Weakly-Guided Self-Supervised Pretraining for Temporal Activity Detection

Kumara Kahatapitiya$^{1*}$, Zhou Ren$^{2}$, Haoxiang Li$^{2}$, Zhenyu Wu$^{2}$, Michael S. Ryoo$^{1}$, Gang Hua$^{2}$

$^{1}$Stony Brook University
$^{2}$Wormpex AI Research

Abstract

Temporal Activity Detection aims to predict activity classes per frame, in contrast to video-level predictions in Activity Classification (i.e., Activity Recognition). Due to the expensive frame-level annotations required for detection, the scale of detection datasets is limited. Thus, commonly, previous work on temporal activity detection resorts to fine-tuning a classification model pretrained on large-scale classification datasets (e.g., Kinetics-400). However, such pretrained models are not ideal for downstream detection, due to the disparity between the pretraining and the downstream fine-tuning tasks. In this work, we propose a novel weakly-guided self-supervised pretraining method for detection. We leverage weak labels (classification) to introduce a self-supervised pretext task (detection) by generating frame-level pseudo labels, multi-action frames, and action segments. Simply put, we design a detection task similar to downstream, on large-scale classification data, without extra annotations. We show that the models pretrained with the proposed weakly-guided self-supervised detection task outperform prior work on multiple challenging activity detection benchmarks, including Charades and MultiTHUMOS. Our extensive ablations further provide insights on when and how to use the proposed models for activity detection. Code is available at github.com/kkahatapitiya/SSDet.

Introduction

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 (Mahajan et al. 2018; Dai et al. 2021b), segmentation (Poudel, Liwicki, and Cipolla 2019), video understanding (Ghadiyaram, Tran, and Mahajan 2019), reinforcement learning (Schwarz et al. 2021) and language modeling (Liu et al. 2020). 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 pre-trained task and the downstream task (Abnar et al. 2022). 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 (Deng et al. 2009) (image domain) or Kinetics-400 (Carreira and Zisserman 2017) (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 pretraining on activity classification improves downstream detection perfor-
Figure 2: Performance comparison between models pretrained for classification and the proposed weakly-guided self-supervised detection, on downstream Charades (Sigurdsson et al. 2016) 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 $\rightarrow$ Detection-pretrained $\rightarrow$ Detection-pretrained (Ensemble). Model names are shown for Classification pretrained versions in space (red circles).

mance, 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 (Piergiovanni and Ryoo 2018, 2019; Kahatapitiya and Ryoo 2021) or graphical (Ghosh et al. 2020; Mavroudi, Haro, and Vidal 2020) 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 disparity effectively.

In this work, we propose a weakly-guided self-supervised pretraining method 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. 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 (Sigurdsson et al. 2016) (see Fig. 2) and MultiTHUMOS (Yeung et al. 2018), 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.

Related Work

Video understanding: Spatio-temporal (3D) convolutional architectures (CNNs) are commonly used for video modeling (Tran et al. 2015; Carreira and Zisserman 2017; Xu, Das, and Saenko 2017). Among these, multi-stream architectures fusing different modalities (Simonyan and Zisserman 2014; Feichtenhofer, Pinz, and Zisserman 2016) or different temporal resolutions (Feichtenhofer et al. 2019) have achieved state-of-the-art results. To improve the efficiency of video models, Neural Architecture Search (NAS) has also been explored recently in (Ryoo et al. 2020; Feichtenhofer 2020). Multiple other directions either try to take advantage of long-term motion (Yue-Hei Ng et al. 2015; Varol, Laptev, and Schmid 2017; Piergiovanni and Ryoo 2018), graphical modeling (Zhao, Thabet, and Ghanem 2021; Mavroudi, Haro, and Vidal 2020), object detections (Baradel et al. 2018; Zhou et al. 2019) or attention mechanisms (Chang et al. 2021; Fan et al. 2021) to improve video understanding.

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 (Shou, Wang, and Chang 2016; Escorcia et al. 2016; Buch et al. 2017; Yeung et al. 2016; Shou et al. 2017; Zhai et al. 2019; Tirupattur et al. 2021; Liu et al. 2021; Guo et al. 2022), and (2) Temporal Activity Detection which explicitly predicts classes per frame (Piergiovanni and Ryoo 2019; Kahatapitiya and Ryoo 2021; Dai et al. 2021a). We focus on the latter. Datasets for such tasks provide frame-level annotations with possibly multiple classes per frame (Caba Heilbron et al. 2015; Sigurdsson et al. 2016; Yeung et al. 2018).

Limited Supervision: This includes unsupervised (Sener and Yao 2018; Kukleva et al. 2019; Gong et al. 2020), self-supervised (Jain, Ghodrati, and Snoek 2020; Chen et al. 2020a), weakly-supervised (Sun et al. 2015) or semi-supervised (Ji, Cao, and Niebles 2019) settings, based on the level of annotations used (Chen et al. 2022). Self-supervision in particular, explores two directions: pretext tasks (Misra, Zitnick, and Hebert 2016; Wei et al. 2018; Purushwalkam et al. 2020; Zhukov et al. 2020; Recasens et al. 2021) or contrastive learning (He et al. 2020; Chen et al. 2020b; Chen and He 2021).
Prior work on temporal activity localization has explored limited supervision either during pretraining (Zhang et al. 2022; Xu et al. 2021a; Alwassel, Giancola, and Ghanem 2021; Xu et al. 2021b), or the downstream (Richard, Kuehne, and Gall 2017; Nguyen et al. 2018; Liu et al. 2019; Yu et al. 2019; Liu, Jiang, and Wang 2019; Shi et al. 2020). 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 however, is with the flavor of frame-level predictions (activity detection), rather than predicting temporal proposals with boundaries and class labels (TAL). Thus, ours is orthogonal to above work on pretraining, but can be complementary to those on downstream finetuning.

**Weakly-guided Self-supervised 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 (Carreira and Zisserman 2017). 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. 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) as downstream detection. Next, we will introduce our pseudo labeling, volume augmentations, and how we combine these ideas.

**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 (Carreira and Zisserman 2017), 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 (Caba Heilbron et al. 2015) with multiple actions per clip, instead of Kinetics–400 (Carreira and Zisserman 2017) 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.

**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. 3 (top) and elaborated in Fig. 4. It 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 (Zhang et al. 2018). However, here we focus more on preserving the temporal 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. 3 (middle) and elaborated in Fig. 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,

\[
\alpha[t] = \begin{cases} 
T_{[0,1]}(n_1 - t) & \text{if } n_2 + r \geq n_1, \\
T_{[0,1]}(n_2 + 2r - 2t) & \text{otherwise}, 
\end{cases}
\]

The truncation operator \(T_{[0,1]}(\cdot)\) 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 Clip1 \(\rightarrow\) Clip2, whereas in scenario 2, it works as Clip1 \(\rightarrow\) Clip2 \(\rightarrow\) Clip1. 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 (Zhang et al. 2018).

**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 (Yun et al. 2019) 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. 3 (bottom) and Fig. 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 \(M\) 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
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.

Although the latter strategy seems flexible, applying multiple augmentations per clip is not computationally efficient. With the same compute budget as in Volume MixUp, we use either (1) joint training or (2) model ensembling. In Joint training, we combine the three augmentations during training. A simpler setting is to apply only one 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.

In Model ensembling, we apply only a single selected augmentation among the proposed Volume Freezing, MixUp, and CutMix during training. At inference, we combine predictions coming from these 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.
Experiments

To validate the benefits of our proposed method, we pretrain on commonly-used Kinetics-400 (Carreira and Zisserman 2017) and evaluate on rather-complex Charades (Sigurdsson et al. 2016) and MultiTHUMOS (Yeung et al. 2018) for downstream detection, using the efficient video backbone X3D (Feichtenhofer 2020). In addition to applying the proposed augmentations at the input level, we also run a few experiments with manifold augmentations (Verma et al. 2019), where each augmentation method is applied to the feature maps at a random depth of the network.

Kinetics-400 Detection Pretraining

By default, we initialize with our backbone X3D-M (medium) with checkpoints provided in original work (Feichtenhofer 2020), 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).

Charades Evaluation

We initialize X3D (Feichtenhofer 2020) 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 (Kahatapitiya and Ryoo 2021). 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 (Sigurdsson et al. 2016) 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).

<table>
<thead>
<tr>
<th>Model</th>
<th>Mod</th>
<th>Pretrain cls.</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-stream 3D (Carreira et al.)</td>
<td>R+F</td>
<td>✓</td>
<td>17.22</td>
</tr>
<tr>
<td>3D ResNet-50 (He et al.)</td>
<td>R</td>
<td>✓</td>
<td>18.60</td>
</tr>
<tr>
<td>STGCN (Ghosh et al.)</td>
<td>R+F</td>
<td>✓</td>
<td>19.09</td>
</tr>
<tr>
<td>VS-ST-MPN (Mavroudi et al.)</td>
<td>R+O</td>
<td>✓</td>
<td>23.70</td>
</tr>
<tr>
<td>MS-TCT (Dai et al.)</td>
<td>R</td>
<td>✓</td>
<td>25.40</td>
</tr>
<tr>
<td>PDAN (Dai et al.)</td>
<td>R+F</td>
<td>✓</td>
<td>26.50</td>
</tr>
<tr>
<td>X3D (Feichtenhofer)</td>
<td>R</td>
<td>✓</td>
<td>(22.36) 23.94</td>
</tr>
<tr>
<td>SE* (Piergiovanni et al.)</td>
<td>R</td>
<td>✓</td>
<td>(22.24) 23.92</td>
</tr>
<tr>
<td>TGM + SE* (Piergiovanni et al.)</td>
<td>R</td>
<td>✓</td>
<td>23.84</td>
</tr>
<tr>
<td>SlowFast\text{det} (Feichtenhofer et al.)</td>
<td>R</td>
<td>✓</td>
<td>(24.11) 25.50</td>
</tr>
<tr>
<td>Coarse-Fine (Kahatapitiya et al.)</td>
<td>R</td>
<td>✓</td>
<td>(24.73) 25.32</td>
</tr>
</tbody>
</table>

Table 1: Performance on Charades (Sigurdsson et al. 2016). 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 *.

Results: We report the performance of state-of-the-art methods comparing their pretraining strategy in Table 1. These numbers are for the Charades standard evaluation protocol (Sigurdsson et al. 2016). We see a clear improvement from the model ensembles pretrained with the proposed detection task across multiple methods. The vanilla X3D (Feichtenhofer 2020) 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 (Piergiovanni and Ryoo 2018) with +2.13% mAP and in TGM (Piergiovanni and Ryoo 2019) with +1.60% mAP improvement. Finally, we see the benefits in fully end-to-end trained multi-stream networks such as SlowFast\text{det} (+2.52% mAP) and Coarse-Fine Networks (Kahatapitiya and Ryoo 2021) (+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 (Dai et al. 2021a).
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 and report using mAP. However, when combining augmentations, ensembles work best compared to joint-training which can create confusing inputs with multiple augmentations.

Table 2: Ablations on Charades (Sigurdsson et al. 2016) 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).

![Table 2](image)

| Ablations | In Table 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 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 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.

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 and report using mAP.

Table 3: Performance on MultiTHUMOS (Yeung et al. 2018). 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 variants with X3D backbone are denoted with *.

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.


