Self-supervised Pretraining with Classification Labels
for Temporal Activity Detection

Kumara Kahatapitiya\textsuperscript{1,*}, Zhou Ren\textsuperscript{2}, Haoxiang Li\textsuperscript{2}, Zhenyu Wu\textsuperscript{2} and Michael S. Ryoo\textsuperscript{1}
\textsuperscript{1}Stony Brook University \textsuperscript{2}Wormpex AI Research

Abstract

Temporal Activity Detection aims to predict activity classes per frame, in contrast to video-level predictions as done 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 performance due to the disparity between the pretraining and the downstream fine-tuning tasks. This work proposes a novel self-supervised pretraining method for detection leveraging classification labels to mitigate such disparity by introducing frame-level pseudo labels, multi-action frames, and action segments. We show that the models pretrained with the proposed 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 and models will be released online.

1. 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 [14, 44], segmentation [54], video understanding [21], reinforcement learning [60] and language modeling [41]. 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 [5].

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 [15] (image domain) or Kinetics-400 [7] (video domain), which often leaves a disparity between pretraining and downstream tasks.

To mitigate such disparity, several works have been proposed for image domain tasks. For instance, in 3D scene understanding tasks such as object pose estimation and depth estimation, [48] proposed self-supervised learning to generate synthetic data for pretraining. [88] proposed a self-training method for object detection and segmentation, by designing pretext tasks similar to the downstream setting.

Similarly, in temporal activity detection that is defined as predicting (one or more) activity classes per frame, we have the same observation: although pretraining on activity classification improves downstream detection performance,
we first extend video-level labels of classification clips (with overlapping actions in fine detail. However, it can be difficult to go beyond making a classification decision per segmented video, by annotating every frame with multiple ongoing activities. Use of sequential models such as RNNs have been popular [6, 17, 77–79], and fully convolutional approaches also showed promising results [63, 64, 76, 85].

Limited Supervision: In contrast to supervised methods, limited supervision requires no or partial annotations. It includes weakly-supervised [38, 40, 42, 49, 50, 58, 62, 67], unsupervised [24, 39, 61], self-supervised [9, 30], and semi-supervised [31] settings. In particular, we are interested in self-supervision, which explores two directions: pretext tasks or contrastive learning. Pretext tasks include generative modeling [12, 51, 83], predicting spatial structure [2, 16, 23, 34, 36], temporal structure [46, 55, 74, 75, 87] or different views [25, 57]. On the other hand, contrastive learning [10, 11, 27] in video focus on contrasting between different spatio-temporal [3, 29, 56] or multi-modal [26, 37] views.

This work focuses on designing a self-supervised pretext task for temporal activity detection, using only video-level classification labels.

3. Self-supervised Pretraining with Classification Labels for Activity Detection

We introduce a self-supervised pretraining task for activity detection, which uses classification data and no additional labels. This idea is primarily motivated by 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 [7]. This is because (1) video models need large-scale data to mitigate overfitting during training, and (2) detection annotations (frame-level) are too expensive.

Figure 2. Performance comparison between models pretrained for classification and the proposed self-supervised detection, on downstream Charades [65] activity detection setting. Our model ensembles pretrained with 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 place (red circles).

it is limited by the disparity, since in activity classification pretraining, a model can aggregate temporal information, learning to only look at the bigger picture of an input clip, while the downstream activity detection task is more fine-grained, which requires the model to retain temporal information as much as possible, looking at the composition of atomic actions.

As for video understanding tasks, it is challenging to mitigate the pretraining and downstream disparity. Previous works have proposed specific temporal [35, 52, 53] or graphical [22, 45] modeling. Such additional modeling focus on capturing 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 those modeling techniques to alleviate the data disparity.

In this work, we propose a self-supervised pretraining method for activity detection by using classification data only and no additional labels. We generate pretraining data to capture fine-grained details in a multi-action-per-frame setting and use detection as the pretraining task — a step closer to the downstream detection (see Fig. 1). Specifically, we first extend video-level labels of classification clips (with a single action per clip) to every frame, creating frame-level pseudo labels. Then, we propose three self-supervised augmentation techniques to generate multi-action frames and action segments within a clip; namely, 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 activity detection task. As shown in Fig. 2, our evaluations validate the benefits of the proposed pretraining strategy on multiple temporal activity detection benchmarks, including Charades [65] and MultiTHUMOS [77]. We further investigate the extent of the detection-pretrained features in our ablations and recommend when and how to use them best.
Figure 3. **Volume Augmentations for our self-supervised detection pretraining:** Volume Freeze, Volume MixUp and Volume CutMix. We first extend video-level labels (of single-action videos from Kinetics-400 [7]) 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 [81, 82]. 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.

sive 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 have only a single action per clip with video-level annotations, whereas, in the downstream detection task, one needs to predict multiple actions for 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 pretraining task that closely resembles the downstream task. 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 as downstream detection. In the following subsections, we will introduce our pseudo labeling, volume augmentations, and how we combine these ideas.

### 3.1. Frame-level Pseudo Labels

Downstream detection is about fine-grained predictions of activity classes, which requires frame-level annotations to train. However, in the pretraining data that we consider (Kinetics-400 [7]), each clip contains a single action and a video-level label. Since we want to design a pretraining task as close as possible to the downstream detection, we create 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 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 creating frame-level pseudo labels for classification data.

### 3.2. Self-Supervised Volume Augmentations

Based on the frame-level pseudo labels, we design a self-supervised detection task 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: (1) Volume Freeze, (2) Volume MixUp, and (3) Volume CutMix. Next, we will explain these concepts in detail.

#### 3.2.1 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 contain only a single action per clip, in which, it is a challenge to have such segments. Our solution here is to create an action-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. Here, we assume that no action can be performed without any motion and label the frozen segment with a new background label. Volume Freeze augmentation is shown in Fig. 3 (top) and elaborated Fig. 4. It can be denoted as follows:

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

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

where \(VF(v)\) and \(VF(l)\) denote the augmented video and associated label using 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 \(n\). 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 gives only a small gain.

### 3.2.2 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 [32]. 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 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,

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

The “truncation” operator \(T_{[0, 1]}(\cdot)\) clips the mask values within the range of \([0, 1]\). More detailed definition is in Appendix A.1. In our case, it makes \(\alpha[t]\) to be a piecewise linear function w.r.t. \(t\) in the overlapping segment between two clips, and makes \(\alpha[t]\) of the non-overlapping segment to be either 0 or 1, as illustrated in Fig. 5. In scenario 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 \(r\).

### 3.2.3 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 [81] 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). We show an example in Fig. 3 (bottom) and elaborate in Fig. 6.
Transient View: In Transient View, we keep the window size \( \frac{1}{2} \) of a frame may cause problems for convolution kernels. With the same notations as before, the augmented clip and labels can be denoted as,

\[
\text{VC}(v_1, v_2)[t] = M \odot v_1[t] + (1 - M) \odot v_2[t],
\]
\[
\text{VC}(l_1, l_2)[t] = 0.5 \cdot l_1[t] + 0.5 \cdot l_2[t].
\]

### 3.3. Combining Augmentations

In the previous subsections, we defined the components of the proposed pretraining scheme: namely, frame-level pseudo labeling and volume augmentations. We always use the pseudo labeling method to generate frame-level labels in classification data. When considering the augmentations, we try two strategies to combine them: (1) joint training and (2) model ensembling.

**Joint training:** Here, we combine the three augmentations during training. Each augmentation is randomly applied to a given batch of input clips. Thus, an augmented sample may see no augmentation, or up to all three augmentations. Although this 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:** In this strategy, we apply only a single 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 method, without worrying about the input confusion at training. However, this incurs more compute requirement at inference, compared to a jointly trained single model.

### 4. Experiments

To validate the benefits of our proposed method, we pretrain on Kinetics-400 [7] and evaluate on Charades [65] and MultiTHUMOS [77] for downstream detection, using the efficient video backbone X3D [18]. In addition to applying the proposed augmentations at the input level, we also run a few experiments with manifold augmentations [73], where each augmentation method is applied to the feature maps at a random depth of the network.

#### 4.1. Kinetics-400 Detection Pretraining

**Dataset:** Kinetics-400 [7] is a large-scale activity classification dataset commonly used for pretraining video models. It contained 240k training and 20k validation videos at release, but due to the unavailability of some videos, our version contains \(~220k\) training and \(~17k\) 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.

**Pretraining:** We start with an X3D-M (medium) [18] checkpoint pretrained for classification on Kinetics-400 [7]. This allows faster adoption and shorter pretraining schedules.
for our self-supervised detection. We pretrain X3D for 100k iterations with a batch size of 64 and an initial learning rate of 0.05, and reduce the learning rate by a factor of 10 after 80k iterations. We use a dropout rate of 0.5 before the last fully-connected layer. 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.

4.2. Charades Evaluation

Dataset: Charades [65] 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.

Training and Inference: We initialize X3D [18] 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 SlowFastdet, we follow the same two-staged training strategy as in [35]. Namely, each stream is first trained separately, followed by joint training with newly-initialized fusion parameters. We train all methods on Charades with Binary Cross-Entropy (BCE) as localization and classification losses. 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 [65] 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, and do not perform such class-dependent weighting. The detection performance here is measured by mean Average Precision (mAP).

Results: We report the performance of state-of-the-art methods comparing their pretrained strategy in Table 1. These numbers are for the Charades standard evaluation protocol [65]. We see a clear improvement from the model ensembles pretrained with the proposed detection task across multiple methods. The vanilla X3D [18] 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 [52] with a +2.13% mAP and in TGM [53] with a +1.66% mAP improvement. Finally, we see the benefits in fully end-to-end trained multi-stream networks such as SlowFastdet (+2.52% mAP) and Coarse-Fine Networks [35] (+1.85% mAP). Note that SlowFastdet here is a variant of original SlowFast [19], which is adopted with X3D [18] for detection in [35]. We further show the performance vs. compute evaluation for these methods in Fig. 2. Our models, even without ensembling, achieves significant gains. Our model ensembles increase the compute requirement compared to classification pretrained single models, however, these are still more than an order of magnitude efficient compared to the previous best performing model PDAN [13].

4.3. 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 [65] (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 2a, Volume Freezing provides a relative improvement of +1.51% mAP over classification pretrained X3D
| Volume Freeze | mAP (%) | Volume MixUp | mAP (%) | Volume CutMix | mAP (%) | Method | mAP (%) |
|---------------|---------|-------------|---------|--------------|---------|--------|---------|
| Baseline (cls.) | 17.28   | Baseline (cls.) | 17.28   | Baseline (cls.) | 17.28   | Baseline (cls.) | 17.28   |
| Single segment | 18.79 | Hard boundaries | 18.86 | Hard boundaries | 18.89 | Joint train (VF/VM/VC) | 19.11 |
| Two segments   | 18.83 | Seamless | 19.18 | Seamless (manifold) | 19.24 | Ensemble (VF/VM/VC) | 20.50 |

(a) Volume Freeze with a single or two separate frozen segments. Multiple frozen segments does not give a considerable benefit. — X3D [18]

(b) Volume MixUp with hard or seamless (soft) boundaries (i.e., changing alpha values). Seamless boundaries preserve temporal consistency and works better. MixUp at [73] does not give a considerable benefit. — X3D [18]

(c) Volume CutMix with transient windows or transient views (with a constant window for each clip). Transient views show a slightly better performance. — X3D [18]

(d) Combining Augmentations with joint-training or as an ensemble of separately-trained models. Joint-training can create confusing inputs with multiple augmentations. Avoiding such, ensembling can obtain the best of each method. — X3D [18]

| Pretraining | Act. per frame | Boundary |
|-------------|---------------|----------|
|             | =1            | >1       | False | True |
| cl.s.       | Act. (VF)     | Act. (VM) | Act. (VC) |

- det. (VF) |-0.14| 8.39 | 20.57 | 17.34 | 16.18 |
- det. (VM) |+1.30| 9.93 | 20.78 | 18.91 | 17.96 |
- det. (VC) |+1.16| 9.79 | 20.58 | 18.02 | 18.03 |

Statistics from the validation set, which show the improvement for single vs. multi-action frames, and action boundary vs. non-boundary regions (for a boundary considered with a dilation of 3). We see a consistently larger improvement in multi-action frames compared to single-action frames, as our pretraining introduces multi-action frames. However, improvement from introducing boundaries in a pretraining has a subtle impact in the downstream detection. — X3D [18]

| Pretraining | Fine/Fast | Coarse | Slow |
|-------------|-----------|--------|------|
|             | Act. per frame |                   |       |
|             | =1            | >1       | False | True |
| cl.s.       | Act. (VF)     | Act. (VM) | Act. (VC) |

- det. (VF) |17.28 | 18.79 | 18.99 | 17.22 |
- det. (VM) |19.18 | 18.99 | 19.00 | 18.78 |
- det. (VC) |18.99 | 18.83 | 18.83 | 18.60 |

(e) At lower temporal resolutions (< pretraining resolution), detection pretrained models are not improved as much as classification pretrained ones, by comparing the Coarse/Slow stream vs. Fine/Fast stream. Classification captures an overview of a clip which can be better generalized to different temporal resolutions. However, detection pretrained models still consistently outperform others. — SlowFast [19]/ Coarse-Fine [35]

(f) Performance of multi-stream architectures with streams pretrained with different methods. Here, we see an interesting observation: even though detection pretrained models are consistently better as single-stream networks (e.g: either Coarse/Slow or Fine/Fast), when combined as multi-stream networks, performance varies. We further investigate why this happens in Table 2g and Table 2h. Model ensembles however, gives consistent improvements as expected. — SlowFast [19]/ Coarse-Fine [35]

(g) In temporal aggregation, classification pretrained models perform better. We show this with the fusion module in Coarse-Fine [35], which aggregates Fine features with Gaussians at a given standard deviation. Here, if we increase the standard deviation, we temporally aggregate (dilate) more. Classification pretrained features show consistently higher improvement with such aggregation. — Coarse-Fine [35]

Table 2. Ablations on Charades [65] activity detection, evaluating our design choices and showing when our detection pretrained models can be most beneficial (i.e., relative improvement from detection pretrained models are not as much as their counterparts at different temporal resolutions (Table 2g) or strong temporal aggregation (Table 2h)). Here, We show the performance in mean Average Precision (mAP) for fine-grained predictions (i.e., making decisions per frame rather than evenly-sampled 25 frames from each validation clip). Relative changes of negative, positive-but-small and positive are shown in corresponding color, whereas the best performances that we highlight are underlined. The specific model/models used for evaluation in each table is mentioned at the end of each caption.

In general, we consider a single random frozen segment in a given clip. Having multiple such frozen segments does not give a considerable boost (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 2b. Volume MixUp applied in a random feature-level as in [73] 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 2c.

Combining augmentations: As illustrated in Table 2d, when combining the three volume augmentation methods, an ensemble of models pretrained separately works considerably better (+1.39% mAP) than a single model jointly-trained with all augmentations. The problem with combining multiple augmentations in a single input instance is that, it may result in a confusing/cluttered input to the network. In
contrast, each model in an ensemble can avoid such clutter by training separately, while complementing other models within the ensemble at inference.

Statistics showing the points of improvement: Table 2e 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 the multi-action setting, as we introduce multi-action frames in pre-training. Even though we introduce action boundaries during pretraining, it does not show a contrasting change between boundary and non-boundary regions in the downstream.

Multi-stream methods and ensembling: We see some interesting results when combining multiple streams of detection pretrained models. As shown in Table 2f, SlowFastdet [19], shows a clear improvement (+0.91% mAP) with two detection pretrained streams, whereas Coarse-Fine [35] does not (−0.46% mAP). However, when a classification pretrained model is used as the Fine stream, it works better (+0.28% mAP) than the original. This gives an intriguing observations on detection models (1) at different temporal resolutions and, (2) in temporal aggregation, which are further explored in Table 2g and Table 2h, respectively. When considering model ensembles, in Coarse-Fine, a classification pretrained Fine stream fused with detection pretrained Coarse stream ensembles (VF/VM/VC) gives a +0.72% mAP improvement over a single Coarse-Fine network. If we further include, Fine stream ensembles (VF/VM/VC), it gives an additional boost of +0.32% mAP.

At different temporal resolutions: In Table 2g, 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 [35], fusion module aggregates Fine features with Gaussians (at defined standard deviation). As in Table 2h, by evaluating at different standard deviations (different aggregation scales), we see that classification pretrained features give a better temporal aggregation compared to detection counterparts. This is due to the same reason mentioned above: classification features capture an overview, hence better generalize across scales.

4.4. MultiTHUMOS Evaluation

Dataset: MultiTHUMOS [77] is a small-scale dataset which contains a subset of THUMOS [33] 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 [65], this has a significantly smaller number of videos, but each clip is longer in duration.

Training and Inference: We follow the same training recipe as in Charades, starting with a checkpoint pretrained for our detection. At inference, we make predictions for every frame and report mean Average Precision (mAP).

Results: In Table 3, we present the performance of state-of-the-art models pretrained with either classification task or the proposed detection task. Detection pretrained models consistently outperform classification pretrained ones, in vanilla backbones such as X3D [18] (+3.71% mAP), and with temporal modeling on-top of pre-extracted features as in TGM [53] (+3.99% mAP) or PDAN [13] (+5.15% mAP). PDAN, with our pretraining, significantly efficient X3D backbone and only RGB modality achieves competitive performance compared to multi-modal I3D [7] counterparts.

5. Conclusion

This work introduced a new self-supervised pretraining strategy for temporal activity detection, only using classification labels. We defined a self-supervised pretraining detection 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 benchmarks. As takeaways, we further provide recommendations on when to use such pretrained models based on our observations.

| Model                  | Modality | Pretraining | mAP (%) |
|------------------------|----------|-------------|---------|
| Two-stream I3D [7]     | R+F      | ✓           | 36.40   |
| I3D + TGM + super-events [53] | R+F | ✓ | 46.40 |
| I3D + PDAN [13]        | R+F      | ✓           | 47.60   |
| X3D [18]               | R        | ✓           | 39.17 (+3.71) 40.88 |
| X3D + TGM + super-events [53] | R | ✓ | 39.16 (+3.99) 43.15 |
| X3D + PDAN [13]        | R        | ✓           | 39.20 (+5.15) 44.35 |

Table 3. Comparison with the state-of-the-art methods for activity detection on MultiTHUMOS [77]. We report the performance (mAP), input modalities used (R: RGB or F: optical flow), and the pretraining method: classification (cls.) or the proposed self-supervised 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 pretraining are in bold with relative improvements in green, while the best performance from each pretraining strategy is underlined.
A. Appendix

A.1. On the Truncation Operator $T_{[0,1]}$

As shown in Fig. A.1 (left), the truncation operator $T_{[0,1]}$ makes sure that the alpha mask ($\alpha_1[t]$) is within the range of $[0,1]$, even in the non-overlapping region. It can be defined as,

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

where any value $x \geq 1$ or $x < 0$ is capped at either 1 or 0 respectively. Based on this, alpha mask $\alpha_1[t]$ is defined so that the augmented clips have a smooth transition as Clip$_1$ $\rightarrow$ Clip$_2$ (in Scenario 1), or as Clip$_1$ $\rightarrow$ Clip$_2$ $\rightarrow$ Clip$_1$ (in Scenario 2).

A.2. On the Spatial Mask $M(t)$

Spatial mask $M$ defines a vertical plane to split each frame within the overlapping region into two windows (see Fig. A.1 (right)). The location of this vertical plane ($w_t$) can either depend on $\alpha_1[t]$ (in Transient Window) or be constant (in Transient View). This can be given as,

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

and,

$$ w_t = \begin{cases} 
\lfloor W \alpha_1[t] \rfloor & \text{if Transient Window,} \\
\lfloor W/2 \rfloor & \text{if Transient View,}
\end{cases} $$

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

References

[1] Samira Abnar, Mostafa Dehghani, Behnam Neyshabur, and Hanie Sedghi. Exploring the Limits of Large Scale Pre-training. arXiv preprint arXiv:2110.02095, 2021.

[2] Unaiza Ahsan, Rishi Madhok, and Irfan Essa. Video Jigsaw: Unsupervised Learning of Spatiotemporal Context for Video Action Recognition. In WACV, pages 179–189. IEEE, 2019.

[3] Yutong Bai, Haoqi Fan, Ishan Misra, Ganesh Venkatesh, Yongyi Lu, Yuyin Zhou, Qihang Yu, Vikas Chandra, and Alan Yuille. Can Temporal Information Help with Contrastive Self-Supervised Learning? arXiv preprint arXiv:2011.13046, 2020.

[4] Fabien Baradel, Natalia Neverova, Christian Wolf, Julien Mille, and Greg Mori. Object Level Visual Reasoning in Videos. In ECCV, pages 105–121, 2018.

[5] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the Opportunities and Risks of Foundation Models. arXiv preprint arXiv:2108.07258, 2021.
[38] Hilde Kuehne, Alexander Richard, and Juergen Gall. Weakly supervised learning of actions from transcripts. CVIU, 163:78–89, 2017. 2
[39] Anna Kakleva, Hilde Kuehne, Fadime Sener, and Jurgen Gall. Unsupervised Learning of Action Classes With Continuous Temporal Embedding. In CVPR, pages 1206–12074, 2019. 2
[40] Daochang Liu, Tingting Jiang, and Yizhou Wang. completeness Modeling and Context Separation for Weakly Supervised Temporal Action Localization. In CVPR, pages 1298–1307, 2019. 2
[41] Yinhua Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. In ICLR, 2020. 1
[42] Ziyi Liu, Le Wang, Qilin Zhang, Zhanning Gao, Zhenxing Niu, Nanning Zheng, and Gang Hua. Weakly Supervised Temporal Action Localization via Contrast Based Evaluation Networks. In ICCV, pages 3899–3908, 2019. 2
[43] Chih-Yao Ma, Asim Kadav, Iain Melvin, Zsolt Kira, Ghassan AlRegib, and Hans Peter Graf. Attend and Interact: Higher-Order Object Interactions for Video Understanding. In CVPR, pages 6790–6808, 2018. 2
[44] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. Exploring the Limits of Weakly Supervised Pretraining. In ECCV, pages 181–196, 2018. 1
[45] Effrosyni Mavroudi, Benjamín Béjar Haro, and René Vidal. Representation Learning on Visual-Symbolic Graphs for Video Understanding. In ECCV, pages 71–90. Springer, 2020. 2, 6
[46] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and Learn: Unsupervised Learning using Temporal Order Verification. In ECCV, pages 527–544. Springer, 2016. 2
[47] Megha Nawhal and Greg Mori. Activity Graph Transformer for Temporal Action Localization. arXiv preprint arXiv:2101.08540, 2021. 2
[48] Jia Newell, Alejandro aand Deng. How Useful is Self-Supervised Pretraining for Visual Tasks? In CVPR, 2020. 1
[49] Phuc Nguyen, Ting Liu, Gautam Prasad, and Bohyung Han. Weakly Supervised Action Localization via Sparse Temporal Pooling Network. In CVPR, pages 6752–6761, 2018. 2
[50] Phuc Xuan Nguyen, Deva Ramanan, and Charless C Fowlkes. Weakly-supervised Action Localization with Background Modeling. In ICCV, pages 5502–5511, 2019. 2
[51] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context Encoders: Feature Learning by Inpainting. In CVPR, pages 2536–2544, 2016. 2
[52] AJ Piergiovanni and Michael S Ryoo. Learning Latent Super-Events to Detect Multiple Activities in Videos. In CVPR, pages 5304–5313, 2018. 2, 6
[53] AJ Piergiovanni and Michael S. Ryoo. Temporal Gaussian Mixture Layer for Videos. In ICML, pages 5152–5161, 2019. 2, 6, 8
[54] Rudra PK Poudel, Stephan Liwicki, and Roberto Cipolla. Fast-SCNN: Fast Semantic Segmentation Network. In BMVC, 2019. 1
[55] Senthil Purushwalkam, Tian Ye, Saurabh Gupta, and Abhinav Gupta. Aligning Videos in Space and Time. In ECCV, pages 262–278. Springer, 2020. 2
[56] Rui Qian, Tianjian Meng, Boqing Gong, Ming-Hsuan Yang, Huisheng Wang, Serge Belongie, and Yin Cui. Spatiotemporal Contrastive Video Representation Learning. In CVPR, pages 6964–6974, 2021. 2
[57] Adrià Recasens, Pauline Luc, Jean-Baptiste Alayrac, Luyu Wang, Florian Strub, Corentin Tallock, Mateusz Malinowski, Viorica Pătrăucean, Florent Altché, Michal Valko, Jean-Bastien Grill, Airon van den Oord, and Andrew Zisserman. Broaden Your Views for Self-Supervised Video Learning. In ICCV, pages 1255–1265, October 2021. 2
[58] Alexander Richard, Hilde Kuehne, and Juergen Gall. Weakly Supervised Action Recognition with RNN based Fine-to-coarse Modeling. In CVPR, pages 754–763, 2017. 2
[59] Michael Ryoo, AJ Piergiovanni, Mingxing Tan, and Anelia Angelova. AssembleNet: Searching for Multi-Stream Neural Connectivity in Video Architectures. In ICLR, 2020. 2
[60] Max Schwarzer, Nitasharaj Rajkumar, Michael Noukhovitch, Ankes Anand, Laurent Charlin, Devon Hjelm, Philip Bachman, and Aaron Courville. Pretraining Representations for Data-Efficient Reinforcement Learning. In NeurIPS, 2021. 1
[61] Fadime Sener and Angela Yao. Unsupervised Learning and Segmentation of Complex Activities From Video. In CVPR, pages 8368–8376, 2018. 2
[62] Baifeng Shi, Qi Dai, Yadong Mu, and Jingdong Wang. Weakly-Supervised Action Localization by Generative Attention Modeling. In CVPR, pages 1009–1019. 2020. 2
[63] Zheng Shou, Jonathan Chan, AliReza Zareian, Kazuyuki Miyazawa, and Shih-Fu Chang. CDC: Convolutional-De-Convolutional Networks for Precise Temporal Action Localization in Untrimmed Videos. In CVPR, pages 5734–5743, 2017. 2
[64] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal Action Localization in Untrimmed Videos via Multi-stage CNNs. In CVPR, pages 1049–1058, 2016. 2
[65] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in Homes: Crowdsourcing Data Collection for Activity Understanding. In ECCV, pages 510–526. Springer, 2016. 2, 5, 6, 7, 8
[66] Karen Simonyan and Andrew Zisserman. Two-Stream Convolutional Networks for Action Recognition in Videos. In NeurIPS, pages 568–576, 2014. 2
[67] Chen Sun, Sanketh Shetty, Rahul Sukthankar, and Ram Neva-tia. Temporal Localization of Fine-Grained Actions in Videos by Domain Transfer from Web Images. In ACMMM, pages 371–380. 2015. 2
[68] Jing Tan, Jiaqi Tang, Limin Wang, and Gangshan Wu. Relaxed Transformer Decoders for Direct Action Proposal Generation. In ICCV, pages 13526–13535, October 2021. 2
[69] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning Spatiotemporal Features with 3D Convolutional Networks. In ICCV, pages 4489–4497, 2015. 2
[70] Du Tran, Jamie Ray, Zheng Shou, Shih-Fu Chang, and Manohar Paluri. ConvNet Architecture Search for Spatiotemporal Feature Learning. *arXiv preprint arXiv:1708.05038*, 2017. 2

[71] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A Closer Look at Spatiotemporal Convolutions for Action Recognition. In *CVPR*, pages 6450–6459, 2018. 6

[72] Gül Varol, Ivan Laptev, and Cordelia Schmid. Long-term Temporal Convolutions for Action Recognition. *PAMI*, 40(6):1510–1517, 2017. 2

[73] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold Mixup: Better Representations by Interpolating Hidden States. In *ICML*, pages 6438–6447. PMLR, 2019. 5, 7

[74] Donglai Wei, Joseph J Lim, Andrew Zisserman, and William T Freeman. Learning and Using the Arrow of Time. In *CVPR*, pages 8052–8060, 2018. 2

[75] Dejing Xu, Jun Xiao, Zhou Zhao, Jian Shao, Di Xie, and Yueting Zhuang. Self-Supervised Spatiotemporal Learning via Video Clip Order Prediction. In *CVPR*, pages 10334–10343, 2019. 2

[76] Huijuan Xu, Abir Das, and Kate Saenko. R-C3D: Region Convolutional 3D Network for Temporal Activity Detection. In *ICCV*, pages 5783–5792, 2017. 2

[77] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. Every Moment Counts: Dense Detailed Labeling of Actions in Complex Videos. *IJCV*, 126(2-4):375–389, 2018. 2, 5, 8

[78] Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei. End-to-End Learning of Action Detection from Frame Glimpses in Videos. In *CVPR*, pages 2678–2687, 2016. 2

[79] Jun Yuan, Bingbing Ni, Xiaokang Yang, and Ashraf A Kassim. Temporal Action Localization With Pyramid of Score Distribution Features. In *CVPR*, pages 3093–3102, 2016. 2

[80] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond Short Snippets: Deep Networks for Video Classification. In *CVPR*, pages 4694–4702, 2015. 2

[81] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features. In *ICCV*, pages 6023–6032, 2019. 3, 4

[82] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond Empirical Risk Minimization. In *ICLR*, 2018. 3, 4

[83] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful Image Colorization. In *ECCV*, pages 649–666. Springer, 2016. 2

[84] Chen Zhao, Ali K Thabet, and Bernard Ghanem. Video Self-Stitching Graph Network for Temporal Action Localization. In *ICCV*, pages 13658–13667, 2021. 2

[85] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaou Tang, and Dahua Lin. Temporal Action Detection with Structured Segment Networks. In *ICCV*, pages 2914–2923, 2017. 2

[86] Luowei Zhou, Yannis Kalantidis, Xinlei Chen, Jason J Corso, and Marcus Rohrbach. Grounded Video Description. In *CVPR*, pages 6578–6587, 2019. 2

[87] Dimitri Zhukov, Jean-Baptiste Alayrac, Ivan Laptev, and Josef Sivic. Learning Actionness via Long-range Temporal Order Verification. In *ECCV*, pages 470–487. Springer, 2020. 2

[88] Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin Cubuk, and Quoc V. Le. Rethinking Pre-training and Self-training. In *NeurIPS*, 2020. 1