AttentionNAS: Spatiotemporal Attention Cell Search for Video Classification Supplementary Materials

Xiaofang Wang[2]*, Xuehan Xiong[1], Maxim Neumann[1], AJ Piergiovanni[1], Michael S. Ryoo[1], Anelia Angelova[1], Kris M. Kitani[2], and Wei Hua[1]

[1] Google
[2] Carnegie Mellon University

A Attention Cell Search Space Details

A.1 Keys and Values in Dot-product Attention

We introduce an additional design choice in dot-product attention. In Sec 3.1, a dot-product attention operation is defined as:

$$\begin{aligned} f'_{\text{in}} &= \texttt{ReshapeTo2D}(f_{\text{in}}), \\ W_{\text{dot-prod}} &= \phi(G_1(f'_{\text{in}})G_2(f'_{\text{in}})^T), \\ f_{\text{out}} &= \texttt{ReshapeTo2D}^{-1}(W\texttt{ReshapeTo2D}(G_3(f_{\text{in}}))), \end{aligned} \tag{A}$$

where f_{in} and f_{out} are the input and output feature map of the attention operation respectively, $W_{\text{dot-prod}}$ is the attention weight matrix, and G_1, G_2 and G_3 are all $1 \times 1 \times 1$ convolutional layers.

Let $Q = G_1(f'_{\text{in}})$, $K = G_2(f'_{\text{in}})$ and $V = \texttt{ReshapeTo2D}(G_3(f_{\text{in}}))$. Q, K and V are termed as query, keys and values in dot-product attention [6]. In Eq A, the query, keys and values are computed based on the same feature map, i.e., the operation input f_{in}. It is also common practice in dot-product attention to compute the keys and values based on feature maps other than f_{in}. For example, dot-product attention has been used in Transformer [6] in the following way: the query comes from the decoder while the keys and values come from the encoder, so that every position in the decoded sequence can attend to positions in the input sequence.

In our search space, for a dot-product attention operationin, we also allow computing its keys and values based on the cell input f_0. This allows positions in the operation input f_{in} to attend to positions in the cell input f_0. When computing keys and values based on f_0, the dot-product attention becomes:

$$\begin{aligned} f'_0 &= \texttt{ReshapeTo2D}(f_0), \\ W_{\text{dot-prod}} &= \phi(G_1(f'_{\text{in}})G_2(\boldsymbol{f'_0})^T), \\ f_{\text{out}} &= \texttt{ReshapeTo2D}^{-1}(W\texttt{ReshapeTo2D}(G_3(\boldsymbol{f_0}))). \end{aligned} \tag{B}$$

* Work done while an intern at Google.

The differences between Eq. A and Eq. B are highlighted in boldface and red.

In summary, for dot-product attention operations in the attention cell, we can choose to compute its keys and values based on the operation input f_{in} or the cell input f_0. We include this choice in the cell level search space, i.e., all the dot-product attention operations make the same choice, either computing the keys and values based on their own operation input or the cell input f_0.

A.2 Channel Attention

While our search space mainly focuses on spatiotemporal attention, we include channel attention as an additional choice in the search space. Concretely, when building an attention operation, the search algorithm can choose whether to apply a feature gating layer [8] to the attended feature map f_{out}. The feature gating layer is a typical channel attention mechanism. It first applies average pooling to a 4D feature map across space and time, then learns a weighting factor for each channel, and finally multiplies features at each channel of the original feature map with the learned weighting factor. Note that channel attention does not replace the attention operation described above and is only an additional layer choice within the attention operation.

When using differentiable search, we learn a 2-dim probability distribution $w_{i,j}^{\text{gating}}$ for each node, indicating whether to include a feature gating layer [8] in the attention operation represented by the node.

B Experimental Details

B.1 Training and Inference

We conduct experiments on two benchmark datasets: Kinetics-600 [1] and Moments in Time (MiT) [4]. Kinetics-600 contains about 392K training videos and 30K validation videos from 600 classes. MiT consists of about 800K training videos and 34K validation videos from 339 classes.

After obtaining the attention cells found by our method, we fully train the backbone networks and cells on training videos and report their performance on validation videos. Following non-local blocks [7], we insert 5 cells or non-local blocks into the backbone. For I3D or S3D, they are inserted 5 inception modules (4a to 4e, see Table 1 in [5]). For I3D-R50, we insert them after 5 residual blocks, where 2 cells are inserted after every other residual block in res_3 and 3 cells are inserted after every other residual block in res_4.

During training, we resize the spatial resolution of videos to 256×256 and randomly crop 224×224 pixels or its horizontal flip from videos, for both Kinetics-600 [1] and MiT [4]. For I3D or S3D, we randomly crop 64 consecutive frames from the full-length video as the input clip during training. For I3D-R50, we randomly crop 16 frames with stride 4 from the full-length video.

During inference, we perform fully-convolutional inference both spatially and temporally. We resize the spatial resolution to 256×256, pass the full-length video to the network, and predict the class based on the softmax scores. Our inference procedure does not require the sampling of multiple temporal clips and spatial crops in previous works [3]. The input clip to I3D or I3D has 250 frames for Kinetics-600 and has 75 frames for MiT. The input clip to I3D-50 has 64 frames for Kinetics-600 and has 18 frames for MiT, which is obtained by temporally downsampling the full-length video with stride 4.

We initialize the backbone in all the models (backbone, backbone + non-local blocks or our attention cells) with its ImageNet pre-trained weights. I3D or S3D based models are trained for 135 epochs, and I3D-R50 based models are trained for 150 epochs on Kinetics-600. All the models are trained for 45 epochs on MiT. We adopt a cosine learning rate schedule with a linear warm-up. The initial learning rate is 0.1 for I3D or S3D and 0.4 for I3D-R50. All the models are trained on 50 GPUs with synchronized SGD. The momentum is 0.9. The batch size per GPU is 6 for I3D or S3D and 4 for I3D-R50.

B.2 Attention Cell Implementation

We have three pre-processing steps for the input to the entire attention cell: (1) channel reduction, (2) spatial resize, and (3) temporal grouping. These steps can not only reduce the computation consumed by the cell, but also allow the cell to process feature maps of different temporal and spatial resolutions.

Let (B, T, H, W, C) be the shape of the input to the entire cell. We explicitly write out the batch size dimension B for better explanation. We first reduce the number of channels from C to $C_{\text{reduction}}$ with a $1\times1\times1$ convolutional layer. After channel reduction, the shape becomes $(B, T, H, W, C_{\text{reduction}})$. Then, we resize the spatial resolution of the feature map with bilinear interpolation from (H, W) to $(H_{\text{resize}}, W_{\text{resize}})$, so the shape becomes $(B, T, H_{\text{resize}}, W_{\text{resize}}, C_{\text{reduction}})$. Finally, we divide the feature map into multiple groups of T_{group} frames and obtain a feature map of shape $(nB, T_{\text{group}}, H_{\text{resize}}, W_{\text{resize}}, C_{\text{reduction}})$, where $T = n \cdot T_{\text{group}}$ and zero padding frames are added when necessary. The feature map of shape $(nB, T_{\text{group}}, H_{\text{resize}}, W_{\text{resize}}, C_{\text{reduction}})$ is then passed to attention operations in the cell. During the combination procedure, we resize the spatial resolution back to (H, W) and merge temporal groups back to T frames.

It is not difficult to see that these steps can reduce the computation. We elaborate on the second advantage. Note that the temporal and spatial resolution of test videos can vary (e.g., $250 \times 256 \times 256$) and be different from sampled training clips (e.g., $64 \times 224 \times 224$). This causes the shape of the feature map output by each layer to be different between training and test. However, temporal attention requires the spatial resolution of the feature map to be fixed and spatial attention requires the number of frames to be fixed. To address this issue, we adopt these pre-processing steps so that the input to attention operations always has a fixed shape of $(T_{\text{group}}, H_{\text{resize}}, W_{\text{resize}}, C_{\text{reduction}})$.

B.3 Search Algorithm

GPB We sample training videos from the original dataset as the search-train and search-validation split. No validation videos are used during the search. We maximize the validation performance using GPB. We set the number of trails of GPB to 50, i.e., 50 attention cells are sampled by GPB and evaluated on the search-validation split after trained on the search-train split. Both the search-train split for Kinetics-600 and MiT contain about 360K videos. We train for 60 epochs for Kinetics-600 and 20 epochs for MiT during the search on their corresponding search-train split. We set $K = 4$ and search for an attention cell consisting of 4 attention operations. We use GPB to find one position-agnostic attention cell and insert the same cell architecture at different positions in the backbone network. To simplify the search space explored by GPB, we restrict the k^{th} operation to select only one feature from $\{f_0, f_1, \ldots, f_{k-1}\}$ as its input.

Differentiable Method When using the differentiable search method, we consider a supergraph consisting of 2 levels. Each level in the supergraph has 6 nodes. We do not include more nodes in one level due to the GPU memory constraint. At eacl level, we repeat each attention dimension twice and only include dot-product attention. So the 6 nodes are 2 temporal dot-product, 2 spatial dot-product, and 2 spatiotemporal dot-product attention operations. We also fix that the keys and values of dot-product attention are computed based on the attention cell input (see Eq. B). This is the default supergraph design and we study other supergraph designs in Sec C.

The connection weights and the network weights are learned jointly on training videos. The entire search process of the differentiable method consumes a computational cost similar to fully training one network on the training videos. For example, training I3D with the found attention cells on Kinetics-600 takes about 2.5 days. Searching attention cells for I3D, i.e., training I3D with supergraphs, takes about 3.5 days on Kinetics-600. The increase in the time is due to that supergraphs consume more computation than the final attention cells.

When deriving the attention cell design from the learned connection weights, the hyper-parameters α and β are set to $\alpha = 3, \beta = 2$. Attention cells found by the differentiable method do not have a fixed number of operations, which are determined by the determined connection weights and α and β. Each operation may receive up to β feature maps and computes a weighted sum of these feature maps as its input. We slightly revisit the combination procedure for cells found by the differentiable method. Instead of combining all the operation output feature maps, we only combine the output of the top α nodes (operations) with the highest weights in w^{sink}.

B.4 Comparison of FLOPs

We compare the inference FLOPs of all the models on Kinetics-600 in Table A. Note that although our cells contain multiple operations, the aforementioned preprocessing applied on the cell input can effectively reduce the FLOPs consumed

Table A: Inference FLOPs on Kinetics-600.

Model	Top-1	Top-5	GFLOPs
I3D [2]	75.58	92.93	1136
I3D+NL [7]	76.87	93.44	1305
I3D+Cell	77.86	93.75	1170
S3D [8]	76.15	93.22	656
S3D+NL [7]	77.56	93.68	825
S3D+Cell	78.51	93.88	692
I3D-R50 [7]	78.10	93.79	938
I3D-R50+Cell	79.83	94.37	1034

by attention operations. As shown in Table A, our cells only add a small amount of computation to the backbone network and use fewer FLOPs than non-local blocks. The FLOPs are computed when the input clip has 250 frames with spatial resolution 256×256.

C Ablation Study of Supergraph Designs

In the differentiable method, we represent the attention cell search space as a supergraph. Using different supergraph designs allows us to analyze what desgin choice is important for the performance of the discovered attention cells. Specifically, we compare the following three supergraph designs:

Table B: Comparison between different supergraph designs.

Model	Top-1	Top-5
I3D [2]	75.58	92.93
I3D+NL [7]	76.87	93.44
I3D+SG-1 Cell	77.86	93.75
I3D+SG-2 Cell	77.82	93.75
I3D+SG-3 Cell	77.71	93.87

SG-1. SG-1 is our default choice described Sec B.3. It contains 2 levels, where each level has 6 nodes. SG-1 only contains dot-product attention and the 6 nodes at each level are 2 temporal dot-product, 2 spatial dot-product, and 2 spatiotemporal dot-product attention operations. In SG-1, the keys and values of dot-product attention are computed based on the cell input (see Eq. B).

SG-2. Same SG-1, SG-2 also contains 2 levels and each level has 6 nodes. SG-2 include both map-based attention and dot-product attention. The 6 nodes at each level are 1 temporal dot-product, 1 spatial dot-product, 1 spatiotemporal dot-product, 1 temporal map-based, 1 spatial map-based, and 1 spatiotemporal map-based attention operation. In SG-2, the keys and values of dot-product attention are also computed based on the cell input (see Eq. B).

SG-3. SG-3 is the same as SG-1 except that the keys and values of dot-product attention are computed based on the input to each attention operation (see Eq. A), instead of the cell input.

Comparing SG-1 and SG-2 tells us which attention type (map-based or dot-product) is more important. As shown in Table B, SG-1 and SG-2 achieve a very close top-1 accuracy and the same top-5 accuracy on Kinetics-600. However, we observe that most operations (20 of out 28) in the 5 position-specific cells discovered from SG-2 are dot-product attention. This shows that dot-product attention is more important than map-based attention, and explains why SG-1 can achieve high accuracy with only dot-product attention.

SG-3 achieves similar performance with SG-1 and also outperforms non-local blocks. This shows that our search space is not sensitive to whether to compute the keys and values based on the input to each dot-product operation or based on the cell input.

D Attention Cell Visualization

We visualize the position-agnostic attention cell found by GPB and the differentiable method in Fig. A. The position-specific cells found by the differentiable method are shown in Fig. B. These cells are found for I3D and on Kinetics-600. We show the attention dimension and type of each operation, as well as the connectivity between the operations.

The cell found by GPB contains both map-based attention and dot-product attention and contains one path that first applies spatial attention and then temporal attention. Cells found by the differentiable method only contain dot-product attention as we only include dot-product attention in the supergraph (SG-1). We observe that all the cells found by the differentiable method prefer decomposing spatiotemporal attention into temporal and spatial attention, as they all contain paths that first apply temporal attention and then spatial attention. This shares a similar spirit to S3D [8] that decomposes a 3D convolution into a 2D spatial convolution and a 1D temporal convolution. As a side note, our cells choose to first apply temporal and then spatial attention, while S3D first applies spatial convolution and then temporal convolution.

References

1. Carreira, J., Noland, E., Banki-Horvath, A., Hillier, C., Zisserman, A.: A short note about kinetics-600. arXiv preprint arXiv:1808.01340 (2018)
2. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: CVPR (2017)
3. Feichtenhofer, C., Fan, H., Malik, J., He, K.: Slowfast networks for video recognition. In: ICCV (2019)

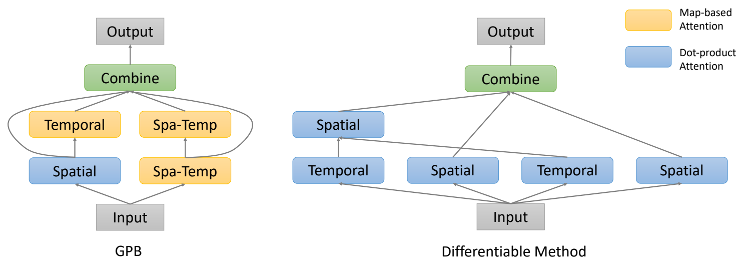

Fig. A: Visualization of the position-agnostic cell discovered by GPB and the differentiable method for I3D and on Kinetics-600. 'Spa-Temp' stands for the spatiotemporal attention dimension.

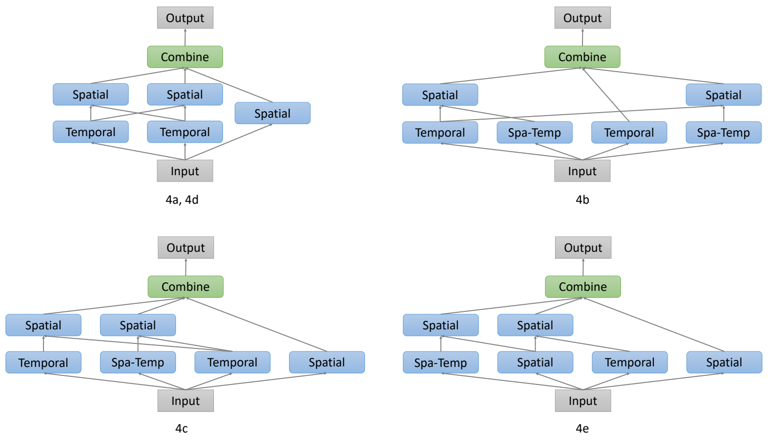

Fig. B: Visualization of the position-specific cells discovered by the differentiable method for I3D and on Kinetics-600. 'Spa-Temp' stands for the spatiotemporal attention dimension. The text under each cell indicates the inception module after which the cell is inserted (4a to 4e, see Table 1 in [5]) in the Inception network. The learned attention cell for 4a and 4d are the same.

4. Monfort, M., Andonian, A., Zhou, B., Ramakrishnan, K., Bargal, S.A., Yan, T., Brown, L., Fan, Q., Gutfreund, D., Vondrick, C., et al.: Moments in time dataset: one million videos for event understanding. TPAMI (2019)
5. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: CVPR (2015)
6. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017)
7. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: CVPR (2018)
8. Xie, S., Sun, C., Huang, J., Tu, Z., Murphy, K.: Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In: ECCV (2018)