Right on Time: Multi-Temporal Convolutions for Human Action Recognition in Videos

Alexandros Stergiou, Ronald Poppe

1Utrecht University
Buys Ballot Building, Princetoplein 5,
3584 CC, Utrecht, The Netherlands
{a.g.stergiou, r.w.poppe}@uu.nl

Abstract

The variations in the temporal performance of human actions observed in videos present challenges for their extraction using fixed-sized convolution kernels in CNNs. We present an approach that is more flexible in terms of processing the input at multiple timescales. We introduce Multi-Temporal networks that model spatio-temporal patterns of different temporal durations at each layer. To this end, they employ novel 3D convolution (MTConv) blocks that consist of a short stream for local space-time features and a long stream for features spanning across longer times. By aligning features of each stream with respect to the global motion patterns using recurrent cells, we can discover temporally coherent spatio-temporal features with varying durations. We further introduce sub-streams within each of the block pathways to reduce the computation requirements. The proposed MTNet architectures outperform state-of-the-art 3D-CNNs on five action recognition benchmark datasets. Notably, we achieve at 87.22% top-1 accuracy on HACS, and 58.39% top-1 at Kinectics-700. We further demonstrate the favorable computational requirements. Using sub-streams, we can further achieve a drastic reduction in parameters (~60%) and GLOPs (~74%). Experiments using transfer learning finally verify the generalization capabilities of the multi-temporal features.

Introduction

Disparities on how humans perform tasks and interact present significant challenges for the recognition of such actions in video (Herath, Harandi, and Porikli 2017; Stergiou and Poppe 2019a). Much of the differences in visual appearance can be captured by the hierarchical convolution operations in deep convolutional neural networks (CNNs). For action recognition in videos, 2D convolutions have been successfully extended to 3D convolutions to additionally extract informative patterns over time. However, although connections between the three dimensions do exist (Dong and Atick 1995), treating the temporal information uniformly to spatial information comes with significant limitations towards modeling variations in the execution of actions over time. For example, Figure 1 presents frames of three basketball passes within the same game. While the spatial scale of the action is similar, the duration of the action varies between the three examples. Similar observations can be made for virtually all other actions.

Current efforts in action recognition are based on 3D convolutions (Ji et al. 2013) with fixed-sized kernels. Through the hierarchical application of convolutions, the receptive field of the kernels in both spatial and temporal dimensions is enlarged in deeper layers. The variation in temporal performance is significant but not in the order of magnitude, and to some extend unrelated to spatial variations. By focusing on the differences in temporal movements that we expect to observe between videos, we can be more flexible in extracting spatio-temporal patterns at different timescales while keeping the spatial modeling power unaffected.

In this work, we improve the current spatio-temporal feature extraction process by introducing novel MTConv convolution blocks that split features into two streams focusing on short-term and longer-term movements, respectively (Figure 2). Additionally, these two streams are aligned with global motion information. By downsampling the activations in the long stream, we effectively and efficiently create larger temporal receptive fields. MTConv allows for the discovery of multi-temporal patterns at each layer of the network and additionally minimize the computational footprint of 3D-CNNs. Our contributions are as follows:

• Introduction of novel MTConv convolution blocks that enable the extraction of complex space-time features at dif-
have shown to scale well to deeper networks that can model highly complex spatio-temporal features (Hara, Kataoka, and Satoh 2018, Kataoka et al. 2020).

A significant drawback of 3D convolutions is their large computational requirements and much of the recent literature is aimed at efficiency improvements for 3D-CNNs. Works by Tran et al. (2018) and Qiu, Yao, and Mei (2017) have considered decoupling 3D convolutions into temporally and spatially, thus strongly reducing the number of trainable parameters. Others have decoupled convolutions based on horizontal and vertical motions (Li et al. 2019, Stergiou and Poppe 2019b). These approaches have additionally led to performance improvements. Recently, 3D grouped convolutions (Chen et al. 2018, Tran et al. 2019) have shown the benefits of performing convolutions in groups to decrease the number of GFLOPs without losing accuracy. Another group-based approach has considered temporal shifting frame activations (Lin, Gan, and Han 2019), emulating the effects of 3D convolutions while relying on frame-based 2D convolutions. This temporal shift module (TSM) has recently been combined with a 3D convolution gating method to enable or disable gate shifting (Sudhakaran, Escalera, and Lanz 2020).

Combining 3D convolutions and sub-streams. Although the two-stream optical-flow 2D CNNs have been largely surpassed by 3D-CNN architectures, recent works have focused on using 3D convolutions in a stream-based manner (Carreira and Zisserman 2017). Feichtenhofer et al. (2019) proposed a network based on frame sampling of slow and fast frame rates as two adjacent inputs to a two-stream 3D-CNN. With the same rationale in mind, Qiu et al. (2019) have shown how global paths using entire videos as inputs and local paths with local spatio-temporal segments can be used in two separate network pathways. Others have considered block-based approaches with octave convolutions (Chen et al. 2019) to model temporal variation in the frequency domain.

These methods have shown great promise towards the extraction of robust spatio-temporal features, while simultaneously reducing the overall computational requirements. Yet, they do not explicitly address the modeling of variations in the duration of the actions, nor do they focus on the extraction of coherent spatio-temporal features. Because existing network models rely only on a linear flow of information, complex temporal motion over extended periods of time cannot be recognized effectively. To solve this issue, we extend the two-stream approach with 3D convolutions that simultaneously operate on short-term and longer-term spatio-temporal patterns. We align the activations using the Squeeze and Recursion approach (Stergiou and Poppe 2020) to enforce coherence on the learned spatio-temporal patterns, despite variations in temporal duration.

Multi-Temporal Networks

In this section, we introduce multi-temporal convolutions (MTConvs) and the structure of the blocks in which they operate (MTBlocks, shown in Figure 3). We then detail the network architectures (MTNets) that are built with these blocks. All layer and block inputs \(a(C \times T \times H \times W)\) are volumes of \(C\).
feature channels, \( T \) frames, and with a height and width of \( H \) and \( W \), respectively. MTConvs use sub-volumes that are denoted by \( a_s \) and \( a_l \) for the input of the short-term and longer-term streams, respectively. Layer numbers \( (i) \) are indicated as \( a_s^{[i]} \) and \( a_l^{[i]} \) or \( a_t^{[i]} \).

**MT convolutions**

MTConvs use a dual-stream (or dual pathway) approach for feature extraction at the layer level, not as two pathways throughout a network. A user-set channel ratio parameter \( r \) decouples the input spatio-temporal activation maps into the short and the long streams. Each of the streams include a portion of the layer channels \( \mathbf{C} \), based on Equation \( 1 \) to ensure that the resulting channels \( C_s \) and \( C_l \) satisfy the conditions \( C_s \geq C_l \) and \( 0.5 \leq r < 1 \) respectively. Values \( r \geq 0.5 \) are chosen because the long stream acts complementary to the short one, modeling more abstract motions over larger time spans. To this end, it has a larger temporal receptive field but a lower number of input channels.

\[
C \approx C_s + C_l, \text{ where } \quad C_s = [r \times \mathbf{C}] \text{ and } C_l = [(1 - r) \times \mathbf{C}]
\]  

(1)

Short-term spatio-temporal patterns are extracted in the short stream, similar to vanilla 3D convolutions. Their activation maps are \( T \times H \times W \). By including the majority of the channels \( C_s \), the short stream can make distinctions based on detailed spatio-temporal features. However, patterns in larger space-time scales are not addressed due to the limited kernel size.

The long stream uses a downsampled version of the original volume, with size \( T/2 \times H/2 \times W/2 \). Consequently, with the same kernel size as the short stream, the receptive field is twice as large. This effectively makes the long stream complementary to the short one as it models variations with larger durations over larger regions, while further maintaining low computational complexity by only using a fraction of the original \( C \) channels with \( C_l \) channels.

The output activations of both streams are calculated with Equation \( 2 \) with batch normalization \((bn)\) and a ReLU activation \((relu)\) as well as the inclusion of cross-stream connection at the final term:

\[
a_s^{[i]} = relu(bn(a_s^{[i-1]} \ast W_s^T))
\]

\[
a_l^{[i]} = relu(bn(a_l^{[i-1]} \ast W_l^T) + bn(pool(a_s^{[i-1]} \ast W_s^{T/2})))
\]

(2)

**Cross-stream connections.** The general purpose of establishing cross-stream connections is to enrich the activation maps while simultaneously minimizing the divergence between the feature spaces of the short and long streams. This is achieved through lateral connections from the short to the long stream that include downsampling activations to a size of \( T/2 \times H/2 \times W/2 \) and using point-wise spatio-temporal convolutions \((1 \times 1 \times 1)\) for transitioning from feature-space \( C_s \) to feature-space \( C_l \). This effectively aligns long-stream features with those from the short stream.
Based on the MTBlock type (see next section), two downsampling approaches are used. First, repeat blocks use average pooling to produce spatio-temporally reduced activations based on uniform local patches. This type of pooling is suitable as the block input and number of output channels is the same and therefore the features exhibit a modest degree of variation.

Second, downsampling blocks use smooth maximum approximation kernels. With this method, named SoftPool, larger activations have greater effect on the downsampled volume than lower activations. This follows the notion that, in channel-increasing blocks, it is more sensible to highlight the new activations that now have a greater effect than the block’s output. This creates a feature amplification, similar to Chen et al. (2018), in which spatio-temporal patterns are then extracted from the amplified features. The last convolution in the layer is responsible for increasing the feature complexity (He et al. 2016; Huang et al. 2017; Feichtenhofer 2020; Radosavovic et al. 2020). MTNets thus have initial downsampling MTBlocks, and contain repeat MTBlocks in deeper layers. The block combinations of various network configurations can be seen in Table 1, while the architecture of each block is summarized in Figure 3.

### MTBlock variants

The proposed MTNet architectures follow the widely used structure with initial layer blocks that decrease the dimensionality of the inputs while increasing the number of features, and proceeding blocks used to hierarchically increase the feature complexity (He et al. 2016, Huang et al. 2017, Feichtenhofer 2020, Radosavovic et al. 2020). MTNets thus have initial downsampling MTBlocks, and contain repeat MTBlocks in deeper layers. The block combinations of various network configurations can be seen in Table 1, while the architecture of each block is summarized in Figure 3.

### Dowsnample blocks

The structure for this block is similar to residual blocks of ResNets with the exception of using two $1 \times 1 \times 1$ convolutions at the beginning to reduce the computational cost by $C_k/2$. Here, $C_k$ is the factor based on which the outputs of the first convolution will be smaller than the block’s output. This creates a feature amplification, similar to Chen et al. (2018), in which spatio-temporal patterns are then extracted from the amplified features. The last convolution in the layer is responsible for increasing the number of channels. In all of our networks, channel numbers are increased by a factor of two.

### Repeat blocks

We use a number of repeat blocks that incorporate both spatio-temporal and purely-spatial kernels. This kernel combination is motivated by recent successes with the decoupling of 3D convolutions (e.g., Lin, Gan,
and Han (2019); Shan et al. (2020); Tran et al. (2018)), as well as the inclusion of spatial-only kernels (Zhou et al. 2018) in each stream, as the appearance aspect plays an integral part when analyzing videos. This decoupling also provides a large reduction in the number of parameters and computation requirements while including more operations (4 \times \text{Conv} + 2 \times \text{GRU}) in comparison to Residual blocks (3 \times \text{Conv}) or Dense blocks (2 \times \text{Conv}). This increases the non-linear modeling capabilities of the blocks.

Squeeze and Recursion. Because channel dependencies across time can be effectively modeled with sequential chains of recurrent operations, we include recurrent units at each stream that can uncover generally informative video features. We use Squeeze and Recursion (Stergiou and Poppe 2020) as the alignment mechanism to extract globally coherent features. We replace the originally proposed LSTMs (Hochreiter and Schmidhuber 1997) with Gated Recurrent Units (GRU, Cho et al. 2014). The resulting cell is shown in Figure 4.

The cell contains an update gate $(z(t))$, that uses the per-instance input $(\text{pool}(a[l])_{(t)})$ and the previous cell state $(h_{t-l-1})$ to pass relative information forward. In addition, a reset gate $(r(t))$ is used to ignore less time-consistent features. The update and reset gates act in a complementary manner on the same inputs. Based on the activations produced by the reset gate, a candidate hidden state is computed $(\tilde{h}(t))$ with reduced influence from the previous state $(h_{t-l-1})$ based on $r(t)$. The produced state is the fusion of a proportion of the previous state $(z(t) \ast h_{t-l-1})$ and the supplementary portion of the candidate hidden state $((1-z(t)) \ast \tilde{h}(t))$. This process is summarized in Equation 4:

\[
\begin{align*}
    z(t) &= \{\sigma(W_x \ast [h_{t-l-1}, \text{pool}(a[l])_{(t)}] + b_z)\} \\
    r(t) &= \{\sigma(W_r \ast [h_{t-l-1}, \text{pool}(a[l])_{(t)}] + b_r)\} \\
    \tilde{h}(t) &= \tanh(W_h \ast [r(t) \ast h_{t-l-1}, \text{pool}(a[l])_{(t)}] + b_h) \\
    h(t) &= z(t) \ast h_{t-l-1} + (1-z(t)) \ast \tilde{h}(t) \tag{4}
\end{align*}
\]

Experiments and Results

We evaluate the proposed MTNets on five popular action recognition benchmark datasets. We compare MTNets with different channel ratio ($r$) values as well as vanilla 3D convolutions based on their achieved classification accuracy and computational complexity.

Datasets

The five action recognition datasets that we use differ in terms of the number of videos, number of classes, and the variation in the performance of the actions:

- **HACS**: Human Action Clips Segments (Zhao et al. 2019) includes ~500K action segments (from 20k video) of 200 different actions. All clips are 60 frames long.
- **K-700**: Kinetics-700 (Carreira et al. 2019) is the continuation of the previous 600/400 variants and includes ~600k clips of 700 different classes.
- **MIT**: Moments in Time (Monfort et al. 2019) contains 399 classes of over 800k, 3 second-long videos.
- **UCF-101** (Soomro, Zamir, and Shah 2012) includes 101 action classes and a total of ~13k videos.
- **HMDB-51** (Kuehne et al. 2011) contains ~7k videos of 51 classes.

Experiments setup

We use a random crop of size $320 \times 320$ with a 16 frame sample with an interval of 2 frames. We adopt a multi-grid training scheme (Wu et al. 2020) to have an average clip size of $\approx 8 \times 180 \times 180$ over all cycles, and an initial learning rate of 0.1 with a step size decrease of 0.1. We used SGD with $1e^{-6}$ weight decay. For all our experiments, we set the batch size to 32. For HACS, all weights were initialized with a standard Kaiming initialization (He et al. 2015) while for K-700, MIT, UCF-101 and HMDB-51, weights were initialized based on the models trained on HACS.

Streams ablation study

We perform a number of experiments to study the effect of the channel ratio ($r$) between the two streams. Results are presented in Table 2.

| Rate | Stream(s) | MTNet-32 | MTNet-48 | MTNet-64 | MTNet-132 |
|------|-----------|----------|----------|----------|-----------|
| r = 0.5 | 1 \times 1 | 69.760 | 71.368 | 73.732 | 74.981 |
|      | 4 \times 1 | 68.304 | 70.054 | 72.161 | 73.873 |
| r = 0.625 | 1 \times 1 | 71.482 | 75.262 | 78.403 | 78.892 |
|      | 5 \times 5 | 71.767 | 75.562 | 77.698 | 78.262 |
| r = 0.75 | 1 \times 1 | 74.217 | 77.875 | 83.743 | 84.376 |
|      | 6 \times 2 | 73.883 | 77.149 | 81.726 | 82.287 |
| r = 0.875 | 1 \times 1 | 76.368 | 79.312 | 85.625 | 86.890 |
|      | 7 \times 7 | 75.963 | 79.259 | 84.642 | 85.920 |
| (vanilla 3D) | 8 | 73.452 | 76.631 | 82.268 | 82.696 |
Ratio selection. The top performing short and long stream configurations include ratios of $r = 0.875$ meaning a channel distribution of 7:1 between the short and long streams. This yields an average improvement of +1.9% over vanilla 3D convolutions while reducing the average GFLOPs across the four tested architectures by 20.52%. Further improvements in computational costs can be achieved when using a ratio of $r = 0.75$ with a 3:1 short to long channel distribution. Compared to vanilla 3D convolutions ($r = 1$), this results in an average reduction of 35.06% in GFLOPs, with only a very small decrease in top-1 accuracy (< 0.5%).

Sub-stream evaluation. The incorporation of sub-streams ($g = 8$) in the short and long streams has shown great potential to reduce the computation cost, with almost negligible effects on the classification accuracy. This is evident from Table 3 where the inclusion of sub-streams in MTBlocks decreases the number of parameters by 58.67% and 59.98% and reduces GFLOPs by 71.82% and 76.14% for the 64 and 132-layer MTNets, respectively. Despite the strong reduction in parameters and computation costs, these networks maintain high performance on HACS, Kinetics-700 and Moments in Time. This makes sub-stream MTNets an efficient compute-to-performance alternative in settings with limited resources. Figure 5 provides a summary of the effect of channel ratio $r$ and the number of sub-streams $g$.

Network comparisons

We demonstrate the performance gains across the five proposed architectures on the three largest benchmark datasets in Table 4. Models for HACS are trained from scratch. These models are used to initialize the weights for K-700 and MiT. MTNet-132 and MTNet-200 surpass current architectures on all three datasets with improvements in the range of 1.59-2.89% on HACS, 0.27-1.06% on K-700 and 0.99-1.96% on MiT. The significantly smaller MTNet-64 also outperforms current models on HACS, similarly to the sub-stream variants of MTNet-64 and MTNet-132.

Table 3: K-700 & MiT top-1 accuracies, number of parameters, GFLOPs and latency for different configurations of layers and number of sub-streams.

| MTNet | Params (M) | GFLOPs | Latency (msec) | top-1(%) |
|-------|------------|--------|----------------|---------|
|       | F          | B      | K-700          | MiT     |
| 32    | 21.64      | 35.1   | 27.81          | 56.88   | 49.831 | 26.492 |
| 64    | 39.78      | 42.80  | 40.93          | 65.62   | 53.393 | 29.076 |
| 132   | 43.02      | 64.53  | 53.17          | 95.40   | 56.964 | 32.239 |
| 200   | 85.30      | 101.55 | 103.4          | 164.44  | 57.603 | 34.708 |
| 132   | 124.25     | 144.08 | 152.59         | 218.95  | 58.387 | 35.524 |
| 64    | 24.68      | 11.07  | 9.16           | 47.96   | 54.814 | 31.890 |

Figure 6: Top-1 K-700 accuracy vs. clip loading latency. Small latency time equals faster training times. Latency should be considered w.r.t. the number of layers used.

We additionally study the efficiency of the proposed MT-Blocks and MTNets based on the per-video latency for a full forward pass, shown in Figure 7. In proportion to the number of layers used by each of the MTNet architectures, their latencies remain consistently lower than other networks that use vanilla 3D convolutions such as the 3D and (2+1)D variants of Resnets as well as I3D. We also observe that the
use of sub-streams in the tested MTNet-64 and MTNet-132 neither hinders nor improves the latency times significantly. This demonstrates their value in low-power scenarios.

Table 5: Action recognition accuracy on UCF-101 and HMDB-51, after pre-training.

| Model | Pre-training | UCF-101 | HMDB-51 |
|-------|--------------|---------|---------|
|       | top-1 (%)    | top-5 (%) | top-1 (%) | top-5 (%) |
| r3d-50 (Kataoka et al. 2020) | 93.126 | 96.293 | 72.192 | 94.342 |
| r3d-101 (Kataoka et al. 2020) | 95.756 | 98.423 | 79.560 | 95.910 |
| r(2+1)d-50 (Tran et al. 2018) | 93.923 | 97.834 | 73.056 | 94.381 |
| r(2+1)d-101 (Tran et al. 2018) | 95.503 | 98.705 | 75.837 | 95.312 |
| i3d (Carreira and Zisserman 2017) | 92.453 | 97.619 | 71.768 | 94.128 |
| ir-CSN-152 (Tran et al. 2019) | 92.130 | 97.981 | 72.391 | 94.138 |
| SF r3d-50 (Feichtenhofer et al. 2019) | 96.564 | 99.138 | 74.205 | 95.974 |
| SF r3d-101 (Feichtenhofer et al. 2019) | 97.325 | 99.557 | 77.536 | 96.253 |
| SRTG r3d-101 (Stergiou and Poppe 2020) | 98.871 | 99.871 | 78.891 | 96.319 |
| SRTG (r(2+1)d-101) (Stergiou and Poppe 2020) | 97.954 | 99.825 | 76.185 | 96.225 |

Table 4: Action recognition top-1 and top-5 accuracies over HACS, K-700 and MiT. Models on HACS are trained from scratch.

| Model | Input | HACS | K-700 | MiT |
|-------|-------|------|-------|-----|
|       | (T × (H × W)) | top-1 (%) | top-5 (%) | top-1 (%) | top-5 (%) |
| r3d-34 (Kataoka et al. 2020) | 16 × 224² | 74.818 | 92.839 | 46.138 | 67.108 |
| r3d-50 (Kataoka et al. 2020) | 16 × 224² | 78.361 | 93.763 | 49.083 | 72.541 |
| r3d-101 (Kataoka et al. 2020) | 16 × 224² | 80.402 | 95.179 | 52.583 | 74.631 |
| r(2+1)d-34 (Tran et al. 2018) | 16 × 224² | 75.703 | 93.571 | 46.625 | 68.229 |
| r(2+1)d-50 (Tran et al. 2018) | 16 × 224² | 81.340 | 94.514 | 49.927 | 73.396 |
| i3d (Carreira and Zisserman 2017) | 16 × 224² | 82.957 | 95.683 | 52.536 | 75.177 |
| SF r3d-50 (2x nets) (Feichtenhofer et al. 2019) | 16 × 224² | 79.948 | 94.482 | 53.015 | 69.193 |
| SF r3d-101 (2x nets) (Feichtenhofer et al. 2019) | 16 × 224² | 81.659 | 96.326 | 56.462 | 76.819 |
| SRTG r3d-101 (Stergiou and Poppe 2020) | 16 × 224² | 84.326 | 96.852 | 56.826 | 77.439 |
| SRTG (r(2+1)d-101) (Stergiou and Poppe 2020) | 16 × 224² | 87.224 | 97.585 | 58.387 | 77.647 |

Feature transferability with MTNets

An important aspect of CNNs is their ability to be fine-tuned on a target dataset after pre-training on a source dataset. To test the transfer learning capabilities of MTNets, we use pre-trained weights from the networks trained on HACS and Kinetics-700 and we fine-tune on the smaller UCF-101 and HMDB-51. Accuracy comparisons of MTNets and other models that were fine-tuned similarly appear in Table 5.

For both UCF-101 and HMDB-51, MTNets-132 and MTNets-200 achieve the best performance after fine-tuning. These results indicate that the learned spatio-temporal features are general, which is a favorable characteristic. The improvements of MTNets are evident in comparison to models pre-trained on same datasets, such as r3d and r(2+1)d networks, as well as models pre-trained on significantly larger datasets such as ir-CSN that was pre-trained on 65M videos (Ghadiyaram, Tran, and Mahajan 2019). This demonstrates that the selection of space-time features with varying temporal durations is effective.

Conclusions

We have focused on modeling the variations in the temporal performance of human actions in videos. Our proposed CNN-based approach uses spatio-temporal information of different time scales in two parallel streams in novel MTConv and MTBlocks, with each stream being calibrated based on global motion using Squeeze and Recursion blocks. Both streams can be divided into multiple sub-streams. This improves the computational efficiency, with 60% fewer parameters and 74% fewer GFLOPs, and only a marginal accuracy decrease.

The derived multi-temporal networks (MTNets) consistently perform on-par or outperform state-of-the-art networks on HACS, K-700 MIT, UCF-101 and HMDB-51 action recognition datasets. We further demonstrated the efficiency gains originating from the low latencies of MTNets. Excellent performance in transfer-learning experiments finally demonstrates the generalization ability of the MTNets.
suggesting that the learned features are meaningful.

Acknowledgments

This publication is supported by the Netherlands Organization for Scientific Research (NWO) with a TOP-C2 grant for Automatic recognition of bodily interactions (ARBITER).

References

Baccouche, M.; Mamalet, F.; Wolf, C.; Garcia, C.; and Baskurt, A. 2011. Sequential deep learning for human action recognition. In International Workshop on Human Behavior Understanding (HBU), 29–39. Springer.

Carreira, J.; Noland, E.; Hillier, C.; and Zisserman, A. 2019. A short note on the Kinetics-700 human action dataset. arXiv preprint arXiv:1907.06987.

Carreira, J.; and Zisserman, A. 2017. Quo vadis, action recognition? A new model and the Kinetics dataset. In Computer Vision and Pattern Recognition (CVPR), 4724–4733. IEEE.

Chen, Y.; Fan, H.; Xu, B.; Yan, Z.; Kalantidis, Y.; Rohrbach, M.; Yan, S.; and Feng, J. 2019. Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. In International Conference on Computer Vision (ICCV), 3435–3444. IEEE.

Chen, Y.; Kalantidis, Y.; Li, J.; Yan, S.; and Feng, J. 2018. Multi-Fiber networks for Video Recognition. In European Conference on Computer Vision (ECCV), 352–367.

Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In Conference on Empirical Methods in Natural Language Processing (EMNLP), 1724–1734.

Diba, A.; Sharma, V.; and Van Gool, L. 2017. Deep temporal linear encoding networks. In Computer Vision and Pattern Recognition (CVPR), 2329–2338. IEEE.

Dong, D. W.; and Atick, J. J. 1995. Statistics of natural time-varying images. Network: Computation in Neural Systems 6(3): 345–358.

Feichtenhofer, C. 2020. X3D: Expanding Architectures for Efficient Video Recognition. In Conference on Computer Vision and Pattern Recognition (CVPR), 203–213. IEEE.

Feichtenhofer, C.; Fan, H.; Malik, J.; and He, K. 2019. SlowFast networks for video recognition. In International Conference on Computer Vision (ICCV), 6202–6211. IEEE.

Feichtenhofer, C.; Pinz, A.; and Wilde, R. 2016. Spatiotemporal residual networks for video action recognition. In Advances in Neural Information Processing Systems (NIPS), 3468–3476.

Ghadiyaram, D.; Tran, D.; and Mahajan, D. 2019. Large-Scale Weakly-Supervised Pre-Training for Video Action Recognition. In Conference on Computer Vision and Pattern Recognition (CVPR). IEEE.
Shan, K.; Wang, Y.; Wang, Z.; Liang, T.; Tang, Z.; Chen, Y.; and Li, Y. 2020. MixTConv: Mixed Temporal Convolutional Kernels for Efficient Action Recognition. arXiv preprint arXiv:2001.06769.

Simonyan, K.; and Zisserman, A. 2014. Two-stream convolutional networks for action recognition in videos. In Advances in Neural Information Processing Systems (NIPS), 568–576.

Soomro, K.; Zamir, A. R.; and Shah, M. 2012. UCF101: A dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:1212.0402.

Stergiou, A.; and Poppe, R. 2019a. Analyzing human-human interactions: A survey. Computer Vision and Image Understanding 188: 102799.

Stergiou, A.; and Poppe, R. 2019b. Spatio-Temporal FAST 3D Convolutions for Human Action Recognition. In International Conference on Machine Learning Applications (ICMLA), 1830–1834. IEEE.

Stergiou, A.; and Poppe, R. 2020. Learn to cycle: Time-consistent feature discovery for action recognition. arXiv preprint arXiv:2006.08247.

Sudhakaran, S.; Escalera, S.; and Lanz, O. 2020. Gate-Shift Networks for Video Action Recognition. In Conference on Computer Vision and Pattern Recognition (CVPR), 1102–1111. IEEE.

Tran, D.; Wang, H.; Torresani, L.; and Feiszli, M. 2019. Video Classification With Channel-Separated Convolutional Networks. In International Conference on Computer Vision (ICCV), 5552–5561. IEEE.

Tran, D.; Wang, H.; Torresani, L.; Ray, J.; LeCun, Y.; and Paluri, M. 2018. A Closer Look at Spatiotemporal Convolutions for Action Recognition. In Conference on Computer Vision and Pattern Recognition (CVPR), 6450–6459. IEEE.

Wang, L.; Xiong, Y.; Wang, Z.; Qiao, Y.; Lin, D.; Tang, X.; and Van Gool, L. 2016. Temporal segment networks: Towards good practices for deep action recognition. In European Conference on Computer Vision (ECCV), 20–36. Springer.

Wu, C.-Y.; Girshick, R.; He, K.; Feichtenhofer, C.; and Krähenbühl, P. 2020. A Multigrid Method for Efficiently Training Video Models. In Conference on Computer Vision and Pattern Recognition (CVPR), 153–162. IEEE.

Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; and He, K. 2017. Aggregated residual transformations for deep neural networks. In Computer Vision and Pattern Recognition (CVPR), 5987–5995. IEEE.

Zhao, H.; Torralba, A.; Torresani, L.; and Yan, Z. 2019. HACS: Human action clips and segments dataset for recognition and temporal localization. In International Conference on Computer Vision (ICCV), 8668–8678. IEEE.

Zhou, Y.; Sun, X.; Zha, Z.-J.; and Zeng, W. 2018. MiCT: Mixed 3D/2D Convolutional Tube for Human Action Recognition. In Conference on Computer Vision and Pattern Recognition (CVPR), 449–458. IEEE.