AssembleNet++: Assembling Modality Representations via Attention Connections - Supplementary Material -

Michael S. Ryoo[1,2], AJ Piergiovanni[1], Juhana Kangaspunta[1], and Anelia Angelova[1]

[1] Robotics at Google
[2] Stony Brook University
{mryoo,ajpiergi,juhana,anelia}@google.com

A Appendix

A.1 Convolutional blocks with (2+1)D residual modules

Each convolutional block is implemented by alternating 2-D residual modules and (2+1)D residual modules as was done in [5]. Each (2+1)D residual module does 1D temporal convolution first, and then 2D spatial convolution followed by 1x1x1 convolution. This (2+1)D residual module is also similar the ones used in [2]. We use the filter size of 3x3 for spatial convolutional layers, and the size of 3 for temporal convolutional layers. Temporal dilation from [5] was used to control the temporal resolution of each (2+1)D block. 2D and (2+1)D residual modules are repeated multiple times in each block. As a result, our residual blocks have a total of 9, 12, 18, and 9 convolutional layers for levels 1 to 4, making the depth of the network comparable to ResNet-50.

Since we follow the AssembleNet (2+1)D block design, the total number of filers in each level is maintained as a constant, regardless the number of blocks in the level. That is, our model has the number of parameters equivalent to the two-stream version of ResNet-50.

Each RGB input block has 1 spatial convolutional layer (filter size 7x7, stride 2x2), 1 temporal convolutional layer (filter size 5, stride 1), and one max pooling layer (pool and stride size 2x2). Each optical flow input block has 1 spatial convolutional layer (filter size 7x7, stride 2x2) and one max pooling layer (pool and stride size 2x2). The object input block has only one max pooling layer (pool and stride size 4x4).

A.2 Computation overhead of peer-attention

Our approach is adding very little computation overhead. As was done in previous differentiable architecture search [1, 4], once the one-shot search is finished and the attention connection weights (i.e., h) are obtained, only a single peer node is selected by the softmax for each block and the others are discarded. Figure 1(c) in the paper shows an example of the final model.

As a result, similar to what was reported in [3] with channel-wise self-attention, our peer-attention only causes 0.151% increase in the total computation (FLOPs). The increase in the number of parameters is 1.68%. Peer-attention only adds one fully connected layer after every block. Each FC layer has an identical number of parameters as a 1x1 conv layer, while spending significantly less amount of computation (compared to 1x1) as it does not have spatial resolution.

A.3 Learned architecture

We also provide our model in table form in Table 1. In particular, the 3rd column of the table shows the connectivity: a list of blocks where the input to that block is coming from. The fusion of the convolutional block outputs with different tensor shapes is done using a spatial pooling and a 1x1 conv layer, before the weighted summation. Further, notice that (as described in Section 3.5) there is the one-shot peer-attention search module implemented before every input to a convolutional block (i.e., Figure 2 in the paper), in addition to Table 1. The final block (i.e., block 14) is followed by a FC layer to generate logits. Temporal max pooling is used to combine logits of different frames in videos. Finally, cross entropy loss is used to train the network.

Table 1. The table form of our model with detailed parameters. This model corresponds to Figure 4 in the paper. "C, dilation, stride" in the Table correspond to the ResNet channel parameter, temporal dilation rate, and spatial stride. Note that the object input block does not have any convolutional layer, and 151 is the number of object categories which decide the size of its input channel.

Index	Level	Input connections	Block parameters: C, dilation, stride
0	0	[RGB]	32, 2, 4
1	0	[RGB]	32, 4, 4
2	0	[Flow]	32, 1, 4
3	0	[Flow]	32, 1, 4
4	0	[Object]	151, 1, 4
5	1	[0, 1, 2, 3, 4]	32, 1, 1
6	1	[0, 1, 4]	32, 4, 1
7	1	[2, 3, 4]	32, 8, 1
8	1	[2, 3, 4]	32, 1, 1
9	2	[0, 1, 2, 4, 5, 6, 7, 8]	64, 4, 2
10	2	[2, 3, 4, 7, 8]	64, 1, 2
11	2	[0, 4, 5, 6, 7]	128, 8, 2
12	3	[4, 11]	256, 8, 2
13	3	[2, 3, 4, 5, 6, 7, 8, 10, 11]	256, 4, 2
14	4	[4, 12, 13]	512, 2, 2

References

1. Bender, G., Kindermans, P.J., Zoph, B., Vasudevan, V., Le, Q.: Understanding and simplifying one-shot architecture search. In: International Conference on Machine Learning (ICML) (2018)
2. Feichtenhofer, C., Fan, H., Malik, J., He, K.: Slowfast networks for video recognition. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2019)
3. Hu, J., Shen, L., Albanie, S., Sun, G., Wu, E.: Squeeze-and-excitation networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
4. Liu, H., Simonyan, K., Yang, Y.: DARTS: Differentiable architecture seach. In: International Conference on Learning Representations (ICLR) (2019)
5. Ryoo, M., Piergiovanni, A., Tan, M., Angelova, A.: AssembleNet: Searching for multi-stream neural connectivity in video architectures. In: International Conference on Learning Representations (ICLR) (2020)