VIDEO ACTION RECOGNITION VIA NEURAL ARCHITECTURE SEARCHING

Wei Peng[1] *Xiaopeng Hong*[2,1] *Guoying Zhao*[1,⋆]

[1]Center for Machine Vision and Signal Analysis, University of Oulu, Finland
[2] Xi'an Jiaotong University, Xi'an, P. R. China

ABSTRACT

Deep neural networks have achieved great success for video analysis and understanding. However, designing a high-performance neural architecture requires substantial efforts and expertise. In this paper, we make the first attempt to let algorithm automatically design neural networks for video action recognition tasks. Specifically, a spatio-temporal network is developed in a differentiable space modeled by a directed acyclic graph, thus a gradient-based strategy can be performed to search an optimal architecture. Nonetheless, it is computationally expensive, since the computational burden to evaluate each architecture candidate is still heavy. To alleviate this issue, we, for the video input, introduce a temporal segment approach to reduce the computational cost without losing global video information. For the architecture, we explore in an efficient search space by introducing pseudo 3D operators. Experiments show that, our architecture outperforms popular neural architectures, under the training from scratch protocol, on the challenging UCF101 dataset, surprisingly, with only around *one* percentage of parameters of its manual-design counterparts.

***Index Terms*—** Automated machine learning, neural architecture search, video action recognition

1. INTRODUCTION

Video action recognition [1], which is a hot topic of video analysis and understanding, has drawn considerable attention from both academia and industry, since it has great value to many potential applications, like behaviour analysis [2], security, and video affective computing [3]. On one hand, new and large-scale datasets, such as Kinetics [4], Something-Something [5], make great contribution to video action recognition. On the other hand, video analysis and understanding benefits a lot from the recent advance in deep neural networks, which have already been successfully applied to a large variety of tasks like object detection [6] and machine translation. However, designing a neural network structure with high performance is a hard work. It requires large amounts of time and efforts of human experts. Specifically, in the action recognition task, to get satisfying performance, the long-range spatial-temporal connections should be taken into consideration. It thus inevitably leads to complicated network architectures and expensive computational costs. Moreover, one has to feed such a network with much training data.

Recently, Neural Architecture Search (NAS) [7] has shown great superiority over manually designed neural architectures. In image classification, automatically designed architectures like NASNet [8], and AmoebaNet [9], outperform popular human-designed networks, such as VGG [10], and ResNet [11]. In semantic image segmentation, Auto-Deeplab [12] also performs well even without a pre-train procedure. Such promising progresses are mostly benefited from the current strong computational capability, the elaborate search space [13] as well as the efficiency of search strategies like reinforcement learning [7], evolutionary algorithms [9] and gradient methods [14]. However, involved tasks are limited at cases with low-dimensional inputs like text and images.

In this paper, we aim at realizing automatic neural architecture design for action recognition. Fueled by the promising progress of NAS, we propose to introduce NAS into the action recognition task with an efficient fashion. Firstly, a video processing scheme like in temporal segment network (TSN) [15] is adopted to capture global temporal information and reduces the computational cost. It also contributes to the data augmentation. Secondly, to characterize the spatial-temporal dynamics in videos, we provide an efficient solution to search 3D neural operators, by processing the spatial and temporal features separately in a search space with *pseudo 3D* [16] operators. Thirdly, we introduce a directed acyclic graph to model the search space and relax the discrete space into a continuous and a differentiate one [14]. It is thus allowed to achieve highly competitive performance with a small order of magnitude in computational resources against the previous ones with thousands of GPU days to search.

The contributions of this paper are three-fold: Firstly, to our best knowledge, this is the first attempt to automatically design a spatial-temporal neural network for video action recognition issues. Secondly, we design a continuous pseudo 3D neural network search space, where high-efficient operators can be explored with proxyless strategy. Finally, we evaluate the proposed method on UCF101 [17] dataset. Compared with popular action recognition neural architectures

⋆ Corresponding author.

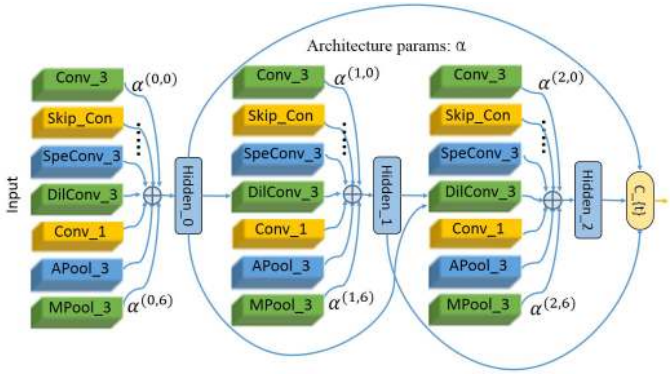

Fig. 1: Overview of the neural module to be searched. The architecture of the module is determined by different nodes (Hidden_i) connections and operators. There are three nodes in this module. Concatenating the outputs of them forms the module output. For each connection between nodes, there are eight operator candidates which are detailed at the bottom of Table 1. In a continuous search space, the contribution of each operator is parameterized by a weight $\alpha^{(k,j)}$. Note that, for interpretability, connections from previous two modules outputs are omitted in this figure.

without pre-training, our searched architecture gets the best performance with only around one percentage of parameters.

2. PROPOSED METHODS

In this section we detail the modularized architecture $\mathcal{C}$ to be searched and the search strategy. Here, we search for a single (2+1)D convolutional module unit and repeat it for multiple times to build a neural network. During the search procedure, we build a shallow networks, $\mathcal{N}(\mathcal{C})$, to explore in the search space.

For video action recognition task, a TSN-like scheme is introduced to deal with input clips with various lengths. It not only reduces computational costs but also helps in capturing global information and data augmentation. We uniformly segment the clip into N_s segments. In every segment, we randomly sample N_r frames, and regroup them with the original order to form a new clip. These clips of a fixed length of $(N_s \times N_r)$ are then used as the input to our model.

2.1. Neural module architecture

As shown in Fig. 1, we build a directed acylic graph to model the neural module unit $\mathcal{C}$. Assume $\mathcal{C}$ of n nodes is the fundamental unit in our neural network for video action recognition. Then the search procedure is to find out the connections between these n nodes and their corresponding transform functions from a set of candidate operators $\mathcal{O}$ to optimize the performance.

Connections. Here, connections work as the data flow. For each node, which is the "Hidden_i" in Fig. 1, we take all the outputs of nodes before it as its inputs. Besides, from the network-level perspective, the outputs from two previous modules are also taken as its inputs (not shown in Fig. 1). As a result, there will be totally $M = \frac{(n+1)(n+2)}{2} - 1$ connections inside a module of n nodes. For the output of each node, element-wise addition is conducted during its different inputs. Finally, all the outputs of these nodes are concatenated as the module output $\mathcal{C}\_\{t\}$.

Table 1: Symbols and their meaning in the text and figures.

Symbol	Meaning
$\mathcal{C}$	A neural module.
$\mathcal{C}\_\{t\}$	The output of the t-th module in a network.
Hidden_i	The hidden output of the i-th node in a module.
$\mathcal{N}(\mathcal{C})$	A neural network build on $\mathcal{C}$.
Θ	The weights within $\mathcal{N}(\mathcal{C})$.
M	The number of connections in a module.
n	The number of nodes in a module.
$\mathcal{O}$	A set of operators.
$\alpha \in \mathbb{R}^{\mathcal{M}\times\|\mathcal{O}\|}$	The network architecture parameters.
$\alpha^{(k,j)}$	The j-th operator of the k-th connection.
Conv_1	RELU-CONV-BN block with kernel size 1.
Conv_3	RELU-CONV-BN block with kernel size 3.
SpeConv_3	Separable convolution filters with kernel size 3.
DilConv_3	Dilated convolution filters with kernel size 3.
MPool_3	Max pooling with kernel size 3.
APool_3	Average pooling with kernel size 3.
Skip_Con	Skip connection between Nodes.
Zero	No connection between Nodes(Not show in figure).

Operators. All the connections are filled with one or more operators from $\mathcal{O}$. Here, we decouple every operator into a 2D spatial one and a 1D temporal one since the (2+1)D convolution filters are with less parameters compared to the 3D ones but could get superior performance. In this paper, the initial set of operators in $\mathcal{O}$ consist of eight operator candidates, which are listed in the eight bottom rows of Table 1. Here, max or average pooling layers and RELU-CONV-BN blocks are prevalent in modern manually designed networks. 'SpeConv_3' is implemented by three 1D filters. 'DilConv_3' has a spatial perspective filed of 5×5. For computational consideration, we set all the operators with a relative small kernel size.

To find the best operators from $\mathcal{O}$, like [14], we introduce all of them into each connection and model them with the architecture parameters α. Then for each connection, element-wise addition is conducted between the outputs of each weighted operator. Therefore, search processing is reduced to find the best architecture parameters α which maximizes the performance on the validation data. At the end of this search, a discrete architecture can be got by functions like softmax. As a result, the search strategy is to adjust the contribution of each operator at each iteration step and find out the network architecture parameter $\alpha = \{\alpha^{(k,j)}\}$, for $k = 0, ..., M-1, j = 0, .., |\mathcal{O}| - 1$, which minimizes the task loss, as formulated as follows:

$$\alpha^* = \arg\min_{\alpha} \mathcal{L}_{valid}(\Theta(\alpha), \alpha). \quad (1)$$

Here, we introduce the parameter sharing scheme in $\mathcal{N}(\mathcal{C})$, thus Θ is the weights within the network, which is shared by each architecture candidate. $\mathcal{L}_{valid}$ is a loss function on the validation set. To better evaluate the architectures, Θ will also be updated on the training data at each iteration. Let $\mathcal{L}_{train}$ denote the loss function for training. Obviously the losses are determined by both the module architecture α and network weights Θ. Solving Eq. (1) means to find the optimal α, where the weights Θ minimize $\mathcal{L}_{train}$. This bilevel problem is hard since they are interdependent, more precisely, nested. We will discuss it in the next section.

2.2. Search Strategy

In this subsection, we elaborate our searching strategy to optimize the neural network architecture.

Though the search space is relaxed into a continuous and differentiate one, the optimization of Eq. (1) is still challenging or even infeasible. The main cause is that for any change of the architecture α, one has to recompute $\Theta(\alpha)$ by minimizing $\mathcal{L}_{train}$. To alleviate such difficulty, an alternating iteration approximation is performed. More concretely, given a search space, there are two alternative steps to conduct this searching process: Firstly, updating the network weights Θ by minimizing $\mathcal{L}_{train}$, where the architecture is fixed. Secondly, updating the network architecture α by minimizing $\mathcal{L}_{valid}$, by fixing the weights of the network. On this basis, Θ and α are optimized by alternating between gradient descent steps in the network weights and the architecture parameters in an iterative manner. In fact, this task is very similar with the few-shot meta-learning task [18], which is to adapt a model to a new task using a few data and training process. From this perspective, neural architecture search can be treated as a kind of meta-learning, in which NAS is to transfer the network trained on the training data to adapt the validation data by fine-tuning the architecture. As a result, a surrogate function is employed for $\Theta(\alpha)$ in Eq. (1). That is

$$\alpha^* = \arg\min_{\alpha} \mathcal{L}_{valid}(\Theta^{'}, \alpha), \tag{2}$$

$$\Theta^{'} = \Theta - \epsilon \bigtriangledown_{\Theta} \mathcal{L}_{train}(\Theta, \alpha). \tag{3}$$

Here,the ϵ is a very small value in the search step. By using the gradient descent method with respect to the architecture parameters α, we can find a solution of the architecture. That is

$$\alpha = \alpha - \gamma \bigtriangledown_{\alpha}. \tag{4}$$

Here γ is the learning rate, and $\bigtriangledown_{\alpha}$ is calculated by Eq. (5) :

$$\begin{aligned} \bigtriangledown_{\alpha} = & \bigtriangledown_{\alpha} \mathcal{L}_{valid}(\Theta^{'}, \alpha) - \\ & \epsilon \bigtriangledown^2_{\alpha,\Theta} \mathcal{L}_{train}(\Theta, \alpha) \bigtriangledown_{\Theta^{'}} \mathcal{L}_{valid}(\Theta^{'}, \alpha). \end{aligned} \tag{5}$$

Eqs. (4) and (5) give us a solution to optimize the architecture by SGD. However, it is not computational efficient since there is a second-order derivative, in which the matrix-vector products are computational expensive. Fortunately, Hessian-vector products can be used here to approximate the second-order derivative [14], so that the computational complexity can be significantly reduced. Besides, it is worth to mention that when $\epsilon \rightarrow 0$, Eq. (3) indicates that $\Theta^{'} \rightarrow \Theta$, which means a second-order derivative degenerates to a first-order one. Therefore, as a special case of Eq. (5), one can drop out the one-step unrolled weights operation in Eq. (3) for computational consideration, which will reduce about one-third of the computational cost, in case that a certain loss in accuracy is tolerant.

3. EXPERIMENTS

In this section, we study the proposed automatic method of designing action recognition network to demonstrate its advantages over other famous action recognition architectures, e.g., 3D-ResNet [19], C3D network [20], and STC-ResNet [21]. We evaluate our algorithm on the challenging action recognition dataset UCF101, which is a trimmed dataset containing 13320 video clips of 101 classes, with the training from scratch protocol. The training from scratch protocol is commonly used and of great value for issues of limited data or of limited computational resources. Other architectures, such as I3D [22] and TSN, which are either two-stream methods with extra input modality or pre-trained on another larger datasets, are not taken into consideration, owing to the limitation of computational resources.

Implementation. There are two stages to design an automatic action recognition network: training network and updating architecture, which are alternatively performed. We search the network on the *split 1* of UCF101. During the searching processing, we build the network $\mathcal{N}(\mathcal{C})$ with three layers and each layer is constructed by an identical module $\mathcal{C}$, which is the one to be searched. We initialize the first module with four channels and set the number of hidden nodes n to four as a trade-off between accuracy and efficiency. For feature inputs with different resolutions, we add an extra feature map adaption layer before the module. Once there need a feature map resolution reduction, the output channels are doubled correspondingly. The network takes eight RGB frames with size 112×112 as inputs. So for the video input preprocessing, we set the segment number $N_s = 4$ and number of random sample $N_r = 2$. Other data augmentation methods like normalization, random horizontal flip, and random crop are employed. For the objective function, a cross-entropy loss is adopted for the classification task on UCF101. For searching the architecture, the whole search procedure is 50 epochs, thus we choose a relative big learning rate. We also decrease the learning rate at each iteration. What makes this search procedure more efficient is that the search model is not thrown

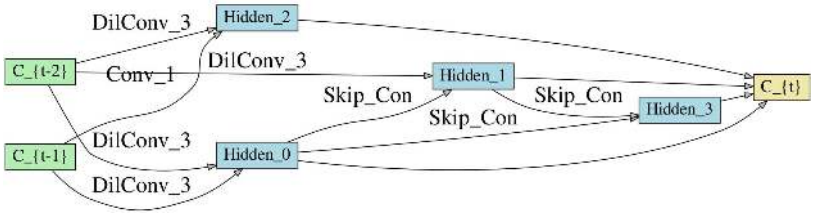

Fig. 2: Neural architecture searched by our method. Here, we keep two connections for each node according to α. The outputs of two previous modules $\mathcal{C}\_\{t-1\}$ and $\mathcal{C}\_\{t-2\}$ are taken as inputs for current module. All the outputs of these four nodes are concatenated to form the final output $\mathcal{C}\_\{t\}$.

away after each validation.

Once we finish the search procedure, we will build and train a final network from scratch. As aforementioned, we construct the network with the returned search result α, as illustrated in Fig. 1. Here, for each node, we keep the top-2 connections according to α, and then choose the most important operator for each connection based on their contributions. Here, a softmax function is adopted on α to evaluate the contribution of each operator. Then a network can be built by the discrete architecture, which is shown in Fig. 2. For the depth of the network, we investigate different repeated times, and find that the network performs best when the depth is set to six. Every two layers, we reduce the feature map resolution by a factor of two along width and height side. Accordingly, we double the channels of feature maps. The whole training process is set as follows: We initialize the first module with eight channels and the training iteration is 600 epochs, the learning rate is 0.025, and deceased by a cosine function w.r.t iteration steps. The network is trained with a momentum of 0.9, a weight decay 0.0003, and a mini-batch of 72. For the testing processing, our model predicts a score for each video clip without additional aggregation. For each input clip, we sample eight frames, apply only center crop on the inputs.

Result. When the 50 search iteration finished, we choose the best architecture according to their performance. We perform the search procedure on a single Nvidia V100 GPU. It takes about 25 hours on the UCF101 to finish this searching process. Searched architecture is shown in Fig. 2. It indicates that the the architecture prefers to the operator with bigger perspective field.

After getting the searched result, we build a network with six layers by repeating this neural module unit. Then, we train on the training data for 600 epochs and test. Table 2 shows the result of action recognition on UCF101 compared with 3D-convNet, 3D-ResNet 18, 3D-ResNet 101, and different kinds of STC-ResNet networks. The result shows that our approach achieves a better accuracy with much fewer parameters than any other models in this table. For instance, 3D-ConvNet, which is a very commonly used architecture, is one hundred times bigger than our network in terms of parameter size. Nevertheless, our model been searched outperforms it by 7% in terms of recognition accuracy.

Table 2: Comparison with manually designed 3D networks training from scratch on UCF101 *split 1*. As the benchmark codes are not available, we can not get the precise model size of STC-ResNet, though it is clearly larger than ResNet with the same blocks.

Architectures	#params	model size	Accuracy
3D-ResNet 18 [19]	33.2M	252M	42.4%
3D-ResNet 101 [19]	100M +	652M	46.7%
3D-ConvNet [20]	79M	305M	51.6%
STC-ResNet 18 [21]	33.2M +	-	42.8%
STC-ResNet 50 [21]	92M +	-	46.2%
STC-ResNet 101 [21]	100M +	-	47.9%
Ours	**0.67M**	**7.32M**	**58.6%**

4. CONCLUSION

In this paper, we perform neural architecture search for the action recognition task for the first time. Specifically, we model the neural network by a directed acyclic graph and efficiently search a spatial-temporal neural architecture in a continuous search space. We demonstrate that our method outperforms other popular models under the training from scratch protocol, with a surprisingly smaller model size. More concretely, our method improves the accuracy with an over 10% increase on the UCF101 dataset with approximately *one percentage* of the size of famous models like 3D-convNet and 3D-ResNet. In future work, we plan to apply the proposed method to other computer vision tasks.

5. ACKNOWLEDGEMENTS

This work was supported by the Academy of Finland ICT 2023 project (Grant No. 313600), Tekes Fidipro program (Grant No. 1849/31/2015) and Business Finland project (Grant No. 3116/31/2017), Infotech Oulu, and the National Natural Science Foundation of China (Grants No. 61772419). As well, the authors wish to acknowledge CSC-IT Center for Science, Finland, for computational resources.

6. REFERENCES

[1] Q. Le, W. Zou, S. Yeung, and A. Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 3361–3368.

[2] M. Keenan and C. Nikopoulos, *Video modelling and behaviour analysis: A guide for teaching social skills to children with autism*, Jessica Kingsley Publishers, 2006.

[3] W. Peng, X. Hong, Y. Xu, and G. Zhao, "A boost in revealing subtle facial expressions: A consolidated eulerian framework," *arXiv preprint arXiv:1901.07765*, 2019.

[4] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al., "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.

[5] R. Goyal, S. Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, et al., "The something something video database for learning and evaluating visual common sense," in *The IEEE International Conference on Computer Vision (ICCV)*, 2017, vol. 1, p. 3.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[7] B. Zoph and Q. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[8] B. Zoph, V. Vasudevan, J. Shlens, and Q. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[9] E.n Real, A. Aggarwal, Y. Huang, and Q. Le, "Regularized evolution for image classifier architecture search," *arXiv preprint arXiv:1802.01548*, 2018.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[12] C. Liu, L. Chen, F. Schroff, H. Adam, W. Hua, A. Yuille, and Li F., "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation," *arXiv preprint arXiv:1901.02985*, 2019.

[13] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean, "Efficient neural architecture search via parameter sharing," *arXiv preprint arXiv:1802.03268*, 2018.

[14] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[15] Limin W., Yuanjun X., Zhe W., Yu Q., Dahua L., Xiaoou T., and Luc V., "Temporal segment networks: Towards good practices for deep action recognition," in *ECCV*, 2016.

[16] Du T., H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6450–6459.

[17] K. Soomro, A. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *Computer Science*, 2012.

[18] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *arXiv preprint arXiv:1703.03400*, 2017.

[19] K. Hara, H. Kataoka, and Y. Satoh, "Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA*, 2018, pp. 18–22.

[20] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4489–4497.

[21] A. Diba, M. Fayyaz, V. Sharma, M. Arzani, R. Yousefzadeh, J. Gall, and L. Van Gool, "Spatio-temporal channel correlation networks for action classification," *arXiv preprint arXiv:1806.07754*, 2018.

[22] Joao Carreira and Andrew Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.