Spatio-Temporal Channel Correlation Networks for Action Classification

Ali Diba[1,4,⋆], Mohsen Fayyaz[2,⋆], Vivek Sharma[3], M.Mahdi Arzani[4], Rahman Yousefzadeh[4], Juergen Gall[2], Luc Van Gool[1,4]

[1]ESAT-PSI, KU Leuven, [2]University of Bonn,[3]CV:HCI, KIT, Karlsruhe, [4]Sensifai
[1]{firstname.lastname}@kuleuven.be, [2]{lastname}@iai.uni-bonn.de,
[3]{firstname.lastname}@kit.edu,[4]{firstname.lastname}@sensifai.com

Abstract. The work in this paper is driven by the question if spatio-temporal correlations are enough for 3D convolutional neural networks (CNN)? Most of the traditional 3D networks use local spatio-temporal features. We introduce a new block that models correlations between channels of a 3D CNN with respect to temporal and spatial features. This new block can be added as a residual unit to different parts of 3D CNNs. We name our novel block 'Spatio-Temporal Channel Correlation' (STC). By embedding this block to the current state-of-the-art architectures such as ResNext and ResNet, we improve the performance by 2-3% on the Kinetics dataset. Our experiments show that adding STC blocks to current state-of-the-art architectures outperforms the state-of-the-art methods on the HMDB51, UCF101 and Kinetics datasets. The other issue in training 3D CNNs is about training them from scratch with a huge labeled dataset to get a reasonable performance. So the knowledge learned in 2D CNNs is completely ignored. Another contribution in this work is a simple and effective technique to transfer knowledge from a pre-trained 2D CNN to a randomly initialized 3D CNN for a stable weight initialization. This allows us to significantly reduce the number of training samples for 3D CNNs. Thus, by fine-tuning this network, we beat the performance of generic and recent methods in 3D CNNs, which were trained on large video datasets, e.g. Sports-1M, and fine-tuned on the target datasets, e.g. HMDB51/UCF101. [1]

1 Introduction

Compelling advantages of exploiting temporal rather than merely spatial cues for video classification have been shown lately [1–3]. In recent works, researchers have focused on improving modeling of spatio-temporal correlations. Like 2D CNNs, 3D CNNs try to learn local correlation along input channels. Therefore, 3D CNNs neglect the hidden information in between channels correlations in both directions: space and time, which limits the performance of these architectures. Another major problem in using 3D CNNs is training the video architectures calls for extra large labeled datasets. All of these issues negatively influence their computational cost and performance. To avoid

[1] ⋆Ali Diba and Mohsen Fayyaz contributed equally to this work. Mohsen Fayyaz contributed to this work while he was at Sensifai.

these limitations, we propose (i) a new network architecture block that efficiently captures both spatial-channels and temporal-channels correlation information throughout network layers; and (ii) an effective supervision transfer that bridges the knowledge transfer between different architectures, such that training the networks from scratch is no longer needed.

Motivated by the above observations, we introduce the spatio-temporal channel correlation (STC) block. The aim of this block is considering the information of inter channels correlations over the spatial and temporal features simultaneously. For any set of transformation in the network (e.g. convolutional layers) a STC block can be used for performing spatio-temporal channel correlation feature learning. The STC block has two branches: a spatial correlation branch (SCB) and a temporal correlation branch (TCB). The SCB considers spatial channel-wise information while TCB considers the temporal channel-wise information. The input features $I \in \mathbb{R}^{H \times W \times T \times C}$ are fed to SCB and TCB. In SCB a spatial global pooling operation is done to generate a representation of the global receptive field which plays two vital roles in the network: (i) considering global correlations in I by aggregating the global features over the input, (ii) providing a channel-wise descriptor for analyzing the between channels correlations. This channel-wise feature vector is then fed to two bottleneck fully connected layers which learn the dependencies between channels. The same procedure happens in TCB, however, for the first step a temporal global pooling is used instead of the spatial global pooling. Output features of these two branches are then combined and returned as the output of the STC block. These output features can be combined with the output features of the corresponding layer(s). By employing such features along-side traditional features available inside a 3D CNN, we enrich the representation capability of 3D CNNs. Therefore, the STC block equipped 3D CNNs are capable of learning channel wise dependencies which enables them to learn better representations of videos. We have added the STC block to the current state-of-the-art 3D CNN architectures such as 3D-ResNext and 3D-ResNet [4]. The STC block is inserted after each residual block of these networks.

As mentioned before, training 3D CNNs from scratch needs a large labeled dataset. It has been shown that training 3D Convolution Networks [2] from scratch takes two months [5] for them to learn a good feature representation from a large scale dataset like Sports-1M, which is then finetuned on target datasets to improve performance. Another major contribution of our work therefore is to achieve supervision transfer across architectures, thus avoiding the need to train 3D CNNs from scratch. Specifically, we show that a 2D CNN pre-trained on ImageNet can act as '*a teacher*' for supervision transfer to a randomly initialized 3D CNN for a stable weight initialization. In this way we avoid the excessive computational workload and training time. Through this transfer learning, we outperform the performance of generic 3D CNNs (C3D [2]) which was trained on Sports-1M and finetuned on the target datasets HMDB51 and UCF101.

2 Related Work

Video Classification with and without CNNs: Video classification and understanding has been studied for decades. Several techniques have been proposed to come up with efficient spatio-temporal feature representations that capture the appearance and motion propagation across frames in videos, such as HOG3D [6], SIFT3D [7], HOF [8], ESURF [9], MBH [10], iDTs [11], and more. These were all hand-engineered. Among these, iDTs yielded the best performance, at the expense of being computationally expensive and lacking scalability to capture semantic concepts. It is noteworthy that recently several other techniques [12] have been proposed that also try to model the temporal structure in an efficient way.

Using deep learning, the community went beyond hand-engineered representations and learned the spatio-temporal representations in an end-to-end manner. These methods operate on 2D (frame-level) or 3D (video-level) information. In the 2D setting, CNN-based features of individual frames are modeled via LSTMs/RNNs to capture long-term temporal dependencies [13, 3], or via feature aggregation and encoding using Bilinear models [1], VLAD [14], Fisher encoding [15] etc. Recently, several temporal architectures have been proposed for video classification, where the input to the network consists of either RGB video clips or stacked optical-flow frames. The filters and pooling kernels for these architectures are 3D (x, y, time). The most intuitive are 3D convolutions ($s \times s \times d$) [3] where the kernel temporal depth d corresponds to the number of frames used as input, and s is the kernel spatial size. Simonyan et al. [16] proposed a two-stream network, cohorts of RGB and flow CNNs. In their flow stream CNNs, the 3D convolution has d set to 10. Tran et al. [2] explored 3D CNNs with filter kernel of size $3 \times 3 \times 3$ and in [5] extended the ResNet architecture with 3D convolutions. Feichtenhofer et al. [17] propose 3D pooling. Sun et al. [18] decomposed the 3D convolutions into 2D spatial and 1D temporal convolutions. Carreira et al. [19] proposed converting a pre-trained 2D Inception-V1 [20] architecture to 3D by inflating all the filters and pooling kernels with an additional temporal dimension d. The Non-local Neural Networks [21] proposes a new building block for CNNs which captures long range dependencies. Feichtenhofer et al. [22] introduce residual connections for learning the dependencies between motion and appearance stream of a two stream CNN. Varol et al. [23] have studied the long-term temporal convolutions for learning better representations of long-term activities in video. The spatio-temporal feature gating method introduced in [24] addresses a similar issue by introducing the feature gating module. Miech et al. [25] introduce the context gating method which applies gating to the features of the output layer. Most of these architectures neglect the channel wise information throughout the whole architecture. To the best of our knowledge, our STC block is the first 3D block that integrates channel wise information over 3D networks' layers.

Transfer Learning: Finetuning or specializing the learned feature representations of a pre-trained network trained on another dataset to a target dataset is commonly referred to as transfer learning. Recently, several works have shown that transferring knowledge within or across modalities (e.g. RGB→RGB [26] vs. RGB→Depth [27], RGB→Optical-Flow [27, 28], RGB→Sound [29], Near-Infrared→RGB [30]) is effective, and leads to significant improvements in performance. They typically amount to jointly learning representations in a shared feature space. Mansimov et al. [31] have studied various methods of weight initialization which are the principle idea for inflation approaches. Our work differs substantially. Our goal is to transfer supervision across architectures (i.e. 2D→3D CNNs), not necessarily limited to transferring infor-

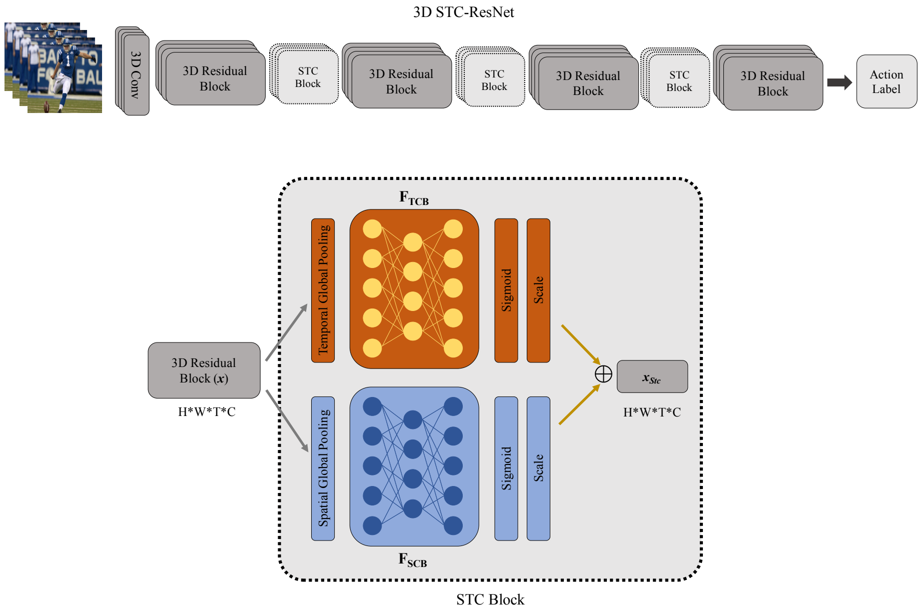

Fig. 1. STC-ResNet. Our STC block is applied to the 3D ResNet. The 3D network uses video clips as input. The 3D feature-maps from the clips are densely propagated throughout the network. The STC operates on the different levels of feature maps in the network to extract spatial and temporal channel relations as new source of information. The output of the network is a video-level prediction.

mation between RGB models only, as our solution can be easily adopted across modalities too.

3 Proposed Method

Our approach with the newly proposed neural block, STC, is to capture different and new information in deep CNNs from videos. The spatio-temporal channel correlation block is meant to extract relations between different channels in the different layers of 3D CNNs. The STC block considers these relations in space and time dimensions. In addition, as another major contribution of our work, we show knowledge transfer between cross architectures (i.e. 2D→3D CNNs), thus avoiding the need to train 3D CNNs from scratch. Details about the transfer learning is given in Section 3.2.

3.1 Spatio-Temporal Channel Correlation (STC) Block

STC is a computational block which can be added to any 3D CNN architecture. Therefore, we have added our STC block to the ResNet and ResNext 3D CNNs introduced by [4]. After each convolutional block in ResNet and ResNext, the STC blocks are inserted

to enrich the feature representation. As it was mentioned previously, this new block is exploiting both spatial and temporal information by considering the filters correlation in both spatial and temporal dimension. As input to the STC block, we consider feature maps coming from previous convolution layers.
The STC block has a dual path structure which represents different levels of concept and information. Each of these paths have different modules; channel or filter information embedding and capturing dependencies. Our approach is inspired by the Squeeze-and-Excitation [32] method which uses global average pooling (spatial and temporal) following with two bottleneck fully connected layers and sigmoid activation. In contrast to [32], the STC block has two branches or in other words a dual path; one considering pure channel-wise information and the other takes temporal channel-wise information. Since we are solving video classification, it makes sense to extract more meaningful representations in both spatial and temporal approaches. The STC is capturing channel dependencies information based on this theory. In the following we describe both branches and their integration into the known 3D architectures like 3D-ResNet [4].

Notation. The output feature-maps of the 3D convolutions and pooling kernels at the l^{th} layer extracted for an input video is a tensor $X \in \mathbb{R}^{H \times W \times T \times C}$ where H, W, T and C are the height, width, temporal depth and number of channels of the feature maps, respectively. The 3D convolution and pooling kernels are of size $(s \times s \times d)$, where d is the temporal depth and s is the spatial size of the kernels.

Temporal Correlation Branch (TCB): In this path the feature map will be squeezed by both spatial and temporal dimensions to extract channel descriptors. If we consider X as the input to STC, the output of the first stage, which is a global spatio-temporal pooling is:

$$z_{tcb} = \frac{1}{W \times H \times T} \sum_{i}^{W} \sum_{j}^{H} \sum_{t}^{T} x_{ijt} \ . \tag{1}$$

To obtain the filters non-linear relations, we apply two fully connected layers. The feature dimension is reduced in the first FC layer to C/r (r is reduction ratio) and is increased again to C by the second FC layer. Since we used global spatial-temporal pooling over all dimensions of receptive fields, in the next operation, channel-wise information will be extracted. Right after the sigmoid function, the output of the temporal branch (x_{tcb}) will be calculated by rescaling X using the s_{tcb} vector. So s_{tcb}, output of the bottleneck layers, and x_{tcb}, the branch output, are calculated in this way:

$$s_{tcb} = F_{tcb}(z_{tcb}, W) = W_2(W_1 z_{tcb}) \tag{2}$$

$$x_{tcb} = s_{tcb} \cdot X \ . \tag{3}$$

W is the parameter set for the bottleneck layers, including $W_1 \in \mathbb{R}^{\frac{C}{r} \times C}$, $W_2 \in \mathbb{R}^{C \times \frac{C}{r}}$ which are FC layers parameters respectively. F_{tcb} is the symbol of fully-connected functions to calculate the s_{tcb}.

Spatial Correlation Branch (SCB): The main difference in this branch compared to the temporal branch is in the aggregation method. The spatial branch shrinks the

Table 1. 3D ResNet vs. STC-ResNet and STC-ResNext. All the proposed architectures incorporate 3D filters and pooling kernels. Each convolution layer shown in the table corresponds the composite sequence BN-ReLU-Conv operations.

Layers	Output Size	3D-ResNet101	3D STC-ResNet101	3D STC-ResNext101
3D Convolution	$56 \times 56 \times 8$	$7 \times 7 \times 7$ conv, stride 2		
3D Pooling	$56 \times 56 \times 8$	$3 \times 3 \times 3$ max pool, stride 1		
Res_1	$28 \times 28 \times 8$	$\begin{bmatrix} conv, 1 \times 1 \times 1, 64 \\ conv, 3 \times 3 \times 3, 64 \\ conv, 1 \times 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} conv, 1 \times 1 \times 1, 64 \\ conv, 3 \times 3 \times 3, 64 \\ conv, 1 \times 1 \times 1, 256 \\ fc, [16, 256] \end{bmatrix} \times 3$	$\begin{bmatrix} conv, 1 \times 1 \times 1, 128 \\ conv, 3 \times 3 \times 3, 128 \quad C = 32 \\ conv, 1 \times 1 \times 1, 256 \\ fc, [16, 256] \end{bmatrix} \times 3$
Res_2	$14 \times 14 \times 4$	$\begin{bmatrix} conv, 1 \times 1 \times 1, 128 \\ conv, 3 \times 3 \times 3, 128 \\ conv, 1 \times 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} conv, 1 \times 1 \times 1, 128 \\ conv, 3 \times 3 \times 3, 128 \\ conv, 1 \times 1 \times 1, 512 \\ fc, [32, 512] \end{bmatrix} \times 4$	$\begin{bmatrix} conv, 1 \times 1 \times 1, 256 \\ conv, 3 \times 3 \times 3, 256 \quad C = 32 \\ conv, 1 \times 1 \times 1, 512 \\ fc, [32, 512] \end{bmatrix} \times 4$
Res_3	$7 \times 7 \times 2$	$\begin{bmatrix} conv, 1 \times 1 \times 1, 256 \\ conv, 3 \times 3 \times 3, 256 \\ conv, 1 \times 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} conv, 1 \times 1 \times 1, 256 \\ conv, 3 \times 3 \times 3, 256 \\ conv, 1 \times 1 \times 1, 1024 \\ fc, [64, 1024] \end{bmatrix} \times 23$	$\begin{bmatrix} conv, 1 \times 1 \times 1, 512 \\ conv, 3 \times 3 \times 3, 512 \quad C = 32 \\ conv, 1 \times 1 \times 1, 1024 \\ fc, [64, 1024] \end{bmatrix} \times 23$
Res_4	$4 \times 4 \times 1$	$\begin{bmatrix} conv, 1 \times 1 \times 1, 512 \\ conv, 3 \times 3 \times 3, 512 \\ conv, 1 \times 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} conv, 1 \times 1 \times 1, 512 \\ conv, 3 \times 3 \times 3, 512 \\ conv, 1 \times 1 \times 1, 2048 \\ fc, [128, 2048] \end{bmatrix} \times 3$	$\begin{bmatrix} conv, 1 \times 1 \times 1, 512 \\ conv, 3 \times 3 \times 3, 512 \quad C = 32 \\ conv, 1 \times 1 \times 1, 2048 \\ fc, [128, 2048] \end{bmatrix} \times 3$
Classification Layer	$1 \times 1 \times 1$	$4 \times 4 \times 1$ avg pool		
		400D softmax		

channel-wise information with respect to the temporal dimension and does global spatial pooling on the input feature map. Therefore this branch is considering the temporal-channel information extraction to enrich the representation in each layer. The calculation of the first operation of the branch comes as following:

$$z_{scb} = \frac{1}{W \times H} \sum_{i}^{W} \sum_{j}^{H} x_{ijT} \tag{4}$$

After the pooling layer, we obtain z_{scb} which is a vector with size of $T \times C$. Afterward, there are the fully connected layers to extract the temporal based channel relations. In this branch the first FC layer size is $(T \times C)/r$ and the second FC size is C. Here is the computation description:

$$s_{scb} = F_{scb}(z_{scb}, W) = W_2(W_1 z_{scb}) \tag{5}$$

$$x_{scb} = s_{scb} \cdot X \tag{6}$$

with $W_1 \in \mathbb{R}^{\frac{(T \times C)}{r} \times (T \times C)}$ and $W_2 \in \mathbb{R}^{C \times \frac{T \times C}{r}}$. By considering both of the branches, the final output of the block (x_{stc}) is computed by averaging over x_{tcb} and x_{scb}.

$$x_{stc} = avg(x_{tcb}, x_{scb}) \tag{7}$$

In the case of 3D ResNet or ResNext, this output will be added to the residual layer to have the final output of the Convolution (Conv) blocks.

3.2 Knowledge Transfer

In this section, we describe our method for transferring knowledge between architectures, i.e. pre-trained 2D CNNs to 3D CNNs. Therefore we bypass the need to train the 3D CNNs from scratch with supervision or training with large datasets.

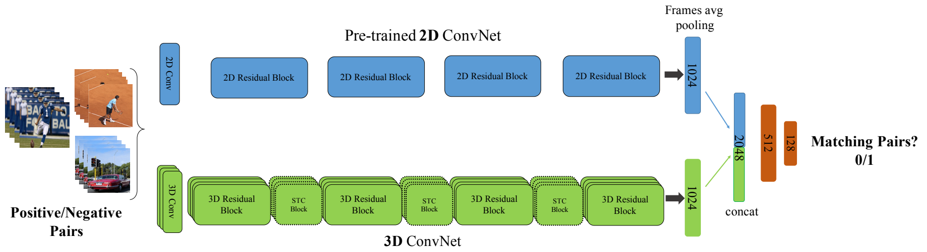

Fig. 2. Architecture for knowledge transfer from a pre-trained 2D CNN to a 3D CNN. The 2D network operates on RGB frames, and the 3D network operates on video clips for the same time stamp. The 2D CNN acts as a teacher for knowledge transfer to the 3D CNN, by teaching the 3D CNN to learn mid-level feature representation by solving an image-video correspondence task. The model parameters of the 2D CNN are frozen, while the task is to effectively learn the model parameters of the 3D CNN only.

Lets $\mathcal{I}$ be a pre-trained 2D CNN which has learned a rich representation from labeled images dataset, while $\mathcal{V}$ being a 3D CNN which is randomly initialized using [33] and we want to transfer the knowledge of the representation from $\mathcal{I}$ to $\mathcal{V}$ for a stable weight initialization. This allows us to avoid training $\mathcal{V}$ from scratch, which has million more parameters, and would require heavy computational workload and training time of months [5]. In the current setup, $\mathcal{I}$ acts as a teacher for knowledge transfer to the $\mathcal{V}$ architecture.

Intuitively, our method uses correspondence between frames and video clips available by the virtue of them appearing together at the same time. Given a pair of X frames and video clip for the same time stamp, the visual information in both frames and video are the same. We leverage this for learning mid-level feature representations by an image-video correspondence task between the 2D and 3D CNN architecture, as depicted in Figure 2. We use 2D ResNet [34] pre-trained on ImageNet [35] as $\mathcal{I}$, and the STC-ResNet network as $\mathcal{V}$. The 2D ResNet CNN has 4 convolution blocks and one fully connected layer at the end, while our 3D architecture has 4 3D-convolution blocks with an STC block and we add a fully-connected layer after the last block. We concatenate the last fc layers of both architectures, and connect them with the 2048-dimensional fc layer which is in turn connected to two fully connected layers with 512 and 128 sizes (fc1 , fc2) and to the final binary classifier layer. We use a binary matching classifier: given X frames and a video clip, decide whether the pairs belong to each other or not. For a given pair, X frames are fed sequentially into the network $\mathcal{I}$ and we average the last 2D fc features over the X frames, resulting in a 1024-D feature representation. In parallel the video clip is fed to the network $\mathcal{V}$, and we extract the 3D fc features (1024-D), and concatenate them, which is then passed to the fully connected layers for classification. For training, we use a binary classification loss.

During the training, the model parameters of $\mathcal{I}$ are frozen, while the task is to effectively learn the model parameters of $\mathcal{V}$ without any additional supervision than correspondences between frames and video. The pairs belonging to the same time stamp from the same video are positive pairs, while the pairs coming from two different videos by randomly sampling X frames and video clips from two different videos is a negative pair. Note that, during back-propagation, only the model parameters for $\mathcal{V}$ are updated, i.e., transferring the knowledge from $\mathcal{I}$ to $\mathcal{V}$. In our experiments we show that a stable weight initialization of $\mathcal{V}$ is achieved, and when fine-tuned on the target dataset, it adapts quickly, thus avoiding training the model from scratch. We also show that by using our proposed knowledge transfer method, 3D CNNs can be trained directly on small datasets like UCF101 and achieve a better performance than training from scratch.

Since our transfer learning is unsupervised and there is no need of video label, we have applied on a collection of unlabeled videos. Further, our experiments in Section 4 demonstrate that our proposed transfer learning of STC-ResNext outperforms the generic 3D CNNs by a significant margin which was trained on a large video dataset, Sports-1M [36], and finetuned on the target datasets, HMDB51 or UCF101.

4 Experiments

In this section, we first introduce the datasets and implementation details of our proposed approach. Afterwards, we provide an extensive study on the architecture of the proposed STC-ResNet and STC-ResNext, which are 3D CNNs. Following, we evaluate and compare our proposed methods with the baselines and other state-of-the-art methods. Finally, we compare our transfer learning: $2D \rightarrow 3D$ CNN performance with generic state-of-the-art 3D CNN methods.

4.1 Datasets

We evaluate our proposed method on three challenging video datasets with human actions, namely HMDB51 [37], UCF101 [38], and Kinetics [19]. Table 2 shows the details of the datasets. For all of these datasets, we use the standard training/testing splits and protocols provided by the datasets. For HMDB51 and UCF101, we report the average accuracy over the three splits and for Kinetics, we report the performance on the validation and test set.

Kinetics: Kinetics is a new challenging human action recognition dataset introduced by [19], which contains 400 action classes. There are two versions of this dataset: untrimmed and trimmed. The untrimmed videos contain the whole video in which the activity is included in a short period of it. However, the trimmed videos contain the activity part only. We evaluate our models on the trimmed version. We use all training videos for training our models from scratch.

UCF101: For evaluating our STC-Nets architectures, we first trained them on the Kinetics dataset, and then fine-tuned them on UCF101. Furthermore, we also evaluate our models by training them from scratch on UCF101 using randomly initialized weights to be able to investigate the effect of pre-training on a huge dataset, such as Kinetics.

HMDB51: Same as UCF101 evaluation we fine-tune the models on HMDB51, which were pre-trained from scratch on Kinetics. Also, we similarly evaluate our models by training them from scratch on HMDB51 using randomly initialized weights.

Table 2. Details of the datasets used for evaluation. The 'Clips' shows the total number of short video clips extracted from the 'Videos' available in the dataset.

Data-set	# Clips	# Videos	# Classes
HMDB51 [37]	6,766	3,312	51
UCF101 [38]	13,320	2,500	101
Kinetics [19]	306,245	306,245	400

4.2 Implementation Details

We use the PyTorch framework for the implementation and all the networks are trained on 8 Tesla P100 NVIDIA GPUs. Here, we describe the implementation details of our two schemes, 3D CNN architectures and knowledge transfer from 2D to 3D CNNs for stable weight initialization.

STC-Nets.

Training: We train our STC-Nets (STC-ResNet/ResNext) from scratch on Kinetics. Our STC-Net operates on a stack of 16/32/64 RGB frames. We resize the video to 122px when smaller, and then randomly apply 5 crops (and their horizontal flips) of size 112×112. For the network weight initialization, we adopt the same technique proposed in [33]. For the network training, we use SGD, Nesterov momentum of 0.9, weight decay of 10^{-4} and batch size of 128. The initial learning rate is set to 0.1, and reduced by a factor of 10 manually when the validation loss is saturated. The maximum number of epochs for the whole Kinetics dataset is set to 200. Batch normalization also has been applied. The reduction parameter in STC blocks, r, is set to 4.

Testing: For video prediction, we decompose each video into non-overlapping clips of 16/32/64 frames. The STC-Net is applied over the video clips by taking a 112×112 center-crop, and finally we average the predictions over all clips to make a video-level prediction.

Knowledge Transfer: $2D \rightarrow 3D$ ***CNNs.*** We employ 2D ResNet architecture, pre-trained on ImageNet [35], while the 3D CNN is our STC-ResNet network. To the 2D CNN, 16 RGB frames are fed as input. The input RGB images are randomly cropped to the size 112×112, and then mean-subtracted for the network training. To supervise transfer to the STC-ResNet, we replace the previous classification layer of the 2D CNN with a 2-way softmax layer to distinguish between positive and negative pairs. We use stochastic gradient descent (SGD) with mini-batch size of 32 with a fixed weight decay of 10^{-4} and Nesterov momentum of 0.9. For network training, we start with learning rate set to 0.1 and decrease it by a factor of 10 every 30 epochs. The maximum number of epochs is set to 150. For training data, we use approx. 500K unlabeled videos from YouTube8m dataset [39].

4.3 Ablation Study on Architecture Design

To evaluate our STC block on 3D CNNs model, we conducted an architecture study and evaluated different configurations. For this work, we mainly focused on 3D versions for ResNet and ResNext with different input size and depth. Our choice is based on the recently presented good performance of these networks in video classification [4].

Model Depth: We first analyze the impact of the architecture depth with 3D-ResNet and 3D-ResNext and we have done a series of evaluations on the network size. For the architecture study, the model weights were initialized using [33].

We employ three different sizes of 3D STC-ResNet; 18, 50, 101 with STC blocks. Evaluations results of these 3D STC-ResNet models are reported in the Table 3. As it can be observed, by adding small overhead of STC blocks, STC-Nets can achieve reasonable performance even in a smaller version of ResNet, since our STC-ResNet50 is comparable in accuracy with a regular ResNet101.

Table 3. Evaluation results of 3D STC-ResNet model with network sizes of 18, 50, and 101 on UCF101 split 1. All models were trained from scratch.

Model Depth	Accuracy %
3D-ResNet 101	46.7
STC-ResNet 18	42.8
STC-ResNet 50	46.2
STC-ResNet 101	**47.9**

Temporal Input Size: The number of input frames plays a key role in activity recognition. Therefore, we have reported the performance of our 3D STC-ResNet and 3D STC-ResNext with different number of input frames in Table 4. Our evaluation shows that longer clips as input will yield better performance, which confirms the observations made in [4, 19].

TCB vs SCB: We also have studied the impact of the TCB and SCB branches in our STC-Nets. Since each of them considers a different concept in the branch, we evaluated the performance for three settings: SCB only, TCB only, and SCB-TCB combination (STC). In Table 5, the importance of the channel correlation branches is shown. As it is shown, incorporating both branches to capture different types of correlations is performing better than SCB or TCB alone.

Table 4. Evaluation results of STC-ResNet and 3D STC-ResNext models with temporal depths of 16, 32, and 64 frames for all three splits of UCF101 and HMDB51.

Model	UCF101	HMDB51
STC-ResNet 101 (16 frames)	90.1	62.6
STC-ResNet 101 (32 frames)	93.2	68.9
STC-ResNet 101 (64 frames)	93.7	70.5
STC-ResNext 101 (16 frames)	92.3	65.4
STC-ResNext 101 (32 frames)	95.8	72.6
STC-ResNext 101 (64 frames)	96.5	74.9

Table 5. Performance comparison using different channel correlation blocks (TCB vs SCB) for UCF101 split 1.

Channel Correlation Branch	Accuracy %
SCB	46.1
TCB	47.2
TCB + SCB	**47.9**

Frame Sampling Rate: Finding the right configuration of input-frames which are fed to the CNNs for capturing the appearance and temporal information plays a very critical role in temporal CNNs. For this reason, we investigated the impact of the frame sampling rate for the input stream. The STC-ResNet101 has been used for the ablation study on frame sampling rate for training and testing. We evaluate the model by varying the temporal stride of the input frames in the following set $\{1, 2 ,4, 16\}$. Table 6 presents the accuracy of STC-ResNet101 trained on inputs with different sampling rates. The best results are obtained with sampling rate of 2, which we also used for other 3D CNNs in the rest of the experiments.

Table 6. Evaluation results of different frame sampling rates for the STC-ResNet101 model. Trained and tested on UCF101 split 1.

Input Stride	1	2	4	16
Accuracy %	44.6%	**47.9**%	46.8%	40.3%

4.4 Knowledge Transfer

To apply our proposed supervision transfer, we have tested 2D ResNet as basic pretrained model on ImageNet, while 3D-ResNet and our STC-ResNet are randomly initialized using [33] and used as target 3D CNNs. We show that a stable weight initialization via transfer learning is possible for 3D CNN architectures, which can be used as a good starting model for training on small datasets like UCF101 or HMDB51. Since the transfer learning pipeline for 3D CNNs have been tested with two different deep architectures (3D-ResNet and STC-Nets), we clearly show the generalization capacity of our method in deep architectures, which can be easily adopted for other deep networks and tasks which use the similar architectures.

Table 7. Transfer learning results for 3D CNNs by 2D CNNs over all three splits of UCF101 and HMDB51. All models have the same depth of 101.

3D CNNs	**UCF101**	**HMDB51**
3D-ResNet-Baseline	88.9	61.7
3D-ResNet-Inflation	90.4	62.6
3D-ResNet-**Transfered**	91.3	64.2
STC-ResNet-Baseline	90.1	62.6
STC-ResNet-**Transfered**	92.6	66.1

Table 7 shows the results. The baseline is trained from scratch using random initialization. As it is shown, our transfer method performs better than the baseline for the standard 3D-ResNet as well as for our proposed STC-ResNet. Using inflation also improves the baseline, but it is outperformed by our approach. Note that inflation can only be used if the structure of the 2D and 3D network are the same, while our approach allows to transfer the knowledge from any 2D CNN to a 3D CNN, e.g., from 2D-ResNet to the 3D STC-ResNet as in Table 7, which is not possible by inflation.

Table 8. Comparison results of our models with other state-of-the-art methods on Kinetics dataset. * denotes the pre-trained version of C3D on the Sports-1M.

Method	**Top1-Val**	**Top5-Val**
DenseNet3D	59.5	-
Inception3D	58.9	-
C3D* [4]	55.6	-
3D ResNet101 [4]	62.8	83.9
3D ResNext101 [4]	65.1	85.7
RGB-I3D [19]	68.4	88
STC-ResNet101 (16 frames)	64.1	85.2
STC-ResNext101 (16 frames)	66.2	86.5
STC-ResNext101 (32 frames)	**68.7**	**88.5**

4.5 Comparison with the state-of-the-art

Finally, after exploring and studying on STC-Net architectures and the configuration of input-data and architecture, we compare our STC-ResNet and STC-ResNext with the state-of-the-art methods by pre-training on Kinetics and finetuning on all three splits of the UCF101 and HMDB51 datasets. For UCF101 and HMDB51, we report the average

Table 9. Accuracy (%) performance comparison of STC-Nets (STC-ResNet/ResNext) with state-of-the-art methods over all three splits of UCF101 and HMDB51.

Method	**UCF101**	**HMDB51**
DT+MVSM [40]	83.5	55.9
iDT+FV [41]	85.9	57.2
C3D [2]	82.3	56.8
Conv Fusion [17]	82.6	56.8
Two Stream [16]	88.6	–
TDD+FV [42]	90.3	63.2
RGB+Flow-TSN [43]	94.0	68.5
P3D [44]	88.6	–
RGB-I3D [19]	95.6	74.8
RGB+Flow-I3D [19]	**98.0**	**80.7**
Inception3D	87.2	56.9
3D ResNet 101 (16 frames)	88.9	61.7
3D ResNet 101-Transfered Knowledge	91.3	64.2
3D ResNext 101 (16 frames)	90.7	63.8
STC-ResNext 101 (16 frames)	92.3	65.4
STC-ResNext 101 (64 frames)	96.5	74.9

accuracy over all three splits. The results for supervision transfer technique experiments were reported in the previous part of experiments.

Table 8 shows the result on Kinetics dataset for STC-Nets compared with state-of-the-art methods. The STC-ResNext101 with 32 frames input depth achieves higher accuracies than RGB-I3D which has the input size of 64 frames.

Table 9 shows the results on the UCF101 and HMDB51 datasets for comparison of STC-Nets with other RGB based action recognition methods. Our STC-ResNext101 (64 frames) model outperforms the 3D-ResNet [5], Inception3D, RGB-I3D[19] and C3D [2] on both UCF101 and HMDB51 and achieves 96.5% and 74.9% accuracy respectively. We also trained Inception3D, a similar architecture to the I3D [19], without using ImageNet on Kinetics and fine-tuned it on UCF101 and HMDB51 to be able to have a fair comparison. As shown in Table 9, STC-ResNext performs better than 3D-ResNext by almost 2% on UCF101. Moreover, we note that the state-of-the-art CNNs [19, 43] use expensive optical-flow maps in addition to RGB input-frames, as in I3D which obtains a performance of 98% on UCF101 and 80% on HMDB51. Because of such a high computation needs, we are not able to run the similar experiments, but as it can be concluded from Table 9, our best RGB model has superior performance than the other RGB based models.

Note that in our work we have not used dense optical-flow maps, and still achieving comparable performance to the state-of-the-art methods [43]. This shows the effectiveness of our STC-Nets to exploit temporal information and spatio-temporal channel correlation in deep CNNs for video clips. This calls for efficient methods like ours instead of computing the expensive optical-flow information (beforehand) which is very computationally demanding, and therefore difficult to obtain for large scale datasets.

5 Conclusion

In this work, we introduced a new ‘Spatio-Temporal Channel Correlation’ (STC) block that models correlations between the channels of a 3D CNN. We clearly show the benefit of exploiting spatio-temporal channel correlations features using the STC block. We equipped 3D-ResNet and 3D-ResNext with our STC block and improved the accuracies by 2-3% on the Kinetics dataset. Our STC blocks are added as a residual unit to other parts of networks and learned in an end-to-end manner. The STC feature-maps model the feature interaction in a more expressive and efficient way without an undesired loss of information throughout the network. Our STC-Nets are evaluated on three challenging action recognition datasets, namely HMDB51, UCF101, and Kinetics. The STC-Net architectures achieve state-of-the-art performance on HMDB51, UCF101 and comparable results on Kinetics in comparison to other temporal deep neural network models. We expect that the proposed STC blocks will also improve other 3D CNNs. Further, we show the benefit of transfer learning between cross architectures, specifically supervision transfer from 2D to 3D CNNs. This provides a valuable and stable weight initialization for 3D CNNs instead of training them from scratch which is also very expensive. Our transfer learning approach is not limited to transfer supervision between RGB models only, as our approach for transfer learning can be easily adopted for transfer across modalities.

Acknowledgements: This work was supported by DBOF PhD scholarship, KU Leuven:CAMETRON project, and KIT:DFG-PLUMCOT project. The authors would like to thank Sensifai engineering team. Mohsen Fayyaz and Juergen Gall have been financially supported by the DFG project GA 1927/4-1 (Research Unit FOR 2535) and the ERC Starting Grant ARCA (677650).

References

1. Diba, A., Sharma, V., Van Gool, L.: Deep temporal linear encoding networks. In: CVPR. (2017)
2. Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3d convolutional networks. In: ICCV. (2015)
3. Yue-Hei Ng, J., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., Toderici, G.: Beyond short snippets: Deep networks for video classification. In: CVPR. (2015)
4. Hara, K., Kataoka, H., Satoh, Y.: Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and imagenet? In: CVPR. (2018)
5. Tran, D., Ray, J., Shou, Z., Chang, S.F., Paluri, M.: Convnet architecture search for spatiotemporal feature learning. arXiv:1708.05038 (2017)
6. Klaser, A., Marszałek, M., Schmid, C.: A spatio-temporal descriptor based on 3d-gradients. In: BMVC. (2008)
7. Scovanner, P., Ali, S., Shah, M.: A 3-dimensional sift descriptor and its application to action recognition. In: ACM'MM. (2007)
8. Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: CVPR. (2008)
9. Willems, G., Tuytelaars, T., Van Gool, L.: An efficient dense and scale-invariant spatio-temporal interest point detector. In: ECCV. (2008)

10. Dalal, N., Triggs, B., Schmid, C.: Human detection using oriented histograms of flow and appearance. In: ECCV. (2006)
11. Wang, H., Schmid, C.: Action recognition with improved trajectories. In: ICCV. (2013)
12. Fernando, B., Gavves, E., Oramas, J.M., Ghodrati, A., Tuytelaars, T.: Modeling video evolution for action recognition. In: CVPR. (2015)
13. Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., Darrell, T.: Long-term recurrent convolutional networks for visual recognition and description. In: CVPR. (2015)
14. Girdhar, R., Ramanan, D., Gupta, A., Sivic, J., Russell, B.: Actionvlad: Learning spatio-temporal aggregation for action classification. In: CVPR. (2017)
15. Tang, P., Wang, X., Shi, B., Bai, X., Liu, W., Tu, Z.: Deep fishernet for object classification. arXiv:1608.00182 (2016)
16. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: NIPS. (2014)
17. Feichtenhofer, C., Pinz, A., Zisserman, A.: Convolutional two-stream network fusion for video action recognition. In: CVPR. (2016)
18. Sun, L., Jia, K., Yeung, D.Y., Shi, B.E.: Human action recognition using factorized spatio-temporal convolutional networks. In: ICCV. (2015)
19. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: CVPR. (2017)
20. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML. (2015)
21. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. CVPR (2018)
22. Feichtenhofer, C., Pinz, A., Wildes, R.: Spatiotemporal residual networks for video action recognition. In: NIPS. (2016) 3468–3476
23. Varol, G., Laptev, I., Schmid, C.: Long-term temporal convolutions for action recognition. TPAMI (2017)
24. Xie, S., Sun, C., Huang, J., Tu, Z., Murphy, K.: Rethinking spatiotemporal feature learning for video understanding. arXiv preprint arXiv:1712.04851 (2017)
25. Miech, A., Laptev, I., Sivic, J.: Learnable pooling with context gating for video classification. arXiv preprint arXiv:1706.06905 (2017)
26. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv:1503.02531 (2015)
27. Gupta, S., Hoffman, J., Malik, J.: Cross modal distillation for supervision transfer. In: CVPR. (2016)
28. Diba, A., Pazandeh, A.M., Van Gool, L.: Efficient two-stream motion and appearance 3d cnns for video classification. In: ECCV Workshops. (2016)
29. Arandjelović, R., Zisserman, A.: Look, listen and learn. ICCV (2017)
30. Limmer, M., Lensch, H.P.: Infrared colorization using deep convolutional neural networks. In: ICMLA. (2016)
31. Mansimov, E., Srivastava, N., Salakhutdinov, R.: Initialization strategies of spatio-temporal convolutional neural networks. arXiv preprint arXiv:1503.07274 (2015)
32. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. arXiv preprint arXiv:1709.01507 (2017)
33. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: ICCV. (2015)
34. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. (2016)
35. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR. (2009)

36. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: CVPR. (2014)
37. Kuehne, H., Jhuang, H., Stiefelhagen, R., Serre, T.: Hmdb51: A large video database for human motion recognition. In: High Performance Computing in Science and Engineering. (2013)
38. Soomro, K., Zamir, A.R., Shah, M.: Ucf101: A dataset of 101 human actions classes from videos in the wild. arXiv:1212.0402 (2012)
39. Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B., Vijayanarasimhan, S.: Youtube-8m: A large-scale video classification benchmark. arXiv:1609.08675 (2016)
40. Cai, Z., Wang, L., Peng, X., Qiao, Y.: Multi-view super vector for action recognition. In: CVPR. (2014)
41. Wang, H., Schmid, C.: Action recognition with improved trajectories. In: ICCV. (2013)
42. Wang, L., Qiao, Y., Tang, X.: Action recognition with trajectory-pooled deep-convolutional descriptors. In: CVPR. (2015)
43. Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Van Gool, L.: Temporal segment networks: Towards good practices for deep action recognition. In: ECCV. (2016)
44. Qiu, Z., Yao, T., Mei, T.: Learning spatio-temporal representation with pseudo-3d residual networks. In: 2017 IEEE International Conference on Computer Vision (ICCV). (2017)