Supplementary Materials for Dynamic Dual Gating Neural Networks

Fanrong Li[1,2], Gang Li[1], Xiangyu He[1], Jian Cheng[1,2,3*]
[1]Institute of Automation, Chinese Academy of Sciences
[2]School of Future Technology, University of Chinese Academy of Sciences,
[3]CAS Center for Excellence in Brain Science and Intelligence Technology
lifanrong2017@ia.ac.cn, gangli0426@gmail.com, {xiangyu.he, jcheng}@nlpr.ia.ac.cn

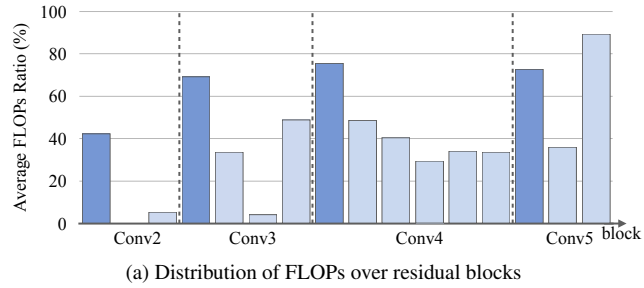
1. Implementation Details for Real Speedup Experiments

This section elaborates the implementation details of the realistic speedup experiments on the CPU and FPGA accelerator. We evaluated the real speedup of our method on two different hardware environments: a CPU (I7-6700, 24G RAM, and Ubuntu 16.04 OS) and an embedded FPGA accelerator (Ultra96 SoC). On CPU, we replace convolutions with our sparse implementation using the MKL library and enable multi-threading by default. On FPGA, we use a 16x16 systolic array as the baseline, which supports the weight stationary dataflow. We add the dynamic index module to support our proposed dynamic computing.

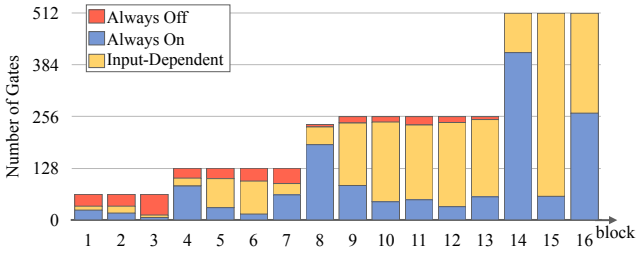
During calculation, due to the structured sparsity, our method can perform efficiently by using the im2col method for convolution. As for spatial gating, all the output feature maps share the same spatial mask, so the sparse pattern is still structured. And the entire corresponding column in the input activation matrix can be removed. As for the channel gating, the masked channels can be not only the output sparsity, but also the input sparsity of the next layer. So the corresponding columns and rows of the weight matrix can be removed. And the masked channels in the input activation correspond to rows in the input matrix. As a result, after removing these rows and columns, both weight and activation matrices can maintain the regular structure, enabling efficient calculation.

2. Distribution of Gates

To further understand the learned inference paths, we analyze the FLOPs distribution over different residual blocks, as shown in Figure 1a. We observe that the blocks containing downsampling layers are allocated more FLOPs, which means those blocks are more important than the others. In addition, we study the distribution of the learned gates and



(a) Distribution of FLOPs over residual blocks



(b) Channel gates distribution

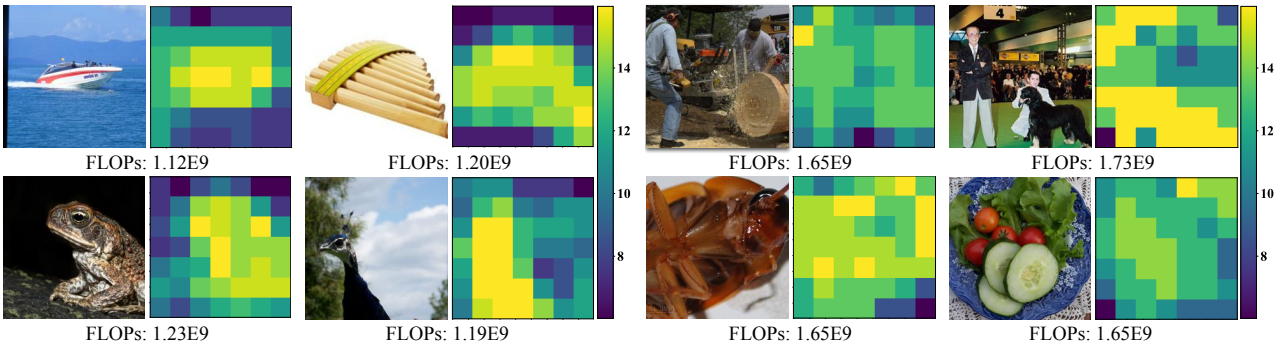
Figure 1: The distribution of computation (a) and different gates (b) in our DGNet-34 with $T_d = 0.4$ model trained on ImageNet. (b) categorize gates as always off if they are off for all the samples in the validation set. A gate is always on means that it is always on for all the samples. And a gate is input-dependent means that it is on for some of the test samples and off for the others.

observe that 96.3% of spatial gates are input-dependent, as the spatial gating modules learn the important regions to execute. As for the channel gates, Figure 1b shows their distribution. We can observe more input-dependent gates in the deep layers and fewer in the shallow layers. One reason for this could be that deep layers can capture semantic information to discriminate between categories.

3. Visualization Results

Figure 2 shows more results of the spatial cost maps. Here we also choose ResNet-34 with dual gating (DGNet-

*Corresponding author.



(a) "Easy" samples

(b) "Hard" samples

Figure 2: Results from ImageNet validation set. (a) gives results using fewer FLOPs, and (b) shows the results of "hard" samples.

34) in the experiments, and the target density is set to $T_d = 0.4$. The average FLOPs of the model is $1.50E9$ over the validation sets. As shown in Figure 2, "easy" samples can complete the processing with fewer FLOPs, and those images are indeed easier to identify because there is only a single object located in the center of the image. However, "hard" samples need more FLOPs, and those images usually contain several objects or parts of an object, which are harder to be classified. Besides, dual gating can focus the computation on those informative regions.