Gate-Shift-Fuse for Video Action Recognition

Swathikiran Sudhakaran*, Sergio Escalera*, and Oswald Lanz*

Abstract—Convolutional Neural Networks are the de facto models for image recognition. However 3D CNNs, the straight forward extension of 2D CNNs for video recognition, have not achieved the same success on standard action recognition benchmarks. One of the main reasons for this reduced performance of 3D CNNs is the increased computational complexity requiring large scale annotated datasets to train them in scale. 3D kernel factorization approaches have been proposed to reduce the complexity of 3D CNNs. Existing kernel factorization approaches follow hand-designed and hard-wired techniques. In this paper we propose Gate-Shift-Fuse (GSF), a novel spatio-temporal feature extraction module which controls interactions in spatio-temporal decomposition and learns to adaptively route features through time and combine them in a data dependent manner. GSF leverages grouped spatial gating to decompose input tensor and channel weighting to fuse the decomposed tensors. GSF can be inserted into existing 2D CNNs to convert them into an efficient and high performing spatio-temporal feature extractor, with negligible parameter and compute overhead. We perform an extensive analysis of GSF using two popular 2D CNN families and achieve state-of-the-art or competitive performance on five standard action recognition benchmarks.

Index Terms—Action recognition, video classification, spatial gating, channel fusion.

I. INTRODUCTION

UNDERSTANDING human actions in videos is one of the challenging and fundamental problems in computer vision with a wide range of applications from video surveillance to robotics and human-computer interaction. Recently, video action recognition research has witnessed great progress owing to the adoption of Deep Neural Networks. However, such Deep Neural Models are yet to achieve the success of their image recognition counterparts. A key challenge lies in addressing the question of how to better encode the space-time features.

This work was partially supported in part by the Spanish projects under Grants PID2019-105093GB-I00, TED2021-131317B-I00, and PDC2022-133305I00, and start-up grant IN2814 of Free University of Bozen-Bolzano. This work was supported in part by Amazon AWS Machine Learning Research Awards (MLRA), ICREA through ICREA Academia Programme, in part by NVIDIA AI Technology Centre (NV AITC), EMEA, and in part by CINECA Award through ISCRA Initiative, for the availability of high Performance Computing Resources and Support. Recommended for acceptance by J. C. Niebles. (Corresponding author: Swathikiran Sudhakaran.)

Swathikiran Sudhakaran is with Samsung AI Center, CB1 2JH Cambridge, U.K. (e-mail: swathikiran@gmail.com).
Sergio Escalera is with the Computer Vision Center, Universitat de Barcelona, 08007 Barcelona, Spain (e-mail: sergio@maia.ub.es).
Oswald Lanz is with the Free University of Bozen-Bolzano, 39100 Bolzano, Italy (e-mail: oswald.lanz@unibz.it).

Manuscript received 28 October 2021; revised 7 April 2023; accepted 12 April 2023. Date of publication 19 April 2023; date of current version 4 August 2023. This article has supplementary downloadable material available at https://doi.org/10.1109/TPAMI.2023.3268134, provided by the authors.

Inspired from the improved performance of 2D Convolutional Neural Networks (CNNs) on image recognition tasks, [25] explores several approaches to extend 2D CNNs for action recognition by considering videos as a set of frames. This includes using a single frame from the video to stacking multiple frames and applying it to a 2D CNN. However it is found that both of these configurations perform similarly, indicating that the learned video representation is not capable of capturing motion information in the video. Two-Stream networks [16], [51] address this problem by stacking optical flow images to capture short-term motion features and incorporate them with the appearance features obtained from a single frame. Such approaches require costly optical flow computations and ignore valuable appearance features from the video. TSN [62] and ActionVLAD [18] propose to extract features from multiple frames instead, followed by a late temporal aggregation via mean pooling for improved appearance feature extraction. Such approaches still relied on optical flow images for capturing motion information. The works of [7], [36], [54] formulate late temporal aggregation as a sequence learning problem by encoding the frame-level features using variants of Recurrent Neural Networks (RNNs). All these approaches perform temporal modelling in a shallow manner.

Fine-grained recognition can benefit from deeper temporal modeling. Full-3D CNNs (C3D [21], [58]) process the video in space-time by expanding the kernels of a 2D ConvNet along the temporal dimension. Deep C3Ds are designed to learn powerful representations in the joint spatio-temporal feature space with more parameters (3D kernels) and computations (kernels slide over 3 densely sampled dimensions). In practice however, they may under-perform due to the lack of sufficiently large datasets for training them at scale. To cope with these issues arising from the curse of dimension one can narrow down network capacity by design. Fig. 1 shows several C3D kernel decomposition approaches proposed for spatio-temporal feature learning in video. A most intuitive approach is to factorize 3D spatio-temporal kernels into 2D spatial plus 1D temporal, resulting in a structural decomposition that disentangles spatial from temporal interactions (P3D [46], R(2+1)D [60], S3D [72]). An alternative design is separating channel interactions and spatio-temporal interactions via group convolution (CSN [59]), or modeling both spatial and spatio-temporal interactions in parallel with 2D and 3D convolution on separated channel groups (GST [38]). Temporal convolution can be constrained to hard-coded time-shifts that move some of the channels forward in time or backward (TSM [37]). These approaches learn structured kernels with a hard-wired connectivity and propagation pattern across the network. There is no data dependent decision taken at any point in the network.
We propose a novel spatio-temporal feature extraction published in CVPR [19].

We perform an extensive ablation analysis of the percentage points in accuracy is obtained on Something with their updated versions. We also present an extensive analysis to identify the best design choice for applying our light weight module to ResNet based backbone architectures. Third, we evaluate the performance of our module on two additional datasets: Kinetics 400 and Something-something-v2 and replace two datasets (EPIC-Kitchens-55 and Diving48) used in [55] with their updated versions. We also extend the related works section by reviewing more recent works on video action recognition and the experimental section by performing an extensive comparison of the performance of our work with more recently developed approaches.

The rest of the paper is organized as follows. Section II reviews recent works on action recognition. Section III presents the technical details of Gate-Shift-Fuse (GSF). Experimental results are reported in Sections IV and V concludes the paper.

II. RELATED WORK

The development of large-scale video datasets such as Kinetics [26], [52], Moments in Time [43], Something Something [19], EPIC-Kitchens-100 [5], etc., to mention a few, inspired video action recognition research significantly. Such datasets with varying properties, such as requirement of spatial reasoning [26], temporal reasoning [19], spatio-temporal reasoning [5], resulted in the development of a plethora of action recognition approaches. The development of recent large-scale datasets such as HowTo100M [42], HVU [6], Ego4D [20], etc., are expected to further video understanding research beyond simple tasks such as action recognition.

A. Action Recognition

Fusing Appearance and Flow: A popular extension of 2D CNNs to handle video is the Two-Stream architecture by Simonyan and Zisserman [51]. Their method consists of two separated CNNs (streams) that are trained to extract features from a sampled RGB video frame paired with the surrounding stack of optical flow images, followed by a late fusion of the prediction scores of both streams. The image stream encodes the appearance information while the optical flow stream encodes the motion information, that are often found to complement each other for action recognition. Several works followed this approach to find a suitable fusion of the streams at various depths [16] and to explore the use of residual connections [14] or multiplicative gating functions [15] between them. These approaches rely on optical flow images for motion information, and a single RGB frame for appearance information, which is limiting when reasoning about the temporal context is required for video understanding.
**Video as a Set or Sequence of Frames:** Later, other approaches were developed using multiple RGB frames for video classification. These approaches sparsely sample multiple frames from the video, which are applied to a 2D CNN followed by a late integration of frame-level features using average pooling [62], multilayer perceptrons [79], recurrent aggregation [7], [36], or attention [17], [54]. To boost performance, most of these approaches also combine video frame sequence with externally computed optical flow. This shows to be helpful, but computationally intensive.

**Modeling Short-Term Temporal Dependencies:** Other research has investigated the middle ground between late aggregation (of frame features) and early temporal processing (to get optical flow), by modeling short-term dependencies. This includes differencing of intermediate features [44] and combining Sobel filtering with feature differencing [57]. Other works [10], [45] develop a differentiable network that performs TV-L1 [74], a popular optical flow extraction technique. The work of [30] instead uses a set of fixed filters for extracting motion features, thereby greatly reducing the number of parameters. DMC-Nets [50] leverage motion vectors in the compressed video to synthesize discriminative motion cues for two-stream action recognition at a lower computational cost compared to raw flow extraction.

**Video as a Space-Time Volume:** Unconstrained modeling and learning of action features is possible when considering video in space-time. Since video can be seen as a temporally dense sampled sequence of images, expanding 2D convolution operation in 2D-CNNs to 3D convolution is a most intuitive approach to spatio-temporal feature learning [3], [21], [58]. The major drawback of 3D CNNs is the huge number of parameters involved. This results in increased computations and the requirement of large scale datasets for pre-training. Carreira and Zisserman [3] addressed this limitation by inflating video 3D kernels with the 2D weights of a CNN trained for image recognition. Several other approaches focused on reducing the number of parameters by disentangling the spatial and temporal feature extraction operations. P3D [46] proposes three different choices for separating the spatial and temporal convolutions and develops a 3D-ResNet architecture whose residual units are a sequence of such three modules. R(2+1)D [60] and S3D-G [72] also show that a 2D convolution followed by 1D convolution is enough to learn discriminative features for action recognition. CoST [31] performs 2D convolutions, with shared parameters, along the three orthogonal dimensions of a video sequence. CT-Net [32] proposes to tensorize the channel dimension of features as a multiplication of K sub-dimensions, followed by 3D separable convolutions along each of the K channel sub-dimensions. This allows the model to gradually increase the spatio-temporal receptive field with reduced computational complexity while maintaining interaction of features across both spatio-temporal and channel dimensions. MultiFiber [4] uses multiple lightweight networks, the fibers, and multiplexer modules that facilitate information flow using point-wise convolutions across the fibers. CIDC [33] performs channel independent directional convolution by applying 2D convolution independently on individual feature channels while considering the temporal dimension as channels. Other works involve Neural Architecture Search (NAS) for developing an optimum architecture [48], [49], [65]. X3D [12] develops a progressive algorithm for expanding the spatio-temporal resolution of inputs, network depth, number of channels, etc., of 3D CNN architectures. The final architecture developed with the search algorithm results in improved performance with less computational complexity compared to hand-designed 3D CNNs.

**Modeling Long-Term Dependencies:** Majority of existing approaches rely on the network depth to combine the information present in frames that are further apart in time. [67] develops a feature bank that stores features from the entire video which can then be integrated as contextual information to a video recognition model for understanding the present. Non-local [63] is a plug and play module that computes the features at a specific position in space-time as the weighted sum of features at all other positions, thereby capturing long-range dependencies. Timeception [22] introduces multi-scale separable temporal convolutions to reason about videos of longer duration. Eidetic 3D LSTM [66] integrates 3D convolution into RNNs and introduces a gated controlled self-attention module to enable the memory to interact with its past values. The former enables short-term temporal modeling while the latter allows the model to encode long-term temporal information. VideoGraph [23] constructs a unidirectional graph whose nodes represent the latent action concepts. A novel graph embedding layer is then developed to learn the relationship between these concepts. Zhang et al. [76] introduces 4D convolution to perform long-term modeling hierarchically. Inspired by the performance improvement obtained by Transformers in Natural Language Processing, Vision Transformer (ViT) [8] is developed for image recognition. TimeSformer [1] extends ViT by developing different variants of self-attention schemes for spatio-temporal feature learning. However, even the efficient variant of TimeSformer suffers from increased computational complexity and larger parameter count.

**Multi-Scale Architectures:** Inspired by the effectiveness of Two-stream architectures using RGB and optical flow images, [11], [13], [25] developed multi-stream architectures operating on two streams of RGB images. A multi-resolution CNN architecture is developed in [25] where low resolution frames are processed by one stream and high resolution center crops are processed by the second stream. [11] proposes to apply alternate frames of high resolution through a compact network in addition to a second stream processing low resolution images. This allows the network to encode features at multiple spatial scales and increase the number of processed frames while maintaining reasonable computational complexity. SlowFast [13] encodes frames using a dual pathway network. One pathway encodes frames sampled at a slower frame rate while the second encodes frames sampled at a faster frame rate, enabling the network to capture information at multiple temporal scales. These approaches rely on multiple networks to encode information at different scales. To address this issue, TPN [73] introduces a feature level pyramid network to encode features at multiple hierarchies thereby allowing the network to extract features at different temporal scales. Since the pyramid structure...
is constructed at feature level, TPN do not require multiple architectures, thereby reducing the number of parameters and computational complexity. MSTI-Net [68] proposes to split the feature tensors across the channel dimension into four groups and apply convolution operation with different kernel size and aggregate them hierarchically. Varying the kernel size of the convolution operation enables the network to encode spatio-temporal features at multiple scales.

B. Efficient Video Understanding

Recently, the focus of research is moving to the development of efficient (from a computational point of view) and effective (from a performance point of view) approaches for video understanding. Such approaches perform efficient modeling by (1) reducing redundant computations or (2) developing novel efficient architectures.

Reducing Computations by Selective Processing: [47] and [70] develop a conditional computing approach that decides whether to compute fine resolution features in a data dependent manner, to avoid additional compute overhead. AdaFrame [71] trains a policy gradient network to determine which frame to process next, thereby skipping processing of non-informative frames in a video. SCSampler [28] develops a light-weight model for identifying the most salient clip for recognizing the action present in a video. AdaFuse [41] learns a decision policy to reduce temporal redundancy by removing feature channels that are uninformative and reuse the feature channels from past frames for efficient inference. A reinforcement policy based learning approach for selecting the optimum input frame resolution is developed in [40], [2] follows the approach of knowledge distillation to train a student network that processes a smaller number of frames using a teacher network trained with a larger number of frames. All these approaches are based on existing spatio-temporal feature extractors or architectures that require large memory requirements due to the model size. Even though these approaches reduce the computations in inference time, majority of them suffer from heavy computations during the training stage.

Efficient Architectures: CNNs provide different levels of feature abstractions at different layers of the hierarchy. It has been found that the bottom layer features are less useful for extracting discriminative motion cues [56], [72], [82]. In [56] it is proposed to apply 1D convolution layers on top of a 2D CNN for video action recognition. The works of [82] and [72] show that it is more effective to apply full 3D and separable 3D convolutions at the top layers of a 2D CNN for extracting spatio-temporal features. These approaches resulted in performance improvement over full 3D architectures with less parameters and computations. Static features from individual frames represent scenes and objects and can also provide important cues in identifying the action. This is validated by the improved performance obtained with two-path structures that apply a parallel 2D convolution in addition to the 3D convolution [38], [80]. MiCT [80] is designed by adding 3D convolution branches in parallel to the 2D convolution branches of a BN-Inception-like CNN. GST [38] makes use of the idea of grouped convolutions for developing an efficient architecture for action recognition. They separate the features at a hierarchy across the channel dimension and separately perform 2D and 3D convolutions followed by a concatenation operation. In this way, the performance is increased while reducing the number of parameters. MVFNet [69] extends the idea of GST by replacing the 3D convolution with three 1D convolutions applied to each of the two spatial dimensions and temporal dimension. STM [24] proposes two parallel blocks for extracting motion features and spatio-temporal features. Their network rely only on 2D and 1D convolutions and feature differing for encoding motion and spatio-temporal features. TSM [37] proposes to shift the features across the channel dimension as a way to perform temporal interaction between the features from adjacent frames of a video. This parameter-less approach has resulted in similar performance to 3D CNNs. RubiksNet [9] replaces the convolution operation in 3D CNNs with learnable spatio-temporal shifting. TEA [34] develops a novel ResNet based architecture that weights channels with motion information while suppressing those without relevant information and that can encode long range temporal information in a hierarchical aggregation fashion. MSNet [29] develops a plug in module that can estimate the correspondences between adjacent frame level features and convert these into effective motion features that can be injected into the backbone CNN for efficient spatio-temporal modeling.

In all previous approaches, spatial, temporal, and channel-wise interaction is hard-wired. Here, we propose the Gate-Shift-Fuse (GSF), which control interactions in spatial-temporal decomposition and learns to adaptively route features through time and combine them in a data dependent manner, at almost no additional parameters and computational overhead.

III. Gate-Shift-Fuse

We present Gate-Shift-Fuse (GSF), a module capable of converting a 2D CNN into a high performing spatio-temporal feature extractor with minimal overhead.

A. Gate-Shift-Fuse Module

Several 3D kernel factorization approaches have been developed for efficient video recognition. Fig. 2 illustrates the network schematics of some of these approaches. S3D, or R(2+1)D, P3D, decompose 3D convolutions into 2D spatial plus 1D temporal convolutions. TSM replaces 1D temporal convolution with parameter-free and zero FLOP channel-wise temporal shift operations. GST uses group convolution where one group applies 2D spatial and the other 3D spatio-temporal convolution. GST furthermore applies point-wise convolution before and after the block to allow for interactions between spatial and spatio-temporal groups, and for channel reduction and up-sampling. In these modules, the feature flow is hard-wired by design, meaning that features are forwarded from one block to the next without data-dependent pooling, gating or routing decision.

GSF design, in Fig. 2, is inspired by GST and TSM but replaces the hard-wired channel split and concat aggregation with a learnable spatial gating block followed by a learnable fusion
block to fuse the two splits of input features. The function of gate block is to selectively route the features through time-shifts to encode temporal information. The fuse block then merge the shifted (temporal) features with the residual (spatial) features. The gate and fuse blocks are learnable and perform gating and fusion operations in a data dependent manner.

Based on the conceptual design in Fig. 2, we instantiate GSF as in Fig. 3. We first describe the equations governing the implementation of GSF. Let $X$ be the $C \times T \times W \times H$ shaped input tensor to GSF, where $C$ is the number of channels and $WH$, $T$ are the spatial and temporal dimensions, respectively. Let $X = [X_1, X_2]$ be the group=2 split of $X$ along the channel dimension, $W_g = [W_{g1}, W_{g2}]$ be two $1 \times C/2 \times 3 \times 3 \times 3$ shaped gating kernels and $W_f = [W_{f1}, W_{f2}]$ be two fusion kernels of shape $1 \times 2 \times 3 \times 3$. Then, the GSF output $Z = [Z_1, Z_2]$ is computed as

$$
Y_1 = \tanh(W_{g1} \ast X_1) \odot X_1
$$

$$
Y_2 = \tanh(W_{g2} \ast X_2) \odot X_2
$$

$$
R_1 = X_1 - Y_1
$$

$$
R_2 = X_2 - Y_2
$$

$$
Y_{1, shift} = shift_{fw}(Y_1)
$$

$$
Y_{2, shift} = shift_{bw}(Y_2)
$$

$$
F_1 = \sigma(W_{f_1} \ast [\phi(Y_{1,shift}; R_1)])
$$

$$
F_2 = \sigma(W_{f_2} \ast [\phi(Y_{2,shift}; R_2)])
$$

$$
Z_1 = F_1 \odot Y_{1,shift} + (1 - F_1) \odot R_1
$$

$$
Z_2 = F_2 \odot Y_{2,shift} + (1 - F_2) \odot R_2,
$$

where ‘$*$’ represents convolution, ‘$\odot$’ is Hadamard product, $shift_{fw}, shift_{bw}$ is forward, backward temporal shift, $\sigma$ is sigmoid function, $[\cdot, \cdot]$ is concatenation operation and $\phi$ is spatial average pooling.

The output features $X$ obtained from the spatial convolution, inherited from the 2D CNN base model, is applied to the gating module. The gating module splits the features into two groups across the channel dimension and generates two gating planes of shape $1 \times T \times W \times H$, each for the two groups. The gating planes are then applied as spatial weight maps to each of the two feature groups to generate a pair of gated features and their residuals. The gated features are then group-shifted forward and backward in time to inject temporal information. The group-shifted features and their corresponding residuals are then applied to the fusion module. The fusion module generates a channel weight map of shape $C \times T \times 1 \times 1$ for each of the two groups. The shifted features and their residuals are then fused via weighted averaging using the channel weight maps. This way, GSF selectively mixes spatial and temporal information through learnable spatial gating and fusion mechanisms.
Gating: Gating is implemented with a single spatio-temporal 3D kernel and tanh activation. With a 3D kernel we utilize short-range spatio-temporal information in the gating. \( \tanh \) provides spatial gating planes with values in the range \((-1, +1)\) and is motivated as follows. When the gating value at a feature location is 0 and that of the time-shifted feature was +1, then a weighted temporal feature averaging is performed at that location. If the gating value of the time-shifted feature was -1 instead, then a weighted temporal feature differencing is performed. Using \( \tanh \), the gating can thus learn to apply either of the two modes, location-wise.

Fusion: The simplest choice for fusing the time-shifted temporal features with the residual is summation, as done in our preliminary work Gate-Shift Module (GSM) [55]. However, since different channels of a feature tensor encode different level of information, fusion by summation may be sub-optimal. Instead GSF performs weighted averaging on the two feature groups. For this, the group-shifted features and their corresponding residuals are concatenated followed by spatial average pooling operation, resulting in a tensor of shape \( 2 \times C \times T \). Then a 2D convolution is applied on this tensor followed by \( \text{sigmoid} \) non-linearity to obtain a weight map with shape \( 1 \times C \times T \). This weight map is then reshaped to \( C \times T \times 1 \times 1 \) and is used to fuse the time-shifted features and their residuals via weighted averaging. When the fusion weight at a feature channel is 0, only the residual features with the spatial information is retained for further processing. On the other hand, if the channel weight is 1, the network will discard the residual spatial features and propagate the time-shifted temporal features. With other weight values, the model mixes the two feature groups selectively by mixing the channels.

Parameter Count, Complexity: The parameter count in the gating is \( 2 \times (27 \cdot C/2) = 27 \cdot C \) and that of the fusion module is \( 2 \times (2 \cdot 3 \cdot 3) = 36 \). The total parameter overhead introduced by GSF is \( 27 \cdot C + 36 \approx 27 \cdot C \) since \( C \gg 36 \) in practice. On the other hand, a typical C3D block such as the \( 3 \times 1 \times 1 \) used in S3D [72] (Fig. 2) has a parameter count of \( C_{out} \cdot C \cdot 3 \) where \( C_{out} \geq C \). Thus the parameter overhead introduced by GSF is far less than that of a C3D block since \( 27 \ll 3 \cdot C_{out} \).

The complexity of gating module is \( H \cdot W \cdot T \cdot 27 \cdot C \) while that of the fusion module is \( C \cdot T \cdot 36 \). The total complexity of GSF is \( H \cdot W \cdot T \cdot 27 \cdot C + C \cdot T \cdot 18 \approx H \cdot W \cdot T \cdot 27 \cdot C \). For a C3D block performing 1D temporal convolution, the complexity is \( H \cdot W \cdot T \cdot 3 \cdot C \cdot C_{out} \cdot 3 \). Since \( 27 \ll 3 \cdot C_{out} \), one can see that the complexity introduced by GSF is far less than that of a 1D temporal convolution layer. Thus GSF is far more efficient in terms of parameters and computations compared to the temporal convolution used in efficient implementations such as 3D or R(2+1)D.

B. Gate-Shift-Fuse Networks

GSF can be plugged into any 2D CNN architecture to inject temporal information into the features extracted by the backbone CNN. In this work, we choose two widely adopted 2D CNN families, Inception and ResNet, as the backbone CNNs. As mentioned in Section III-A, GSF can be inserted into any layer in the backbone CNN after a 2D convolution. For the Inception family, GSF is applied to the least parametrized branch, i.e., the branch with the least number of convolutions (cf. Fig. 4). For ResNet based CNN, GSF is applied after the second convolution layer in the bottle neck layer of the backbone (cf. Fig. 5). Section IV-C presents the experiments conducted to determine the optimum configuration for applying GSF to the two CNN families considered in this work.

We train the backbone CNN with GSF using the TSN framework. A fixed number of frames are uniformly sampled from the video and applied to the network. The network outputs a class prediction for each of the frames. The frame level predictions are then averaged to obtain the video level prediction. Training is done using cross-entropy loss computed from the video level prediction and the ground truth video label.

IV. EXPERIMENTS AND RESULTS

This section presents an extensive set of experiments to evaluate GSF.

A. Datasets

We evaluate Gate-Shift-Fuse (GSF) on five standard action recognition benchmarks, Something Something [19] (SS-V1 and SS-V2), Kinetics 400 [26], Diving48 [35] and EPIC-Kitchens-100 [5]. Something-V1 and Something-V2 consists of fine-grained object manipulation actions comprising of 174 categories. Something-V1 consists of \( 100 \) K videos while Something-V2 is an updated version of Something-V1 with \( 220 \) K videos. Kinetics 400 is a large scale dataset consisting of 400 action categories with at least 400 clips per action class. Diving48 dataset contains around 18 K videos with 48 fine-grained diving categories. EPIC-Kitchens-100 is the largest egocentric action recognition dataset with \( 90 \) K video clips. Each clip in the dataset is labelled with a verb and noun class from a
Fig. 6. t-SNE projection of output features obtained from the layer preceding the final linear layer for (a) TSN baseline and (b) TSN baseline with GSF. Samples from the 10 action groups defined in [19] are visualized.

Fig. 7. Action classes with the highest improvement over TSN baseline. X-axis shows the number of corrected samples for each class. Y-axis labels are in the format true label (GSF)/predicted label (TSN).

possible set of 97 verb classes and 300 noun classes. The verb and noun category pairs are combined to form an action class. All the datasets considered for evaluating GSF differ considerably in the type of actions present in the videos. Something datasets consist of actions that are object agnostic. Thus understanding the actions carried out in the videos require strong temporal reasoning. On the other hand, the videos in EPIC-Kitchens-100 consists of actions that are object specific and requires encoding of the spatio-temporal patterns in the videos for recognizing an action. Both Kinetics 400 and Diving48 datasets contain human actions captured from a third person perspective and require understanding of the temporal dynamics of the human body in the video. While the background of the videos in Kinetics 400 can reveal useful information regarding the action category, the videos in Diving48 are captured with a uniform background thereby requiring encoding of the temporal progression of the actions.

B. Implementation Details

Training: We train the models end-to-end using Stochastic Gradient Descent (SGD) with momentum 0.9, weight decay $10^{-4}$ and an initial learning rate of 0.01, for 60 epochs. We use a cosine learning rate schedule with warmup (10 epochs) for adjusting the learning rate. The batch size used is 32 and a dropout of 0.5 is applied to avoid overfitting. 8 or 16 frames sampled from each video is used for predicting the action present in the video. Random scaling, cropping and flipping
Fig. 8. Histogram of the gating weights generated by the gating module on videos from SS-v1 dataset.
Fig. 9. Saliency tubes generated by TSN (left) and GSF (right) on sample videos taken from the validation set of Something Something-V1 dataset. Action labels for samples (a)-(d) are shown as text on columns.
along with temporal jittering as presented in [62] are used as data augmentations during training. The backbone CNNs are pretrained on ImageNet. Unless otherwise specified, we use the same setting for training GSFs on all the datasets used in this work.

Testing: Similar to contemporary works, we follow two approaches for evaluation: efficiency protocol and accuracy protocol. In the efficiency protocol, a single clip consisting of 8 or 16 frames are uniformly sampled from the video with each frame obtained by cropping the center portion of the video frame. In the accuracy protocol, we sample two clips for all the dataset except for Kinetics 400 from each video and extract three different crops from each frame. Thus a total of 6 clips are sampled from the input video. For Kinetics 400, we sample 10 clips and 3 crops from each video. The predictions of the individual clips are averaged to obtain the final prediction for the video.

Source code of our implementation can be found online at https://github.com/swathikirans/GSF.

C. Model Development

The model development is performed using SS-V1 dataset. We use 8 frames sampled from the video as input to the models and use the efficiency protocol for evaluating the testing performance.

Inception Backbones: We use BNInception and InceptionV3 as the two CNN backbones from the Inception family of models. Since the architectural designs are similar for the two models, we choose BNInception for identifying the optimum model configuration.

Inception models have a multi-branch architecture as shown in Fig. 4. We first perform analysis to identify which branch is more suitable for GSF to be plugged into. The results of this analysis are provided in Table I. From the table, one can see that GSF performs the best when GSF is plugged into the branch with the least number of convolution layers.

As mentioned in Section III-A, temporal information is injected into the features by channel shifting. One can control the amount of temporal information by choosing the number of channels shifted. Shifting all the channels will allow the model to encode more temporal information. However, this is achieved at the expense of spatial information present in the features. There should be an optimum balance between the two. In Table II we report the performance obtained by varying the number of channels shifted in each layers of the BNInception backbone. It can be seen from the table that shifting all the channels results in the best performance.

We use the configuration found from these two experiments for constructing GSFs with BNInception and InceptionV3 backbones.

ResNet Backbones: The block diagram of the bottleneck layer of ResNet based CNNs are shown in Fig. 5. Table III reports the performance of different configurations obtained by applying GSF after different convolution layers present in the block of ResNet50. Following the results from Table III, we apply GSF to after the second convolution of the ResNet block as shown in Fig. 5.

Similar to Inception architectures, we also conduct analysis on ResNet by shifting different number of channels. The results are reported in Table IV. Unlike with the case of Inception models,
the best performance is obtained when only 25% of the channels are shifted. This can be attributed to the fact that shifting all the channels will affect the spatial modelling capability of the network. In the case of Inception based models, the branches without GSF will encode the spatial information. Hence, GSF can be applied to all the channels in the corresponding branch to inject temporal information to the features.

### D. Ablation Analysis

In this section, we report the ablation analysis performed on the validation set of Something-V1 dataset. In all the experiments, we apply 8 frames as input to the network and report accuracy using the efficiency protocol.

We first ablate the various components of GSF and validate the design choices. We choose BNInception as the backbone model due to its reduced complexity. The results are reported in Table V. By removing the channel fusion, GSF converges to GSM [55], our prior work on top of which GSF is built upon, and underperforms by 1.2%. This validates the importance of the learned channel fusion. We then fix the spatial gating to a fixed value of 1 which results in an accuracy of 45.57%. This is similar to Temporal Shift Module (TSM) [37] where temporal shifting is applied to all the channels. With a fixed gating of -1, the networks performs differencing operation between the adjacent frames and results in 45.89%. This is a special case of Temporal Difference Network (TDN) [44] without the image subnetwork. Finally, we fix the gating to 0 which converges the model to the baseline, TSN, without any temporal interaction between the frame level features, and drops the accuracy by 31%.

Next we compare the performance improvement obtained by adding different number of GSF to the backbone CNN. Table VI shows the results of this experiment conducted on BNInception backbone. Baseline is the standard Temporal Segment Network (TSN) architecture, with an accuracy of 17.25%. We then applied GSF at the last Inception block of the CNN. This improved the recognition performance by 14.65%. By adding GSF to the last 5 layers of the backbone, the performance improves by 27.99%.

Thus, a consistent improvement in performance can be observed by plugging in GSF to the backbone layers of the CNN. With GSF added to all the Inception blocks in the CNN, an absolute improvement of 31% is obtained, over the baseline, to reach an accuracy of 48.44%. This huge improvement is obtained only with an overhead of 0.48% and 0.55% in terms of parameters and complexity, respectively.

We also conducted the above experiment on ResNet-50 backbone. The GSF follows the setting mentioned in Section IV-C and the results are reported in Table VII. By plugging in GSF to the last four bottleneck layers of the backbone, a significant improvement of 28% in recognition accuracy is obtained. This is achieved only with a very small overhead in number of parameters and computations. As observed in the case of BNInception backbone, the recognition performance increased consistently by plugging in GSF to all the bottleneck layers of the backbone CNN.

### E. State-of-the-Art Comparison

**Something-Something**: Table VIII compares the recognition performance of GSF with state-of-the-art approaches on SS-V1 and SS-V2 datasets. For fair comparison, we compare only with those methods that use RGB frames for feature extraction. We also report the number of frames used by each approach along with the number of parameters and the computational complexity in terms of FLOPs in the table. Since different approaches use different backbone CNNs, we also report the backbones used for fair comparison. The table is split into two, the first part lists approaches that use 2D CNN or efficient implementations of 3D CNN as backbone. The second part shows the methods that rely on heavier 3D CNNs for spatio-temporal feature extraction. Our baseline TSN [62] obtains 17.52% and 32.71% recognition accuracy on SS-V1 and SS-V2 datasets, respectively. Applying GSF to the backbone CNN, an absolute gain of +33.19% and +30.69% is obtained on the two datasets. This huge improvement is obtained with an increase of only +0.48% and +0.55% on the number of parameters and FLOPs. Compared to the top performing approaches that use 2D CNN, such as TEA [34] and MSNet [29], GSF performs competitively. On the other hand, approaches that uses 3D CNN such as I3D [3] and Non-local networks [63], considerably falls behind GSF. The approach from [39] uses a deeper 3D CNN (I3D ResNet-152). When compared to the performance obtained with a backbone CNN of equivalent depth (ResNet-50), GSF outperforms the approach from [39] by +2.2%. By plugging in the MotionSqueeze module [29] on the GSF version of ResNet-50,

| Model                        | Accuracy (%) | Params. | FLOPs |
|------------------------------|--------------|---------|-------|
| BNInception (baseline)       | 17.25        | 10.45M  | 16.43G|
| BNInception + 1 GSF          | 31.19        | 10.46M  | 16.44G|
| BNInception + 5 GSF          | 45.24        | 10.48M  | 16.46G|
| BNInception + 10 GSF         | 48.44        | 10.5M   | 16.55G|

| Table VI: Recognition Accuracy by Varying the Number of GSF Added to BNInception Backbone |
|---------------------------------------------|
| Gating | Fusion | Model | Accuracy (%) |
|--------|--------|-------|---------------|
| Learnable | Learnable | GSF | 48.44       |
| Learnable | Sum | GSM | 47.24       |
| 1      | Sum | TSM | 45.57       |
| -1     | Sum | TDN | 45.89       |
| 0      | Sum | TSN | 17.25       |

| Table VII: Recognition Accuracy by Varying the Number of GSF Added to ResNet50 Backbone |
|---------------------------------------------|
| Model                        | Accuracy (%) | Params. | FLOPs |
|------------------------------|--------------|---------|-------|
| ResNet-50 (baseline)         | 17.22        | 23.86M  | 33.06G|
| ResNet-50 + 4 GSF            | 45.97        | 23.92M  | 33.09G|
| ResNet-50 + 8 GSF            | 48.54        | 23.95M  | 33.14G|
| ResNet-50 + 16 GSF           | 50.17        | 23.97M  | 33.4G  |
our model surpasses the recently developed state-of-the-art approaches MVF-Net [69] and CT-Net [32], with same backbone, achieving an accuracy of 54.04% on SS-V1.

**Kinetics400:** We compare the performance of GSF with state-of-the-art methods on Table IX. All two GSF models are trained with 16 frames and the accuracy protocol with 10 clip sampling during inference. As done previously, we split the table into two sections: top section comparing methods that use 2D CNN backbones and bottom section lists the approaches that use 3D CNN backbone. As mentioned in Section IV-A, Kinetics400 consists of videos of short duration (10 s) that do not require strong temporal reasoning. This is reflected in Table IX, where the approaches that use 3D CNN for feature extraction performs significantly better than those that use 2D CNNs. Compared to approaches that use 2D CNN, the two GSF models perform competitively with the InceptionV3 based model outperforming its ResNet-50 counterpart by 0.7%.

**EPIC-Kitchens-100:** Comparison of GSF with state-of-the-art approaches on the validation set of EPIC-Kitchens-100 is performed on Table X. For this dataset, we apply autoaugment [77].

| Method      | Backbone     | #Frames | Params (M) | GFLOPs | Accuracy (%) |
|-------------|--------------|---------|------------|--------|--------------|
| TSN [62]    | BN-Inception | $10 \times 10$ | $32.87 \times 1 \times 1$ | 17.44 | 32.71 |
| MultiScale TRN [79] | BN-Inception | 8 x NA | NA | 34.44 | 48.8 |
| MNet [30]   | ResNet-101   | 10 x 10 | NA | 43.92 | - |
| S3D-G [72]  | InceptionV1  | 64 x 1 x 1 | 11.6 | 71.38 x 1 x 1 | 48.2 | - |
| TrajectoryNet [78] | ResNet-18 | 16 x NA | NA | 47.8 | - |
| ABM [81]    | ResNet-50    | 16 x 3 x 1 | NA | 46.81 | 61.25 |
| TSM [37]    | ResNet-50    | 16 x 1 x 1 | 24.3 | 65 x 1 x 1 | 47.3 | 61.2 |
| RubikNet [6] | ResNet-50    | 8 x 1 x 1 | 8.9 | 15.8 x 1 x 1 | 46.4 | 59.0 |
| STM [34]    | ResNet-50    | 16 x 3 x 10 | 24 | 66.5 x 3 x 10 | 50.7 | 64.2 |
| blVNet-TAM [11] | blResNet-50 | 16 x 1 x 2 | 25 | 47 x 1 x 2 | 48.4 | 61.7 |
| blVNet-TAM [11] | blResNet-101 | 32 x 2 x 2 | 40.2 | 128.6 x 1 x 2 | 53.1 | 65.2 |
| CorrNet [61] | ResNet-101   | 32 x 1 x 10 | NA | 224 x 1 x 10 | 51.1 | - |
| TPN [73]    | ResNet-50    | 8 x 10 x 1 | NA | NA | 49.0 | 62 |
| TAN [34]    | ResNet-50    | 8 x 3 x 10 | NA | 35 x 3 x 10 | 51.7 | NA |
| MSNet [29]  | ResNet-50    | 8 x 1 x 1 | 24.6 | 34 x 1 x 1 | 50.9 | 65 |
| MSNet [29]  | ResNet-50    | 16 x 1 x 1 | 24.6 | 67 x 1 x 1 | 52.1 | 64.7 |
| MRFN [69]   | ResNet-50    | 16 x 3 x 2 | - | 68 x 3 x 2 | 52.6 | 65.2 |
| CT-Net [32] | ResNet-50    | 16 x 3 x 2 | - | 75 x 3 x 2 | 53.4 | 65.9 |
| BD [3] from [64] | ResNet-50 | 32 x 1 x 2 | 28 | 153 x 1 x 2 | 41.6 | - |
| Non-local from [64] | ResNet-50 | 32 x 1 x 2 | 35.3 | 168 x 1 x 2 | 44.4 | - |
| GCN + Non-local [64] | ResNet-50 | 32 x 1 x 2 | 62.2 | 303 x 1 x 2 | 46.1 | - |
| GST [38]    | ResNet-50    | 16 x 1 x 1 | 21 | 59 x 1 x 1 | 48.6 | 62.6 |
| Martinez et al. [39] | ResNet-50 | NA | NA | 50.1 | NA |
| Martinez et al. [39] | ResNet-152 | NA | NA | 53.4 | NA |
| MoViNet-A3 [27] | NA | NA | - | 50.1 | NA |

| Method | Backbone | Accuracy (%) |
|--------|----------|--------------|
| bLVNet-TAM [11] | blResNet-50 | 75.5 |
| STM [24] | ResNet-50 | 73.7 |
| TSM [37] | ResNet-50 | 74.1 |
| Martinez et al. [39] | NA | 74.3 |
| TEA [34] | ResNet-50 | 76.1 |
| MNet [30] | ResNet-50 | 76.4 |
| MVFNNet [69] | ResNet-50 | 77.0 |
| CT-Net [32] | ResNet-50 | 77.3 |
| BD [30] | ResNet-101 + NL | 77.7 |
| Martinez et al. [39] | NA | 78.8 |
| ip-CSN [30] | ResNet-152 | 79.2 |
| TPN [30] | ResNet-101 | 78.9 |
| SlowFast [30] | ResNet-101 + NL | 79.8 |
| X3D-XXL [30] | - | 80.4 |
| MoViNet-A6 [27] | - | 81.5 |

| Method | Backbone | Accuracy (%) |
|--------|----------|--------------|
| GSF | ResNet-50 | 74.81 |
| GSF | InceptionV3 | 75.57 |
in addition to random cropping and random flipping as augmentations. All models are trained with 16 frames as a multitask problem for predicting verb, noun and action classes. Accuracy protocol is used during inference. InceptionV3 and ResNet-50 models are initialized with Kinetics400 pretrained weights while ResNet-101 backbone is initialized with ImageNet pretrained weights. From the table one can see that GSF models outperform the other approaches significantly, except for MoViNet [27] which is a NAS based approach using 3D convolutions. One interesting finding is that in contrast to what was observed for Something and Kinetics datasets, the ResNet based backbones result in better performance than InceptionV3 based model.

**Diving48:** We compare the performance of Diving48 dataset on Table XI. We use 16 frames to train the GSF model with InceptionV3 backbone and inference is performed following the accuracy protocol. All training settings are the same as mentioned in Section IV-B except for the batch size. For Diving48, we used a smaller batch size of 8 during training. From the table, one can see that GSF model outperforms approaches that uses 3D CNN backbones such as I3D [3], SlowFast [13] and GST [38] by a significant margin.

**F. Discussion**

We perform further analysis of GSF in this section. The t-SNE plot of features obtained from the output of the layer preceding the final Fully Connected (FC) layer of BNInception backbone is plotted in Fig. 6(a). We visualize the 10 action groups defined in [19] in the plot. From the figure, one can see better separation of the features when GSF is plugged into the backbone CNN. This allows the model to correctly recognize the action present in the video and results in improved recognition performance compared to the baseline TSN.

We further analyse the gating weights generated by GSF on BNInception backbone CNN across its various layers. We plot a histogram of the weights averaged across all the videos on Fig. 8. From the figure, one can see that the gating weights are negative for the bottom layers while it becomes positive at the top layers. The negative weights result in feature differencing while the positive weights result in feature averaging, as explained in Section III-A of the manuscript. Image difference captures the salient motion occurring between the image pair. Since image features obtained from a CNN are invariant to translation and small changes to appearance, image feature difference allows in encoding of strong temporal information. Once the temporal information is encoded via feature differencing in the bottom layers, the top layers perform feature aggregation via feature averaging. Instead of simply shifting the features as done in TSM, the gating module allows this flexibility of performing feature differencing and averaging to the model, resulting in an improved recognition performance.

The action categories that benefited the most by plugging in GSF to the backbone CNN are reported in Fig. 7. The Y-axis labels show the true labels (predicted by GSF) and the labels predicted by TSN, separated by “/”. From the plot, it can be seen that the baseline TSN confuses actions that are of similar nature but following a different (reversed) order, such as unfolding something—folding something, moving something and something closer to each other and moving something and something away from each other, etc. With GSF modules in the backbone CNN, the model is capable of encoding the temporal ordering of frames to correctly recognize the action. This can further be seen from the visualization of the saliency tubes [53] for the samples from the most improved classes as shown in Fig. 9. From the figure, one can see that GSF predicts the action category by focusing on the relevant frames where the corresponding action is taking place as opposed to the baseline TSN. In Fig. 9(c) and (d), even though the baseline model is focusing on the relevant spatio-temporal features, the lack of temporal reasoning renders the model to make wrong predictions.

We analyse the computational complexity and memory requirement for GSFs and state-of-the-art models in Fig. 10. X-axis shows the computational complexity in GFLOPs and Y-axis shows the recognition accuracy. The size of the marker indicates the number of parameters present in each model. It can be seen from the plot that GSF outperforms existing approaches in terms of recognition accuracy while maintaining a reasonable compute and memory requirements. Table XII compares the inference latency of GSF against TSM and GST. We report the time taken for a forward pass with a single input video consisting of 8 frames. We measure the latency on a single NVIDIA Titan V GPU. Compared to the other two architectures, GSF is slightly slow with a latency of 12.4 ms compared to 7.59 ms of TSM, 8.28 ms of GST and 10.46 ms of RubiksNet. However, this latency of 12.4 ms is equivalent to ≈80 videos per second, which is a reasonable speed for real time applications. On the other hand, the improvement in accuracy obtained by GSF on SS-V1 is large compared to the other three architectures (+4.57% over

---

**TABLE X**

Comparison to State-of-the-Art on the Validation Set of EPIC-Kitchens-100. * Results reported from [5]

| Method      | Backbone  | Accuracy (%) |
|-------------|-----------|--------------|
| TSN [62]    | ResNet-50 | 60.18        |
| TSN         | ResNet-50 | 65.88        |
| TSN         | ResNet-50 | 67.66        |
| TSN         | ResNet-50 | 67.66        |
| SlowFast [13] | ResNet-50 | 65.56        |
| MoViNet-A6 [27] | ResNet-50 | 72.2          |
| GSF         | InceptionV3 | 68.35        |
| GSF         | ResNet-50  | 88.76        |
| GSF         | ResNet-101 | 90.97        |

---

**TABLE XI**

Comparison to State-of-the-Art on Diving48

| Method       | Accuracy (%) |
|--------------|--------------|
| I3D [3] from [75] | 48.5        |
| S3D [72] from [75] | 50.6        |
| SlowFast [13] from [1] | 77.6        |
| GST [38] from [75] | 78.9        |
| TimeSformer-L [1] | 81           |
| TQN [75]     | 81.8         |
| GSF          | 90.51        |
TSM, +3.17% over GST and +3.77% over RubiksNet). Note that all models except RubiksNet are run with half precision. We report additional runtime analyses in the supplementary document, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2023.3268134.

V. CONCLUSION

We presented Gate-Shift-Fuse (GSF), a novel spatio-temporal feature extraction module capable of converting a 2D CNN into a high performing spatio-temporal feature extractor with negligible computational overhead. GSF relies on spatial gating and channel fusion to exchange information between neighbouring frames in a data dependent manner. We performed an extensive analysis to study the effectiveness of GSF using two CNN model families on five standard action recognition benchmarks. Compared to other approaches, GSF achieves state-of-the-art or competitive performance on these datasets with far less model complexity, indicating its effectiveness as an efficient spatio-temporal feature extractor.

REFERENCES

[1] G. Bertasius, H. Wang, and L. Torresani, “Is space-time attention all you need for video understanding?” in Proc. Int. Conf. Mach. Learn., 2021, pp. 813–824.
[2] N. Bhardwaj, M. Srinivasan, and M. M. Khapra, “Efficient video classification using fewer frames,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 354–363.
[3] J. Carreira and A. Zisserman, “Quo vadis, action recognition? A new model and the kinetics dataset,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 6299–6308.
[4] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng, “Multi-fiber networks for video recognition,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 352–367.
[5] D. Damen et al., “Rescuing egocentric vision,” 2020, arXiv:2006.13256.
[6] A. Diba et al., “Large scale holistic video understanding,” in Proc. Eur. Conf. Comput. Vis., Springer, 2020, pp. 593–610.
[7] J. Donahue et al., “Long-term recurrent convolutional networks for visual recognition and description,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2015, pp. 2625–2634.
[8] A. Dosovitskiy et al., “An image is worth 16x16 words: Transformers for image recognition at scale,” in Proc. Int. Conf. Learn. Representations, 2021.
[9] L. Fan et al., “RubiksNet: Learnable 3D-shift for efficient video action recognition,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 505–521.
[10] L. Fan, W. Huang, C. Gan, S. Ermon, B. Gong, and J. Huang, “End-to-end learning of motion representation for video understanding,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 6016–6025.
[11] Q. Fan, C.-F. R. Chen, H. Kuehne, M. Pistoia, and D. Cox, “More is less: Learning efficient video representations by big-little network and depthwise temporal aggregation,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 2264–2273.
[12] C. Feichtenhofer, “X3D: Expanding architectures for efficient video recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 203–213.
[13] C. Feichtenhofer, H. Fan, J. Malik, and K. He, “SlowFast networks for video recognition,” in Proc. Int. Conf. Comput. Vis., 2019, pp. 6201–6210.
[14] C. Feichtenhofer, A. Pinz, and R. Wildes, “Spatiotemporal residual networks for video action recognition,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 3468–3476.
[15] C. Feichtenhofer, A. Pinz, and R. P. Wildes, “Spatiotemporal multiplier networks for video action recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 4768–4777.
[16] C. Feichtenhofer, A. Pinz, and A. Zisserman, “Convolutional two-stream network fusion for video action recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 1933–1941.
[17] R. Girshar and D. Ramanan, “Attentional pooling for action recognition,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 34–45.
[18] R. Girshar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell, “ActionVLAD: Learning spatio-temporal aggregation for action classification,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 971–980.
[19] R. Goyal et al., “The ‘something something’ video database for learning and evaluating visual common sense,” in Proc. Int. Conf. Comput. Vis., 2017, pp. 5842–5850.
[20] K. Granman et al., “Ego4D: Around the world in 3,000 hours of egocentric video,” 2021, arXiv:2110.07058.
[21] K. Hara, H. Katoaka, and Y. Satoh, “Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and ImageNet?,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 6546–6555.
[22] N. Hussein, E. Ravves, and A. Smeulders, “Timecomplex for complex action recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 254–263.
[23] N. Hussein, E. Ravves, and A. Smeulders, “VideoGraph: Recognizing minutes-long human activities in videos,” 2019, arXiv:1905.05143.
[24] B. Jiang, M. Wang, W. Gan, W. Wu, and J. Yan, “STM: SpatioTemporal and motion encoding for action recognition,” in Proc. Int. Conf. Comput. Vis., 2019, pp. 2000–2009.
[25] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, “Large-scale video classification with convolutional neural networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2014, pp. 1725–1732.
[26] W. Kay et al., “The kinetics human action video dataset,” 2017, arXiv:1705.06950.
[27] D. Kondratyuk et al., “MoViNets: Mobile video networks for efficient video action recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2021, pp. 16020–16030.
[28] B. Korbar, D. Tran, and L. Torresani, “SCSampler: Sampling salient clips from video for efficient action recognition,” in Proc. Int. Conf. Comput. Vis., 2019, pp. 6232–6242.
[29] H. Kwon, M. Kim, S. Kwak, and M. Cho, “MotionSqueeze: Neural motion feature learning for video understanding,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 345–362.
[30] M. Lee, S. Lee, S. Son, G. Park, and N. Kwak, “Motion feature network: Fixed motion filter for action recognition,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 387–403.
[31] C. Li, Q. Zhong, D. Xie, and S. Pu, “Collaborative spatiotemporal feature learning for video action recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 7872–7881.
[32] K. Li, X. Li, Y. Wang, J. Wang, and Y. Qiao, “CT-net: Channel tensorization network for video classification,” in Proc. Int. Conf. Learn. Representations, 2021.
[33] X. Li, B. Shuai, and J. Tighe, “Directional temporal modeling for action recognition,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 275–291.
[34] Y. Li, B. Ji, X. Shi, J. Zhang, B. Kang, and L. Wang, “TEA: Temporal excitation and aggregation for action recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 909–918.
[35] Y. Li, Y. Li, and N. Vasconcelos, “RESOUND: Towards action recognition without representation bias,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 520–535.
[36] Z. Li, K. Gavriluyk, E. Ravves, M. Jain, and C. G. Snoek, “VideoLSTM convolves, attends and flows for action recognition,” Comput. Vis. Image Understanding, vol. 166, pp. 41–50, 2018.
[37] J. Lin, C. Gan, and S. Han, “Temporal shift module for efficient video understanding,” in Proc. Int. Conf. Comput. Vis., 2019, pp. 7082–7092.
[38] C. Luo and A. Yuille, “Grouped spatial-temporal aggregation for efficient action recognition,” in Proc. Int. Conf. Comput. Vis., 2019, pp. 5512–5521.

TABLE XII

| Model     | Latency (ms) | Accuracy on SS-VI (%) |
|-----------|--------------|-----------------------|
| TSM       | 7.59         | 45.6                  |
| GST       | 8.28         | 47.0                  |
| RubiksNet | 10.46        | 46.4                  |
| GSF       | 12.40        | 50.17                 |

COMPARISON OF LATENCY INCURRED BY GSF WITH RESPECT TO OTHER EFFICIENT APPROACHES. WE REPORT THE TIME TAKEN FOR ONE FORWARD PASS OF A VIDEO CLIP CONSISTING OF 8 FRAMES. "∗" ALL MODELS EXCEPT RUBIKSNET ARE RUN WITH HALF PRECISION.
[39] B. Martinez, D. Modolo, Y. Xiong, and J. Tighe, “Action recognition with spatial-temporal discriminative filter banks,” in Proc. Int. Conf. Comput. Vis., 2019, pp. 5481–5490.

[40] Y. Meng et al., “AR-net: Adaptive frame resolution for efficient action recognition,” in Proc. Eur. Conf. Comput. Vis., Springer, 2020, pp. 86–104.

[41] Y. Meng et al., “AdaFuse: Adaptive temporal fusion network for efficient action recognition,” in Proc. Int. Conf. Learn. Representations, 2021.

[42] A. Miech, D. Zhukov, J.-B. Alayrac, M. Tapaswi, I. Laptev, and J. Sivic, “HowTo100M: Learning a text-video embedding by watching hundred million narrated video clips,” in Proc. Int. Conf. Comput. Vis., 2019, pp. 2630–2640.

[43] M. Montfort et al., “Moments in time: One million videos for event understanding,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 42, no. 2, pp. 502–508, Feb. 2020.

[44] J. Ng and L. Davis, “Temporal difference networks for video action recognition,” in Proc. Winter IEEE Appl. Conf. Comput. Vis., 2018, pp. 1587–1596.

[45] A. J. Piergiovanni and M. S. Ryoo, “Representation flow for action recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern. Recognit., 2019, pp. 9945–9953.

[46] Z. Qiu, T. Yao, and T. Mei, “Learning spatio-temporal representation with pseudo-3D residual networks,” in Proc. Int. Conf. Comput. Vis., 2017, pp. 53–64.

[47] M. S. Ryoo, A. Piergiovanni, M. Tan, and A. Angelova, “AssembleNet++: Assembling modality representations via attention connections,” in Proc. Int. Conf. Comput. Vis., Springer, 2020, pp. 654–671.

[48] N. Quader, J. Lu, P. Dai, and W. Li, “Towards efficient coarse-to-fine networks for action and gesture recognition,” in Proc. Eur. Conf. Comput. Vis., Springer, 2020, pp. 35–51.

[49] M. S. Ryoo, A. Piergiovanni, J. Kangasputna, and A. Angelova, “AssembleNet: Searching for multi-stream neural connectivity in video architectures,” in Proc. Int. Conf. Learn. Representations, 2020.

[50] Z. Shou et al., “DMC-Net: Generating discriminative motion cues for fast compressed video action recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern. Recognit., 2019, pp. 1268–1277.

[51] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” in Proc. Adv. Neural Inf. Process. Syst., 2014, pp. 568–576.

[52] L. Smaier, J. Carreira, E. Noland, E. Clancy, A. Wu, and A. Zisserman, “A short note on the kinetics-700–2020 human action dataset,” 2020, arXiv: 2010.10864.

[53] A. Stergiou, G. Kapioudis, G. Kalliatakis, C. Chrysoulas, R. Veltkamp, and M. S. Ryoo, “AssembleNet: Searching for multi-stream neural connectivity in video architectures,” in Proc. Int. Conf. Learn. Representations, 2020.

[54] L. Sun, K. Jia, D. Yeung, and B. E. Shi, “Human action recognition using factorized spatio-temporal convolutional networks,” in Proc. Int. Conf. Comput. Vis., 2015, pp. 4597–4605.

[55] L. Sun, Z. Kuang, L. Sheng, W. Ouyang, and W. Zhang, “Optical flow guided feature: A fast and robust motion representation for video action recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern. Recognit., 2018, pp. 1390–1399.

[56] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3D convolutional networks,” in Proc. Int. Conf. Comput. Vis., 2015, pp. 4489–4497.

[57] D. Tran, H. Wang, L. Torresani, and M. Feiszli, “Video classification with channel-separated convolutional networks,” in Proc. Int. Conf. Comput. Vis., 2019, pp. 5552–5561.

[58] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, “A closer look at spatiotemporal convolutions for action recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern. Recognit., 2018, pp. 6450–6459.

[59] H. Wang, D. Tran, L. Torresani, and M. Feiszli, “Video modeling with correlation networks,” in Proc. IEEE Conf. Comput. Vis. Pattern. Recognit., 2020, pp. 349–358.

[60] L. Wang et al., “Temporal segment networks: Towards good practices for deep action recognition,” in Proc. Eur. Conf. Comput. Vis., 2016, pp. 20–36.

[61] L. Wang and A. Gupta, “Video as space-time region graphs,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 399–417.

[62] X. Wang et al., “AttentionNAS: Spatiotemporal attention cell search for video classification,” in Proc. Eur. Conf. Comput. Vis., Springer, 2020, pp. 449–465.

[63] Y. Wang, L. Jiang, M.-H. Yang, L.-J. Li, M. Long, and L. Fei-Fei, “Eidetic 3D LSTM: A model for video prediction and beyond,” in Proc. Int. Conf. Learn. Representations, 2019.

[64] C. Wu, C. Feichtenhofer, H. Fan, K. He, P. Krähenbühl, and R. Girshick, “Long-term feature banks for detailed video understanding,” in Proc. IEEE Conf. Comput. Vis. Pattern. Recognit., 2019, pp. 284–293.

[65] H. Wu, J. Liu, X. Zhu, M. Wang, and Z.-J. Zha, “Multi-scale spatial-temporal integration convolutional tube for human action recognition,” in Proc. Int. Joint Conf. Artif. Intell., 2020, pp. 753–759.

[66] W. Wu, D. He, T. Lin, F. Li, C. Gan, and E. Ding, “MVNet: Multi-view fusion network for efficient video recognition,” in Proc. 35th AAAI Conf. Artif. Intell., 2021, pp. 2943–2951.

[67] Z. Wu, H. Li, Y. Zheng, C. Xiong, Y.-G. Jiang, and L. S. Davis, “A coarse-to-fine framework for resource efficient video recognition,” in Int. J. Comput. Vis., vol. 129, pp. 2965–2977, 2021.

[68] Z. Wu, C. Xiong, C.-Y. Ma, R. Socher, and L. S. Davis, “AdaFrame: Adaptive frame selection for fast video recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern. Recognit., 2019, pp. 1278–1287.

[69] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy, “Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 305–321.

[70] C. Yang, Y. Xu, J. Shi, B. Dai, and B. Zhou, “Temporal pyramid network for action recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern. Recognit., 2020, pp. 591–600.

[71] C. Zach, T. Pock, and H. Bischof, “A duality based approach for realtime TV-L 1 optical flow,” in Proc. Joint Pattern Recognit. Symp., 2007, pp. 214–223.

[72] C. Zhang, A. Gupta, and A. Zisserman, “Temporal query networks for fine-grained video understanding,” in Proc. IEEE Conf. Comput. Vis. Pattern. Recognit., 2021, pp. 4486–4496.

[73] S. Zhang, S. Guo, W. Huang, M. R. Scott, and L. Wang, “V4D: 4D convolutional neural networks for video-level representation learning,” in Proc. Int. Conf. Learn. Representations, 2020.

[74] L. Zhao, T. Liu, X. Peng, and D. Metaxas, “Maximum-entropy adversarial data augmentation for improved generalization and robustness,” 2020, arXiv: 2010.08001.

[75] Y. Zhao, Y. Xiong, and D. Lin, “Trajectory convolution for action recognition,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 2208–2219.

[76] B. Zhou, A. Andonian, A. Oliva, and A. Torralba, “Temporal relational reasoning in videos,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 803–818.

[77] Y. Zhou, X. Sun, Z. Zha, and W. Zeng, “MiCT: Mixed 3D/2D convolutional tube for human action recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern. Recognit., 2018, pp. 449–458.

[78] X. Zhu, C. Xu, L. Hui, C. Lu, and D. Tao, “Approximated bilinear modules for temporal modeling,” in Proc. Int. Conf. Comput. Vis., 2019, pp. 3493–3502.

[79] M. Zolfaghari, K. Singh, and T. Brox, “ECO: Efficient convolutional network for online video understanding,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 695–712.

Suddhakaran Sudhakaran received the PhD (Cum Laude) degree in computer science from the University of Trento, Italy, in 2019. He is a researcher with Samsung AI Centre, Cambridge, U.K. Prior to joining Samsung AI Centre, he was a postdoc with the University of Trento. During his PhD, he was part of the Technologies of Vision (TeV) Lab in FBK, advised by Dr. Oswald Lanz. His research interests are in the area of efficient and effective video representation learning, specifically human behavior understanding.
Sergio Escalera is full professor with the Department of Mathematics and Informatics, Universitat de Barcelona, research line director with the Computer Vision Center and distinguished professor with Aalborg University. He is vice-president of ChaLearn Looking with People events. He is co-creator of Codalab open source platform for challenges organization and co-founder of the NeurIPS competition and Datasets and Benchmarks tracks. He is also fellow of the ELLIS European Laboratory for Learning and Intelligent Systems working within the Human-centric Machine Learning program, fellow of AAIA Asia-Pacific Artificial Intelligence Association, fellow of the International Association for Pattern Recognition and vice-chair of IAPR TC-12: Multimedia and visual information systems. He received an Amazon Research Award, a CVPR best paper award nominee and a CVPR outstanding reviewer award. His research interests include inclusive and transparent analysis of humans from visual, video and multi-modal data.

Oswald Lanz received the graduated degree in mathematics and PhD degree in computer science from the University of Trento. He is full professor with the Faculty of Engineering, Free University of Bolzano-Bozen, and head of Visual Computing and Learning research line. He is research director with Covision Lab. Prior to this he was research scientist and head of research unit with FBK Trento. He is member of ELLIS the European Lab for Learning and Intelligent Systems, co-author in best paper awards with ACM Multimedia and ICIAP, and 2019 recipient of Amazon Machine Learning Research Awards. He holds patents on object detection and tracking in video. His research interests are in computer vision and multi-modal learning with a special interest in structured representation learning for video understanding.