than 230k Kinetics700 videos AVA annotations are provided, i.e. human bounding boxes and 80 class action labels. The annotations are publicly available at [7].

Availability of the videos is a big challenge in working with Kinetics dataset. As a solution, Kinetics700 download repository provides documentations for failed downloads and we used [20]. The annotation protocols for both AVA and Kinetics datasets are similar in the AVA-Kinetics dataset, thus the same data preparation scripts are used as discussed for AVA dataset in the previous section. A key insight here is the duration of samples which is not strictly 10s but rather between 10s and 3min. To maintain uniformity in

	AVA		Kinetics	
	Train	Val	Train	Val
Annotations	5,000	1,000	5,000	1,000
Unique Boxes	3,847	832	4,121	790
Key-frames	2,972	700	3,140	639
Videos	27	6	3,140	639
Frames	729,814	162,182	565,200	115,020

Table 4.1: Reduced dataset

the extracted frames, a fixed number of frames were sampled around the central annotated frame. We chose 6s around the key frame to sample frames assuming that the same sampling frequency as used for AVA i.e. 30fps. The number of frames thus generated i.e. 180 were suitable both in terms of storage and computation power. Video samples where the central frame with annotations is too close to the start and end of the input are excluded to be consistent.

4.1.3 Dataset reduction

As seen in previous sections that working with the entire dataset is infeasible for both AVA and Kinetics dataset. In order to cater to time and resource constraints, we conducted the experiments on a reduced set of classes and number of samples. We have written scripts to downsize the dataset to an arbitrary number of classes and samples within each. We have worked with 10 classes in this project- ('watch (a person)' (80), 'talk to (e.g., self, a person, a group)' (79), 'listen to (a person)' (74), 'sit' (11), 'carry/hold (an object)' (17), 'walk' (14), 'touch (an object)' (59), 'bend/bow (at the waist)' (1), 'lie/sleep' (8), 'ride (e.g., a bike, a car, a horse)' (49)), where the number in the brackets refer to the ordering of the class label followed by SlowFast [14]. The choice of classes is majorly focused on the most frequently occurring actions in the Armasuisse dataset for which we need to create a demo in the beginning of the project. The arguments to be provided to the scripts include the number of samples per class that need to be present in the training and validation sets respectively. We have majorly worked on the experimental evaluation using 500 samples per class for training and 100 samples per class for the validation. For more details, please refer to Table 4.1

4.1.4 Data pre-processing

Data pre-processing step can vary for training and testing stages. During training, for every frame in a given clip, we scale the shorter dimension randomly between the lengths of 256 and 320, and adjust the longer dimension keeping the aspect ratio intact. As suggested in [14], this step is used to introduce scale-invariance. The input frame is then cropped to a fixed dimension of 224×224. As for the data augmentation, we use horizontal flipping of individual frames with a probability of 0.5.

During validation step, we cut down the shorter dimension of a frame to 224 and cut a centre crop of 224×224. Sometimes the ground-truth bounding boxes can get cropped in this process which needs to be taken care of. The pixel intensity is normalized to values between 0 and 1. The inputs are normalised using Z-score normalization (subtracting mean and dividing by standard deviation).

4.1.5 Data-loader

The data is loaded into each SlowFast pathway as sequence of frames or clips. A clip contains an array of bounding box coordinates, and for each box a label vector that is one-hot encoded for all the classes occurring in the respective boxes. The data-loader for the auxiliary task is created inheriting from data-loader defined for the action detection task. A given input clip is first divided into sub-clips. A random number is generated between 0 and the total number of classes for order prediction. Based on this number, subclips are shuffled and arranged along the temporal axis in a particular configuration. Note that since the randomness is incorporated in the temporal properties and not spatial characteristics, we don't require to change the bounding box annotations here. The shuffled-data is loaded into the slowfast pathway in the similar manner as the input for the action detection task.

4.2 Training setup

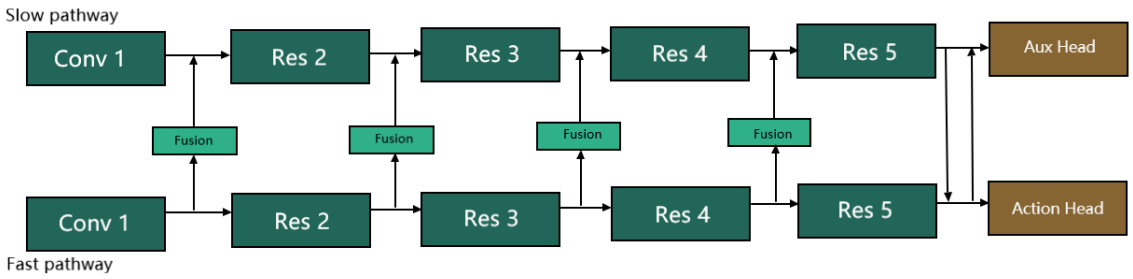


Figure 4.1: Conceptual model of Slowfast highlighting different pathways and lateral connections. In a Multi-task setting, an additional auxiliary head (Aux Head) is added for training the self-supervision task.

As discussed in the previous section, the SlowFast network [14] has two different convolutional pathways based on different frame rates, number of channels etc. The model is illustrated with a conceptual block diagrame in Figure 4.1. These pathways are fused together through lateral connections that also contain trainable parameters. For a more detailed look at the number of parameters in each layer, refer to Fig. 4.2

The number of parameters in the fully connected layer for action classification aka Action Head in fig 4.1 depends on the number of classes at the output. The feature map after the RoI align has 20304 dimensions and mapping this to 10 classes results in 230050 weights. For 80 classes, this number increases to 1240400. The given SlowFast architecture can be used at it is for experiments on Transfer Learning setup. However, we introduce an additional head for learning the self-supervised task, called Aux Head, see fig. 4.1. It shares the same convolution layers with the action detection part, just that it classifies an input data for a different objective. We also analyse the number of FLOPs performed in a single forward pass. For the configuration of Multi-task setting, approximately 112G FLOPs and 225M activations are calculated in a given forward pass. This analysis definitely depends on a given choice of α and β. In this project we kept the α value as 4 and β as $\frac{1}{8}$.

Layer	Parameters			Output dimensions ($C\times T\times S^2$)		
	Slow	Fast	Fusion	Slow	Fast	Fusion
Input				$3\times8\times224^2$	$3\times32\times224^2$	
Conv 1	10	6	1	$64\times8\times56^2$	$8\times32\times56^2$	$16\times8\times56^2$
Res 2	221	5	10	$256\times8\times56^2$	$32\times32\times56^2$	$64\times8\times56^2$
Res 3	1′261	27	41	$512\times8\times28^2$	$64\times32\times28^2$	$128\times8\times28^2$
Res 4	35′509	461	164	$1'024\times8\times14^2$	$128\times32\times14^2$	$256\times8\times14^2$
Res 5	21′125	318		$2'048\times8\times14^2$	$256\times32\times14^2$	
Per Pathway	58′126	817	216	2′048	256	
Total		**59′159**			**2′304**	
	Action		Pretext	Action		Pretext
Fully connected	23		9	10 classes		6 classes

Figure 4.2: Number of parameters in units of 1k and dimensions of feature maps for each layer in SlowFast architecture.

4.2.1 Weight Initialisation

To initialise the weights of the model, we take the pre-trained weights from the SlowFast repository which are provided for the objectve of action detection. Note that these weights correspond to a model trained with AVA dataset, since Kinetics dataset is used only for action recognition task [14]. Some experiments have also been conducted with Kinetics action recogntion weights. These weights can be accessed at [21]. The weights corresponding to AVA dataset, SlowFast architecture, R101 depth, frame length x sample rate of 8x8, AVA version 2.2 and a performance MAP of 29.1 is used for the experiments in this work. For kinetics only experiments, we used the weights from the same setting but a pretrained model of Kinetics600. An important point here is to note that the action detection weights (model) are obtained by training a model which had been initialised with action recognition (Pretrain model) weights. Thus it was a challenge to achieve fully unsupervised domain adaptation setting.

Some experiments have been performed using a random initialisation technique proposed in [26]. The weights of 3D batch normalisation are initialised with 1 and the biases with 0. The fully connected layers are all initialised with a standard deviation of 0.01 and 0 bias. Given the resource constraints, it was not pragmatic to train the entire SlowFast model from scratch. Instead, we decided to keep most of the initial layers frozen while training only a few deeper layers close to the action/aux head. All the frozen layers are initialised with FB pretrained weights. The unfrozen layers can either be FB initialised or randomly initialised based on the experiment being conducted. We provide a visualisation of the different initialisation strategies in Fig 4.3. The dark green blocks are initialised with FB pre-trained weights and are kept frozen throughout the entire experiment. Light green blocks are initialised with pre-trained weights and fine-tuned. White blocks are randomly initialised and therefore also trained. Different model configurations help evaluate the performance of our method under different setups. In an idealistic scenario, training models from scratch i.e. the weights of all the layers would have provided accurate insights about domain adaptation. However, we used these different initialisation steps to reduce the number of trainable parameters so that necessary experiments can be carried out with the GPU and time constraints. Also, instead of using a different smaller network would have worked however, we argue that freezing a big, provably state of the

art model for action detection is a better way of assessing performance of domain adaptation. The argument here is- if self-supervision can improve domain adaptation on action detection with only a few trainable layers, it is expected to show even more promising results when trained from scratch.

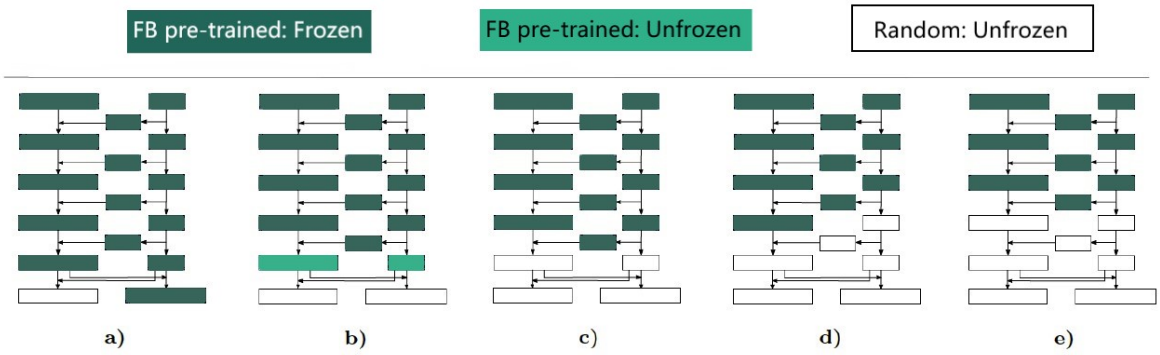


Figure 4.3: Different weight initialisation strategies based on frozen/unfrozen layers and FB/random weights.

4.2.2 Metrics

We used Stochastic Gradient Descent (SGD) as the optimizer for training our models. The learning rate is updated on the basis of a half-period cosine schedule. In general we decreased the learning rate from 0.1 to 0. The batch size is adjusted according to the GPU memory available. We chose a sigmoid activation rather than a soft-max given the fact that the dataset contains multiple labels per sample and it is more practical to assess the objective - a given class or not. Binary Cross Entropy (BCE) is used as the loss function for the action detection pathway. For a given batch $m \in 1, 2, ..., M$, we define the BCE as:

$$l_m = -w_m \cdot \frac{1}{N} \cdot \sum_{n=1}^{N} (y_{m,n} \cdot log(x_{m,n}) + (1 - y_{m,n}) \cdot log(1 - x_{m,n}))$$

where the weights w_m for different batches are kept as 1. The loss is calculated for each batch separately during training. Here $x \in \mathbb{R}_{M \times N}$ per class prediction for each batch while $y \in \mathbb{R}_{M \times N}$ is the one-hot encoded label ground truth. The total loss is an average over all the batches:

$$l(x, y) = \frac{1}{M} \cdot \sum_{m=1}^{M} l_m$$

For evaluation of the performance of the action detection pipeline, we used Mean Average Precision (mAP). To calculate this, we first calculate the Precision and Recall for each class based on a threshold value. If TP, FP and FN stand for True Positive, False Positive and False Negative respectively, the precision and recall are defined as:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

We then use these values to draw the Precision-Recall (PR) curve for each class for a given threshold. The area under the curve is called Average Precision (AP). mAP is then calculated by taking the mean over all classes of the Average Precision values calculated for each class.

In order to correctly identify the bounding boxes during the localisation task of action detection pipeline, we use Intersection over Union (IoU) as the metric. If the ground truth and predicted bounding boxes are denoted by B_{gt} and B_{pr}, the IoU is defined as:

$$IoU = \frac{B_{gt} \cap B_{pr}}{B_{gt} \cup B_{pr}}$$

A given prediction is labelled as True Positive if $IoU > 0.5$ and the prediction for action class is same as the ground truth. A sample is called False Positive if the label for action class is correct but the bounding box is either duplicate or the IoU of the detection is less than 0.5. Similarly, a sample is False Negative if the predicted label is incorrect but the IoU value is greater than 0.5.

The auxiliary task is a separate classification problem (order prediction) than the action detection. Thus, we introduce an additional fully connected layer, called Aux Head that maps the features of the penultimate layer to 6 classes. Since we desire mutual exclusivity between the classes here, we opt for a softmax activation unlike the action classification task where a sample can belong to multiple categories. One similar arguments, we use Cross Entropy (CE) loss. For a given batch $m \in 1, 2, ..., M$, we define the CE as:

$$l_m = -log(\frac{e^{x_m \cdot y_m}}{\sum_{n=1}^{6} e^{x_m \cdot y_n}})$$

Here, $x \in \mathbb{R}_{M \times N}$ per class prediction for each batch while $y \in \mathbb{R}_{M \times 1}$ is the one-hot encoded label ground truth. The total loss is an average over all the batches:

$$l(x, y) = \frac{1}{M} \cdot \sum_{m=1}^{M} l_m$$

We use the classification accuracy as well as confusion matrices to evaluate the performance of the clip order prediction task.

4.3 Baseline

4.3.1 Pre-trained FB model

In order to have a fair comparison, we first evaluate the performance of a pre-trained FB model on our reduced dataset. We use the SlowFast model with R101 depth (ResNet backbone), trained on AVA v2.2 (after pre-training on Kinetics600), and reaching a mAP of 29.1%. This particular model is used as baseline since we use the weights of this pre-trained model as initialization for majority of the experiments. We will refer to this model as FB pre-trained. The performance of this model is evaluated twice, once on the ground-truth bounding boxes (annotated by humans) and additionally also on the predicted bounding boxes of the reduced dataset. All evaluations are made using the evaluation module of PySlowFast with a batch size of 1.

The 10 chosen classes are nicely detected as can be seen though the reasonable mAP values reported in Table 4.3. AP values obtained for validation using ground-truth boxes are reasonably better than the ones based on predicted boxes. One reason could be that the errors in prediction of bounding boxes can

Class tag	Class label
C1	bend/bow (at the waist)
C2	carry/hold (an object)
C3	lie/sleep
C4	listen to (a person)
C5	ride (e.g., a bike, a car, a horse)
C6	sit
C7	talk to (e.g., self, a person, a group)
C8	touch (an object)
C9	walk
C10	watch (a person)

Table 4.2: Tags for different classes

Dataset	Tag	B-Box	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	mAP
AVA	FB	PR	44.5	35.4	66.1	30.6	60.9	24.33	49.8	42.4	68.8	16.38	43.94
		GT	81.2	44.5	95.16	50.3	90.8	35.11	53.04	62.6	85.8	63.9	**63.9**
KIN	FB	PR	38.5	15.1	56.5	17.4	50.6	21.2	42.3	30.6	26.6	11.9	31.1
		GT	62.2	25	85.2	49.8	85.1	37.9	63.1	45.2	71.9	35.9	**56.15**

Table 4.3: Comparison of AP (class-wise) and mAP values using predicted (PR) and ground-truth (GT) bounding boxes. The results are obtained by evaluating a pre-trained FB model on both datasets. Highlighted figures are treated as baselines for further experiments.

propagate to classification too. Predicted bounding boxes for the AVA key-frames are provided by FB. These are obtained by a Detectron2 model. The predicted boxes for the Kinetics700 dataset were obtained as a part of this work. We used Mask R-CNN model to detect the bounding box. We used Detectron2 for this step. All the key frames of Kinetics 700 dataset were inferred and the bounding boxes detected around humans are saved in file following the annotations protocol as used by AVA. Due to superior performance on both the datasets, we would use ground truth bounding boxes as baselines for the rest of our experiments.

4.3.2 Fine-tuning on reduced dataset

We now fine-tune the FB pre-trained model on our reduced dataset and subsequently evaluate its performance. Note here that the pre-trained models provided in the PySlowFast repository were trained for 80 classes and thus in order to use them as a fair baseline for our work, fine-tuning them for 10 classes is rather important. The weight initialisation of the model is done according to the configuration mentioned in Fig 4.3 (b). Only Res5 and fully connected layer for action classification head are trained for 15 epochs with an

initial learning rate of 0.1 and batch size of 4. We can see the training curves for the models fine-tuned on AVA and Kinetics dataset in fig. 4.4. The validation mAP achieved on both the datasets is in proximity to the values reported in Table 4.3. Thus, we can assume safely that the reduced dataset is sufficient to train the last few layers of the SlowFast model. Fixing weights in the convolutional layers 1-4 helped reduce the number of trainable parameters to 36% of the entire network capacity thus making the model less prone to over-fitting.

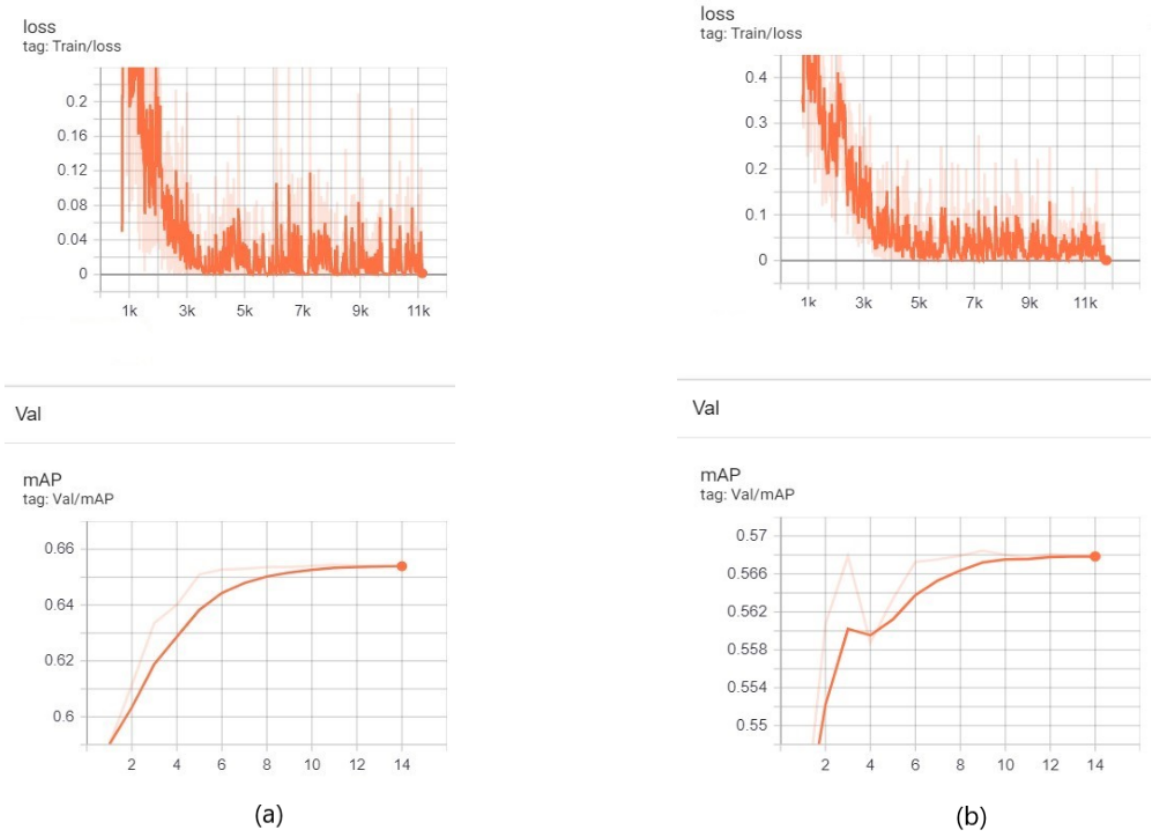


Figure 4.4: Training plots and Validation mAP for fine-tuning FB pre-trained model on reduced datasets of (a) AVA and (b) Kinetics

Once these models are trained, we evaluate their performance on same or different datasets to set the baselines for Within-Domain (WD) and Cross-Domain (CD) performance without any self-supervision. The detailed evaluation of the fine-tuned models is shown in Table. 4.4. As evident, evaluating a model on a dataset it has been trained on comes with a higher performance as compared to evaluating it on a different dataset. Such a behavior is observed for both the scenarios when AVA and Kinetics were used as Source and Target respectively, and vice-versa. This baseline posits the presence of a challenge of domain shift across datasets for the objective of action detection and the rest of the experiments in this work will try to address that.

Tgt	Src	Tag	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	mAP	Perf. drop
AVA	AVA	WD	75.5	33.6	87.7	55.5	95.1	63.1	72.1	64.5	82.9	40.0	67.03	**-16.8%**
	KIN	CD	63.4	31.8	83.5	29.4	85.7	34.5	67.9	53.1	83.6	25.1	55.8	
KIN	KIN	WD	69.8	30	90.5	46.2	90.3	36.7	66.7	43.3	73.8	40	58.76	**-18.1%**
	AVA	CD	49.8	20.4	80.14	37.8	83	28.9	51.5	37.1	58.6	32.5	48.1	

Table 4.4: Fine-tuned models trained on Source (Src) domains evaluated on different Target (Tgt) domains. As evident, compared to the case of WD, CD shows a significant performance drop thus reinstating the severity of domain shift across datasets for action detection.

4.4 Transfer Learning

The basic idea of transfer learning as explained in the previous sections is to train a self-supervision objective on target domain (or source or both, depending on the type of supervision followed), and use the learnt weights as initialisation for the training of the main objective (here, action detection). Thus it is imperative to train the auxiliary or the pretext task well, so that the downstream task can leverage the required supervision. To train the pretext task, we followed the same training configuration as mentioned previously in Fig 4.3 (b). Note that the model architecture remains the same as in Fig. 4.1, only the Action head is discarded here. For optimizing on different hyper-parameters, we conducted all the experiments on the AVA dataset and used the same optimal values of parameters for training the pretext task on the Kinetics dataset in the end.

4.4.1 Optimizing pretext task

Learning rate

As shown in Fig. 4.5, lower learning rates gave significant improvements in performance for the pretext task. Not only it improved the overall accuracy from 18.2% to 88.9%, the per-class performance saw major boost too. We kept the learning rate fixed to a value of 0.01 for the rest of the experiments.

Number of frames

The SlowFast model mentions the optimal number of frames for the action detection task to be 32. However, using the same for pretext task training resulted in poor performance for one of the classes. Instead, increasing the number of frames in the input clip to 64 improved the overall performance from 76.44% to 88.9% as well as improved the mis-classification happening earlier in certain class samples as seen in Fig. 4.6. For the rest of the experiments, we fixed the number of frames in the input clip as 64 for training the pretext task.

Alpha

The parameter α controls how densely the frames will be sampled by the Fast pathway as compared to the slower one. Since this parameter is crucial for temporal information, we checked its effect on the performance of the clip order prediction task as well. We found that the alpha value of 4 as used by the SlowFast action detection network was optimal for the pretext task too as seen in Fig. 4.7

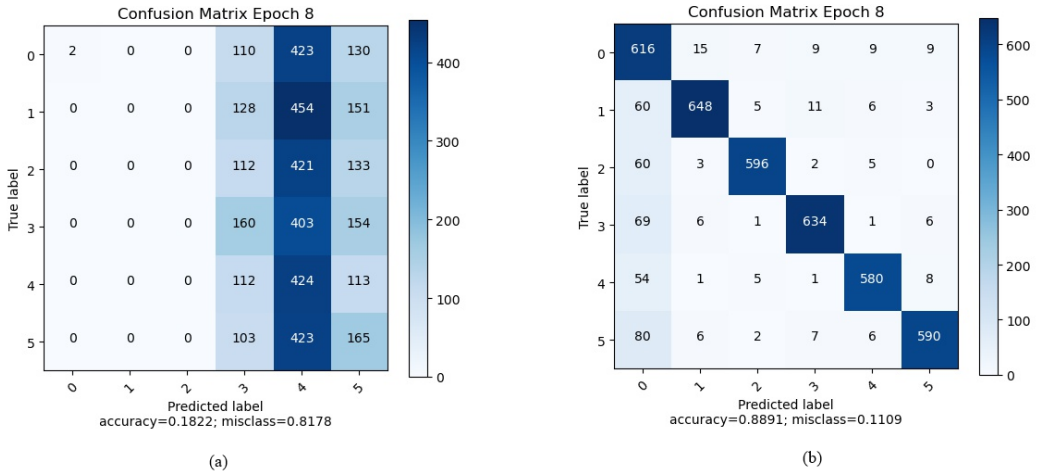


Figure 4.5: Confusion matrices obtained after training the pretext task for 8 epochs with learning rates (a) 0.1 and (b) 0.01

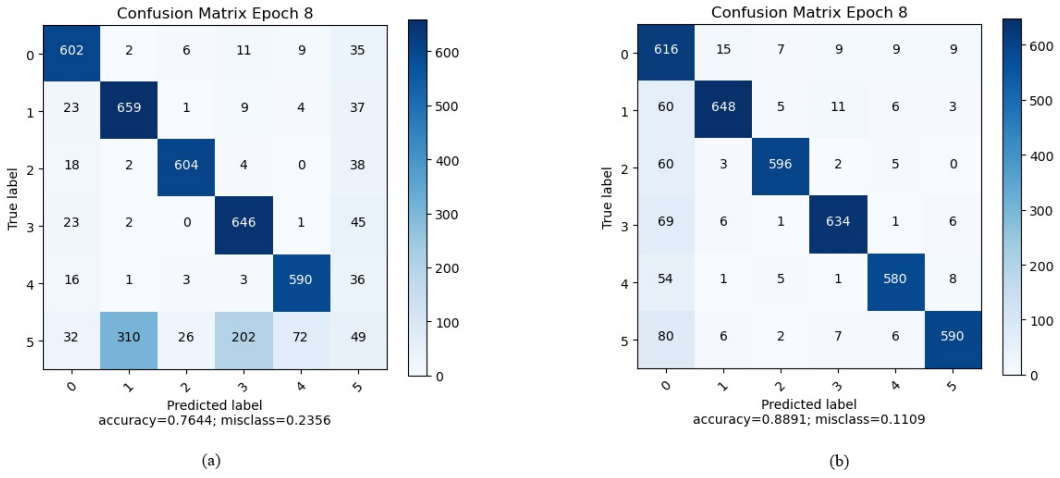


Figure 4.6: Confusion matrices obtained after training the pretext task for 8 epochs with number of frames (a) 32 and (b) 64

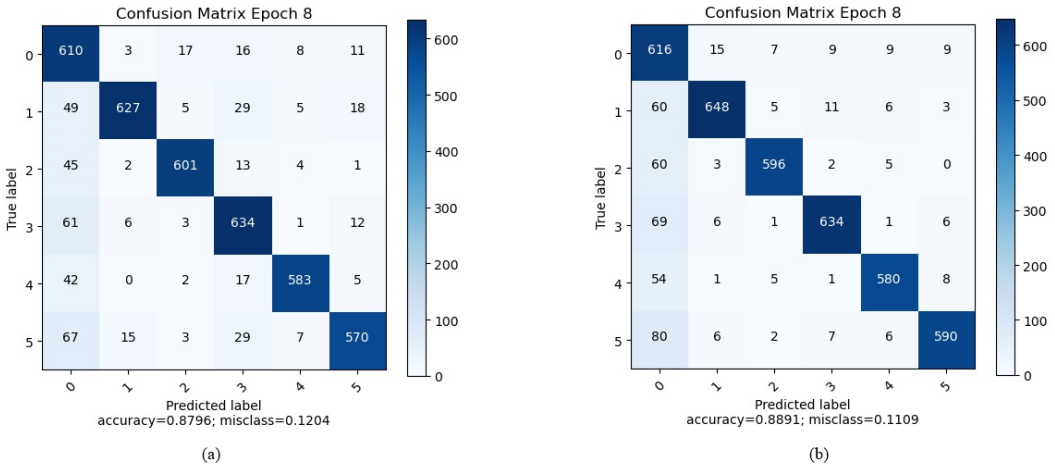


Figure 4.7: Confusion matrices obtained after training the pretext task for 8 epochs with α values of (a) 8 and (b) 4

Based on the above tuning of few major hyper-parameters, and a few minor training level observations, we obtained the final optimally trained pretext model for AVA dataset. We used the exact same hyper-parameters and trained a corresponding model on Kinetics dataset as well. As shown in the curves depicted in Fig. 4.8, the training loss values seem to decrease and converge for pretext task hinting at a successful training. Moreover, we checked the performance of these trained models on a held-out validation set for both datasets. The confusion matrix thus obtained can be seen in Fig. 4.9. The accuracy achieved on AVA and Kinetics were 92.54% and 91.12% respectively. This analysis further validates that the pretext task has been trained optimally over both the datasets.

4.4.2 Knowledge Transfer

The final objective in Transfer Learning-based approach would be to use the supervision from one domain as a starting point to train on the other domain in order to improve the ability to generalise across domains. On similar lines, once we obtained the optimal model trained for pretext task on AVA dataset, we used the weights as an initialisation for the model to be trained for an action detection task. We now will refer to the action detection task as the main task. The clip order prediction is already being referred to as pretext or auxiliary task. Once the model is trained, we test it on a held-out validation set from the Kinetics dataset to evaluate cross-domain (CD) and on AVA validation set to assess within domain (WD) performance of the trained main task. We repeat the same experiment with Kinetics dataset to obtain both CD and WD performances. Note that the configuration explained above resembles the scenario of Source supervision (SS), where both the pretext and main tasks are trained on one dataset (say AVA) and tested on another (say Kinetics). We will first focus our analysis to TL with SS and compare it to the baselines to gain insights. We will discuss other forms of supervisions separately in later sections. As seen in Table 4.5, the mAP acheived by Transfer Learning is higher than the baseline for both the domain configurations. This clearly indicates that self-supervision helps in transferring knowledge across domains. Moreover, in majority of the classes,

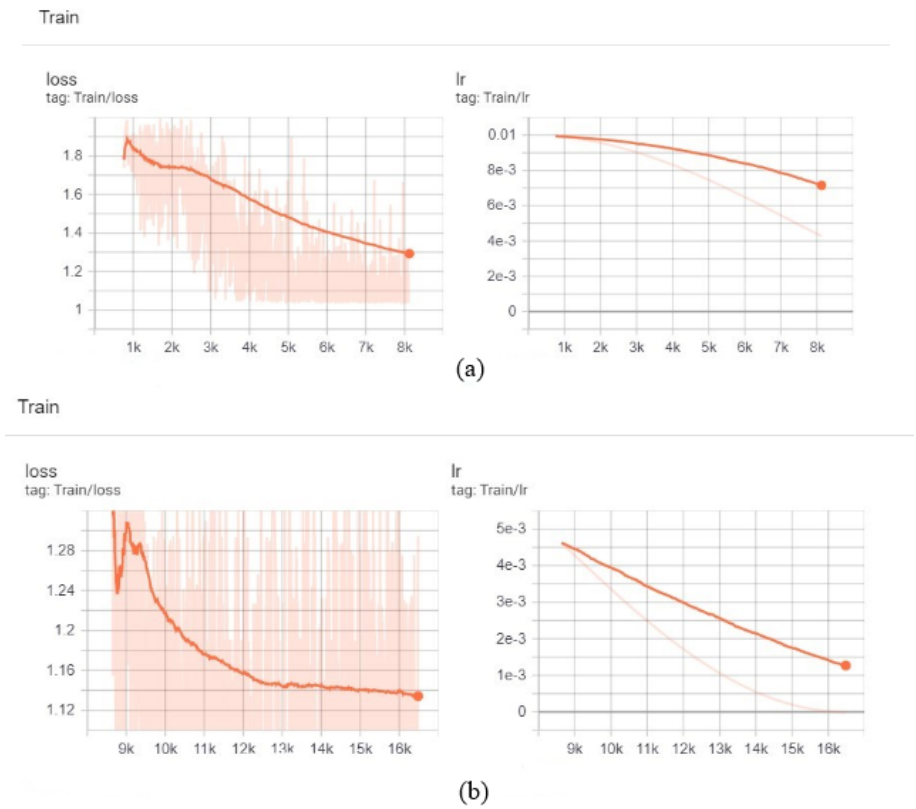


Figure 4.8: Training plots showing the decrease in training loss (left) and updates in learning rates (right) per batch while training the pretext task of clip order prediction on (a) AVA and (b) Kinetics datasets.

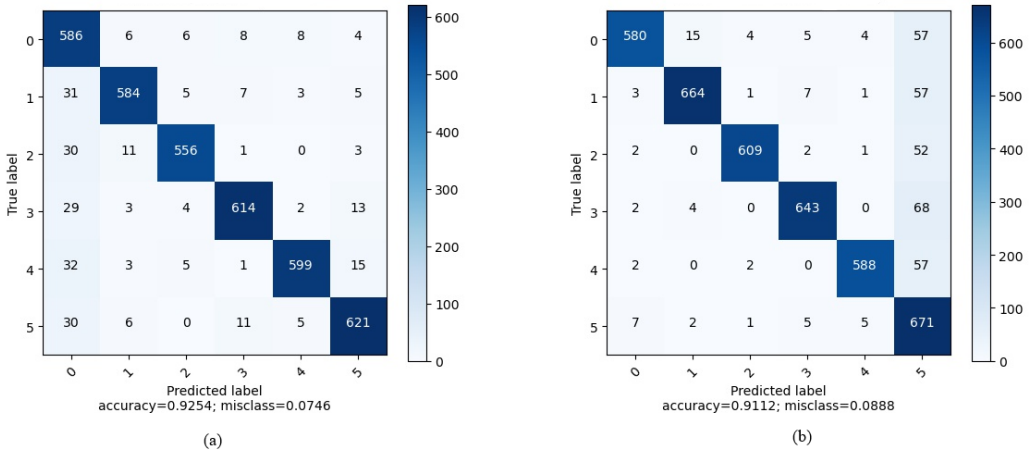


Figure 4.9: Confusion matrices obtained during validation for (a) AVA and (b) Kinetics datasets. Both the models achieve high classification accuracy.

we see an improved performance. Thus transfer learning improves not just the overall performance but also shows promise in improving the metrics of a class of interest.

It also seems rational to investigate how self-supervision is affecting the performance on the same domain. We refer to this analysis as Within-Domain (WD) performance. The pretext model is trained on a given dataset first. We use the weights obtained on convergence as initialization for training the main task on the same dataset. We finally compute the performance on the held-out validation set. Note here that there is no overlap between data used for training and data used for validation as required. Table 4.6 shows the advantages of using self-supervision for improving the performance of action detection on a given dataset. Not only there is an overall improvement in the mAP values for the two datasets but also an improved AP for majority of the classes.

Src	Tgt	Tag	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	mAP
KIN	AVA	BL	63.4	31.8	83.5	29.4	85.7	34.5	67.9	53.1	83.6	25.1	55.8
		TL	58.2	35.3	88.1	36.5	89.9	35.7	71.2	61.9	78.5	36.9	**59.2**
AVA	KIN	BL	49.8	20.4	80.14	37.8	83	28.9	51.5	37.1	58.6	32.5	48.1
		TL	49.1	22.1	82.5	41.1	85.4	30.1	56.1	39.9	62.4	32.7	**50.2**

Table 4.5: Cross Domain (CD) analysis of Transfer Learning (TL). Baseline (BL) results are taken from table 4.4. TL results are obtained using Source Supervision (SS). TL has improved the cross-domain performance for both the domain pairs.

Src	Tgt	Tag	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	mAP
AVA	AVA	BL	75.5	33.6	87.7	55.5	95.1	63.1	72.1	64.5	82.9	40.0	67.03
		TL	81.9	46.2	92.3	60.4	99.4	80.5	67.8	66.4	82.9	36.1	**71.4**
KIN	KIN	BL	69.8	30	90.5	46.2	90.3	36.7	66.7	43.3	73.8	40	58.76
		TL	68.7	27.1	91.2	50.4	91.9	42.7	66.9	43.9	73.3	48.3	**60.5**

Table 4.6: Within Domain (WD) analysis of Transfer Learning (TL). Baseline (BL) results are taken from table 4.4. TL has improved the within-domain performance for both the domain pairs.

4.4.3 Kinetics-only weights

As stated in the description of experimental setup, we have used the FB weights from a model of action detection trained on AVA dataset. The authors of SlowFast [14] have trained all of the action detection models by initializing with the weights learnt during training an action recognition model. The action recognition objective is trained only with Kinetics dataset as Kinetics provide the necessary ground truths. However, in this sequence of traininng and initialisations, one can argue that the weights learnt during the training of action detection have seen the samples as well as ground-truths from both Kinetics and AVA. In such a scenario, the results obtained for Transfer learning in the previous section will be questionable. Although we argue that within the scope of this work where the main focus is to address the hypothesis of efficacy of self-supervision in domain adaptation, the weights do not make a huge difference in the obtained results, it becomes imperative to analyse the scenario where the UDA setting is unquestionable.

We choose to evaluate the performance of Transfer Learning with action recognition weights obtained from Kinetics. The weights are taken from exactly the same configuration as chosen for action detection task i.e. under Model_Zoo.md section of PySlowFast repository, we choose the *Pretrain Model (Kinetics 600)* provided under the AVA dataset with Slowfast architecture, R101 depth, frame length x sample rate of 8×8, version 2.2 and an mAP of 29.1. These weights have only seen the Kinetics dataset where the ground-truth corresponds to frame level action label (note: this is different from the ground truth we use for action detection which is per person annotations). Thus keeping Kinetics as the source domain and AVA as target fits the conventional definition of UDA. Also, note here that the baselines need to be changed accordingly. The baselines for this particular experiment are obtained using Kinetics-only weights following the same setup as used in Table 4.4. Finally, note that the Kinetics-only weights and baselines are only used for the analysis in this section, the rest of the work continues to build upon the usual setup and baselines based on AVA action-detection weights.

As can be seen in Table 4.7, transfer learning has improved the performance over baseline in both CD and WD cases. This bolsters our claim that self-supervision-based transfer learning improves model performance. Fig 4.10 shows the convergence in training loss and updates in mAP over a validation set. All these plots are obtained for training an action detection objective using transfer learning approach. We notice that plots obtained in (c) using Kinetics-only weights are similar in behavior to those in (a) and (b) obtained using AVA action detection weights. Thus, we don't expect drastic changes in results using one weights instead of others and continue the rest of our analysis with the usual settings.

Src	Tgt	Tag	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	mAP
KIN	AVA	CD-BL	34.3	26.8	85.8	19.7	59.9	33.6	70.0	24.7	59.3	17.7	43.2
		CD-TL	40.5	36.8	84.8	15.6	75.5	28.4	64.4	27.3	55.7	19.9	**44.9**
KIN	KIN	WD-BL	59.3	27.1	89.8	43.2	96.3	30.0	59.5	39.1	67.6	38.9	55.1
		WD-TL	61.2	26.8	91.6	40.1	94.5	31.4	65.4	35.8	68.7	46.0	**56.1**

Table 4.7: Cross domain (CD) and Within Domain (WD) analysis of Transfer Learning (TL) using Kinetics-only action recognition weights. Baseline (BL) results are generated with the same weights. TL has improved the both CD and WD performance under UDA settings.

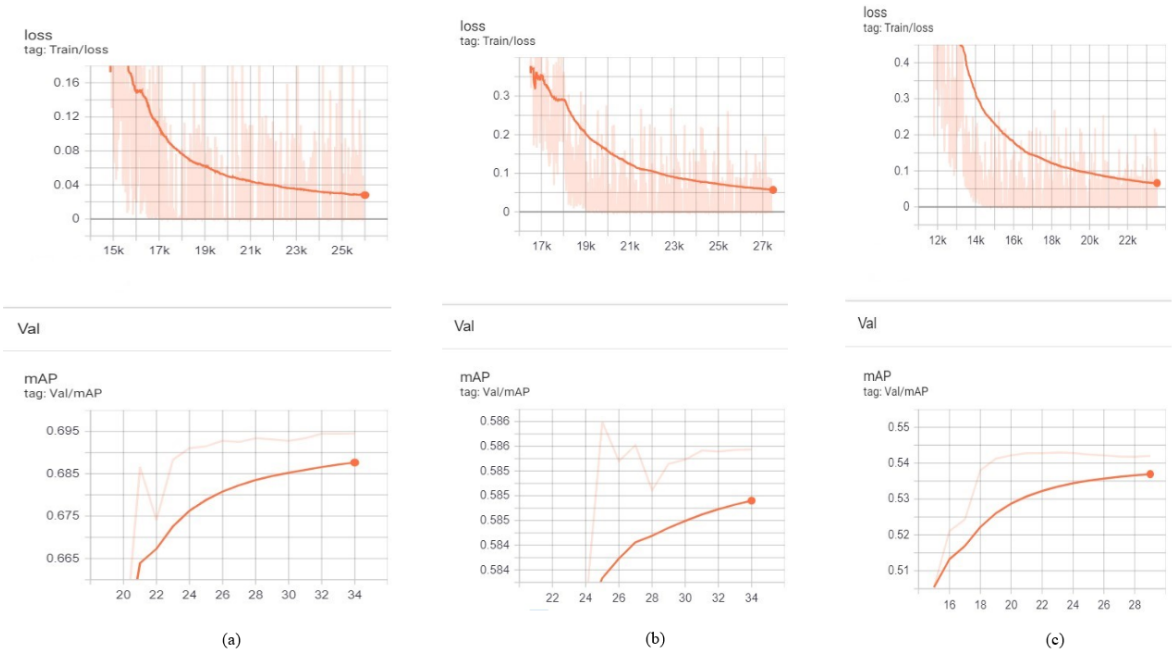


Figure 4.10: Convergence of training loss (above) and mAP achieved on validation set (below) for action detection models trained under transfer learning setup on (a) AVA and (b) Kinetics datasets with pretext task trained using AVA action detection weights, and (c) Kinetics dataset with pretext task trained using Kinetics action recognition weights. Note the similarity in plots especially between (b) and (c)

4.4.4 Different supervision forms

The results obtained until now come under the umbrella of Source supervision (SS) where both the pretext and main tasks are trained on one domain and tested on another. Thus, in this setting we do not assume to leverage any information about the Target domain i.e. the target domain is rather a black box for us. However, there have also been works in literature which consider accessing unlabelled samples from the target domain and using them to learn properties about the domain so that the classifier in question can generalize better. Such argument fits well when working with self-supervision as training the pretext task doesn't need labels to be provided from dataset. As discussed previously, there are also possibilities of inducing supervision from Target domain as it does not violate the principles of unsupervised domain adaptation (UDA). The pretext task generates labels on its own without seeing the ground-truth labels provided by the dataset. Thus unlabelled samples from the target domain can be used to train the pretext task and used to transfer the knowledge to the training process of the main task. We will refer to it as Target Supervision (TS). The third possibility is the case of Joint Supervision (JS) which leverages the unlabelled samples from both the source and target domains to train the pretext task.

We used both the methods to train our pretext task and see how well it transfers the knowledge to the main task of action detection. In case of Target supervision, we train our pretext task using unlabelled data from target domain, use the weights learnt to initialise training the main task over Source domain. During test time, we evaluate our main task on Target domain to obtain CD performance. For Joint supervision, we had two options- either train the pretext task on a huge dataset consisting of combined Source and Target samples, or use the weights learnt while training the pretext task for Source domain as initialisation for training the task on Target domain. We chose to work with the latter as it uses already trained weights and thus saved training a model from scratch on a larger dataset (thus saving time and resources). Once the pretext task is trained, we follow the usual process of training the main task on Source and testing it on Target domain. Table 4.8, we see cross domain performance of transfer learning using Source supervision (SS), Target supervision (TS) and Joint supervision (JS) as compared against baseline (BL). One immediate inference is that all three types of supervisions are better than the baseline, thus corroborating our claims on the efficacy of self-supervision be it in any form. Also, a closer look will reveal that the the per-class AP has improved too.

4.5 Multi-task Learning

As shown in Fig. 4.1, during multi-task learning we train both the main and pretext tasks together using separate fully connected layers aka heads. The encoder is still based on SlowFast module, the additional heads are added on top of it to learn different objectives simultaneously. We utilise the optimization procedure proposed in [64]. Broadly speaking, we use two separate data loaders and two separate optimizers, one for the main and the other for pretext task. We have two loss functions as well- a supervised loss for the main task and a self-supervised loss for the pretext task. During one epoch of training, we optimise action classification and then optimise the clip order prediction task. For more details, we refer the readers to the implementation. The model configuration followed is similar to that for Transfer Learning and baselines i.e. as mentioned in Fig.4.3 (b). Thus, Res-5 is fine-tuned and the action and the rotation head are trained from scratch. Start learning rates of 0.1 and 0.01 are used for the action classification and the clip order prediction respectively. Samples are fed to the pipeline in batches of size 4.

Table 4.9 shows the cross-domain performance of MTL. It is interesting to note here that MTL could not surpass beyond the baseline in one particular domain pair setting. The improvements for the other domain pair is also not a huge increase. Similarly, the MTL has shown improvements over baseline in within-domain

Src	Tgt	Tag	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	mAP
KIN	AVA	BL	63.4	31.8	83.5	29.4	85.7	34.5	67.9	53.1	83.6	25.1	55.8
		SS	58.2	35.3	88.1	36.5	89.9	35.7	71.2	61.9	78.5	36.9	**59.2**
		TS	56.1	46.6	81.2	40.2	85.7	33.8	66.2	55.1	84.2	33.9	58.3
		JS	56.9	44.4	83.9	41.6	87.9	29.0	55.1	54.2	86.8	35.8	57.6
AVA	KIN	BL	49.8	20.4	80.14	37.8	83	28.9	51.5	37.1	58.6	32.5	48.1
		SS	49.1	22.1	82.5	41.1	85.4	30.1	56.1	39.9	62.4	32.7	50.2
		TS	57.2	20.8	81.6	42.3	81.5	32.6	55.5	39.4	64.6	36.4	51.2
		JS	55.7	20.3	81.2	37.2	84.4	32.4	57.1	42.3	61.3	39.3	**51.6**

Table 4.8: Cross Domain analysis of Transfer Learning under different supervision. Baseline (BL) results are taken from table 4.4. Irrespective of the supervision type, we see an improvement over baseline.

scenarios as depicted in Table 4.10. These results show that MTL is definitely a useful technique when dealing with domain adaptation using self-supervision. However, the achieved improvements are not as large as TL. For further improvements, the initialisation method needs to be taken into consideration. The weights used to initialise the feature extractor 4.3(b) already experienced substantial AVA and Kinetics training. The weights might be highly biased due to the sequence of training used by SlowFast that the important information needed to be learnt using self-supervision is not being able to be incorporated. Therefore, the future experiments in this direction should aim at understanding the learning of the model from a more basic setting, i.e. using more randomly initialised layers. The more weights are learnt from scratch in a joint training setting, the improved mutual information will be between the main and pretext tasks' weights. Note that if more layers are unfrozen and randomly initialized, we would need to expand our training set from the present 5k training samples. This scaling up of model as well as the dataset will come at the cost of increased training time and require more computational resources, precisely why this work does not delve further into this direction.

One small scale experiment which showed promise was the effect of ratio of learning rates between the main and the pretext task. We noticed that by increasing the factor by which learning rates differ for the main and the pretext tasks, the performance of MTL on a CD scenario improved actually (see Table 4.11). This motivated us to prefer a large ratio between learning rates of main and pretext tasks (typically 100) in order to improve the training of the Multi-task setting. A closer look at the training curves in Fig. 4.11 reveal that LRF affects the training of the pretext task in particular. The main task is kept at a constant learning rate of 0.1 as stated by the SlowFast default parameters. Indeed, the training loss converges nicely for the main task in both scenarios when LRF is (a) 1 and (b) 100. The learning rates of pretext task is adjusted based on the given learning rate of the main task and chosen LRF. This is done because the major focus is not on training the pretext task optimally but training in a way that learns information relevant for the main task. This is evident in the figure as well. In case (a) LRF of 1 decreases the loss of pretext task quickly within the given epochs as compared to (b) where the decline in loss is relatively slower. However, as witnessed in Table 4.11, the CD performance is better for model with LRF 100 rather than 1. Therefore, it becomes crucial in MTL to analyse the convergence of training losses of both tasks while improving mAP on the validation set.

Src	Tgt	Tag	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	mAP
KIN	AVA	BL	63.4	31.8	83.5	29.4	85.7	34.5	67.9	53.1	83.6	25.1	55.8
		MTL	59.2	36.8	89.7	28.9	86.4	37.6	71.6	39.3	83.1	28.3	**56.1**
AVA	KIN	BL	49.8	20.4	80.14	37.8	83	28.9	51.5	37.1	58.6	32.5	**48.1**
		MTL	40.4	20.2	79.2	37.2	82.0	29.8	47.6	39.6	56.2	37.9	47.03

Table 4.9: Cross Domain (CD) analysis of Multi-task Learning (MTL). Baseline (BL) results are taken from table 4.4. TL results for MTL are obtained using Target Supervision (TS). MTL has improved the cross-domain performance for one domain pair, and a attained competitive performance for the other.

Src	Tgt	Tag	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	mAP
AVA	AVA	BL	75.5	33.6	87.7	55.5	95.1	63.1	72.1	64.5	82.9	40.0	67.03
		MTL	74.4	45.2	89.8	54.3	94.7	63.6	76.0	63.7	82.6	39.4	**68.4**
KIN	KIN	BL	69.8	30	90.5	46.2	90.3	36.7	66.7	43.3	73.8	40	58.76
		MTL	67.8	29.9	89.2	48.8	92.3	38.7	67.3	42.5	72.5	47.4	**59.7**

Table 4.10: Within Domain (WD) analysis of Multi-task Learning (MTL). Baseline (BL) results are taken from table 4.4. MTL has improved the within-domain performance for both the domain pairs.

A scenario where both the main and pretext tasks are training nicely might not be sufficient to transfer the knowledge from one task to another. This is clearly different from the case of TL where the major focus was to first optimize the training of pretext task and then optimize separately for the main task. This also hints at a possible improvement using regularization based on weights learnt during the pretext optimization so that knowledge transfer is improved. To conclude, unlike TL, MTL will need additional optimization process to improve performances even when both the tasks train optimally in their respective turns.

Finally, as seen in the case of Transfer Learning, we can induce the supervision in a variety of ways. The results presented till now in this section are based on Target Supervision (TS) as it generated the best results for MTL. However, for the completeness of the analysis, we present the CD performance of MTL using different forms of supervision in Table 4.12. Unlike TL, the improvement over baseline is not unanimously better among all supervision types. We witness some improvements in the case of transferring knowledge from Kinetics to AVA. However, the reverse scenario fails to see any such improvement. One possible explanation is that different types of supervisions need extensive tuning of hyper-parameters. We have optimised TS by fine-tuning all the relevant parameters. Such exhaustive fine-tuning of hyper-parameters has not been done for SS or JS. Also as argued above, increasing the size of dataset, number of trainable layers and randomization of weights should improve these results and we keep them as a part of future exploration.

Src	Tgt	LRF	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	mAP
KIN	AVA	1	53.2	26.8	87.1	27.2	54.2	20.1	66.8	30.5	76.5	26.1	46.8
		10	61.8	36.2	91.4	30.2	75.0	33.2	54.2	40.7	80.6	28.6	53.2
		100	59.2	36.8	89.7	28.9	86.4	37.6	71.6	39.3	83.1	28.3	**56.1**

Table 4.11: The effect of Learning rate factor (LRF) on the Cross Domain (CD) performance of MTL. LRF is defined as the ratio between learning rates of main and pretext tasks. Note how a higher ratio results in improved performance.

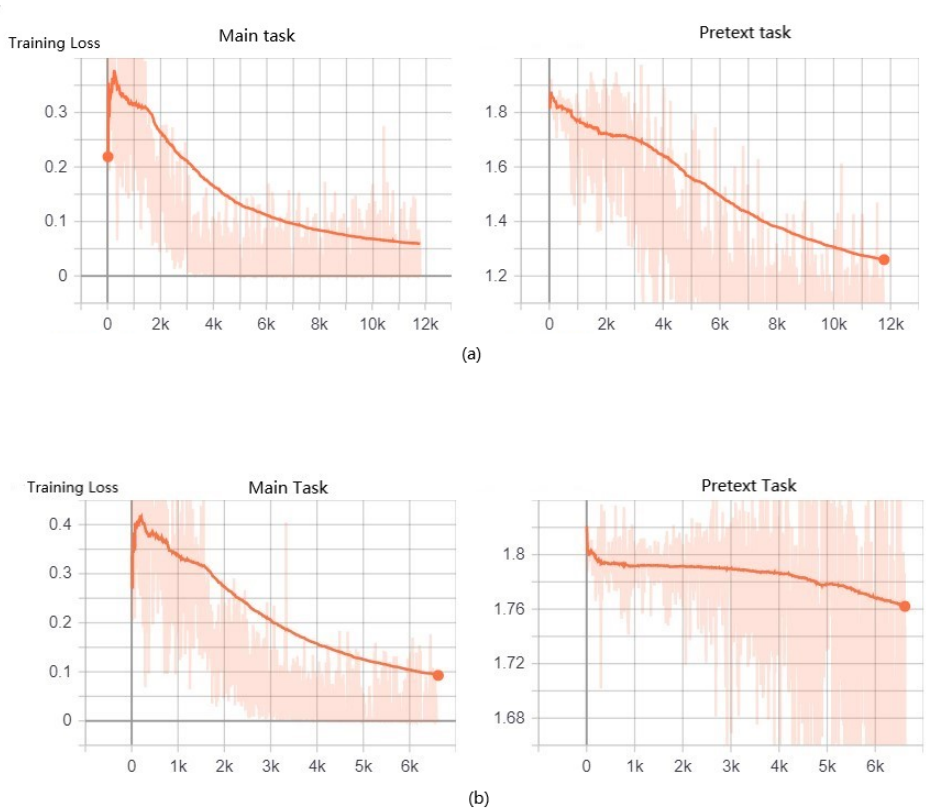


Figure 4.11: Convergence of training loss for the main task (left) and the pretext task (right) in a multi-task setting using a learning rate factor (LRF) (a) 1 and (b) 100. The loss for pretext task in (a) seem to be converging better, however test time performance of (b) is better which is of more interest. The convergence in loss for main task is almost similar in both settings

Src	Tgt	Tag	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	mAP
KIN	AVA	BL	63.4	31.8	83.5	29.4	85.7	34.5	67.9	53.1	83.6	25.1	55.8
		SS	57.2	25.9	84.9	28.1	84.5	28.7	69.3	54.5	78.8	19.7	53.2
		TS	59.2	36.8	89.7	28.9	86.4	37.6	71.6	39.3	83.1	28.3	56.1
		JS	67.4	31.4	84.6	29.5	84.8	37.9	65.9	49.6	89.7	21.4	**56.2**
AVA	KIN	BL	49.8	20.4	80.14	37.8	83	28.9	51.5	37.1	58.6	32.5	**48.1**
		SS	36.8	18.6	76.1	34.7	72.3	31.3	34.9	37.6	56.5	37.9	43.7
		TS	40.4	20.2	79.2	37.2	82.0	29.8	47.6	39.6	56.2	37.9	47.03
		JS	36.1	21.9	78.3	37.7	76.2	25.4	36.8	38.3	53.9	36.1	44.1

Table 4.12: Cross Domain analysis of Multi-task Learning under different supervision. Baseline (BL) results are taken from table 4.4. Unlike TL, the improvements over baseline is limited to few instances only.

4.6 Computation Analysis

All the experiments mentioned above are carried out on a single GeForce GTX TITAN X GPU with memory of 50GB available and a 48h time limit for a submitted job on SLURM. In order to train models needing more training time such as multi-task learning, we saved checkpoints at the time limit and resumed training till the required number of epochs are finished. This project is granted a quota of 3 GPUs which can be used simultaneously. In majority of the cases, one GPU is used to train one model so that three different models can be trained in parallel. The batch size used in the experiments majorly depended upon the GPU memory limit. A typical action detection baseline method trained with a training configuration mentioned in Fig. 4.3 (b) took 8-11 hours for 15 epochs with the reduced datasets. Similarly, a clip order predcition task following the same training configuration took 10-12 hours. Transfer Learning is an offline method, so the total time to generate results depends on the addition of time taken by the main and pretext tasks. However, once a pretext task was trained, it was used for many different experiments with the main task. Finally, a multi-task training with the same training configuration as above took 20-24 hours to train. Note here that since this is a joint training, different experiments could not be parallelised between the main and pretext tasks. The decision to reduce the dataset was primarily driven by the limitations of the GPUs. The baselines were also generated for the reduced dataset and not directly taken from the SlowFast paper [14]. This work mentions training for 68 epochs using 211k samples from AVA dataset for an objective of action detection. This scenario was not feasible to be replicated as they used synchronized SGD training on 128 GPUs in few experiments while using 8 GPUs for most of the other experiments, whereas we were limited to 3 GPUs only. The scale of our experiments is significantly smaller and future works can focus on a multi-GPU training that can enable dataset expansion as well as better training for the multi-task setting.

Chapter 5

Discussion

Given that the results can be overwhelming in terms of numbers, we draw some key insights from the previous section and present both quantitative and qualitative impact of our work in this chapter.

5.1 Cross-Domain Analysis

Fig. 5.1 shows the Cross Domain (CD) performance of methods proposed in this work against the baseline. Note here that the baseline doesn't use any additional knowledge about the target domain whereas both TL and MTL use self-supervision. We notice that Transfer learning shows significant improvement in domain adaptation from AVA to Kinetics and vice-versa. Multi-task Learning also improves over the baseline for one scenario of adapting from Kinetics to AVA. It doesn't improve beyond baseline in the case of AVA to Kinetics. Moreover, transfer learning has shown superior performance than MTL. Given the constrained experimental setup we used in this work, it will be rudimentary to conclude that TL is better than MTL. We will need to scale up the experiments in terms of more data and more trainable parameters to optimize MTL better. However the main objective of this work, to show the efficacy of self-supervision for domain adaptation has been clearly established here. The best performing model has improved the performance by 7.3% for adapting from AVA to Kinetics and by 6.1% for Kinetics to AVA over the baseline. Thus, self-supervision is useful for domain adaptation.

5.2 Within-Domain Analysis

It also becomes interesting to observe how self-supervision can help in boosting performance on the same dataset once we have seen that it shows promising results on a different domain. Fig. 5.2 shows the within domain performance of baseline compared against the two self-supervised methods presented in this work. It is interesting to observe that both the methods, MTL and TL improve over the baseline for both the datasets. Self-supervision improves the performance by 6.5% on AVA dataset and 2.9% on Kinetics dataset over the said baselines. This suggests weights learnt by a self-supervised task can help gain better performance for the downstream main task as compared to generically trained weights on popular datasets like ImageNet [10].

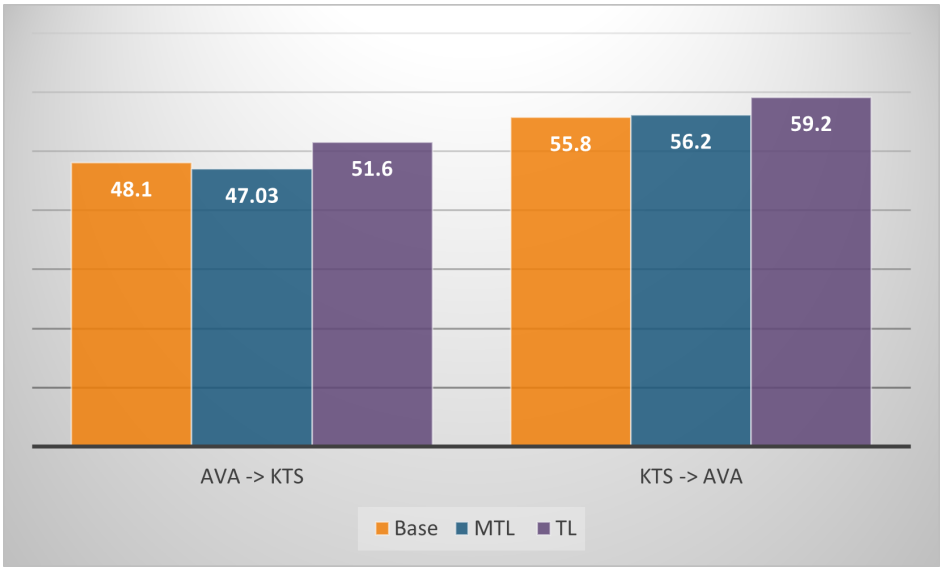


Figure 5.1: Self-supervision based on clip order prediction improving cross-domain performance by 7.3% (left) and 6.1% (right) over the baseline in the two scenarios.

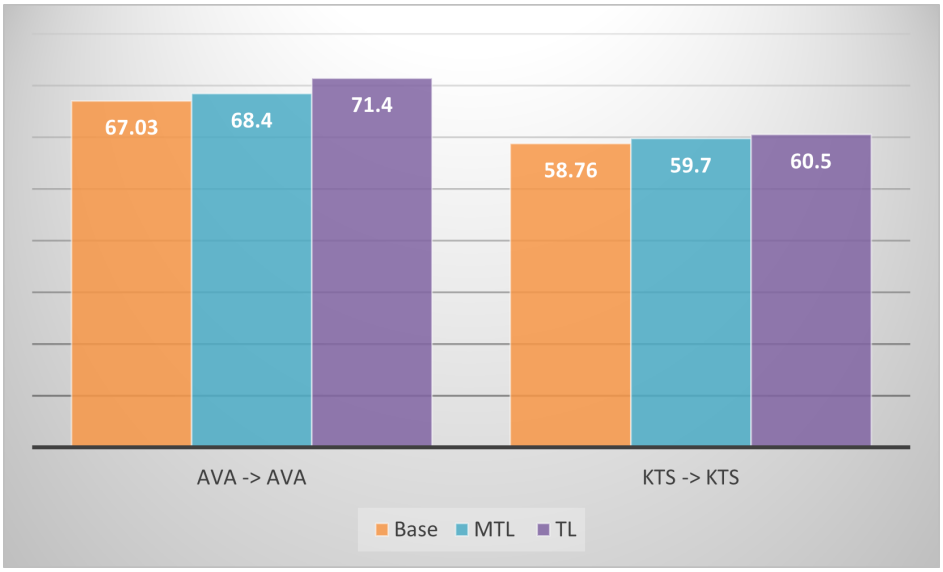


Figure 5.2: Self-supervision based on clip order prediction improving within-domain performance by 6.5% (AVA) and 2.9% (Kinetics) over the baseline in the two scenarios.

5.3 T-SNE analysis

We analyse one scenario each from the Cross-domain and Within-domain experiments and visualise the t-SNE plot of features obtained after Res 5 layer and before the action head , see Fig. 4.1. To analyse how features of different classes are affected by self-supervision, we take the case of cross domain analysis where the source is AVA and Target is Kinetics dataset in Fig.5.3. Baseline corresponds to a model trained without any self-supervision on AVA and tested on Kinetics. We can see that the features learnt for the 10 classes are quite overlapping among each other. On the other hand, proposed method corresponds to a Transfer learning based method that uses source supervision as seen in Table 4.5. The features learnt for different classes are more separated from each other where small clusters of a particular color are visible for some classes such as *ride*, *lie* and *talk*. Thus, self-supervsion improves the inter-class separability among the classes.

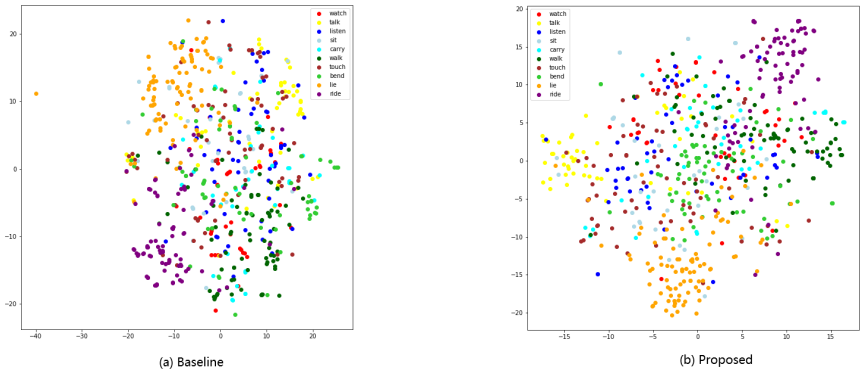


Figure 5.3: Comparison of features after Res5 layer for the baseline and proposed methods in a CD scenario of AVA to Kinetics. We see that self-supervision based on clip order prediction improves the inter-class separability.

Next, we analyse how features behave when a model is tested on the same domain. For the within-domain (WD) analysis, we consider the case of training and evaluating on AVA dataset as seen in Table 4.6. Here, we see in Fig. 5.4 that the baseline model as already some inter-class separation. The features belonging to a particular class, shown in the same color, are not overlapping as much as they were in the CD scenario of testing the same model on Kinetics dataset seen above. However, the features of a given class are also quite scattered. We ideally would want the features of a given class to be as close to each other as possible in a classification task. The features learnt by the proposed TL method are closer to our expectations. Here, the features of different classes are not only separated from one another, but also quite compact among the same class members. Especially for classes such as *ride*, *lie*, *talk*, *sit* etc, the improvements are huge. Thus, self-supervision has improved the intra-class variation.

5.4 Class-wise performance

Motivated by the qualitative analysis of features obtained by using self-supervision in the training pipeline, we now compare how significant the performance improvement is for each individual class. Since we have already analysed the case of CD experiment from AVA to Kinetics, we will consider the reverse scenario here in which the source is Kinetics and the target is AVA. In Fig. 5.5, we plot the AP values obtained for

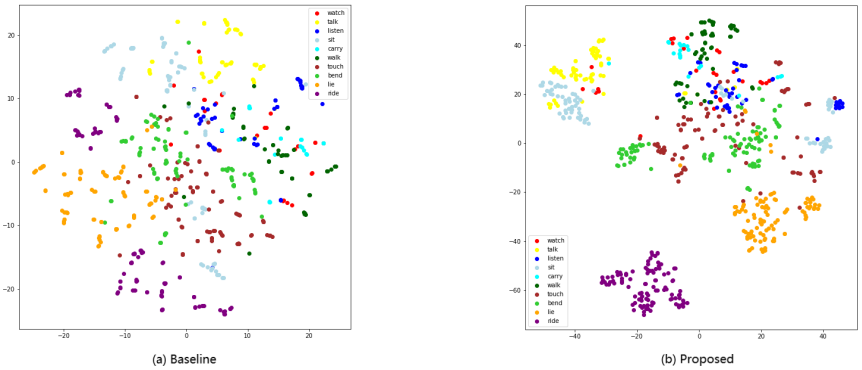


Figure 5.4: Comparison of features after Res5 layer for the baseline and proposed methods in a WD scenario of AVA. We see that self-supervision based on clip order prediction improves the intra-class variation.

each class in a CD scenario of Kinetics to AVA. Base corresponds to baseline which doesn't use any self-supervision, whereas SSL corresponds to transfer learning method using Self-Supervision Learning. We see that SSL unanimously improves the AP value above baseline for all the classes. This is also promising as it shows that if the number of classes is scaled from 10 to 80 as in AVA-Kinetics dataset, self-supervision can be highly beneficial over a large majority of classes.

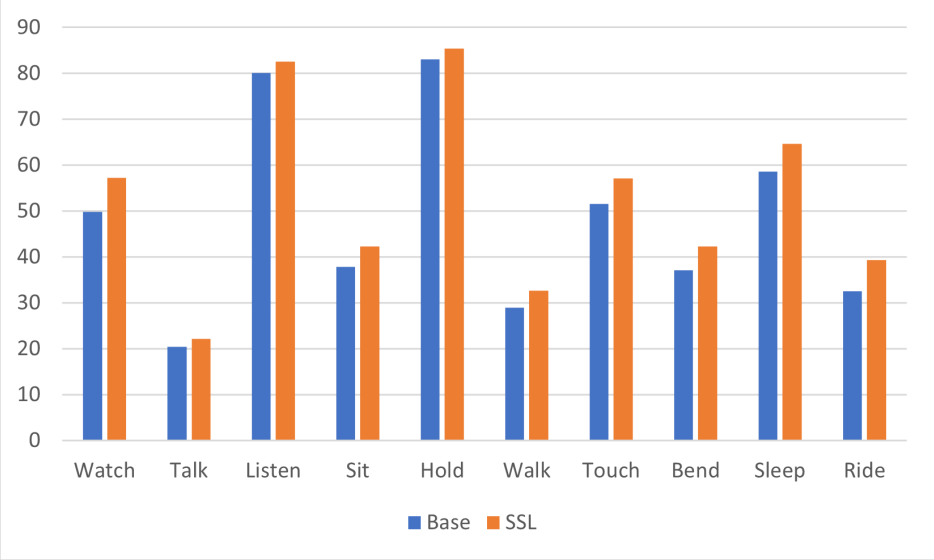


Figure 5.5: Comparison of Average Precision (AP) in percentage values obtained per class in a CD scenario of Kinetics to AVA between baseline (Base) and Self-supervision based TL (SSL). Performance is improved over baseline for all the classes, max improvement by 20.9% and minimum by 2.9% .

5.5 Comparison of supervision types

Throughout the result section we have seen the usage of different forms for supervision for training TL and MTL methods. We will have a closer look at the performance (% mAP) of TL using different supervision types in both the CD configurations. As seen in Fig. 5.6, the performance is improved over baseline for both the configuration using either of the supervision types. In the case of AVA to Kinetics, we see an increasing trend, where the target supervision improves the performance over source supervision. This is expected too, as getting some detail about the target domain should ideally be better than the case when no information about the target domain is used at all. Thus, TS performing better than SS follows the conventional understanding of UDA. Moreover, JS which combines the knowledge between both source and target domains has the highest performance among all. This observation again follows the reasoning that there might be some level of domain alignment between AVA and Kinetics that using supervision from AVA adds to the existing knowledge about Kinetics as seen in case of TL.

However, in the reverse scenario of Kinetics to AVA, we unfortunately do not see such clear logical pattern. We witness that SS performs the best while JS performs the least, although all three supervision types are better than the baseline. One plausible explanation of this behavior might be due to the sequential pre-trained weights used by AVA, where first an action recognition task is trained on Kinetics, then an action detection task is built on top of the learnt weights using AVA, then the clip order prediction task is trained on top of these weights using Kinetics or AVA or both, and finally an action detection task is trained with these weights on Kinetics before evaluating on AVA. As can be realised from the highly complex chain of initialisation and training, leakage of supervision can happen and we might not see the results as expected. In order to get more explainable results for this setting, future works can build upon training models from scratch and evaluating different supervision types. However, the experiment does conclude that self-supervision improves upon the baseline irrespective of the type of supervision used which is the major point we wish to establish here in this work.

5.6 Qualitative analysis

So far we have seen how self-supervision has improved the cross domain performance in a UDA setting. We have also seen how within-domain and class-wise performance can benefit from self-supervision. However, the analysis has been limited to numbers only. In this section we shall have a closer look at the qualitative advantages that self-supervision provides in domain adaptation. Fig.5.7 shows a few success (top) and failure (bottom) cases. Green bounding box correspond to prediction of our proposed model which uses self-supervision. Red bounding box refers to the prediction of baseline model which does not use any self-supervision. Green labels are predictions by the proposed method, red labels are predictions made by the baseline and the white label refers to the ground truth. Note that in AVA-Kinetics dataset, more than one ground truth labels are possible for a given bounding box as it is assumed that a person can be performing multiple actions simultaneously. We consider the UDA setting of Kinetics to AVA and the proposed model is based on Transfer Learning since it gave the best mAP among all the models discussed.

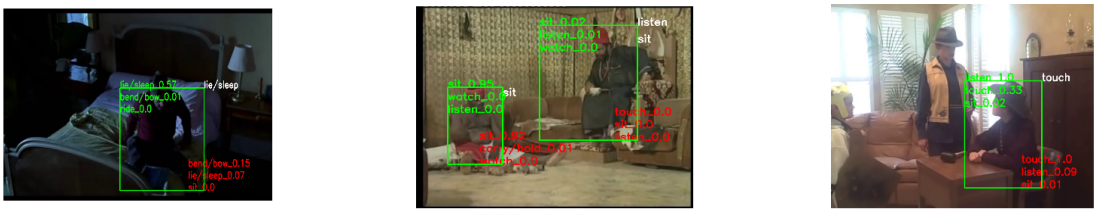
Among the results shown, we discuss in detail three specific cases in Fig. 5.8. Fig. (a) shows a case where our model generates correct label while the baseline fails to do so. For the frame shown, the ground truth is a single class label of *lie/sleep*. The baseline predicts the correct label as its second best guess however the confidence value is quite low at a mere 7%. On the contrary, our proposed model is correct in its top-1 prediction with a much higher confidence level of 57%. This serves as an example of our claim that the self-supervision based model proposed in this work adapts better to the target domain.

Figure 5.6: Percentage mAP values compared among different supervision types and with the baseline for the two CD configurations. The results are obtained using a TL-based method. The primary conclusion is that self-supervision based on clip order prediction improves the performance in domain adaption irrespective of the type of supervision used.

Fig. (b) shows an example where there are two persons of interest in the frame. For the person on left sitting on the ground, the ground truth label is *sit*. Both the baseline and the proposed methods predict it as their most confident prediction, however our model is more confident in the prediction (95%) compared to baseline (92%). For the person on the right sitting on the throne, there are multiple annotations available. The ground truth consists of two classes *sit* and *listen*. We see that the baseline is able to predict these as the second and third guess, however the top-1 prediction is incorrect. To the contrary, our method predicts both the classes as the first and second most probable prediction and with a higher confidence level. This example illustrates that self-supervision based methods are rather more confident in their correct predictions and handle the case of multiple annotations efficiently.

Finally, in Fig. (c) we see a case where the proposed model fails to predict the correct ground truth label of *touch* whereas the baseline does it. This is an example where the baseline works better than our model. However, if we closely look at the predictions by our method, the top-3 predicted labels are *listen*, *touch* and *sit*. These are reasonable predictions given that the person of interest in the sample frame is actually sitting, touching the table and listening to the person standing next to him. Thus, even when our method fails to match the ground-truth label, the predictions are highly rational. Thus, self-supervision incorporates a better understanding of the spatial and temporal surroundings of the person of interest as well.

Figure 5.7: Qualitative results showing success (top) and failure (bottom) cases of a TL model in a CD scenario of Kinetics to AVA.



(a) (b) (c)

Figure 5.8: A few specific examples where (a) Our method adapts better to the target domain (b) Our method is more confident when correct and handles multiple annotations better, and (c) Our method gives reasonable predictions when incorrect.

Chapter 6

Conclusion

In this work, we proposed and analysed a video clip order prediction based self-supervision as a method to improve domain adaptation for the objective of action detection. We proposed two methods to incorporate the supervision learnt by the clip order prediction model- first a transfer learning model based on sequential transfer of knowledge between the self-supervised and supervised objectives, and second a multi-task learning model based on simultaneous training and knowledge transfer. We observed that the best of these models improved the cross domain performance from AVA to Kinetics by 7.3% and from Kinetics to AVA by 6.1% over the baseline. Moreover, we observed that self-supervision improves the within-domain performance where the pretext task helps in an improved understanding of the domain for the downstream main task of action detection. An improvement of 6.5% on AVA and 2.9% on Kinetics over baseline model echo the efficacy of our proposal. This reinstates the claim that the weights learnt by using self-supervision capture better information about the dataset than the popular pre-trained weights used in the literature for downstream tasks such as action detection. We also established that the performance gain in domain adaptation is not just at the global level, but also the improvement in the Average Precision for individual classes is promising, where we atleast improved by 2.9% for one class while going as much as 20.9% for another class. This work has also addressed the usage of various forms of self-supervision whether it is from source, target or both, and we have observed that any form of supervision improves the cross-domain performance over the baseline method. The usefulness of our work is further illustrated through visualising the features learnt for the action detection task in a UDA setting where we observed that self-supervision improved the inter-class separability and intra-class variation of the features learnt. Finally the qualitative analysis bolstered our claims made through quantitative figures by establishing that our method is more accurate, confident, and rational in prediction on the target domain.

The future works include expansion of the proof of concept to training all SlowFast network layers using entire classes and samples from the AVA and Kinetics700 domains in order to draw more concrete conclusions. This requires reducing training time and compute required, which will be possible with exploring different encoder architectures and self-supervised tasks. The SlowFast model by FB [14] has been trained with parallel processing over 8 GPUs and future work will require us to incorporate it in the current proposed method. This will further help in learning weights from scratch and assessing their benefits. This will also help in improving the performance of Multi-task learning as it is more computationally demanding than transfer learning. One major constraint in this work was the weight initialisation based on FB pre-trained models, given how they have been used in the SlowFast training for action recognition and detection tasks. Future work will also include analysing the UDA performance in a setting where the weights are unambiguously based on the domain of interest. The current work uses clip order prediction as self-supervision that is

more focused on the temporal aspects of the input. We can define a self-supervision task where both the spatial and temporal characteristics of the video inputs are exploited. This can be either in the form of designing a single pretext task with more number of classes that classifies inputs based on spatio-temporal properties or adding multiple pretext tasks where each focus on one aspect as proposed by [64]. An equally popular direction of research in domain adaptation for videos is adversarial learning, and future work can explore the ideas presented in [36] that combines both adversarial learning and self-supervision via a contrastive loss formulation. Finally, the performance of Multi-task learning can further be improved by investigating ways of enforcing additional constraints on the features learnt during joint optimization. One possible way of doing this can be via regularization imposed on the weights learnt by the pretext task as discussed in [30].

Appendix A

Abbreviations

- SSL: Self-Supervised Learning
- AP: Average Precision
- mAP: Mean Absolute Precision
- CNN: Convolution Neural Network
- CE: Cross Entropy
- BCE: Binary Cross Entropy
- FB: Facebook
- KIN: Kinetics
- WD: Within Domain
- CD: Cross Domain
- Src: Source
- Tgt: Target
- TL: Transfr Learning
- SS: Source Supervision
- TS: Target Supervision
- JS: Joint Supervision
- UDA: Unsupervised Domain Adaptation
- BL: Baseline
- LRF: Learning rate factor
- IoU: Intersection over Union

Bibliography

[1] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *arXiv preprint arXiv:1906.00910*, 2019.

[2] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. *arXiv preprint arXiv:1608.06019*, 2016.

[3] Uta Buchler, Biagio Brattoli, and Bjorn Ommer. Improving spatiotemporal self-supervision by deep reinforcement learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 770–786, 2018.

[4] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 132–149, 2018.

[5] 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.

[6] 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.

[7] Min-Hung Chen, Zsolt Kira, and Ghassan AlRegib. Temporal attentive alignment for video domain adaptation. *arXiv preprint arXiv:1905.10861*, 2019.

[8] Jinwoo Choi, Gaurav Sharma, Manmohan Chandraker, and Jia-Bin Huang. Unsupervised and semi-supervised domain adaptation for action recognition from drones. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1717–1726, 2020.

[9] Gabriela Csurka. A comprehensive survey on domain adaptation for visual applications. *Domain adaptation in computer vision applications*, pages 1–35, 2017.

[10] 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.

[11] AM Derrington and P Lennie. Spatial and temporal contrast sensitivities of neurones in lateral geniculate nucleus of macaque. *The Journal of physiology*, 357(1):219–240, 1984.

[12] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.

[13] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. Citeseer, 2014.

[14] 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.

[15] Basura Fernando, Hakan Bilen, Efstratios Gavves, and Stephen Gould. Self-supervised video representation learning with odd-one-out networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3636–3645, 2017.

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

[17] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[18] Rohit Girdhar, Joao Carreira, Carl Doersch, and Andrew Zisserman. Video action transformer network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2019.

[19] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[20] GitHub. chi0tzp/kineticx-downloader. https://github.com/chi0tzp/KineticX-Downloader, 2020. Accessed: April 01, 2021.

[21] GitHub. facebookresearch/slowfast. https://github.com/facebookresearch/SlowFast/blob/master/MODEL_ZOO.md, 2020. Accessed: April 02, 2021.

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

[23] 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.

[24] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6546–6555, 2018.

[25] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[27] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

[28] 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.

[29] Rui Hou, Chen Chen, and Mubarak Shah. Tube convolutional neural network (t-cnn) for action detection in videos. In *Proceedings of the IEEE international conference on computer vision*, pages 5822–5831, 2017.

[30] Lukas Hoyer, Dengxin Dai, Yuhua Chen, Adrian Köring, Suman Saha, and Luc Van Gool. Three ways to improve semantic segmentation with self-supervised depth estimation. *arXiv preprint arXiv:2012.10782*, 2020.

[31] Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H Adelson. Learning visual groups from co-occurrences in space and time. *arXiv preprint arXiv:1511.06811*, 2015.

[32] Arshad Jamal, Vinay P Namboodiri, Dipti Deodhare, and KS Venkatesh. Deep domain adaptation in action space. In *BMVC*, volume 2, page 4, 2018.

[33] Dinesh Jayaraman, Frederik Ebert, Alexei A Efros, and Sergey Levine. Time-agnostic prediction: Predicting predictable video frames. *arXiv preprint arXiv:1808.07784*, 2018.

[34] Dinesh Jayaraman and Kristen Grauman. Slow and steady feature analysis: higher order temporal coherence in video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3852–3861, 2016.

[35] Longlong Jing, Xiaodong Yang, Jingen Liu, and Yingli Tian. Self-supervised spatiotemporal feature learning via video rotation prediction. *arXiv preprint arXiv:1811.11387*, 2018.

[36] Minseon Kim, Jihoon Tack, and Sung Ju Hwang. Adversarial self-supervised contrastive learning. *arXiv preprint arXiv:2006.07589*, 2020.

[37] Diederik P Kingma, Danilo J Rezende, Shakir Mohamed, and Max Welling. Semi-supervised learning with deep generative models. *arXiv preprint arXiv:1406.5298*, 2014.

[38] Avisek Lahiri, Sri Charan Ragireddy, Prabir Biswas, and Pabitra Mitra. Unsupervised adversarial visual level domain adaptation for learning video object detectors from images. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1807–1815. IEEE, 2019.

[39] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Colorization as a proxy task for visual understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6874–6883, 2017.

[40] Chen-Yu Lee, Tanmay Batra, Mohammad Haris Baig, and Daniel Ulbricht. Sliced wasserstein discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10285–10295, 2019.

[41] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 667–676, 2017.

[42] Ang Li, Meghana Thotakuri, David A. Ross, João Carreira, Alexander Vostrikov, and Andrew Zisserman. The ava-kinetics localized human actions video dataset, 2020.

[43] Jerry Li, Aleksander Madry, John Peebles, and Ludwig Schmidt. On the limitations of first-order approximation in gan dynamics. In *International Conference on Machine Learning*, pages 3005–3013. PMLR, 2018.

[44] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[45] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *International conference on machine learning*, pages 2208–2217. PMLR, 2017.

[46] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 181–196, 2018.

[47] Aravindh Mahendran, James Thewlis, and Andrea Vedaldi. Cross pixel optical-flow similarity for self-supervised learning. In *Asian Conference on Computer Vision*, pages 99–116. Springer, 2018.

[48] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, pages 527–544. Springer, 2016.

[49] Behnam Neyshabur, Srinadh Bhojanapalli, and Ayan Chakrabarti. Stabilizing gan training with multiple random projections. *arXiv preprint arXiv:1705.07831*, 2017.

[50] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer, 2016.

[51] Mehdi Noroozi, Hamed Pirsiavash, and Paolo Favaro. Representation learning by learning to count. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5898–5906, 2017.

[52] Andrew Owens, Jiajun Wu, Josh H McDermott, William T Freeman, and Antonio Torralba. Ambient sound provides supervision for visual learning. In *European conference on computer vision*, pages 801–816. Springer, 2016.

[53] Deepak Pathak, Ross Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2701–2710, 2017.

[54] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[55] Xiaojiang Peng and Cordelia Schmid. Multi-region two-stream r-cnn for action detection. In *European conference on computer vision*, pages 744–759. Springer, 2016.

[56] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.

[57] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[58] Zheng Shou, Xudong Lin, Yannis Kalantidis, Laura Sevilla-Lara, Marcus Rohrbach, Shih-Fu Chang, and Zhicheng Yan. Dmc-net: Generating discriminative motion cues for fast compressed video action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1268–1277, 2019.

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

[60] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *arXiv preprint arXiv:1406.2199*, 2014.

[61] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852. PMLR, 2015.

[62] Swathikiran Sudhakaran, Sergio Escalera, and Oswald Lanz. Lsta: Long short-term attention for egocentric action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9954–9963, 2019.

[63] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7464–7473, 2019.

[64] Yu Sun, Eric Tzeng, Trevor Darrell, and Alexei A Efros. Unsupervised domain adaptation through self-supervision. *arXiv preprint arXiv:1909.11825*, 2019.

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

[66] Yicong Tian, Rahul Sukthankar, and Mubarak Shah. Spatiotemporal deformable part models for action detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2642–2649, 2013.

[67] 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.

[68] 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.

[69] Michael Tschannen, Josip Djolonga, Marvin Ritter, Aravindh Mahendran, Neil Houlsby, Sylvain Gelly, and Mario Lucic. Self-supervised learning of video-induced visual invariances. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13806–13815, 2020.

[70] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. *arXiv preprint arXiv:1609.02612*, 2016.

[71] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2015.

[72] Xiaolong Wang, Kaiming He, and Abhinav Gupta. Transitive invariance for self-supervised visual representation learning. In *Proceedings of the IEEE international conference on computer vision*, pages 1329–1338, 2017.

[73] Ximei Wang, Liang Li, Weirui Ye, Mingsheng Long, and Jianmin Wang. Transferable attention for domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5345–5352, 2019.

[74] 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.

[75] Laurenz Wiskott and Terrence J Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural computation*, 14(4):715–770, 2002.

[76] Shaoan Xie, Zibin Zheng, Liang Chen, and Chuan Chen. Learning semantic representations for unsupervised domain adaptation. In *International conference on machine learning*, pages 5423–5432. PMLR, 2018.

[77] Dejing Xu, Jun Xiao, Zhou Zhao, Jian Shao, Di Xie, and Yueting Zhuang. Self-supervised spatiotemporal learning via video clip order prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10334–10343, 2019.

[78] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.

[79] Xiao-Yu Zhang, Haichao Shi, Changsheng Li, Kai Zheng, Xiaobin Zhu, and Lixin Duan. Learning transferable self-attentive representations for action recognition in untrimmed videos with weak supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9227–9234, 2019.

[80] Chengxu Zhuang, Alex Lin Zhai, and Daniel Yamins. Local aggregation for unsupervised learning of visual embeddings. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6002–6012, 2019.