PGT: A Progressive Method for Training Models on Long Videos

Bo Pang* Gao Peng* Yizhuo Li Cewu Lu†
Shanghai Jiao Tong University
{pangbo, penggao, liyizhuo, lucewu}@sjtu.edu.cn

Abstract

Convolutional video models have an order of magnitude larger computational complexity than their counterpart image-level models. Constrained by computational resources, there is no model or training method that can train long video sequences end-to-end. Currently, the main-stream method is to split a raw video into clips, leading to incomplete fragmentary temporal information flow. Inspired by natural language processing techniques dealing with long sentences, we propose to treat videos as serial fragments satisfying Markov property, and train it as a whole by progressively propagating information through the temporal dimension in multiple steps. This progressive training (PGT) method is able to train long videos end-to-end with limited resources and ensures the effective transmission of information. As a general and robust training method, we empirically demonstrate that it yields significant performance improvements on different models and datasets. As an illustrative example, the proposed method improves SlowOnly network by 3.7 mAP on Charades and 1.9 top-1 accuracy on Kinetics with negligible parameter and computation overhead. Code is available at: https://github.com/BoPang1996/PGT.

1. Introduction

Semantic information often flows across a long time in videos. However, end-to-end modeling a long video as a whole is not feasible for current convolutional methods since their computational complexities linearly increase with the number of frames [51]. The main-stream solution is splitting a video into multiple short clips [59, 60, 6, 34], but in this way, video models can only access local fragmentary temporal information, thus, fail to model long semantics [64, 51]. Is this trade-off between computational complexity and semantic integrity unavoidable, or might there be a specific training method tailored for video tasks that can model long semantics with acceptable complexity?

The main cause of this problem is that 3D convolutional models [53, 2, 54] treat a video signal $I(x, y, t)$ as an integrated information block and have to process it as a whole. With the video growing longer, the information block becomes larger and the processing complexity increases to an infeasible point. Since in the temporal dimension the development of video semantics has high-order Markov property, violently splitting long videos and processing short clips with convolutions to model local features will hurt semantic integrity. For example, the model will never know an action of "pour milk" is making latte, unless it also sees the previous action of "grinding coffee beans".

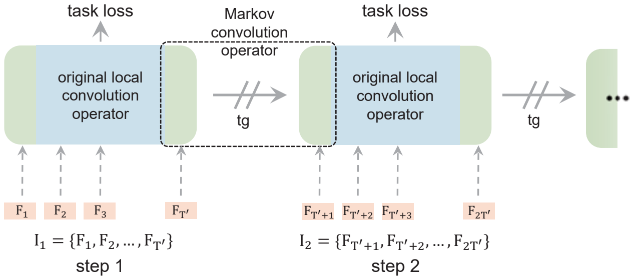
task loss
Markov convolution operator
task loss
original local convolution operator
tg
original local convolution operator
tg
•••
F_1
F_2
F_3
$F_{T'}$
$F_{T'+1}$
$F_{T'+2}$
$F_{T'+3}$
$F_{2T'}$
$I_1 = \{F_1, F_2, \ldots, F_{T'}\}$
step 1
$I_2 = \{F_{T'+1}, F_{T'+2}, \ldots, F_{2T'}\}$
step 2

Figure 1: **Progressive training** (PGT) treats videos as serial fragments and optimizes a CNN model with multiple progressive steps on long videos. The Markov convolutional operator designed to transfer temporal features among steps is adopted on the first and last frames of each step, and the gradient is truncated between them. "tg" denotes truncating gradients. F_i is the ith video frame and I_p is the input of the pth progressive step containing multiple frames.

To avoid the trade-off between computational complexity and semantic integrity — *i.e.*, to end-to-end train a model on long videos with much lower complexity, in this paper, we propose the progressive training (PGT) method (see Fig. 1). Inspired by Truncated Back-Propagation through Time (TBPTT) [63] originally designed for recurrent neural networks to model long natural language sequences, the central idea of PGT is to 1) treat a video as serial fragments satisfying high-order Markov property instead of an integrated signal block, 2) disassemble the integrated forward and backward propagation into multiple serial portions like TBPTT (see §2), which doesn't break the Markov dependency of the calculation flow. Modeling a long video in

*Equal contribution.

†Cewu Lu is the corresponding author, member of Qing Yuan Research Insitute, MoE Key Lab of Artificial Intelligence, AI Institute, CS department of Shanghai Jiao Tong University, and Qi Zhi Institute.

multiple steps won't lead to high resource consumption and the Markov property ensures the integrity of temporal semantics after disassembling, akin to how TBPTT enables training RNN on long sequences.

Because the common convolutional operator is a kind of local operator which does not satisfy the Markov property, we design several Markov convolutional operators with only a few modifications on the original convolutional operator so that they can easily replace the original one in modern video models when training. With these operators, the progressive training schedule mixing local and Markov features is proposed (see Fig. 2 and Fig. 3): The temporal information is propagated progressively forward in multiple steps where within each progressive step, local operators capture current features together with those transferred from previous steps through the Markov operators. In this schedule, the Markov operators propagate temporal information among the progressive steps throughout the temporal dimension and the serial multi-step splitting reduces the computational resource requirements.

The proposed PGT method is effective and pretty simple. It is easy to implement and typically requires small changes to a video model with negligible parameter or complexity overhead. Empirically, it works with default learning rate schedules and hyper-parameters already in use except for weight decay rates (longer inputs need stronger regularization). Extensive experiments show that the progressive training method works robustly out-of-the-box for different models (RegNet3D [40], ResNet [13], SlowFast [6]), datasets (Kinetics-200 [66], Kinetics-400 [20], Charades [42], AVA [11]), and training settings (e.g. from scratch or pre-trained). We observe consistent performance improvements without tuning. As an example, the progressive method improves SlowOnly network by 3.7 mAP on Charades and 1.9 top-1 accuracy on Kinetics. We hope this simple and effective method will provide the community with new insights into modeling long videos.

2. Related Work

Convolutional Video Networks Video convolutional backbones are developed from image-level backbone networks [13, 21, 44, 50, 33]. Inchoate methods [43, 8] directly apply image networks on optical-flow inputs [57, 38] to model temporal information. Then researchers extend 2D convolutional operators to 3D ones by extending the temporal dimension [53, 12] and 3D convolution based models [53, 2, 54, 66, 19, 29, 41, 9, 39, 49, 28] (including ones adopting 2D spatial plus 1D temporal filters) become the mainstream method. Recently, non-local network [60] is designed to model global video features instead of local ones. A two-stream structure SlowFast [6] is proposed to balance the spatial and temporal information. X3D [5] finds efficient video architectures by progressively extending each dimensions (e.g. spatial, temporal, channel dimensions). AssembleNet [41] proposes a method to automatically form connections among CNN blocks.

Methods to Handle Long Videos How to use modern convolutional networks to handle long video tasks is less studied, due to limited computational resources and the behaviour of the convolutional operator. One simple method is to use large sampling strides to represent a long video with only a few frames [59, 7, 26], which causes serious information loss. A better strategy is to model long videos on the top of pre-computed deep features of short clips [22, 31, 52, 69, 64, 51] by pooling [52, 31], CNN [69, 17], graph [16, 68], memory blocks [51], or attention methods [64]. However, because the features of short clips only contain local information and cannot be updated, these two-step methods without end-to-end training are likely suboptimal. Thus, we propose the first method to train deep CNN models end-to-end on long videos, requiring almost the same computational resources as short clips.

Sequential Methods for Video Tasks Modeling videos as sequences is an alternative strategy for convolutional models. [69, 4, 25, 48] adopt LSTM [14] layers (a kind of recurrent operator) to model video frame features generated by image-level CNN models. [35, 36] tailor better recurrent layers that are easy to stack deep for higher-dimensional video information. These recurrent-based methods have advantages over convolutional ones for tasks sensitive to sequence order, such as video future prediction [32, 56, 67], trajectory prediction [47], and video description [4, 55]. While for tasks that more focus on integrated features like action recognition [65, 25, 37, 2, 6, 24, 23], there is still a gap between the recurrent and convolutional models.

Truncated Back-Propagation through Time Back-Propagation Through Time, or BPTT [62], is the training algorithm used to update weights in recurrent neural networks like LSTMs [14]. It unrolls input timesteps (frames for video), calculates and accumulates errors for each timestep. Then the network is rolled back up to update the weights. But, for a long sequence with length n, BPTT will consume lots of memory and the complexity will be extremely large. Truncated Back-Propagation through Time (TBPTT) [63] solves this problem by periodically updating the network. It unrolls and forward propagates the input for k_1 steps, then backward propagates accumulated errors of the past k_2 unrolled steps to update the network ($k_1 \leq k_2 < n$). This process is repeated till unrolling all the inputs. Through splitting the integrated training process ($k_1 = k_2 = n$) into several sub-processes, TBPTT reduces the resource requirement. More importantly, because of the Markov property of RNN, this splitting does not change the flowing path of temporal information.

In this paper, inspired by TBPTT, we slightly modify the convolutional operator to satisfy the Markov property and design a progressive training method for long videos.

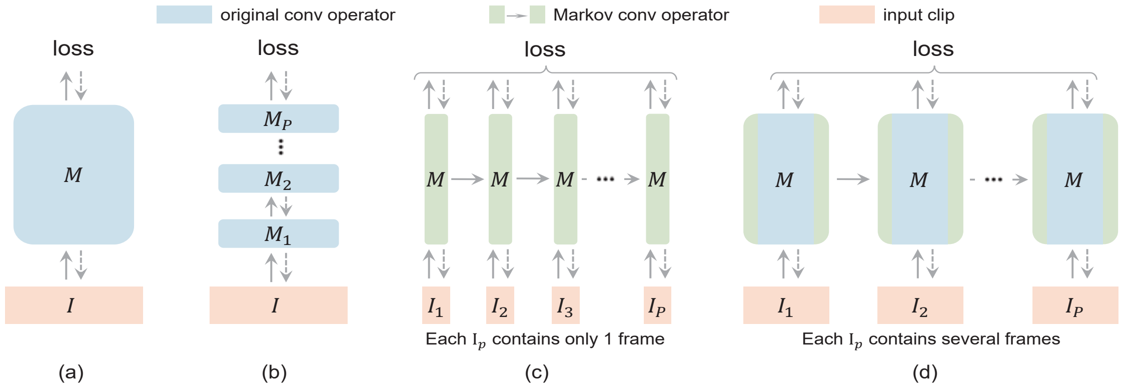

Figure 2: **Conceptual comparison of (a) baseline, (b) splitting model, and (c, d) splitting input progressive training**. The solid and dashed lines mean forward and backward paths. **(a)** The conventional training method trains the model M with input I in an integrated step. **(b)** In the splitting model setting, we split M into several parts. Although these parts satisfy the one-way dependency and can be calculated progressively in forward-propagation, they break this one-way dependency in back-propagation, failing to achieve the serial progressive training. **(c)** The splitting input setting satisfies the one-way dependency constraint in both forward and backward propagation, thus, can be optimized step by step progressively. Here, only the Markov operator is adopted to transfer features among progressive steps. **(d)** This setting is similar to (c). Differences are that (d) has a larger progressive length for each step and within each step, besides Markov operators adopted at the edge of each step to transfer temporal information, original convolutional operators are adopted to capture internal temporal features.

3. Progressive Training for Video Models

Before introducing the proposed progressive training method, let's consider a reference convolutional video processing model (*e.g.* C3D [53], I3D [2]) that operates on videos of shape $T \times H \times W$ (number of frames × height × width). Due to the computational resource limitation, common methods are to split the raw video into short ones with length $T' < T$. This separation forces models to focus on short local temporal features, wasting the large receptive field of deep models (*e.g.* SlowOnly with the receptive field of 39 only models 8 frames) and breaking the semantic integrity. Although there are methods for long videos [31, 52, 69, 64, 51], they design extra modules to model fixed features generated by backbone models which only take short clips as input, instead of end-to-end training the backbone and extra modules. Thus, the whole is likely sub-optimal, since the backbone still models short local features.

To end-to-end train long videos and ensure the complexity won't surpass the acceptable range, we propose the progressive training method. Inspired by Truncated Back-Propagation Through Time designed for recurrent methods, which reduces the complexity by separating each time stamp and truncating the computing graph of back-propagation, we realize that the end-to-end feature extraction and optimization process of a long video need not be finished in only one step. Instead, it can be a serial progressive process. We will show in experiments that this method can significantly enhance model's performance without introducing any parameter or complexity overhead. After analysis, we believe this improvement is mainly due to the increase of the temporal receptive field (see § 4.3).

3.1. Progressive Method

The progressive training (PGT) method aims at disassembling an integrated computing process into several portions that can be calculated serially to reduce computing resources requirements. To make sure that the temporal semantics are not broken by the disassembling process, the equivalent disassembling is the best choice — *i.e.* the calculation flow is exactly the same before and after disassembling. To achieve this equivalency, the serial disassembling process needs to satisfy such a constraint:

Constraint 1: *Among the split portions, there is only one-way dependency — if the computing process of portion A depends on results from portion B, then the computing of portion B cannot depend on portion A.*

The disassemble progress satisfying this constraint can be formally expressed as:

$$M(I) \Leftrightarrow M_P(I_P, M_{P-1}(I_{P-1}, M_{P-2}(I_{P-2}, ...))), \quad (1)$$

where $I = \{I_p | p \in \{1, 2, ..., P\}\}$ and $M = \{M_p | p \in \{1, ..., P\}\}$ are whole input and model. I_p and M_p are small portions of them, pth in the progressive order. P is the total number of split portions. This means that the disassembling can be achieved by splitting the input, splitting the model, or both of them. For example (see Fig. 2):

- *Splitting input*: we split an input video $I = \{F_1, F_2, ..., F_T\}$ with T frames to several frame groups $I_p = \{F_{(p-1)\times T'+1}, ..., F_{p\times T'}\}$, where T' is the length of each frame group. The computation of each group can only depend on the current and previous groups (here, model M is not split, — *i.e.* $M = M_1 = M_2 = ... = M_P$);

- *Splitting model*: we split a deep model to several layer groups and compute group by group to the results (here, we do not split the input. Thus, $I_2 = I_3 = ... = I_P = \text{None}$).

Although with Constraint 1 we can split the forward propagation equivalently, progressively training a long video also needs to serially disassemble the back-propagation process by truncating the gradient among the portions to completely isolate them to reduce the training resource requirements. To make sure each portion can get gradients to update their parameters after truncating, the disassembling needs to satisfy another constraint:

Constraint 2: *The one-way dependency should be maintained in the back-propagation process — Assume the computing of portion A depends on portion B and B does not depend on A. B should be able to update its parameters without gradients back-propagated from portion A.*

Now let's go back to the "*Splitting model*" example, which doesn't satisfy this constraint (see Fig. 2 (b)). Because fore layers can only get gradients from hind layers, if truncated, the fore layers will have no gradient to update their parameters. Thus, in the training phase, the split portions still need to be optimized together, failing to reduce the complexity of each portion by disassembling. Thus, in this paper, to satisfy both the two constraints, we only consider the "splitting input" setting:

$$M(I) \Leftrightarrow M(I_P, M(I_{P-1}, M(I_{P-2}, ...))), \qquad (2)$$

Eq. 2 is similar to the calculation process of RNN, but our progressive method does not aim at building a RNN model, instead, providing a method to train any model (such as CNNs, transformers) satisfying the constraints in a progressive manner to reduce the training resource requirements.

For an integrated input $I \in R^{T\times H\times W}$ with T frames, we call the length T' of $I_p \in R^{T'\times H\times W}$ as the "progressive length" and P as the "number of progressive steps".

3.2. Basic Progressive Training for Deep CNN

In this part, we will discuss how to apply the basic progressive training (PGT) method on convolutional networks.

Local operator The conventional convolution is a local operator that models both the past and future information. Formally, one-dimensional temporal convolution operator can be expressed as:

$$\text{Conv}(\text{f}_{\text{past}}, \text{f}_{\text{cur}}, \text{f}_{\text{future}}) \qquad (3)$$

where $\text{f}_{\text{past}}, \text{f}_{\text{cur}}, \text{f}_{\text{future}}$ are features of past, current and future frames from the previous layer. In deep models, features of each frame relies on the intermediate results from far past and future frames (30 frames in I3D; 18 ones in SlowOnly) — *i.e.* the adjacent frames rely on each other. This violates the one-way dependency constraints, making it difficult to adopt PGT to serialize the computation.

Basic Markov operator To solve the problem, we slightly modify the original local temporal convolutional operator to Markov one — *i.e.* hind frames depend on fore frames, while fore ones do not depend on hind ones in both forward and backward processes (Constraint 1 & 2). The basic Markov convolutional operator (MCO) is to replace f_{future} with zero-padding in forward-propagation and truncate gradients to prevent it back-propagating through time:

$$\begin{aligned} \text{f}_{\text{res}} &= \text{Conv}(\text{f}_{\text{past}}, \text{f}_{\text{cur}}, 0) \\ \frac{\partial \text{f}_{\text{res}}}{\partial \text{f}_{\text{past}}} &:= 0 \end{aligned} \qquad (4)$$

This simple modification does not aim at improving the original model. It is a compromise to make the deep CNN model satisfy the one-way dependency in Constraint 1 & 2. We can apply the most fine-grained progressive training method (see Fig. 2 (c)) with it: every progressive step only propagates information forward and backward one frame — *i.e.* the progressive length $T' = 1$ and the number of progressive steps $P = T$.

Basic progressive schedule The "Fine-grained" progressive method above is not necessary, because it makes the computing complexity of each step too low to take advantage of all the hardware's parallel computing power. More importantly, compared with the original local convolutional operator, the basic Markov one has a worse ability to extract temporal features since it weakens the temporal information flows. To this end, we set the progress length T' to the same as the ordinary length of short clips generated by conventional preprocessing (*e.g.* 8 or 32), and meanwhile use the original and Markov convolutional operators in combination. Specifically, we adopt modified operators on the first and last frames of each progressive step, where the beginning one truncates the gradients and the ending one zero-pads f_{future}, to satisfy the one-way dependency and make Markov property exist between them. And for the frames in the middle, the original operator is utilized (see Fig. 2d and Fig. 3). Note that the original and the modified Markov operators share the same parameters.

With these basic Markov operator and progressive schedule, we apply the progressive training method on CNN models and we call it our *basic progressive training method* (Fig. 3). It's worth noting that in this basic method, the modification is tiny and can be applied on any convolutional network. In the following parts, we develop more delicate Markov operators (§3.3) and progressive schedules (§3.4) to get our full progressive training method.

3.3. Further Designs on Markov operators

When designing Markov convolutional operators, we make sure that they won't introduce more parameters or heavy computation overhead. A further criterion is that operators should strengthen the efficiency of temporal information propagation, unlike the basic Markov convolutional operator which only restricts the information transfer to satisfy the two constraints. We experiment with the following two advanced operators (see Fig. 3):

Cascade Markov convolutional operator (CMCO) In basic Markov operator, f_{past} is just features of the adjacent

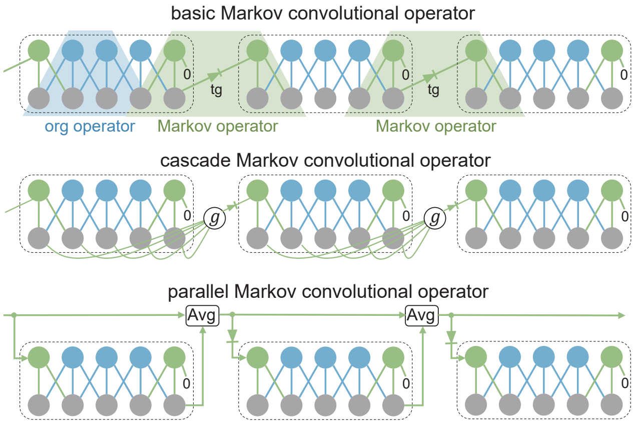
basic Markov convolutional operator
0
tg
0
tg
0
org operator
Markov operator
Markov operator
cascade Markov convolutional operator
0
g
0
g
0
parallel Markov convolutional operator
Avg
Avg
0
0
0

Figure 3: **Illustration of the proposed Markov convolutional operators**. Compared with the original convolutional operator, the Markov convolutional operator (MCO) satisfies the one-way dependency constraint. Cascade MCO enhances the temporal information flow between two progressive steps. Parallel MCO intensifies long-term features transfer. "tg" denotes "truncate gradient".

previous frame. To better propagate the integrated feature of the last progressive step, CMCO sets $\mathrm{f}_{\mathrm{past}}$ as the aggregation of features from all the frames in the previous progressive step. The aggregation function $g(\cdot)$ can be average/max pooling or other reasonable choices. CMCO builds denser information paths to enhance feature propagation.

Parallel Markov convolutional operator (PMCO) In CMCO, we enhance the connection between the current and adjacent previous progress steps. In PMCO, we intensify the information flows from all the previous steps. Specifically, we set $\mathrm{f}_{\mathrm{past}}$ as the momentum average of the features from all the previous progressive steps. PMCO allows features to transfer better over a long time horizon.

These operators do not change the core computation of the basic Markov convolutional operator, just modify the input features by adding some preprocessing. Note that the preprocessing only contains a few addition operations and the overhead can be ignored.

3.4. Further Designs on Progressive Schedule

Here, we further refine the progressive schedule. In the basic version, the progressive length T' and number of progressive step P are fixed values. Thus, the positions of the local and Markov operators in a long sequence are fixed too. Because the behaviour of the local and Markov operators is not the same, as the Markov one has a relatively weaker feature extraction ability than the local operator, the temporal feature capture will be uneven, leading to information propagation bottlenecks.

To avoid above problems caused by such unevenness, we propose the dynamic progressive regularization (DPR). Specifically, we randomly jitter the value of T' and P around the values of basic schedule and keep $T' \times P$ floating around a constant value. This method adjusts the ratio of local operators to Markov operators and meanwhile makes the position of Markov operators more evenly distributed in the long video to reduce the effect caused by information propagation bottlenecks. From another point of view, similar to Dropout [45], DPR which randomly adjust the paths of forward and backward propagation is kind of a model regularization. Thus, we call it dynamic progressive regularization. In experiments, we adopt two DPR settings:

- *DPR-A*: We randomly choose T' from the set $\{0.75T'_b, T'_b, 1.25T'_b\}$ where T'_b is the progressive length of the basic schedule. Note that when the progressive length is $1.25T'_b$, the training complexity of each step will be higher.
- *DPR-B*: We randomly choose T' from the set $\{0.5T'_b, 0.75T'_b, T'_b\}$. This setting adopts more Markov operators and won't cause complexity increase.

3.5. Implementation Details

Optimizer We choose SGD as our optimizer and its specific settings like momentum and learning rate schedule are kept the same with corresponding baselines. For different tasks and datasets, optimization details are given in their experiment sections. Different from the conventional training method, when training with PGT, we accumulate the gradients of each progressive step and update parameters after the backward propagations of all the progressive steps.

Progressive implementation We keep the same frame sampling stride as baselines. For example, a common sampling method is $T \times \tau = 8 \times 8$, where T and τ are the number of sampling frames and the sampling stride. We keep the stride τ and only enlarge T to train a long video instead of short clips. Considering the video length in commonly used datasets, we set the number of progressive step $P = 5$ for the basic schedule, unless otherwise specified. We let the two adjacent progressive steps overlap by one frame for better performance. Thus, the total frame in progressive input is $T = (T'-1)*P+1$. For DPR, given the total number of input frames T_b of basic schedule, the number of progressive step P is calculated as $P = \mathrm{round}[(T_b-1)/(T'-1)]$ to keep the total length basically the same.

Inference method Like training, the mainstream inference method also splits long videos into short clips, tests them separately, and averages the outputs to get final results. This method needs to test multiple views and usually, there are overlaps among the views. The proposed progressive training method trains a long video end-to-end, thus, when inference, we still test a long video directly. Although its inference complexity of one view is higher, we can adopt much fewer views to cover the whole video. We experiment with two inference methods:

- *Original long view* (orig long): In this mode, we get rid of the Markov operators and only utilize the original convolution for all the frames in the long video. It is the same as testing a long video in only one progressive step.

Table 1: **PGT's performance on Mini-Kinetics-200**. We express the inference complexity as single-view GFLOPs × number of progressive steps $P\times$ number of views v. "full PGT" adopts PMCO and DPR-B.

model	top-1	top-5	GFLOPs $\times P\times$ v
RegNet0.4G-3D	74.5	92.3	$6.59 \times 1 \times 30$
RegNet0.4G-3D, + basic PGT	76.6	93.0	$6.59 \times 5 \times 6$
RegNet0.4G-3D, + full PGT	77.5	93.6	$6.59 \times 5 \times 6$
SlowOnly, R50	77.1	93.4	$54.5 \times 1 \times 30$
SlowOnly, R50, + basic PGT	78.9	94.0	$54.5 \times 5 \times 6$
SlowOnly, R50, + full PGT	79.6	94.2	$54.5 \times 5 \times 6$

- *Progressive long view* (PG long): Just like the training phase, a long video is tested in multiple progressive steps.

4. Experiments on Kinetics

We first evaluate our progressive training method on the large scale action recognition benchmark Kinetics-400 [20] and provide ablations on its subset Mini-Kinetics-200 [66].

Dataset Kinetics-400 contains 240k training and 20k validation videos covering 400 action categories. Its subset Mini-Kinetics-200 includes 200 categories and each one contains 400 training samples and 25 validation samples, resulting in 80k training and 5k validation samples in total.

Training We adopt ResNet-3D (SlowOnly) [6], RegNet-3D [40], and SlowFast [6] as our baselines. The baseline training recipe follows [6]. We run SGD for 196 epochs on 16 GPUs with a mini-batch of 8 clips per GPU with initial learning rate of 0.2. The half-period cosine learning rate schedule [27] is adopted. We use a weight decay of 10^{-4}, momentum of 0.9, and a linear learning rate warm-up [10] from 0.02 over 34 epochs. Video clips are resized with shorter side $\in [256, 320]$ and inputs are randomly cropped patches with size of 224×224. Temporal sampling method is $T \times \tau = 8 \times 8$. For progressive training, we enlarge the weight decay to 2×10^{-4}, set $T' = 8$ and $P = 5$. The total training epoch reduces to 100 and other recipes keep the same with baselines.

Inference Following [6], 10 clips with size $T \times \tau = 8 \times 8$ are uniformly sampled from a video along temporal dimension. Each frame is resized with shorter side of 256 pixels and three patches of size 256×256 are taken to cover the spatial dimensions. Thus, there are 10×3 views in total for baselines. For PGT, each clip has a size of $T' \times P \times \tau = 8 \times 5 \times 8$, covering a much longer range. Thus, we only take 2 temporal views (2×3 views in total).

4.1. Ablation Study

This section provides ablation studies on Mini-Kinetics-200 comparing accuracy and computational complexity. For convenient comparison, we express complexity as one-view Flops × progressive steps $P\times$ number of views v.

Basic and full progressive training In Tab. 1, we report the performance comparison between the progressive training (PGT) method and baselines to preliminarily reveal PGT's effectiveness. Since PGT does not modify models' overall structure, the one-view complexities of PGT are the same as the baselines. PGT processes 5 times longer videos than the baseline. Thus, it needs fewer temporal views to cover the whole video. Their total complexities are (almost) the same — *i.e.* PGT does not involve any overhead. From Tab. 1, it is seen that PGT improves the top-1 accuracy of RegNet0.4G [40] by 3.0% and SlowOnly-50 [6] by 2.5%, revealing the importance of intact global temporal features that PGT focuses on.

Next, Tab. 2 shows a series of ablations on the progressive training designs and inference methods, mainly using the RegNet0.4G-3D network, analyzed in turn.

Markov operators Tab. 2a shows the performance of various Markov convolutional operators. As a naive Markov operator, the basic MCO improves the performance significantly as the previous paragraph states, although it sacrifices some feature extraction capabilities to transfer semantics among progressive steps. CMCO and PMCO alleviate this weakness to some extent and further improve the performance by 0.5% and 0.7%. For the following experiments (except ablations), we employ PMCO as our default.

PGT schedules Different feature extraction capacities of the Markov and local convolutional operators make temporal features uneven. We solve this problem by introducing the dynamic progressive regularization (DPR). Tab. 2b shows the effectiveness of it. It is seen that more progressive steps lead to better performance. DPR-A and DPR-B have almost the same improvements, where DPR-A is only a little bit ahead. Since DPR-B has lower complexity, we employ it as our default for following experiments.

Inference methods For PGT, we test the performance with longer clips and fewer views. In Tab. 2c, we also test the baseline model with this setting (first two lines) and we can see that it achieves similar performance to the multi-view short-clip setting, without substantial improvements. This reveals that it is the progressive training process that makes the model extract better temporal features instead of the longer test setting. The same conclusion can be drawn by comparing the 1st & 3rd or 2nd & 4th rows.

Then we compare the two inference methods designed for PGT mentioned in §3.5. It is seen that the *original long view* inference method performs better on Kinetics. In the next section, we will show that the *progressive long view* method is more suitable for Charades which contains longer activities consisting of several sub-actions.

4.2. Main Results

Tab. 3 shows the comparison with the SOTA results on Kinetics-400 for PGT method with different backbones: RegNet [40], ResNet [13], SlowFast [6], and Nonlocal [60].

In comparison to the advanced video baselines, our PGT method consistently provides a performance boost with negligible complexity overhead. For single-stream models such as RegNet and ResNet, PGT improves the perfor-

Table 2: **Ablations on Mini-Kinetics-200**. Experiments are mainly based on RegNet0.4G-3D. For baseline, $T \times \tau = 8 \times 8$. $P = 5$ when adopting PGT.

(a) **Markov Operators** Operators enhanced temporal information flow achieve better performances than the basic one.

model	operator	top-1	top-5
RegNet0.4G-3D	baseline	74.5	92.3
RegNet0.4G-3D	basic MCO	76.6	93.0
RegNet0.4G-3D	CMCO-avg	77.1	93.3
RegNet0.4G-3D	CMCO-max	76.9	93.2
RegNet0.4G-3D	PMCO	77.3	93.3
SlowOnly, R50	baseline	77.1	93.4
SlowOnly, R50	basic MCO	78.9	94.0
SlowOnly, R50	PMCO	79.3	94.1

(b) **PGT Schedules** More progressive steps perform better and DPR is more effective for more steps.

P	schedule	top-1	top-5
1 step	baseline	74.5	92.3
5 step	basic	76.6	93.0
5 step	DPR-A	77.3	93.4
5 step	DPR-B	77.2	93.4
4 step	basic	76.3	92.8
4 step	DPR-B	76.7	93.0
3 step	basic	75.8	92.7
2 step	basic	75.1	92.6

(c) **Inference Methods** Performances of the baselines and progressive training methods with different inference schemes.

train method	test method	top-1	top-5
baseline	10×3 short clip	74.5	92.3
baseline	1×3 orig long	74.7	92.4
basic PGT	10×3 short clip	75.8	92.7
basic PGT	1×3 orig long	76.1	92.7
basic PGT	1×3 PG long	75.6	92.7
basic PGT	2×3 orig long	76.6	93.0
full PGT	2 × 3 orig long	77.5	93.6
full PGT	2 × 3 PG long	76.7	93.1

Table 3: **Comparison with the SOTAs on Kinetics-400.** "flow" column indicates whether to adopt the optical flow and "preT" denotes pre-trained on ImageNet.

model	flow	preT	top-1	top-5	inference GFLOPs×P × v
I3D [2]		✓	72.1	90.3	108×1×N/A
Two-Stream I3D	✓	✓	75.7	92.0	216×1×N/A
S3D-G [66]	✓	✓	77.2	93.0	143×1×N/A
Nonlocal R50 [60]		✓	76.5	92.6	282×1×30
Nonlocal R101		✓	77.7	93.3	359×1×30
STC [3]			68.7	88.5	N/A×1×N/A
ARTNet [58]			69.2	88.3	23.5×1×250
S3D [66]			69.4	89.1	66.4×1×N/A
ECO [70]			70.0	89.4	N/A×1×N/A
I3D [2]	✓		71.6	90.0	216×1×N/A
R(2+1)D [54]	✓		73.9	90.9	304×1×115
X3D-M [5]			76.0	92.3	6.2×1×30
X3D-XL [5]			79.1	93.9	48.4×1×30
RegNet0.4G-3D [40]			70.3	89.3	6.59×1×30
RegNet0.4G-3D + PGT			72.1	90.7	6.59×5×6
R50-3D [6]			74.0	91.3	54.5×1×30
R50-3D + PGT			75.9	92.4	54.5×5×6
R101-3D [6]			75.5	91.9	110×1×30
R101-3D + PGT			77.1	92.9	110×5×6
SlowFast R50 [6]			75.8	92.0	65.7×1×30
SlowFast R50 + NL			76.5	92.4	80.8×1×30
SlowFast R50 + PGT			76.8	92.6	65.7× 5×6
SlowFast R101 [6]			77.2	92.8	126×1×30

mance by ∼1.8 accuracy. As for SlowFast with two streams, PGT provides 1.0% improvements, which is higher than adopting Nonlocal with 10%∼20% complexity overheads.

Comparisons of the performance and complexity trade-off are shown in Fig. 4. The horizontal axis measures the single-step single-view GFLOPs with 256×256 pixels input. We can see that PGT achieves better performance with lower complexity as the red arrow shows. The red boxes show that PGT leads to almost the same improvements as SlowFast does with lower complexities.

4.3. Analysis on Receptive Field

Modern deep convolutional video models often have large theoretical temporal receptive fields, such as 39 for SlowOnly network. Theoretical receptive field (TRF) is the upper bound of effective receptive field (ERF) [30] and from [30], we know the ERF will gradually increase during the training process, however, the input short clip generated by the mainstream temporal cropping training method is much shorter than the TRF, making model's ERF difficult to achieve a relative larger value after training.

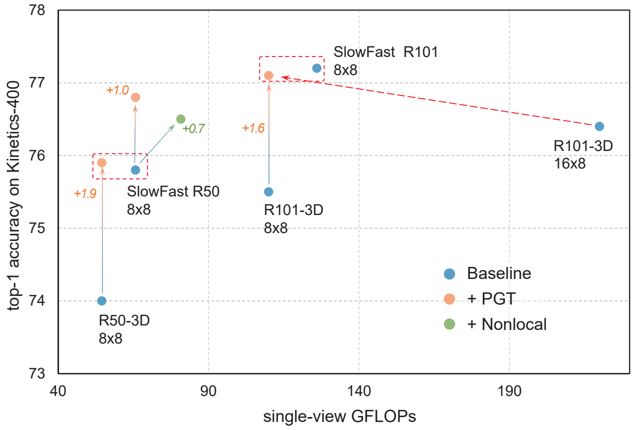

Figure 4: **Accuracy & complexity comparison** on Kinetics-400. PGT method achieves consistent improvements on all the baselines with negligible overhead. As the red dashed line and red boxes shows, the PGT method achieves a better performance/complexity trade-off.

Here we conduct experiments to reveal that the proposed progressive training method can alleviate this problem. We adopt the SlowOnly network and train it on the Kinetics-200 dataset [66]. Qualitative results are shown in Fig. 5. We can see that compared with the original method trained on 8 frames, our progressive training method with $T' = 8$ and $P = 5$ has a 40% larger ERF.

5. Experiments on Charades

We then evaluate the proposed method on Charades dataset [42] containing longer range activities spanning 30s on average. It contains about 9.8k training and 1.8k validation videos in 157 action categories (multi-labelled).

Implementation details We adopt SlowFast [6] and ResNet-3D (SlowOnly) [6] as our baselines. The models are pre-trained on Kinetics-400 or Kinetics-600 [1] following prior works [6, 5]. For PGT, we set $T' = 16$ and $P = 5$ for both training and inference. PMCO and CMCO-max are adopted together here. We adopt "pg long" inference method and get the final results with max-pooling over frames, instead of average-pooling since Charades is a multi-classification dataset.

Results Charades contains longer range activities. Thus,

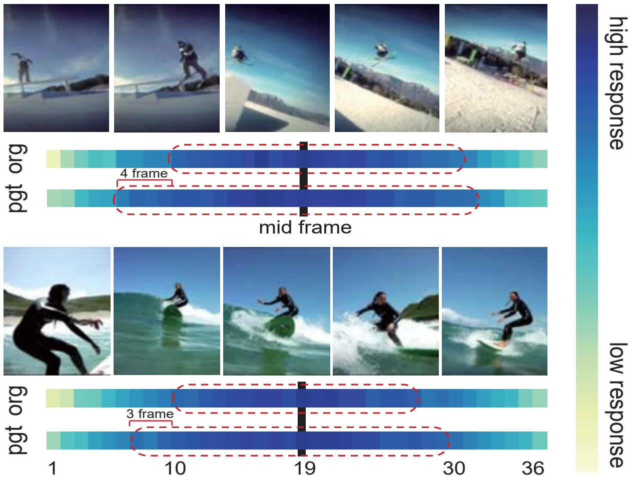

Figure 5: **Effective receptive field** (ERF) of original (org) and progressive (pgt) training method. The color bars show the ERF of the 19th frame with 36 input frames in total. From the red dashed boxes we can see that the progressive training method has a 40% larger ERF on previous frames.

Table 4: **Comparison with the SOTA on Charades.** All PGT settings are based on $T \times P \times \tau = 16 \times 5 \times 8$ and their corresponding baselines are $T \times \tau = 16 \times 8$.

model	backbone	pretrain	mAP	GFLOPs$\times P \times v$
Nonlocal [60]	R101	IN+K400	37.5	544 ×1×30
STRG, +NL [61]	R101	IN+K400	39.7	630×1×30
Timeception [15]	R101	K400	41.1	N/A
LFB, +NL [64]	R101	K400	42.5	529×1×30
X3D-XL [5]	-	K400	43.4	48.4×1×30
SlowOnly [6]	R50	K400	37.3	109×1×30
SlowOnly, + PGT	R50	K400	40.3	109×5×6
SlowOnly	R101	K400	39.0	187 ×1×30
SlowOnly, + PGT	R101	K400	42.7	187×5× 6
SlowFast [6]	R50	K400	39.6	130×1×30
SlowFast, +PGT	R50	K400	43.8	130×5×6
SlowFast [6]	R101	K400	42.1	213×1×30
SlowFast, +NL	R101	K400	42.5	234×1×30
SlowFast, +PGT	R101	K400	44.3	213×5 ×6
SlowFast, +NL	R101	K600	45.2	234×1×30
SlowFast, +PGT	R101	K600	47.7	213×5× 6

Table 5: **Performances on AVA-v2.2.** Our PGT variants have progressive step $P = 5$.

model	pretrain	val mAP
SlowOnly, R50, 8×8 [6]	K400	21.9
SlowOnly, R50, 16×8 [6]	K400	22.9
SlowOnly, R50, 8×5×8, + PGT	K400	23.5
SlowOnly, R101, 8×8 [6]	K400	23.4
SlowOnly, R101, 8×5×8, + PGT	K400	24.5
SlowFast, R101, 8×8, + NL [5]	K600	27.4
SlowFast, R101, 8×5×8, + PGT	K600	27.6

it can better reflect the advantages of PGT designed for long videos. Tab. 4 shows the results with ResNet [13], SlowFast [6], and Nonlocal [60] as backbones. It is seen that our PGT consistently provides a performance boost: **3.1** mAP improvements on average for SlowOnly and SlowFast. It is worth noting that for SlowFast-R50, the improvement is up to **4.2** mAP. Moreover, compared with Nonlocal that introduces 10% overhead and LFB [64] with 182% overhead, our PGT provides much more performance improvements.

Note that in Tab. 4, we report the performances achieved by "progressive long view" inference method. The performances of "original long view" inference method are ∼3% lower in mAP, which reveals that "pg long" method is more suitable for long activities consisting of several sub-actions.

6. Experiments on AVA

The AVA [11] dataset is designed for action detection, which is labelled with bounding-boxes and action categories for each person in 437 movies. Following standard protocol, we report performance (mAP) on 60 classes.

Implementation We adopt SlowOnly [6] and SlowFast [6] as our baselines. Following [11, 46, 18, 6], we utilize the off-the-shelf human detection results originally adopted by SlowFast as our region proposals. The models are pre-trained on Kinetics-400 or Kinetics-600 following prior works [5, 6, 64]. For PGT, similar to Charades, we adopt "pg long" inference method and max-pooling over frames to get multi-classification results.

Results Comparisons with baselines of SlowOnly and SlowFast are shown in Tab. 5. It is seen that the proposed PGT provides consistent performance improvements. After adopting the progressive training method, SlowOnly-R50 model achieves a better performance than the much larger SlowOnly-R101 model. Consistent with the experiments on Kinetics and Charades, PGT method provides higher performance improvements than the Nonlocal module with lower computational complexity. Note that compared with the original training method adopting a twice longer input length which introduces twice inference complexity, our PGT version has a better performance (+0.6 mAP vs. 16 × 8 original training version). This verifies that PGT is an efficient training strategy for processing long videos.

7. Conclusion

We propose the progressive training (PGT) method for end-to-end training models on long videos. With the redesigned Markov convolutional operators, we split an integrated computation progress into several serial progressive steps satisfying one-way dependency, which reduces the resource requirement while ensuring the integrity of temporal semantics. With out-of-box settings, it works on multiple advanced video backbones and benchmarks, achieving 1∼4% higher performances with negligible computation or parameter overhead. We hope PGT can provide a new option for researchers who need to process long videos.

Acknowledges This work is supported in part by the National Key R&D Program of China, No. 2017YFA0700800, National Natural Science Foundation of China under Grants 61772332 and Shanghai Qi Zhi Institute, SHEITC (018-RGZN-02046).

References

[1] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about kinetics-600. *arXiv preprint arXiv:1808.01340*, 2018.

[2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6299–6308, 2017.

[3] Ali Diba, Mohsen Fayyaz, Vivek Sharma, M Mahdi Arzani, Rahman Yousefzadeh, Juergen Gall, and Luc Van Gool. Spatio-temporal channel correlation networks for action classification. In *Eur. Conf. Comput. Vis.*, pages 284–299, 2018.

[4] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2625–2634, 2015.

[5] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 203–213, 2020.

[6] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Int. Conf. Comput. Vis.*, pages 6202–6211, 2019.

[7] Christoph Feichtenhofer, Axel Pinz, and Richard P Wildes. Spatiotemporal multiplier networks for video action recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4768–4777, 2017.

[8] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1933–1941, 2016.

[9] Rohit Girdhar, Joao Carreira, Carl Doersch, and Andrew Zisserman. Video action transformer network. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 244–253, 2019.

[10] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[11] Chunhui Gu, Chen Sun, David A Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, et al. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6047–6056, 2018.

[12] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6546–6555, 2018.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 770–778, 2016.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.

[15] Noureldien Hussein, Efstratios Gavves, and Arnold WM Smeulders. Timeception for complex action recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 254–263, 2019.

[16] Noureldien Hussein, Efstratios Gavves, and Arnold W. M. Smeulders. Videograph: Recognizing minutes-long human activities in videos. *arXiv preprint arXiv:1905.05143*, 2019.

[17] Noureldien Hussein, Efstratios Gavves, and Arnold W. M. Smeulders. PIC: permutation invariant convolution for recognizing long-range activities. *CoRR*, abs/2003.08275, 2020.

[18] Jianwen Jiang, Yu Cao, Lin Song, Shiwei Zhang4 Yunkai Li, Ziyao Xu, Qian Wu, Chuang Gan, Chi Zhang, and Gang Yu. Human centric spatio-temporal action localization. In *ActivityNet Workshop on CVPR*, 2018.

[19] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1725–1732, 2014.

[20] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[22] Fu Li, Chuang Gan, Xiao Liu, Yunlong Bian, Xiang Long, Yandong Li, Zhichao Li, Jie Zhou, and Shilei Wen. Temporal modeling approaches for large-scale youtube-8m video understanding. *arXiv preprint arXiv:1707.04555*, 2017.

[23] Yong-Lu Li, Xinpeng Liu, Han Lu, Shiyi Wang, Junqi Liu, Jiefeng Li, and Cewu Lu. Detailed 2d-3d joint representation for human-object interaction. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[24] Yong-Lu Li, Liang Xu, Xinpeng Liu, Xijie Huang, Yue Xu, Shiyi Wang, Hao-Shu Fang, Ze Ma, Mingyang Chen, and Cewu Lu. Pastanet: Toward human activity knowledge engine. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[25] Zhenyang Li, Kirill Gavrilyuk, Efstratios Gavves, Mihir Jain, and Cees GM Snoek. Videolstm convolves, attends and flows for action recognition. *Computer Vision and Image Understanding*, 166:41–50, 2018.

[26] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Int. Conf. Comput. Vis.*, pages 7083–7093, 2019.

[27] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[28] Yujing Lou, Yang You, Chengkun Li, Zhoujun Cheng, Liangwei Li, Lizhuang Ma, Weiming Wang, and Cewu Lu. Human correspondence consensus for 3d object semantic understanding. In *Eur. Conf. Comput. Vis.*, pages 496–512. Springer, 2020.

[29] Chenxu Luo and Alan L. Yuille. Grouped spatial-temporal aggregation for efficient action recognition. In *Int. Conf. Comput. Vis.*, pages 5511–5520. IEEE, 2019.

[30] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolu-

tional neural networks. In *Adv. Neural Inform. Process. Syst.*, pages 4898–4906, 2016.

[31] Antoine Miech, Ivan Laptev, and Josef Sivic. Learnable pooling with context gating for video classification. *arXiv preprint arXiv:1706.06905*, 2017.

[32] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *Adv. Neural Inform. Process. Syst.*, pages 2863–2871, 2015.

[33] Bo Pang, Yizhuo Li, Jiefeng Li, Muchen Li, Hanwen Cao, and Cewu Lu. Tdaf: Top-down attention framework for vision tasks. In *AAAI*, 2021.

[34] Bo Pang, Yizhuo Li, Yifan Zhang, Muchen Li, and Cewu Lu. Tubetk: Adopting tubes to track multi-object in a one-step training model. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6308–6318, 2020.

[35] Bo Pang, Kaiwen Zha, Hanwen Cao, Chen Shi, and Cewu Lu. Deep rnn framework for visual sequential applications. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 423–432, 2019.

[36] Bo Pang, Kaiwen Zha, Hanwen Cao, Jiajun Tang, Minghui Yu, and Cewu Lu. Complex sequential understanding through the awareness of spatial and temporal concepts. *Nat. Mach. Intell.*, 2(5):245–253, 2020.

[37] Bo Pang, Kaiwen Zha, Yifan Zhang, and Cewu Lu. Further understanding videos through adverbs: A new video task. In *AAAI*, pages 11823–11830, 2020.

[38] AJ Piergiovanni and Michael S Ryoo. Representation flow for action recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 9945–9953, 2019.

[39] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *Int. Conf. Comput. Vis.*, pages 5533–5541, 2017.

[40] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 10428–10436, 2020.

[41] Michael S. Ryoo, A. J. Piergiovanni, Mingxing Tan, and Anelia Angelova. Assemblenet: Searching for multi-stream neural connectivity in video architectures. In *Int. Conf. Learn. Represent.*, 2020.

[42] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *Eur. Conf. Comput. Vis.*, pages 510–526. Springer, 2016.

[43] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Adv. Neural Inform. Process. Syst.*, pages 568–576, 2014.

[44] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[45] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014.

[46] Chen Sun, Abhinav Shrivastava, Carl Vondrick, Kevin Murphy, Rahul Sukthankar, and Cordelia Schmid. Actor-centric relation network. In *Eur. Conf. Comput. Vis.*, pages 318–334, 2018.

[47] Jianhua Sun, Qinhong Jiang, and Cewu Lu. Recursive social behavior graph for trajectory prediction. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 660–669, 2020.

[48] Lin Sun, Kui Jia, Kevin Chen, Dit-Yan Yeung, Bertram E Shi, and Silvio Savarese. Lattice long short-term memory for human action recognition. In *Int. Conf. Comput. Vis.*, pages 2147–2156, 2017.

[49] Lin Sun, Kui Jia, Dit-Yan Yeung, and Bertram E Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *Int. Conf. Comput. Vis.*, pages 4597–4605, 2015.

[50] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1–9, 2015.

[51] Jiajun Tang, Jin Xia, Xinzhi Mu, Bo Pang, and Cewu Lu. Asynchronous interaction aggregation for action detection. *Eur. Conf. Comput. Vis.*, 2020.

[52] Yongyi Tang, Xing Zhang, Lin Ma, Jingwen Wang, Shaoxiang Chen, and Yu-Gang Jiang. Non-local netvlad encoding for video classification. In *Eur. Conf. Comput. Vis.*, pages 0–0, 2018.

[53] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Int. Conf. Comput. Vis.*, pages 4489–4497, 2015.

[54] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6450–6459, 2018.

[55] Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. Translating videos to natural language using deep recurrent neural networks. *arXiv preprint arXiv:1412.4729*, 2014.

[56] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. *Int. Conf. Learn. Represent.*, 2017.

[57] Lei Wang, Piotr Koniusz, and Du Huynh. Hallucinating IDT descriptors and I3D optical flow features for action recognition with cnns. In *Int. Conf. Comput. Vis.*, pages 8697–8707. IEEE, 2019.

[58] Limin Wang, Wei Li, Wen Li, and Luc Van Gool. Appearance-and-relation networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1430–1439, 2018.

[59] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks for action recognition in videos. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(11):2740–2755, 2018.

[60] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7794–7803, 2018.

[61] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *Eur. Conf. Comput. Vis.*, pages 399–417, 2018.

[62] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proc. IEEE*, 78(10):1550–1560, 1990.

[63] Ronald J Williams and Jing Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Comput.*, 2(4):490–501, 1990.

[64] Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krahenbuhl, and Ross Girshick. Long-term feature banks for detailed video understanding. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 284–293, 2019.

[65] Zuxuan Wu, Xi Wang, Yu-Gang Jiang, Hao Ye, and Xiangyang Xue. Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. In *ACM Int. Conf. Multimedia*, pages 461–470, 2015.

[66] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning for video understanding. *Eur. Conf. Comput. Vis.*, 2018.

[67] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Adv. Neural Inform. Process. Syst.*, pages 802–810, 2015.

[68] Yuliang Xiu, Jiefeng Li, Haoyu Wang, Yinghong Fang, and Cewu Lu. Pose Flow: Efficient online pose tracking. In *Brit. Mach. Vis. Conf.*, 2018.

[69] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4694–4702, 2015.

[70] Mohammadreza Zolfaghari, Kamaljeet Singh, and Thomas Brox. Eco: Efficient convolutional network for online video understanding. In *Eur. Conf. Comput. Vis.*, pages 695–712, 2018.