Video Action Recognition with Neural Architecture Search

Yuanding Zhou 972498829@MAIL.DLUT.EDU.CN
Dalian University of Technology, Dalian, Liaoning Province, 116024, CN

Baopu Li* BAOPULI@BAIDU.COM
Baidu Research, Sunnyvale, CA 94089, USA

Zhihui Wang* ZHWANG@DLUT.EDU.CN
Haojie Li HJLI@DLUT.EDU.CN
Dalian University of Technology, Dalian, Liaoning Province, 116024, CN
** Corresponding Author*

Editors: Vineeth N Balasubramanian and Ivor Tsang

Abstract

Recently, deep convolutional neural networks have been widely used in the field of video action recognition. Current approaches tend to concentrate on the structure design for different backbone networks, but what kind of network structures can process video both effectively and quickly still remains to be solved despite the encouraging progress. With the help of neural architecture search (NAS), we search for three hyperparameters in the video processing network, which are the number of frames, the number of layers per residual stage and the channel number for all layers. We relax the entire search space into a continuous search space, and search for a set of network architectures that balance accuracy and computational efficiency by considering accuracy as the primary optimization goal and computational complexity as the secondary optimization goal. We conduct experiments on UCF101 and Kinetics400 datasets, validating new state-of-the-art results of the proposed NAS based scheme for video action recognition.

Keywords: Video Action Recognition, Neural Architecture Search

1. Introduction

The purpose of video action recognition is to classify people's action based on their behaviors in a video. This task is essentially a multi-category classification problem with video as input and the output is an action label. In recent years, deep convolutional neural networks have been widely used in video action recognition tasks. Two-stream network Simonyan and Zisserman (2014) uses a pretrained 2D network to process images, and then adds temporal information for fusion. I3D Jiang et al. (2017) directly applies 3D convolution to model the entire video. These two kinds of approaches have shown its advantages with remarkable results.

However, the question of what kind of network can better represent video information is still a challenging problem that has not been fully solved. Towards this direction, X3D Feichtenhofer (2020) applies six scalable dimensions for the network structure, and gets a network based on the way of expansion, which achieves the best results so far on the Kinetics dataset Kay et al. (2017). This task, however, can be further solved with neural network architecture search(NAS) since NAS turns to be a competitive solution for better architecture design for many problems in the field of computer vision. Generally speaking, NAS can automatically design a more effective network framework than hand-crafted one, and it has come a long way in the field of image classification Zoph and Le (2017). Then, it finds its wide applications in other related domains such as object detection Chen et al. (2019), semantic segmentation Zhang et al. (2020), image super resolutionWu et al. (2021), depth estimation Chen et al. (2021) and so on. However, NAS is still rarely applied in the video domain. One work Piergiovanni et al. (2019a) utilizes evolutionary algorithm based methods, but its searching process tends to be very long.

Inspired by the great sucess of NAS in other computer vision domains, we propose a novel NAS scheme for video action recognition. Specifically, three of the six dimensions proposed in X3D are selected and integrated into our search space. As shown in the Table 1 and Table 2 , the search space in this paper is a 3D ResNet He et al. (2016) with 33 convolution layers, where the number of channels used in each layer, the content of operations and the number of operation stacks are all obtained by searching. The network built on the whole search space can be understood as a Supernet, and each choice of network structure is a subset of the Supernet. We divide the whole network into several stages, and each stage is further divided into two parts: head layer and stack layer. The head layer is responsible for searching the number of channels used in each stage. It will select candidate channels in each stage, and transforms the feature input in the previous stage into the feature of the chosen number of channels and downsamples them. The features processed by head layer will then be passed into a certain number of stack layer. Stack layer first searches for the appropriate number of stacking layers from the pre-designed maximum number of layers, and then selects the specific operation in these layers. Among these operations, we change the network input of different frames through dilation, so that the network can focus on different temporal resolution without losing frames.

After the search space is designed, we relax it to a continuous search space so that we can update the parameters of the Supernet by gradient back propagation just like optimizing a neural network. We consider accuracy as the primary goal of network architecture optimization and computational complexity as the secondary optimization goal. The optimal path is searched from the Supernet and our final network structure is constructed by the Viterbi algorithm Viterbi (1967). The final results are obtained by retraining on this network structure. It is found that the new video recognition network searched by the proposed novel NAS scheme yields better recognition results compared to state-of-the-art (SOTA) methods.

Our contributions can be summarized as follows:

- Considering the needs of both accuracy and efficiency, we design a novel feasible search space that consists of the channel number, convolution type and stack depth for the problem of video action recognition.
- We present a practical search algorithm through the relaxation, yielding much more efficient search speed in the large search space.
- Extensive results demonstrate that our method yields new state-of-the-art results on Kinetics400 and UCF101.

2. Related Work

2.1. Video Action Recognition

With the development of deep learning (DL), many wonderful video action recognition methods have emerged. The classic two-stream network Simonyan and Zisserman (2014) divides the video into spatial image information and temporal information. There are many remarkable methods like ResNet He et al. (2016) or Inception Szegedy et al. (2015) regarding the extraction of image features. The spatial part carries information about the background and the target subject depicted by the video. For the temporal part, the motion information of each frame is extracted by the optical flow, and the classification results are generated jointly after fusing the two parts of information. However, due to the high computational cost of optical flow, the extraction of temporal information by this method is gradually replaced by other methods. Slowfast Feichtenhofer et al. (2019) uses inputs with different frame rates, the slow path learns spatial information with few frames and the fast path learns temporal information with a large number of frames, and then combines it with non-local network to model the relationship between frames from a global perspective and obtains very good results. 2D convolution can model image features, while video has a temporal dimension information in addition to image information, so another idea is based on 3D convolution, such as C3D Tran et al. (2015), which directly models the information of the whole video. This type of method no longer requires the fusion strategy of two-stream networks and achieves relatively higher accuracy, but the computational cost of this type of method is also high. So there are many methods between 2D and 3D dedicated to reducing the computational complexity of 3D convolution, such as P3D Wei et al. (2019), R(2+1)D Tran et al. (2018), S3D Xie et al. (2018). The basic block of this paper uses dilation convolution in the temporal dimension to simulate different frame numbers of inputs. The number of channels of the ResNet network in our work is obtained by architectural search, which is also different from any previous network structure.

2.2. Neural Architecture Search

Neural Architecture Search (NAS) aims to enable the computer automatically search for a better network structure based on the given data and constraints. The architecture search can be divided into three parts, which are the search space, the search strategy, and evaluation. The search space defines which hyperparameters need to be searched. The performance evaluation strategy can reflect the merit of network structure in the search space. The search strategy is the way to efficiently find the best results. With the development of architectural search, many promising search strategies have emerged, the authors in Baker et al. (2017) and Zoph and Le (2017) have used reinforcement learning to search networks. In Real et al. (2017), evolutionary algorithms were introduced to solve the search problem. In the follow-up paper Real et al. (2019), AmoebaNet was further proposed to make the selection in evolution favor younger models. Both of the previously mentioned methods treat the search space as a discrete space. If the search space can be made continuous and the objective function can be differentiated, then the search operation can be performed more efficiently based on gradient information. DARTS Liu et al. (2019) designs a continuous search space based on such an idea, which drastically reduces the search time. The search space used in this paper is a continuous search space, but our search space is different from the above methods.

NAS has achieved highly competitive performance in image classification task(Zoph et al. (2018); Liu et al. (2018)) and other related tasks. However, there are not many methods to combine video action recognition with NAS. The reason is that NAS has high requirements for computation resource, and the search efficiency of early methods is relatively low, so there are few methods to combine NAS with video tasks. The three classic works of EvaNAS Piergiovanni et al. (2019b), AssembeNet Ryoo et al. (2020) and tiny video network Piergiovanni et al. (2019a), all of which use evolutionary algorithms as search strategy. Peng et al. (2019) combines DARTS directly with the video task, but results show that such an approach lacks the integration of temporal information and the final results are not good.

3. Method

In this section, we will introduce the method of this paper in detail. According to the six scalable architecture parameters proposed in X3D Feichtenhofer (2020) , we choose three of them to form our search space, namely temporary resolution, number of layers per residual stage (bottleneck width) and the channel number for all layers (channel). In this paper, the network structure is composed of several stages. The first stage is responsible for receiving input and transforming it into the basic feature map, and it is not involved in the search process. The following stages includes one head layer and a certain number of stack layers. The head layer is responsible for searching the number of channels in each stage, and stack layer is responsible for searching the bottleneck width and the temporal resolution. The search space considered in this work is shown in Table 1 and Table 2. At the end of this section, the search process will be described, and the searched final network structure is shown in Table 3.

3.1. Head Layer

As shown in Table 1, each stage in the search space has one head layer. The head layer of each stage has a predefined number of candidate channels to choose from. The search process is to learn what channel number of head layer that is more suitable for this stage. After the head layer of each stage is processed, it will pass through a certain number of stack layers and then each candidate head layer of the next stage is passed in to continue the search. The part about stack layers will be introduced in sec 3.2. We assume that the head layer h_i connections to m subsequent head layers, then the path between h_i and the head layer h_j from the next stage can be represented by a parameter β_{ij}. Just like DARTS Liu et al. (2019), we relax the head layer connections as a continuous representation. We define it as the sum of the output of all candidate head layers, and the weight of each path is computed by softmax. It can be expressed as follows:

$$p_{ij} = \frac{exp(\beta_{ij})}{\sum_{k=1}^{m} exp(\beta_{ik})} \tag{1}$$

After searching the most suitable number of channels for this stage, the head layer will down-sample the input feature map to obtain the features of larger receptive field and send them to the next stage.

Stage 2	head layer	candidate channels : 48
		candidate channels : 56
		candidate channels : 64
Stage 3	head layer	candidate channels : 72
		candidate channels : 96
		candidate channels : 112
	max stack layer operation number : 5	
Stage 4	head layer	candidate channels : 128
		candidate channels : 144
		candidate channels : 160
		candidate channels : 176
	max stack layer operation number : 10	
Stage 5	head layer	candidate channels : 192
		candidate channels : 208
		candidate channels : 224
	max stack layer operation number : 5	
Stage 6	head layer	candidate channels : 240
		candidate channels : 256
		candidate channels : 272
		candidate channels : 288
	max stack layer operation number : 5	
Stage 7	head layer	candidate channels : 480
		candidate channels : 496
		candidate channels : 512
	max stack layer operation number : 1	

Table 1: The search space of this work. The head layer of each stage can select one of the following three or four candidate channels as the channel number of this stage. The stack layer of each stage will first select one of the numbers less than max stack layer operation number as the number of stack layers in this stage. Each stack layer then selects a specific operation, which will be described in the Table 2 of stack layer section.

3.2. Stack Layer

In each stage of the search space, after selecting the appropriate number of channels through the head layer and completing the down-sampling operation, a certain number of stack layers will pass through. We define four operation types that stack layer can choose in advance, which are basic operation with temporary dilation of 1, 2, 4 and skip connection. The basic operation is shown in the Figure. 1, and the search space of all these operations in stack layers is illustrated in Table 2. Just like the basic operation of R(2+1)D Tran et al. (2018), we decompose the original 3D convolution of $T \times H \times W$ into 2D convolution $1 \times H \times W$ in spatial dimension and 1D convolution $T \times 1 \times 1$ in temporal dimension, which reduces the calculation cost. The basic operation of this paper also adds 1D dilation convolution in temporal dimension, using different filters allows temporal convolution to focus on different temporal resolution without losing temporal granularity. We also consider the structure of R(2+1)D, when the dilation is 1, the basic operation is the same as that of R(2+1)D. In addition to selecting the type of operation in each stack layer, this paper also searches the number of stack layer stacks. We have designed the maximum number of stack layers in each stage in advance. When this stage does not need so many basic operations, the network can choose skip operation. The output of the previous stack layer will not be processed and will be directly transferred to the next stack layer.

	Stack Layer Operation Choice	Receptive Field with Kernel Size 3
1	Basic Operation with dilation 1	3
2	Basic Operation with dilation 2	5
3	Basic Operation with dilation 4	7
4	Skip	without convolution

Table 2: The four operations that can be chosen for the stack layer.

The search process of stack layer is similar to that of head layer. Let O be the set of candidate operations. We use an architecture parameter α_o to represent the candidate operation $o \epsilon O$ in stack layer S. We relax the stack layer by defining it as a weighted sum of outputs from all candidate operations. The architecture coefficient of the operation is computed as a softmax of architecture parameters over all operations in the stack layer. It can be expressed as follows:

$$w_o^s = \frac{exp(\alpha_o^s)}{\sum_{o' \epsilon O} exp(\alpha_{o'}^s)} \tag{2}$$

The output of the stack layer S can be expressed as follows:

$$x_{s+1} = \sum_{o \epsilon O} w_o^s \cdot o(x_s) \tag{3}$$

3.3. Architecture Search

The size of the search space is about 5.84×10^{18}, and we can search via back propagation in the relaxed search space for a better search efficiency. At the beginning of the search, the weight of each path in the Supernet is not trained. The network architecture tends to choose a faster convergence architecture, which makes the network narrow. In order to solve this problem, we first train a certain number of epochs so that each path in Supernet has an initial weight. The next part is just like what DARTS Liu et al. (2019) did. In each epoch, in the first step, we optimize the network weight w with a formula of $\nabla_w L_{train}(w, \alpha, \beta)$ on the training set. In the second step, we fix the network weight and optimize the network architecture parameters α and β by applying formula $\nabla_{\alpha,\beta} L_{val}(w, \alpha, \beta)$ on the validate set. We repeat this alternative process until the network architecture obtained by the search basically does not change.

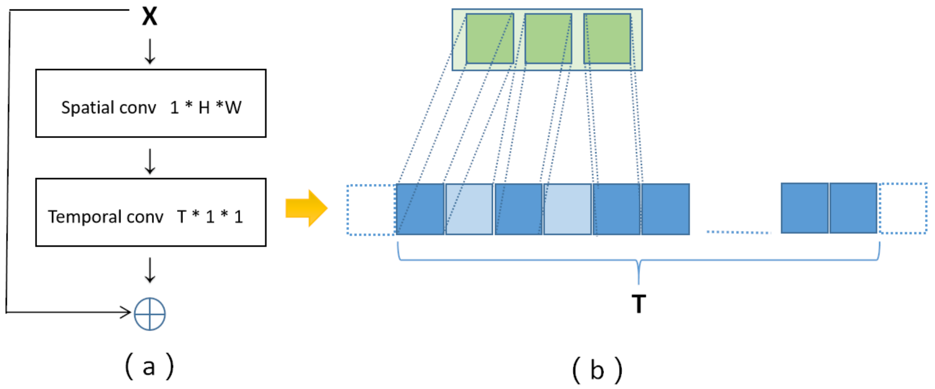

Figure 1: The basic operation of stack layer. In (a), the traditional 3D convolution is decomposed into a 2D spatial convolution for the height H and width W of the picture and a 1D temporal convolution for the time dimension T. (b)shows effects of the optional temporal convolution basic operation with dilation 2. Green block is the convolution kernel, and blue block is the feature T of the temporal dimension in the feature map.

In the search process, we take the accuracy as the primary evaluation goal, and we also take the FLOPs (floating-point operations) as the secondary evaluation goal, so that the obtained network architecture takes into account both the accuracy and the computational complexity in terms of FLOPs. For each stage G_i, the computational complexity can be described as :

$$c^i = c_s^i + \sum_{j=i+1}^{i+m} p_{ij} \cdot (c_h^{ij} + c_s^j) \tag{4}$$

where c_s^i means the cost (FLOPs) of stack layers from the G_i stage, and m denotes the number of candidate head layers on the subsequent stage, p_{ij} denotes the path probability between stage G_i and G_j , and c_h^{ij} denotes the cost (FLOPs) of the head layer in stage G_j which processes the data from stage G_i. Then we can use a recursive approach to calculate the computational complexity (FLOPs) of the entire network structure. The final loss function can be described as

$$L_{final} = L_{cross-entropy} + \lambda \cdot log_k(c) \tag{5}$$

where λ and k are the hyper-parameters to control the magnitude of the model cost, and c is the total FLOPs of the network.

In the selection of stack layer, we use the method of argmax to choose the maximum value of all operations. While in the connection of head layer, Viterbi algorithm Viterbi (1967) is taken to choose the connection path with the maximum sum of weights to generate the final network architecture diagram. The Viterbi algorithm is widely used to find the most probable path between hidden states. For the proposed NAS search algorithm, there are some possible paths to pass between each stage, and there are corresponding weights on each path. What we need in the end is the path with the largest weight, and the Viterbi algorithm uses the idea of recursion to solve this kind of problems. Starting from the first layer, the n paths with the largest weights among the n nodes in this and the next layer are calculated and retained each time, which greatly reduces the computation by discarding the paths that are not likely to be the greatest weight in advance. Since there are too many possible paths for our method, the Viterbi algorithm is chosen to find the path with the highest weight more efficiently than the greedy algorithm Faigle (1979) which X3D used.

	input : 3 * 16 * 224 * 224	
Stage 1	head layer	channels : 32
Stage 2	head layer	channels : 64
Stage 3	head layer	channels : 96
	stack layer number : 0	
Stage 4	head layer	channels : 160
	stack layer number : 6	Basic Operation with dilation 4
		Basic Operation with dilation 4
		Basic Operation with dilation 4
		Basic Operation with dilation 1
		Basic Operation with dilation 1
		Basic Operation with dilation 4
Stage 5	head layer	channels : 224
	stack layer number : 3	Basic Operation with dilation 2
		Basic Operation with dilation 4
		Basic Operation with dilation 4
Stage 6	head layer	channels : 288
	stack layer number : 4	Basic Operation with dilation 2
		Basic Operation with dilation 2
		Basic Operation with dilation 2
		Basic Operation with dilation 1
Stage 7	head layer	channels : 512
	stack layer number : 1	Basic Operation with dilation 2
Global Average Pooling and Classifier		

Table 3: The network architecture obtained from the NAS on the Kinetics400 dataset.

4. Experiment

In this section, we will first introduce the two datasets used in this paper. Then the results of the search and training on the Kinetics400 Kay et al. (2017) dataset will be shown. Finally, some ablation experiments and analysis are performed on UCF101 Soomro et al. (2012) dataset. The implementation details are provided in the Appendix.

4.1. DataSet

Kinetics400 Kay et al. (2017) dataset contains 400 human action categories, each of them contains at least 400 videos. The dataset contains 225,946 training videos and 18,584 validation videos, each of which lasts for about 10 seconds.

UCF101 Soomro et al. (2012) is an action recognition dataset of realistic action videos collected from YouTube. 13320 videos from 101 action categories are provided. The videos of 101 action categories are divided into 25 groups, each group contains 4-7 videos of one action. UCF101 is diverse in terms of action, with a large variation in camera motion, background, lighting conditions, etc.

4.2. Result

In this section, we compare our searched network with the existing SOTA video action recognition methods on Kinetics400 and UCF101 datasets. In order to be fair, when the compared methods have multiple input frames, we try to choose the version with 16 input frames for the comparison. The methods we compare also only use the RGB modality information for processing. During inference, the most intuitive way is to take the whole video to produce a classification result, but it is generally not done due to the limitation of memory and computational efficiency. A more common approach nowadays is proposed by Slowfast Feichtenhofer et al. (2019), which divides a whole video into multiple 'view', and each view represents a temporal clip with a spatial crop. Sending each view into the network to get a softmax score, and then averaging the softmax score to get the final prediction result. This inference method is better than directly using the whole video for classification. Following the same inference strategy, we divide the whole video into 10 clips on average, take 16 pictures in each clip, resize the short edge of the picture to 256, and then cut three 224×224 crops from the left, middle and right of the image. At last we get 30 views per video and the final prediction is an average of all views' softmax scores.

From the results shown in Table 4 and Table 5, the network structure obtained from the NAS achieves the highest top1 accuracy on both Kinetics400 and UCF101 datasets. For Kinectics400, the searched network architecture achieves 78.2% top1 accuracy with 15.78 GFLOPS, while on UCF101, our searched model obtains 95.3% top1 accuracy. We have taken into account both accuracy and computational complexity in the search process. The decomposition of the traditional 3D convolution into a spatial convolution and a temporal convolution has greatly reduced the computational complexity, making the method of this paper the lowest among these methods in terms of computational complexity. Our network structure also does not need to be pre-trained on ImageNet or other image datasets. The final result also confirms the initial idea that each extended dimension of X3D can indeed be searched for using architecture search to obtain a network structure that is more suitable for video processing than the artificial design.

The reason why NAS can obtain better results than a human-designed network is that it considers more possible ways of combining hyperparameters related to architecture. We use a supernet to enumerate all the possibilities in the search space. The network architecture that people designed before this by experiment or experience is often a subset of the supernet. In addition, the differential search method together with Viterbi algorithm enable efficient recognition of the optimal network structure.

Model	Pretraining Dataset	Top-1	Top-5	GFLOPs × views	Input Frames
S3D-G	ImageNet	69.4	89.1	66.4 × 30	64
ECO_{EN}	-	70.7	89.4	267 × 1	92
R(2+1)D	-	72.0	90.0	152 × 115	32
I3D	ImageNet	72.1	90.3	108 × N/A	64
TSN	-	72.5	90.2	80 × 10	25
R(2+1)D	Sports-1M	74.3	91.4	152 × 115	32
SlowFast 4*16 R50	-	75.6	92.1	36.1 × 30	4 + 32
TEA	ImageNet	76.1	92.5	66 × 30	16
Nonlocal R50	ImageNet	76.5	92.6	282 × 30	32
X3D-L	-	77.5	92.9	24.8 × 30	16
Ours	-	78.2	93.1	15.78 × 30	16

Table 4: Comparison to the state-of-the-art on Kinetics400 val set. We report the inference cost with a single "view" (temporal clip with spatial crop) × the numbers of such views used (GFLOPs×views).

Model	Pretraining Dataset	Accuracy	GFLOPs
Peng et al. (2019)	-	58.6	N/A
TRN	-	83.5	83.83
C3D	Sports-1M	85.2	38.57
P3D	ImageNet + Sports-1M	88.6	18.51
TSN	ImageNet + Kinetics	91.1	80
R(2+1)D RGB	Sports-1M	93.6	41.69
ECO	Kinetics	94.8	267
Ours	Kinetics	95.3	15.78

Table 5: Comparison results of our method with other methods on UCF101 Dataset.

4.3. Ablation Study and Analysis

In this section, ablation study about the search dimension and their combination is conducted to explore the influence of search dimensions on the final results, and the result is shown in Table 6. All our ablation experiments are performed on UCF101 dataset.

Channels, Stack Number and Operation. Theoretically speaking, there should be six ablation experiments. But once the specific stack operation is searched and determined in this stage, the stack number is also determined. Therefore, the results from searching the channel and stack operation are the same as those obtained from the previous three dimensions, so they are not listed. When searching for the stack number, there is a situation that the stack number is not the same as the original one, and the original stack operation cannot be put in the new network structure. As a result, when searching stack number, all stack operations are fixed as basic operation with dilation 1, which is actually temporal convolution without dilation, so it returns to the basic R(2+1)D network operation. Experiments show that adding a certain number of temporal dilation convolutions is helpful for the accuracy improvement.

	Search Epoch	Channel	Stack Operation	Stack Number	Accuracy	GFLOPs
1	70	✓			89.08	17.11
2	70		✓		89.90	18.19
3	70			✓	88.81	18.19
4	70	✓		✓	90.64	19.27
5	70		✓	✓	90.59	20.40
6	40	✓	✓	✓	86.92	18.47
Ours	70	✓	✓	✓	91.52	15.78

Table 6: Ablation studies on UCF101 dataset. ✓ symbol means this dimension is searched.

As illustrated in Table 6, generally speaking, channels, stack number and stack operation are all important hyperparameters for network structure. The results of ablation experiment searching a single dimension show that search for stack operation achieves the highest accuracy among the three factors. The reason may be that this part reflects the temporal information. The stack operation of this work allows the network to pay attention to the number of frames with different intervals, so as to better capture the long interval temporal information, facilitate the establishment of the connection between the frames, and produce a better overall classification result. It can also be noticed clearly that when integrating the three dimension's search together, the proposed NAS method yields the best network architecture for video action recognition.

Search Epochs. We also explore the effects of the number of search epochs on the final results. The original search process uses 70 epochs in total, with the first 10 epochs producing the supernet's initial weight. Then we search for 60 epochs, we use training set to update operation weights, and validation set to update architecture parameters. Starting from the 50th epoch, we use the inference set to test the resulting network architecture. Figure. 2 (a) illustrates the change of accuracy with epoch during the search on the Kinetics400 dataset. And Figure. 2 (b) shows the relationship between loss and epoch. The result shows that the search result has converged at the 70th epoch since there is almost no change for the accuracy. In the ablation experiment, we reduce the number of the later search stages to half of the original one, and explore the differences between the network structure and the final network structure obtained at this time. The results show that more search epochs will produce better accuracy.

5. Conclusions

Video action recognition remains a challenging problem to be fully solved despite the remarkable progress of deep learning related techniques. In this work, we have proposed a novel NAS based approach to effectively search a better network architecture with better recognition performance and less computational burden. A new search space that considers the number of channels used in each layer, the content of operations and the number of operation stacks has been first suggested, followed by an efficient differentiable searching method. Extensive experiments on Kinetics-400 and UCF-101 validate an encouraging performance of the searched network architecture for video action recognition.

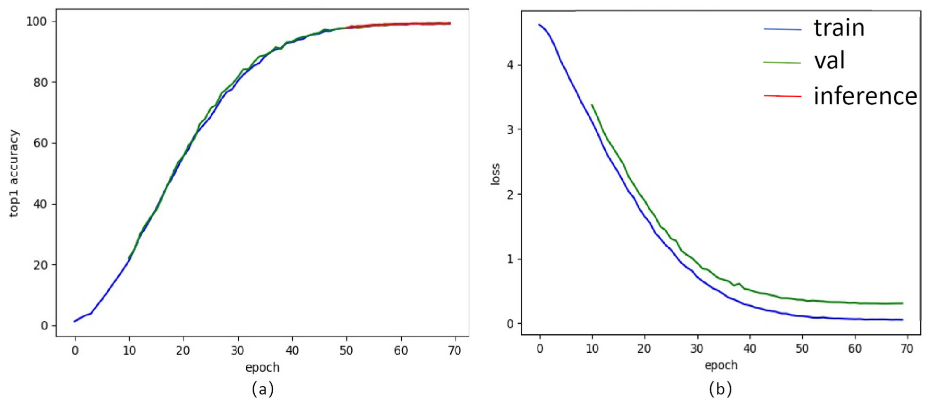

Figure 2: (a)The accuracy change with epoch during the search in the Kinetics400 dataset. (b)The loss change during the search in the Kinetics400 dataset. Blue line is the result on the training set, green line is the result on validate set and red line is the result on the inference set.

References

Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=S1c2cvqee.

Yoshua Bengio and Yann LeCun, editors. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL https://iclr.cc/archive/www/doku.php%3Fid=iclr2015:accepted-main.html.

Haoze Chen, Zhijie Zhang, Chenyang Zhao, Jiaqi Liu, Wuliang Yin, Yanfeng Li, Fengxiang Wang, Chao Li, and Zhenyu Lin. Depth classification of defects based on neural architecture search. *IEEE Access*, 9:73424–73432, 2021. doi: 10.1109/ACCESS.2021.3077961. URL https://doi.org/10.1109/ACCESS.2021.3077961.

Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Xinyu Xiao, and Jian Sun. Detnas: Backbone search for object detection. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 6638–6648, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/228b25587479f2fc7570428e8bcbabdc-Abstract.html.

Ulrich Faigle. The greedy algorithm for partially ordered sets. *Discret. Math.*, 28(2): 153–159, 1979. doi: 10.1016/0012-365X(79)90092-X. URL https://doi.org/10.1016/0012-365X(79)90092-X.

Christoph Feichtenhofer. X3D: expanding architectures for efficient video recognition. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 200–210. IEEE, 2020. doi: 10.1109/CVPR42600.2020.00028. URL https://doi.org/10.1109/CVPR42600.2020.00028.

Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 6201–6210. IEEE, 2019. doi: 10.1109/ICCV.2019.00630. URL https://doi.org/10.1109/ICCV.2019.00630.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90.

Zhuolin Jiang, Viktor Rozgic, and Sancar Adali. Learning spatiotemporal features for infrared action recognition with 3d convolutional neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 309–317. IEEE Computer Society, 2017. doi: 10.1109/CVPRW.2017.44. URL https://doi.org/10.1109/CVPRW.2017.44.

Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *CoRR*, abs/1705.06950, 2017. URL http://arxiv.org/abs/1705.06950.

Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=BJQRKzbA-.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=S1eYHoC5FX.

Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Skq89Scxx.

Wei Peng, Xiaopeng Hong, and Guoying Zhao. Video action recognition via neural architecture searching. In *2019 IEEE International Conference on Image Processing, ICIP 2019, Taipei, Taiwan, September 22-25, 2019*, pages 11–15. IEEE, 2019. doi: 10.1109/ICIP.2019.8802919. URL https://doi.org/10.1109/ICIP.2019.8802919.

A. J. Piergiovanni, Anelia Angelova, and Michael S. Ryoo. Tiny video networks. *CoRR*, abs/1910.06961, 2019a. URL http://arxiv.org/abs/1910.06961.

A. J. Piergiovanni, Anelia Angelova, Alexander Toshev, and Michael S. Ryoo. Evolving space-time neural architectures for videos. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 1793–1802. IEEE, 2019b. doi: 10.1109/ICCV.2019.00188. URL https://doi.org/10.1109/ICCV.2019.00188.

Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 2902–2911. PMLR, 2017. URL http://proceedings.mlr.press/v70/real17a.html.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 4780–4789. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33014780. URL https://doi.org/10.1609/aaai.v33i01.33014780.

Michael S. Ryoo, A. J. Piergiovanni, Mingxing Tan, and Anelia Angelova. Assemblenet: Searching for multi-stream neural connectivity in video architectures. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=SJgMK64Ywr.

Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 568–576, 2014. URL https://proceedings.neurips.cc/paper/2014/hash/00ec53c4682d36f5c4359f4ae7bd7ba1-Abstract.html.

Naresh K. Sinha and Michael P. Griscik. A stochastic approximation method. *IEEE Trans. Syst. Man Cybern.*, 1(4):338–344, 1971. doi: 10.1109/TSMC.1971.4308316. URL https://doi.org/10.1109/TSMC.1971.4308316.

Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012. URL http://arxiv.org/abs/1212.0402.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition,* *CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1–9. IEEE Computer Society, 2015. doi: 10.1109/CVPR.2015.7298594. URL https://doi.org/10.1109/CVPR.2015.7298594.

Du Tran, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 4489–4497. IEEE Computer Society, 2015. doi: 10.1109/ICCV.2015.510. URL https://doi.org/10.1109/ICCV.2015.510.

Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 6450–6459. IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00675. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Tran_A_Closer_Look_CVPR_2018_paper.html.

Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theory*, 13(2):260–269, 1967. doi: 10.1109/TIT.1967.1054010. URL https://doi.org/10.1109/TIT.1967.1054010.

Jiangchuan Wei, Hanli Wang, Yun Yi, Qinyu Li, and Deshuang Huang. P3D-CTN: pseudo-3d convolutional tube network for spatio-temporal action detection in videos. In *2019 IEEE International Conference on Image Processing, ICIP 2019, Taipei, Taiwan, September 22-25, 2019*, pages 300–304. IEEE, 2019. doi: 10.1109/ICIP.2019.8802979. URL https://doi.org/10.1109/ICIP.2019.8802979.

Yan Wu, Zhiwu Huang, Suryansh Kumar, Rhea Sanjay Sukthanker, Radu Timofte, and Luc Van Gool. Trilevel neural architecture search for efficient single image super-resolution. *CoRR*, abs/2101.06658, 2021. URL https://arxiv.org/abs/2101.06658.

Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XV*, volume 11219 of *Lecture Notes in Computer Science*, pages 318–335. Springer, 2018. doi: 10.1007/978-3-030-01267-0_19. URL https://doi.org/10.1007/978-3-030-01267-0_19.

Mingwei Zhang, Weipeng Jing, Jingbo Lin, Nengzhen Fang, Wei Wei, Marcin Wozniak, and Robertas Damasevicius. NAS-HRIS: automatic design and architecture search of neural network for semantic segmentation in remote sensing images. *Sensors*, 20(18):5292, 2020. doi: 10.3390/s20185292. URL https://doi.org/10.3390/s20185292.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=r1Ue8Hcxg.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8697–8710. IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00907. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Zoph_Learning_Transferable_Architectures_CVPR_2018_paper.html.

Appendix A.

A.1. Implement details

Data Augmentation. For UCF101 and Kinetics400 datasets, we use the same data augmentation method in the process of search and training. For the spatial dimension, video frames are scaled to the size of 256 $\times$ 342 and then randomly crop 224 $\times$ 224 pixels from the frame. For the temporal dimension, we randomly sample 16 consecutive frames from the video with temporal jittering while training. In the inference process, 10 clips and 3 crops are used to make a fair comparison with the common practice.

Search. For the search part, we divide the data used for training in the original dataset into three parts: training, validate and inference, accounting for 0.6, 0.2 and 0.2 of the total respectively. The search process uses a total of 70 epochs. We use the training set to optimize the operation weight w of the network. First, we let the network train for 10 epochs, so that each path in Supernet has an initial weight w_0. In the next 60 epochs, we first use the validate set to optimize the network architecture parameters α and β, and then use the training set to optimize the operation weight w. In the last 20 epochs, we add the inference set data to evaluate the final effect. Our search was performed on two Tesla V100 32G GPUs, each with a batch size of 8. When we optimize the operation weight w of the network, we use the SGD optimizer Sinha and Griscik (1971) with 0.9 momentum and 4×10^{-5} weight decay. The learning rate strategy uses cosine annealing learning rate schedule Loshchilov and Hutter (2017). The initial learning rate was 0.02 and the lowest was 1×10^{-4}. For the optimization of network architecture, we use Adam optimizer Bengio and LeCun (2015) with 10^{-3} weight decay, $\beta = (0.5, 0.999)$. And the learning rate is the fixed 3×10^{-4} for the two parameters of the architecture. The loss function we use is a cross entropy loss plus a computational complexity constraint.

Retrain. After the search, we get the final architecture on this dataset and the weight information trained on each path corresponding to this architecture. We retrain the network based on the architecture and weight information. We use the whole training dataset for training, and get the final results on the origin validation set. Because we don't need to save supernet information, our batch size can be larger in training than in searching. In this paper, we use 16. The whole model is trained for 150 epochs, and the optimizer uses SGD with 0.9 momentum and 4×10^{-5} weight decay. The learning rate strategy uses cosine annealing learning rate schedule Loshchilov and Hutter (2017). The initial learning rate was 0.1 and the lowest was 1×10^{-4} . The dropout probability is 0.5 after the final global average pooling layer. Finally, it is sent to the linear layer to classify according to the number of categories of each dataset. The loss function we use is a cross entropy loss.