Towards Efficient Video Understanding and Generation: Free Training Signals to Faster Inference

A thesis proposal presented

by

Kumara Kahatapitiya

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

Stony Brook University

April 2024

Abstract

Video data is critical for learning systems that can perceive, interact with and reason about the world. Going beyond images, videos can provide useful cues about motion, causality, and even geometry, while often consisting of multiple modes of data. Yet, they also heighten the challenges in modeling such as computational and annotation costs. This observation stands in both discriminative (*e.g.* activity recognition, video question-answering) and generative video tasks (*e.g.* video generation or editing), leading to efficiency bottlenecks across video modeling pipelines, from training to inference. In this thesis, we introduce techniques that can mitigate such inefficiencies, with a focus on (1) making inference faster, and (2) training with freely-available signals.

First, we look into the inference pipelines of two challenging video modeling setups, namely, fine-grained activity recognition (*i.e.*, temporal activity detection) and diffusion-based video editing. In activity detection, traditional video models sample inputs at fixed temporal resolutions, having to consume redundant information. To avoid this, we introduce *Coarse-Fine Networks* which sub-samples inputs *dynamically* in time, by learning the importance of each frame, and in turn, reducing the compute footprint significantly. On the other hand, in video editing, typical state-of-the-art models incur heavy memory and computational costs to generate temporally-coherent frames, in the form of diffusion inversion or cross-frame attention. To alleviate such latency overheads, we introduce *Object-Centric Diffusion*, which allocates fewer computations towards background regions that are often-unedited or arguably less-important for perceptual quality, showing up to 10× speed-ups *in zero-shot*, for a comparable synthesis quality.

Next, we investigate the training pipelines of video models in terms of their annotation costs. To avoid the need for expensive frame-level labels when pretraining for activity detection, we introduce *Weakly-guided Self-supervised Detection Pretraining*. It leverages weak video-level labels to design a pretext task that emulates detection (*i.e.*, per-frame prediction). While requiring *no extra annotations*, our proposal outperforms prior art at the same training budget. Going beyond, we introduce *Video-conditioned Text Representations*, a video vision-language model (VLM) that supports learning form auxiliary semantic concepts (given as text), without any annotations. Starting from an image-VLM, we not only augment visual embeddings with temporal information, but also adapt text embeddings to video by grounding on visual modality. Our method shines especially in challenging setups where language can be more-revealing than vision (*e.g.* few-shot recognition), while gaining form label-free semantics.

Contents

List of Figures

List of Tables

Publications

The work presented in this thesis proposal also appeared in the following peer-reviewed articles or preprints:

1. Coarse-Fine Networks for Temporal Activity Detection in Videos
   Kumara Kahatapitiya, Michael S. Ryoo
   CVPR 2021

2. Weakly-guided Self-supervised Pretraining for Temporal Activity Detection
   Kumara Kahatapitiya, Zhou Ren, Haoxiang Li, Zhenyu Wu, Michael S. Ryoo, Gang Hua
   AAAI 2023

3. VicTR: Video-conditioned Text Representations for Activity Recognition
   Kumara Kahatapitiya, Anurag Arnab, Arsha Nagrani, Michael S. Ryoo
   CVPR 2024

4. Object-Centric Diffusion for Efficient Video Editing
   Kumara Kahatapitiya, Adil Karjauv, Davide Abati, Yuki M. Asano, Fatih Porikli, Amirhossein Habibian
   arXiv 2024

5. Language Repository for Long Video Understanding
   Kumara Kahatapitiya, Kanchana Ranasinghe, Jongwoo Park, Michael S. Ryoo
   arXiv 2024

Chapter 1

Introduction

1.1 Overview

Understanding and generating video content are critical for the progress of computer vision systems. Both directions facilitate the emergence of rich spatio-temporal representations, that can model important perceptual cues or nuances. Large-scale data is central to learning such representations, as in any present-day learning pipeline, and video data in specific, provides a more-natural and unique form of information beyond images. For instance, videos contain motion information, causal relationships, and even implicit geometry. More often than not, they are also coupled with other modes of data such as audio, speech or text that are complementary to vision. Despite being useful, the added temporal dimension and multiple modes of data can also be redundant, raising interesting questions about trade-offs. As such, video models are prone to efficiency bottlenecks, adding to any complications in image models, especially in the context of model complexity and annotation costs.

This observation motivates us to investigate the sources of inefficiency in video models, from training to inference, and introduce solutions that either (1) make inference faster, or (2) enable label-efficient training with freely-available signals. In this thesis, we first consider inference pipelines of two challenging video modeling setups, namely, fine-grained activity recognition (*i.e.*, temporal activity detection) and diffusion-based video editing. Next, we shift our attention towards training pipelines of video models designed for detection, recognition, or long-form understanding. We discuss each setting in detail in the following subsections.

1.1.1 Faster Inference

We investigate the inference pipelines of discriminative video models, specifically in the context of temporal activity detection (*i.e.*, fine-grained, per-frame activity recognition). This is a challenging application domain due to the nature of available benchmarks, which are small-scale (due to expensive annotations) and consist of complex overlapping activities spanning across longer periods of time. One main modeling difficulty here is capturing long-term motion from a continuous video, that may require either convolutions with large kernels [131, 133] or spatio-temporal attention modules acting on many tokens [5, 10]. Even with such compute-heavy operations, abstracting useful long-term dependencies is not sufficiently effective. Hence, a reliable and efficient video representation is necessary for activity detection. In recent years, video models have relied on frame striding or temporal pooling to cover a larger time interval without increasing the number of model parameters or computations. Since striding loses fine-grained details, such models often come with higher frame-rate pathways, forming two-stream (or multi-stream) architectures [41, 160]. These models confirm the benefits of frame striding paired with multi-stream architectures for extracting multi-scale temporal representations. Yet, they still rely on fixed sampling grids. Meaning, even with temporal striding (or, sub-sampling), such models still waste computations on redundant information. For instance, it is often unnecessary to feed almost identical frames to a model when there is little to no motion. Conversely, if a video is displaying a rapid motion, considering all such frames may be desired. This observation shows the need for a mechanism that samples frames non-uniformly conditioned on the input (*i.e.*, dynamically), that is also learnable. Such a proposal can significantly improve the inference speed of long-form fine-grained video models, which we explore in this thesis.

Similarly, we also consider diffusion-based video editing as a generative video modeling pipeline. In recent years, diffusion models [61, 63, 158, 178] have achieved incredible success, showing superior diversity and quality in generations that surpass the capabilities of earlier variants of generative models [50, 83]. Moreover, Latent Diffusion Models (LDMs) [158] have enabled scalability to high-resolution inputs, while also generalizing to different domains with minimal re-training. Subsequent literature builds on-top of LDMs with applications to image generation [158], editing [6], inpainting [158], as well as video generation [12] and editing [140, 203, 243]. However, diffusion models are also prone to various trade-offs, such as the inefficiency of diffusion sampling process (*i.e.*, denoising). During inference, it needs to apply a neural network (*e.g.* UNet [159]) iteratively for a number of steps, to go from noise to data distribution. Despite

the availability of techniques such as step distillation [117, 165] and accelerated samplers [110, 111, 178] that expedite image synthesis, efficient solutions for video generation and editing are still lacking. Moreover, video synthesis is far-more challenging compared to image synthesis due to the need for temporal consistency among generated frames, which relies on specialized components such as diffusion inversion [140, 178] and cross-frame attention [140, 243]. Without such expensive operations, outputs typically exhibit undesirable flickering artifacts. Considering these shortcomings, we devise a comprehensive study to understand and find solutions for the inefficiencies in diffusion-based video editing pipelines.

1.1.2 Free Training Signals

When exploring label-efficient training pipelines for video models, again we consider the domain of temporal activity detection. This is especially relevant due to the need for fine-grained (*i.e.*, per-frame) labels, that are expensive to collect in a large-scale, and hence, unavailable in current benchmarks. Therefore, activity detection models resort to being pretrained on classification datasets, learning to aggregate temporal information which is not ideal for the downstream that requires fine-grained predictions. Yet, this pretraining step is indispensable as also observed in many other applications such as object detection [34, 113], segmentation [138], reinforcement learning [168] or language modeling [104], as models learn to become more robust and generalize to unseen distributions by looking at more data. The gain from such pretraining however, depends on the alignment of pretraining and downstream tasks, which is minimal in the context of temporal activity detection. Simply put, models look at the bigger picture during pretraining, whereas they need to look at fine-grained details in the downstream. To address this disparity, prior work have proposed specific temporal [76, 132, 135] or graphical [47, 115] modeling in the downstream to capture aspects not seen during pretraining, which is insufficient. In contrast, we consider augmenting the pretraining data without any extra annotation effort, in a way that mimics the downstream, exploring a different training signal that is free and better-aligned.

In another direction, we investigate how vision-language models (VLMs) can benefit from freely-available semantic information, with application to diverse video modeling tasks from low-supervision regimes to long-form reasoning. In recent years, VLMs [72, 147] have emerged dominant, showing strong generalization capabilities across numerous domains including image classification [221, 233, 237], open-vocabulary object detection [53, 121], text-to-image retrieval [177, 225] and robot manipulation [73, 236]. Thanks to the paired visual-text encoders of VLMs, any semantic concept (given as a text input) can be embedded

in a joint latent space, giving intriguing zero-shot or few-shot transfer capabilities [3, 236] at inference. In the video domain however, training VLMs from scratch has been challenging due to the lack of paired data at scale. As a compromise, the common practice is to adapt pretrained image-VLMs to video, by introducing temporal information, especially to visual embeddings. Going beyond, we consider how important text embeddings can be when training video-VLMs, and how to best utilize semantic information [71, 196, 236] that is abundantly available. For instance, certain attributes (*e.g.* objects, scene or human subjects) are directly tied with specific activities, and can simplify their recognition. The presence of attributes such as [`rope`, `gym`, `one-person`] can narrow down the potential activity to `battling ropes` or `rope climbing`. Text embeddings of VLMs are especially suited to take advantage of such semantics. Any concept represented as text can be visually-grounded based on paired embeddings (in zero-shot), to extract attributes relevant for a given input that benefit recognition tasks. Such visually-grounded semantics are cheap in-terms of both annotation and compute costs, yet highly-useful, as we show in this thesis.

1.2 Organization

In this thesis, we propose techniques that alleviate efficiency bottlenecks of video models (both discriminative and generative), across both training and inference pipelines. The rest of the thesis is organized in the chronological order, aligning with the timeline of our contributions.

In Chapter 3, we first look into the inference pipelines of activity detection models, which usually spend resources for processing redundant information that comes with fixed resolutions (*i.e.*, regular sampling grids). Based on this observation, we propose (1) a new approach that allows a learnable dynamic selection of temporal frames, as well as (2) a method to fuse such sampled (*i.e.*, temporally *coarse*) representations with conventional, more temporally *fine* representations. We introduce the *Coarse-Fine Networks*. Our proposed Grid Pooling operation obtains better Coarse representations, while our Multi-stage Fusion mechanism best combines such Coarse representations with the conventional Fine representations. Unlike [41, 160], Grid Pooling learns to dynamically select informative frames (*i.e.*, in an input data-dependent manner), while managing the compute requirement without sacrificing the detection performance.

In Chapter 4, we propose a *weakly-guided self-supervised* pretraining method for activity detection, relying on large-scale classification data and *no extra annotations*. This mitigates the need for expensive fine-grained labels. The intuition is to aug-

ment pretraining data with video-level labels to perform frame-level predictions (*i.e.*, detection) during pretraining — bridging the gap with downstream conditions. Specifically, we first extend weak video-level labels of classification clips to create pseudo frame-level labels. Then, we propose three self-supervised augmentation techniques to generate multi-action frames and action segments within a clip. Namely, we introduce *Volume Freeze*, *Volume MixUp* and *Volume CutMix*. Volume Freeze creates a motion-less segment within a clip introducing segmented actions, whereas Volume MixUp and Volume CutMix seamlessly merge multiple clip segments into one, which tries to mimic the downstream data distribution of multiple actions per frame. Based on the augmented data, models are pretrained on an activity detection task. Our evaluations validate the benefits of the proposed pretraining strategy on multiple temporal activity detection benchmarks such as Charades [174] and MultiTHUMOS [229], with multiple models such as X3D [40], SlowFast [41] and Coarse-Fine [76]. We further investigate the extent of the detection-pretrained features in our ablations and, recommend when and how to use them best.

In Chapter 5, we go a step further to propose a framework for video understanding that can effectively learn from semantic concepts (given as text), *without any annotations*. We introduce *Video-conditioned Text Representations*, focusing on adapting text information to the video domain. More specifically, we design a video-VLM that jointly-trains both textual and visual features generated by an image-VLM. By finetuning text embeddings, we observe significant gains in our framework, compared to just finetuning visual embeddings (similar to the observations in [237]). We can also make use of freely-available auxiliary semantic information, represented in the form of visually-grounded text embeddings. Our video-conditioned text embeddings are unique to each video, allowing more-flexibility to move in the latent space and generalize to complex downstream tasks. Optionally, our video-conditioned auxiliary text can further help optimize this latent space. We evaluate `VicTR` on few-shot, zero-shot, short-form and long-form activity recognition, validating its strong generalization capabilities among video-VLMs.

In Chapter 6, we examine the inference pipelines of current state-of-the-art diffusion-based video editing frameworks, identifying key efficiency bottlenecks such as diffusion inversion and heavy cross-frame interactions. We show that significant improvements can be achieved by adopting off-the-shelf optimizations, namely, efficient diffusion samplers and token reduction techniques such as token merging (ToMe) [14, 15]. Additionally, we argue that video editing users may be particularly sensitive to the quality of edited foreground objects, as opposed to

background regions. Consequently, we propose two new and efficient techniques that harness this object-centric aspect of editing applications. Our first solution, *Object-Centric Sampling*, separates the diffusion process between edited objects and background regions, focusing the majority of the sampling steps on foreground areas. As a result, it can generate unedited regions with considerably fewer steps, all while ensuring faithful reconstructions in all areas. With our second solution, *Object-Centric ToMe*, we incentivise tokens in less-crucial background areas to be merged, while also exploiting temporal redundancies that are abundant in videos. The combination of such techniques, coined *Object-Centric Diffusion* (OCD), can seamlessly be applied to any video editing method without retraining or finetuning. As demonstrated in experiments, OCD speeds-up inference by a factor of 10× and 6×, in inversion-based and ControlNet-based pipelines respectively, while also decreasing memory consumption up to 17× for a comparable quality.

Finally, in Chapter 7, we provide the conclusions of our investigations and discuss ongoing research directions, namely, on efficient context-utilization of long-video LLMs and on the extension off-the-shelf video diffusion models to longer timespans.

Chapter 2

Literature Review

2.1 Video Understanding

Video understanding is about reasoning based on spatio-temporal inputs. Compared to image inputs, videos bring additional useful cues such as motion or multiple modalities (*e.g.* audio) into play, but also any associated complications such as increased compute requirements and redundancy in data. Convolutional networks (CNNs) [19, 185, 193, 209] and Recurrent models [37, 229] have been the state-of-the-art in video modeling, prior to the rise of Transformers [5, 9, 108, 162]. Multi-stream models [19, 41] that make use of different spatio-temporal views [41, 153] or modalities (*e.g.* optical-flow [19, 56], audio [66, 124, 155]) have emerged, tackling benchmark tasks such as activity recognition [79, 86], localization [52, 174, 229] or text-to-video retrieval [213]. To handle longer video inputs, models have focused on efficient temporal modeling [76, 131, 133], or memory mechanisms [163, 200, 202]. While Neural Architecture Search (NAS) has enabled efficient model designs [40, 160, 161], self-supervised methods [42, 56, 141, 153] have alleviated the high demand for annotated data. More recently, language-supervision has been of interest for video understanding due to the strong generalization capabilities shown in the image domain.

2.2 Fine-grained Activity Prediction

Making predictions per frame is significantly challenging compared to activity classification (i.e., making predictions per video). It has two flavors: (1) Temporal Activity Localization (TAL) which predicts activity proposals: boundaries and corresponding classes, assuming continuity of actions [17, 37, 54, 107, 183, 228,

238], and (2) Temporal Activity Detection which explicitly predicts classes per frame [32, 76, 135]. We focus on the latter. In this setup, sequential models such as LSTMs, [37, 228], fully-convolutional [172, 173, 211, 244], and transformer-based [33] approaches have shown promising results. Datasets for such tasks provide frame-level annotations with possibly multiple classes per frame [18, 174, 229].

2.3 Dynamic Sub-sampling

Selective processing of information has been of interest to the computer vision community. From Deformable convolutions [31] to Graphical networks [102, 166, 210], various core components of neural networks are based on this idea. Multiple recent works also try to address dynamic sampling of inputs, either spatially [44, 68, 152], temporally [85, 205, 206, 251] or spatio-temporally [118]. In Chapter 3, we explore temporal sub-sampling formulated as an end-to-end differentiable pipeline, which is also lightweight.

2.4 Limited Supervision in Videos

This includes unsupervised [49, 87, 169], self-supervised [23, 69], weakly-supervised [181] or semi-supervised [70] settings, based on the level of annotations used [29]. Self-supervision in particular, explores two directions: pretext tasks [122, 139, 154, 197, 250] or contrastive learning [25, 27, 58].

Prior work on temporal activity localization have explored limited supervision either during pretraining [4, 215, 216, 240], or the downstream [100, 106, 125, 157, 171, 232]. In Chapter 4, we focus on pretraining, defining a pretext task (as in self-supervision) which also depends on video-level weak annotations to do fine-grained predictions (as in weak-supervision). We keep the downstream settings unchanged, with full supervision. Our formulation has the flavor of temporal activity detection, predicting frame-level labels instead of temporal proposals as in TAL. Thus, our proposal is orthogonal to above work on pretraining, but can be complementary to those on downstream finetuning.

2.5 Vision-Language Models

Vision-Language Models (VLMs) are usually trained on internet-scale paired visual-language (*e.g.* image-text) data. Seminal work such as CLIP [147] and ALIGN [72] have shed the light on the capabilities of such models, especially for zero-shot

transfer. Since then, VLM literature has flourished, with applications in open-vocabulary object detection [53, 121], open-set classification [142], retrieval [7, 177, 225], captioning [223], segmentation [148, 214], robot manipulation [73, 78, 236] and many other domains. Although VLMs are generally trained on image-text data, there are intuitive variants which are trained either only on images [186] or only on text [127]. The commonly-used similarity-based objective of VLMs has also been repurposed to specialized domains, through prompt learning [248] or engineering [53, 128]. The text encoder of VLMs can be a powerful mapping from semantic concepts to latent embeddings [119]. Many foundation models [3, 230, 233] follow similar design principles as VLMs, thriving in zero-shot [55] or few-shot [248] settings. Recent work combining Large Language Models (LLMs) with VLMs show how language can act as a communication-medium between models [196, 236, 245]. In [119], authors use an LLM to represent object classes as a set of its semantic attributes, to learn a better classifier.

As for video-VLMs, they are either trained from scratch on video-text data [212, 223], or more-often than not, finetuned initializing from a pretrained image-VLM [30, 98, 221]. Some are even trained on both image and video data paired with text [7]. The success of VLMs in the image domain has fueled similar research directions in the video domain.

2.6 Adapting Image-Text Models to Video

Adapting image-text models is a common practice when designing video-VLMs. A general and effective recipe for such adaptation is proposed in [30]. It consists of temporal modeling, multi-modal fusion, auxiliary training objectives, and both image/video data at scale. All others usually make use of a subset of these concepts. CLIP-ViP [219] is trained with different sources of data and multiple cross-modal training objectives. VideoCoCa [221] extends CoCa [230] with attention-pooled frame embeddings, which are used to decode text captions in a generative framework. MOV [142] is trained with additional audio/flow encoders through cross-modal attention, keeping image-text encoders frozen. Video-specific prompts can also be learned with such frozen encoders [75]. Vi-Fi [150] shows that simply finetuning CLIP image-text encoders without any specialized modules can generate video representations efficiently.

Apart from the above, there exists a body of prior work that closely-relates to our proposal in Chapter 5, *Video-conditioned Text Representations*. ActionCLIP [190] upgrades its CLIP image-encoder with (1) parameter-free temporal layers (TSM [97]) within the backbone, and (2) a temporal transformer head, while keeping

the text-encoder fixed. Similarly, CLIP4clip [112] just uses a temporal transformer head to update visual embeddings. CLIPHitchhiker's [8] generates *text-conditioned video* embeddings by temporally-pooling frame embeddings, conditioned on each text query. In this case, a given video generates multiple different visual embeddings, one per each text embedding. EVL [99] completely discards text. It acts as an initialization for a visual-only backbone, consisting of CLIP image encoder and a temporal, class-conditioned decoder. X-CLIP [126] introduces trainable temporal layers within its backbone image encoder, and generates video-specific text prompts. Meaning, it finetunes both encoders similar to VicTR. However, it does not allow interaction among text embeddings, nor with fine-grained visual information (but only, with temporally-aggregated information). Hence, it shows limited gains from adapting text to video domain. In contrast, our video-conditioned text embeddings that are unique for each video, interacts with both fine-grained visual embeddings and other text embeddings, to enable a better contrastive framework, and in-turn, a more-flexible alignment in the latent space, as we show in Chapter 5.

2.7 Text-based Video Generation and Editing

There has been a surge in text-to-video generation methods using diffusion models. The early works inflate the image generation architectures by replacing most operations, *i.e.*, convolutions and transformers, with their spatio-temporal counterparts [62, 64, 65, 176, 247]. Despite their high temporal consistency, these substantial model modifications require extensive model training on large collection of captioned videos.

To avoid extensive model trainings, recent works adopt off-the-shelf image generation models for video editing in one-shot and zero-shot settings. One-shot methods rely on test-time model adaption on the editing sample, which is unfeasible for real-time applications [38, 203]. Zero-shot methods integrate training-free techniques into the image generation models to ensure temporal consistency across frames [45, 80, 103, 140, 243] most commonly by: *i)* providing strong structural conditioning from the input video [26, 38, 243] inspired by ControlNet [242]; *ii)* injecting the activations and attention maps extracted by diffusion inversion [178, 187] of the input video into the generation [45, 140]; *iii)* replacing the self-attention operations in the architecture with temporal counterparts operating on neighboring frames [140, 203, 243]. Introducing cross-frame attention greatly increases the temporal consistency, especially when involving more and more frames in the operation [243]. Despite their effectiveness, all these solutions

come with additional computational costs, which are currently underexplored in the literature.

2.8 Efficient Diffusion Models

Due to their sequential sampling, diffusion models are computationally very expensive. Several studies analyze the denoising U-Net architecture to enhance its runtime efficiency, achieved either through model distillation [81] or by eliminating redundant modules [95]. Other ways to obtain efficiency gains include enhanced noise schedulers [110, 111] and step distillation techniques [95, 117, 165]. Moreover, the work in [92] enables efficiency in image editing applications by caching representations of regions left unedited, yet it cannot handle videos and it strictly requires human-in-the-loop to operate. The most relevant to our *Object-Centric Diffusion* is the Token Merging technique presented in [15] that demonstrated advantages for image generation [14]. Although very proficient, we observe it does not readily work when deployed in video editing settings. For this reason, we optimize this technique for the task at hand, such that it exploits redundancy of tokens across frames and directing its token fusing towards background regions. Finally, Token Merging was applied to video editing use cases in VidToMe [93], where its introduction is aimed at improving temporal consistency of generated frames. Differently, we tailor it in OCD to improve efficiency without sacrificing fidelity, resulting in much faster generations, as we show in Chapter 6.

Chapter 3

Coarse-Fine Networks for Efficient Activity Detection

3.1 Overview

Learning to represent videos is important. It requires embedding both spatial and temporal information in a sequence of frames, often implemented with 3D convolutions. Leaning to build good video representations is crucial for various vision tasks including action classification, video object segmentation, and complex human activity recognition as well as temporal localization of such activities.

One of the main challenges in video representation learning is in capturing long-term motion from a continuous video. In order for a convolutional neural network to abstract long-term motion information across many frames, a large number of (spatio-)temporal conv. layers (or such layers with large kernels) are necessary, requiring many parameters. This, combined with the difficulty in obtaining large-scale annotated videos and increased computation time, makes the learning of the video representation very challenging for non-atomic activities. This is even more challenging for temporal activity detection (i.e., localization), as the activities may very often temporally overlap. A mechanism to reliably and efficiently capture various motion in videos is necessary.

Use of frame striding or temporal pooling (i.e., lowering the frame rate) has been a successful strategy to cover a larger time interval without increasing the number of model parameters. Since such striding loses fine details of frame changes, it was often paired with another CNN tower taking an input with a higher frame rate, forming a two-stream (or multi-stream) CNN architecture as was done in SlowFast [41] and AssembleNet [160]. These models confirmed the benefits of

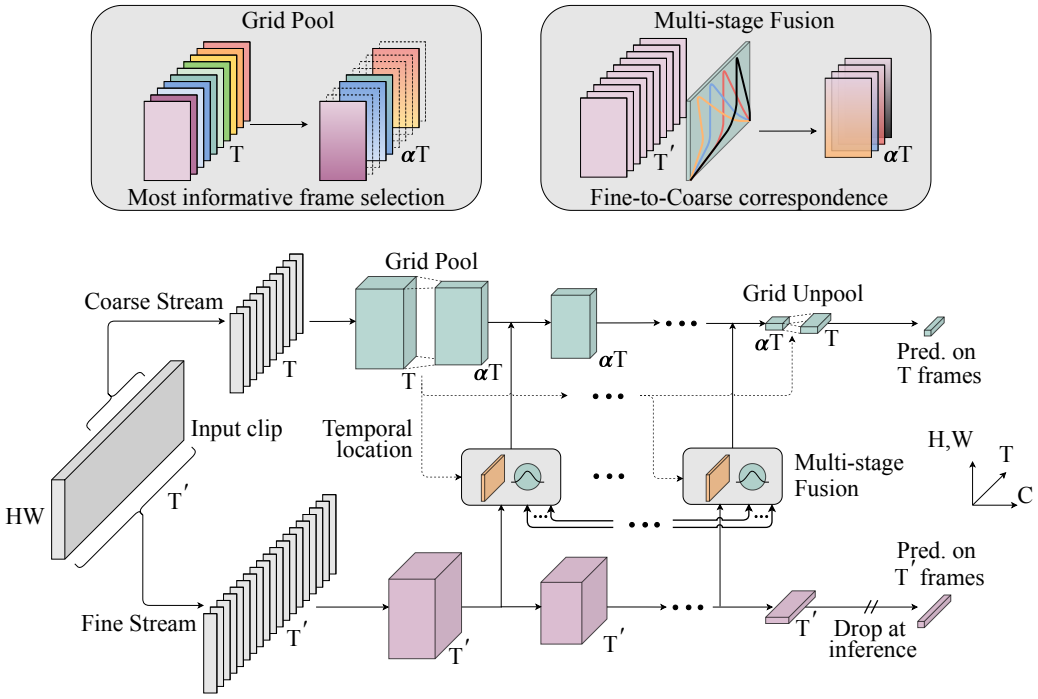

Figure 3.1: **Coarse-Fine Networks** process information at two different temporal resolutions. The Coarse stream learns to sample the most informative frame locations through a learnable downsampling operation: *Grid Pool*, whereas the Fine stream process the entire temporal duration of the input to extract a fine-grained context. The connections in-between the two streams: *Multi-stage Fusion*, provide multiple abstraction-levels of the fine-grained context, calibrated to the temporal locations of the coarse frames. For Charades dataset [174], we configure our network to use $T = 64$, $T^{'} = 128$ and $\alpha = 1/4$.

frame striding as well as multi-stream architectures to combine representations with multiple temporal resolutions.

However, although using temporal striding (with a multi-stream multi-resolution architecture) allows the model to more easily process long-term motion, they are limited as it ignores 'importance' of each frame. Informativeness of each frame is different. It is often unnecessary and redundant to feed almost identical frames as an input to the model when there is no/little motion in video frames. On the other hand, if a human in the video is displaying a rapid motion, taking all such frames into consideration is desired. Uniform temporal striding or pooling is incapable of such dynamic frame selection.

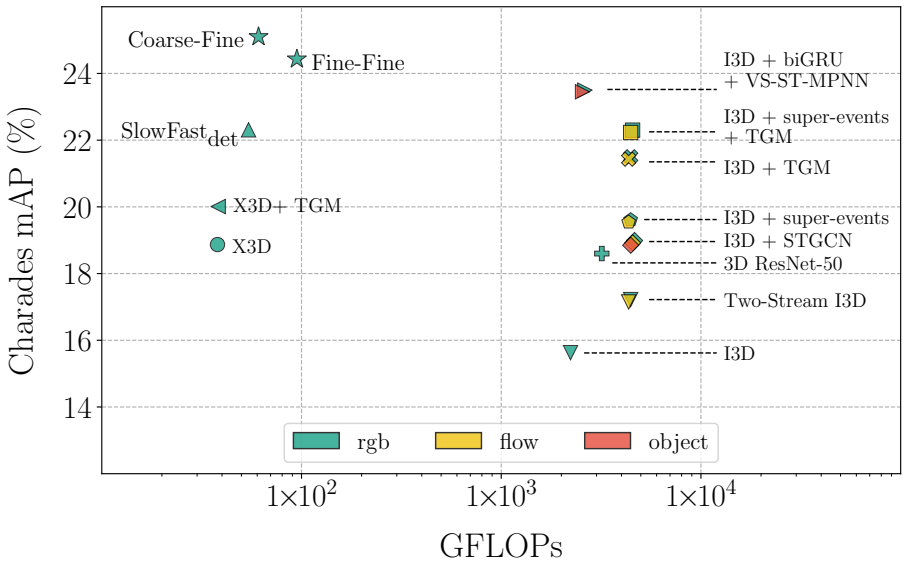

Figure 3.2: **Performance/complexity trade-off** of state-of-the-art methods for activity 'localization' in Charades [174]. Our Coarse-Fine Networks achieve superior performance than the previous best-performing method in literature, with more than one order of magnitude reduction in compute. Moreover, we do not use any additional modalities such as optical flow or object detections.

In this paper, we propose (1) a new approach that allows a learnable dynamic selection of temporal frames within the model, as well as (2) a method to fuse such sampled (i.e., temporally 'coarse') representations with conventional, more temporally 'fine' representations. We introduce the *Coarse-Fine Networks*. A new component named temporal Grid Pooling is presented to obtain better Coarse representations, and the Multi-stage Fusion is introduced to best combine such Coarse representations with the conventional Fine representations. Unlike [41, 160], our Grid Pooling learns to dynamically select informative frames. Fig. 3.1 illustrates the overview of the model, and Fig. 3.2 shows the benefits of the model, which we discuss more in the paper.

3.2 Coarse-Fine Networks

Coarse-Fine Networks explore how video architectures can benefit from different abstractions of temporal resolution and long-term temporal information. As shown in Fig. 3.1, we do this by processing the information at two different temporal

resolutions: coarse and fine, in a two-stream architecture. The Coarse stream learns to (differentiably) select the most informative frame locations, essentially performing a learned temporal downsampling to abstract a lower temporal resolution. In contrast, the Fine stream processes the input at the original temporal resolution and provide a fine-grained context to the Coarse stream through a fusion mechanism. To abstract this context information, the Fine stream always looks at the full temporal duration of the input clip (which later gets pooled with Gaussians), whereas the Coarse stream can either look at a shorter clip or the entire clip depending on the inference interval.

In Coarse-Fine Networks, we address two key challenges: (i) how to abstract the information at a lower temporal resolution meaningfully, and, (ii) how to utilize the fine-grained context information effectively. First, to abstract coarse information, we propose *Grid Pool* (Sec. 3.2.1), a learnable temporal downsampling operation which adaptively samples the most informative frame locations with a differentiable process. Secondly, to effectively use the fine-grained context provided by the Fine stream, we introduce *Multi-stage Fusion* (Sec. 3.2.2), a set of lateral connections between the Coarse and Fine streams, which looks at multiple abstraction levels of fine-grained information.

3.2.1 Grid Pool

Our temporal Grid Pool operation learns the most informative frame locations from a given input clip, and samples the representations corresponding to the locations based on interpolation. In fact, it can be viewed as a learnable temporal downsampling layer with a small compute overhead, which can replace the conventional temporal pooling operations. However, in contrast to these pooling operations, our Grid Pool samples by interpolating on a non-uniform grid with learnable (and adaptive) grid locations as shown in Fig. 3.3. First, a lightweight head (h) projects the input feature ($\mathbf{X}^C$) of temporal length T to a set of confidence values $\{p_i\}_{i=1,\cdots,\alpha T}$, where $\alpha < 1$ and αT is an integer (e.g., $\alpha = 1/4$ and $T = 128$). These confidence values represent how informative each temporal interval with a size of $1/\alpha$ (e.g., 4 frames if $\alpha = 1/4$) is, and is modeled as a function of the input representation $\mathbf{X}^C$:

$$\{p_i\}_{i=1,\cdots,\alpha T} = h(\mathbf{X}^C)\,. \tag{3.1}$$

The intuition here is to sample frames at a higher frame rate where the confidence (i.e., informativeness) is high and at a lower frame rate where it is low. In other words, the stride between the interpolated frame locations should be small where the confidence is high, and vice-versa. We normalize these confidence

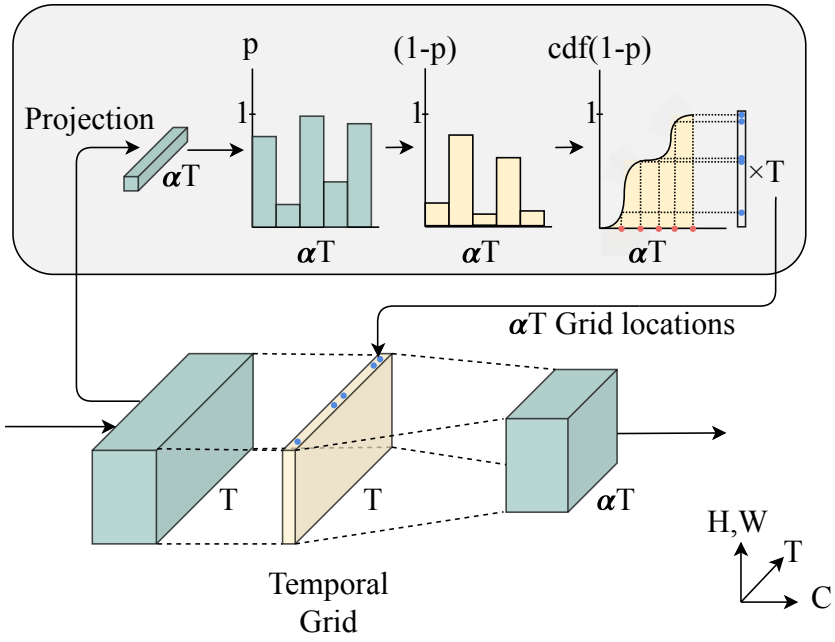

Figure 3.3: **Grid Pool** layer learns a temporal downsampling operation based on non-uniform grid locations. αT number of points are differentiablly sampled from an input feature of length T by learning their importance. We interpret p_i as the importance of each frame location. Since we want to sample with a lower sampling duration (i.e., a higher frame rate) where we have a higher importance, we construct $\texttt{cdf}(1-p_i)$ for sampling.

values p_i since we need the relative (not absolute) confidence to capture the relative importance of frames. To get a set of αT grid locations based on confidence values, we consider the Cumulative Distribution Function $\{\texttt{cdf}(1-p_i)\}_{i=1,\cdots,\alpha T}$, which is a non-uniform and monotonically-increasing function. The input of the Grid Pool layer $\mathbf{X}^C$ is sampled/interpolated based on these grid locations to get the output $\tilde{\mathbf{X}}^C$, while making it fully differentiable for backpropagation. This process can be represented as,

$$
\begin{aligned}
q_t &= T \cdot \texttt{cdf}(1-p_t) = T \cdot \frac{\sum_{i=1}^{t}(1-p_i)}{\sum_{i=1}^{\alpha T}(1-p_i)}, \\
\tilde{\mathbf{X}}^C &= I\Big(\mathbf{X}^C, \{q_t\}_{t=1,\cdots,\alpha T}\Big),
\end{aligned}
\tag{3.2}
$$

where q_t represents the grid location t, and $I(\cdot)$ represent the temporal sampling function. Here, when a grid location is non-integer, the corresponding sampled

frame is a temporal interpolation between the adjacent frames. We do not perform any spatial sampling in the Grid Pool layer.

Grid Unpooling: A temporal interpolation based on a non-uniform grid as such can affect the temporal structure of the propagated features. Before feberating the final output, the frame-wise predictions of the network should be re-aligned properly for activity detection tasks. To do this, we introduce a *Grid Unpool* operation, which is coupled with the grid locations learned by the Grid Pool layer. This does not have any learnable parameters and simply performs the inverse operation of the former. First, αT re-sample grid locations are calculated based on the inverse mapping of the `cdf`, based on which, the logits are re-sampled to acquire the original temporal structure. The idea is to re-sample with a low frame-rate in the regions where we used a high frame-rate in Grid Pool, and vice-versa. Any non-integer frame location is temporally interpolated similar to Eq. 3.2. Finally, these logits are uniformly upsampled through interpolation to fit the input temporal resolution. For classification tasks, the Grid Unpool operation may not be necessary as a global pooling of the logits is considered as the prediction.

3.2.2 Multi-stage Fusion

We introduce Multi-stage Fusion, a set of lateral connections between the two streams as shown in Fig. 3.4, to fuse the context from the Fine stream with the Coarse stream. We consider three main design choices here: (i) it should be capable of filtering out which fine-grained information should be passed down to the Coarse stream, (ii) it should have a calibration step to properly align the fine features with the coarse features based on their relative temporal locations, and (iii) it should be able to learn and benefit from multiple abstraction-levels of fine-grained context at each fuse-location in the Coarse stream. Our design tries to address these aspects.

Filtering fine-grained information: First, to decide which fine-grained context should be passed through to the fusion process, the fine feature $\mathbf{X}^F_{l_i}$ at abstraction-level l_i is multiplied with a self-attention mask. This mask is calculated by processing the fine feature through a lightweight head (g) consisting of point-wise convolutional layers followed by a sigmoid non-linearity.

$$\bar{\mathbf{X}}^F_{l_i} = \mathbf{X}^F_{l_i}\, g(\mathbf{X}^F_{l_i})$$

Fine-to-Coarse correspondence: The attention-weighted fine feature $\bar{\mathbf{X}}^F_{l_i}$ still needs to be calibrated for the temporal location of each coarse feature. Since the Coarse and Fine streams does not necessarily process the same, properly aligned temporal length because of our non-uniform Grid Pooling, we need to explicitly compute the frame correspondence. To make this calibration, we use a set of temporal Gaussian distributions centered at each coarse frame location $\{\mu^C_j\}_{j=1,\cdots,\alpha T}$ which abstract a location-dependent weighted average of the fine feature. We use αT such *Coarse-centric Gaussians*, each having a temporal length of $T^{'}$ and a standard deviation σ which is a fraction of this length. We found that considering the center and scale of these Gaussians to be hyperparameters rather than making them learnable, gives a better performance, possibly due to relatively simpler training. This calibration step can be viewed as,

$$
\begin{aligned}
G^C_j &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{(t-\mu_j)^2}{2\sigma^2} \Big|_{j=1,\cdots,\alpha T}, \\
\hat{\mathbf{X}}^F_{l_i} &= \bar{\mathbf{X}}^F_{l_i} \cdot G^C,
\end{aligned}
$$

where G^C is the stacked Coarse-centric Gaussians and $t \in [0, T^{'}-1]$.

Multiple abstraction-levels: The feature $\hat{\mathbf{X}}^F_{l_i}$ still corresponds to a single abstraction-level l_i of fine features, where we have Multi-stage Fusion connections in multiple abstraction-levels, i.e., depths of the network. Therefore, we allow each fusion connection to look at the features from all abstraction levels by concatenating them channel-wise (after adjusting the spatial resolution through max pooling), and performing point-wise (i.e. $1\times1\times1$) convolutions to get the final scale ($\mathbf{A}_{l_i}$) and shift ($\mathbf{B}_{l_i}$) features at each fusion location. This can be represented as,

$$
\begin{aligned}
\hat{\mathbf{X}}^F &= \oplus_{i=1}^{n} \hat{\mathbf{X}}^F_{l_i}, \\
\mathbf{A}_{l_i} &= f^{\mathbf{A}}_{l_i}(\hat{\mathbf{X}}^F), \mathbf{B}_{l_i} = f^{\mathbf{B}}_{l_i}(\hat{\mathbf{X}}^F), \\
\hat{\mathbf{X}}^C_{l_i} &= \mathbf{A}_{l_i} \tilde{\mathbf{X}}^C_{l_i} + \mathbf{B}_{l_i}.
\end{aligned}
$$

where $\oplus$ is the channel-wise concatenation of features from n abstraction-levels and, $f^{\mathbf{A}}_{l_i}$ and $f^{\mathbf{B}}_{l_i}$ represent projection heads to calculate the scale and shift features at each fusion location l_i, respectively. This design enables Multi-stage Fusion to process multiple abstraction-levels of fine-grained context through filtering and temporal calibration.

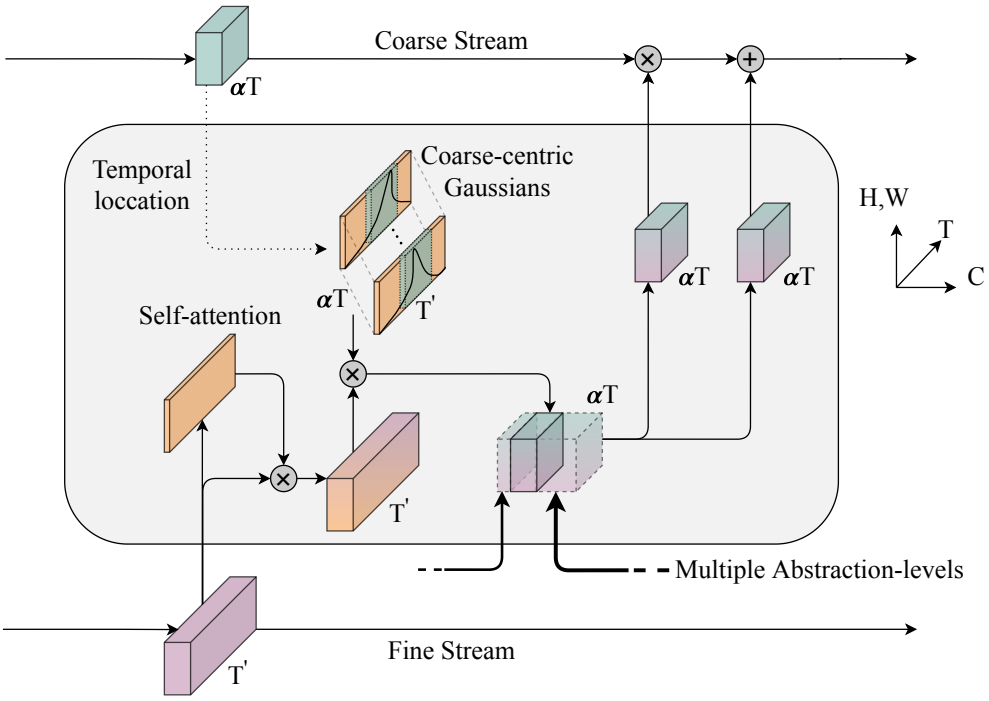

Figure 3.4: **Multi-stage Fusion** feeds multiple abstraction-levels of fine-grained context to the Coarse stream. First, Fine stream features get filtered through a self-attention mask. Then, these features get calibrated for each coarse frame, based on Gaussian weights centered at the corresponding coarse frame location. Finally, such calibrated features from multiple abstraction-levels get point-wise convolved to calculate the scale and shift features which provide an affine transformation to the coarse features.

3.2.3 Model Details

The network architecture used as the backbone in Coarse-Fine Networks is adopted from X3D [40], which follow a ResNet [57] structure, but designed for efficiency in video models. Both Coarse and Fine streams are initialized with separate sets of parameters, but have the same number of layers and filters as shown in Table 3.1, except for the addition of Grid Pool in the Coarse stream. Since the X3D architecture perform no temporal downsampling or pooling, it follows aggressive striding at the input level to be computationally efficient, which is set be a stride of 10 in our case. This allows the input of the Coarse stream to cover a large temporal region, compared to what common backbones such as I3D [20] cover during training. This is beneficial, particularly in datasets with longer temporal

Stage	Filters		Output size $T\times S^2$	
	Coarse	Fine	Coarse	Fine
input	stride 10, 1^2		$T\times 224^2$	$T'\times 224^2$
conv_1	1×3^2, 3×1, 24		$T\times 112^2$	$T'\times 112^2$
res_2	$\begin{bmatrix} 1\times 1^2, 54 \\ 3\times 3^2, 54 \\ 1\times 1^2, 24 \end{bmatrix}\times 3$		$T\times 56^2$	$T'\times 56^2$
grid pool	stride $1/\alpha$, 1^2	None	$\alpha T\times 56^2$	$T'\times 56^2$
res_3	$\begin{bmatrix} 1\times 1^2, 108 \\ 3\times 3^2, 108 \\ 1\times 1^2, 48 \end{bmatrix}\times 5$		$\alpha T\times 28^2$	$T'\times 28^2$
res_4	$\begin{bmatrix} 1\times 1^2, 216 \\ 3\times 3^2, 216 \\ 1\times 1^2, 96 \end{bmatrix}\times 11$		$\alpha T\times 14^2$	$T'\times 14^2$
res_5	$\begin{bmatrix} 1\times 1^2, 432 \\ 3\times 3^2, 432 \\ 1\times 1^2, 192 \end{bmatrix}\times 7$		$\alpha T\times 7^2$	$T'\times 7^2$
conv_5	1×1^2, 432		$\alpha T\times 7^2$	$T'\times 7^2$
pool_5	None$\times 7^2$			
fc_1	1×1^2, 2048		$\alpha T\times 1^2$	$T'\times 1^2$
fc_2	1×1^2, #classes			
grid unpool	stride α, 1^2	None	$T\times 1^2$	$T'\times 1^2$

Table 3.1: **Coarse-Fine Network Architecture** is adopted from X3D [40], more specifically from the version X3D-M. Both streams have the same design and parameters, except for the addition of Grid Pool layer and Grid Unpool operation in the Coarse stream. The Fine stream process the entire temporal length of the input T' to provide a fine-grained context, whereas the Coarse stream can look at a segmented clip of length T, for which the frame-wise predictions are required. Here, $\alpha < 1$ and αT is an integer. The kernel shapes follow the standard notation $\{T \times S^2, C\}$.

duration. The Coarse stream takes in segmented clips of $T = 64$ frames to follow the standard X3D architecture after the Grid Pool layer (with $\alpha = 1/4$) during training, while processing the input fully convolutionally at inference (i.e., $T = 128$ frames during testing). In contrast, the Fine stream always process the entire input

clip, which is capped at a maximum of $T^{'} = 128$ frames. This limit should be adjusted based on the dataset to include the entire duration of input clips. We found $T^{'} = 128$ frames with a stride of 10 is adequate to cover the entire temporal length of more than 90% of the Charades [174] videos.

The main difference between the Coarse stream and the Fine stream is the Grid Pool layer and the corresponding Grid Unpool operation. We want to perform this learned temporal downsampling as early as possibly in the network to reduce the compute, but at the same time, having good enough features to learn the grid locations. Thus, we place the Grid Pool layer after the first residual block $\texttt{res}_2$. We find that downsampling by a factor of 4 works well in practice, to have a good compute/performance trade-off (Table 3.3d). To calculate the confidence values (p) in the Grid Pool layer, we use a lightweight head (h) of 3 strided convolutions with a total temporal stride of 4 and a spatial stride of 8, followed by spatial average pooling and a sigmoid non-linearity. The Grid Unpool operation has no learnable parameters. It is coupled with the grid locations predicted by the Grid Pool layer to perform the inverse operation of the former to recover the original temporal structure at the logits level.

We try to follow a lightweight design in Multi-stage Fusion as well. The self-attention mask $\hat{\mathbf{X}}^F_{l_i}$ is calculated through a head (g) of 2 point-wise (i.e. $1 \times 1 \times 1$) convolutions followed by a sigmoid non-linearity. The Coarse-centric Gaussians (G^C) have no learnable parameters, and the peak magnitude of each mask is normalized to 1. The standard deviation σ is set to be $T^{'}/8$, empirically. The two heads $f^{\mathbf{A}_{l_i}}$ and $f^{\mathbf{A}_{l_i}}$ at each fusion location which project the concatenated multi-stage features ($\hat{\mathbf{X}}^F$) to scale ($\mathbf{A}_{l_i}$) and shift ($\mathbf{B}_{l_i}$) features contain a single point-wise convolution each. The scale features go through an additional sigmoid non-linearity. We further discuss the complexity (compute and parameters) of these operations in our ablations (subsection 3.3.2).

3.3 Experiments

We evaluate Coarse-Fine Networks on two large-scale benchmarks for activity detection: Charades [174] and MultiTHUMOS [229]. Note that we focus on the temporal detection (i.e., localization) task of generating multi-label activity annotations at every time step, which is more challenging than video classification. Activities may temporally overlap (e.g., sitting and eating), and the model must be trained to annotate all of them at each time step.

3.3.1 Charades

Dataset: Charades [174] is a large scale dataset consisting of ~9.8k continuous videos with frame-wise annotations of 157 common household activities. The dataset is split as ~7.9k training and ~1.8k validation videos. Each video contains an average of 6.8 activity instances, often with multiple activity classes per frame, and has longer clips averaging a duration of ~30 seconds. Such a long duration makes it a suitable dataset to test Coarse-Fine Networks.

Training: We initialize both Coarse and Fine streams of our network with the X3D backbone pretrained on Kinetics-400 [79]. For the actual training of the Coarse-Fine network as well as baselines, we follow a two-stage training process: first, training the two streams separately, followed by finetuning the combined streams.

In the first stage, the Coarse stream considers an input of 64 frames sampled at a stride of 10, whereas the Fine stream considers 16 frames sampled at the same stride. This allows both streams to process same-sized features after Grid Pool layer. We use $\alpha = 1/4$ in our experiments. Each stream is trained for 100 epochs with a batch size of 16 at a learning rate of 0.02 at the start, which is decreased by a factor of 10 at 60 and 80 epochs.

In the second stage, the two streams are trained together as Coarse-Fine Networks, with Multi-stage Fusion parameters initialized from scratch. We train this for another 100 epochs with the same schedule and batch size, but use 10× increased learning rate for the newly-initialized parameters of the fusion layers. Here, the Fine stream process the entire duration of the input, which is capped at 128 frames (sampled at a stride of 10) for Charades [174]. During both stages, each input is randomly sampled in [256,320] pixels, spatially cropped to 224×224 and applied a random horizontal flip. We use a dropout rate of 0.5 before the logits layer. The logits go through a sigmoid to make multi-label predictions for each frame. We use an average of classification and localization loss for training, similar to previous methods [132, 135].

Inference: At inference, we make predictions for every frame. Even though our input is sampled at a stride of 10, we consider the labels for all frames (at a stride of 1) and interpolate the logits to fit the original temporal length. In other words, we evaluate our models so that the predictions are more fine-grained at original temporal resolution. However, all state-of-the-art methods in Table 3.2 report the performance for 25 equally-sampled frames per each input, which is the original Charades localization evaluation [174] setting. Therefore, to make a fair comparison, we evaluate our models in the same setting, only making predictions

model	flow	obj.	mAP (%)	GFLOPs	Param (M)
I3D (Inception) [20]			15.63	2223.03	12.45
Two-stream I3D [20]	✓		17.22	4446.10	24.90
3D ResNet-50 [57, 184]			18.60	3187.63	46.52
X3D [40]			18.87	37.96	3.29
X3D-L [40]			20.03	147.04	5.78
I3D + super-events [132]	✓		19.41	4446.15	26.18
I3D + TGM [135]	✓		21.50	4446.66	27.00
I3D + super-events + TGM [135]	✓		22.30	4446.75	28.28
I3D + STGCN [47]	✓	✓	19.09	4450.94	29.18
I3D + biGRU + VS-ST-MPNN [115]		✓	23.70	2223.03+	12.45+
X3D [40] + TGM [135]			20.01	38.26	4.35
$\text{SlowFast}_{\text{det}}$ (with X3D)			22.31	54.31	7.41
Fine-Fine (ours)			**24.43**	94.80	7.80
Coarse-Fine (ours)			**25.10**	73.37	7.82

Table 3.2: **Comparison with the state-of-the-art methods for activity detection on Charades** [174]. We report the performance (mAP), compute requirement to process a clip of 128×10 frames (GFLOPs), and the number of parameters (M). These results correspond to the original Charades localization evaluation settings. Coarse-Fine Networks significantly outperform the previous state-of-the-art with +1.4%mAP relative improvement, while reducing the compute requirement by more than one order of magnitude. It is worth noting that we do not use additional input modalities, i.e., optical-flow or object detections as any of the previous methods. The source of [115] is not available to us to calculate its exact complexity values.

for 25 equally-sampled frames per input. The evaluation script from the Charades challenge scales the Average Precision for each class with a corresponding class weights. At inference, the inputs are center-cropped to 224×224. We report the performance as mean Average Precision (mAP) and measure the compute requirement to process an input clip of 128×10 frames, for which our network processes only 128 frames due to input striding. The compute is reported as GFLOPs (floating-point operations $\times 10^9$) and the number of learnable parameters in millions (M), i.e., $\times 10^6$. We do not take advantage of any multi-crop inference.

Results: We compare the performance of the Coarse-Fine Networks with the state-of-the-art methods on Charades [174] localization task (i.e., temporal activity

detection) in Table 3.2. For this evaluation, we use the standard test setting (i.e., the official 'Charades_v1_localize' evaluation) where we make predictions for equally-sampled 25 frames in the validation set. This is the same procedure followed in previous work [115, 132, 135]. We report the performance (mAP), compute requirement to process a clip of 128×10 frames (GFLOPs) and the number of parameters (M).

We are able to confirm that our Coarse-Fine Network performs better than all previous methods, establishing the new state-of-the-art number of 25.10 on Charades localization. The Coarse-Fine Network, which only uses RGB, is not only superior to the previous RGB models but also is better than the methods using additional inputs modalities (i.e. optical-flow and object detection). It shows a relative improvement of +1.4%mAP compared to the previous best performing method [115], which benefits from additional training data (for its object detector training) and additional input modality (objects).

We also note that the Coarse-Fine Network is extremely computationally efficient. Compare to the previous models, it often requires only ~1/75 of computations (e.g., 73 vs. 4446 GFLOPS). Further, this computation is without considering the overhead for optical flow computation or object detection in prior works. The significant computation efficiency of the Coarse-Fine Networks is due to the better utilization of the RGB features, which discards the need for processing additional modalities, as well as an efficient usage of X3D modules with our temporal Grid Pooling and Multi-stage Fusion, which we confirm further with our ablations in the next section.

We further report a version of our method: Fine-Fine Networks, in which the Grid Pool layer is removed from the Coarse stream, to highlight the importance of the Coarse features. Fine-Fine Networks still benefit from our Multi-stage Fusion. The Grid Pool operation dynamically sample important frames to generating a coarse temporal resolution, which gives the Coarse-Fine Networks an relative performance gain of +0.67% mAP and 23% computation reduction. We also evaluated the baseline extension of X3D as a two-stream network (with different temporal resolutions) in a form similar to [41], which we name $\text{SlowFast}_{\text{det}}$. This does not have our Grid Pool layer or the Multi-stage Fusion mechanism. The result shows the benefits of the component, giving a relative improvement of 2.79%mAP. A larger version of X3D (i.e., X3D-L) shows that the performance improvement of Coarse-Fine compared to X3D is not merely due to the increased compute.

It is important to also note that all previous methods work on pre-extracted features from a frozen backbone, essentially making them late-modeling techniques, either using graph-based methods [47, 115] or abstracting long-term temporal

information [132, 135]. In contrast, our method allows end-to-end training of feature fusions at intermediate locations of the network, enabling it to learn better representations using only RGB information.

Fig. 3.2 further highlights the benefit of Coarse-Fine Networks compared to previous state-of-the-arts. We show the performance/complexity trade here, with the x-axis (complexity in GFLOPs) in log-scale. Our method shows comparable performance with the previous best performing method which outperforms all previous state-of-the art methods, while being extremely efficient in design.

3.3.2 Ablations

Here, we discuss multiple ablation experiments validating our design decisions, specifically on our Grid Pool layer and Multi-stage Fusion. We use the Charades dataset (with the localization setting as above).

In our ablation experiments, we take advantage of more robust evaluation metric to compare our approach and the baselines. We make the model to generate a multi-class activity annotation at every time step, and compare it with the ground truth to measure the mAP. This is very similar to the official 'Charades_v1_localize setting' used above, except that (i) it is evaluated for $\times 25$ more frames for the completeness and that (ii) we measure mAP per activity class and average them.

Fusion location: First, we explore which locations in our two stream architecture would be ideal to implement the fusion connections. We consider the late fusion as a baseline, in which, the only fusion happens just before the logits layer. This is similar to the previous methods in [132, 135]. Next, We extend this fusion to multiple intermediate levels, specifically, after each `res` block, in which we fuse the two streams at only equivalent abstraction levels, i.e., at the same depth. This is similar to the fusion in SlowFast [41]. Finally, we consider multiple abstraction-levels of Fine features for the fusion, which gives our Multi-stage Fusion. The results of this ablation is given in Table 3.3a. Note that here we evaluate our fusion in a Fine-Fine Network to decouple the effect of Grid Pool from our fusion. Multi-stage Fusion shows a +0.81% mAP improvement compared to only using a late fusion. The improvement of considering multiple abstraction-levels is marginal, at +0.15% mAP, suggesting that features at the same abstraction level can provide the most complementary information.

Fusion dimensions: We experiment on the significance of different dimensions in the fusion features. Either having only channel dimension (C) similar to [132], channel-spatial (CHW) dimensions or all channel-temporal-spatial (CHWT) similar to [41] is considered here. The results of this experiment is reported in Ta-

Fusion location	mAP (%)	GFLOPs
late only	21.84	77.15
late+intermid one-to-one	22.50	81.80
late+intermid multi-stage	22.65	94.80

(a) **Fusion location**: Using the fusion connections only before the logits, in-between each `res` block w/ or w/o considering multiple abstraction-levels at each fusion location. (Fine-Fine)

Fusion dimensions	mAP (%)	GFLOPs
C	18.11	76.45
CHW	19.86	93.12
CTHW	22.65	94.80

(b) **Fusion dimensions**: The Dimensions of Multi-stage Fusion features. When temporal dimension (T) is available, we use Coarse-centric Gaussians for location calibration. (Fine-Fine)

Fusion location	Fusion mask	mAP (%)	GFLOPs
late	none	20.59	75.98
	self-attention	21.84	77.15
multi-stage	none	21.42	92.69
	self-attention	22.65	94.80

(c) **Fusion mask**: The effect of using a self-attention mask to filter the fine-grained context (refer Fig. 3.4), with different fusion connections. (Fine-Fine)

Grid Pool input	α	mAP (%)	GFLOPs
T=128, stride=10	1/8	11.88	10.43
	1/4	18.12	16.53
T=256, stride=5	1/8	15.56	20.63
	1/4	18.16	32.82

(d) **Grid Pool configuration**: Variations of the sampling rate α with different temporal sizes of the input at the Grid Pool layer. (Coarse-only)

Pooling type	mAP (%)	GFLOPs
Max	16.21	15.42
Average	16.64	15.42
Striding	17.49	15.42
Grid Pool	18.12	16.53

(e) **Pooling type**: Different types of temporal pooling operations used after $\texttt{res}_2$ block. A temporal stride of 4 (equivalent to $\alpha = 1/4$) used here. (Coarse-only)

Two-stream Network	mAP (%)	GFLOPs
$\text{SlowFast}_{\text{det}}$	20.61	54.31
$\text{SlowFast}_{\text{det}}$ (w/ Grid Pool)	20.82	55.42
$\text{SlowFast}_{\text{det}}$ (w/ Fusion)	22.79	72.16
Coarse-Fine (w/ Grid Pool w/ Fusion)	23.24	73.37

(f) **Importance of Grid Pool and Multi-stage Fusion combined**: $\text{SlowFast}_{\text{det}}$ is a baseline w/o Grid Pool and Multi-stage Fusion. It samples input at different frame-rates (Fast is ×4 faster) and uses fusion connections similar to that of SlowFast [41]. (Coarse-Fine)

Table 3.3: **Ablations on Charades** [174] **localization** comparing the design choices of Grid Pool and Multi-stage Fusion. We show the performance in mean Average Precision (mAP), and measure the compute requirement for a temporal clip of $T = 128$ at inference in GFLOPs (floating-point operations $\times 10^9$). In these tables, we report the performance for fine-grained predictions, making decisions for every frame. The network configuration used in each experiment is shown within the braces of each caption (Fine-Fine, Coarse only or Coarse-Fine). Fine-Fine Networks are used in Fusion experiments to decouple the effect of Grid Pool, and similarly, Coarse only Networks are used in Grid Pool experiments decouple the effect of Multi-stage Fusion.

ble 3.3b. Note that the dimensions which are not available in any of the above scenarios are average pooled before the fusion. We see that having all CHWT dimensions in the fusion feature has a large improvement compared to the baseline, specifically +4.54% mAP. Introduction of the temporal dimension (T) shows the most improvement, which is +2.79% mAP. This is in fact mainly due to the temporal Gaussians in our Fusion that calibrate the features based on the location, without which, we can not see such improvement (i.e., +0.61% mAP over a single stream, when naively selecting corresponding temporal locations in the two streams for fusion w/o Gaussians).

Fusion mask: Here, we evaluate how important it is to filter the Fine features at the input of fusion, results of which, is shown in Table 3.3c. In the Multi-stage Fusion setting, having a self-attention mask to adaptively weight each Feature point gives an improvement of +1.23% mAP compared to feeding the Fine feature directly.

Pooling type: Next, we explore the performance gain caused by the proposed (temporal) Grid Pool layer. We compare against conventional temporal pooling operations such as max pooling, average pooling and even simple temporal striding. Here, we report the numbers for a Coarse-only network to decouple the effect of Grid Pool from that of Multi-stage Fusion. In these experiments, we set $\alpha = 1/4$, which essentially means a $\times 4$ temporal downsampling, and perform the downsampling after the $\texttt{res}_2$ block. Max pooling, average pooling use a similar setting of kernel size of 4 and a stride of 4. Grid pooling gives a consistent improvement over other methods, specifically +1.91% mAP and +1.48% mAP over commonly used max pooling and average pooling, respectively. We also note that a simple temporal striding can outperform max pooling and average pooling by +1.28% mAP and +0.85% mAP, respectively. We suspect that the inferior performance of max/average pooling is due to the aggressive temporal striding at the input of X3D, which is a stride of 10 by default (i.e., after the pooling, the stride is 40). In such a large window, pooling tends to blur most of the temporal information.

Grid Pool configuration: We try different configurations of Grid Pooling to evaluate its performance and compute requirement. Similar to above, we use the Coarse-only network. We consider an input of temporal length $T = 128$ at a stride of 10 or $T = 256$ at a stride of 5, to cover entire duration of Charades videos [174]. We try temporally downsampling each of these with $\alpha = 1/4$ ($\times 4$ downsampling) or $\alpha = 1/8$ ($\times 8$ downsampling). The performance of these configurations is given in Table 3.3d. $\times 8$ downsampling shows a significantly lower performance indicating

that it looses too much information with such an aggressive stride, i.e., more frames are important and need to be sampled by the Grid Pool layer. Moreover, increasing the number of input frames does not necessarily improve the performance (only +0.02% mAP) with $\alpha = 1/4$. Among these configurations, $T = 128$ with $\alpha = 1/4$ shows the best performance/compute trade-off.

Grid Pool and Multi-stage Fusion combined: Finally, we evaluate the combined performance of Grid Pool and Multi-Stage Fusion. To do this, we consider a two-stream baseline without these components, which we call $\text{SlowFast}_{\text{det}}$. This performs ×4 temporal downsampling in the Coarse stream based on striding, and use direct frame correspondences between Fine and Coarse streams for fusion, similar to SlowFast [41] while still using X3D modules like ours. The results of this study is given in Table 3.3f. When compared with this baseline, introduction of either Grid Pool or Multi-stage Fusion provides consistent improvements of +0.21% mAP and +2.18% mAP respectively. Our Coarse-Fine Networks outperform this baseline by a margin of +2.63% mAP.

Trade-off with X3D: Coarse-Fine network is designed to use a similar amount of computation as a two-stream version of X3D-M. Another way to use the extra compute is by increasing the number of layers. To understand if the increased compute is meaningful, we test X3D-L, a larger version of X3D (Table 3.2). X3D-L shows of 20.03% mAP with a compute of 147.04 GFLOPS. Coarse-Fine Networks outperform this in both accuracy and speed with 25.10% mAP at 73.37 GFLOPS.

3.3.3 MultiTHUMOS

Dataset: MultiTHUMOS [229] is an extension of the THUMOS [74] dataset with the untrimmed videos densely annotated for 65 different action classes. It provides frame-level action annotations for 30 hours of video across 413 videos, split as 200 for training and 213 for validation. On average, it contains 1.5 labels per frame and 10.5 action classes per video. It contains significantly smaller number of videos compared to Charades [174] and each video has a larger temporal length, which make the training difficult. We created a segmented version of this data, where each clip is limited to a maximum of 1280 frames, which gives a similar evaluation setting to Charades. For the computational efficiency, we use the temporal striding of 10.

Training: We follow a training process similar to what we did for Charades. We follow two stages of training with our Coarse and Fine streams pretrained on Kinetics-400 [79], i.e., separately and combined. We use the same hyperparameter settings and training schedules as in Charades (refer subsection 3.3.1). We use a

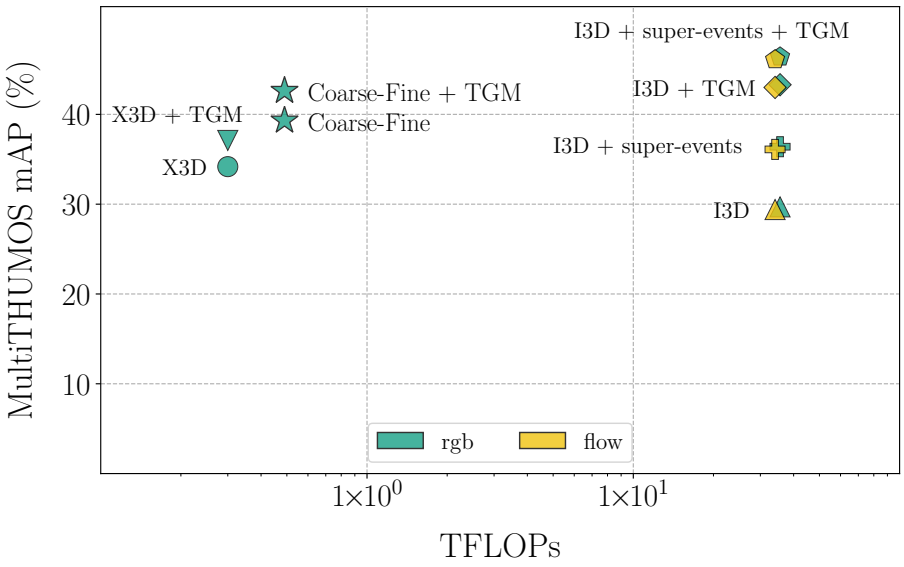

Figure 3.5: **Performance/complexity trade-off** of state-of-the-art methods for activity detection on MultiTHUMOS [229]. Our Coarse-Fine Networks /w TGM show a comparable performance with the state-of-the-arts with ∼75x speed, without using additional input modalities.

dropout rate of 0.5 before the logits layer. The logits go through a sigmoid to make multi-label predictions for each frame. We use an average of classification and localization loss for training.

Inference: At inference, we make predictions for every frame. Even though our input is sampled at a stride of 10, we consider the labels for all frames (at a stride of 1) and interpolate the logits to fit the original temporal length. Each input is center-cropped to 224×224. We report the performance (mAP), compute requirement to process an input clip of 1024×10 frames as TFLOPs ($\times 10^9$) and the number of learnable parameters(M). The length of 1024×10 frames is only considered as a reference for reporting the complexity values, and there are longer clips in the dataset with up to ×5 frames. We process these frames fully convolutionally.

Results: We show the performance (mAP) of Coarse-Fine Networks on Multi-THUMOS [229] activity detection with the corresponding compute requirement (TFLOPs, i.e., $\times 10^{12}$) in Fig. 3.5. We observe an improvement of +4.63% from X3D [40] to Coarse-Fine. While our Coarse-Fine network is almost 75 times faster than the previous model (0.49 TFLOPS (Coarse-Fine) vs. 35.57 TFLOPS (I3D + TGM)), it still achieves comparable performance to the previous state-of-the-art. Models using the X3D backbone including ours lose motion details due to the aggressive

1/10 striding compared to I3D [20] that doesn't do striding, making them less effective when combined with other temporal modeling methods (e.g., TGM [135]). Still, our Corase-Fine Networks were able to overcome such limitation and perform comparably. Coarse-Fine /w TGM shows a further improvement of +2.21%mAP.

3.4 Conclusion

We presented Coarse-Fine Networks, which is a two-stream architecture to combine temporally Coarse representations with Fine representations. We introduced the approach of temporal Grid Pooling that learns to differentiably select informative frames while discarding the other, to obtain Coarse representations. We also introduced the Multi-stage Fusion to best combine the Coarse stream with the Fine stream. We confirmed that the Coarse-Fine networks obtain the best known performance on Charades localization, while spending much less computation time.

3.5 Appendix

3.5.1 Pretraining details

We initialize our models from scratch and pretrain for classification in Kinetics-400 [79] prior to finetuning for activity detection. Kinetics-400 is commonly used as a pretraining dataset for video models. It contained 240k training and 20k validation videos at release, but due to availability, our version contains ~220k training and ~17k validation videos. Each video contains a single action out of 400 human action categories.

Similar to [40], we do not pretrain on ImageNet [35] prior to Kinetics-400 pretraining. We follow the recently-proposed multigrid training [201] recipe for efficient training of our baseline. We train for 200k iterations with a base batch size of 128 on 4 Titan RTX GPUs. The learning rate is initialized at 0.2 and decreased by a factor of 10 at 80k, 130k and 170k iterations. We fix the batch size considered for batch normalization to be 8 in-spite of the varying batch size in multigrid training, following an observation in [51]. Each clip is sampled at a stride of 10 frames at the base multigrid configuration. During training, each input is randomly sampled in [256,320] pixels, spatially cropped to 224×224 and applied a random horizontal flip similar to [41, 175, 192]. We use a dropout rate of 0.5 before the last fully-connected layer.

Losses	mAP (%)
Grid Pool	18.12
Grid Pool w/ $\mathcal{L}_{\text{rec}}$	15.28
Grid Pool w/ $\mathcal{L}_{\text{flow}}$	15.89

Table 3.4: **Different losses used to learn the grid locations:** In the default setting we use classification and localization supervision to learn the grid locations. We further explore auxiliary losses through self-supervision: an L1 reconstruction loss or a flow-based loss. However, these auxiliary losses do not improve the performance (Coarse-only).

In our pretraining setting, the baseline model achieves 62.62% Top-1 accuracy (with 3-view testing) on Kinetics-400. This is with a partial training dataset and a ×4 shorter schedule compared to [40]. We also tried directly using X3D checkpoint (from Facebook) pretrained in the same setting, since both our Coarse and Fine network are composed of the X3D modules with the same number of modules/layers. The difference in this case is that the checkpoint is trained with 240k Kinetics-400 training videos and a longer schedule (without multigrid training [201]), which gives 71.48% Top-1 accuracy on Kinetics-400.

3.5.2 Self-supervision for Grid Pool

We further explored multiple self-supervised losses to constrain the learned grid locations similar to [44]. Since the input itself can provide ques (i.e., motion or static) on important frame locations, such self-supervision can potentially benefit learning grid locations. In other words, learning grid locations does not necessarily need to rely on supervised losses such as classification or localization.

A flow-based loss was used to minimize the flow within a grid cell, while maximizing the flow in-between cells, forcing the cell boundaries to lie in the regions of maximum flow change. A reconstruction loss was used to minimize the L1 distance between the input and an upsampled output of the Grid Pool layer to force a faithful downsampling operation. Our overall loss function can be given as,

$$\mathcal{L} = 0.5\,\mathcal{L}_{\text{class}} + 0.5\,\mathcal{L}_{\text{loc}} + 0.1\,\mathcal{L}_{\text{aux}}\,,$$

where $\mathcal{L}_{\text{class}}$ and $\mathcal{L}_{\text{loc}}$ represent classification and localization losses respectively. $\mathcal{L}_{\text{aux}}$ is either the reconstruction loss $\mathcal{L}_{\text{rec}}$ or flow-based loss $\mathcal{L}_{\text{flow}}$. The results of this experiment is shown in Table 3.4. None of these auxiliary losses improved the

performance compared to using only the classification/localization supervision. The reconstruction loss may not be ideal as it can force the sampled frame in each grid closer to an average of all frames within the grid. The flow-based loss is more intuitive in comparison, as the motion information can prevent the sampling of redundant frames. However, since we perform aggressive striding at the input of X3D (usually a stride of 10), optical-flow computation can be noisy. Moreover, the online flow computation method that we use (Representation Flow [134]) can be compute heavy to accommodate more iterations to converge, which adds to the noise in the computed flow. These limitations may have affected the performance in the self-supervised setting.

Chapter 4

Pretraining for Activity Detection without Fine-grained Labels

4.1 Overview

Pretraining has become an indispensable component in the deep learning pipeline. Most computer vision tasks leverage large-scale labeled or unlabeled data to do pretraining in a supervised or unsupervised way, which gives performance boosts in downstream tasks, especially when training data is scarce. Such benefits of pretraining have been observed in many applications including object detection [34, 113], segmentation [138], video understanding [46], reinforcement learning [168] and language modeling [104]. This behavior can be attributed to models becoming more robust by looking at more data, which helps generalize to unseen distributions in the downstream tasks.

Even though pretraining generally helps downstream tasks, the amount of boost depends on the compatibility of the pretrained task and the downstream task [1]. The pretraining task (or distribution) should be as close as possible to the downstream task (or distribution) to achieve the highest possible gain. However, in a traditional pretraining pipeline, such compatibility may not always be an option. We only have a few large-scale labeled datasets limited to general tasks such as classification. Hence, models for most downstream tasks are usually pretrained in a classification task on either ImageNet-1K [35] (image domain) or Kinetics-400 [20] (video domain), which often leaves a disparity between pretraining and downstream tasks.

For instance, in temporal activity detection— which is defined as predicting (one or more) activity classes per frame— we have the same observation: although

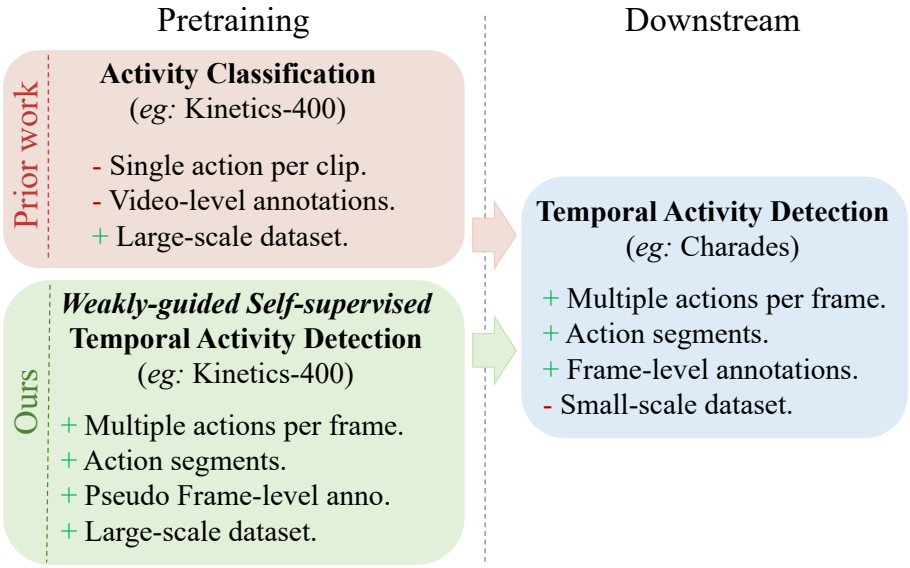

Figure 4.1: **Our *weakly-guided self-supervised* pretraining strategy:** Previous work on temporal activity detection are usually pretrained on large-scale activity classification datasets (*e.g.*, Kinetics-400 [20]). However, there is a disparity between pretraining and downstream tasks, which hurts the detection performance. To bridge this gap, we propose a new self-supervised pretext task (detection) which leverages already-available weak labels (classification) to introduce frame-level pseudo labels, multi-action frames and action segments, similar to downstream. In fact, we design a detection pretraining task on large-scale classification data, without extra annotations.

pretraining on activity classification improves downstream detection performance, it is limited by the disparity between tasks. As a model can learn to aggregate temporal information when pretraining for activity classification (looking at the bigger picture), it may not be well-suited to do downstream activity detection, which is fine-grained and requires the model to retain temporal information as much as possible (looking at the composition of atomic actions). To address this issue, multiple previous work have proposed specific temporal [76, 132, 135] or graphical [47, 115] modeling in the downstream to capture aspects not seen in the pretraining data, such as long-term motion, human-object interactions, or multiple overlapping actions in fine detail. However, it can be difficult for such finetuning techniques to alleviate the data disparity effectively.

In this work, we propose a *weakly-guided self-supervised* pretraining method

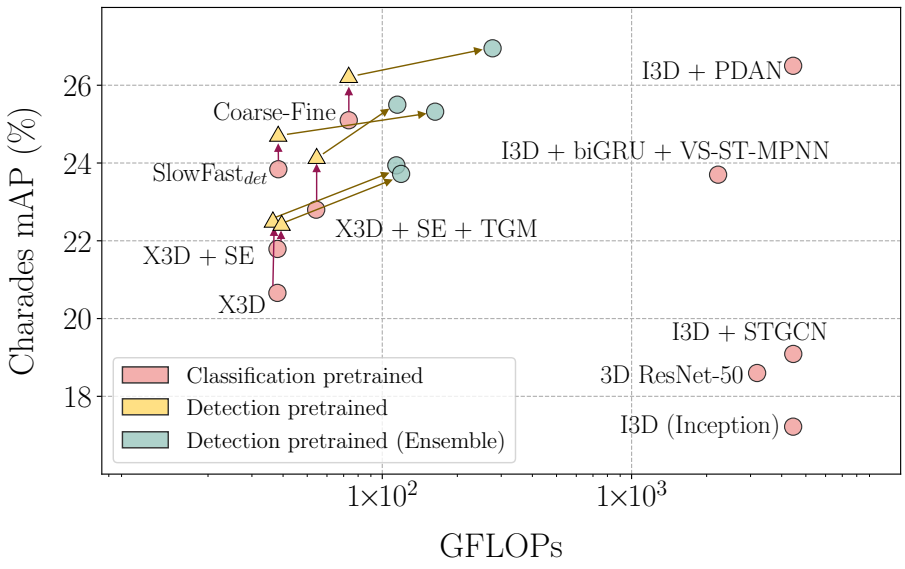

Figure 4.2: **Performance comparison** between models pretrained for classification and the proposed *weakly-guided self-supervised* detection, on downstream Charades [174] activity detection setting. Representative models pretrained for detection, using *Volume Freeze*, *Volume MixUp* and *Volume CutMix* achieve significant performance boosts over their classification pretrained counterparts. Relative improvement is shown as Classification-pretrained → Detection–pretrained → Detection-pretrained (Ensemble). Model names are shown for Classification pretrained versions in space (red circles).

for activity detection, using large-scale classification data with **no extra annotations**. We augment pretraining data to capture fine-grained details and use detection as the pretraining (or pretext) task — a step closer to bridging the gap with downstream detection (see Fig. 4.1). Specifically, we first extend weak video-level labels of classification clips to create pseudo frame-level labels. Then, we propose three self-supervised augmentation techniques to generate multi-action frames and action segments within a clip. Namely, we introduce *Volume Freeze*, *Volume MixUp* and *Volume CutMix*. Volume Freeze creates a motion-less segment within a clip introducing segmented actions, whereas Volume MixUp and Volume CutMix seamlessly merge multiple clip segments into one, which tries to mimic the downstream data distribution of multiple actions per frame. Based on the augmented data, models are pretrained on an activity detection task. Our evaluations validate the benefits of the proposed pretraining strategy on multiple temporal

activity detection benchmarks such as Charades [174] (see Fig. 4.2) and MultiTHUMOS [229], with multiple models such as X3D, SlowFast and Coarse-Fine. We further investigate the extent of the detection-pretrained features in our ablations and, recommend when and how to use them best.

Our method leverages weak labels during pretraining, having downstream settings unchanged. Also, we design a pretext task based on augmentations similar to the work in self-supervision. Considering the traits of both domains, we term our work as *weakly-guided self-supervision*.

4.2 Weakly-Guided Self-Supervised Detection Pretraining

We introduce a self-supervised pretraining task for activity detection, which leverages already-available weak labels in large-scale classification datasets. This idea is primarily motivated based on removing the disparity between classification pretraining and downstream detection. Almost all the temporal activity detection works are pretrained for classification on large-scale datasets such as Kinetics-400 [20]. This is because (1) video models need large-scale data to mitigate overfitting during training, and (2) detection annotations (frame-level) are too expensive to collect for a large enough dataset. Even with such classification-based pretraining at scale, the performance on downstream detection task is unsatisfactory. One reason for this is the complexity of the downstream task: predicting fine-grained activity classes per frame is challenging. Also, it can be partially attributed to the striking difference in tasks (and data distributions) during pretraining and downstream detection. As shown in Fig. 4.1, pretraining videos in general (eg: Kinetics-400) have only a single action per clip with video-level annotations, whereas, in a downstream detection task (eg: Charades), usually a model needs to predict multiple actions per each frame. It means that although such classification-based pretraining leveraged large-scale labeled data for training, the inherent bias which comes with it acts as a limiting factor for the downstream performance.

We try to bridge this gap by proposing a *weakly-guided self-supervised* pretraining task that closely resembles the downstream task. It shows similarities to both weak- (as we leverage weak labels) and self-supervision (as we design a pretext task based on augmentations). Specifically, we introduce frame-level pseudo labels followed by multi-action frames and action segments through a set of data augmentation strategies. By doing so, we benefit from the scale of data, while having a similar data distribution (in terms of having overlapping and segmented actions)

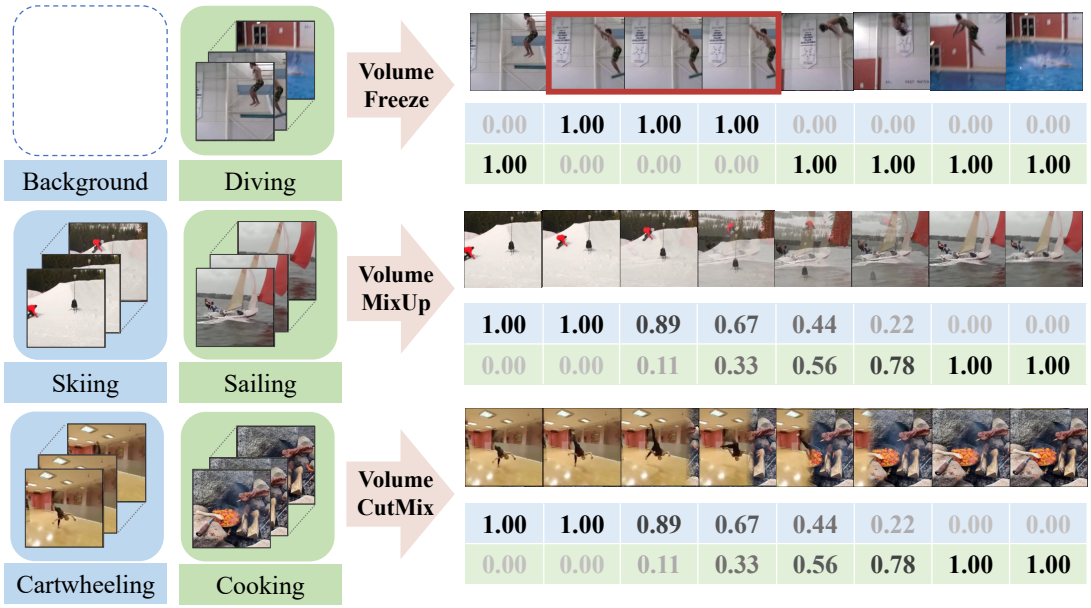

Figure 4.3: **Volume Augmentations for our *weakly-guided self-supervised* detection pretraining:** *Volume Freeze*, *Volume MixUp* and *Volume CutMix*. We first extend video-level labels (of single-action videos from Kinetics-400 [20]) into every frame, creating frame-level pseudo labels. Next, to introduce action segments and multi-action frames similar to downstream detection, we propose the above three augmentation strategies. Volume Freeze stops the motion of a video segment, creating a background segment (assuming no action can be performed without motion). Hard-labels are assigned for action and background accordingly. Volume MixUp and CutMix introduce a seamless spatio-temporal (random) transition between two clips inspired by similar ideas in image domain [234, 241]. Here, labels are weighted to create soft-labels based on the alpha values or the area of each frame, respectively. Augmented frames are best viewed zoomed-in.

as downstream detection. Next, we will introduce our pseudo labeling, volume augmentations, and how we combine these ideas.

4.2.1 Frame-level Pseudo Labels

Downstream detection is about fine-grained predictions of activity classes, which requires frame-level annotations to train. However, large-scale classification datasets used for pretraining contain video-level annotations. For instance, we consider commonly-used Kinetics-400 [20], which contains a *single* action per clip with

a video-level label. As we wish to design a pretraining task that closely-resembles downstream detection, we generate frame-level labels from the available video-level labels, by replicating the same label for every frame. Such labels can be noisy because not every frame in a clip may contain the annotated single video-level action. However, we know such clips do not contain any additional actions, at least in the context of the original action categories. It is worth noting that we do not create new labels, thus no extra annotation effort is spent generating frame-level pseudo labels for classification data.

One may also consider a pretraining dataset such as ActivityNet [18] with multiple actions per clip, instead of Kinetics-400 [20] with a single action. In such a setting, an off-the-shelf action proposal generator can be used to get such pseudo frame-level labels for the proposed pretraining. However, in this paper, we consider Kinetics pretraining as commonly-used in most prior work.

4.2.2 Volume Augmentations

Based on the frame-level pseudo labels, we design a self-supervised pretext task for detection on the pretraining data. The idea here is to introduce action segments and multi-action frames similar to the downstream data. To do this, we propose three augmentation methods specifically for video data (i.e., spatio-temporal volume): (1) Volume Freeze, (2) Volume MixUp and, (3) Volume CutMix. Next, we will explain these concepts in detail.

Volume Freeze

Since downstream data contains multiple action segments per clip, we want to introduce the notion of action segments in pretraining data as well. However, the videos in the pretraining dataset (Kinetics-400) contain only a single action per clip, in which, it is a challenge to have such segments. Our solution here is to create an motion-less (background) segment within a clip. We do this by randomly selecting a frame in a given clip, and replicating it for a random time interval (or number of frames). We call this 'Background'. Such background segments are appended to the original clip at the corresponding frame location, maintaining the temporal consistency as much as possible. We label the frozen segment with a *new* background label (zero-label) assuming it does not depict the original action, without any motion. Although this is a strong assumption (i.e., some actions can be classified based on appearance only, without motion), it allows the model to differentiate motion variations, giving a notion of different action segments. Volume Freeze augmentation is shown in Fig. 4.3 (top) and elaborated Fig. 4.4. It

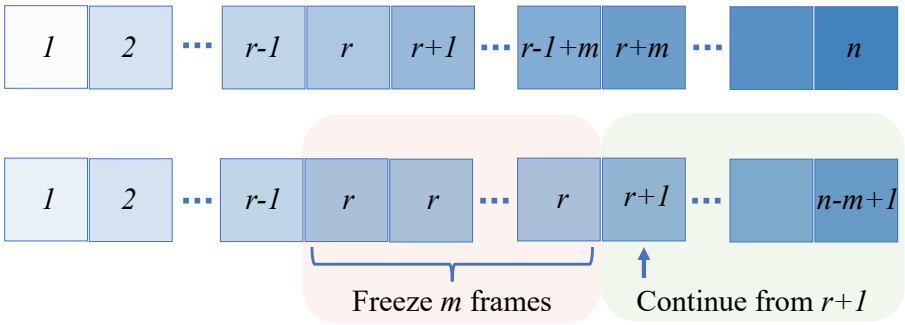

Figure 4.4: **Volume Freeze:** Given an input clip of length n, a randomly selected frame r is replicated for a random m duration and appended in place. Overflowing frames from the end of the clip ($t > n$) are discarded. Labels are hard labels: either action or background. Frame number is shown here with each frame.

can be denoted as follows,

$$\text{VF}(v) = \text{concat}(v[1:r-1], \{v[r]\}^m, v[r+1:n-m+1]),$$
$$\text{VF}(l) = \text{concat}(l[1:r-1], \{0\}^m, l[r+1:n-m+1]),$$

where $\text{VF}(v)$ and $\text{VF}(l)$ denote the augmented video and associated label in Volume Freeze. Also, v and l correspond to a given video clip of length n and its frame-level pseudo label (one-hot), respectively. We freeze a frame for random m times (denoted by $\{\cdot\}^m$) at a random temporal location $r \in [1, n-1]$, where $m \in [2, n-r+1]$, and we concatenate it to the original clip to create an augmented clip of the same original length n, discarding overflowing frames. This guarantees that our model does not benefit from seeing more frames compared to baseline. Also, the information loss from discarding frames is not significant, as our clip-sampling already has a significant randomness. The labels for the augmented clip are created accordingly, where we have zero labels for the frozen segment, and original frame-level labels elsewhere. We further experiment with freezing multiple segments within a clip, which has a limited gain.

Volume MixUp

With Volume MixUp, we introduce multi-action frames to pretraining clips, which originally have a single action per clip. More specifically, we combine randomly selected two clips with a random temporal overlap, so that the overlapping region contains two actions per frame. This is inspired by the MixUp operation in image domain [241]. However, here we focus more on preserving the temporal

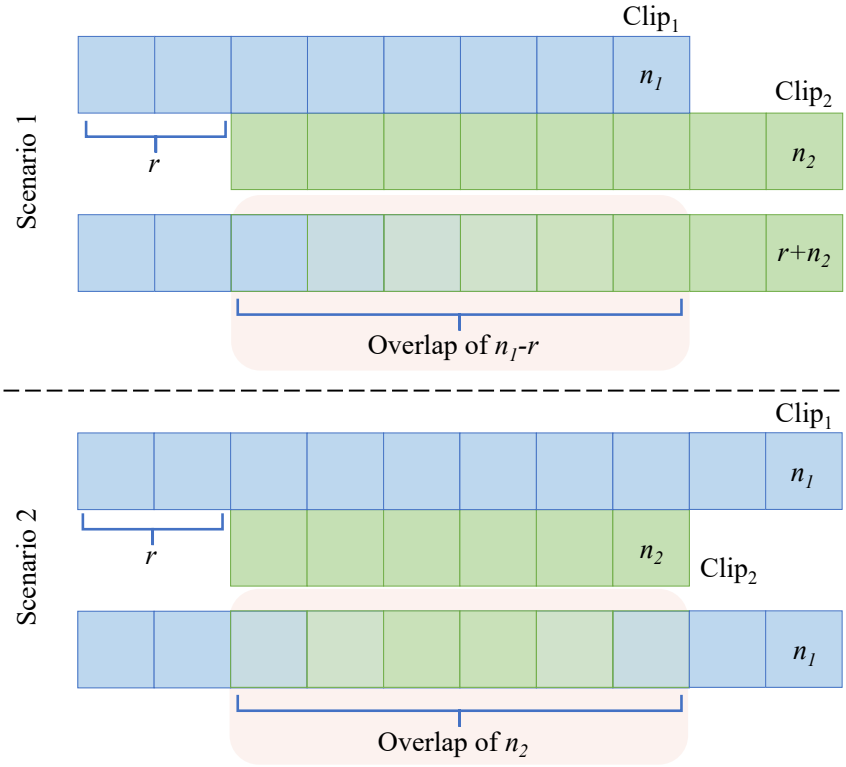

Figure 4.5: **Volume MixUp:** Given two input clips of length n_1, n_2, one clip is randomly shifted by r to create a random overlap. When mixing, a seamlessly-varying alpha mask is applied in the overlapping region so that we have smooth transitions between clips. Soft-labels are created based on the alpha values. There can be two cases based on clip lengths n_1, n_2 and the random shift r: scenario 1 (top-left): $\text{Clip}_1 \rightarrow \text{Clip}_2$, or, scenario 2 (top-right): $\text{Clip}_1 \rightarrow \text{Clip}_2 \rightarrow \text{Clip}_1$. Clip length is shown here at the end of each clip. Alpha mask is also used to weight clip labels accordingly.

consistency in Volume MixUp when combining two clips, by having seamlessly varying temporal alpha masks for each clip. It means, we have a smooth transition from one clip to the other within the temporal overlap. The labels for each clip are weighted with the corresponding temporal alpha mask to create soft labels. Such an augmented example with Volume MixUp is given in Fig. 4.3 (middle) and elaborated in Fig. 4.5. This can also be denoted as,

$$\text{VM}(v_1, v_2)[t] = \alpha[t] \cdot v_1[t] + (1 - \alpha[t]) \cdot v_2[t - r],$$
$$\text{VM}(l_1, l_2)[t] = \alpha[t] \cdot l_1[t] + (1 - \alpha[t]) \cdot l_2[t - r],$$

for two video clips v_1 and v_2 of length n_1 and n_2 respectively. $v_i[t]$ and $l_i[t]$ denote the t-th video frame and its corresponding one-hot labels, and $\alpha[t]$ represents the scalar alpha values at time t for mixing frames. Both clips are temporally padded to accommodate corresponding lengths n_1, n_2 and random shift r. The seamless temporal alpha mask for the overlapping region is defined as,

$$\alpha[t] = \begin{cases} \mathsf{T}_{[0,1]}(\frac{n_1 - t}{n_1 - r}) & \text{if } n_2 + r \geq n_1, \\ \mathsf{T}_{[0,1]}(\frac{|n_2 + 2r - 2t|}{n_2}) & \text{otherwise,} \end{cases}$$

The *truncation* operator $\mathsf{T}_{[0,1]}(\cdot)$ clips the mask values within the range of $[0, 1]$. It is defined in detail in appendix. This makes $\alpha[t]$ to be a piecewise linear function w.r.t. t. In scenario 1 ($n_2 + r \geq n_1$), the augmented clip transit as $\text{Clip}_1 \rightarrow \text{Clip}_2$, whereas in scenario 2, it works as $\text{Clip}_1 \rightarrow \text{Clip}_2 \rightarrow \text{Clip}_1$. It depends on the clip lengths n_1, n_2 and the random shift r. More details are in the Appendix. The two-clips are selected randomly (without any constraints), and hence the resulting mixed-up clip may contain artifacts. However, such randomness helps to generalize better, as also seen in [241].

Volume CutMix

Similar to Volume MixUp, we introduce multi-action frames with Volume CutMix. Here, given two clips, we define an overlapping region and assign a seamlessly changing spatial window for each clip within this region. This is inspired by CutMix [234] operation in image domain. In Volume CutMix however, we focus on a seamless transition between clips in time. We introduce two strategies for Volume CutMix: (1) Transient Window and (2) Transient View (Constant Window). See Fig. 4.3 (bottom) and Fig. 4.6.

Transient Window : This is closely-related to our Volume MixUp. Given two clips, we insert a random relative shift r to create a random overlapping region. Clips are temporally padded at the ends to accommodate different clip lengths and shift. This can have the same two scenarios as before, depending on n_1, n_2 and r. However, rather than defining a scalar alpha mask per frame, now we define a 2D spatial window $\mathbf{M}$ as a mask, which changes seamlessly in time, within the overlapping region. The soft-labels for the overlapping region are weighted based on the area of each window. For convenience, we define the two windows based on a moving vertical plane as shown in Fig. 4.6. In between two windows, we have a short but smooth spatial transition, instead of a hard spatial boundary. This

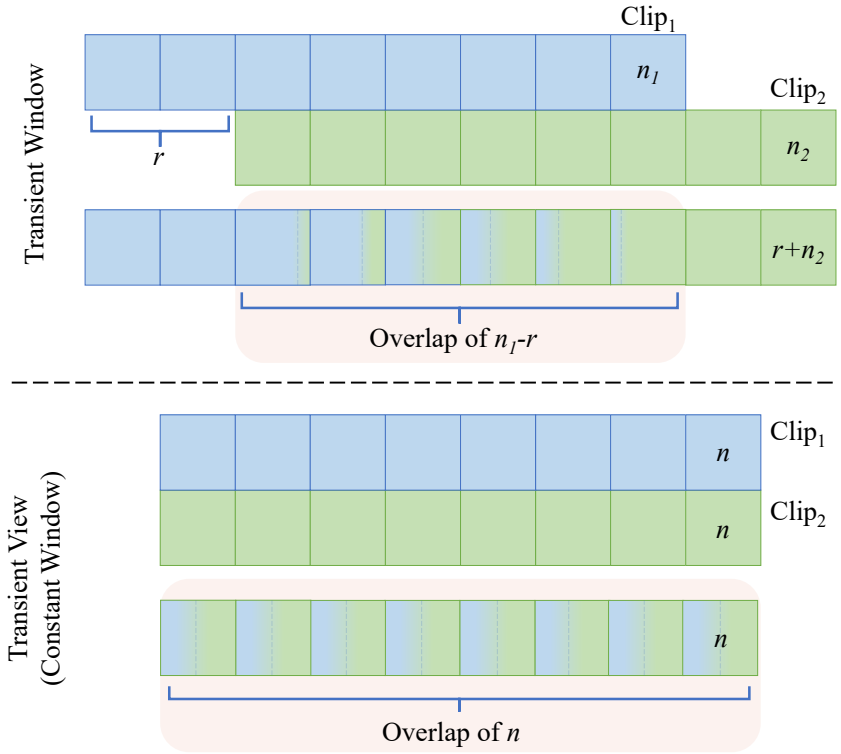

Figure 4.6: **Volume CutMix:** We have two settings: (1) Transient Window (top-left) and, (2) Transient View (top-right). In Transient Window, random relative shift r is given similar to Volume MixUp. Smooth transition between clips is achieved when the transient window is moving from left to right (this setting can have the same two scenarios as in Volume MixUp). In Transient View, we have constant windows (half-sized) looking at transient views of the content inside (i.e., the content of each frame is moved inside the corresponding window with time, in addition to the natural motion of the clip).

operation can be denoted as,

$$\begin{aligned} \text{VC}(v_1, v_2)[t] &= \mathbf{M}[t] \odot v_1[t] + (1 - \mathbf{M}[t]) \odot v_2[t - r], \\ \text{VC}(l_1, l_2)[t] &= |\mathbf{M}[t]| \cdot l_1[t] + (1 - |\mathbf{M}[t]|) \cdot l_2[t - r]\,, \end{aligned}$$

where $\mathbf{M}[t]$ (defined below) is the spatial mask at time t. v_i and l_i represent a clip and the corresponding one-hot label . The symbols $\odot$ and $|\cdot|$ mean Hadamard (element-wise) product and *area* of the mask (defined as the average of all its elements), respectively. More details are in the Appendix.

Transient View: In this setting, we keep the window size constant for each clip (half of the frame) within the overlapping region (not random, but n in this case). For each window to cover the spatial range of each clip, we move each clip within the constant window from left-to-right, in time. This artificial movement is introduced in addition to the natural motion in each clip. We have a constant clip length and no random shift in this case, since a zero-padding in only one-half of a frame may cause problems for convolution kernels. With the same notations as before, the augmented clip and labels can be denoted as,

$$\begin{aligned} \mathrm{VC}(v_1, v_2)[t] &= \mathbf{M} \odot v_1[t] + (1 - \mathbf{M}) \odot v_2[t], \\ \mathrm{VC}(l_1, l_2)[t] &= 0.5 \cdot l_1[t] + 0.5 \cdot l_2[t]. \end{aligned}$$

The spatial mask $\mathbf{M}$ defines a vertical plane to split each frame within the overlapping region into two windows. The location of this vertical plane (w_t) can either depend on $\alpha[t]$ (in Transient Window) or be constant (in Transient View).

4.2.3 Combining Augmentations

In the previous subsections, we defined the components of our pretraining scheme: namely, frame-level pseudo labeling and volume augmentations. When combining augmentations, we use either (1) joint training or (2) model ensembling.

Joint training: Here, we combine the three augmentations during training. A simpler setting is to apply only a single randomly-selected augmentation per clip (referred to as Joint train - single). Or else, we can apply up to all 3 augmentations per clip with a random probability (referred to as Joint train). Although the latter strategy seems flexible, applying multiple of the proposed augmentations on a given sample can create confusing inputs, which are hard to train with.

Model ensembling: Here, we apply only a single selected augmentation among the proposed Volume Freezing, MixUp, and CutMix during training. At inference, we combine predictions coming from such separate models trained with each augmentation. By doing so, we can combine the benefits of each augmentation, without worrying about the input confusion at training. However, this incurs more compute requirement at inference, compared to a jointly trained single model. For fair comparison, we always report the numbers for joint training (i.e., same compute budget) alongside ensembles.

4.3 Experiments

To validate the benefits of our proposed method, we pretrain on commonly-used Kinetics-400 [20] and evaluate on rather-complex Charades [174] and MultiTHUMOS [229] for downstream detection, using the efficient video backbone X3D [40]. In addition to applying the proposed augmentations at the input level, we also run a few experiments with manifold augmentations [189], where each augmentation method is applied to the feature maps at a random depth of the network.

4.3.1 Kinetics-400 Detection Pretraining

By default, we initialize with our backbone X3D-M (medium) with checkpoints provided in original work [40], as in common-practice for activity detection. This allows shorter pretraining schedules and better convergence for both our method and baseline. We pretrain X3D for 100k iterations with a batch size of 64 and an initial learning rate of 0.05 which is reduced by a factor of 10 after 80k iterations. We use a dropout rate of 0.5. From each clip, we sample 16 frames at a stride of 5, following the usual X3D training setup. During training, first, each input is randomly sampled in [256, 320] pixels, spatially cropped to 224×224, and applied a random horizontal flip. Next, we extend the labels to every frame as we described earlier, and apply one of the proposed volume augmentations to a batch of input clips.

It is important to note that both our method and baseline are always pretrained for the *exact* same number of iterations (i.e., gradient steps) and see a similar amount of data. Although, Volume MixUp and CutMix combines multiple clips per datapoint, each clip has a partial visibility, and each datapoint has the same number of total frames. This results in the same pretraining cost (see Appendix for details).

4.3.2 Charades Evaluation

We initialize X3D [40] with checkpoints from our detection pretraining. From each clip, we sample 16 frames at a stride of 10 and train for 100 epochs with a batch size of 16. Initially, we have a learning rate of 0.02, which is decreased by a factor of 10 at 80 epochs. For Coarse-Fine and SlowFast$_{\text{det}}$, we follow the same two-staged training strategy as in [76]. We train all methods on Charades with Binary Cross-Entropy (BCE) as localization and classification losses. Our models and baselines are always trained for same number of total iterations for fair comparison. At inference, we make predictions for 25 equally-sampled frames

per each input in the validation set, which is the standard Charades localization evaluation protocol [174] followed by all previous work. Also, it is important to note that the original evaluation script from the Charades challenge scales the Average Precision for each class with a corresponding class weight. However, in our ablations, we report the performance on predictions for every frame, which gives a more fine-grained evaluation without class-dependent weighting. Performance is measured using mean Average Precision (mAP).

Results: We report the performance of state-of-the-art methods comparing their pretraining strategy in Table 4.1. These numbers are for the Charades standard evaluation protocol [174]. We see a clear improvement from the model ensembles pretrained with the proposed detection task across multiple methods. The vanilla X3D [40] backbone without any additional modeling achieves the biggest relative improvement of +3.28% mAP. Detection pretraining also helps any lightweight temporal modeling on top of pre-extracted features as in super-events [132] with a +2.13% mAP and in TGM [135] with a +1.66% mAP improvement. Finally, we see the benefits in fully end-to-end trained multi-stream networks such as $\text{SlowFast}_{\text{det}}$ (+2.52% mAP) and Coarse-Fine Networks [76] (+1.85% mAP). We also show the performance of our joint-trained single models, for fair comparison under the same compute budget. Our models consistently outperforms baselines. It is important to note that even though our detection ensembles are compute-heavy compared to baselines, they are still an order-of-magnitude efficient compared to prior state-of-the-art PDAN [32]. Note that $\text{SlowFast}_{\text{det}}$ here is a variant of original SlowFast [41], with X3D [40] backbone, adopted for detection in [76]. We show the performance vs. compute trade-off graph in Fig. 4.2.

Ablations: In Table 4.2, we discuss the benefit of each augmentation, both separately and combined, followed by an interesting observation in multi-stream models. Each of our volume augmentation provide consistent gains, with +1.51% mAP in Volume Freeze, +1.90% mAP in Volume MixUp and +1.71% mAP in Volume CutMix. When combining augmentations, if we apply multiple of them to a given input, it may result in confusing frames. Rather, different augmentations can be complementary when used as ensembles, giving +3.22% mAP over the baseline (see Table 4.2a). In multi-stream models, we observe that our detection pretrained models do not show similar gains as baselines, (1) at different temporal resolutions or (2) in temporal aggregation (see Table 4.2b). When selecting models based on this observation, we see consistent improvement. A detailed discussion on this and more ablations are included in the Appendix

Model	Modality	Pretraining		mAP (%)
		cls.	det.	
Two-stream I3D [21]	R+F	✓		17.22
3D ResNet-50 ([57])	R	✓		18.60
STGCN ([47])	R+F	✓		19.09
VS-ST-MPNN [115]	R+O	✓		23.70
MS-TCT ([33])	R	✓		25.40
PDAN ([32])	R+F	✓		<u>26.50</u>
X3D ([40])	R	✓		20.66
			✓	(22.36) **23.94**
SE* [132]	R	✓		21.79
			✓	(22.24) **23.92**
TGM + SE* [135]	R	✓		23.84
			✓	(24.11) **25.50**
$\text{SlowFast}^*_{\text{det}}$ ([41])	R	✓		22.80
			✓	(24.73) **25.32**
Coarse-Fine [76]	R	✓		25.10
			✓	(26.19) **<u>26.95</u>**

Table 4.1: **Performance on Charades [174]:** We report the performance (mAP), input modalities used (R: RGB, F: optical flow or O: object), and the pretraining method: classification (cls.) or the proposed detection (det.). These results correspond to the original Charades localization evaluation setting (i.e., evaluated on evenly-sampled 25 frames from each validation clip). Model ensembles trained with our detection pretraining significantly outperform their counterparts, consistently. Coarse-Fine achieves a new state-of-the-art performance of 26.95% mAP even with RGB modality only, when pretrained with our proposed method. Improved results from our pretrained ensembles are in bold and joint-trained single-models are within (·). The best performance from each pretraining is underlined. Model variants with X3D backbone are denoted with *.

4.3.3 MultiTHUMOS Evaluation

We follow the same training recipe as in Charades, starting with a checkpoint pretrained for our detection. At inference, we make predictions per every frame

Pretraining method	Aug. / Combining Aug.	mAP (%)
Baseline (cls.)	-	17.28
Ours (det.) w/ single augmentation	Volume Freeze	(+1.51) 18.79
	Volume MixUp	(+1.90) 19.18
	Volume CutMix	(+1.71) 18.99
Ours (det.) w/ multiple augmentations	Joint train	(+1.83) 19.11
	Ensemble	(+3.22) 20.50

(a) **Single stream of X3D [40]:** Each of the volume augmentations provide consistent improvements over the classification pretrained baseline. However, when combining augmentations, ensembles work best compared to joint-training which can create confusing inputs with multiple augmentations.

Model	Coarse/ Slow	Fine/ Fast	Two-stream
Slow_{det} (cls.) - Fast_{det} (cls.)	17.49	17.28	20.31
Slow_{det} (VC) - Fast_{det} (VC)	18.60	18.99	(+0.91) 21.22
Coarse (cls.) - Fine (cls.)	18.13	17.28	23.29
Coarse (VC) - Fine (VC)	18.85	18.99	(-0.46) 22.83
Coarse (VC) - Fine (cls.)	18.85	17.28	(+0.28) 23.57
Coarse (Ensemble) - Fine (cls.)	-	-	24.29
Coarse (Ensemble) - Fine (cls. + Ensemble)	-	-	24.61

(b) **Multi-stream $\text{SlowFast}_{\text{det}}$ [41]/ Coarse-Fine [76]:** Here, we see an interesting observation. Even though detection pretrained models are consistently better as single-stream networks (eg: either Coarse/Slow or Fine/Fast), when combined as multi-stream networks, performance varies. We further investigate why this happens in Appendix. Model ensembles give consistent improvements as expected.

Table 4.2: **Ablations on Charades [174]** with our volume augmentations in single or multi-steam models. Each augmentation gives performance boosts, and best combined as ensembles. Detection pretrained models do not show gains as good as baselines at different temporal resolutions or in temporal aggregation. This is discussed in detail in Appendix. Here, We show the performance in mean Average Precision (mAP) for fine-grained predictions (i.e., making decisions per every frame rather than evenly-sampled 25 frames from each validation clip).

and report using mAP.

Model	Modality	Pretraining		mAP (%)
		cls.	det.	
Two-stream I3D [21]	R+F	✓		36.40
TGM + SE [135]	R+F	✓		46.40
PDAN ([32])	R+F	✓		<u>47.60</u>
X3D ([40])	R	✓		37.17
			✓	(38.92) **40.88**
TGM + SE* [135]	R	✓		39.16
			✓	(41.55) **43.15**
PDAN* ([32])	R	✓		39.20
			✓	(42.13) **<u>44.35</u>**

Table 4.3: **Performance on MultiTHUMOS [229]:** We report the performance (mAP), input modalities used (R: RGB or F: optical flow), and the pretraining method: classification (cls.) or the proposed detection (det.). Model ensembling trained with our detection pretraining significantly outperform their counterparts consistently, and shows overall competitive results even with RGB modality only. Improved results from our pretrained ensembles are in bold and joint-trained single-models are within (·). The best performance from each pretraining strategy is underlined. Model varients with X3D backbone are denoted with *.

Results: In Table 4.3, we show that the state-of-the-art models pretrained with the proposed detection, consistently outperform those trained with classification, both in vanilla backbones such as X3D [40] (+3.71% mAP), and in models which perform temporal modeling on-top of pre-extracted features as in TGM [135] (+3.99% mAP) or PDAN [32] (+5.15% mAP). PDAN, with our pretraining, significantly efficient X3D backbone and only RGB modality achieves competitive performance compared to multi-modal I3D [20] counterparts.

4.4 Conclusion

This work introduced a new weakly-guided self-supervised pretraining strategy for temporal activity detection, leveraging already-available weak labels. We defined a detection pretraining task with frame-level pseudo labels and three volume augmentation techniques, introducing multi-action frames and action segments

to the single-action classification data. Our experiments confirmed the benefits of the proposed method across multiple models and challenging benchmarks. As takeaways, we further provide recommendations on when to use such pretrained models based on our observations.

4.5 Appendix

4.5.1 Detailed ablations on Charades

This section presents multiple ablations evaluating our design decisions and provides recommendations on when to use our detection pretrained models. Note that, in these experiments, we report the performance evaluated for every frame on Charades [174] (in contrast to the standard evaluation protocol of evaluating only on 25 frames per clip), which is measured using mAP (without class weights). This is similar to the original setting, but provides more robust and fine-grained performance metrics.

Number of frozen segments in Volume Freeze: As shown in Table 4.4a, Volume Freezing provides a relative improvement of +1.51% mAP over classification pretrained X3D [40]. By default, we consider a single random frozen segment in a given clip. Benefit from having multiple such frozen segments is minimal (only +0.04% mAP).

Variations of Volume MixUp: We consider Volume MixUp of two clips with hard or smooth (having seamlessly changing temporal alpha masks) boundaries. Among these, smooth boundaries preserve the temporal consistency better, giving a +0.32% mAP boost over the former, as shown in Table 4.4b. Volume MixUp applied in a random feature-level as in [189] is not much better (only +0.06% mAP) than the same augmentation applied always at the input level.

Windowing strategies in Volume CutMix: Among the windowing methods of Volume CutMix discussed earlier, transient view performs slightly better (+0.10% mAP) than transient window as shown in Table 4.4c.

Training schedule: We always pretrain our detection models with the same training schedule as baseline classification models (in terms of both gradient iterations and computations) as shown in Table 4.4d. Our detection pretrained models consistently outperforms baselines. As often practiced in temporal activity detection, if we initialize our models with checkpoints provided in original works, it makes the models train faster (100k vs. 300k iterations) and converge to a better optimum.

Volume Freeze	mAP (%)
Baseline (cls.)	17.28
Single segment	18.79
Two segments	18.83

(a) **Volume Freeze** with a single or two separate frozen segments. Multiple frozen segments does not give a considerable benefit. — X3D [40]

Volume MixUp	mAP (%)
Baseline (cls.)	17.28
Hard boundaries	18.86
Seamless	19.18
Seamless (manifold)	19.24

(b) **Volume MixUp** with seamless boundaries preserve temporal consistency and outperforms hard boundaries. Gain from Manifold MixUp [189] is minor. — X3D [40]

Volume CutMix	mAP (%)
Baseline (cls.)	17.28
Transient window	18.89
Transient view (w/ Constant window)	18.99

(c) **Volume CutMix** with transient windows or transient views (with a constant window for each clip). Transient views show a slightly better performance. — X3D [40]

Model	Pretrain	mAP (%)	Pretrain FLOPs (P)
X3D - (300k from scratch)	cls.	16.02	91.10
	det.	17.89	91.10
X3D - (100k from cls. ckpt.)	cls.	17.28	30.36
	det.	19.28	30.36

(d) **Performance w.r.t the training schedule and checkpoint.** We always pretrain our detection models with the same schedule as baselines (same iterations → same computations $\times 10^{15}$). Initializing models from checkpoints provided in original works, give faster and better convergence. — X3D [40]

Pretrain	Act. per frame		Boundary	
	=1	>1	False	True
cls.	8.63	18.72	17.34	16.18
det. (VF)	-0.14	+1.85	+1.41	+1.49
det. (VM)	+1.30	+2.06	+1.87	+1.94
det. (VC)	+1.16	+1.86	+1.68	+1.85

(e) **Stats from the val set** on the improvement (cls.→det.) for single vs. multi-action frames, and action boundary vs. non-boundary regions. We see a consistently larger improvement in multi-action frames compared to single-action frames. Improvement from introducing boundaries in minimal. — X3D [40]

Table 4.4: **Ablations on our design choices:** Here, We show the performance in mean Average Precision (mAP) for fine-grained predictions on Charades [174] (i.e., making decisions per every frame rather than evenly-sampled 25 frames from each validation clip). Negative changes are de-emphasized, whereas the best performances that we highlight are underlined.

Statistics showing the points of improvement: Table 4.4e shows the performance boosts of each augmentation, measured under different settings to highlight what happens (1) in multi-action frames and (2) around action boundaries. Our augmentations significantly improve the mAP in a multi-action setting, as we introduce multi-action frames in pretraining. Even though we introduce action boundaries during pretraining, it does not show a contrasting change between boundary and

Pretrain	Fine/Fast	Coarse	Slow
cls.	17.28	+0.85	+0.21
det. (VF)	18.79	-0.27	-0.27
det. (VM)	19.18	+0.12	-0.01
det. (VC)	18.99	-0.14	-0.39

(a) **At lower temporal resolutions** (< pretrained resolution), detection pretrained models are not improved (Fine/Fast→Coarse/Slow) as much as classification pretrained ones. Classification captures an overview of a clip which can be better generalized to different temporal resolutions. However, detection pretrained models still consistently outperform others. — $\text{SlowFast}_{\text{det}}$ [41]/ Coarse-Fine [76]

Coarse-Fine	$T/32$	$T/16$	$T/8$
cls.	22.80	+0.68	+0.49
det. (VC)	22.84	+0.14	-0.01

(b) **In temporal aggregation**, classification pretrained models perform better. We show this with the fusion module in Coarse-Fine [76], which aggregates Fine features with Gaussians at a given standard deviation $\{T/32, T/16, T/8\}$. Here, if we increase the standard deviation, we temporally aggregate (dilate) more, where classification pretrained features show consistently higher improvement. — Coarse-Fine [76]

Table 4.5: **Ablations on when to use detection vs. classification pretrained models:** Detection pretrained models are under-performing their counterparts at different temporal resolutions (Table 4.5a) or strong temporal aggregation (Table 4.5b)). Here, We show the performance in mean Average Precision (mAP) for fine-grained predictions on Charades [174] (i.e., making decisions per every frame rather than evenly-sampled 25 frames from each validation clip). Negative changes are de-emphasized, whereas the best performances that we highlight are underlined.

non-boundary regions in the downstream.

At different temporal resolutions: In Table 4.5a, we consider classification and detection pretrained models at a different (×4 lower) temporal resolution, by comparing the coarse/slow stream vs. the fine/fast stream. Classification pretrained models consistently give a better relative change than the detection pretrained models (the consistent gain of Coarse/Slow w.r.t. Fine/Fast). This is because, when pretrained with classification, models can capture an overview of an input clip, allowing it to better generalize for different temporal resolutions. However, the absolute performance metrics are always better in detection counterparts.

In temporal aggregation: In Coarse-Fine [76], fusion module aggregates Fine features with Gaussians (at defined standard deviation). As in Table 4.5b, by evaluating at different standard deviations (different aggregation scales), we see that classification pretrained features give a better temporal aggregation compared

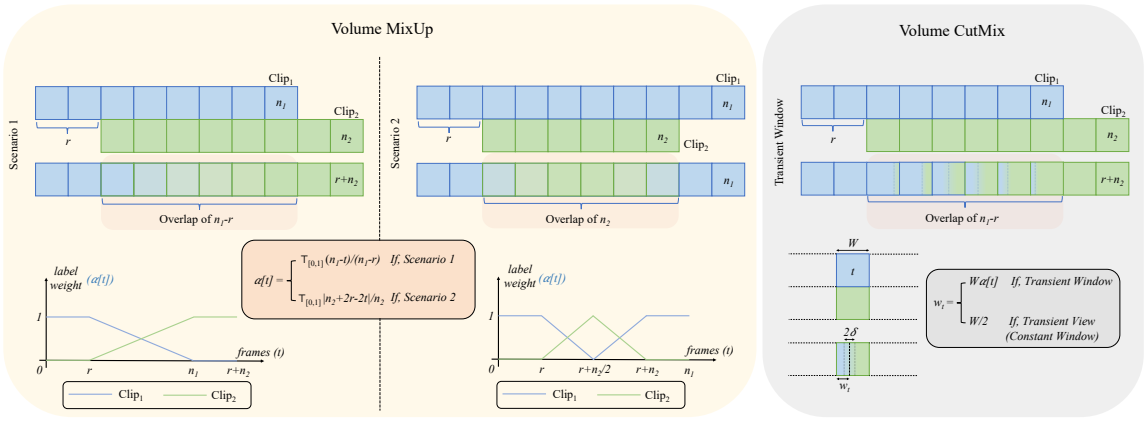

Figure 4.7: **Detailed view of masks** used in Volume MixUp (left) and Volume CutMix (right). In Volume MixUp, a temporal alpha mask ($\alpha[t]$) is defined, which is further visualized above (left) for both scenarios. When $n_2 + r \geq n_1$ (Scenario 1), $\alpha[t]$ is defined so that the augmented clip transit from $\text{Clip}_1 \rightarrow \text{Clip}_2$. Otherwise, transition happens as $\text{Clip}_1 \rightarrow \text{Clip}_2 \rightarrow \text{Clip}_1$. A truncation operation ($\text{T}_{[0,1]}$) is applied to clip the mask value into the range of $[0,1]$. Here, the labels (one-hot) of each clip are summed with weights $\alpha[t]$ and $(1-\alpha[t])$ to create soft-labels. In Volume CutMix above (right), a spatial mask ($\mathbf{M}[t]$) is defined for each frame at time t, creating two windows in the overlapping region (split by a vertical plane). In Transient Window setting, the location of the vertical plane (w_t) depends on $\alpha[t]$ (same one as in Volume MixUp), and in Transient View (Constant Window), w_t is half of the frame-width (W). A small spatial region of 2δ is defined between widows to have a smooth spatial transition. The labels (one-hot) of each clip are summed with weights $|\mathbf{M}[t]|$ and $(1-|\mathbf{M}[t]|)$ to create soft-labels. Given a matrix, $|\cdot|$ computes its "area" as an average of all its elements.

to detection counterparts. This is due to the same reason mentioned above: classification features capture an overview, hence better generalize across scales. Based on these observations, we recommend using classification models in temporal aggregation or change of temporal resolution.

4.5.2 On the Truncation Operator

As shown on Fig. 4.7 (left), the truncation operator $\text{T}_{[0,1]}$ makes sure that the alpha mask ($\alpha[t]$) is within the range of [0,1], even in the non-overlapping region. It can be defined as,

$$\mathsf{T}_{[0,1]}(x) = \begin{cases} 1 & \text{if } x \geq 1, \\ 0 & \text{if } x < 0, \\ x & \text{otherwise,} \end{cases}$$

where any value $x \geq 1$ or $x < 0$ is capped at either 1 or 0 respectively. Based on this, alpha mask $\alpha[t]$ is deifined so that the augmented clips have a smooth transition as $\text{Clip}_1 \rightarrow \text{Clip}_2$ (in Scenario 1), or as $\text{Clip}_1 \rightarrow \text{Clip}_2 \rightarrow \text{Clip}_1$ (in Scenario 2).

4.5.3 On the Spatial Mask

Spatial mask **M** defines a vertical plane to split each frame within the overlapping region into two windows (see Fig. 4.7 (right)). The location of this vertical plane (w_t) can either depend on $\alpha[t]$ (in Transient Window) or be constant (in Transient View). This can be given as,

$$\mathbf{M}[t][:, j] = \begin{cases} 1 & \text{if } j < w_t - \delta, \\ 0 & \text{if } j \geq w_t + \delta, \\ \dfrac{w_t + \delta - j}{2\delta} & \text{otherwise,} \end{cases}$$

and,

$$w_t = \begin{cases} \lfloor W\alpha[t] \rceil & \text{if Transient Window,} \\ \lfloor W/2 \rceil & \text{if Transient View,} \end{cases}$$

where W is the width of the frame, and δ is a small value defining the smooth spatial transition between windows. $\lfloor \cdot \rceil$ will round the operand to the nearest integer.

4.5.4 Details on datasets

Kinetics-400 [20] is a large-scale activity classification dataset commonly-used for pretraining video models. It contains 240k training and 20k validation videos Each clip contains a single action out of 400 human action categories, and comes with video-level annotations. Kinetics clips are usually ~10s long.

Charades [174] is a mid-scale activity classification or temporal detection dataset consisting of ~9.8k continuous videos with frame-level annotations of 157 common household activities. The dataset is split as ~7.9k training and ~1.8k validation videos. Each video contains an average of 6.8 activity instances, often with multiple activity classes per frame, and has longer clips averaging a duration of ~30s.

MultiTHUMOS [229] is a small-scale dataset which contains a subset of THUMOS [74] untrimmed videos densely annotated for 65 different action classes. It provides action segment annotations for 413 videos, split as 200 for training and 213 for validation. On average, it contains 1.5 labels per frame and 10.5 action classes per video. When compared to Charades [174], this has a significantly smaller number of videos, but each clip is longer in duration.

Chapter 5

Video-conditioned Text as Free Semantic Supervision

5.1 Overview

Video understanding poses significant challenges, often adding to the complications in image domain such as model complexity and annotation costs. The additional temporal dimension and different modalities of data introduce useful cues, but also can be redundant, raising interesting questions about trade-offs. Activity Recognition (*i.e.*, classification) in particular— as the prominent task in video understanding— has long been explored by the community in these research directions. Whether it is efficient architecture variants ranging from CNNs [40, 97, 160] to Transformers [5, 9, 39], training schemes from fully-supervised [19, 41] to self-supervised [42, 141, 153] or data regimes from unimodal [185, 209] to multimodal [56, 124], the progress has been steady and exciting. More recently, with the availability of internet-scale paired image-text data, the direction of vision-language models (VLMs) [72, 147] have emerged dominant, achieving strong generalization across numerous benchmarks. However, the progress of VLMs in the video domain is yet to be caught-up to its full potential.

Following the seminal VLMs such as CLIP [147] and ALIGN [72], there have been significant strides in tasks such as image classification [221, 233, 237], open-vocabulary object detection [53, 121], text-to-image retrieval [177, 225] and robot manipulation [73, 236]. Such models are usually pretrained on paired image-text data based on a contrastive learning framework. The idea is to have two separate backbones— an Image Encoder and a Text Encoder, that generate embeddings in a joint latent space. To optimize this space, the corresponding pairs of embeddings

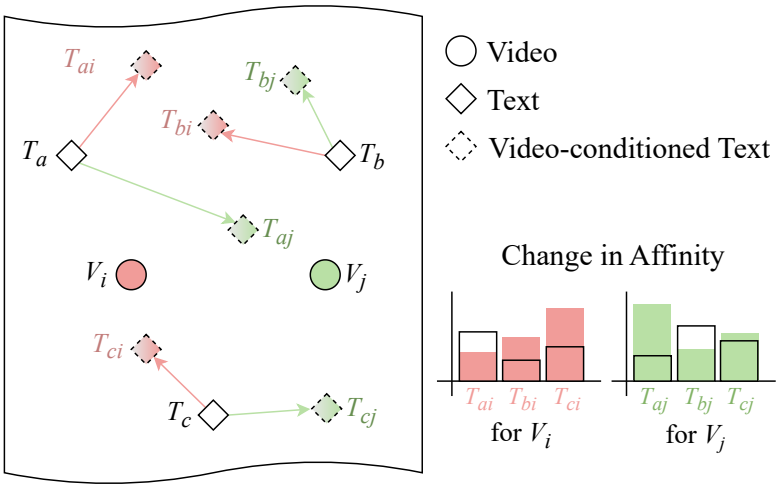

Figure 5.1: **Video-conditioned Text Representations:** Pretrained image-VLMs can generate reasonable visual embeddings for videos (*e.g.* by temporally-pooling frame embeddings), together with paired text embeddings. However, usually, these text embeddings are not dependent on visual information— meaning, they are common for every video. Such representations lack the flexibility to align properly in a shared vision-language latent space, when optimized based on a contrastive similarity (*i.e.*, *Affinity*) w.r.t. all videos. However, with *Video-conditioned Text* representations that specialize uniquely for each video, we grant more freedom for text embeddings to move in the latent space, and adapt to different scenarios (*e.g.* more-challenging recognition tasks).

are drawn closer, by increasing their similarity (*i.e.*, *Affinity*). The key advantage of such models is that, at inference, any semantic concept (given as a text input) can be embedded in the same space, giving intriguing zero-shot or few-shot transfer capabilities [3, 236]. For instance, CLIP [147] excels at classifying unseen attribute categories (*e.g.* objects, scenes), or even counting such occurrences [236]. However, these VLMs do not perform well in tasks that require specialized knowledge, such as localizing (*e.g.* detection/segmentation) or temporal reasoning (*e.g.* activity recognition), at least not out-of-the-box, as their training objective has not seen any location or temporal cues. Yet, with task-specific finetuning, such models can readily be adapted to specialized domains [53, 126].

In the video domain, training VLMs from scratch may show a limited success [212]— while also being expensive— due to the lack of paired data at scale. As a compromise, the common practice is to adapt pretrained image-VLMs to video, by introducing temporal information. Such methods either insert temporal modules

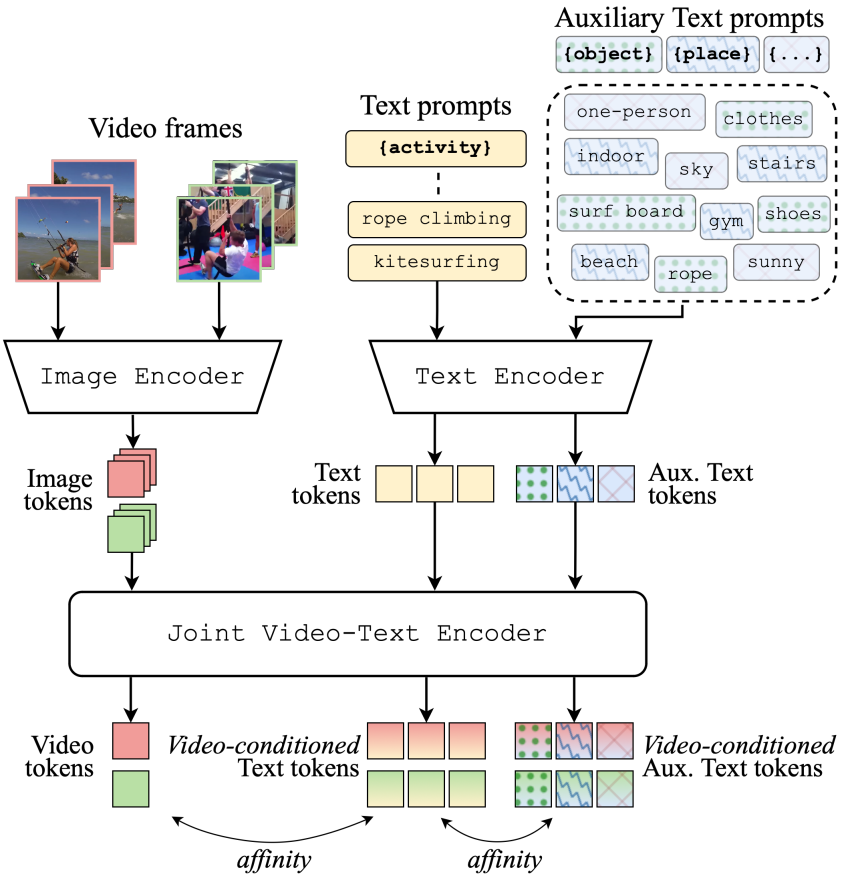

Figure 5.2: **Overview of VicTR:** First, we extract image (*i.e.*, frame) and text tokens using a pretrained image-VLM. Next, such tokens go through a joint video-text encoder, generating video tokens and *video-conditioned* text tokens, based on which, we compute affinity-based logits for classification. Optionally, any semantic concept (given as auxiliary text) can also be processed similarly, to help guide the classifier. This is motivated based on the co-occurrence of semantics (*e.g.* `rope`, `gym`, `one-person`) and categories-of-interest, *i.e.*, activity classes in our setting (*e.g.* `rope climbing`). Here, the color change of text tokens represents the idea of video-conditioning.

within the image backbone itself to have cross-frame interactions [126], or use a post-processing video head on-top of the image backbone [8, 99, 112, 190]. In both cases, image embeddings are enhanced as video embeddings. However, the use of text embeddings varies among different approaches. Text may either be

discarded [99], kept frozen [112, 190], used as conditioning [8] (to further enhance video embeddings), or fully-updated jointly with video [126]. More often than not, the main focus is on visual embeddings (*i.e.*, converting image → video), and the impact of updating text has been limited.

Nevertheless, video models benefit from semantic information [71, 196, 236]. In fact, certain attributes (*e.g.* objects, scene or human subjects) are directly tied with specific activities, and can simplify their recognition. For instance, the presence of attributes such as [`rope`, `gym`, `one-person`] can narrow down the potential activity to `battling ropes` or `rope climbing`. VLMs are especially suited to take advantage of such semantics. Any concept represented as text can be visually-grounded based on paired embeddings (in zero-shot), to extract relevant attributes for a given input that benefit recognition tasks. Such visually-grounded semantics are cheap in-terms of both annotation and compute costs, yet highly-useful.

Motivated by the above we propose `VicTR`, focusing on adapting text information to the video domain. More specifically, we generate *Video-conditioned Text* embeddings (see Fig. 5.1), while jointly-training both textual and visual features generated by an image-VLM. By finetuning text embeddings, we observe significant gains in our framework, compared to just finetuning visual embeddings (similar to the observations in [237]). We can also make use of freely-available auxiliary semantic information, represented in the form of visually-grounded text embeddings. Fig. 5.2 shows an overview of the proposed architecture. Our video-conditioned text embeddings are unique to each video, allowing more-flexibility to move in the latent space and generalize to complex downstream tasks. Optionally, our video-conditioned auxiliary text can further help optimize this latent space. We evaluate `VicTR` on few-shot, zero-shot, short-form and long-form activity recognition, validating its strong generalization capabilities among video-VLMs.

5.2 Background: image-VLMs to video

In this section, we introduce the generic framework for adapting image-VLMs to video, and discuss how prior work fit into it. We consider CLIP [147] as the image-VLM, which is widely-adapted thanks to its convincing performance and open-source models. It consists of two encoders: Image and Text, optimized together on internet-scale paired image-text data. Image Encoder ($\texttt{Enc}_{\texttt{img}}$) is a ViT [36]. Given an input image $I \in \mathbb{R}^{H \times W \times 3}$, it is broken down to patch embeddings (*i.e.*, tokens) and processed through multiple transformer layers. The class token [`cls`] is sampled as the visual embedding $e_{\texttt{img}}$. Text Encoder ($\texttt{Enc}_{\texttt{txt}}$) is a causal transformer, operating on tokenized text. Each class-label (or, any semantic con-

cept) given as text T, is first converted into a prompt based on a template such as "`a photo of {class}.`", and tokenized with Byte Pair Encoding (BPE) [170] at the input of Text Encoder. Following multiple causal transformer layers, the `[EOS]` (*i.e.*, end-of-sequence) token is extracted as the text embedding $e_{\texttt{txt}}$.

$$e_{\texttt{img}} = \texttt{Enc}_{\texttt{img}}(I),$$
$$e_{\texttt{txt}} = \texttt{Enc}_{\texttt{txt}}(T).$$

The two encoders are jointly-optimized with Cross-Entropy loss, where logits are computed based on the similarities (*i.e.*, *affinities*) between visual and text embeddings. The corresponding pairs of embeddings (*i.e.*, positives) are drawn together (↑ affinity) in a joint embedding space, whereas the others (*i.e.*, negatives) are pushed apart (↓ affinity).

$$\texttt{Affinity}(e_{\texttt{img}}, e_{\texttt{txt}}) = \frac{\langle e_{\texttt{img}}, e_{\texttt{txt}} \rangle}{\|e_{\texttt{img}}\|_2 \|e_{\texttt{txt}}\|_2}.$$

When adapting this framework to the video domain, the above Image encoder, Text encoder and the learning objective usually stays the same. But now, video frames $V \in \mathbb{R}^{\mathcal{T} \times H \times W \times 3} = [I^1, I^2, \cdots, I^{\mathcal{T}}]$ become inputs to the Image encoder (while each being processed separately), and further go through a Video Head $\texttt{Head}_{\texttt{vid}}$ to induce temporal reasoning capabilities. Optionally, text embedding $e_{\texttt{txt}}$ may also be updated or used as a conditioning within the Video Head.

$$e_{\texttt{vid}}, [e_{\texttt{txt}}] = \texttt{Head}_{\texttt{vid}}(e^1_{\texttt{img}}, \cdots, e^{\mathcal{T}}_{\texttt{img}}, [e_{\texttt{txt}}]).$$

Here, $[\cdot]$ denotes optional embeddings. This Video Head may just be a temporal pooling layer or a temporal transformer as in [112, 190], or may even consist of more-specialized modules. Text embeddings could either be discarded as in [99], used as a conditioning as in [8], or jointly-updated with video embeddings as in [126]. Finally, logits are computed based on video-text affinities if text is not discarded, or as a linear mapping of video embeddings if text is discarded. This generic framework is shown in Fig. 5.3 (top-left), along with variations of prior work in Fig. 5.3 (bottom-left).

5.3 Video-conditioned Text Representations

In `VicTR`, we adapt a pretrained image-VLM (*e.g.* CLIP [147]) to video, focusing more on text representations. Refer to Fig. 5.3 (right) for a detailed view. The

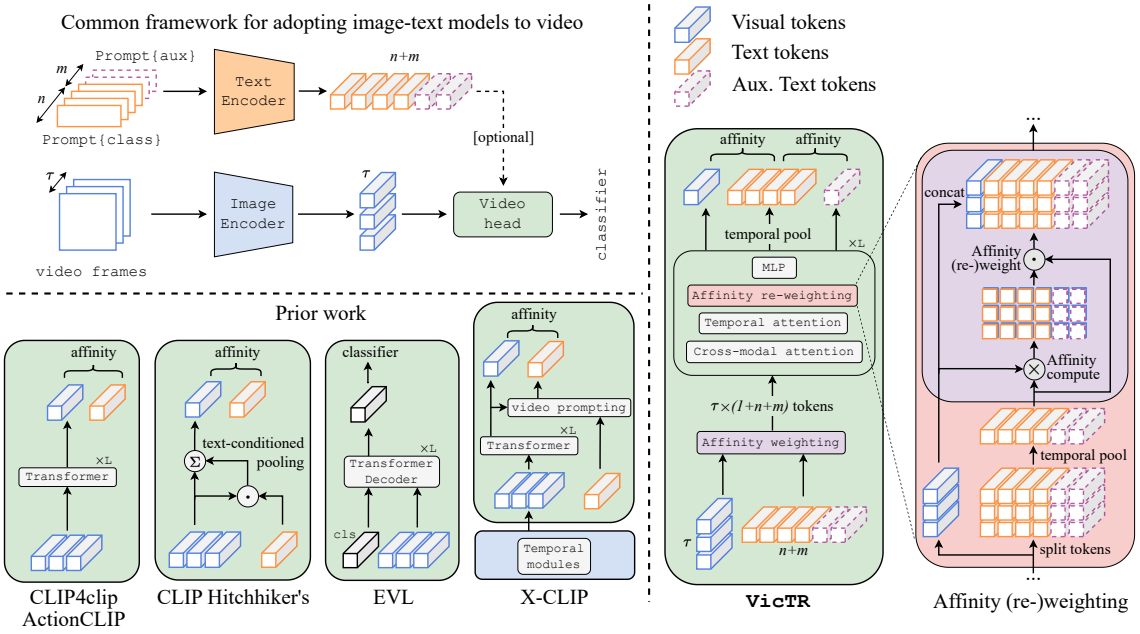

Figure 5.3: **Detailed view of VicTR compared to prior art:** There exist multiple closely-related work on adapting pretrained image-VLMs to video, that follow a common framework (top-left): a video head followed by modality-specific encoders. Often, text information is kept unchanged [112, 190], or even discarded [99] (bottom-left). CLIP Hitchhiker's [8] uses text as conditioning. X-CLIP [126] jointly-optimizes visual and text tokens. Yet, it provides limited information for text to contrast-against: only temporally-aggregated visual embeddings, showing marginal gains from updating text. In contrast, VicTR allows text to contrast against both fine-grained visual and other text information, while also jointly-optimizing both modalities. We generate *video-conditioned text* representations, *i.e.*, text uniquely-specialized for each video (refer to Fig. 5.1). Our video head consists of three key operations: (1) *Token-boosting*, (2) *Cross-modal attention*, and (3) *Affinity (re-)weighting* (right). Token-boosting creates dedicated text tokens per video and per timestep, weighted by per-frame affinities. These enable us to model variations of semantics (represented as text) over time. Affinity (re)-weighting highlights or down-plays each text class, grounded on visual information. Cross-modal attention enables message passing between both visual-textual and textual-textual modes. Also, optionally, VicTR can make use of auxiliary semantics (*e.g.* object, scene, human-subjects) given as visually-grounded text (refer to Fig. 5.2).

image-VLM has not seen any temporal information during training. While it obviously affects the temporal reasoning capabilities of the visual embeddings—which most prior work focus on addressing, it also affects the text embeddings as well. The learnt latent space (and, the affinity-based objective) depends on both these embeddings. Thus, we consider text equally as important, if not more, in contrast to prior work

VicTR consists of a joint video-text model as $\texttt{Head}_{\texttt{vid}}$, which consumes both visual and text embeddings from the image-VLM. It outputs text embeddings uniquely-specified for each video, *i.e.*, *Video-conditioned Text* embeddings. It relies on three main components: (1) *Token-boosting*, (2) *Cross-modal attention*, and (3) *Affinity (re-)weighting*. Optionally, it can also benefit from any semantic concept available as auxiliary text, to optimize its latent space. Following subsections look at each of these in detail.

Let us first introduce a few additional notations. Consider a fixed vocabulary of n activity-classes given by $[T^1, T^2, \cdots, T^n]$, and optional m auxiliary semantic categories given by $[A^1, A^2, \cdots, A^m]$. The corresponding text embeddings can be denoted as $\{e^x_{\texttt{txt}} \mid x = 1, 2, \cdots, n\}$ and $\{e^y_{\texttt{aux}} \mid y = 1, 2, \cdots, m\}$. Also, given an input video V^i of $\mathcal{T}$ frames, the corresponding image embeddings can be denoted as $\{e^{i,t}_{\texttt{img}} \mid t = 1, 2, \cdots, \mathcal{T}\}$. The inputs to our Video Head are $e^{i,t}_{\texttt{img}}$, $e^x_{\texttt{txt}}$ and $e^y_{\texttt{aux}}$ tokens. As visual embeddings are extracted per-frame and the text embeddings per prompt, there is no interaction among frame tokens, among text tokens or, across frame-text tokens up to this point.

5.3.1 Token-boosting

To introduce *video-conditioned text* embeddings, we first create a dedicated set of text tokens per video, by replicating the outputs of the backbone text encoder. Going further, we also create text tokens per each frame. This is done by weighting text tokens with the corresponding frame-text affinities. Formally, given $(n+m)$ text tokens, we end up with $\mathcal{T} \times (n+m)$ dedicated text tokens per video, at the input of our video head. Refer to Fig. 5.3 (right).

$$e^{i,t,x}_{\texttt{txt}} = e^x_{\texttt{txt}} \cdot \texttt{SigAffinity}(e^{i,t}_{\texttt{img}}, e^x_{\texttt{txt}}),$$
$$e^{i,t,y}_{\texttt{aux}} = e^y_{\texttt{aux}} \cdot \texttt{SigAffinity}(e^{i,t}_{\texttt{img}}, e^y_{\texttt{aux}}).$$

Here, $\texttt{SigAffinity}(\cdot)$ corresponds to affinity-weights normalized in $[0, 1]$ range. We convert the values given by $\texttt{Affinity}(\cdot)$ that lie in $[-1, 1]$, to be affinity-weights,

by scaling with a learnable weight (w) and feeding through a sigmoid.

$$\texttt{SigAffinity}(\cdot) = \texttt{Sigmoid}(w \cdot \texttt{Affinity}(\cdot)).$$

Although such affinity-weights based on the original image-VLM embeddings are not ideal for temporal reasoning, it initializes a noisy-version of our *video-conditioned text* embeddings that gets updated iteratively, later in the network. Such a token-boosting brings multiple other benefits. (1) More tokens means higher the model capacity. It can help learn better representations, but also adds a compute overhead (which we handle through other measures, as discussed later). (2) It also highlights relevant text tokens by grounding text on visual embeddings, while diminishing irrelevant ones. Subsequent attention mechanisms attend less to such diminished tokens, simplifying the gradient flow during learning. In other words, it acts as a soft-selection of relevant semantics, specific to each video. (3) Finally, it enables our model to capture variations of semantic categories over time. How certain attributes appear (or, disappear) over time is an important motion cue for activity recognition.

Next, we concatenate such boosted text tokens with visual tokens (corresponding to $\mathcal{T}$ frames), and feed $\mathcal{T} \times (1 + n + m)$ tokens to the subsequent layers.

$$z^{i,t} = \texttt{Concat}(e^{i,t}_{\texttt{img}},\ e^{i,t,x}_{\texttt{txt}}\big|_{x=\{1,\cdots,n\}},\ e^{i,t,y}_{\texttt{aux}}\big|_{y=\{1,\cdots,m\}}).$$

Such $Z^i_0 = [z^{i,1},\ \cdots,\ z^{i,\mathcal{T}}]$ tokens go through L transformer layers in our Video Head. Each layer (l) consists of cross-modal attention, temporal attention, affinity (re-)weighting and linear (MLP) layers.

5.3.2 Cross-modal and Temporal attention

We consider our token representation to be two-dimensional (*i.e.*, cross-modal and temporal), and apply divided self-attention (MSA) on each axis as in [5, 9]. First, we have a Cross-modal attention layer. Here, each visual token could attend to all text tokens at the same timestep, and each text token could attend to both the visual token and other text tokens at the same timestep. Since text tokens are already affinity-weighted, attention weights do not draw information from irrelevant semantic classes. Next, we have a Temporal attention layer. Here, both visual and text tokens go through a shared set of parameters, learning temporal cues in visual modality (*i.e.*, $e_{\texttt{img}} \rightarrow e_{\texttt{vid}}$), and modeling variations of semantics across time in textual modality.

$$\hat{Z}^i_l = Z^i_l + \texttt{MSA}_{\texttt{cross}}(\texttt{LN}(Z^i_l)),$$
$$\bar{Z}^i_l = \hat{Z}^i_l + \texttt{MSA}_{\texttt{temporal}}(\texttt{LN}(\hat{Z}^i_l)).$$

Here, $\texttt{LN}(\cdot)$ stands for LayerNorm operation. Having a divided attention across two-axes instead of a joint-attention eases the compute requirement of our video head.

5.3.3 Affinity (re-)weighting

As previously discussed, the original affinities based on the image-VLM embeddings can be noisy, in the context of temporal reasoning. Now, as we have updated both our visual (*i.e.*, video) and text tokens with cross-modal and temporal information, they are in a better state to re-compute affinities. Hence, we compute new affinity values and re-weight the text tokens accordingly. Refer to Fig. 5.3 (rightmost). First, we split video and text tokens as in,

$$\left[\bar{e}^{\,i,t}_{\texttt{vid},l},\ \bar{e}^{\,i,t,x}_{\texttt{txt},l}\Big|_{x=\{1,\cdots,n\}},\ \bar{e}^{\,i,t,y}_{\texttt{aux},l}\Big|_{y=\{1,\cdots,m\}}\right] = \bar{z}^{\,i,t}_{l}.$$

Next, we temporally-pool the text tokens to come up with a compressed representation, on which we perform affinity re-weighting. This is similar to token-boosting, but done with updated video-text embeddings that are already video-conditioned. Without loss of generality, the same operations apply for auxiliary text tokens.

$$\begin{aligned}
\bar{e}^{\,i,x}_{\texttt{txt},l} &= \texttt{Pool}(\bar{e}^{\,i,t,x}_{\texttt{txt},l}),\\
\bar{e}^{\,i,t,x}_{\texttt{txt},l} &= \bar{e}^{\,i,x}_{\texttt{txt},l} \cdot \texttt{SigAffinity}(\bar{e}^{\,i,t}_{\texttt{vid},l},\ \bar{e}^{\,i,x}_{\texttt{txt},l}).
\end{aligned}$$

Finally, such affinity (re-)weighted text tokens are concatenated with visual tokens, as $\bar{Z}^{i}_{l}$, and go through an MLP.

$$Z^{i}_{l+1} = \bar{Z}^{i}_{l} + \texttt{MLP}(\bar{Z}^{i}_{l}).$$

5.3.4 Affinity-based classifier

Following L transformer layers in our Video Head, we temporally-pool all tokens. We end up with a single video embedding, n activity-text embeddings and m aux-text embeddings. We further aggregate auxiliary embeddings, leaving a single embedding per each of the k semantic categories (*e.g.* object, scene, human-subjects). Finally, we compute logits based on affinity, similar to the CLIP [147] objective, and use Cross-Entropy loss for optimization.

$$\begin{aligned}
\texttt{logit}^{\,i,x} &= \texttt{Affinity}(e^{\,i}_{\texttt{vid},L},\ e^{\,i,x}_{\texttt{txt},L}),\\
\texttt{logit}^{\,i,x,y}_{\texttt{aux}} &= \texttt{Affinity}(e^{\,i,x}_{\texttt{txt},L},\ e^{\,i,y}_{\texttt{aux},L})\ \Big|_{y=\{1,\cdots,k\}}.
\end{aligned}$$

5.3.5 Discussion on design decisions

Auxiliary semantic information: We rely on optional semantics (or, attributes) in the form of visually-grounded auxiliary text, to improve our *video-conditioned text* embeddings. This is guided by the loss on $\texttt{logit}_{\texttt{aux}}$. The vocabulary of such auxiliary texts is fixed (*i.e.*, common for all videos) per dataset. On Charades, we consider 97 auxiliary text classes, and on Kinetics-400, we use 88 classes (refer the appendix for more details). To highlight only the relevant semantics for a given video, we visually-ground them via (1) cross-modal attention with visual embeddings, and (2) affinity weighting. Finally, to compute $\texttt{logit}_{\texttt{aux}}$, we create one *representative embedding* per each of the k semantic categories, by average pooling aux embeddings within a category ($k = 4$ for Charades and $k = 3$ for Kinetics-400).

Alternative weighting schemes: Our text (re-)weighting method is similar to a contrastive training objective (as in CLIP [147]), which is based on visual-text affinities. We find this complementary nature beneficial. It highlights relevant text (and diminish irrelevant ones) within each intermediate layer of our Video Head. This iterative process fixes the initial noisy affinities resulting from the original image-VLM embeddings, when fused with better temporal cues in subsequent layers. We also explored other weighting schemes such as learnable weights or attention-based weights, which are not directly-connected to the training objective. They do not provide any improvements.

Visual-only or Text-only classifiers: We also explored different classifiers (*i.e.*, how we compute logits), considering (1) a visual-only classifier as in [99], (2) a text-only classifier, or (3) an affinity-based classifier as in [126, 147]. The last performs the best. Even though we primarily focus on updating text embeddings, it still makes sense to rely on video-text affinities to be the training objective (or, classifier), as it is complementary to the components within our Video Head.

5.4 Experiments

To validate the merits of VicTR, we experiment on few-shot and zero-shot activity recognition (on HMDB-51 [86] and UCF-101 [179]), as well as short-form (on Kinetics-400 [79]) and long-form recognition (on Charades [174]). Following sub-sections will detail our implementation, evaluation settings, datasets and the results.

Implementation details: We use a pretrained CLIP [147] as our image-VLM backbone. Our Video Head is randomly-initialized having 4 transformer blocks

similar to [190], which is applied on-top of CLIP backbones. We consider an embedding dimension of 512/768 (w/ heads 8/12) corresponding to CLIP B/16 and L/14 backbone variants. Our output video-text embeddings are further mapped into 256-dimensional embeddings prior to computing affinity-based logits. We use an AdamW [109] optimizer with a cosine schedule for training. On Kinetics-400 [79], we finetune our model for 30 epochs with a batch size of 256 using 8e-6/8e-5 learning rates for backbone/newly-initialized parameters, similar to [126]. On Charades [174], we finetune for 50k iterations with a batch size of 64 using 5e-7/5e-4 learning rates for backbone/newly-initialized parameters, similar to [8]. We use augmentations and input sampling strategies similar to [126] for Kinetics-400 and similar to [99] for Charades.

Evaluation settings: In our experiments, we compare against prior art VLMs on each dataset. Since the direction of adapting image-VLMs to video is relatively-recent, their absolute performance may not be the state-of-the-art in some cases (*e.g.* long-form recognition), but we report numbers in comparable settings. For each experiment, we report pretraining settings, #frames-per-view, #views-at-inference and compute-per-view (GFLOPs) as supplementary metrics. We evaluate single-label activity recognition performance with Top-1 (%) accuracy, and multi-label recognition with Average Precision (mAP%). When reporting FLOPs, we consider the cost of computing a single affinty-based logit (*i.e.*, the cost for one video-text pair) similar to [126].

5.4.1 Few-shot and Zero-shot Transfer

Data: We consider the downstream datasets HMDB-51 [86] and UCF-101 [179] to evaluate few-shot and zero-shot performance of our model. UCF-101 is a classification dataset collected from YouTube. It contains ∼13k clips annotated with 101 action classes. HMDB-51 is relatively small and contains ∼7k clips with 51 annotated classes. Both datasets have three splits of training/test data. In few-shot evaluation, we randomly sample 2, 4, 8, or 16 clips per class to create our training sets, same as in [126]. We use a model pretrained on Kinetics-400 [79] for 10 epochs and finetune on few-shot examples for 50 epochs, using 32-frames per view as in [126].

Few-shot results: In Table 5.1, we report top-1 accuracy on the first test split among three, in each dataset, using a single view at inference. VicTR significantly outperforms prior art, either w/o image-text pretraining (TSM [97], TimeSformer [9], Video-Swin [108]) or w/ such pretraining (X-CLIP [126], X-Florence [126]).

Model		HMDB-51				UCF-101			
	k:	2	4	8	16	2	4	8	16
Methods w/o image-text pretraining									
TSM [97]		17.5	20.9	18.4	31.0	25.3	47.0	64.4	61.0
TimeSformer [9]		19.6	40.6	49.4	55.4	48.5	75.6	83.7	89.4
Video-Swin-B [108]		20.9	41.3	47.9	56.1	53.3	74.1	85.8	88.7
Methods w/ image-text pretraining									
X-CLIP [126]		53.0	57.3	62.8	64.0	76.4	83.4	88.3	91.4
X-Florence [126]		51.6	57.8	64.1	64.2	84.0	88.5	92.5	94.8
VicTR (B/16)		60.0	63.2	66.6	70.7	87.7	92.3	93.6	95.8

Table 5.1: **Few-shot Transfer:** On HMDB-51 [86] and UCF-101 [179], we compare our method against prior art, reporting top-1 accuracy (on the first split among three test splits as in [126]). We use models pretrained on Kinetics-400 [79] for 10 epochs, and finetune on few-shot samples for 50 epochs. We randomly-sample $k = \{2, 4, 8, 16\}$ clips per class as few-shot training samples in each setting. VicTR shows a significant boost over X-CLIP [126]. Non-VLMs are de-emphasized.

Although our method uses similar backbones as X-CLIP, it even outperforms X-Florence (an extension of a more-generic foundation model) on both datasets consistently. This shows the effectiveness of our *video-conditioned text* embeddings when generalizing to downstream with few training samples.

Zero-shot results: We report zero-shot transfer performance in Table 5.2. We use a model pretrained for 10 epochs on Kinetics-400 [79] with 32-frames per view, similar to [126], and transfer to the downstream. We report mean and standard deviation on three-splits. VicTR-B/16 outperforms X-CLIP [126] by 6.4% on HMDB-51 and by 0.4% on UCF-101. Also, the performance of our model is more stable across splits. This validates that the learned *video-conditioned* text embeddings can be generalized, even w/o seeing the same categories as in the downstream, during pretraining.

5.4.2 Short-form Activity Recognition

Data: Kinetics-400 [79] is a large-scale activity recognition dataset, with 240k training and 20k validation videos. Each clip contains video-level annotations for a single activity out of 400 categories, having short ∼10s duration.

Model	#Frames	HMDB-51	UCF-101
Methods w/o image-text pretraining			
MTE [217]	-	19.7 ± 1.6	15.8 ± 1.3
ASR [191]	16	21.8 ± 0.9	24.4 ± 1.0
ZSECOC [143]	-	22.6 ± 1.2	15.1 ± 1.7
UR [249]	1	24.4 ± 1.6	17.5 ± 1.6
TS-GCN [43]	16	23.2 ± 3.0	34.2 ± 3.1
E2E [16]	16	32.7	48.0
ER-ZSAR [24]	-	35.3 ± 4.6	51.8 ± 2.9
Methods w/ image-text pretraining			
ActionCLIP [190]	32	40.8 ± 5.4	58.3 ± 3.4
X-CLIP [126]	32	44.6 ± 5.2	72.0 ± 2.3
VicTR (B/16)	32	51.0 ± 1.3	72.4 ± 0.3

Table 5.2: **Zero-shot Transfer:** On HMDB-51 [86] and UCF-101 [179], we compare our method against prior art, reporting input format (#Frames) and top-1 accuracy (%) as mean±std across the three splits of test set as in [126]. We use models pretrained on Kinetics-400 for 10 epochs. VicTR outperforms similar video-VLM adaptations. Non-VLMs are de-emphasized.

Results: We report the performance of VicTR on Kinetics-400 short-form activity recognition in Table 5.3. We consider L/14 with 8-frames per view, while using 4×3 such views at inference similar to [126]. Our method shows a competitive performance at a similar footprint to closely-related video-VLMs [99, 126]. It is also competitive with CoVeR-L [239] which is trained with 10× more data. VicTR outperforms MTV [220] by +2.7%, ViViT [5] by +3.5% and TokenLearner [162] by +1.6%, all trained on a similar scale of data, while being more-efficient.

5.4.3 Long-form Activity Recognition

Data: Charades [174] is a small-yet-challenging activity recognition dataset with ∼9.8k long-form videos. It comes with frame-level annotations of 157 daily household activities. Yet, the benchmark setting requires making video-level predictions. The data is split as ∼7.9k for training and ∼1.8k for validation. Each video contains multiple overlapping activities, having an average duration of ∼30s.

Results: We report the performance of VicTR on Charades long-form activity recognition in Table 5.4. Here, we consider both B/16 and L/14 model variants with 32-frames per view, while having 4×1 such views at inference. Our method

Model	Pretrain	#Frames	#Views	GFLOPs	Top-1
Methods w/o image-text pretraining					
Video-Swin-L (384↑) [108]	IN-21K	32	10×5	2107	84.9
TimeSformer-L [9]	IN-21K	96	1×3	2380	80.7
MTV-L [220]	JFT-300M	32	4×3	1504	84.3
Video-SwinV2-G (384↑) [108]	IN-21K+	8	4×5	-	86.8
MViTv2-L [94] (312↑)	-	40	5×3	2828	86.1
ViViT-L FE [5]	JFT-300M	32	1×3	3980	83.5
TokenLearner [162]	JFT-300M	64	4×3	4076	85.4
CoVeR-L [239]	JFT-3B	-	1×3	-	87.2
Methods w/ image-text pretraining					
ST-Adapter [129]	CLIP	32	1×3	2749	87.2
Text4Vis [204]	CLIP	32	1×3	1662	87.1
EVL [99]	CLIP	8	1×3	674	86.3
X-CLIP [126]	CLIP	8	4×3	658	87.1
VicTR (L/14)	CLIP	8	4×3	656	87.0

Table 5.3: **Short-form Activity Recognition:** On Kinetics-400 [79], we compare our method against prior art, reporting pretraining settings, input format, compute cost (GFLOPs) and top-1 accuracy (%). Here, #Frames represents the number of frames per view, while #Views represents the number of temporal×spatial crops during inference. The compute cost is reported per view. VicTR shows a competitive performance among video-VLMs with a similar cost. Non-VLMs are de-emphasized.

outperforms prior video-VLMs by a considerable margin. In fact, VicTR-B/16 shows +5.2% mAP boost over CLIP Hitchhiker's [8], and +5.5% mAP boost over ActionCLIP [190] with a similar footprint. This is a significant improvement considering the challenging Charades settings. Our method is also competitive with non-VLMs, whereas other video-VLMs lag behind. It highlights the limitations of current VLMs in long-context temporal modeling.

5.4.4 Ablation Study

In Table 5.5, we provide evidence to validate our main hypotheses. Namely, we evaluate the impact of auxiliary semantics and the effectiveness of updating text embeddings.

Model	Pretrain	#Frames	#Views	GFLOPs	mAP
Methods w/o image-text pretraining					
I3D + NL [193]	K400	128	10×3	544	37.5
EvaNet [136]	K400	64	-	-	38.1
LFB-101 [199]	K400	32	10×3	529	42.5
SlowFast-50 [41]	K400	8+32	10×3	66	38.0
SlowFast-101 + NL [41]	K400	16+64	10×3	234	42.5
X3D-XL (312↑) [40]	K400	16	10×3	48	43.4
MViT [39]	K400	32	10×3	237	47.7
AssembleNet-101 [160]	-	128	5×1	1200	58.6
Methods w/ image-text pretraining					
ActionCLIP [190]	CLIP	32	10×3	563	44.6
CLIP4clip [112]	CLIP	32	1×1	-	32.0
CLIP Hitchhiker's [8]	CLIP	32	1×1	-	44.9
VicTR (B/16)	CLIP	32	4×1	567	50.1
VicTR (L/14)	CLIP	32	4×1	2602	57.6

Table 5.4: **Long-form Activity Recognition:** On Charades [174], we compare our method against prior art, reporting pretraining settings, input format, compute cost (GFLOPs) and mean Average Precision (mAP%). The compute cost reported is per view. Here, #Views represents the number of temporal×spatial crops, each having #Frames per view). VicTR achieves strong a performance among the methods pretrained w/ image-text data by a considerable margin. Non-VLMs are de-emphasized.

Model	Kinetics-400	Charades
VicTR	84.4	50.1
VicTR (No Aux. Text)	84.2	49.8
VicTR (w/ CLIP Visual emb.)	84.0	49.7
VicTR (w/ CLIP Text emb.)	83.3	41.7

Table 5.5: **Ablating main hypotheses:** On Kinetics-400 [79] and Charades [174], we measure the importance of auxiliary text prompts. We also show that updating text is most critical in our framework, rather than updating visual embeddings (i.e., temporally-pooled CLIP image embeddings is as good as our video embeddings).

Auxiliary semantics do help. We rely on extra semantic information to guide our latent embedding space. We see that such auxiliary text is giving +0.2% gain

Model	mAP
VicTR	50.1
VicTR (No Affinity weighting)	48.8
VicTR (w/ joint-attention)	44.8
VicTR (Text Classifier)	41.2
VicTR (Visual Classifier)	43.1

Table 5.6: **Ablating design decisions:** On Charades [174], we evaluate different design decisions of VicTR. First, we show the effectiveness of Affinity Weighting and divided attention in our framework. We also replace our visual-text affinity-based logits with simpler visual-only or text-only logits to show the benefit of ours.

on Kinetics-400 and +0.3% mAP gain on Charades. This conveys the potential of semantics, but also the limitations of not having ground-truth annotations corresponding to them.

Updating text embeddings is more effective. To evaluate which of our embeddings (video or *video-conditioned text*) are critical, we replace them with the corresponding original CLIP [147] embeddings (*i.e.*, temporally-pooled frame, or text). We see that the proposed *video-conditioned text* are significantly-more effective, and when replaced, the performance drops −1.1% on Kinetics-400 and −8.4% mAP on Charades. In contrast, when our video embeddings are replaced, the performance drops only −0.4% and −0.4% mAP, respectively. Meaning, the CLIP frame embeddings are on-par with our video embeddings, but our *video-conditioned text* embeddings are significantly improved.

In Table 5.6, we ablate and justify our design decisions. Namely, we evaluate our affinity weighting mechanism, divided attention, and affinity-based classifier.

Affinity-weighting and divided attention do help. We see a +1.3% mAP performance gain by having our affinity (re-)weighting mechanism. While joint-attention may be more expressive compared to divided attention, it can incur training difficulties. As a result, we see the divided attention enjoying a significant +5.3% mAP boost.

Affinity-based classifier is required. As we previously discussed, our affinity weighting mechanism makes more-sense in the context of the same affinity-based loss formulation. To verify this, we replace such affinity-based logits with text-only or visual-only logits, which are just linear mappings of the corresponding embeddings. These significantly underperforms, with −8.9% mAP and −7.0% mAP, respectively.

5.5 Conclusion

In this paper, we introduced VicTR, a framework for adapting image-VLMs to video, with a focus on *video-conditioned text* embeddings. It can also benefit from freely-available auxiliary semantic information in the form of visually-grounded text, to guide the learned latent space. Our evaluations verified the importance of updating text embeddings, across multiple activity recognition benchmarks, under few-shot, zero-shot, short-form and long-form settings. We believe that this work reveals the importance of using language embeddings for temporal reasoning.

5.6 Appendix

5.6.1 Additional discussion

Details on auxiliary text classes: On Charades [174], we use 97 auxiliary classes: 43 objects, 15 places, 5 people-counts and 34 atomic-actions. People-count prompts are manually-selected, whereas the others are already annotated in the dataset. On Kinetics-400 [79], we use 88 auxiliary classes: 40 objects, 43 places and 5 people-counts. Atomic-actions on Kinetics-400 are too diverse to be categorized as a concise set, and thus omitted. On Kinetics-400, people-counts are similarly selected, and the others are generated by prompting ChatGPT3.5 with the set of 400 activity classes. The auxiliary vocabulary for each dataset is given below.

On Charades [174], we have the following:

Objects: *bag, bed, blanket, book, box, broom, chair, closet, cabinet, clothes, cup, glass, bottle, dish, door, doorknob, doorway, floor, food, groceries, hair, hands, laptop, light, medicine, mirror, paper, notebook, phone, camera, picture, pillow, refrigerator, sandwich, shelf, shoe, sofa, couch, table, television, towel, vacuum, window.*

Places: *basement, garage, pantry, recreation room, walk-in closet, laundry room, stairs, hallway, dining room, entryway, home office, bathroom, kitchen, bedroom, living room.*

People: *no people, one person, two people, three people, several people.*

Atomic-actions: *doing nothing, awakening, closing, cooking, dressing, drinking, eating, fixing, grasping, holding, laughing, lying, making, opening, photographing, playing, pouring, putting, running, sitting, smiling, sneezing, snuggling, standing, taking, talking, throwing, tidying, turning, undressing, walking, washing, watching, working.*

On Kinetics-400 [79], we have the following:

Objects: *bow and arrow, flowers, leaves or tree, computer, bed or baby crib, glass or bottle, dumbbell, treadmill or gym equipment, trampoline, mechanical bull or roller skates, bowling ball, cabinet or windows or dining table, sailboat or jet ski, fishing rod, cleaning supplies, grooming tools, pool, shoes, toilet, rope or ladder, barbecue grill or campfire, makeup tools, shovel, laundry or clothes, books or drawing materials, baseball, basketball or golf club, gymnastics mat, ice skates, dessert, fruits or vegetables, food items, fire extinguisher, hammer or meat grinder, musical instruments, board game, sporting equipment, gas pump, shopping cart, newspaper, animals, car, tractor or bicycle, rock climbing gear, electric sharpener or shredder.*

Places: *home, living room, dining room, bathroom, kitchen, bedroom, backyard or garden, staircase, hair salon, restaurant, outdoor, mountain or cliff, grass field, snow or ice, river or sea, sky, gym or fitness center, supermarket, foundary or workshop, forest, sports field, stadium, court or arena, massage palor, dance floor or stage, road or sidewalk, swimming pool, restaurant or bar, entrance or doorway, hospital or emergency room, bowling alley, building or skyscraper, theatre or auditorium, farm, recording studio or music room, news room, repair shop, garage, archery or shooting range, beach, underwater or sea bed, office or workspace, park, arcade or casino, school or classroom.*

People: *no people, one person, two people, three people, several people.*

On the selection of datasets: In literature, activity recognition is considered as the prominent video classification task. To understand the effectiveness of our *video-conditioned text* representations, we tackle a variety of activity recognition benchmarks. This includes few-shot and zero-shot activity recognition (on HMDB-51 [86], UCF-101 [179]), short-form recognition (on Kinetics-400 [79]) and long-form recognition (on Charades [174]). It is worth noting that Kinetics-400 usually contains single-person activities, whereas Charades includes multiple people and complex overlapping activities. Together, these provide a thorough spread of scenarios for both single-label and multi-label classification. Our evaluation setting is similar to many other prior work which evaluate on classification [99, 126, 190], yet extensive as it includes diverse contexts.

Compute requirement: Token-boosting increases the footprint of our model. However, our Video-Head is still lightweight, requiring minimal additional computations. In fact, it amounts for only 0.2% (0.5B) of total FLOPs in B/16 16-frame model (285B), and only 0.1% (0.6B) in L/14 8-frame model (656B). This is because of three reasons: (1) having fewer layers (*i.e.*, 4 layers vs. 12/24 layers) and lightweight attention modules (*i.e.*, temporal and cross-modal attention vs. spatial attention) compared to the image-VLM backbone [147], (2) processing significantly fewer tokens (*i.e.*, only temporal and text-class tokens remain), and (3) doing text-

Model	Rich text	HMDB-51	UCF-101
X-CLIP [126]	✗	44.6 ± 5.2	72.0 ± 2.3
VicTR (w/ CLIP Text emb.)	✗	43.9 ± 0.7	67.2 ± 0.7
VicTR	✗	51.0 ± 1.3	72.4 ± 0.3
VicTR (w/ CLIP Text emb.)	✓	43.9 ± 1.5	70.7 ± 0.3
VicTR	✓	52.1 ± 0.5	77.4 ± 0.2

Table 5.7: **Impact of more-descriptive text:** We replace class labels in HMDB-51 [86] and UCF-101 [179] with rich class-descriptions generated by ChatGPT3.5. On zero-shot evaluation, our video-conditioned text embeddings benefit significantly-more from rich text inputs, compared to the CLIP [147] text embeddings.

conditioning only after the backbone (*i.e.*, for the most part, all text embeddings go through shared computations). Ovrall, VicTR has a comparable footprint to prior work such as [99, 126, 190], providing a fair comparison (see respective GFLOPs in Table 5.3 and Table 5.4).

Other forms of semantic information: In our framework, we use a fixed vocabulary of auxiliary prompts as semantic inputs, that is specific to each dataset. Another way of providing semantic information is in the form of captions. If available, a detailed set of captions may provide better semantic supervision. However, they come with a significant cost, since they need to be annotated per-video. In contrast, our auxiliary prompts are freely-available and can be selected with only a minimal effort, as they are common for all videos in a dataset. Our model learns to highlight relevant information for a given video implicitly, via affinity weighting, without needing any ground-truth annotations.

5.6.2 Additional experiments

Impact of more-descriptive text: By default, we use class labels with the standard CLIP [147] prompt template to generate text embeddings. However, if available, more-descriptive text such as human-annotated captions (expensive) or machine-generated descriptions (inexpensive) can provide richer information for our cross-modal attention, improving *video-conditioned text* representations. We validate this claim by replacing class-labels with rich class-descriptions from ChatGPT3.5 (Table 5.7). On zero-shot evaluation, the relative gains from our text improve on both HMDB-51 [86] (+7.1% → +8.2%) and UCF-101 [179] (+5.2% → +6.7%), also raising the absolute performance.

Model	Type	Params	NExT-QA
Random	-	-	20.0
CaKE-LM [180]		2.7B	34.9
InternVideo [194]	Enc-Dec	1.3B	49.1
SeViLA [231]		4.1B	63.6
Just-Ask [222]		75M	38.4
X-CLIP [126]	Enc only	194M	43.8
VicTR (B/16)		167M	45.5

Table 5.8: **Video reasoning with VQA:** On NExT-QA [207] zero-shot evaluation, our model outperforms comparable baselines. Large-scale models with LLM decoders are de-emphasized.

Other reasoning tasks: The primary scope of this paper is on a broad spectrum of recognition tasks. Yet, it is also applicable to other reasoning tasks such as video VQA. In Table 5.8, we evaluate VicTR on NExT-QA [207] under zero-shot settings, showing gains over comparable baselines with encoder-only designs (*i.e.*, no LLM decoders). This validates that our model can readily be extended to other tasks with jointly-embedded video and text.

Chapter 6

Object-Centric Diffusion for Fast Video Editing

6.1 Overview

Diffusion models [61, 178] stand as the fundamental pillar of contemporary generative AI approaches [63, 82, 158]. Their success primarily stems from their unparalleled diversity and synthesis quality, surpassing the capabilities of earlier versions of generative models [50, 83]. On top of that, recent methods such as Latent Diffusion Models (LDMs) [158] exhibit scalability to high-resolution inputs and adaptability to different domains without requiring retraining or finetuning, especially when trained on large-scale datasets [167]. These traits catalyzed impactful adaptations of pretrained LDMs across a spectrum of applications, including text-guided image generation [158], editing [6], inpainting [158], as well as video generation [12] and editing [140, 203, 243].

Nevertheless, diffusion models come with various trade-offs. One immediate drawback is their inefficiency due to the sampling process, which involves iterating a denoising neural network across numerous diffusion steps. Despite the availability of techniques like step distillation [117, 165] and accelerated samplers [110, 111, 178] that expedite image synthesis, efficient sampling solutions for video generation are still lacking. Besides, special attention must be paid to maintaining temporal coherency among frames when creating or modifying a video, in order to avoid flickering artifacts or lack of correlation between frames. To tackle this challenge, approaches like diffusion inversion [140, 178] and cross-frame self-attention [140, 243] have been introduced, but all at the cost of further increasing the computational load.

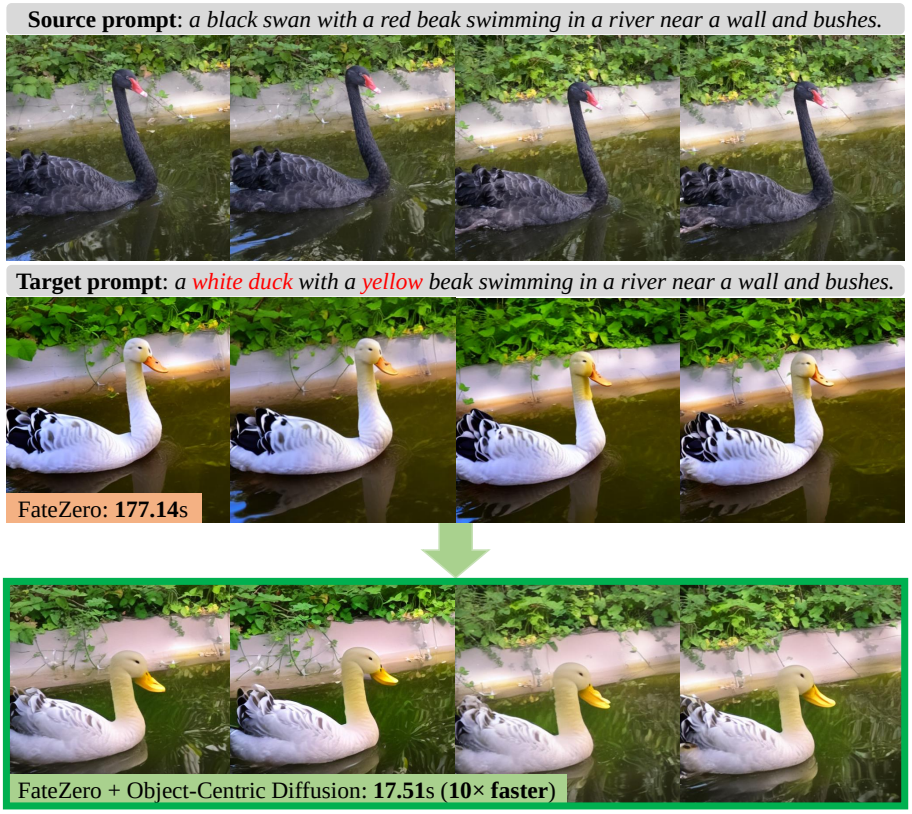

Figure 6.1: **OCD speeds up video editing.** We show exemplar editing results of FateZero [140] with and without our OCD optimizations. When including our techniques, the editing is 10× faster than the baseline with similar generation quality.

This paper centers on video editing models and presents novel solutions to enhance their efficiency. We first examine the current video editing frameworks, identifying the key elements that increase their latency. These encompass memory overheads, such as those associated with attention-based guidance from diffusion inversion, as well as computational bottlenecks, including excessive cross-frame attention and an unnecessarily high number of sampling steps. We show that significant improvements can be achieved by adopting off-the-shelf optimizations namely efficient samplers and leveraging token reduction techniques in attention layers such as token merging (ToMe) [14, 15].

Additionally, we argue that video editing users may be particularly sensitive to the quality of edited foreground objects, as opposed to slight degradations

in the background regions. Consequently, we propose two new and efficient techniques that harness this object-centric aspect of editing applications. Our first solution, *Object-Centric Sampling*, involves separating the diffusion process between edited objects and background regions. This strategy enables the model to focus the majority of its diffusion steps on the foreground areas. Consequently, it can generate unedited regions with considerably fewer steps, all while ensuring faithful reconstructions in all areas. With our second solution, *Object-Centric ToMe*, we incentivise each token reduction layer to merge more tokens within less crucial background areas and exploit temporal redundancies that are abundant in videos. The combination of such techniques, coined Object-Centric Diffusion (OCD), can seamlessly be applied to any video editing method without retraining or finetuning.

As we demonstrate in extensive experiments, the combined implementation of these object-centric solutions enhances the quality of video editing in both salient foreground object and background, while considerably reducing the overall generation cost. We apply our proposed solutions to inversion-based and ControlNet-based video editing models, speeding them up by a factor of 10× and 6× respectively while also decreasing memory consumption up to 17× for a comparable quality (see Fig. 6.1).

We summarize our contributions as follows:

- We analyze the cost and inefficiencies of recent inversion-based video editing methods, and suggest simple ways to considerably speed them up.
- We introduce Object-Centric Sampling, which separates diffusion sampling for the edited objects and background areas, limiting most denoising steps to the former to improve efficiency.
- We introduce Object-Centric ToMe, which reduces number of cross-frame attention tokens by encouraging their fusion in background regions.
- We showcase the effectiveness of OCD by optimizing two recent video editing models, obtaining very fast editing speed without compromising fidelity.

6.2 Efficiency Bottlenecks

To investigate the main latency bottlenecks in video editing pipelines, we run a benchmark analysis on a representative pipeline, derived from FateZero [140]. More specifically, in this case-study we are primarily interested in the role of components such as diffusion inversion and cross-frame attention, as these techniques are crucial for the task and ubiquitous in literature [45, 93, 140, 203, 243]. To study the impact of such components, we perform text-guided edits on 8 frame

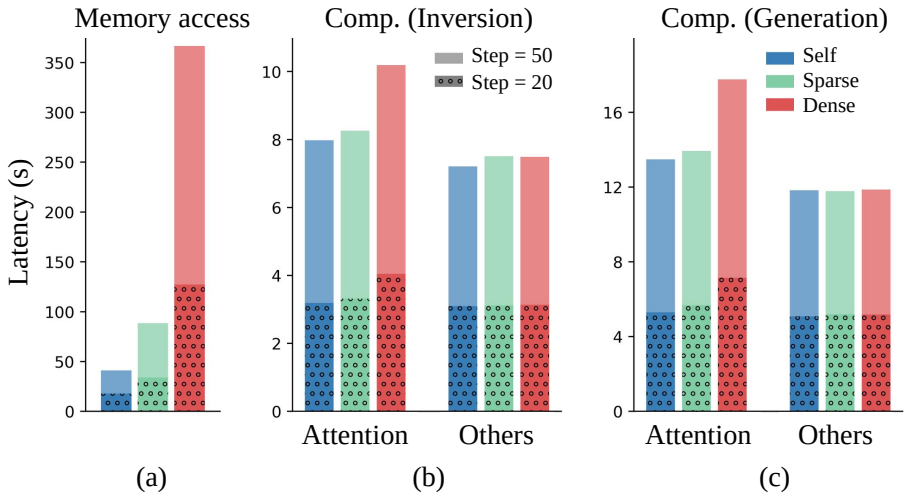

Figure 6.2: **Latency analysis** of video editing models. At various diffusion steps, latency is dominated by memory access operations. Among pure computations, attention alone is the bottleneck, especially when using dense cross-frame interactions. As attention is the main responsible for most of the memory overhead, we hypothesize that reducing the number of its tokens have a significant impact on latency.

clips while varying (1) the number of diffusion steps (50→20), and (2) the span of cross-frame attention, *i.e.*, self (1-frame), sparse (2-frames) or dense (8-frames). In doing so, we probe the model's latency using DeepSpeed [120]. Based on the latency measurements aggregated over relevant groups of operations as reported in Fig. 6.2, we make the following observations:

Observation 1. As testified by Fig. 6.2 (a), inversion-based pipelines are susceptible to memory operations. Not only the latency due to memory access dominates the overall latency in all the tested settings, but it also scales exponentially with the span (i.e., #frames) of cross-frame attention.

Observation 2. When ruling out memory and considering computations only, as in Fig. 6.2 (b) and Fig. 6.2 (c), we appreciate that attention-related operations are as expensive as (or often more) than all other network operations combined (*i.e.*, convolutions, normalizations, MLPs etc.). This latter finding is consistent both in inversion and generation phases.

Observation 3. More-powerful cross-frame attention operations (i.e., dense), that increase the temporal stability of generations [243], come with a high latency cost,

as observed in Fig. 6.2 (b) and Fig. 6.2 (c).

6.2.1 Off-the-shelf acceleration

We explore whether the solutions available for image generation mitigate the key computational bottlenecks observed for video editing. More specifically, we study whether token merging [15], as an effective strategy to improve the attention cost for image generation [14], reduces the high memory and computational cost of video editing. Moreover, given the linear relation between the number of diffusion steps and memory costs of inversion-based editing to store the attention maps and activations [45, 140, 187, 203], we explore how using faster noise schedulers improves the video editing efficiency. In what follows, we describe how the off-the-shelf accelerations are adopted for video editing:

Faster self-attention Aiming to speed-up attention operations, we utilize ToMe [14, 15] to merge redundant tokens. More specifically, ToMe first splits the input self-attention tokens into source (*src*) and destination (*dst*) sets. Then, the similarity between the source token $\mathbf{x}_i$ and the destination token $\mathbf{x}_j$ is computed based on normalized cosine similarity as follows:

$$\text{Sim}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2}\left[\frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \left\|\mathbf{x}_j\right\|} + 1\right]. \tag{6.1}$$

Finally, the top r% similarities are selected, and source tokens are piggybacked into the corresponding destination tokens by means of average pooling. This process reduces successive operations by removing merged tokens, as only destination and unmerged-source tokens (*unm*) are forwarded, at the cost of a slight information loss.

In our first round of experimentation, we observed the potential of ToMe in speeding up video editing, as a simple reduction of 1/16 in key-value tokens reduces the model latency by a factor of 4×. Nevertheless, its naive application completely breaks generation quality, as testified in Fig. 6.3 (d). However, we could restore an outstanding generation quality by applying two simple yet crucial tricks. Specifically, we *i)* pair the token locations (both *dst* and *unm*), for the same network layer and diffusion timestep, between inversion and generation (see ToMe-paired Fig. 6.3(e)), and *ii)* we resample the destination token locations for every frame (Fig. 6.3 (f)). We refer the reader to the supplementary material for more details about these ToMe optimizations.

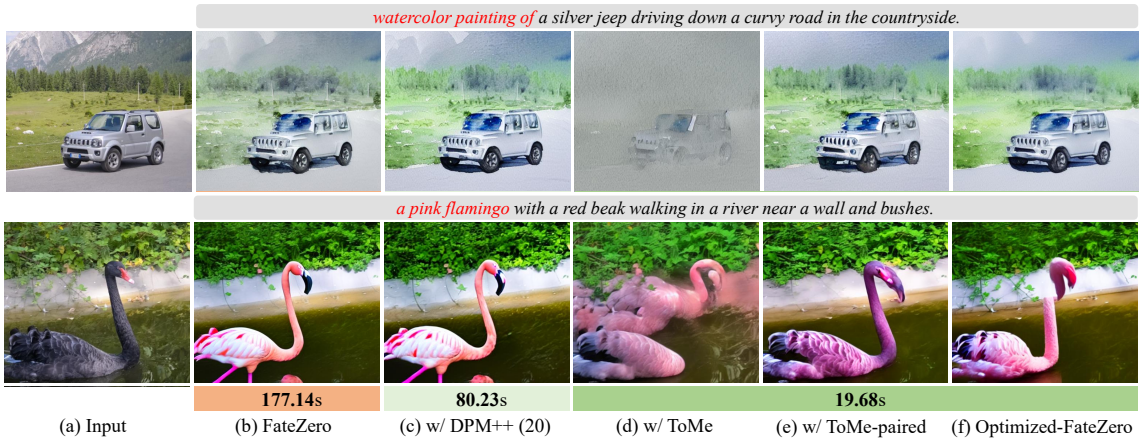

Figure 6.3: **Off-the-shelf accelerations:** First, we replace the (b) default sampler with (c) DPM++ [110, 111], allowing to reduce sampling steps from 50→20 without a heavy degradation. Then, by applying ToMe, memory and computational overhead decreases, yet results degrade significantly (d). We therefore implement (e) pairing of ToMe indexes between inversion and generation and (f) per-frame resampling of destination tokens, regaining the quality. Altogether, we coin the resulting model *Optimized-FateZero*.

Faster noise scheduler Although we observe that the commonly-used DDIM scheduler [178] suffers considerable quality degradation when reducing the number of diffusion steps (without any finetuning), we notice this is not the case with a more modern DPM++ [111], that can operate with as few as 20 iterations. As illustrated in the example reported in Fig. 6.3 (c), this replacement does not affect the quality of the editing, yet it impacts the generation latency tremendously.

In the remainder of the paper, we refer to models optimized with these off-the-shelf accelerations as *optimized*, and will serve as strong baselines to our proposed Object Centric Diffusion. Such a baseline is extremely fast, yet prone to sporadic generation artifacts (Fig. 6.3 (f), Fig. 6.7), that we shall fix with the following object-centric solutions.

6.3 Object-Centric Diffusion

Although off-the-shelf optimizations can notably improve latency, they occasionally introduce some editing artifacts in foreground objects, that we solve by exploiting

the object-centric nature of video editing tasks, also enabling further reductions in latency. Following our key assumption regarding the significance of foreground objects for many video editing applications, we propose two add-ons for diffusion models to channel their computational capacity towards foreground regions, and we describe them in Sec. 6.3.1 and in Sec. 6.3.2. For both contributions, we assume we have access to a foreground mask m, highlighting in every frame the locations of objects to be edited. Such mask can be obtained, for instance, with a pretrained segmentation model [84], visual saliency [144, 235] or by cross-attention with text-prompts.

6.3.1 Object-Centric Sampling

The process of sampling in diffusion models entails gradually transitioning from the initial noise $\mathbf{z}_T \sim \mathcal{N}$ to a sample $\mathbf{z}_0$ from the real data distribution, obtained through a series of denoising steps $p(\mathbf{z}_{t-1}|\mathbf{z}_t)$. Each step involves running a computationally-heavy denoiser (*e.g.* UNet [159]), and high-quality generations demand a large number of sampling steps. We hereby introduce an efficient object-centric diffusion sampling technique tailored for editing tasks, prioritizing high-quality generation specifically in designated foreground regions.

Our Object-Centric Sampling scheme is described in Algorithm 1. Specifically, we first consider the foreground mask m and split the latent variable $\mathbf{z}_T$ into foreground and background latents denoted by $\mathbf{z}_T^f$ and $\mathbf{z}_T^b$. This operation is performed by a generic *gather* function, splitting representation tokens or feature map pixels into two disjoint sets. Then, instead of performing a single diffusion process, we disentangle the generation of foreground and background regions: such a decoupled scheme allows us to reduce the sampling steps on the latter based on an hyperparameter φ. Once foreground and background latents have been generated, we rely on a *scatter* operation to recompose a full feature map. Nevertheless, we observed that performing this recomposition step at the end of the whole diffusion process (*i.e.*, merging variables $\mathbf{z}_0^f$ and $\mathbf{z}_0^b$) typically results into weak editing performance, in the form of color distribution-shift and boundary artifacts between foreground and background regions. We therefore introduce two further solutions to this issue. First, we introduce a normalization step (inspired by batch normalization) applying an affine transformation to the foreground latents, shifting and scaling them to match the mean and standard deviation of background representations. Even more importantly, we move the scattering of foreground and background latents *within* the sampling process, at a given timestep T_b which is controlled by a hyperparameter γ. This choice introduces

Algorithm 1: Object-Centric Sampling

Require: Number of training steps T ▷ Default: 1000
Require: Number of inference steps N ▷ Default: 20
Require: Blending steps ratio $\gamma \in [0,1]$ ▷ Default: 0.25
Require: Background acceleration rate $\varphi > 1$
Require: Foreground mask $m \in \{0,1\}^{H \times W}$

$\Delta T \leftarrow T/N$ ▷ Calculate step size
$T_b \leftarrow \gamma * T$ ▷ Calculate step to start blending

// Split foreground and background latents
$\mathbf{z}_T \sim \mathcal{N}(0, I)$
$\mathbf{z}_T^f \leftarrow \texttt{gather}(\mathbf{z}_T, m)$
$\mathbf{z}_T^b \leftarrow \texttt{gather}(\mathbf{z}_T, 1-m)$

// Sampling the foreground latents at normal rate
for $t = T \quad ; \quad t > T_b \quad ; \quad t = t - \Delta T$ **do**
  $\mathbf{z}_{t-1}^f \sim p(\mathbf{z}_{t-1}^f | \mathbf{z}_t^f)$
end for

// Sampling the background latents at faster rate
for $t = T \quad ; \quad t > T_b \quad ; \quad t = t - \varphi * \Delta T$ **do**
  $\mathbf{z}_{t-1}^b \sim p(\mathbf{z}_{t-1}^b | \mathbf{z}_t^b)$
end for

// Normalize, merge and continue sampling
$\bar{\mathbf{z}}_t^f \leftarrow \texttt{normalize}(\mathbf{z}_t^f, \mathbf{z}_t^b)$
$\mathbf{z}_t \leftarrow \texttt{scatter}(\bar{\mathbf{z}}_t^f, \mathbf{z}_t^b)$
for $t = T_b \quad ; \quad t > 0 \quad ; \quad t = t - \Delta T$ **do**
  $\mathbf{z}_{t-1} \sim p(\mathbf{z}_{t-1} | \mathbf{z}_t)$
end for

some diffusion timesteps $t \leq T_b$, that take care of smoothing out such artifacts and seamlessly blend foreground and background latents. We empirically observed that allocating ≈25% (*i.e.*, $\gamma = 0.25$) of the sampling steps to this blending stage is usually sufficient to generate spatially and temporally consistent frames. We also note that for localized editing tasks, *e.g.* shape and attribute manipulations of objects, the background sampling stage can be completely skipped, which results in even faster generation and higher fidelity reconstruction of background regions, as demonstrated in Fig. 6.7.

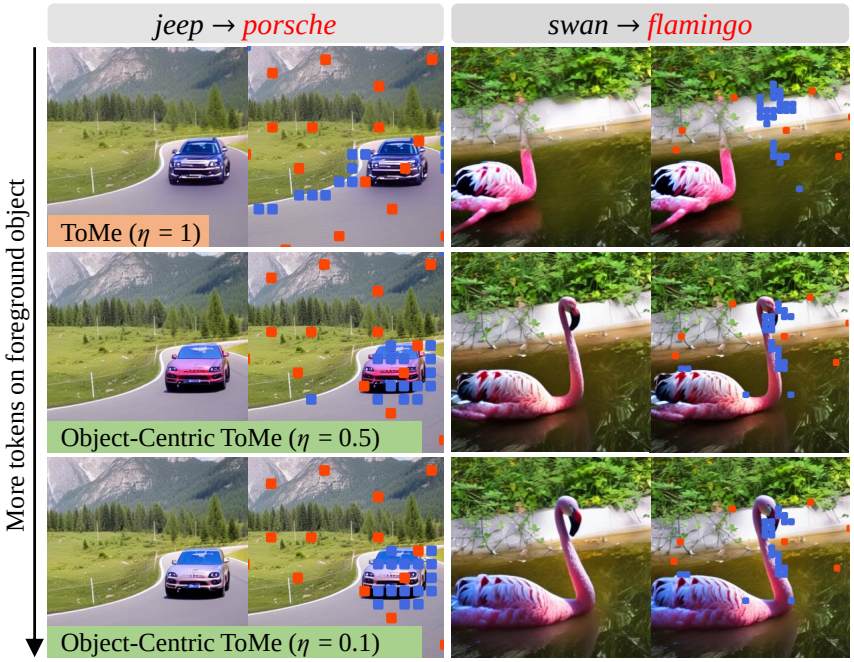

Figure 6.4: **Object-Centric Token Merging:** By artificially down-weighting the similarities of source tokens of foreground objects, we accumulate in their locations tokens that are left unmerged (in blue). Destination tokens (in red) are still sampled randomly within a grid, preserving some background information. Merged source tokens (not represented for avoiding cluttering) will come from the background.

6.3.2 Object-Centric Token Merging

We further introduce an effective technique that promotes token merging in background areas while discouraging information loss in representations of foreground objects. More specifically, whenever applying ToMe, we associate each source token $\mathbf{x}_i$ a binary value $m_i \in \{0,1\}$, obtained by aligning foreground masks to latent resolution, that specifies whether it belongs to the foreground region. Then, we simply account for m_i in computing similarity between the source token $\mathbf{x}_i$ and the destination token $\mathbf{x}_j$, as follows:

$$\eta\text{-Sim}(\mathbf{x}_i, \mathbf{x}_j, m_i) = \begin{cases} \text{Sim}(\mathbf{x}_i, \mathbf{x}_j) & \text{if} \quad m_i = 0; \\ \eta \cdot \text{Sim}(\mathbf{x}_i, \mathbf{x}_j) & \text{if} \quad m_i = 1, \end{cases}$$

where $\eta \in [0,1]$ is a user-defined weighting factor ($\eta = 1$ corresponds to the original ToMe). By reducing the value of η, we deliberately weaken the similarities of source

tokens corresponding to edited objects, therefore reducing their probability of being merged. The behavior of the weighting factor can be appreciated in Fig. 6.4: as η decreases, the unmerged tokens (in blue) tend to locate more and more on the edited objects, avoiding information loss in their locations.

Merging spatio-temporal token volumes Diffusion-based video editing heavily relies on cross-frame attention operations to increase the temporal-consistency in generated frames [93, 243], incurring in severe computational bottlenecks. The standard ToMe merges tokens in the spatial dimension only, missing the opportunity to tackle redundant tokens in the temporal dimension, which are prominent in videos. As a remedy, we apply ToMe *within spatiotemporal volumes*. Besides taking advantage of temporal redundancy, this strategy allows for greater flexibility in choosing how to trade-off merging spatial vs. temporal information by simply varying the size of the volumes.

6.4 Experiments

We test OCD in the context of two families of video editing models. Specifically, we look into inversion-based video editing, where we rely on FateZero [140] as base model, and control-signal-based architectures, for which we optimize a ControlVideo [243] baseline. To provide fair comparisons in each setting, we rely on checkpoints and configurations provided by the corresponding authors. We evaluate each model using a benchmark composed of DAVIS [137] video sequences and edit prompts utilized in the original baseline methods.

Implementation details For both video editing models, we pad the mask m to a rectangular shape and implement the gather operation in Algorithm 1 as a crop around the foreground object. Although potentially suboptimal for non-rectangular objects, this strategy proves effective in practice (Tab. 6.5) and enables the use of regular dense convolutions within UNet, which are typically highly optimized and even faster than sparse counterparts. Such an approximation would however not be necessary for other convolution-free editing pipelines, *e.g.* fully based on transformer architectures [130]. As common in the literature and because we target zero-shot video editing, we optimize some of the hyperparameters (*e.g.* blending steps ratio γ, similarity re-weighting factor η) per-sequence for FateZero. For ControlVideo, we adopt the original hyperparameters reported in the paper. η is selected in $\{0.1, 0.5, 0.9\}$ based on heuristics: a rule-of-thumb, is

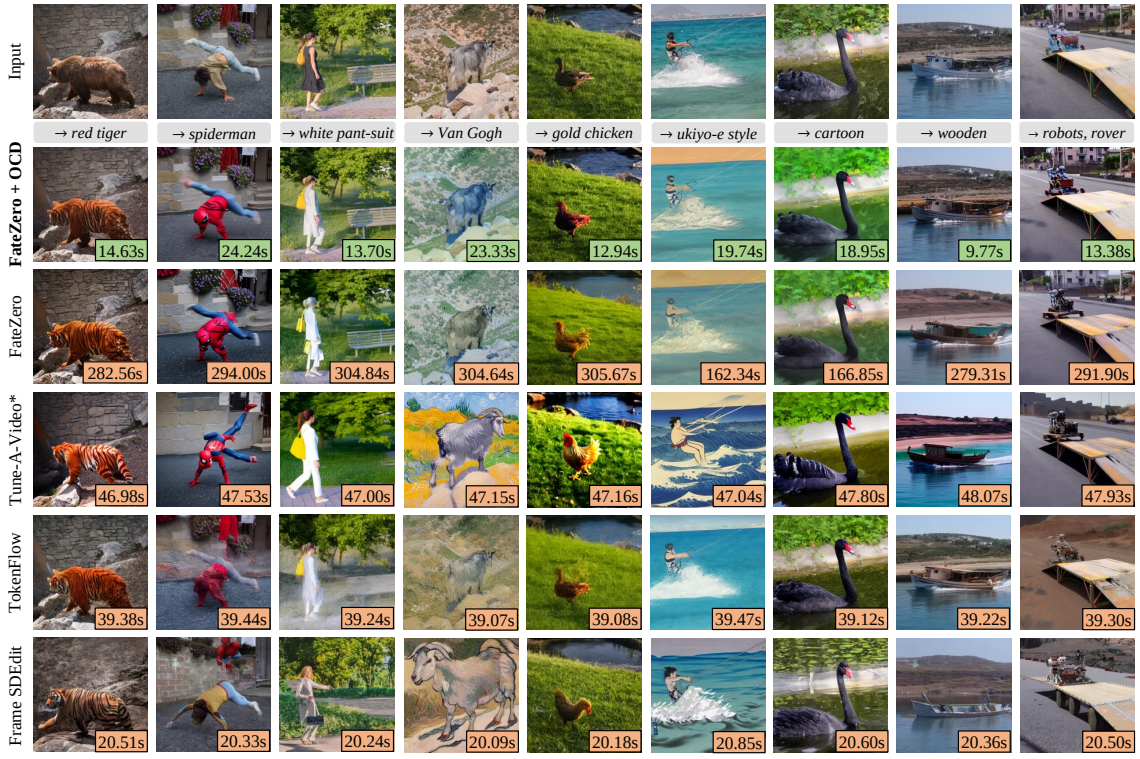

Figure 6.5: **Qualitative comparison with the sota editing methods:** OCD yields significantly-faster generations over the baseline we build on-top of (i.e., FateZero [140]) and other state-of-the-art methods, without sacrificing quality. Tune-A-Video [203] is finetuned on each sequence (denoted with *, finetuning time not included in latency).

to use a lower value (preserving more foreground), when the object has complex motion or texture (i.e., it is challenging to reconstruct). As for the saliency masks, we utilize the segmentation masks available within the DAVIS dataset. We refer the reader to supplementary materials for more-detailed setup.

Evaluation metrics For evaluating editing quality, we rely on fidelity metrics such as Temporal-Consistency of CLIP embeddings (Tem-con) and average CLIP-score aggregated over all sequence-prompt pairs. To report latency, we measure the average wall-clock time to edit a video on a single V100 GPU.

Model	Tem-con ↑	Cl-score ↑	Latency (s) ↓	
			Inversion	Generation
Framewise Null + p2p [60, 123]	0.896	0.318	1210.60	130.24
Tune-A-Video + DDIM [178, 203]	0.970	0.335	16.50	33.09
TokenFlow + PnP [45, 187]	0.970	0.327	10.56	28.54
VidToMe [93]	0.961	0.326	9.28	39.83
FateZero [140]	0.961	0.344	135.80	41.34
Optimized-FateZero	0.966	0.334	9.54	10.14
+ Object-Centric Diffusion	0.967	0.331	**8.22**	**9.29**

Table 6.1: **Quantitative results in inversion-based pipelines:** Our method achieves significant speed-up compared to the baseline and other the state-of-the-art methods (either video or framewise), without sacrificing generation quality.

6.4.1 Inversion-based Video Editing

Following the benchmark in [140], we report a quantitative comparison based on 9 sequence-prompt pairs. We include inversion-based video models such as Tune-A-Video + DDIM [178, 203], TokenFlow + PnP [45, 187] and VidToMe [93] in the evaluation. We also show results for a frame-based editing model, namely Framewise Null + p2p [60, 123]. We finally report results of the Optimized-FateZero baseline, obtained by the off-the-shelf accelerations in Sec. 6.2.1.

Main results A qualitative comparison of OCD and state-of-the-art models is reported in Fig. 6.5, where our model enjoys the lowest latency, while ensuring a comparable editing quality. We report fidelity and latency measurements in Tab. 6.1, where OCD achieves remarkable latency gains of 10× w.r.t. FateZero, and proves significantly faster than other state-of-the-art methods with comparable fidelity metrics. Although Optimized-FateZero is also very fast, we observe via visual assessment that its editing quality can be suboptimal, even though highly localized geneation artifacts might be overlooked by fidelity metrics (for clear examples of this, we refer to the ablation in Fig. 6.7).

6.4.2 ControlNet-based Video Editing

We follow the evaluation benchmark presented in [243] for evaluating ControlNet-based algorithms. The comparison is again based on sequences from DAVIS, and

comprises 125 sequence-prompt pairs (detailed in the supplement), for which per-frame depth maps are extracted using [149] and used as control signal. Besides our main baseline ControlVideo [243], we include Text2Video-Zero [80] in the comparisons. Once again, we will also report the performances of off-the-shelf optimizations (Optimized-ControlVideo).

Main results Fig. 6.6 illustrates some visual examples. As the figure shows, our proposal shows a significant 6× speed-up over the baseline, obtaining a comparable generation quality. We also report a quantitative comparison of ControlNet-based methods in Tab. 6.2, observing similar outcomes. We notice that Text2Video-Zero is slightly faster than our method, due to its sparse instead of dense cross-frame attention. However, for the same reason it underperforms in terms of temporal-consistency among generated frames. We refer the reader to the supplementary material for additional qualitative comparison.

6.4.3 Analysis

Ablation study We ablate our main contributions within the FateZero base model. Specifically, we illustrate the impact of our proposals in Table 6.3, where we report Tem-con and latency as metrics, and in Fig. 6.7. Although off-the-shelf optimizations may grant reasonable speed-ups, we found that for extreme token reduction rates their editing results can suffer from generation artifacts. As the figure shows, the addition of Object-Centric ToMe can easily fix most defects on foreground regions by forcing ToMe to operate on other areas, a benefit that comes without major impacts on latency. Finally, Object-Centric Sampling enables significant latency savings performing most diffusion iterations on foreground areas only. Somewhat surprisingly, we observe that Object-Centric Sampling helps the generation of background areas too: this is due to the fact that, as we run fewer denoising steps, its representations remain closer to the ones of the original video (resulting from inversion).

Impact of saliency mask We compare OCD by using saliency masks from three sources: i) human-labeled masks, ii) predictions from a state-of-the-art segmentation models (Grounded-SAM [156]), and iii) cross-attention maps w.r.t. text prompts (available within the model itself). We did not observe meaningful differences either in generations (see Fig. 6.8 for an example) or in aggregated fidelity metrics (Tab. 6.4), suggesting that OCD is robust to the source of saliency.

Object-Centric Sampling at different object sizes The impact on latency of the proposed Object-Centric Sampling depends on the size of the foreground areas:

Model	Tem-con ↑	Cl-score ↑	Latency (s) ↓
Text2Video-Zero [80]	0.960	0.317	**23.46**
ControlVideo [243]	0.972	0.318	152.64
Optimized-ControlVideo	0.978	0.314	31.12
+ Object-Centric Diffusion	0.977	0.313	25.21

Table 6.2 & Figure 6.6: **Comparison with ControlNet-based pipelines:** we report both quantitative and qualitative results based on depth conditioning. With comparable generation quality, our method achieves a 6× speed-up compared to the baseline.

in the presence of very small edits it can allow significant savings, whereas for prominent objects its editing cost would be only mildly-affected. We study this behavior in Tab. 6.5, where we divide 18 sequence-prompt pairs in three sets of 6 pairs each, depending on the number of foreground pixels (or foreground ratio Δ): Large ($\Delta > 20\%$), Medium ($\Delta \in [10, 20]\%$) and Small ($\Delta < 10\%$). As we show in Tab. 6.5, in the absence of Object-Centric Sampling, the latency of video editing

Component	Tem-con ↑	Latency (s) ↓	
		Inversion	Generation
Optimized-FateZero	0.966	8.90	10.38
+ Object-Centric ToMe	0.967	9.05	10.54
+ Object-Centric Sampling	0.966	**7.59**	**9.58**

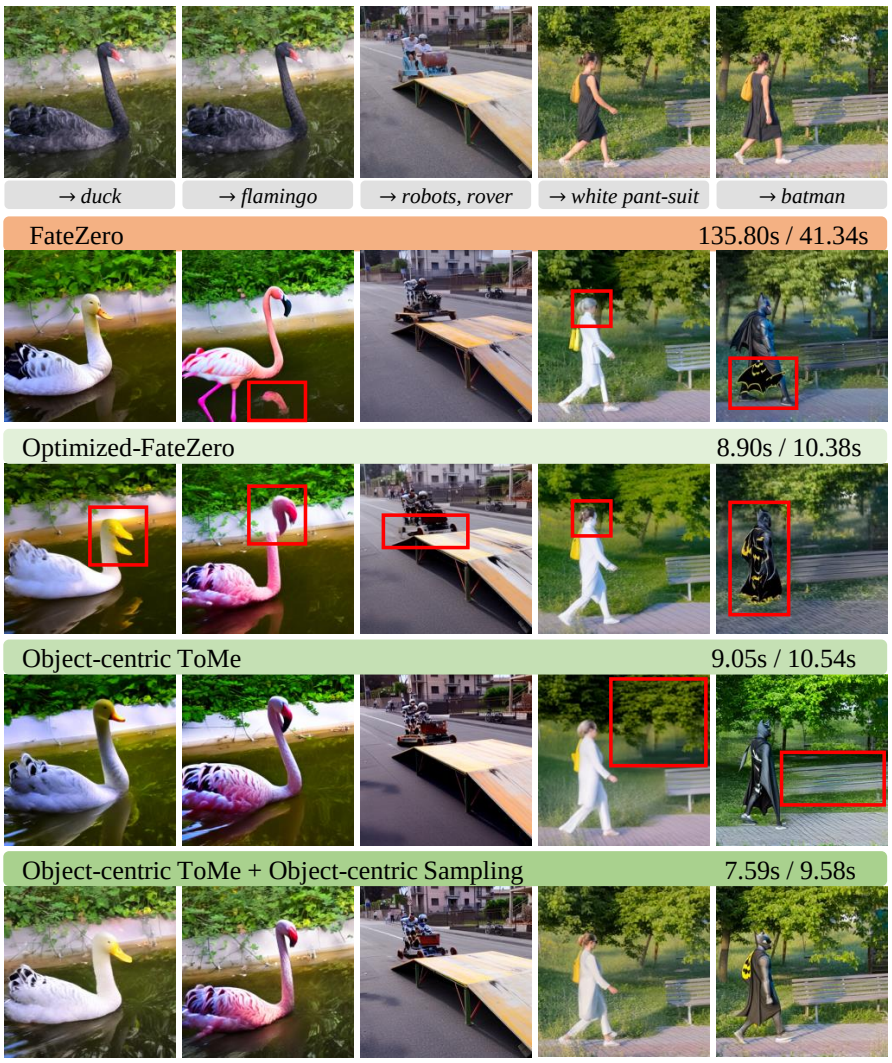

Table 6.3 & Figure 6.7: **Ablation of OCD components:** Object-Centric ToMe improves temporal-consistency and fidelity without sacrificing latency. Object-Centric Sampling further improves latency. Artifacts are highlighted in red.

Saliency mask	Tem-con ↑	Cl-score ↑
GT mask	0.967	0.331
Grounded SAM [156]	0.967	0.329
Cross-attention maps	0.966	0.329

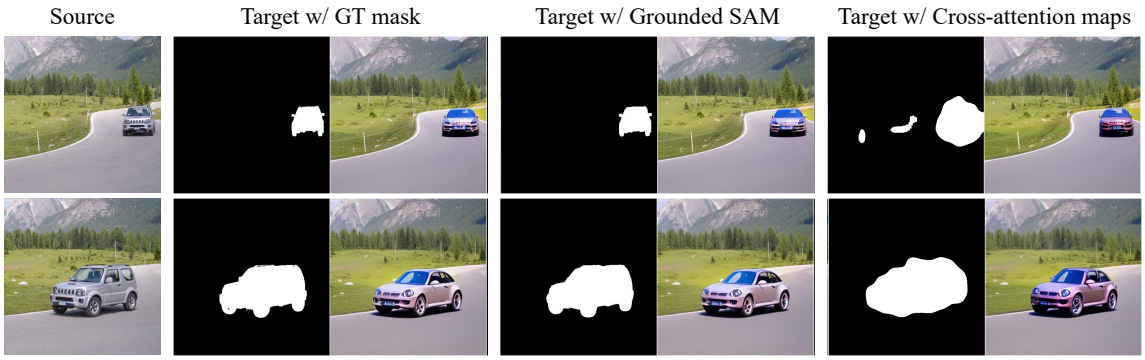

Table 6.4 & Figure 6.8: **Editing quality with different saliency masks:** Even with coarser masks OCD generates faithful edits, showing its resiliency to less optimal saliency sources.

does not depend on the size of the foreground objects. However, although we remark that our proposal reduces latency for all object sizes, we observe smaller gains on large objects, as the foreground area covers most of the frame resolution, matching Object-Centric Sampling with a regular diffusion process operating on the whole frame. Savings gradually improve as we go towards medium and small sized objects, showing additional speed-ups of 1.3× and 1.8× respectively.

Limitations We highlight some potential limitations of our model to be tackled by future work. Although Object-Centric Diffusion represent valuable ideas in general, they are particularly effective for local changes to some peculiar objects, and are slightly less-suitable for global editing (*e.g.* changing the style/textures of a video altogether). Moreover, as most zero-shot methods, in order to achieve the best trade-off between fidelity and latency, our framework still requires to search the best hyperparameters per-sequence.

6.5 Conclusion

In this paper, we introduced solutions for speeding up diffusion-based video editing. In this respect, we first presented an analysis of sources of latency in

Object size	Avg. #pixel (fg. ratio Δ)	w/o Obj-Cen. Sampling			w/ Obj-Cen. Sampling		
		Tem-con ↑	Latency (s) ↓		Tem-con ↑	Latency (s) ↓	
			Inversion	Generation		Inversion	Generation
Large	53.5k (20.4%)	0.976	12.70	11.08	0.967	9.77 (2.93↓)	9.65 (1.43↓)
Medium	35.2k (13.4%)	0.956	13.09	11.08	0.953	9.97 (3.12↓)	8.82 (2.26↓)
Small	17.7k (6.7%)	0.953	13.02	10.77	0.948	**7.25** (5.77↓)	**6.22** (4.55↓)

Table 6.5: **Impact of Object-Centric Sampling at different object sizes.** We achieve more latency savings with smaller foreground objects without sacrificing the generation quality. Latency reductions are shown in gray.

inversion-based models, and we identified and adopted some off-the-shelf techniques such as fast sampling and Token Merging that, when properly modified for the task, bring significant cost reduction with only slight quality degradation. Furthermore, motivated by the fact that video editing typically requires modifications to specific objects, we introduce Object-Centric Diffusion, comprising techniques for *i)* encouraging the merging of tokens in background regions and *ii)* limiting most of the diffusion sampling steps on foreground areas. Our solutions, which fix generation artifacts on foreground objects and further reduce editing latency, are validated on inversion-based and ControlNet-based models, by achieving 10× and 6× faster edits for comparable quality.

6.6 Appendix

6.6.1 Additional discussion

Off-the-shelf optimizations of ToMe

Pairing token locations from inversion Many inversion-based editing pipelines rely on sharing attention maps between inversion and generation stages (*e.g.* FateZero [140], Plug-and-Play [187]). As such, when applying ToMe [14, 15], it is important that locations of destination (*dst*) and unmerged (*unm*) tokens are paired in the two stages, at each corresponding attention layer and diffusion step. If that is not the case, tokens or attention maps coming from inversion are not compatible with the ones available at generation time. In practice, we compute which tokens to be merged during inversion, and merge the tokens at the same locations in generation attention maps. By doing so, we make sure the tokens that remain after merging correspond to the same locations (or, token indices), and

hence, the attention maps from inversion and generation can rightly be fused. We found this strategy to be of primary importance, as testified by Fig. 3 (d-e) in the main paper.

Re-sampling destination tokens per-frame ToMe for stable diffusion [14] samples *dst* tokens randomly in a single image. When extending this to multiple frames, we initially sample the same random locations in each frame, finding this strategy to be sub-optimal. Instead, if we re-sample different random locations in each frame (or, in each temporal window in our spatio-temporal implementation), it allows us to preserve different information in each frame (or, window) after merging. We found this to be beneficial, especially at higher merging rates (*e.g.* see Fig. 3 (e to f) in the main paper).

How to search for destination match In the original ToMe for stable diffusion [14], for each source (*src*) token, we search its corresponding match within a pool of *dst* tokens coming from the full spatial extent of the given image ($H \times W$). The naive transposition of this strategy to our video use-case allows, for any *src* token, its candidate match to be searched within a pool of *dst* tokens coming from the full spatio-temporal extent of the video ($T \times H \times W$). We find that this strategy for searching *dst* match, named hereby *Temporally-Global Search*, can lead to generation artifacts. Differently, we consider restricting the temporal extent of the *dst* pool to be the same temporal-window ($s_t \times H \times W$) as the *src* token, as in our *Temporally-Windowed Search*. As shown in Fig. 6.9, the latter gives better reconstructions in general, whilst allowing more flexibility to control where merges are happening, temporally. This way, the user can also better trade-off the temporal-redundancy, smoothness and consistency by controlling the spatio-temporal window size.

Merging queries, keys or values? In our early experiments, we consider applying ToMe to all queries (with unmerging, as in [14]), keys and values. We however find that, with extreme reduction rates, merging queries can easily break the reconstructions. As such, we limit ToMe to operate on keys and values only. We also observe that in dense cross-frame attention modules, merging queries only provide a slight latency reduction.

Capped merging in low-res UNet stages As observed in [14], the high resolution UNet [159] stages are the most expensive in terms of self-attention (or, cross-frame attention) modules, and the ones that can benefit the most by applying ToMe. Contrarily to the original formulation which does not optimize low-resolution layers, we do apply ToMe in all layers as we observe it has a meaningful impact on latency. We however cap the minimum #tokens preserved after merging in low-resolution layers, in order to avoid degenerate bottlenecks (*e.g.* collapsing to a

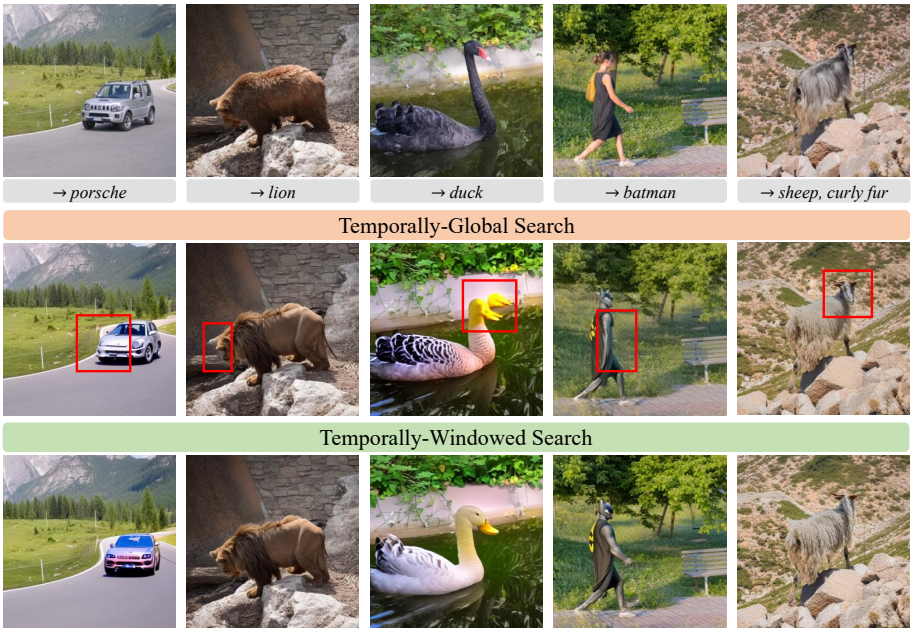

Figure 6.9: **Qualitative ablation on searching *dst* match** Each *src* token may search for its corresponding match within a pool of *dst* tokens. This pool can be comprised of either the whole spatio-temporal extent (i.e., all frames) as in *Temporally-Global Search*, or only the same temporal window as the corresponding *src* token as in *Temporally-Windowed Search*. Among these two strategies, the latter allows more flexibility, providing more-consistent generations with better fidelity.

single representation). Specifically, we maintain at-least 4 and 16 tokens per-frame after merging at 8×8 and 16×16 resolutions, respectively.

Benchmark settings

Evaluation metrics We consider two metrics for quantitative evaluation: CLIP-score and Temporal-consistency, similar to prior work [140, 243]. CLIP-score is computed as the cosine similarity between CLIP [146] visual embedding of each frame and CLIP text embedding of the corresponding edit prompt, aggregated over all frames and sequences. It measures the semantic fidelity of the generated video. Temporal-consistency is computed as the cosine similarity between the CLIP visual embeddings of each consecutive pairs of frames, aggregated over all pairs and sequences. It conveys the visual quality of the generated video, measuring how temporally coherent frames are. We highlight that, despite their

	Sequence	Source prompt	Target prompts
Large object	blackswan	a black swan with a red beak swimming in a river near a wall and bushes.	• a white duck with a yellow beak swimming in a river near a wall and bushes. • a pink flamingo with a red beak walking in a river near a wall and bushes. • a Swarovski crystal swan with a red beak swimming in a river near a wall and bushes.
	bear	a brown bear walking on the rock against a wall.	• a red tiger walking on the rock against a wall. • a yellow leopard walking on the rock against a wall.
Medium object	breakdance	a man wearing brown tshirt and jeans doing a breakdance flare on gravel.	• a woman with long-hair wearing green-sweater and jeans doing a breakdance flare on gravel. • a spiderman wearing red-blue spidersuit doing a breakdance flare on gravel. • a chimpanzee wearing a black jeans doing a breakdance flare on gravel.
	boat	a white color metal boat cruising in a lake near coast.	• a heavily-rusted metal boat cruising in a lake near coast. • a light-brown color wooden boat cruising in a lake near coast.
	car-turn	a silver jeep driving down a curvy road in the countryside.	• a Porsche car driving down a curvy road in the countryside.
Small object	mallard	a brown mallard running on grass land close to a lake.	• a white duck running on grass land close to a lake. • a golden chicken running on grass land close to a lake. • a gray goose running on grass land close to a lake.
	lucia	a woman wearing a black dress with yellow handbag walking on a pavement.	• a woman wearing a white pant-suit with yellow handbag walking on a pavement. • a woman wearing a black dress and a hat with red handbag walking on a pavement. • a batman wearing a black bat-suit walking on a pavement.
	soapbox	two men driving a soapbox over a ramp.	• two robots driving a mars-rover over a ramp.

Table 6.1: **Sequence-prompt pairs used to evaluate inversion-based pipelines**: All sequences here are from DAVIS [137]. We show the sequence-prompt pairs used to evaluate our Object-Centric Sampling, separately for Large, Medium and Small objects. [continued...]

use is due in fair comparisons due to their popularity, both these fidelity metrics are far from perfect. For instance, we find the CLIP score to be sensitive to global semantic discrepancies, yet it often overlooks generation artifacts and smaller

	Sequence	Source prompt	Target prompts
FateZero [140] Benchmark	blackswan	a black swan with a red beak swimming in a river near a wall and bushes.	• a white duck with a yellow beak swimming in a river near a wall and bushes. • a pink flamingo with a red beak walking in a river near a wall and bushes. • a Swarovski crystal swan with a red beak swimming in a river near a wall and bushes. • cartoon photo of a black swan with a red beak swimming in a river near a wall and bushes.
	car-turn	a silver jeep driving down a curvy road in the countryside.	• a Porsche car driving down a curvy road in the countryside. • watercolor painting of a silver jeep driving down a curvy road in the countryside.
	kite-surf	a man with round helmet surfing on a white wave in blue ocean with a rope.	• a man with round helmet surfing on a white wave in blue ocean with a rope in the Ukiyo-e style painting.
	train (in-the-wild)	a train traveling down tracks next to a forest filled with trees and flowers and a man on the side of the track.	• a train traveling down tracks next to a forest filled with trees and flowers and a man on the side of the track in Makoto Shinkai style.
	rabbit (in-the-wild)	a rabbit is eating a watermelon.	• pokemon cartoon of a rabbit is eating a watermelon.

Table 6.1: **Sequence-prompt pairs used to evaluate inversion-based pipelines**: Most sequences here are from DAVIS [137], except for a few in-the-wild videos used in [140]. The Benchmark pairs correspond to the original FateZero [140] quantitative evaluation setting.

pixel-level details. Furthermore, Temporal-Consistency can be simply exploited by a fake video repeating a frame over time. For these reasons, extensive visual comparisons are still required to assess different models, and future research should be encouraged towards more informative quantitative protocols for video editing.

Sequence-prompt pairs We present the sequence-prompt pairs considered in our evaluation of inversion-based pipelines in Table 6.1. Most sequences here are from DAVIS [137] dataset, with the exception of a few in-the-wild videos introduced in [140]. The Benchmark setting corresponds to the original quantitative evaluation of FateZero [140], which includes 9 sequence-prompt pairs. We also present the sequence-prompt pairs used to evaluate our Object-Centric Sampling (see Table 5 in the main paper), categorized based on the foreground object size: Large, Medium and Small. In Table 6.2, we show the 125 sequence-prompt pairs used in ControlNet-based pipelines, provided by the authors of ControlVideo [243].

Sequence	Source prompt	Target prompts
blackswan	a black swan moving on the lake.	• A black swan moving on the lake • A white swan moving on the lake. • A white swan moving on the lake, cartoon style. • A crochet black swan swims in a pond with rocks and vegetation. • A yellow duck moving on the river, anime style.
boat	a boat moves in the river.	• A sleek boat glides effortlessly through the shimmering river, van gogh style. • A majestic boat sails gracefully down the winding river. • A colorful boat drifts leisurely along the peaceful river. • A speedy boat races furiously across the raging river. • A rustic boat bobs gently on the calm and tranquil river.
breakdance-flare	a man dances on the road.	• A young man elegantly dances on the deserted road under the starry night sky. • The handsome man dances enthusiastically on the bumpy dirt road, kicking up dust as he moves. • A man gracefully dances on the winding road, surrounded by the picturesque mountain scenery. • The athletic man dances energetically on the long and straight road, his sweat glistening under the bright sun. • The talented man dances flawlessly on the busy city road, attracting a crowd of mesmerized onlookers.
bus	a bus moves on the street.	• A big red bus swiftly maneuvers through the crowded city streets. • A sleek silver bus gracefully glides down the busy urban avenue. • A colorful double-decker bus boldly navigates through the bustling downtown district. • A vintage yellow bus leisurely rolls down the narrow suburban road. • A modern electric bus silently travels along the winding coastal highway.
camel	a camel walks on the desert.	• A majestic camel gracefully strides across the scorching desert sands. • A lone camel strolls leisurely through the vast, arid expanse of the desert. • A humpbacked camel plods methodically across the barren and unforgiving desert terrain. • A magnificent camel marches stoically through the seemingly endless desert wilderness. • A weathered camel saunters across the sun-scorched dunes of the desert, its gaze fixed on the horizon.
car-roundabout	a jeep turns on a road.	• A shiny red jeep smoothly turns on a narrow, winding road in the mountains. • A rusty old jeep suddenly turns on a bumpy, unpaved road in the countryside. • A sleek black jeep swiftly turns on a deserted, dusty road in the desert. • A modified green jeep expertly turns on a steep, rocky road in the forest. • A gigantic yellow jeep slowly turns on a wide, smooth road in the city.

Table 6.2: **Sequence-prompt pairs used to evaluate ControlNet-based pipelines**: All sequences are from DAVIS [137]. These pairs correspond to the original ControlVideo [243] quantitative evaluation setting. [continued...]

Sequence	Source prompt	Target prompts
car-shadow	a car moves to a building.	• A sleek black car swiftly glides towards a towering skyscraper. • A shiny silver vehicle gracefully maneuvers towards a modern glass building. • A vintage red car leisurely drives towards an abandoned brick edifice. • A luxurious white car elegantly approaches a stately colonial mansion. • A rusty blue car slowly crawls towards a dilapidated concrete structure.
car-turn	a jeep on a forest road.	• A shiny silver jeep was maneuvering through the dense forest, kicking up dirt and leaves in its wake. • A dusty old jeep was making its way down the winding forest road, creaking and groaning with each bump and turn. • A sleek black jeep was speeding along the narrow forest road, dodging trees and rocks with ease. • A massive green jeep was lumbering down the rugged forest road, its powerful engine growling as it tackled the steep incline. • A rusty red jeep was bouncing along the bumpy forest road, its tires kicking up mud and gravel as it went.
cows	a cow walks on the grass.	• A spotted cow leisurely grazes on the lush, emerald-green grass. • A contented cow ambles across the dewy, verdant pasture. • A brown cow serenely strolls through the sun-kissed, rolling hills. • A beautiful cow saunters through the vibrant, blooming meadow. • A gentle cow leisurely walks on the soft, velvety green grass, enjoying the warm sunshine.
dog	a dog walks on the ground.	• A fluffy brown dog leisurely strolls on the grassy field. • A scruffy little dog energetically trots on the sandy beach. • A majestic black dog gracefully paces on the polished marble floor. • A playful spotted dog joyfully skips on the leaf-covered path. • A curious golden dog curiously wanders on the rocky mountain trail.
elephant	an elephant walks on the ground.	• A massive elephant strides gracefully across the dusty savannah. • A majestic elephant strolls leisurely along the lush green fields. • A mighty elephant marches steadily through the rugged terrain. • A gentle elephant ambles peacefully through the tranquil forest. • A regal elephant parades elegantly down the bustling city street.
flamingo	a flamingo wanders in the water.	• A graceful pink flamingo leisurely wanders in the cool and refreshing water, its slender legs elegantly stepping on the soft sand. • A vibrant flamingo casually wanders in the clear and sparkling water, its majestic wings spread wide in the sunshine. • A charming flamingo gracefully wanders in the calm and serene water, its delicate neck curving into an elegant shape.

Table 6.2: **Sequence-prompt pairs used to evaluate ControlNet-based pipelines**: All sequences are from DAVIS [137]. These pairs correspond to the original ControlVideo [243] quantitative evaluation setting. [continued...]

Sequence	Source prompt	Target prompts
flamingo	a flamingo wanders in the water.	• A stunning flamingo leisurely wanders in the turquoise and tranquil water, its radiant pink feathers reflecting the shimmering light. • A magnificent flamingo elegantly wanders in the sparkling and crystal-clear water, its striking plumage shining brightly in the sun.
gold-fish	golden fishers swim in the water.	• Majestic golden fishers glide gracefully in the crystal-clear waters. • Brilliant golden fishers swim serenely in the shimmering blue depths. • Glittering golden fishers dance playfully in the glistening aquamarine waves. • Gleaming golden fishers float leisurely in the peaceful turquoise pools. • Radiant golden fishers meander lazily in the tranquil emerald streams.
hike	a man hikes on a mountain.	• A rugged man is trekking up a steep and rocky mountain trail. • A fit man is leisurely hiking through a lush and verdant forest. • A daring man is scaling a treacherous and jagged peak in the alpine wilderness. • A seasoned man is exploring a remote and rugged canyon deep in the desert. • A determined man is trudging up a snowy and icy mountain slope, braving the biting cold and fierce winds.
hockey	a player is playing hockey on the ground.	• A skilled player is furiously playing ice hockey on the smooth, glistening rink. • A young, agile player is energetically playing field hockey on the lush, green grass. • An experienced player is gracefully playing roller hockey on the sleek, polished pavement. • A determined player is passionately playing street hockey on the gritty, urban asphalt. • A talented player is confidently playing air hockey on the fast-paced, neon-lit table.
kite-surf	a man is surfing on the sea.	• A muscular man is expertly surfing the gigantic waves of the Pacific Ocean. • A handsome man is gracefully surfing on the crystal clear waters of the Caribbean Sea. • A daring man is fearlessly surfing through the dangerous, choppy waters of the Atlantic Ocean. • An athletic man is skillfully surfing on the wild and untamed waves of the Indian Ocean. • A young man is confidently surfing on the smooth, peaceful waters of a serene lake.
lab-coat	three women stands on the lawn.	• Three stunning women are standing elegantly on the lush green lawn, chatting and laughing. • Three young and vibrant women are standing proudly on the well-manicured lawn, enjoying the sunshine. • Three fashionable women in colorful dresses are standing gracefully on the emerald green lawn, taking selfies. • Three confident women with radiant smiles are standing tall on the soft, green lawn, enjoying the fresh air. • Three beautiful women, each dressed in their own unique style, are standing on the lush and verdant lawn, admiring the scenery.

Table 6.2: **Sequence-prompt pairs used to evaluate ControlNet-based pipelines**: All sequences are from DAVIS [137]. These pairs correspond to the original ControlVideo [243] quantitative evaluation setting. [continued...]

Sequence	Source prompt	Target prompts
longboard	a man is playing skateboard on the alley.	• A young man is skillfully skateboarding on the busy city street, weaving in and out of the crowds with ease. • An experienced skateboarder is fearlessly gliding down a steep, curvy road on his board, executing impressive tricks along the way. • A daring skater is performing gravity-defying flips and spins on his board, effortlessly navigating through a challenging skatepark course. • A talented skateboarder is carving up the smooth pavement of an empty parking lot, creating beautiful patterns with his board and body. • A passionate skater is practicing his moves on a quiet neighborhood street, with the sound of his board echoing through the peaceful surroundings.
mallard-water	a mallard swims on the water.	• A vibrant mallard glides gracefully on the shimmering water. • A beautiful mallard paddles through the calm, blue water. • A majestic mallard swims elegantly on the tranquil lake. • A striking mallard floats effortlessly on the sparkling pond. • A colorful mallard glides smoothly on the rippling surface of the water.
mbike-trick	a man riding motorbike.	• A young man riding a sleek, black motorbike through the winding mountain roads. • An experienced man effortlessly riding a powerful, red motorbike on the open highway. • A daring man performing gravity-defying stunts on a high-speed, blue motorbike in an empty parking lot. • A confident man cruising on a vintage, yellow motorbike along the picturesque coastal roads. • A rugged man maneuvering a heavy, dusty motorbike through the rugged terrain of a desert.
rhino	a rhino walks on the rocks.	• A massive rhino strides confidently across the jagged rocks. • A majestic rhino gracefully navigates the rugged terrain of the rocky landscape. • A powerful rhino marches steadily over the rough and rocky ground. • A colossal rhino plods steadily through the craggy rocks, undeterred by the challenging terrain. • A sturdy rhino confidently traverses the treacherous rocks with ease.
surf	a sailing boat moves on the sea.	• A graceful sailing boat glides smoothly over the tranquil sea. • A sleek sailing boat cuts through the shimmering sea with ease. • A majestic sailing boat cruises along the vast, azure sea. • A vintage sailing boat bobs gently on the calm, turquoise sea. • A speedy sailing boat races across the glistening, open sea.

Table 6.2: **Sequence-prompt pairs used to evaluate ControlNet-based pipelines**: All sequences are from DAVIS [137]. These pairs correspond to the original ControlVideo [243] quantitative evaluation setting. [continued...]

Sequence	Source prompt	Target prompts
swing	a girl is playing on the swings.	• A young girl with pigtails is joyfully swinging on the colorful swings in the playground. • The little girl, giggling uncontrollably, is happily playing on the old-fashioned wooden swings. • A blonde girl with a big smile on her face is energetically playing on the swings in the park. • The girl, wearing a flowery dress, is gracefully swaying back and forth on the swings, enjoying the warm breeze. • A cute little girl, dressed in a red coat, is playfully swinging on the swings, her hair flying in the wind.
tennis	a man is playing tennis.	• The skilled man is effortlessly playing tennis on the court. • A focused man is gracefully playing a game of tennis. • A fit and agile man is playing tennis with precision and finesse. • A competitive man is relentlessly playing tennis with his opponent. • The enthusiastic man is eagerly playing a game of tennis, sweat pouring down his face.
walking	a selfie of walking man.	• A stylish young man takes a selfie while strutting confidently down the busy city street. • An energetic man captures a selfie mid-walk, showcasing his adventurous spirit. • A happy-go-lucky man snaps a selfie as he leisurely strolls through the park, enjoying the sunny day. • A determined man takes a selfie while briskly walking towards his destination, never breaking stride. • A carefree man captures a selfie while wandering aimlessly through the vibrant cityscape, taking in all the sights and sounds.

Table 6.2: **Sequence-prompt pairs used to evaluate ControlNet-based pipelines**: All sequences are from DAVIS [137]. These pairs correspond to the original ControlVideo [243] quantitative evaluation setting.

6.6.2 Additional results

Qualitative comparisons

We show additional qualitative comparisons for inversion-based pipelines in Fig. 6.10. Here, we mainly focus on shape editing, and present multiple edited frames of each sequence using FateZero [140], Tune-A-Video [203], TokenFlow [45] and Frame SDEdit [116]. Tune-A-Video requires 1-shot finetuning on a given sequence, whereas TokenFlow and Frame SDEdit are based on stable diffusion [158] checkpoints. FateZero and our implementation rely on Tune-A-Video checkpoints for shape editing, without needing any further finetuning for their respective proposed improvements. Frame SDEdit shows no consistency among frames, being an image editing pipeline. Among video editing pipelines, ours show the best fidelity and temporal-consistency, while also generating outputs faster (see latency measurements given in Table 1 and Fig. 5 in the main paper). Notably, thanks

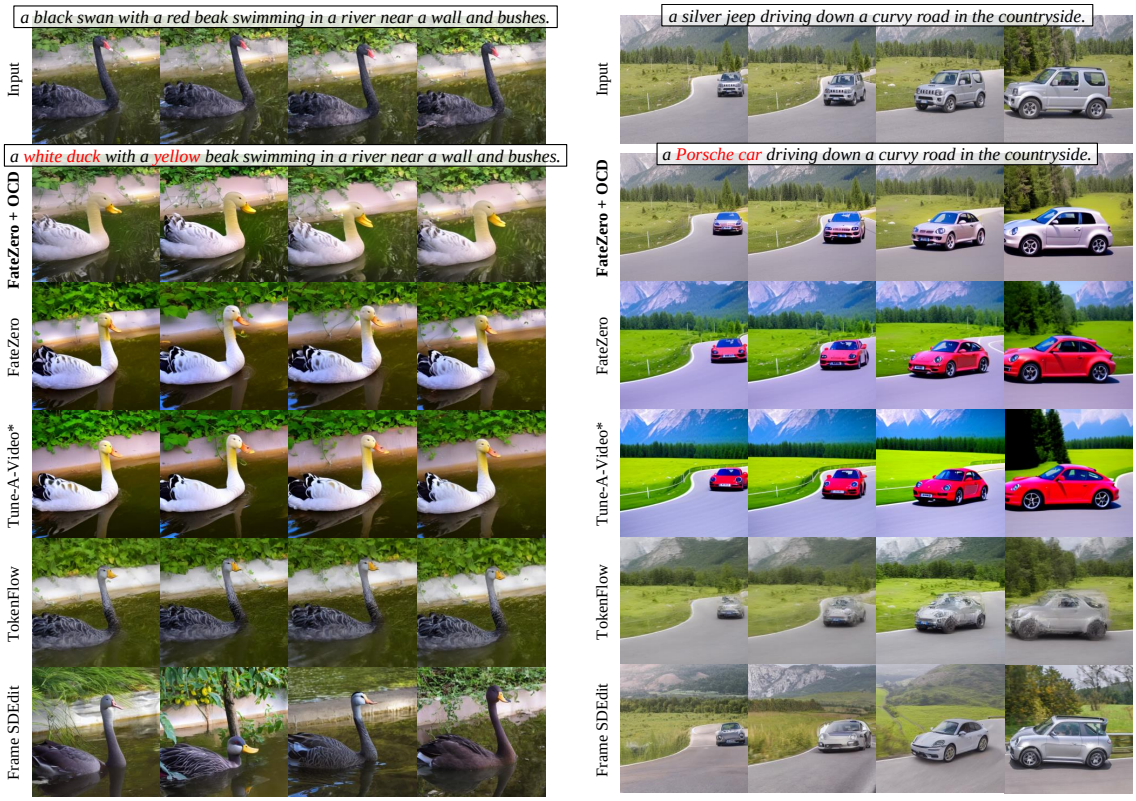

Figure 6.10: **Qualitative comparison on *blackswan* and *car-turn* sequences [137]:** We show shape editing results of our method (Optimized-FateZero + OCD), in comparison with FateZero [140], Tune-A-Video [203], TokenFlow [45] and SDEdit [116]. Our results show better semantic quality (*e.g.* alignement with target prompt) and visual fidelity (*e.g.* temporal consistency, faithful background), while also being more efficient (Table 1 in the main paper). Best viewed zoomed-in.

to Object-Centric Sampling, our pipeline gives more-faithful background reconstructions, as such regions are expected to be un-edited based on the given shape editing prompts.

In Fig. 6.12, we show additional qualitative comparisons for ControlNet-based pipelines sucha as ControlVideo [243] and Text2Video-Zero [80]. Here, all methods are conditioned on Depth-maps, while using SD [158] checkpoints without further finetuning. OCD shows comparable performance with its baseline ControlVideo, while being significantly faster (see latency measurements given in Table 2 in the main paper). It is also more temporally-consistent compared to Text2Video-Zero which uses sparse instead of dense cross-frame attention, while having comparable

Figure 6.11: **Qualitative comparison on *breakdance-flare* and *lucia* sequences [137]:** We show shape editing results of our method (Optimized-FateZero + OCD), in comparison with FateZero [140], Tune-A-Video [203], TokenFlow [45] and SDEdit [116]. Our results show better semantic quality (*e.g.* alignement with target prompt) and visual fidelity (*e.g.* temporal consistency, faithful background), while also being more efficient (Table 1 in the main paper). Best viewed zoomed-in.

latency.

Quantitative comparisons

Other baselines for latency reduction We discuss simple baselines for reducing latency in Table 6.3 and Table 6.4. We report both fidelity (Temporal-consistency, CLIP-score) and latency (inversion, generation, UNet [159] time). Here, UNet time corresponds to just running UNet inference without any additional overheads (*e.g.* memory access), which we use to highlight the cost of such overheads. For ControlVideo [243] baselines, we show results with either Depth or Canny-edge conditioning.

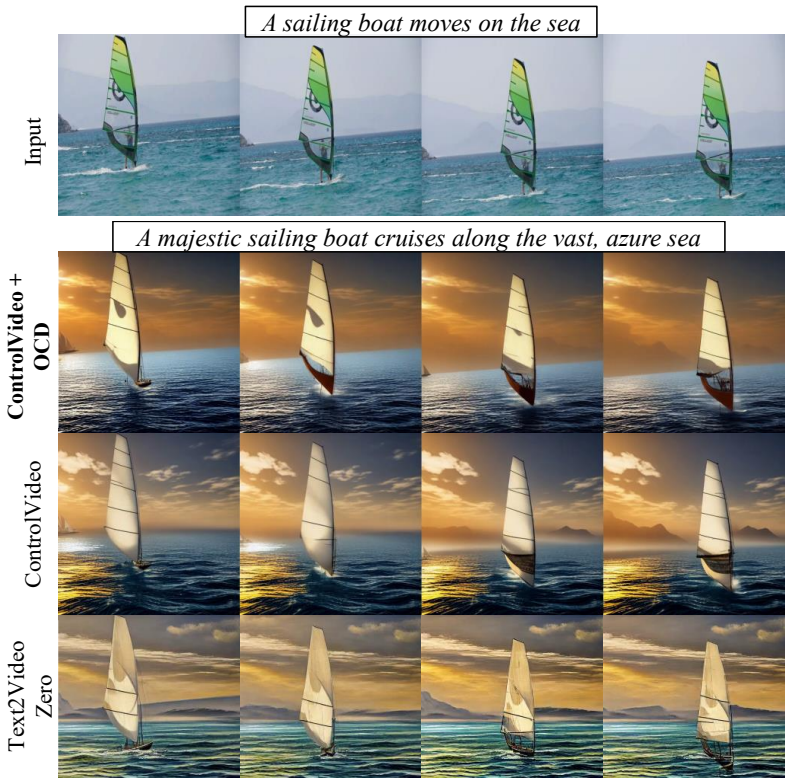

Figure 6.12: **Qualitative comparison on *surf* sequence [137]:** We show shape editing results of our method (Optimized-ControlVideo + OCD), in comparison with ControlVideo [243] and Text2Video-Zero [80]. All methods use Depth conditioning. Our results show comparable quality with baseline ControlVideo while being significantly faster (Tab 2 in the main paper), and better temporal consistency compared to Text2Video-Zero. Best viewed zoomed-in.

In both inversion-based and ControlNet-based settings, we devise our optimized-baselines by reducing diffusion steps 50→20 and applying token merging, which give reasonable reconstructions. This is our default starting point for implementing OCD. Going further, we also consider diff. steps=5, which fails to retain details in reconstructions. Instead of token merging, we can also apply pooling strategies on key-value tokens. Despite giving similar speed-ups, these result in sub-par performance compared to ToMe, especially in shape editing (although not always captured in quantitative numbers). In ControlVideo setup, we can choose to do merging on both UNet and ControlNet [242] models, resulting in further speed-ups with a minimal drop in fidelity. We further observe that we

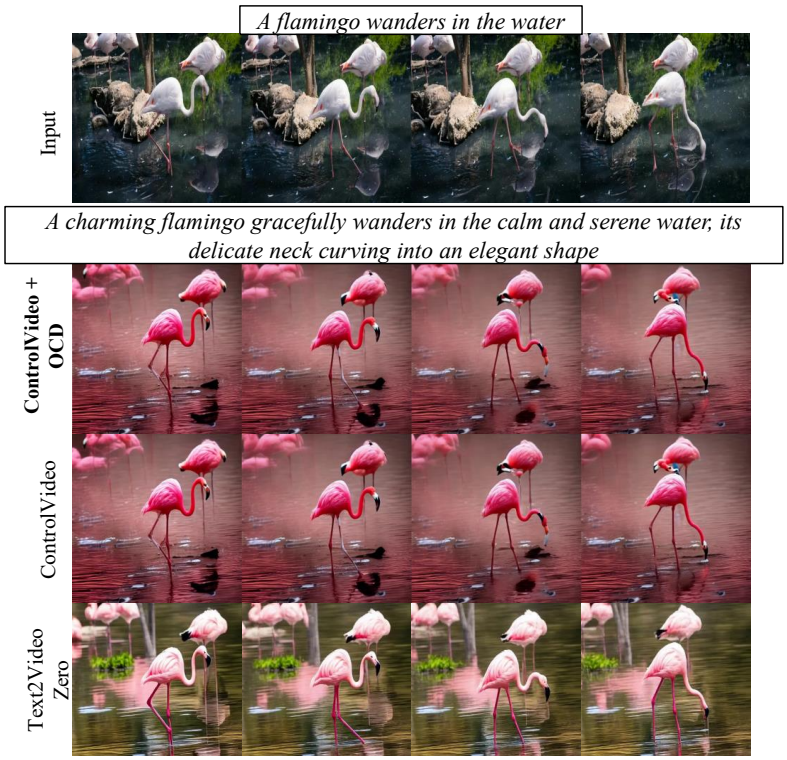

Figure 6.13: **Qualitative comparison on *flamingo* sequence [137]:** We show shape editing results of our method (Optimized-ControlVideo + OCD), in comparison with ControlVideo [243] and Text2Video-Zero [80]. All methods use Depth conditioning. Our results show comparable quality with baseline ControlVideo while being significantly faster (Tab 2 in the main paper), and better temporal consistency compared to Text2Video-Zero. Best viewed zoomed-in.

can re-use the same control signal for multiple diffusion steps, allowing us to run ControlNet at a reduced rate (Reduced/Single inference in Table 6.4).

Cost of memory access vs. computations Inversion-based editing pipelines rely on guidance from the inversion process during generation (*e.g.* based on latents [187] or attention maps [140]). When running inference, such features need to be stored (which may include additional GPU→RAM transfer), accessed and reused. This cost can be considerable, especially for resource-constrained hardware. This cost measured in latency, is shown in Table 6.3, as the difference between inversion and UNet times. Alternatively, it can also be seen as the storage requirement as given in Table 6.5. On FateZero [140], we observe that the storage cost is indeed

Model	CLIP metrics ↑		Latency (s) ↓		
	Tem-con	Cl-score	Inv	Gen	UNet
FateZero [140]	0.961	0.344	135.80	41.34	20.63
+ diff. steps=5	0.968	0.306	14.84	4.98	2.17
+ diff. steps=20	0.961	0.341	61.82	18.41	8.03
+ avg. pool (4,4)	0.958	0.335	9.91	11.80	7.08
+ max pool (4,4)	0.959	0.275	9.96	11.91	6.91
Optimzed-FateZero	0.966	0.334	9.54	10.14	7.79
+ OCD	0.967	0.331	8.22	9.29	6.38

Table 6.3: **Additional FateZero [140] baselines:** We report CLIP metrics of fidelity (Temporal-consistency, CLIP-score) and latency (inversion, generation, UNet [159] time). The difference between inversion and UNet time corresponds to other overheads, dominated by memory access. Fewer diffusion steps (*e.g.* 5) and pooling operations can also gain significant speed-ups, but break reconstructions (not always visible in fidelity metrics).

Model	CLIP metrics ↑		Latency (s) ↓	
	Tem-con	Cl-score	Gen	UNet
ControlVideo [243]	0.972 (0.968)	0.318 (0.308)	152.64	137.68
+ diff. steps=5	0.978 (0.971)	0.309 (0.295)	19.58	13.58
+ diff. steps=20	0.978 (0.971)	0.316 (0.304)	64.61	54.83
+ avg.pool (2,2)	0.977 (0.968)	0.309 (0.295)	30.53	20.56
+ max.pool (2,2)	0.972 (0.973)	0.225 (0.212)	30.32	20.53
Optimized-ControlVideo	0.978 (0.972)	0.314 (0.303)	31.12	21.42
+ OCD	0.977 (0.967)	0.313 (0.302)	25.21	15.61
+ OCD (UNet, ControlNet)	0.976 (0.969)	0.306 (0.297)	25.13	15.41
+ ControlNet Red. Inf.	0.977 (0.968)	0.313 (0.301)	23.62	15.47
+ ControlNet Sin. Inf.	0.973 (0.964)	0.307 (0.293)	22.35	15.48

Table 6.4: **Additional ControlVideo [243] baselines:** We report CLIP metrics of fidelity (Temporal-consistency, CLIP-score) with either Depth or (Canny-edge) conditioning, and latency (generation, UNet [159] time). The difference between generation and UNet time corresponds to other overheads, dominated by ControlNet [242]. Fewer diffusion steps (*e.g.* 5) and pooling operations can also gain significant speed-ups, but break reconstructions (not always visible in fidelity metrics). We also observe that ControlNet inference need not be done at the same frequency as denoising, which can lead to further speed-ups.

Model	Disk-space (GB) ↓
FateZero	74.54
+ diff. steps=5	7.45
+ diff. steps=20	29.82
+ pool (4,4)	3.06
Optimized-FateZero	5.05
+ OCD	4.22

Table 6.5: **Memory requirement for attention maps**: In FateZero [140] setting, we show additional baselines and the corresponding storage requirements which directly affect the memory-access overhead. FateZero stores attention maps of all UNet [159] blocks for all diffusion steps. Our contributions help reduce this cost. It can potentially enable attention maps to be kept on GPU memory itself (w/o having to move between GPU and RAM), further improving latency. Each float is stored in 16bits.

Blending steps ratio (γ)	Latency (s) @ #tokens (Δ) ↓							
	64×64		48×48		32×32		16×16	
	Inv	Gen	Inv	Gen	Inv	Gen	Inv	Gen
1.00	12.31	10.39	-	-	-	-	-	-
0.50	-	-	9.56	8.67	9.05	7.60	8.25	7.01
0.25	-	-	8.03	7.94	7.11	6.00	5.90	4.96
0.05	-	-	6.82	7.17	5.85	4.99	4.43	3.80

Table 6.6: **Control experiment on Object-Centric Sampling:** We evaluate the latency savings at different *hypothetical* object sizes (Δ) and blending step ratios (γ). The baseline is with $\Delta = 64 \times 64$ and $\gamma = 1.0$ (with total 20 diffusion steps). We can get the most savings at a smaller object size and blending step ratio. It is worth noting that this control experiment does not correspond to actual sequence-prompt pairs, and is just intended to give the reader an idea about expected savings.

significant, and affects the latency more than the computations. With OCD, we directly reduce the cost for attention computation, storage and access.

Expected savings of Object-Centric Sampling We run a control experiment to observe the expected latency reductions when using our Object-Centric Sampling, at different object sizes (Δ) and Blending step ratios (γ), given in Table 6.6. Here, we consider hypothetical inputs, so that we can ablate the settings more-clearly.

The baseline sees inputs of size $\Delta = 64 \times 64$, and runs all diffusion steps at full-resolution ($\gamma = 1.0$). In our Object-Centric Sampling, we use $\gamma = 0.25$ by default, whereas Δ depends on objects in a given sequence. As expected, we can obtain the most savings with fewer blending steps and when foreground objects are smaller in size. A user may refer to this guide to get an idea about the expected savings.

Chapter 7

Conclusion and Future Work

In this thesis, we explored efficiency bottlenecks of video modeling pipelines (for both video understanding and generation), in the context of latency costs at inference and annotation costs during training. Across multiple proposals, we introduced remedies to such inefficiencies, focusing on challenging applications including diffusion-based video editing, activity detection and language-based video understanding.

In Chapter 3, we presented Coarse-Fine Networks, which is a two-stream architecture to combine temporally Coarse representations with Fine representations. We introduced the approach of temporal Grid Pooling that learns to differentiably select informative frames while discarding the other, to obtain Coarse representations. We also introduced the Multi-stage Fusion to best combine the Coarse stream with the Fine stream. We confirmed that the Coarse-Fine networks obtain the best known performance on Charades localization, while spending much less computation time.

In Chapter 4, we introduced a new weakly-guided self-supervised pretraining strategy for temporal activity detection, leveraging already-available weak labels. We defined a detection pretraining task with frame-level pseudo labels and three volume augmentation techniques, introducing multi-action frames and action segments to the single-action classification data. Our experiments confirmed the benefits of the proposed method across multiple models and challenging benchmarks. As takeaways, we further provide recommendations on when to use such pretrained models based on our observations.

In Chapter 5, we introduced LangRepo, a framework for adapting image-VLMs to video, with a focus on *video-conditioned text* embeddings. It can also benefit from freely-available auxiliary semantic information in the form of visually-grounded text, to guide the learned latent space. Our evaluations verified the importance of

updating text embeddings, across multiple activity recognition benchmarks, under few-shot, zero-shot, short-form and long-form settings. We believe that this work reveals the value of using language embeddings for temporal reasoning.

In Chapter 6, we introduced solutions for speeding up diffusion-based video editing, where we first analysed latency sources and adopted some off-the-shelf techniques to design reasonably-efficient baselines. Subsequently, motivated by the fact that video editing typically requires modifications to specific objects, we introduce Object-Centric Diffusion, comprising techniques for *i)* encouraging the merging of tokens in background regions and *ii)* limiting most of the diffusion sampling steps on foreground areas. Our solutions, which fix generation artifacts on foreground objects and further reduce editing latency, are validated on inversion-based and ControlNet-based models, by achieving 10× and 6× faster edits for comparable quality.

Through these contributions, we make strides towards efficient video understanding and generation, not only as faster inference pipelines, but also with label-efficient training setups.

7.1 Future Work

We currently explore more-recent video modeling pipelines, focusing on different facets of efficiency. Namely, we consider (1) efficient context-utilization of LLMs, with application to long-form video understanding, and (2) long-video generation as a lightweight extension of off-the-shelf short-term video generation models.

7.1.1 Efficient Context-utilization of Long-video LLMs

In this line of work, we look into the effectiveness of current open-source large language models (LLMs), in the context of long-video reasoning. Such a setting is especially challenging, as it needs to handle long context-lengths (*i.e.*, a large amount of tokens) during LLM inference. Consequently, the corresponding inefficiencies rise when applied to increasingly longer videos.

Even so, recent literature have progressed so far, enabling reasoning capabilities in hours-long video streams [67, 182], in contrast to very-limited temporal spans (*e.g.* seconds or minutes) just a few years ago. The work on efficient spatio-temporal attention mechanisms [5, 9], memory management [164, 202], and multi-modal large-language-models [182, 195, 231] have been key ingredients for such recent improvements.

LLMs, or more-specifically, vision-large-language-models (VLLMs) have been outperforming pure vision models in recent years in all facets, including image-based reasoning [90, 101, 246], grounding [88, 151], video understanding [195, 227, 231], and even robotics [2, 96, 236]. The sheer model scale and the vast pretraining data have enabled such frameworks to capture world knowledge and semantics, beyond what is possible with visual data only. Besides, the ability to process long context-lengths is also key, as it helps modeling long-term dependencies that are crucial for more-complex reasoning and interactions. However, recent studies show that despite the availability of such context-lengths, the effectiveness of models declines with longer input sequences [89]. This promotes the search for alternate representations that can compress input language data without losing meaningful information, essentially managing the context utilization of LLMs.

Motivated by the above, we introduce Language Repository (`LangRepo`), an interpretable representation for LLMs that updates iteratively. As `LangRepo` is all-textual, we rely on text-based operations to write and read information. The write operation (`write-to-repo`) prunes redundant text, creating concise descriptions that keep the context-utilization of LLMs in-check. Its iterative application with increasingly-longer chunks enables it to learn high-level semantics (*e.g.* long temporal dependencies). The read operation (`read-from-repo`) extracts such stored language information at various temporal scales, together with other optional metadata within the repository entries (*e.g.* timestamps). Altogether, our proposed framework is applied to long-term video reasoning tasks including visual question-answering (VQA) on EgoSchema [114], NExT-QA [207] and IntentQA [91], and visually-grounded VQA on NExT-GQA [208], showing strong performance at its scale. Based on these early promising results, we plan to investigate further on how such a representation scales with proprietary trillion-parameter LLMs (*e.g.* GPT-4).

7.1.2 Extending Video Diffusion to Longer Timespans

Here, we consider long-video generation based on diffusion models. The past couple of months have been monumental for generative AI in the context of video, mainly due to the introduction of Sora by OpenAI. In contrast to prior video generation pipelines [11, 22, 38] that usually generate content spanning a few seconds, Sora can generate mind-blowing videos spanning a few minutes, with significant motion throughout frames. Yet, this model is not yet publicly-served as their premium LLMs such as ChatGPT or GPT-4, probably due to the expensive (and slow) inference. The research community already speculates the huge training budget and the sheer scale of the model that is behind Sora [105]. This begs the question whether there exists an efficient framework to adapt off-the-shelf short-

video generation pipelines to generate consistent long-videos with reasonable motion. Such framework can be highly useful, as it would be easier to serve and widely-applicable with already-available many short-video generation backbones.

Generating even short videos is challenging, mainly due to two problems: preserving temporal consistency, and enabling reasonable motion in generated content. Video generation pipelines typically condition the generations on an initial image [11], text prompt [38], or both [22, 48]. Even with such signals, the generations are loosely-constrained, compared to pipelines such as video editing, which can rely on a lot-more information coming from a source video [140, 224]. Long-video generation only heightens these challenges [13, 28, 145, 218]. While consistency can be preserved by being faithful to an initial image, it can result in a lack of diversity among generated frames (*e.g.* not enough motion), basically collapsing into a static video [59]. On the other side of the spectrum, a significantly diverse set of frames can be generated by evolving the conditioning image or text prompt [48], but the temporal consistency may not be preserved. Such a trade-off can be avoided by applying a huge amount of compute budget to the problem (*e.g.* training data, model scale), yet, it brings its own concerns such as inefficiency as we currently see with Sora.

Based on this observation, we consider extending off-the-shelf short video generation pipelines to long-video generation. Motivated by already existing ideas such as feature sharing in video editing [77, 140, 188, 198, 224, 226] and evolving conditions in video generation [48, 145], we plan to formulate the problem of long-video generation, as a chunk-by-chunk generation of short-but-consistent video clips. Chunks can be conditioned on evolving frames or text prompts, that are hand-picked or generated by an off-the-shelf model. For instance, a meta text prompt can be expanded and sub-divided into multiple steps based on an LLM. To maintain consistency with such a setup, we plan to share information among chunks in the form of motion features [226] and attention maps [140]. However, a special focus should be given to avoid error propagation in each subsequent chunk. If we can formulate this framework with zero-to-minimal training, it will be widely-applicable not only in the industry, but also in the academic research community as well. Moreover, we observe multiple shortcomings in current evaluation metrics for video generation. For instance, although CLIP [146, 243] metrics capture semantic alignment, they fail to correlate well with fidelity. Current temporal consistency measures [243] are also brittle, which convey superior quality even for a static video content. To this end, we plan to explore metrics that can faithfully evaluate video generations, both in terms of fidelity and motion content.

Bibliography

[1] S. Abnar, M. Dehghani, B. Neyshabur, and H. Sedghi. Exploring the Limits of Large Scale Pre-training. *ICLR*, 2022. 33

[2] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. 110

[3] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, et al. Flamingo: a Visual Language Model for Few-Shot Learning. *NeurIPS*, 2022. 4, 9, 56

[4] H. Alwassel, S. Giancola, and B. Ghanem. Tsp: Temporally-sensitive pre-training of video encoders for localization tasks. In *ICCV*, pages 3173–3183, 2021. 8

[5] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid. ViViT: A video vision transformer. In *Proceedings of the International Conference on Computer Vision (ICCV)*, Oct. 2021. 2, 7, 55, 62, 67, 68, 109

[6] O. Avrahami, O. Fried, and D. Lischinski. Blended latent diffusion. *ACM Transactions on Graphics (TOG)*, 2023. 2, 75

[7] M. Bain, A. Nagrani, G. Varol, and A. Zisserman. Frozen in Time: A Joint Video and Image Encoder for End-to-End Retrieval. In *ICCV*, pages 1728–1738, 2021. 9

[8] M. Bain, A. Nagrani, G. Varol, and A. Zisserman. A CLIP-Hitchhiker's Guide to Long Video Retrieval. *arXiv preprint arXiv:2205.08508*, 2022. 10, 57, 58, 59, 60, 65, 68, 69

[9] G. Bertasius, H. Wang, and L. Torresani. Is Space-Time Attention All You Need for Video Understanding? In *ICML*, volume 2, page 4, 2021. 7, 55, 62, 65, 66, 68, 109

[10] G. Bertasius, H. Wang, and L. Torresani. Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning (ICML)*, July 2021. 2

[11] A. Blattmann, T. Dockhorn, S. Kulal, D. Mendelevitch, M. Kilian, D. Lorenz, Y. Levi, Z. English, V. Voleti, A. Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023. 110, 111

[12] A. Blattmann, R. Rombach, H. Ling, T. Dockhorn, S. W. Kim, S. Fidler, and K. Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2023. 2, 75

[13] A. Blattmann, R. Rombach, H. Ling, T. Dockhorn, S. W. Kim, S. Fidler, and K. Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22563–22575, 2023. 111

[14] D. Bolya and J. Hoffman. Token merging for fast stable diffusion. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2023. 5, 11, 76, 79, 91, 92

[15] D. Bolya, C.-Y. Fu, X. Dai, P. Zhang, C. Feichtenhofer, and J. Hoffman. Token merging: Your vit but faster. *International Conference on Learning Representations*, 2023. 5, 11, 76, 79, 91

[16] B. Brattoli, J. Tighe, F. Zhdanov, P. Perona, and K. Chalupka. Rethinking Zero-shot Video Classification: End-to-end Training for Realistic Applications. In *CVPR*, pages 4613–4623, 2020. 67

[17] S. Buch, V. Escorcia, C. Shen, B. Ghanem, and J. Carlos Niebles. SST: Single-Stream Temporal Action Proposals. In *CVPR*, pages 2911–2920, 2017. 7

[18] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 961–970, 2015. 8, 38

[19] J. Carreira and A. Zisserman. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In *CVPR*, pages 6299–6308, 2017. 7, 55

[20] J. Carreira and A. 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. 19, 23, 30, 33, 34, 36, 37, 38, 44, 48, 53

[21] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733. IEEE, 2017. 46, 48

[22] H. Chen, Y. Zhang, X. Cun, M. Xia, X. Wang, C. Weng, and Y. Shan. Videocrafter2: Overcoming data limitations for high-quality video diffusion models. *arXiv preprint arXiv:2401.09047*, 2024. 110, 111

[23] M.-H. Chen, B. Li, Y. Bao, G. AlRegib, and Z. Kira. Action Segmentation with Joint Self-Supervised Temporal Domain Adaptation. In *CVPR*, pages 9454–9463, 2020. 8

[24] S. Chen and D. Huang. Elaborative Rehearsal for Zero-shot Action Recognition. In *ICCV*, pages 13638–13647, 2021. 67

[25] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020. 8

[26] W. Chen, J. Wu, P. Xie, H. Wu, J. Li, X. Xia, X. Xiao, and L. Lin. Control-a-video: Controllable text-to-video generation with diffusion models. *arXiv preprint arXiv:2305.13840*, 2023. 10

[27] X. Chen and K. He. Exploring Simple Siamese Representation Learning. In *CVPR*, pages 15750–15758, 2021. 8

[28] X. Chen, Y. Wang, L. Zhang, S. Zhuang, X. Ma, J. Yu, Y. Wang, D. Lin, Y. Qiao, and Z. Liu. Seine: Short-to-long video diffusion model for generative transition and prediction. In *The Twelfth International Conference on Learning Representations*, 2023. 111

[29] Y. Chen, M. Mancini, X. Zhu, and Z. Akata. Semi-supervised and unsupervised deep visual learning: A survey. *TPAMI*, 2022. 8

[30] F. Cheng, X. Wang, J. Lei, D. Crandall, M. Bansal, and G. Bertasius. VindLU: A Recipe for Effective Video-and-Language Pretraining. *arXiv preprint arXiv:2212.05051*, 2022. 9

[31] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 764–773, 2017. 8

[32] R. Dai, S. Das, L. Minciullo, L. Garattoni, G. Francesca, and F. Bremond. PDAN: Pyramid Dilated Attention Network for Action Detection. In *WACV*, pages 2970–2979, 2021. 8, 45, 46, 48

[33] R. Dai, S. Das, K. Kahatapitiya, M. S. Ryoo, and F. Bremond. MS-TCT: Multi-Scale Temporal ConvTransformer for Action Detection. In *CVPR*, pages 20041–20051, 2022. 8, 46

[34] Z. Dai, B. Cai, Y. Lin, and J. Chen. UP-DETR: Unsupervised Pre-training for Object Detection with Transformers. In *CVPR*, pages 1601–1610, 2021. 3, 33

[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. Ieee, 2009. 30, 33

[36] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. 58

[37] V. Escorcia, F. C. Heilbron, J. C. Niebles, and B. Ghanem. Daps: Deep action proposals for action understanding. In *European Conference on Computer Vision*, pages 768–784. Springer, 2016. 7, 8

[38] P. Esser, J. Chiu, P. Atighehchian, J. Granskog, and A. Germanidis. Structure and content-guided video synthesis with diffusion models. In *IEEE International Conference on Computer Vision*, 2023. 10, 110, 111

[39] H. Fan, B. Xiong, K. Mangalam, Y. Li, Z. Yan, J. Malik, and C. Feichtenhofer. Multiscale Vision Transformers. In *ICCV*, pages 6824–6835, 2021. 55, 69

[40] C. Feichtenhofer. X3D: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 203–213, 2020. 5, 7, 19, 20, 23, 29, 30, 31, 44, 45, 46, 47, 48, 49, 50, 55, 69

[41] C. Feichtenhofer, H. Fan, J. Malik, and K. He. Slowfast networks for video recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6202–6211, 2019. 2, 4, 5, 7, 12, 14, 24, 25, 26, 28, 30, 45, 46, 47, 51, 55, 69

[42] C. Feichtenhofer, H. Fan, B. Xiong, R. Girshick, and K. He. A Large-Scale Study on Unsupervised Spatiotemporal Representation Learning. In *CVPR*, pages 3299–3309, 2021. 7, 55

[43] J. Gao, T. Zhang, and C. Xu. I Know the Relationships: Zero-Shot Action Recognition via Two-Stream Graph Convolutional Networks and Knowledge Graphs. In *AAAI*, volume 33, pages 8303–8311, 2019. 67

[44] J. Gao, Z. Wang, J. Xuan, and S. Fidler. Beyond fixed grid: Learning geometric image representation with a deformable grid. In *European Conference on Computer Vision*, pages 108–125. Springer, 2020. 8, 31

[45] M. Geyer, O. Bar-Tal, S. Bagon, and T. Dekel. Tokenflow: Consistent diffusion features for consistent video editing. *International Conference on Learning Representations*, 2024. 10, 77, 79, 86, 100, 101, 102

[46] D. Ghadiyaram, D. Tran, and D. Mahajan. Large-Scale Weakly-Supervised Pre-Training for Video Action Recognition. In *CVPR*, pages 12046–12055, 2019. 33

[47] P. Ghosh, Y. Yao, L. Davis, and A. Divakaran. Stacked spatio-temporal graph convolutional networks for action segmentation. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 576–585, 2020. 3, 23, 24, 34, 46

[48] R. Girdhar, M. Singh, A. Brown, Q. Duval, S. Azadi, S. S. Rambhatla, A. Shah, X. Yin, D. Parikh, and I. Misra. Emu video: Factorizing text-to-video generation by explicit image conditioning. *arXiv preprint arXiv:2311.10709*, 2023. 111

[49] G. Gong, X. Wang, Y. Mu, and Q. Tian. Learning Temporal Co-Attention Models for Unsupervised Video Action Localization. In *CVPR*, pages 9819–9828, 2020. 8

[50] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Neural Information Processing Systems*, 2014. 2, 75

[51] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 30

[52] C. Gu, C. Sun, D. A. Ross, C. Vondrick, C. Pantofaru, Y. Li, S. Vijayanarasimhan, G. Toderici, S. Ricco, R. Sukthankar, et al. AVA: A Video Dataset of Spatio-temporally Localized Atomic Visual Actions. In *CVPR*, pages 6047–6056, 2018. 7

[53] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui. Open-vocabulary Object Detection via Vision and Language Knowledge Distillation. *ICLR*, 2021. 3, 9, 55, 56

[54] H. Guo, Z. Ren, Y. Wu, G. Hua, and Q. Ji. Uncertainty-based spatial-temporal attention for online action detection. In *ECCV*, pages 69–86. Springer, 2022. 7

[55] Z. Guo, R. Zhang, L. Qiu, X. Ma, X. Miao, X. He, and B. Cui. CALIP: Zero-Shot Enhancement of CLIP with Parameter-free Attention. *AAAI*, 2023. 9

[56] T. Han, W. Xie, and A. Zisserman. Self-supervised Co-training for Video Representation Learning. In *NeurIPS*, 2020. 7, 55

[57] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 19, 23, 46

[58] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2020. doi: 10.1109/cvpr42600.2020.00975. URL http://dx.doi.org/10.1109/cvpr42600.2020.00975. 8

[59] Y. He, T. Yang, Y. Zhang, Y. Shan, and Q. Chen. Latent video diffusion models for high-fidelity long video generation. *arXiv preprint arXiv:2211.13221*, 2022. 111

[60] A. Hertz, R. Mokady, J. Tenenbaum, K. Aberman, Y. Pritch, and D. Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022. 86

[61] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Neural Information Processing Systems*, 2020. 2, 75

[62] J. Ho, W. Chan, C. Saharia, J. Whang, R. Gao, A. Gritsenko, D. P. Kingma, B. Poole, M. Norouzi, D. J. Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022. 10

[63] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 2022. 2, 75

[64] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet. Video diffusion models, 2022. 10

[65] W. Hong, M. Ding, W. Zheng, X. Liu, and J. Tang. Cogvideo: Large-scale pretraining for text-to-video generation via transformers. *International Conference on Learning Representations*, 2023. 10

[66] P.-Y. Huang, V. Sharma, H. Xu, C. Ryali, H. Fan, Y. Li, S.-W. Li, G. Ghosh, J. Malik, and C. Feichtenhofer. MAViL: Masked Audio-Video Learners. *arXiv preprint arXiv:2212.08071*, 2022. 7

[67] M. M. Islam, N. Ho, X. Yang, T. Nagarajan, L. Torresani, and G. Bertasius. Video recap: Recursive captioning of hour-long videos. *arXiv preprint arXiv:2402.13250*, 2024. 109

[68] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015. 8

[69] M. Jain, A. Ghodrati, and C. G. Snoek. ActionBytes: Learning from Trimmed Videos to Localize Actions. In *CVPR*, pages 1171–1180, 2020. 8

[70] J. Ji, K. Cao, and J. C. Niebles. Learning Temporal Action Proposals With Fewer Labels. In *ICCV*, pages 7073–7082, 2019. 8

[71] J. Ji, R. Krishna, L. Fei-Fei, and J. C. Niebles. Action genome: Actions as compositions of spatio-temporal scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10236–10247, 2020. 4, 58

[72] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. Le, Y.-H. Sung, Z. Li, and T. Duerig. Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision. In *ICML*, pages 4904–4916. PMLR, 2021. 3, 8, 55

[73] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan. VIMA: General Robot Manipulation with Multimodal Prompts. *arXiv preprint arXiv:2210.03094*, 2022. 3, 9, 55

[74] Y.-G. Jiang, J. Liu, A. R. Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. Thumos challenge: Action recognition with a large number of classes, 2014. 28, 54

[75] C. Ju, T. Han, K. Zheng, Y. Zhang, and W. Xie. Prompting Visual-Language Models for Efficient Video Understanding. In *ECCV*, pages 105–124. Springer, 2022. 9

[76] K. Kahatapitiya and M. S. Ryoo. Coarse-Fine Networks for Temporal Activity Detection in Videos. In *CVPR*, pages 8385–8394, 2021. 3, 5, 7, 8, 34, 44, 45, 46, 47, 51

[77] K. Kahatapitiya, A. Karjauv, D. Abati, F. Porikli, Y. M. Asano, and A. Habibian. Object-centric diffusion for efficient video editing. *arXiv preprint arXiv:2401.05735*, 2024. 111

[78] S. Karamcheti, S. Nair, A. S. Chen, T. Kollar, C. Finn, D. Sadigh, and P. Liang. Language-Driven Representation Learning for Robotics. *arXiv preprint arXiv:2302.12766*, 2023. 9

[79] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 7, 22, 28, 30, 64, 65, 66, 68, 69, 71, 72

[80] L. Khachatryan, A. Movsisyan, V. Tadevosyan, R. Henschel, Z. Wang, S. Navasardyan, and H. Shi. Text2video-zero: Text-to-image diffusion mod-

els are zero-shot video generators. *IEEE International Conference on Computer Vision*, 2023. 10, 87, 88, 101, 103, 104

[81] B.-K. Kim, H.-K. Song, T. Castells, and S. Choi. On architectural compression of text-to-image diffusion models. *arXiv preprint arXiv:2305.15798*, 2023. 11

[82] D. Kim, C.-H. Lai, W.-H. Liao, N. Murata, Y. Takida, T. Uesaka, Y. He, Y. Mitsufuji, and S. Ermon. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. *arXiv preprint arXiv:2310.02279*, 2023. 75

[83] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *International Conference on Learning Representations*, 2014. 2, 75

[84] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023. 81

[85] B. Korbar, D. Tran, and L. Torresani. Scsampler: Sampling salient clips from video for efficient action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6232–6242, 2019. 8

[86] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: A large video database for human motion recognition. In *ICCV*, pages 2556–2563. IEEE, 2011. 7, 64, 65, 66, 67, 72, 73

[87] A. Kukleva, H. Kuehne, F. Sener, and J. Gall. Unsupervised Learning of Action Classes With Continuous Temporal Embedding. In *CVPR*, pages 12066–12074, 2019. 8

[88] X. Lai, Z. Tian, Y. Chen, Y. Li, Y. Yuan, S. Liu, and J. Jia. Lisa: Reasoning segmentation via large language model. *arXiv preprint arXiv:2308.00692*, 2023. 110

[89] M. Levy, A. Jacoby, and Y. Goldberg. Same task, more tokens: the impact of input length on the reasoning performance of large language models. *arXiv preprint arXiv:2402.14848*, 2024. 110

[90] J. Li, D. Li, S. Savarese, and S. Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023. 110

[91] J. Li, P. Wei, W. Han, and L. Fan. Intentqa: Context-aware video intent reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11963–11974, 2023. 110

[92] M. Li, J. Lin, C. Meng, S. Ermon, S. Han, and J.-Y. Zhu. Efficient spatially sparse inference for conditional gans and diffusion models. *Neural Information Processing Systems*, 2022. 11

[93] X. Li, C. Ma, X. Yang, and M.-H. Yang. Vidtome: Video token merging for zero-shot video editing. *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2024. 11, 77, 84, 86

[94] Y. Li, C.-Y. Wu, H. Fan, K. Mangalam, B. Xiong, J. Malik, and C. Feichtenhofer. MViTv2: Improved Multiscale Vision Transformers for Classification and Detection. In *CVPR*, pages 4804–4814, 2022. 68

[95] Y. Li, H. Wang, Q. Jin, J. Hu, P. Chemerys, Y. Fu, Y. Wang, S. Tulyakov, and J. Ren. Snapfusion: Text-to-image diffusion model on mobile devices within two seconds. *Neural Information Processing Systems*, 2023. 11

[96] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE, 2023. 110

[97] J. Lin, C. Gan, and S. Han. TSM: Temporal Shift Module for Efficient Video Understanding. In *ICCV*, pages 7083–7093, 2019. 9, 55, 65, 66

[98] K. Q. Lin, A. J. Wang, M. Soldan, M. Wray, R. Yan, E. Z. Xu, D. Gao, R. Tu, W. Zhao, W. Kong, et al. Egocentric Video-Language Pretraining. *NeurIPS*, 2022. 9

[99] Z. Lin, S. Geng, R. Zhang, P. Gao, G. de Melo, X. Wang, J. Dai, Y. Qiao, and H. Li. Frozen CLIP Models are Efficient Video Learners. *arXiv preprint arXiv:2208.03550*, 2022. 10, 57, 58, 59, 60, 64, 65, 67, 68, 72, 73

[100] D. Liu, T. Jiang, and Y. Wang. Completeness Modeling and Context Separation for Weakly Supervised Temporal Action Localization. In *CVPR*, pages 1298–1307, 2019. 8

[101] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024. 110

[102] L. Liu and J. Deng. Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 8

[103] S. Liu, Y. Zhang, W. Li, Z. Lin, and J. Jia. Video-p2p: Video editing with cross-attention control. *arXiv preprint arXiv:2303.04761*, 2023. 10

[104] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. In *ICLR*, 2020. 3, 33

[105] Y. Liu, K. Zhang, Y. Li, Z. Yan, C. Gao, R. Chen, Z. Yuan, Y. Huang, H. Sun, J. Gao, et al. Sora: A review on background, technology, limitations, and opportunities of large vision models. *arXiv preprint arXiv:2402.17177*, 2024. 110

[106] Z. Liu, L. Wang, Q. Zhang, Z. Gao, Z. Niu, N. Zheng, and G. Hua. Weakly Supervised Temporal Action Localization Through Contrast Based Evaluation Networks. In *ICCV*, pages 3899–3908, 2019. 8

[107] Z. Liu, L. Wang, Q. Zhang, W. Tang, J. Yuan, N. Zheng, and G. Hua. Acsnet: Action-context separation network for weakly supervised temporal action localization. In *AAAI*, volume 35, pages 2233–2241, 2021. 7

[108] Z. Liu, J. Ning, Y. Cao, Y. Wei, Z. Zhang, S. Lin, and H. Hu. Video Swin Transformer. In *CVPR*, pages 3202–3211, 2022. 7, 65, 66, 68

[109] I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization. *ICLR*, 2019. 65

[110] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Neural Information Processing Systems*, 2022. 3, 11, 75, 80

[111] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022. 3, 11, 75, 80

[112] H. Luo, L. Ji, M. Zhong, Y. Chen, W. Lei, N. Duan, and T. Li. CLIP4Clip: An Empirical Study of CLIP for End to End Video Clip Retrieval. *Neurocomputing*, 508:293–304, 2022. 10, 57, 58, 59, 60, 69

[113] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. Van Der Maaten. Exploring the Limits of Weakly Supervised Pretraining. In *ECCV*, pages 181–196, 2018. 3, 33

[114] K. Mangalam, R. Akshulakov, and J. Malik. Egoschema: A diagnostic benchmark for very long-form video language understanding. *Advances in Neural Information Processing Systems*, 36, 2024. 110

[115] E. Mavroudi, B. B. Haro, and R. Vidal. Representation learning on visual-symbolic graphs for video understanding. In *European Conference on Computer Vision*, pages 71–90. Springer, 2020. 3, 23, 24, 34, 46

[116] C. Meng, Y. Song, J. Song, J. Wu, J.-Y. Zhu, and S. Ermon. Sdedit: Image synthesis and editing with stochastic differential equations. *International Conference on Learning Representations*, 2022. 100, 101, 102

[117] C. Meng, R. Rombach, R. Gao, D. Kingma, S. Ermon, J. Ho, and T. Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2023. 3, 11, 75

[118] Y. Meng, C.-C. Lin, R. Panda, P. Sattigeri, L. Karlinsky, A. Oliva, K. Saenko, and R. Feris. Ar-net: Adaptive frame resolution for efficient action recognition. In *European Conference on Computer Vision*, pages 86–104. Springer, 2020. 8

[119] S. Menon and C. Vondrick. Visual Classification via Description from Large Language Models. *arXiv preprint arXiv:2210.07183*, 2022. 9

[120] Microsoft. Microsoft deepspeed. `https://github.com/microsoft/DeepSpeed`, 2023. 78

[121] M. Minderer, A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran, A. Arnab, M. Dehghani, Z. Shen, et al. Simple Open-Vocabulary Object Detection with Vision Transformers. *ECCV*, 2022. 3, 9, 55

[122] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: Unsupervised learning using temporal order verification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 527–544. Springer, 2016. 8

[123] R. Mokady, A. Hertz, K. Aberman, Y. Pritch, and D. Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2023. 86

[124] A. Nagrani, S. Yang, A. Arnab, A. Jansen, C. Schmid, and C. Sun. Attention Bottlenecks for Multimodal Fusion. *NeurIPS*, 34:14200–14213, 2021. 7, 55

[125] P. Nguyen, T. Liu, G. Prasad, and B. Han. Weakly Supervised Action Localization by Sparse Temporal Pooling Network. In *CVPR*, pages 6752–6761, 2018. 8

[126] B. Ni, H. Peng, M. Chen, S. Zhang, G. Meng, J. Fu, S. Xiang, and H. Ling. Expanding Language-Image Pretrained Models for General Video Recognition. In *ECCV*, pages 1–18. Springer, 2022. 10, 56, 57, 58, 59, 60, 64, 65, 66, 67, 68, 72, 73, 74

[127] D. Nukrai, R. Mokady, and A. Globerson. Text-Only Training for Image Captioning using Noise-Injected CLIP. *EMNLP*, 2022. 9

[128] R. Paiss, A. Ephrat, O. Tov, S. Zada, I. Mosseri, M. Irani, and T. Dekel. Teaching CLIP to Count to Ten. *arXiv preprint arXiv:2302.12066*, 2023. 9

[129] J. Pan, Z. Lin, X. Zhu, J. Shao, and H. Li. ST-Adapter: Parameter-Efficient Image-to-Video Transfer Learning. *NeurIPS*, 2022. 68

[130] W. Peebles and S. Xie. Scalable diffusion models with transformers. In *IEEE International Conference on Computer Vision*, 2023. 84

[131] A. Piergiovanni and M. Ryoo. Temporal Gaussian Mixture Layer for Videos. In *ICML*, pages 5152–5161. PMLR, 2019. 2, 7

[132] A. Piergiovanni and M. S. Ryoo. Learning latent super-events to detect multiple activities in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5304–5313, 2018. 3, 22, 23, 24, 25, 34, 45, 46

[133] A. Piergiovanni and M. S. Ryoo. Learning Latent Super-Events to Detect Multiple Activities in Videos. In *CVPR*, pages 5304–5313, 2018. 2, 7

[134] A. Piergiovanni and M. S. Ryoo. Representation flow for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9945–9953, 2019. 32

[135] A. Piergiovanni and M. S. Ryoo. Temporal gaussian mixture layer for videos. In *International Conference on Machine learning*, pages 5152–5161, 2019. 3, 8, 22, 23, 24, 25, 30, 34, 45, 46, 48

[136] A. Piergiovanni, A. Angelova, A. Toshev, and M. S. Ryoo. Evolving Space-Time Neural Architectures for Videos. In *ICCV*, pages 1793–1802, 2019. 69

[137] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool. The 2017 davis challenge on video object segmentation. *arXiv:1704.00675*, 2017. 84, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104

[138] R. P. Poudel, S. Liwicki, and R. Cipolla. Fast-SCNN: Fast Semantic Segmentation Network. In *BMVC*, 2019. 3, 33

[139] S. Purushwalkam, T. Ye, S. Gupta, and A. Gupta. Aligning Videos in Space and Time. In *ECCV*, pages 262–278. Springer, 2020. 8

[140] C. Qi, X. Cun, Y. Zhang, C. Lei, X. Wang, Y. Shan, and Q. Chen. Fatezero: Fusing attentions for zero-shot text-based video editing. *IEEE International Conference on Computer Vision*, 2023. 2, 3, 10, 75, 76, 77, 79, 84, 85, 86, 91, 93, 95, 100, 101, 102, 104, 105, 106, 111

[141] R. Qian, T. Meng, B. Gong, M.-H. Yang, H. Wang, S. Belongie, and Y. Cui. Spatiotemporal Contrastive Video Representation Learning. In *CVPR*, pages 6964–6974, 2021. 7, 55

[142] R. Qian, Y. Li, Z. Xu, M.-H. Yang, S. Belongie, and Y. Cui. Multimodal Open-Vocabulary Video Classification via Pre-Trained Vision and Language Models. *arXiv preprint arXiv:2207.07646*, 2022. 9

[143] J. Qin, L. Liu, L. Shao, F. Shen, B. Ni, J. Chen, and Y. Wang. Zero-Shot Action Recognition with Error-Correcting Output Codes. In *CVPR*, pages 2833–2842, 2017. 67

[144] X. Qin, Z. Zhang, C. Huang, M. Dehghan, O. R. Zaiane, and M. Jagersand. U2-net: Going deeper with nested u-structure for salient object detection. *Pattern Recognition*, 2020. 81

[145] H. Qiu, M. Xia, Y. Zhang, Y. He, X. Wang, Y. Shan, and Z. Liu. Freenoise: Tuning-free longer video diffusion via noise rescheduling. *arXiv preprint arXiv:2310.15169*, 2023. 111

[146] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from

natural language supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 8748–8763. PMLR, 2021. 93, 111

[147] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning Transferable Visual Models From Natural Language Supervision. In *ICML*, pages 8748–8763. PMLR, 2021. 3, 8, 55, 56, 58, 59, 63, 64, 70, 72, 73

[148] K. Ranasinghe, B. McKinzie, S. Ravi, Y. Yang, A. Toshev, and J. Shlens. Perceptual Grouping in Vision-Language Models. *arXiv preprint arXiv:2210.09996*, 2022. 9

[149] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 87

[150] H. Rasheed, M. U. Khattak, M. Maaz, S. Khan, and F. S. Khan. Fine-tuned CLIP Models are Efficient Video Learners. *arXiv preprint arXiv:2212.03640*, 2022. 9

[151] H. Rasheed, M. Maaz, S. Shaji, A. Shaker, S. Khan, H. Cholakkal, R. M. Anwer, E. Xing, M.-H. Yang, and F. S. Khan. Glamm: Pixel grounding large multimodal model. *arXiv preprint arXiv:2311.03356*, 2023. 110

[152] A. Recasens, P. Kellnhofer, S. Stent, W. Matusik, and A. Torralba. Learning to zoom: a saliency-based sampling layer for neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 51–66, 2018. 8

[153] A. Recasens, P. Luc, J.-B. Alayrac, L. Wang, F. Strub, C. Tallec, M. Malinowski, V. Pătrăucean, F. Altché, M. Valko, et al. Broaden Your Views for Self-Supervised Video Learning. In *ICCV*, pages 1255–1265, 2021. 7, 55

[154] A. Recasens, P. Luc, J.-B. Alayrac, L. Wang, F. Strub, C. Tallec, M. Malinowski, V. Pătrăucean, F. Altché, M. Valko, J.-B. Grill, A. van den Oord, and A. Zisserman. Broaden Your Views for Self-Supervised Video Learning. In *ICCV*, pages 1255–1265, October 2021. 8

[155] A. Recasens, J. Lin, J. Carreira, D. Jaegle, L. Wang, J.-b. Alayrac, P. Luc, A. Miech, L. Smaira, R. Hemsley, et al. Zorro: the masked multimodal transformer. *arXiv preprint arXiv:2301.09595*, 2023. 7

[156] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan, et al. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*, 2024. 87, 90

[157] A. Richard, H. Kuehne, and J. Gall. Weakly Supervised Action Learning with RNN based Fine-to-coarse Modeling. In *CVPR*, pages 754–763, 2017. 8

[158] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2022. 2, 75, 100, 101

[159] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2015. 2, 81, 92, 102, 105, 106

[160] M. Ryoo, A. Piergiovanni, M. Tan, and A. Angelova. AssembleNet: Searching for multi-stream neural connectivity in video architectures. In *International Conference on Learning Representations (ICLR)*, 2020. 2, 4, 7, 12, 14, 55, 69

[161] M. S. Ryoo, A. Piergiovanni, J. Kangaspunta, and A. Angelova. AssembleNet++: Assembling Modality Representations via Attention Connections. In *ECCV*, pages 654–671. Springer, 2020. 7

[162] M. S. Ryoo, A. Piergiovanni, A. Arnab, M. Dehghani, and A. Angelova. TokenLearner: Adaptive space-time tokenization for videos. In *Advances in Neural Information Processing Systems (NeurIPS)*, Dec. 2021. 7, 67, 68

[163] M. S. Ryoo, K. Gopalakrishnan, K. Kahatapitiya, T. Xiao, K. Rao, A. Stone, Y. Lu, J. Ibarz, and A. Arnab. Token Turing Machines. *arXiv preprint arXiv:2211.09119*, 2022. 7

[164] M. S. Ryoo, K. Gopalakrishnan, K. Kahatapitiya, T. Xiao, K. Rao, A. Stone, Y. Lu, J. Ibarz, and A. Arnab. Token turing machines. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19070–19081, 2023. 109

[165] T. Salimans and J. Ho. Progressive distillation for fast sampling of diffusion models. *International Conference on Learning Representations*, 2022. 3, 11, 75

[166] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1): 61–80, 2008. 8

[167] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Neural Information Processing Systems*, 2022. 75

[168] M. Schwarzer, N. Rajkumar, M. Noukhovitch, A. Anand, L. Charlin, D. Hjelm, P. Bachman, and A. Courville. Pretraining Representations for Data-Efficient Reinforcement Learning. In *NeurIPS*, 2021. 3, 33

[169] F. Sener and A. Yao. Unsupervised Learning and Segmentation of Complex Activities From Video. In *CVPR*, pages 8368–8376, 2018. 8

[170] R. Sennrich, B. Haddow, and A. Birch. Neural Machine Translation of Rare Words with Subword Units. *ACL*, 2016. 59

[171] B. Shi, Q. Dai, Y. Mu, and J. Wang. Weakly-Supervised Action Localization by Generative Attention Modeling. In *CVPR*, pages 1009–1019, 2020. 8

[172] Z. Shou, D. Wang, and S.-F. Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1049–1058, 2016. 8

[173] Z. Shou, J. Chan, A. Zareian, K. Miyazawa, and S.-F. Chang. Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5734–5743, 2017. 8

[174] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, pages 510–526. Springer, 2016. 5, 7, 8, 13, 14, 21, 22, 23, 26, 27, 28, 35, 36, 44, 45, 46, 47, 49, 50, 51, 53, 54, 64, 65, 67, 69, 70, 71, 72

[175] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 30

[176] U. Singer, A. Polyak, T. Hayes, X. Yin, J. An, S. Zhang, Q. Hu, H. Yang, O. Ashual, O. Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *International Conference on Learning Representations*, 2023. 10

[177] A. Singh, R. Hu, V. Goswami, G. Couairon, W. Galuba, M. Rohrbach, and D. Kiela. FLAVA: A Foundational Language And Vision Alignment Model. In *CVPR*, pages 15638–15650, 2022. 3, 9, 55

[178] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. *International Conference on Learning Representations*, 2021. 2, 3, 10, 75, 80, 86

[179] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild. *arXiv preprint arXiv:1212.0402*, 2012. 64, 65, 66, 67, 72, 73

[180] H.-T. Su, Y. Niu, X. Lin, W. H. Hsu, and S.-F. Chang. Language Models are Causal Knowledge Extractors for Zero-shot Video Question Answering. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4951–4960, 2023. URL https://api.semanticscholar.org/CorpusID:258041332. 74

[181] C. Sun, S. Shetty, R. Sukthankar, and R. Nevatia. Temporal Localization of Fine-Grained Actions in Videos by Domain Transfer from Web Images. In *ACMMM*, pages 371–380, 2015. 8

[182] G. Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. 109

[183] P. Tirupattur, K. Duarte, Y. S. Rawat, and M. Shah. Modeling multi-label action dependencies for temporal action localization. In *CVPR*, pages 1460–1470, 2021. 7

[184] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. 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. 23

[185] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A Closer Look at Spatiotemporal Convolutions for Action Recognition. In *CVPR*, pages 6450–6459, 2018. 7, 55

[186] M. Tschannen, B. Mustafa, and N. Houlsby. Image-and-Language Understanding from Pixels Only. *arXiv preprint arXiv:2212.08045*, 2022. 9

[187] N. Tumanyan, M. Geyer, S. Bagon, and T. Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2023. 10, 79, 86, 91, 104

[188] N. Tumanyan, M. Geyer, S. Bagon, and T. Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1921–1930, 2023. 111

[189] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, D. Lopez-Paz, and Y. Bengio. Manifold Mixup: Better Representations by Interpolating Hidden States. In *ICML*, pages 6438–6447. PMLR, 2019. 44, 49, 50

[190] M. Wang, J. Xing, and Y. Liu. ActionCLIP: A New Paradigm for Video Action Recognition. *arXiv preprint arXiv:2109.08472*, 2021. 9, 57, 58, 59, 60, 65, 67, 68, 69, 72, 73

[191] Q. Wang and K. Chen. Alternative Semantic Representations for Zero-Shot Human Action Recognition. In *ECML-PKDD*, pages 87–102. Springer, 2017. 67

[192] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018. 30

[193] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local Neural Networks. In *CVPR*, pages 7794–7803, 2018. 7, 69

[194] Y. Wang, K. Li, Y. Li, Y. He, B. Huang, Z. Zhao, H. Zhang, J. Xu, Y. Liu, Z. Wang, et al. InternVideo: General Video Foundation Models via Generative and Discriminative Learning. *arXiv preprint arXiv:2212.03191*, 2022. 74

[195] Y. Wang, K. Li, Y. Li, Y. He, B. Huang, Z. Zhao, H. Zhang, J. Xu, Y. Liu, Z. Wang, et al. Internvideo: General video foundation models via generative and discriminative learning. *arXiv preprint arXiv:2212.03191*, 2022. 109, 110

[196] Z. Wang, M. Li, R. Xu, L. Zhou, J. Lei, X. Lin, S. Wang, Z. Yang, C. Zhu, D. Hoiem, et al. Language Models with Image Descriptors are Strong Few-Shot Video-Language Learners. *arXiv preprint arXiv:2205.10747*, 2022. 4, 9, 58

[197] D. Wei, J. J. Lim, A. Zisserman, and W. T. Freeman. Learning and Using the Arrow of Time. In *CVPR*, pages 8052–8060, 2018. 8

[198] B. Wu, C.-Y. Chuang, X. Wang, Y. Jia, K. Krishnakumar, T. Xiao, F. Liang, L. Yu, and P. Vajda. Fairy: Fast parallelized instruction-guided video-to-video synthesis. *arXiv preprint arXiv:2312.13834*, 2023. 111

[199] C.-Y. Wu, C. Feichtenhofer, H. Fan, K. He, P. Krahenbuhl, and R. Girshick. Long-Term Feature Banks for Detailed Video Understanding. In *CVPR*, pages 284–293, 2019. 69

[200] C.-Y. Wu, C. Feichtenhofer, H. Fan, K. He, P. Krahenbuhl, and R. Girshick. Long-Term Feature Banks for Detailed Video Understanding. In *CVPR*, pages 284–293, 2019. 7

[201] C.-Y. Wu, R. Girshick, K. He, C. Feichtenhofer, and P. Krahenbuhl. A multigrid method for efficiently training video models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 153–162, 2020. 30, 31

[202] C.-Y. Wu, Y. Li, K. Mangalam, H. Fan, B. Xiong, J. Malik, and C. Feichtenhofer. MeMViT: Memory-Augmented Multiscale Vision Transformer for Efficient Long-Term Video Recognition. In *CVPR*, pages 13587–13597, 2022. 7, 109

[203] J. Z. Wu, Y. Ge, X. Wang, S. W. Lei, Y. Gu, Y. Shi, W. Hsu, Y. Shan, X. Qie, and M. Z. Shou. Tune-a-video: One-shot tuning of image diffusion models for text-to-video generation. In *IEEE International Conference on Computer Vision*, 2023. 2, 10, 75, 77, 79, 85, 86, 100, 101, 102

[204] W. Wu, Z. Sun, and W. Ouyang. Revisiting Classifier: Transferring Vision-Language Models for Video Recognition. *AAAI*, 2023. 68

[205] Z. Wu, C. Xiong, Y.-G. Jiang, and L. S. Davis. Liteeval: A coarse-to-fine framework for resource efficient video recognition. In *Advances in Neural Information Processing Systems*, pages 7780–7789, 2019. 8

[206] Z. Wu, C. Xiong, C.-Y. Ma, R. Socher, and L. S. Davis. Adaframe: Adaptive frame selection for fast video recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1278–1287, 2019. 8

[207] J. Xiao, X. Shang, A. Yao, and T.-S. Chua. Next-qa: Next phase of question-answering to explaining temporal actions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9777–9786, 2021. 74, 110

[208] J. Xiao, A. Yao, Y. Li, and T. S. Chua. Can i trust your answer? visually grounded video question answering. *arXiv preprint arXiv:2309.01327*, 2023. 110

[209] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. Rethinking Spatiotemporal Feature Learning For Video Understanding. *arXiv preprint arXiv:1712.04851*, 1(2):5, 2017. 7, 55

[210] Z. Xinyi and L. Chen. Capsule graph neural network. In *International Conference on Learning Representations*, 2018. 8

[211] H. Xu, A. Das, and K. Saenko. R-c3d: Region convolutional 3d network for temporal activity detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5783–5792, 2017. 8

[212] H. Xu, G. Ghosh, P.-Y. Huang, D. Okhonko, A. Aghajanyan, F. Metze, L. Zettlemoyer, and C. Feichtenhofer. VideoCLIP: Contrastive Pre-training for Zero-shot Video-Text Understanding. *EMNLP*, 2021. 9, 56

[213] J. Xu, T. Mei, T. Yao, and Y. Rui. MSR-VTT: A Large Video Description Dataset for Bridging Video and Language. In *CVPR*, pages 5288–5296, 2016. 7

[214] J. Xu, S. De Mello, S. Liu, W. Byeon, T. Breuel, J. Kautz, and X. Wang. GroupViT: Semantic Segmentation Emerges from Text Supervision. In *CVPR*, pages 18134–18144, 2022. 9

[215] M. Xu, J.-M. Pérez-Rúa, V. Escorcia, B. Martinez, X. Zhu, L. Zhang, B. Ghanem, and T. Xiang. Boundary-sensitive pre-training for temporal localization in videos. In *ICCV*, pages 7220–7230, 2021. 8

[216] M. Xu, J. M. Perez Rua, X. Zhu, B. Ghanem, and B. Martinez. Low-fidelity video encoder optimization for temporal action localization. *NeurIPS*, 34: 9923–9935, 2021. 8

[217] X. Xu, T. M. Hospedales, and S. Gong. Multi-Task Zero-Shot Action Recognition with Prioritised Data Augmentation . In *ECCV*, pages 343–359. Springer, 2016. 67

[218] Z. Xu, J. Zhang, J. H. Liew, H. Yan, J.-W. Liu, C. Zhang, J. Feng, and M. Z. Shou. Magicanimate: Temporally consistent human image animation using diffusion model. *arXiv preprint arXiv:2311.16498*, 2023. 111

[219] H. Xue, Y. Sun, B. Liu, J. Fu, R. Song, H. Li, and J. Luo. CLIP-ViP: Adapting Pre-trained Image-Text Model to Video-Language Representation Alignment. *arXiv preprint arXiv:2209.06430*, 2022. 9

[220] S. Yan, X. Xiong, A. Arnab, Z. Lu, M. Zhang, C. Sun, and C. Schmid. Multiview Transformers for Video Recognition. In *CVPR*, pages 3333–3343, 2022. 67, 68

[221] S. Yan, T. Zhu, Z. Wang, Y. Cao, M. Zhang, S. Ghosh, Y. Wu, and J. Yu. Video-Text Modeling with Zero-Shot Transfer from Contrastive Captioners. *arXiv preprint arXiv:2212.04979*, 2022. 3, 9, 55

[222] A. Yang, A. Miech, J. Sivic, I. Laptev, and C. Schmid. Just ask: Learning to answer questions from millions of narrated videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1686–1697, 2021. 74

[223] A. Yang, A. Nagrani, P. H. Seo, A. Miech, J. Pont-Tuset, I. Laptev, J. Sivic, and C. Schmid. Vid2Seq: Large-Scale Pretraining of a Visual Language Model for Dense Video Captioning. *arXiv preprint arXiv:2302.14115*, 2023. 9

[224] S. Yang, Y. Zhou, Z. Liu, and C. C. Loy. Rerender a video: Zero-shot text-guided video-to-video translation. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–11, 2023. 111

[225] L. Yao, R. Huang, L. Hou, G. Lu, M. Niu, H. Xu, X. Liang, Z. Li, X. Jiang, and C. Xu. FILIP: Fine-grained Interactive Language-Image Pre-Training. *arXiv preprint arXiv:2111.07783*, 2021. 3, 9, 55

[226] D. Yatim, R. Fridman, O. B. Tal, Y. Kasten, and T. Dekel. Space-time diffusion features for zero-shot text-driven motion transfer. *arXiv preprint arXiv:2311.17009*, 2023. 111

[227] Q. Ye, H. Xu, G. Xu, J. Ye, M. Yan, Y. Zhou, J. Wang, A. Hu, P. Shi, Y. Shi, et al. mplug-owl: Modularization empowers large language models with multimodality. *arXiv preprint arXiv:2304.14178*, 2023. 110

[228] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2678–2687, 2016. 7, 8

[229] S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and L. Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *International Journal of Computer Vision*, 126(2-4):375–389, 2018. 5, 7, 8, 21, 28, 29, 36, 44, 48, 54

[230] J. Yu, Z. Wang, V. Vasudevan, L. Yeung, M. Seyedhosseini, and Y. Wu. CoCa: Contrastive Captioners are Image-Text Foundation Models. *arXiv preprint arXiv:2205.01917*, 2022. 9

[231] S. Yu, J. Cho, P. Yadav, and M. Bansal. Self-chained image-language model for video localization and question answering. *Advances in Neural Information Processing Systems*, 36, 2024. 74, 109, 110

[232] T. Yu, Z. Ren, Y. Li, E. Yan, N. Xu, and J. Yuan. Temporal structure mining for weakly supervised action detection. In *ICCV*, pages 5522–5531, 2019. 8

[233] L. Yuan, D. Chen, Y.-L. Chen, N. Codella, X. Dai, J. Gao, H. Hu, X. Huang, B. Li, C. Li, et al. Florence: A New Foundation Model for Computer Vision. *arXiv preprint arXiv:2111.11432*, 2021. 3, 9, 55

[234] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 6023–6032, 2019. 37, 41

[235] Y. K. Yun and W. Lin. Selfreformer: Self-refined network with transformer for salient object detection. *IEEE Transactions on Multimedia*, 2023. 81

[236] A. Zeng, A. Wong, S. Welker, K. Choromanski, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, J. Lee, V. Vanhoucke, et al. Socratic Models: Composing Zero-Shot Multimodal Reasoning with Language. *arXiv preprint arXiv:2204.00598*, 2022. 3, 4, 9, 55, 56, 58, 110

[237] X. Zhai, X. Wang, B. Mustafa, A. Steiner, D. Keysers, A. Kolesnikov, and L. Beyer. LiT: Zero-Shot Transfer with Locked-image text Tuning. In *CVPR*, pages 18123–18133, 2022. 3, 5, 55, 58

[238] Y. Zhai, L. Wang, W. Tang, Q. Zhang, N. Zheng, and G. Hua. Action coherence network for weakly-supervised temporal action localization. *IEEE Transactions on Multimedia*, 24:1857–1870, 2021. 8

[239] B. Zhang, J. Yu, C. Fifty, W. Han, A. M. Dai, R. Pang, and F. Sha. Co-training Transformer with Videos and Images Improves Action Recognition. *arXiv preprint arXiv:2112.07175*, 2021. 67, 68

[240] C. Zhang, T. Yang, J. Weng, M. Cao, J. Wang, and Y. Zou. Unsupervised pretraining for temporal action localization tasks. In *CVPR*, pages 14031–14041, 2022. 8

[241] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018. 37, 39, 41

[242] L. Zhang, A. Rao, and M. Agrawala. Adding conditional control to text-to-image diffusion models. In *IEEE International Conference on Computer Vision*, 2023. 10, 103, 105

[243] Y. Zhang, Y. Wei, D. Jiang, X. Zhang, W. Zuo, and Q. Tian. Controlvideo: Training-free controllable text-to-video generation. *International Conference on Learning Representations*, 2024. 2, 3, 10, 75, 77, 78, 84, 86, 87, 88, 93, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 111

[244] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, X. Tang, and D. Lin. Temporal action detection with structured segment networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2914–2923, 2017. 8

[245] Y. Zhao, I. Misra, P. Krähenbühl, and R. Girdhar. Learning Video Representations from Large Language Models. *arXiv preprint arXiv:2212.04501*, 2022. 9

[246] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2024. 110

[247] D. Zhou, W. Wang, H. Yan, W. Lv, Y. Zhu, and J. Feng. Magicvideo: Efficient video generation with latent diffusion models. *arXiv preprint arXiv:2211.11018*, 2022. 10

[248] K. Zhou, J. Yang, C. C. Loy, and Z. Liu. Learning to Prompt for Vision-Language Models. *IJCV*, 130(9):2337–2348, 2022. 9

[249] Y. Zhu, Y. Long, Y. Guan, S. Newsam, and L. Shao. Towards Universal Representation for Unseen Action Recognition. In *CVPR*, pages 9436–9445, 2018. 67

[250] D. Zhukov, J.-B. Alayrac, I. Laptev, and J. Sivic. Learning Actionness via Long-range Temporal Order Verification. In *ECCV*, pages 470–487. Springer, 2020. 8

[251] M. Zolfaghari, K. Singh, and T. Brox. Eco: Efficient convolutional network for online video understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 695–712, 2018. 8