Is 3D Convolution with 5D Tensors Really Necessary for Video Analysis?

Habib Hajimolahoseini
Huawei Technologies Canada
habib.hajimolahoseini@huawei.com

Walid Ahmed
Huawei Technologies Canada
walid.ahmed1@huawei.com

Shuangyue Wen
Huawei Technologies Canada
austin.wen1@huawei.com

Yang Liu
Huawei Technologies Canada
yang.liu8@huawei.com

Abstract

In this paper, we present a comprehensive study and propose several novel techniques for implementing 3D convolutional blocks using 2D and/or 1D convolutions with only 4D and/or 3D tensors. Our motivation is that 3D convolutions with 5D tensors are computationally very expensive and they may not be supported by some of the edge devices used in real-time applications such as robots. The existing approaches mitigate this by splitting the 3D kernels into spatial and temporal domains, but they still use 3D convolutions with 5D tensors in their implementations. We resolve this issue by introducing some appropriate 4D/3D tensor reshaping as well as new combination techniques for spatial and temporal splits. The proposed implementation methods show significant improvement both in terms of efficiency and accuracy. The experimental results confirm that the proposed spatio-temporal processing structure outperforms the original model in terms of speed and accuracy using only 4D tensors with fewer parameters.

1 Introduction

During the past few years, 3D convolutional neural networks have become dominant in the area of video analysis, especially for action recognition [3, 7, 6, 38, 23, 26, 22, 41, 25, 40, 36, 15]. However, 3D convolution is computationally very expensive, which may cause problems in real-time applications. That is mostly because the common strategy in video processing models is to expand a well-known 2D image architecture into a 3D spatio-temporal model [8]. For example, C3D [32] and I3D [3] are the 3D versions of VGG-16 [28] and Inception-V1 [31], respectively. Although this type of architectures could reach the state-of-the-art in accuracy, it has been proved that this 2D to 3D expansion approach is not optimal in terms of computational costs [6].

For example, the 3D version of ResNet used in video processing uses around 27 times more mathematical operations than its 2D version used in image recognition [16, 38].

In order to increase the efficiency of deep learning models, different strategies have been proposed in the literature [1, 2, 18, 12, 10, 14, 13, 9, 11]. In video processing applications, early attempts were mostly focused on using 2D convolutions with some tricks in order to include the motion analysis as well. Two-stream networks are examples of this type of methods in which one stream is used for spatial processing while motion analysis is performed in the second stream using 2D convolutions [27, 37]. However, the computational costs added by the optical flow calculation in the motion stream prevent these methods from being efficient in real-time applications. On the other hand, there

Submitted to 4th NeurIPS Efficient Natural Language and Speech Processing Workshop (ENLSP-IV 2024).
Do not distribute.

are some other 2D based networks e.g. TSM [23] and TIN [26] that perform the motion analysis by shifting some of the features along the temporal dimension in order to provide the ability of information exchange between the adjacent frames.

Another straightforward approach is to split the network architecture by performing the 2D convolutions in some layers of the network for spatial processing while applying the 3D convolutions for spatio-temporal analysis in the rest of the model. In [39], it is shown that efficiency and accuracy could be improved by applying 2D convolutions to the early layers of the network to extract high-level semantic information from frames. The temporal representation learning is then performed at the top of the network by applying 3D convolutional layers to these high-level features. ECO architecture is one of the most famous networks in this category [42].

A similar strategy is to use different pathways for spatial and temporal analysis as proposed in the SlowFast architecture [7]. In that network, the spatial processing is performed in the Slow pathway using more features at a lower frame rate, while the temporal analysis is done in the Fast pathway using less features at a higher frame rate. They then extended their work to a single path architecture called X3D, in which a tiny 2D network is progressively expanded in different dimensions to a 3D architecture instead of just adding an extra temporal dimension to a 2D image based network [6].

On the other hand, instead of splitting the network into 2D and 3D layers or different pathways, another alternative solution is to factorize each 3D layer into spatial and temporal domains throughout the network. In this approach, every 3D convolutional block is decomposed into a spatial 2D convolution followed by a temporal 1D convolution. S3D [39], P3D [24] and R(2+1)D [35] are some of the well-known architectures in this category. As presented in R(2+1)D architecture, one benefit of this approach is that the network capacity could be doubled by adding an extra non-linear function in between the 2D and 1D convolutions while preserving the number of parameters [35]. This could lead to a higher accuracy and efficiency comparing to the original 3D convolutional networks.

Factorization of the 3D blocks can go even deeper by decomposing the kernels into a consecutive sequence of one-dimensional filters across all directions in order to improve the efficiency even more [19]. However this implies a strong assumption that the convolutional kernels are of rank-1 so that the matrix decomposition is reversible by cross production of all 1D components [21].

On the other hand, some networks take advantage of channel-wise separable convolutions or group convolutions. In CSN architecture for example, it is shown that separating channel and spatio-temporal interactions could improve both efficiency and accuracy at the same time as it acts as a regularization technique [34]. In that method, all 3D convolutional blocks are split into a point-wise ($1 \times 1 \times 1$) convolution for inter-channel processing and a depth-wise ($3 \times 3 \times 3$) convolution for spatio-temporal analysis inside each channel.

In video analysis, the input to the network is a 5D tensor with the following shape:

$$\text{Tensor Shape} = [B, T, X, Y, C]$$

in which B, T, X, Y and C represent batch size, number of frames, width, height and number of channels, respectively. In 3D convolution, although the kernels are assumed to be 3 dimensional, in reality the following 4D filter is applied to the input tensors:

$$\text{Filter Shape} = [t, w, h, c]$$

where t, w, h and c are the dimensions of filters in time, horizontal, vertical and channel domains. Note that in regular 3D convolutions, the kernel's channel dimension c is equal to the number of channels in the input tensor C ($c = C$). Therefore, all the features are collapsed into a single channel and this process is repeated until the desired number of output channels is achieved. On the other hand, in group convolutions, channels are divided into different groups and all features within a group are collapsed into a single channel. Finally in depth-wise convolution, no interaction between the channels is performed and so it is assumed that $c = 1$, which means that in each input channel, a 3D filter of shape $t \times w \times h$ is applied independently [5]. For the sake of simplicity in presentations, we omit the channel dimension of the filters and assume 3D kernels of shape $t \times w \times h$ throughout the rest of this paper.

Although in theory, the above-mentioned techniques simulate the 3D convolution using a combination of 2D and/or 1D convolutions, in their implementation codes they still use 3D convolutions with 5D tensors, but with specific parameters for spatial and temporal analysis. In other words, instead of

applying a 3D convolution with a $t \times h \times w$ kernel to the 5D input tensor of shape $B \times T \times X \times Y \times C$, 3D simulation approaches apply two different 3D convolutions to the 5D tensors, one with a $1 \times w \times h$ kernel and another one with a $t \times 1 \times 1$ kernel for spatial and temporal analysis, respectively. While this approach of modeling the 3D convolutions may work when training on powerful multipurpose systems, it may cause complications and limitations for some of the edge devices. Furthermore, in almost all of the existing methods, the spatial and temporal convolutions are applied in a sequential form in which, temporal analysis is applied to the output of the spatial analysis. However, other combination options e.g. parallel spatial and temporal processing and different ways of combining them e.g. by summation, concatenation, etc. are not fully studied.

In this paper, we study and propose some new alternative operations for modelling the 3D convolutions using either 2D convolutions with 4D tensors, 1D convolution with 3D tensors, or a combination of them. This is done by splitting the 4d filters used in the 3D convoluitons into spatial and temporal domains with appropriate reshaping and combinations of the tensors. For each method, different ways of combining spatial and temporal analysis i.e. sequential, parallel, summation, concatenation and etc.,are also explored in order to find the most efficient and accurate combination method. The proposed simulation techniques show significant improvement both in terms of efficiency and accuracy.

2 Proposed Method

A regular 3D convolutional layer and its 5D input/output shapes are shown in Fig.1. As shown in

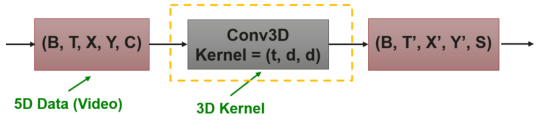

Figure 1: A regular 3D convolutional layer applied to 5D tensors.

this figure, it applies a 3D convolution operation to both special (X, Y) and temporal dimensions (T) at the same time to extract the spatio-temporal information. Note that in almost all of the video analysis models, spatial dimensions w and h of the kernels are the same. Therefore, we assume: $w = h = d$ for the sake of simplicity in representations. The output would have the shape of $B \times T' \times X' \times Y' \times S$, where T', X' and Y' are the new temporal and spatial dimensions and S is the number of output channels.

In order to avoid using 5D tensors, we first reshape the input data into 4D format by multiplying its first 2 dimensions as shown in Fig.2. Then, in order to avoid applying 3D operations, we replace the

Figure 2: Reshaping the input 5D tensor into 4D.

3D convolutional layer in Fig.2 with the proposed structure shown in 3.

As shown in Fig.3, we apply the spatial and temporal processing independent from each other in two parallel branches: spatial branch which analyses the data in its X and Y dimensions, and temporal branch which analyses the input in its temporal domain T.

2.1 Spatial Analysis

Each frame of the video can be considered as a static image with spatial 2D data. In the spatial analysis branch, the first 2D convolution applies the spatial analysis to each frame by multiplying the X and Y dimensions with a $d \times d$ kernel. This will generate a 4D tensor of size: $[B \times T, \frac{X}{s}, \frac{Y}{s}, S]$ in which the lowercase "s" is an integer representing the stride. After the spatial analysis is finished, in order to keep the tensor dimension under 4D, we now multiply the number of vertical and horizontal pixels of each frame by reshaping the tensor into: $[B, T, \frac{X}{s} \times \frac{Y}{s}, S]$. After that, a 1-by-1 convolution

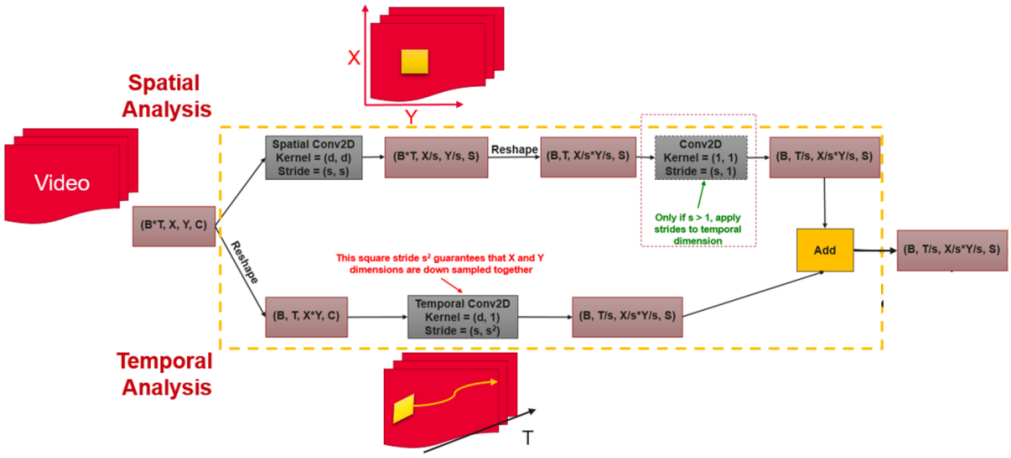

Figure 3: The proposed architecture which replaces the regular 3D convolutional layers.

with stride of s is applied to the temporal dimension (if necessary) in order to pool the temporal dimension by s. The output of the spatial branch will then have the following size: $[B, \frac{T}{s}, \frac{X}{s} \times \frac{Y}{s}, S]$.

2.2 Temporal Analysis

On the other hand, in the temporal analysis branch, we only analyze the relationship between each pixel of the images in consecutive frames. To do this, the input tensor is first reshaped by multiplying the vertical and horizontal pixels of the frames: $[B, T, X \times Y, C]$. Then, a 2D convolution with kernel $d \times 1$ and strides of (s, s^2) is applied to the 2nd and 3rd dimensions of the 4D tensor which performs both the temporal analysis and spatial striding at the same time. The reason behind using s^2 as our spatial stride is that the X and Y dimensions are multiplied and applying an s^2 stride to the 3rd dimension will shrink each of X and Y dimensions by a factor of s. Therefore, this square stride guarantees that the output of the temporal branch will have the exact same size as that of the spatial branch. This will make combining the output of two spatial and temporal branches very straightforward by simply adding them together. The resulting tensor generated by adding of two branches outputs will also have the same size as each of the branch outputs: $[B, \frac{T}{s}, \frac{X}{s} \times \frac{Y}{s}, S]$.

3 Experimental Results

We choose ECO-Lite architecture (with a minor modification) as the baseline in our experiments [42]. This architecture is composed of two parts: a set of 2D layers called 2D-Net for spatial analysis of individual frames, and a set of 3D layers called 3D-Net for spatio-temporal analysis of the feature representations learned from the 2D-Net. The first few layers of Inception-V3 [30] (originally BN-Inception [17] in the ECO paper) is used as the 2D-Net while the last few layers of 3D-Resnet18 [33] is adopted for the 3D-Net. The modified ECO-Lite architecture is presented in Table 1.

There are multiple reasons for choosing the ECO-Lite architecture in our experiments. First of all, it is simple for implementation and efficient during both training and inference. Second of all, almost half of the architecture consists of 2D convolutional layers from Inception-V3 which we do not need to change during our experiments with different 3D simulations techniques in the 3D-Net. This will provide us with a huge benefit in terms of training time as we can initialize the 2D-Net using the Inception-V3 pre-trained weights on a large image dataset e.g. ImageNet [4]. This will also enable us to transfer knowledge between our different 3D simulation experiments as we only change the 3D-Net during our experiments while 2D-Net stays untouched.

Experiments are performed on NVIDIA V100 GPUs. In the experiments, all 3D Convolutional layers in ECO-Lite architecture are replaced by one of the equivalent structures in literature that resemble the 3D convolution including: R(2+1)D [35], P3D-A, -B, and -C [24] and Rank-1 [21]. Two versions of the proposed structure are also implemented, in which the last block in Fig.3 (the yellow add

Layer Name	Output Size	Filters
2D-Net		
conv2D-1	121×121	$[3 \times 3, 32]$
conv2D-2	119×119	$[3 \times 3, 32]$
conv2D-3	119×119	$[3 \times 3, 64]$
pool1	59×59	$[3 \times 3]$
conv2D-4	59×59	$[1 \times 1, 80]$
conv2D-5	57×57	$[3 \times 3, 192]$
pool2	28×28	$[3 \times 3]$
inception (3a)	28×28	$[-256]$
Inception (3b)	28×28	$[-288]$
Inception (3c)	28×28	$[-96]$
3D-Net		
conv3D-1	$N \times 28 \times 28$	$\begin{bmatrix} 3 \times 3 \times 3, 128 \\ 3 \times 3 \times 3, 128 \end{bmatrix} \times 2$
conv3D-2	$\frac{N}{2} \times 14 \times 14$	$\begin{bmatrix} 3 \times 3 \times 3, 256 \\ 3 \times 3 \times 3, 256 \end{bmatrix} \times 2$
conv3D-3	$\frac{N}{4} \times 7 \times 7$	$\begin{bmatrix} 3 \times 3 \times 3, 512 \\ 3 \times 3 \times 3, 512 \end{bmatrix} \times 2$
pool3	$1 \times 1 \times 1$	$[\frac{N}{4} \times 7 \times 7]$
fc, softmax	$1 \times 1 \times 1$	[$1 \times 1 \times 1$, #classes]

Table 1: The modified ECO-Lite architecture, assuming that the input is a video clip with N frames of size 243×243 [42]

Method	#Params(M)	FPS	FLOPs(G)
Conv3D	33.7	773	35.6
R(2+1)D	33.7	764	36.9
Proposed-Cat	32.2	774	32.7
Proposed-Add	16.3	865	17.1
P3D-A	22.5	824	19.5
P3D-B	23.5	791	20.9
P3D-C	22.8	758	19.6
Rank-1	13.7	724	14.7

Table 2: Comparison of the efficiency of different structures resembling the Conv3D module

block) that combines the temporal and spatial branches are adding (Proposed-add) or concatenation (Proposed-cat) operations. Concatenation is done along the last dimension of the tensors (channels).

The number of training parameters, floating points operations per second (FLOPs) and inference speed in terms of frame per second (FPS) of all of the structures are also compared with that of the baseline Conv3D module in Table 2. As seen in this table, the proposed method with adding block (Prop-Add) has only 16.3 million parameters which is 51% less than the baseline (Conv3D). the FLOPs also drops by 51% and the inference speed improves by 12%, which is the highest speed-up among all other implementations.

In order to evaluate the performance of different structures, we use two datasets including Kinetics-400 [20] and UCF-101 [29]. The experimental results for the different structures on these datasets are shown in Table 3. In this table, the 3D layers of ECO-Lite architecture are trained from scratch while the 2D blocks are initialized using the ImageNet weights of Inception-V3. The performance results of pretraining on Kinetics dataset and then finetuning on UCF is also reported in Table 4. In this experiment, the model is first trained on Kinetics dataset for 50 epochs and then finetuned on UCF for 100 epochs.

Method	2D-Net Pre	Kinetics	UCF
Conv3D	ImageNet	55.63	72.85
R(2+1)D	ImageNet	57.91	69.00
Proposed-Cat	ImageNet	59.41	70.15
Proposed-Add	ImageNet	57.90	75.79
P3D-A	ImageNet	56.77	67.88
P3D-B	ImageNet	57.83	69.62
P3D-C	ImageNet	59.40	71.56
Rank-1	ImageNet	56.14	64.12

Table 3: Accuracy comparison when training the 3D layers from scratch on Kinetics-400 and UCF-101, using ImageNet weights for the 2D layers

Method	Pretrained	Top-1(%)
Conv3D	Kinetics	88.48
R(2+1)D	Kinetics	90.31
Proposed-Cat	Kinetics	91.21
Proposed-Add	Kinetics	90.62
P3D-A	Kinetics	90.44
P3D-B	Kinetics	90.31
P3D-C	Kinetics	91.01
Rank-1	Kinetics	86.41

Table 4: Accuracy comparison when training the model on Kinetics-400 for 50 epochs and then fine-tuning on UCF-101

As seen in these tables, the proposed technique is more efficient and even more accurate than the other methods, even the baseline 3D-CNN which uses 5D tensors. More specifically, the Proposed-Cat method has the highest accuracy on both Kinetics and UCF datasets in both cases, even better than the baseline. The possible reason could be the higher non-linearity caused by the new structures which increases the capacity of the networks. If the number of parameters is not an issue, the Proposed-Cat structure is the best choice among all of the techniques. The Proposed-Add shows the highest speed-up among all of the techniques although it does not have the highest accuracy. However, it still preserves the accuracy higher than the baseline. Therefore, it could be a good choice when the speed and efficiency is a priority. This fact is shown in Fig.4, in which the compression ratio is depicted as a function of the speed multiplied by accuracy improvement.

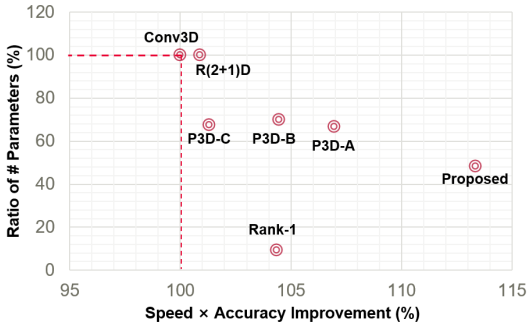

Figure 4: Number of parameters vs. speed and accuracy improvement for different methods. Note that the lower the number of parameters and the higher the speed and accuracy, the better the model.

4 Conclusion

In this work, we studied some techniques for implementing the 3D convolutional layers using 2D and/or 1D convolutions with only 4D and/or 3D tensors. The existing approaches reshapes the 5D tensors at the begigning of the models and analyses the data in two different branches including spatial and temporal domains. We performed this by introducing some appropriate 4D/3D tensor reshaping as well as new combination techniques for spatial and temporal splits. The proposed implementation methods show significant improvement both in terms of efficiency and accuracy. We have performed multiple experiments on both NVIDIA's GPUs as well as Huawei's Ascend AI accelerators. In summary we solved the existing problems with conventional 3D-CNN as follows:

- Appropriate reshaping techniques in the parallel structure of the proposed method allows us to use 4D tensors throughout the entire system instead of 5D
- Also, the 4D tensors in both branches will have the same shapes after processing which makes them more efficient to combine by adding
- Using only 2D kernels enables us to implement the spatial and temporal processing much more efficiently with significantly lower memory consumption, which are critical in real-time applications especially for edge devices in real-world applications.

References

[1] Walid Ahmed, Habib Hajimolahoseini, Austin Wen, and Yang Liu. Speeding up resnet architecture with layers targeted low rank decomposition. *arXiv preprint arXiv:2309.12412*, 2023.

[2] Foozhan Ataiefard, Walid Ahmed, Habib Hajimolahoseini, Saina Asani, Farnoosh Javadi, Mohammad Hassanpour, Omar Mohamed Awad, Austin Wen, Kangling Liu, and Yang Liu. Skipvit: Speeding up vision transformers with a token-level skip connection. *arXiv preprint arXiv:2401.15293*, 2024.

[3] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[5] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[6] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 203–213, 2020.

[7] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 6202–6211, 2019.

[8] H Hajimolahoseini, MR Taban, and HR Abutalebi. Improvement of extended kalman filter frequency tracker for nonstationary harmonic signals. In *2008 international symposium on telecommunications*, pages 592–597. IEEE, 2008.

[9] Habib Hajimolahoseini, Walid Ahmed, and Yang Liu. Training acceleration of low-rank decomposed networks using sequential freezing and rank quantization. *arXiv preprint arXiv:2309.03824*, 2023.

[10] Habib Hajimolahoseini, Walid Ahmed, and Yang Liu. Methods, systems, apparatuses, and computer-readable media for decomposing a layer in a neural network, June 27 2024. US Patent App. 18/087,877.

[11] Habib Hajimolahoseini, Omar Mohamed Awad, Walid Ahmed, Austin Wen, Saina Asani, Mohammad Hassanpour, Farnoosh Javadi, Mehdi Ahmadi, Foozhan Ataiefard, Kangling Liu, et al. Swiftlearn: A data-efficient training method of deep learning models using importance sampling. *arXiv preprint arXiv:2311.15134*, 2023.

[12] Habib Hajimolahoseini, Mohammad Hassanpour, Foozhan Ataiefard, Boxing Chen, and Yang Liu. Single parent family: A spectrum of family members from a single pre-trained foundation model. *arXiv preprint arXiv:2406.19995*, 2024.

[13] Habib Hajimolahoseini, Kaushal Kumar, and DENG Gordon. Methods, systems, and media for computer vision using 2d convolution of 4d video data tensors, April 20 2023. US Patent App. 17/502,588.

[14] Habib Hajimolahoseini, Mehdi Rezagholizadeh, Vahid Partovinia, Marzieh Tahaei, Omar Mohamed Awad, and Yang Liu. Compressing pre-trained language models using progressive low rank decomposition. *Advances in Neural Information Processing Systems*, 2021.

[15] Habib Hajimolahoseini, Mohammad Reza Taban, and Hamid Soltanian-Zadeh. Extended kalman filter frequency tracker for nonstationary harmonic signals. *Measurement*, 45(1):126–132, 2012.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[18] Farnoosh Javadi, Walid Ahmed, Habib Hajimolahoseini, Foozhan Ataiefard, Mohammad Hassanpour, Saina Asani, Austin Wen, Omar Mohamed Awad, Kangling Liu, and Yang Liu. Gqkva: Efficient pre-training of transformers by grouping queries, keys, and values. *arXiv preprint arXiv:2311.03426*, 2023.

[19] Jonghoon Jin, Aysegul Dundar, and Eugenio Culurciello. Flattened convolutional neural networks for feedforward acceleration. *arXiv preprint arXiv:1412.5474*, 2014.

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

[21] Hyein Kim, Jungho Yoon, Byeongseon Jeong, and Sukho Lee. Rank-1 convolutional neural network. *arXiv preprint arXiv:1808.04303*, 2018.

[22] Yan Li, Bin Ji, Xintian Shi, Jianguo Zhang, Bin Kang, and Limin Wang. Tea: Temporal excitation and aggregation for action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 909–918, 2020.

[23] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7083–7093, 2019.

[24] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *proceedings of the IEEE International Conference on Computer Vision*, pages 5533–5541, 2017.

[25] Michael S Ryoo, AJ Piergiovanni, Mingxing Tan, and Anelia Angelova. Assemblenet: Searching for multi-stream neural connectivity in video architectures. *arXiv preprint arXiv:1905.13209*, 2019.

[26] Hao Shao, Shengju Qian, and Yu Liu. Temporal interlacing network. *arXiv preprint arXiv:2001.06499*, 2020.

[27] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

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

[29] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[30] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016.

[31] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[32] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[33] Du Tran, Jamie Ray, Zheng Shou, Shih-Fu Chang, and Manohar Paluri. Convnet architecture search for spatiotemporal feature learning. *arXiv preprint arXiv:1708.05038*, 2017.

[34] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5552–5561, 2019.

[35] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.

[36] Gül Varol, Ivan Laptev, and Cordelia Schmid. Long-term temporal convolutions for action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1510–1517, 2017.

[37] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*, pages 20–36. Springer, 2016.

[38] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[39] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 305–321, 2018.

[40] Ceyuan Yang, Yinghao Xu, Jianping Shi, Bo Dai, and Bolei Zhou. Temporal pyramid network for action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 591–600, 2020.

[41] Shiwen Zhang, Sheng Guo, Weilin Huang, Matthew R Scott, and Limin Wang. V4d: 4d convolutional neural networks for video-level representation learning. *arXiv preprint arXiv:2002.07442*, 2020.

[42] Mohammadreza Zolfaghari, Kamaljeet Singh, and Thomas Brox. Eco: Efficient convolutional network for online video understanding. In *Proceedings of the European conference on computer vision (ECCV)*, pages 695–712, 2018.