Temporal Pyramid Network for Action Recognition

Ceyuan Yang[†,1], Yinghao Xu[†,1], Jianping Shi[2], Bo Dai[1], Bolei Zhou[1]
[1]The Chinese University of Hong Kong, [2]SenseTime Group Limited
{yc019, xy119, bdai, bzhou}@ie.cuhk.edu.hk, shijianping@sensetime.com

Abstract

Visual tempo characterizes the dynamics and the temporal scale of an action. Modeling such visual tempos of different actions facilitates their recognition. Previous works often capture the visual tempo through sampling raw videos at multiple rates and constructing an input-level frame pyramid, which usually requires a costly multi-branch network to handle. In this work we propose a generic Temporal Pyramid Network (TPN) at the feature-level, which can be flexibly integrated into 2D or 3D backbone networks in a plug-and-play manner. Two essential components of TPN, the source of features and the fusion of features, form a feature hierarchy for the backbone so that it can capture action instances at various tempos. TPN also shows consistent improvements over other challenging baselines on several action recognition datasets. Specifically, when equipped with TPN, the 3D ResNet-50 with dense sampling obtains a 2% gain on the validation set of Kinetics-400. A further analysis also reveals that TPN gains most of its improvements on action classes that have large variances in their visual tempos, validating the effectiveness of TPN.[1]

1. Introduction

While great progress has been made by deep neural networks to improve the accuracy of video action recognition [5, 32, 33, 36, 30], an important aspect of characterizing dfferent actions is often missed in the design of these recognition networks - the visual tempos of action instances. Visual tempo actually describes how fast an action goes, which tends to determine the effective duration at the temporal scale for recognition. As shown at the bottom of Figure 1, action classes naturally have different visual tempos (*e.g. hand clapping* and *walking*). In some cases the key to distinguish different action classes is their visual tempos, as they might share high similarities in visual appearance, such as *walking*, *jogging* and *running*. Moreover, as shown at the top of Figure 1, when performing the same action, each performer

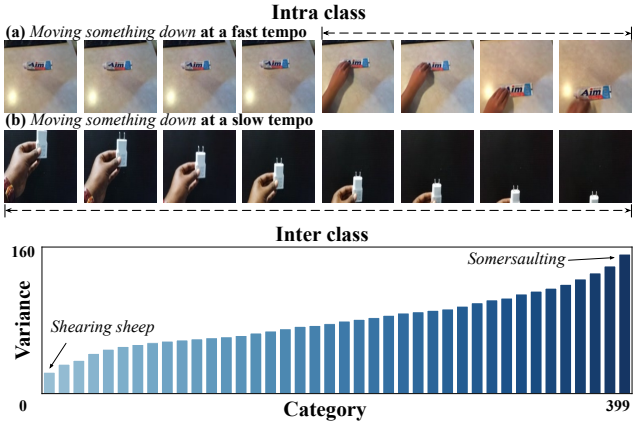

Figure 1. **Visual tempo variation of intra- and inter-class.** The action examples above show that people tend to act at different tempos even for the same action. The plot below shows different action categories sorted by their variances of visual tempos. Specifically *Somersaulting* has the largest variance in the visual tempo of its instances while *Shearing sheep* has the smallest variance. Details of variation measurements can be found in the experiment section.

may act at his/her own visual tempo, due to various factors such as age, mood, and energy level. *e.g.* ,an elder man tends to move slower than a younger man, so as a man with a heavier weight. Precise modeling of such intra- and inter-class variances in visual tempos of action instances can potentially bring a significant improvement to action recognition.

Previous attempts [5, 35, 33] for extracting the dynamic visual tempos of action instances mainly rely on constructing a frame pyramid, where each pyramid level samples the input frames at a different temporal rate. For example, we can sample from the total 64 frames of an video instance at intervals 16 and 2 respectively, to construct a two-level frame pyramid consisting of 4 and 32 frames. Subsequently, frames at each level are fed into different backbone subnetworks, and their output features are further combined together to make the final prediction. By sampling frames at different rates as input, backbone networks in [5, 35] are able to extract features of different receptive fields and represent the input action instance at different visual tempos. These backbone

[1]Code and models are available at this link.
[†] indicates equal contribution.

subnetworks thus jointly aggregate temporal information of both fast-tempo and slow-tempo, handling action instances at different temporal scales.

Previous methods [5, 35, 33] have obtained noticeable improvements for action recognition, however it remains computationally expensive to deal with the dynamic visual tempos of action instances at the input frame level. It is not scalable to pre-define the tempos in the input frame pyramid and then feed the frames into multiple network branches, especially when we use a large number of sampling rates. On the other hand, many commonly-used models in video recognition, such as C3D and I3D [26, 1], often stack a series of temporal convolutions. In these networks, as the depth of a layer increases, its temporal receptive field increases as well. As a result, the features at different depths in a *single* model already capture information of both fast-tempo and slow-tempo. Therefore, we propose to build a temporal pyramid network (TPN) to aggregate the information of various visual tempos at *feature level*. By leveraging the feature hierarchy formed inside the network, the proposed TPN is able to work with input frames fed at a single rate. As an auxiliary module, TPN could be applied in a plug-and-play manner to various existing action recognition models to bring consistent improvements.

In this work we first provide a general formulation of the proposed TPN, where several components are introduced to better capture the information at multiple visual tempos.We then evaluate TPNs on three benchmarks: Kinetics-400 [1], Something-Something V1 & V2 [10] and Epic-Kitchen [2] with comprehensive ablation studies. Without any bells and whistles, TPNs bring consistent improvements when combined with both 2D and 3D networks. Besides, the ablation study shows that TPN obtains most of its improvements from the action classes that have significant variances in visual tempos. This result verifies our assumption that aggregating features in a single model is sufficient to capture the visual tempos of action instances for video recognition.

2. Related Work

Video Action Recognition. Attempts for video action recognition could be divided into two categories. Methods in the first category often adopt a 2D + 1D paradigm, where 2D CNNs are applied over per-frame inputs, followed by a 1D module that aggregates per-frame features. Specifcally, two-stream networks in [24, 7, 6, 16] utilize two separate CNNs on per-frame visual appearances and optical flows respectively, and an average pooling operation for temporal aggregation. Among its variants, TSN [31] proposes to represent video clips by sampling from evenly divided segments. And TRN [38] and TSM [18] respectively replace the average pooling operation with an interpretable relational module and utilize a shift module, in order to better capture information along the temporal dimension. However, due to the deployment of 2D CNNs in these methods, semantics of the input frames could not interact with each other in the early stage, which limits their ability to capture the dynamics of visual tempos. Methods [26, 15] in the second category alternatively apply 3D CNNs that stack 3D convolutions to jointly model temporal and spatial semantics. Along this line of research, Non-local Network [32] introduces a special non-local operation to better exploit the long-range temporal dependencies between video frames. Besides Non-local Network, different modifications for the 3D CNNs, including the inflating 2D convolution kernels [1] and the decomposing 3D convolution kernels [21, 28, 34], can also boost the performances of 3D CNNs. Other effects [30, 36, 29, 22, 23] are taken on irregular convolution/pool for better feature alignment or study action instances in a fine-grained way. Although the aforementioned methods could better handle temporal information, the large variation of visual tempos remains neglected.

Visual Tempo Modeling in Action Recognition. The complex temporal structure of action instances, particularly in terms of the various visual tempos, raises a challenge for action recognition. In recent years, researchers have started exploring this direction. SlowFast [5] hard-codes the variance of visual tempos using an input-level frame pyramid that has level-wise frames sampled at different rates. Each level of the pyramid is also separately processed by a network, where mid-level features of these networks are interactively combined. With the assist of both the frame pyramid and the level-specific networks, SlowFast could robustly handle the variance of visual tempos. The complex temporal structure inside videos, particularly tempo variation, raises a challenge for action recognition. DTPN [35] also samples frames with different frame per seconds (FPS) to construct a natural pyramidal representation for arbitrary-length input videos. However, such a hard-coding scheme tends to require multiple frames , especially when the pyramid scales up. Inspired by feature-level pyramid networks [11, 19, 20, 17] that deal with the large variance of scales in object detection, we instead leverage the feature hierarchy of a backbone network, handling the variance of visual tempos in the feature-level. In this way we could hide the concern about visual tempos inside a single network, and we only need frames sampled at a single rate at the input-level.

3. Temporal Pyramid Network

The visual tempo of an action instance is one of the key factors for recognizing it, especially when other factors are ambiguous. For example, we cannot tell if an action instance belongs to *walking, jogging* or *running* based on its visual appearance. However, it is difficult to capture the visual tempos due to their inter- and intra-class variance across different videos. Previous works [5, 35, 33] address this

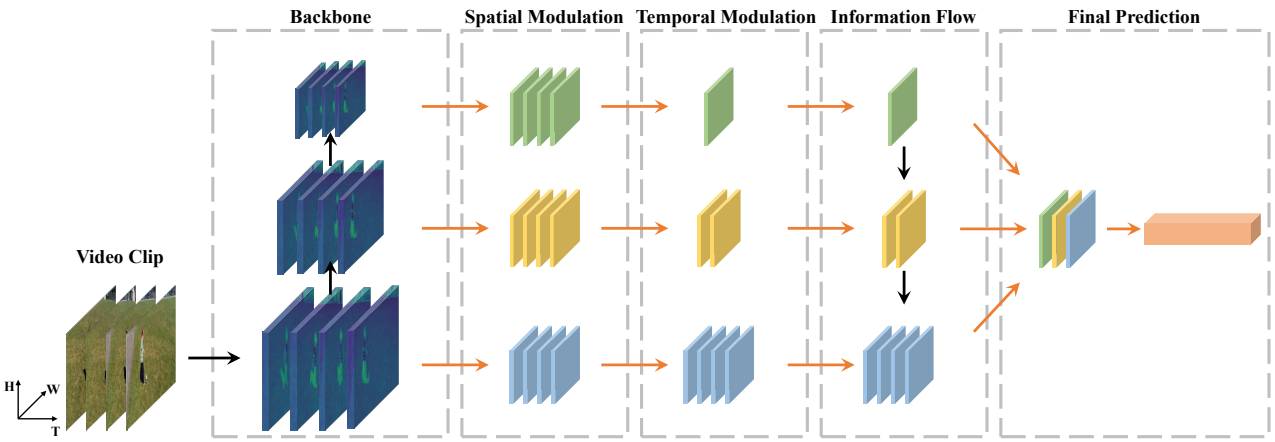

Figure 2. **Framework of TPN:** *Backbone Network* to extract multiple level features. *Spatial Semantic Modulation* spatially downsamples features to align semantics. *Temporal Rate Modulation* temporally downsamples features to adjust relative tempo among levels. *Information Flow* aggregates features in various directions to enhance and enrich level-wise representations. *Final Prediction* rescales and concatenates all levels of pyramid along channel dimension. Note that the channel dimensions in all modules and corresponding operations in (e) are omitted for brevity.

issue at the input-level. They utilize a frame pyramid that contains frames sampled at pre-defined rates to represent the input video instance at various visual tempos. Since each level of the frame pyramid requires a separate backbone network to handle, such an approach may be computationally expensive, especially when the level of pyramid scales up.

Inspired by the observation that features at multiple depths in a *single* network already cover various visual tempos, we propose a feature-level temporal pyramid network (TPN) for modeling the visual tempo. TPN could operate on only a single network no matter how many levels are included in it. Moreover, TPN could be applied to different architectures in a plug-and-play manner. To fully implement TPN, two essential components of TPN must be designed properly, namely 1) the feature source and 2) the feature aggregation. We propose the spatial semantic modulation and temporal tempo modulation to control the relative differences of the feature source in Sec.3.1, and construct multiple types of information flows for feature aggregation in Sec.3.2. Finally we show how to adopt TPN for action recognition in Sec.3.3, taking [5] as an exemplar backbone network.

3.1. Feature Source of TPN

Collection of Hierarchical Features. While TPN is built upon a set of M hierarchical features that have increasing temporal receptive fields from bottom to top, there are two alternative ways to collect these features from a backbone network. 1) *Single-depth pyramid*: a simple way is to choose a feature $\mathbf{F}_{\text{base}}$ of size $C \times T \times W \times H$ at some depth, and to sample along the temporal dimension with M different rates $\{r_1, ..., r_M; r_1 < r_2 < ... < r_M\}$. We refer to such a TPN as a single-depth pyramid consisting of $\{\mathbf{F}^{(1)}_{\text{base}}, ..., \mathbf{F}^{(M)}_{\text{base}}\}$ of sizes $\{C \times \frac{T}{r_1} \times W \times H, ..., C \times \frac{T}{r_M} \times W \times H\}$. Features collected in this way could lighten the workload of fusion as they have identical shapes besides the temporal dimension. However, they may limit in effectiveness as they represent video semantics only at a single spatial granularity. 2) *Multi-depth pyramid*: a better way is to collect a set of M features with increasing depths, resulting in a TPN made of $\{\mathbf{F}_1, \mathbf{F}_2, ..., \mathbf{F}_M\}$ of sizes $\{C_1 \times T_1 \times W_1 \times H_1, ..., C_M \times T_M \times W_M \times H_M\}$, where generally the dimensions satisfy $\{C_{i_1} \geq C_{i_2}, W_{i_1} \geq W_{i_2}, H_{i_1} \geq H_{i_2}; i_1 < i_2\}$. Such a multi-depth pyramid contains richer semantics in the spatial dimensions, yet raises the need of careful treatment in feature fusion, in order to ensure correct information flows between features.

Spatial Semantic Modulation. To align spatial semantics of features in the multi-depth pyramid, a spatial semantic modulation is utilized for TPN. The spatial semantic modulation works in two complementary ways. For each but the top-level feature, a stack of convolutions with level-specific stride are applied to it, matching its spatial shape and receptive field with the top one. Moreover, an auxiliary classification head is also appended to it to receive stronger supervision, leading to enhanced semantics. The overall objective for a backbone network with our proposed TPN thus becomes:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{CE,o} + \sum_{i=1}^{M-1} \lambda_i \mathcal{L}_{CE,i}, \tag{1}$$

where $\mathcal{L}_{CE,o}$ is the original Cross-Entropy loss, and $\mathcal{L}_{CE,i}$ is the loss for i-th auxiliary head. $\{\lambda_i\}$ are balancing coef-

ficients. After spatial semantic modulation, features have aligned shapes and consistent semantics in the spatial dimensions. However, it remains uncalibrated in the temporal dimension, where we introduce the proposed temporal rate modulation.

Temporal Rate Modulation. Recall in the input-level frame pyramid used in [5], the sampling rates of frames could be adjusted dynamically to increase its applicability. On the contrary, TPN is limited in the flexibility, as it operates on features of a backbone network, so that the visual tempos of these features are only controlled by their depths in the original network. To equip TPN with a similar flexibility as in the input-level frame pyramid, a set of hyper-parameters $\{\alpha_i\}_{i=1}^{M}$ are further introduced to TPN for temporal tempo modulation. Specifically, α_i denotes that after spatial semantic modulation, the updated feature at i-level will be temporally downsampled by a factor of α_i, using a parametric sub-net. The inclusion of such hyper-parameters enables us to better control the relative differences of features in terms of temporal scales, so that feature aggregation could be conducted more effectively. With some abuse of notations, we refer to $\mathbf{F}_i$ of size $C_i \times T_i \times W_i \times H_i$ as the i-th feature after both the spatial semantic modulation and the temporal rate modulation in the following content.

3.2. Information Flow of TPN

After collecting and pre-processing the hierarchical features as in Sec.3.1, so that they are dynamic in visual tempos and consistent in spatial semantics, we are ready to step in the second step of TPN construction – how to aggregate these features. Let $\mathbf{F}'_i$ be the aggregated feature at i-th level, generally there are three basic options:

$$\mathbf{F}'_i = \begin{cases} \mathbf{F}_i & \text{Isolation Flow} \\ \mathbf{F}'_i \bigoplus g(\mathbf{F}_i, T_i/T_{i-1}) & \text{Bottom-up Flow} \\ \mathbf{F}'_i \bigoplus g(\mathbf{F}_i, T_i/T_{i+1}) & \text{Top-down Flow} \end{cases}, \quad (2)$$

where $\bigoplus$ denotes element-wise addition. And to ensure the compatibility of the addition between consecutive features, during aggregation a down/up-sampling operation, $g(\mathbf{F}, \delta)$ with $\mathbf{F}$ as the feature and δ is the factor, is applied along the temporal dimension. Besides the above basic flows to aggregate features in TPN, we could also combine them to achieve two additional options, namely *Cascade Flow* and *Parallel Flow*. While applying a bottom-up flow after a top-down flow will lead to the cascade flow, applying them simultaneously will result in the parallel flow. See Fig. 3 for an illustration of all the possible flows. It's worth noting that more complicated flow (*e.g.* path aggregation in [20]) could be built on top of these flows. However, our attempts in this line of research have not shown further improvement. Finally, following Fig.2, all aggregated features in TPN will be rescaled and concatenated for succeeding predictions.

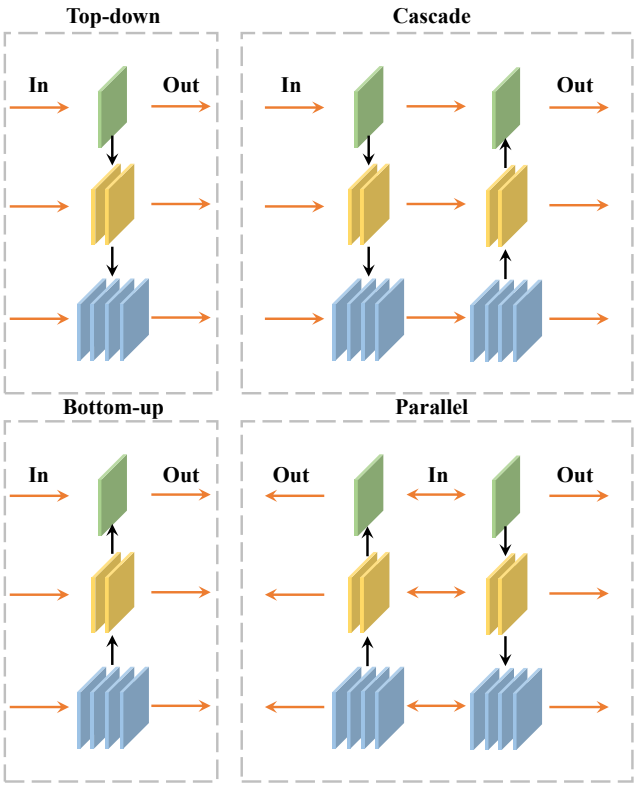

Figure 3. **Information Flow: Black arrows** illustrate the aggregation directions while the orange arrows denote the IO stream from *Temporal Modulation* to *Final Prediction* of Figure 2. The channel dimensions and up/downsample operations are omitted.

3.3. Implementation

Here we introduce the implementation of TPN for action recognition. Following [5], we use inflated ResNet [5] as the 3D backbone network, for its promising performance on various datasets [1]. Meanwhile, original ResNet [12] serves as our 2D backbone. We use the output features of *res2, res3, res4, res5* to build TPN, where they are spatially downsampled by respectively $4, 8, 16$ and 32 times, compared to the input frames. We provide the structure of 3D ResNet-50 in Tab.1 for the reference. In the spatial semantic modulation, a stack of convolutions with $M - i$ stride to process the feature at i-th level in a M-level TPN and the feature dimension would be decreased or increased to 1024. Besides, the temporal rate modulation for each feature is achieved by a convolutional layer and a max-pooling layer. Finally, after feature aggregation through one of the five flows mentioned in Sec. 3.2, features of TPN will be separately rescaled by max-pooling operations, and their concatenation will be fed into a fully-connected layer to make the final predictions. TPN can be also jointly trained with the backbone network in an end-to-end manner.

Stage	Layer	Output size
raw	-	8 ×224 ×224
conv_1	1×7×7, 64, stride 1, 2, 2	8×112×112
pool_1	1×3×3 max, stride 1, 2, 2	8×56×56
res_2	$\begin{bmatrix} 1\times1\times1, 64 \\ 1\times3\times3, 64 \\ 1\times1\times1, 256 \end{bmatrix} \times3$	8×56×56
res_3	$\begin{bmatrix} 1\times1\times1, 128 \\ 1\times3\times3, 128 \\ 1\times1\times1, 512 \end{bmatrix} \times4$	8×28×28
res_4	$\begin{bmatrix} 3\times1\times1, 256 \\ 1\times3\times3, 256 \\ 1\times1\times1, 1024 \end{bmatrix} \times6$	8×14×14
res_5	$\begin{bmatrix} 3\times1\times1, 512 \\ 1\times3\times3, 512 \\ 1\times1\times1, 2048 \end{bmatrix} \times3$	8×7×7
global average pool, fc		1×1×1

Table 1. **3D Backbone.** Following [5], our inflated 3D ResNet-50 backbone for video is shown. Note that both output size and kernel size are in T×W×H shape.

4. Experiments

We evaluate the proposed TPN on various action recognition datasets, including Kinetics-400 [1], Something-Something V1 & V2 [10], and Epic-Kitchen [2]. The consistent improvements show the effectiveness and generality of TPN. Ablation studies on the components of TPN are also included. Moreover, we present several empirical analysis to verify our motivation of TPN, *i.e*. a feature-level temporal pyramid on a single backbone is beneficial for capturing the variance of visual tempos. All experiments are conducted with the single modality (*i.e*. RGB frames) on MMAction [37] and evaluated on the validation set unless specified.

Dataset. Kinetics-400 [1] contains around 240k training videos and 19k validation videos that last for 10 seconds. It includes 400 action categories in total. Something-Something V1 [10] consists of 86k training videos and 11k validation videos belonging to 174 action categories, whose durations vary from 2 to 6 seconds. The second release (V2) of Something-Something increase the number of videos to 220k. Epic-Kitchen [2] includes around 125 verb and 352 noun categories. Following [8], we randomly select 232 videos (23439 segments) for training and 40 videos (4979 segments) for validation.

Training. Unless specified otherwise, our models are defaultly initialized by pre-trained models on ImageNet [3]. Following the setting in [5], the input frames are sampled from a set of consecutive 64 frames at a specific interval τ. Each frame is randomly cropped so that its short side ranges in $[256, 320]$ pixels, as in [32, 5, 25]. The augmentation of horizontal flip and a dropout [13] of 0.5 are adopted to reduce overfitting. And BatchNorm (BN) [14] is not frozen. We use a momentum of 0.9, a weight decay of 0.0001 and a synchronized SGD training over 8 GPUs [9]. Each GPU has a batch-size of 8, resulting in a mini-batch of 64 in total. For Kinetics-400, the learning rate is 0.01 and will be reduced by a factor of 10 at 100, 125 epochs (150 epochs in total) respectively. For Something-Something V1 & V2 [10] and Epic-Kitchen [2], our model is trained for 150 and 55 epochs separately.

Inference. There exist two ways for inference: *three-crop* and *ten-crop* testing. a) *Three-crop* testing refers to three random crops of size 256×256 from the original frames, which are resized firstly to have 256 pixels in their shorter sides. *Three-crop* testing is used as the approximation of spatially fully-convolutional testing as in [25, 32, 5]. b) *Ten-crop* testing basically follows the procedure of [31], which extracts 5 crops of size 224×224 and flips these crops. Specially, we conduct *three-crop* testing on Kinetics-400. We also uniformly sample 10 clips of the whole video and average the softmax probabilities of all clips as the final prediction. For the other two datasets, *ten-crop* testing and TSN-like methods with 8 segments are adopted.

Backbone. We evaluate TPN on both 2D and 3D backbone networks. Specifically, the *slow-only* branch of SlowFast [5] is applied as our backbone network (denoted as I3D) due to its promising performance on various datasets. The architecture of I3D is shown in Table 1, which turns the 2D ResNet [12] into a 3D version via inflating kernels [32, 1]. Specifically, a 2D kernel of size $k \times k$ will be inflated to have the size $t \times k \times k$, with its original weights copied for t times and rescaled by $1/t$. Note that there are no temporal downsampling operations in the *slow-only* backbone. ResNet-50 [12] is used as 2D backbone to show that TPN could combine with various backbones. The final prediction follows the standard protocol of TSN [31] unless specified.

4.1. Results

Results on Kinetics-400. We compare our TPN with other state-of-the-art methods on Kinetics-400. The *multi-depth* pyramid and the parallel flow are used as the default setting for TPN. In detail, the multi-depth pyramid is built on the outputs of *res4* and *res5*. And the hyper-parameters $\{\alpha_i\}_{i=1}^{M}$ are set to be $\{16, 32\}$. As discussed in the spatial semantic modulation, an additional auxiliary head is applied on the output of res4 with a balancing coefficient of 0.5. Sampling intervals of input frames $\tau = 8, 4, 2$ are compared.

The performance of I3D-R50 + TPN (*i.e*. TPN-R50) is included in Table 2. It is worth noting that in Table 2 backbones of methods with the same depth are slightly different, which also affect their final accuracies. TPN-R50 could achieve 77.7% top-1 accuracy, better than others with the same depth.

Model	Frames	Flow	Top-1	Top-5
R(2+1)D [28]	16	✓	73.9	90.9
I3D [1]	16	✓	71.6	90.0
Two-Stream I3D [1]	64	✓	75.7	92.0
S3D-G [34]	64	✓	77.2	93.0
STC-X101 [4]	32		68.7	88.5
Nonlocal-R50 [32]	32		76.5	92.6
Nonlocal-R101 [32]	32		77.7	93.3
SlowFast-R50 [5]	32		77.0	92.6
SlowFast-R101 [5]	32		77.9	93.2
CSN-101 [27]	32		76.7	92.3
CSN-152 [27]	32		77.8	92.8
TPN-R50	32×2		77.7	93.3
TPN-R101	32×2		**78.9**	**93.9**

Table 2. **Comparison with other state-of-the-art methods on the validation set of Kinetics-400.** Note that R50 and R101 denote the backbone networks and their depth respectively.

Backbone	Segments	Testing	Top-1
TSN-50 from [18]	8	*ten-crop*	69.9
TSM-50 from [18]	8	*ten-crop*	72.8
TSN-50 + TPN	8	*ten-crop*	**73.5**

Table 3. **Improvement of 2D backbone on the validation set of Kinetics-400.** Only 8 segments are used for both training and validation for apple-to-apple comparison.

Backbone	Segments	Top1@V1	Top1@V2
TRN-Multiscale [18]	8	38.9	48.8
ECO [39]	8	39.6	-
TSN-50 [18]	8	19.7	30.0
TSM-50 [18]	8	45.6	59.1
TSN-50 + TPN	8	40.6	55.2
TSM-50 + TPN	8	**49.0**	**62.0**

Table 4. **Results on the validation set of Something-Something V1 & V2.** Note that results on V1 & V2 take the center crop of 1 clip/video according to [18]. And our TPN could also achieve 66.9% Top-1 accuracy on the latest leaderboard.

TPN-R101 are also evaluated with the input setting of 32×2, which obtains an accuracy of 78.9%, surpassing other methods with the same numbers of input frames.

Being a general module, TPN could be combined with 2D networks. To show this, we add TPN to the ResNet-50 [12] in TSN (TSN-50 + TPN), and train such a combination with 8 segments (uniform sampling) in an end-to-end manner. Different from the original TSN [31] which takes 25 segments for validation, we utilize only 8 segments and the *ten-crop* testing, comparing apples to apples. As shown in Table 3, adding TPN to TSN-50 could boost the top-1 accuracy by 3.6%.

Model	Frames	NOUN@1	VERB@1
TSN (RGB) [2]	25	36.8	45.7
TSN (Flow) [2]	25	17.4	42.8
TSN (Fusion) [2]	25	36.7	48.2
TSN (our impl.)	8	39.7	48.2
TSN + TPN	8	**41.3**	**61.1**

Table 5. **Results on the validation set of Epic-Kitchen.** TSN is equipped with TPN.

Results on Something-Something. Results of different baselines with and without TPN on the Something-Something are also included in Table 4. For a fair comparison, we use the center crop of size 224×224 in all 8 segments, following the protocol used in TSM [18]. Both TSN and TSM receive a significant performance boost after combined with the proposed TPN. While TSM has a relatively larger capacity compared to TSN, such consistent improvements on both backbones clearly demonstrate the generality of TPN. Besides, on the leaderboard (dated on 03/20/2020), TPN with backbone of TSM-101_{16f} achieves 66.9% Top-1 accuracy, which ranks at the top of all the methods using RGB frames only and at the third place overall next to two RGB + Flow methods.

Results on Epic-Kitchen. As shown in Table 5, we compare TSN+TPN to two baselines on Epic-Kitchen, following the settings in [2]. Consequently, a similar improvement is observed as in other datasets, especially on verb classification, which has an increase of 12.9%.

4.2. Ablation Study

Ablation studies for the components of TPN are conducted on Kinetics-400. Specifically, the I3D-50 backbone and the sparse sampling strategy (*i.e.* 8×8) are adopted unless specified otherwise.

Which feature source contributes the most to the classification? As is mentioned in Sec.3.1, there exist two alternative ways to collect features from the backbone network, namely *single-depth* and *multi-depth*. For the *single-depth* pyramid, the output of *res5* is sampled along the temporal dimension at $\{1, 2, 4, 8\}$ intervals respectively to construct a four-level feature pyramid. For the *multi-depth* pyramid, we choose three possible combinations as shown in Table 6a. The *parallel flow* is adopted as the default option for feature aggregation. Hyper-parameters $\{\alpha_i\}_{i=1}^{M}$ for the *multi-depth* pyramid are chosen to match its shape with the *single-depth* pyramid. For example, if *res4* and *res5* are selected as feature sources, the hyper-parameters will be $\{4, 8\}$.

The results of using different feature sources are included in Table 6a, which suggests that the performance of TPN will drop when we take features from relatively shallow sources *e.g. res2* or *res3*. Intuitively there are two related factors:

Possible Sources	Top-1	Top-5
None	74.9	91.9
res$\{2, 3, 4, 5\}$	74.6	91.8
res$\{3, 4, 5\}$	74.9	92.1
res$\{4, 5\}$	**76.1**	**92.5**
res$\{5\}$	75.7	92.3

(a) **Possible Sources:** None denotes the I3D baseline with the depth of 50. *res*$\{i\}$ means features are collected from the i-th stage in ResNet [12]. Specially, *res*$\{5\}$ takes the *single-level* pyramid as 3.1.

Head	Spatial	Temporal	Flow	Top-1
				74.9
✓				74.6
✓	✓			75.2
✓	✓	✓		75.4
✓	✓	✓	✓	**76.1**
	✓	✓	✓	75.9
		✓	✓	75.6

(b) **Ablation Study on TPN components:** We gradually add auxiliary head (Head), spatial convolutions in semantic modulation (Spatial), temporal rate modulation (Temporal) and information flow (Flow) onto the baseline model.

Information Flow	Top-1	Top-5
Isolation	75.4	92.4
Bottom-up	75.8	92.3
Top-down	75.7	92.3
Cascade	75.9	92.3
Parallel	**76.1**	**92.5**

(c) **Information Flow:** Accuracy of several TPN variants mentioned in Sec 3.2 are shown. The hyper-parameters $\{\alpha_i\}_{i=1}^{M}$ is set as $\{4, 8\}$.

Backbone	$T \times \tau$	w/o TPN	w/ TPN	Δ Acc
I3D-50	8×8	74.9	76.1	+1.2
	16×4	76.1	77.3	+1.2
	32×2	75.7	**77.7**	+2.0
I3D-101	8×8	76.0	77.2	+1.2
	16×4	77.0	78.1	+1.1
	32×2	77.4	**78.9**	+1.5

(d) **Input Frames:** Different number of frames are adopted to evaluate whether TPN has the consistent improvement.

Table 6. **Ablation studies on Kinetics-400.** Backbone is I3D-50 and takes 8×8 frames as input unless specified.

1) different from object detection where the low-level features contribute to the position regression, action recognition mainly relies on high-level semantics. 2) Another factor might be that the I3D backbone [5] only inflates the convolutions in the blocks of *res4* and *res5*, so that both *res2* and *res3* is unable to capture useful temporal information. Unfortunately, inflating all 2D convolutions in the backbone will increase the computational complexity significantly and damage the performance as reported in [5]. Compared to the *multi-depth* pyramid, the *single-depth* pyramid extracts various tempo representations by directly sampling from a single source. Although improvement is also observed, representing video semantics only at a single spatial granularity may be insufficient.

How important are the information flows? In Sec.3.2, several information flows are introduced for feature aggregation. Table 6c lists the performances of different information flows, keeping other components of TPN unmodified. Surprisingly, TPN with the *Isolation Flow* also boosts the performance by 0.58%, indicating that under proper modulations, the features with different temporal receptive fields indeed could help action recognition, even they come from a single backbone network. TPN with the *Parallel Flow* obtains the best result, leading to a performance of 76.1%. The success of parallel flow suggests that lower-level features could be enhanced by higher-level features via the top-down flow for they have larger temporal receptive fields. The semantics of higher-level features could also be enriched by lower-level features via the bottom-up flow. More importantly, such two opposing information flows are not contradictive but complementary to each other.

How important are spatial semantic modulation and temporal rate modulation? The spatial semantic modulation and the temporal rate modulation are respectively introduced to overcome the semantic inconsistency in spatial dimensions and to adjust the relative rates of different levels in the temporal dimension. The effect of these two modulations are studied in Table 6b, from which we observe: 1) TPN with all the components lead to the best result. 2) if the spatial semantic modulation contains no spatial convolutions, we have to up/down-sample the features of TPN simultaneously at spatial and temporal dimensions, which is ineffective for temporal feature aggregation.

How important is the number of input frames? While we use 8 frames sampled at the stride of 8 as the default input in our study experiments, we have also investigated different sample schemes. We denote $T \times \tau$ as T frames sampled with the stride of τ. And in Table 6d, we include results of both I3D-50 and I3D-101 with inputs obtained by different sample schemes. Consequently, compared to the sparser sampling scheme (8×8), the denser sampling scheme (32×2) tends to bring it both rich and redundant information, leading to a slight over-fitting of I3D-50. I3D-50 + TPN, however, does not encounter such an over-fitting, obtaining

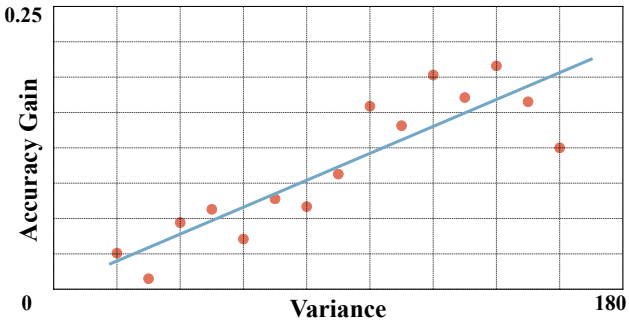

Figure 4. **Performance Gain *vs*. Variance of Visual Tempos.** Each red point denotes the mean accuracy gain within a bin of variance. The blue line is the result of least squares approximation.

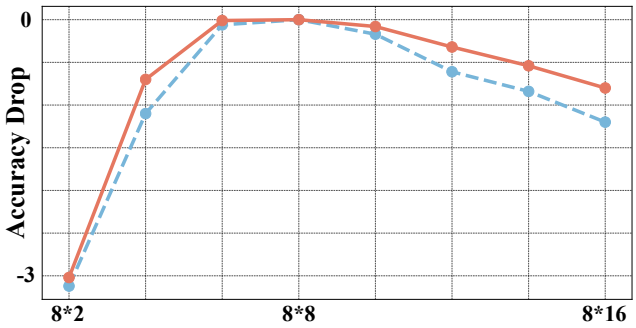

Figure 5. **Robustness to Variance of Visual Tempos.** Both baseline and TPN models are trained on 8×8 frames. The red line pictures the performance drop of baseline with TPN While the blue dash line denotes that of baseline only.

an increase of 2%. Moreover, consistent improvements are observed for the stronger backbone I3D-101.

4.3. Empirical Analysis

To verify whether TPN has captured the variance of visual tempos, several empirical analyses are conducted on TPN.

Per-class performance gain *vs*. per-class variance of visual tempos. At first, we have to measure the variance of visual tempos for a set of action instances. Unlike the concept of scale in object detection, it is non-trivial to precisely compute the visual tempo of an action instance. Therefore, we propose a model-based measurement that utilizes the Full Width at Half Maximum (FWHM) of the frame-wise classification probability curve. FWHM is defined by the difference between the two points of a variable where its value is equal to half of its maximum value. We use a trained 2D TSN to collect per-frame classification probabilities for action instances in the validation set, and compute the FWHM for each instance as a measurement of its visual tempo, since when the sampling fps is fixed, a large FWHM intuitively means the action is going with a slow tempo, vice versa. We thus could compute the variance of visual tempos for each action category. The bottom in Figure 1 shows the variances of visual tempos of all action categories, which reveals that not only the variance of visual tempos is large for some categories, different categories also have significantly different variances of visual tempos.

Subsequently, we also estimate the correlations between per-class performance gains when adopting a TPN module and per-class variances of visual tempos. We at first smooth the bar chart in Figure 1 by dividing them into bins with an interval of 10. We then calculate the mean of performance gains in each bin. Finally, the statistics of all bins is shown in Figure 4, where performance gain is positively correlated with variance of visual tempos. This study strongly supports our motivation that TPN could bring a significant improvement for such actions with large variances of visual tempo.

Robustness of TPN to visual tempo variation. Human recognizes actions easily in spite of the large variance of the visual tempos. Does the proposed TPN module also possess such robustness? To study this, we at first train a I3D-50 + TPN on Kinetics-400 [1] with 8×8 ($T \times \tau$) frames as the input. We then re-scale the original 8×8 input by re-sample the frames with stride τ equals to $\{2, 4, 6, 10, 12, 14, 16\}$ respectively, so that we are adjusting the visual tempo of a given action instance. For instance, when feeding frames sampled as 8×16 or 8×2 into the trained I3D-50 + TPN, we are essentially speeding up / slowing down the original action instance since the temporal scope increases/decreases relatively. Figure 5 includes the accuracy curves of varying visual tempos for I3D-50 and I3D-50 + TPN, from which we can see TPN help improve the robustness of I3D-50, resulting in a curve with moderator fluctuations. Moreover, the robustness to the visual tempo variation becomes clearer as we vary the visual tempo harder, as TPN could adapt itself dynamically according to the need.

5. Conclusion

In this paper, a generic module called Temporal Pyramid Network is proposed to capture the visual tempos of action instances. Our TPN, as a feature-level pyramid, can be applied to existing 2D/3D architectures in the plug-and-play manner, bringing consistent improvements. Empirical analyses reveal the effectiveness of TPN, supporting our motivation and design. We will extend TPN for other video understanding tasks in the future work.

Acknowledgments. This work is supported in part by CUHK Direct Grant and SenseTime Group Limited. We also thank Yue Zhao for the wonderful codebase and insightful discussion.

References

[1] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proc. CVPR*, 2017. 2, 4, 5, 6, 8

[2] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. In *Proc. ECCV*, 2018. 2, 5, 6

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009. 5

[4] 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 *Proc. ECCV*, 2018. 6

[5] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. *Proc. ICCV*, 2019. 1, 2, 3, 4, 5, 6, 7

[6] Christoph Feichtenhofer, Axel Pinz, and Richard P Wildes. Spatiotemporal multiplier networks for video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4768–4777, 2017. 2

[7] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proc. CVPR*, 2016. 2

[8] Antonino Furnari and Giovanni Maria Farinella. What would you expect? anticipating egocentric actions with rolling-unrolling lstms and modality attention. In *Proc. ICCV*, 2019. 5

[9] 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. 5

[10] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *Proc. ICCV*, 2017. 2, 5

[11] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. 2015. 2

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016. 4, 5, 6, 7

[13] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012. 5

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning*, 2015. 5

[15] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2012. 2

[16] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proc. CVPR*, 2014. 2

[17] Hongyang Li, Bo Dai, Shaoshuai Shi, Wanli Ouyang, and Xiaogang Wang. Feature intertwiners. In *International Conference on Learning Representations*, 2019. 2

[18] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proc. ICCV*, 2019. 2, 6

[19] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proc. CVPR*, 2017. 2

[20] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proc. CVPR*, 2018. 2, 4

[21] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *Proc. ICCV*, 2017. 2

[22] Dian Shao, Yue Zhao, Bo Dai, and Dahua Lin. Finegym: A hierarchical video dataset for fine-grained action understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[23] Dian Shao, Yue Zhao, Bo Dai, and Dahua Lin. Intra- and inter-action understanding via temporal action parsing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[24] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, 2014. 2

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 5

[26] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proc. ICCV*, 2015. 2

[27] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *Proc. ICCV*, 2019. 6

[28] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proc. CVPR*, 2018. 2, 6

[29] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *Proc. ICCV*, 2013. 2

[30] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Proc. CVPR*, 2015. 1, 2

[31] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *Proc. ECCV*, 2016. 2, 5, 6

[32] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proc. CVPR*, 2018. 1, 2, 5, 6

[33] Yunbo Wang, Mingsheng Long, Jianmin Wang, and Philip S Yu. Spatiotemporal pyramid network for video action recognition. In *Proc. CVPR*, 2017. 1, 2

[34] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proc. ECCV*, 2018. 2, 6

[35] Da Zhang, Xiyang Dai, and Yuan-Fang Wang. Dynamic temporal pyramid network: A closer look at multi-scale modeling for activity detection. In *Asian Conference on Computer Vision*, pages 712–728, 2018. 1, 2

[36] Yue Zhao, Yuanjun Xiong, and Dahua Lin. Trajectory convolution for action recognition. In *Advances in Neural Information Processing Systems*, 2018. 1, 2

[37] Yue Zhao, Yuanjun Xiong, and Dahua Lin. Mmaction. https://github.com/open-mmlab/mmaction, 2019. 5

[38] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *Proc. ECCV*, 2018. 2

[39] Mohammadreza Zolfaghari, Kamaljeet Singh, and Thomas Brox. Eco: Efficient convolutional network for online video understanding. In *Proc. ECCV*, 2018. 6