MicroNet: Large-Scale Model Compression Competition

Mengze Zeng[*]
Momenta
zengmengze@momenta.ai

Ziheng Wu[*]
Momenta
wuziheng@momenta.ai

Jie Hu[*]
State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences
University of Chinese Academy of Sciences
Momenta
hujie@ios.ac.cn

1 Model Architecture

In the competition, we play in *ImageNet Classification* track. We employ MixConv [7] integrated with SE-module [3] as basic block to form the entry model, named **ProfitableNet**. And the design of overall module arrangement borrows from off-the-shelf efficient mobile model in ProxylessNAS [1]. The schema of ProfitableNet is showed in Appendix A.

2 Sparsity

Benefitting from effective sparsity matrix computation and storage, non-structured pruning plays an important role in cost-efficient inference and storage. In this part, we refer to Dense-Sparse-Dense (DSD) [2] strategy to perform non-structured pruning, keeping accuracy with the aid of finetuning in which zeroing relatively unimportant parameters. The process is depicted in Fig. 1

Implementation of DSD approach to prune CNNs requires following procedures:

- Sensitivity Analysis
- Generating parameters mask
- Finetuning with mask

Sensitivity Analysis is a method to decide which layers need to be pruned and the degree of sparsity for these layers. For each layer with parameters (e.g. Convolution layer), we iteratively zero $10\% \sim 90\%$ with step 5% parameters whose absolute value are relatively small and report the accuracy on a special mini-train dataset split from whole train set for each percentage. After that, we can plot a sensitivity curve like in Fig. 2. By means of the curve, we can easily find which layers are sensitive to be pruned. Empirically, we set a unified lower bound of accuracy and prune the maximum percentage parameters whose test accuracy exceed the bound for each layer.

Generating parameters mask. After sensitivity analysis, we can generate a binary mask for each layer with parameters to record where set to zero. The generated masks are also indispensable during the following finetuning step.

[*]Equal contribution

MicroNet Challenge hosted at NeurIPS 2019, Vancouver, Canada.

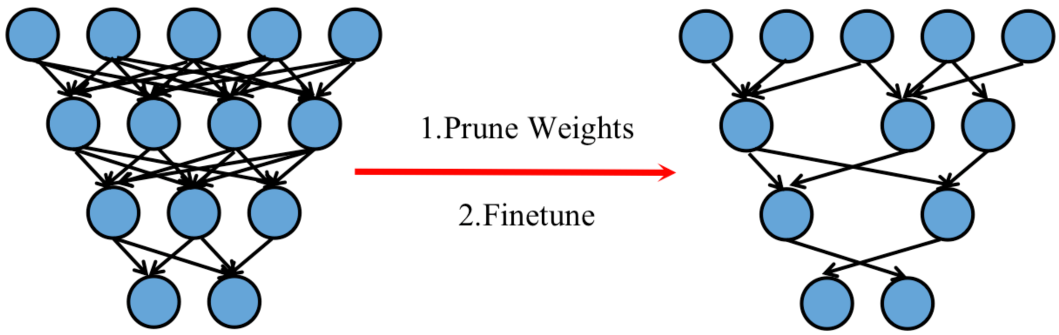

Figure 1: Training flow of non-structured sparsity.

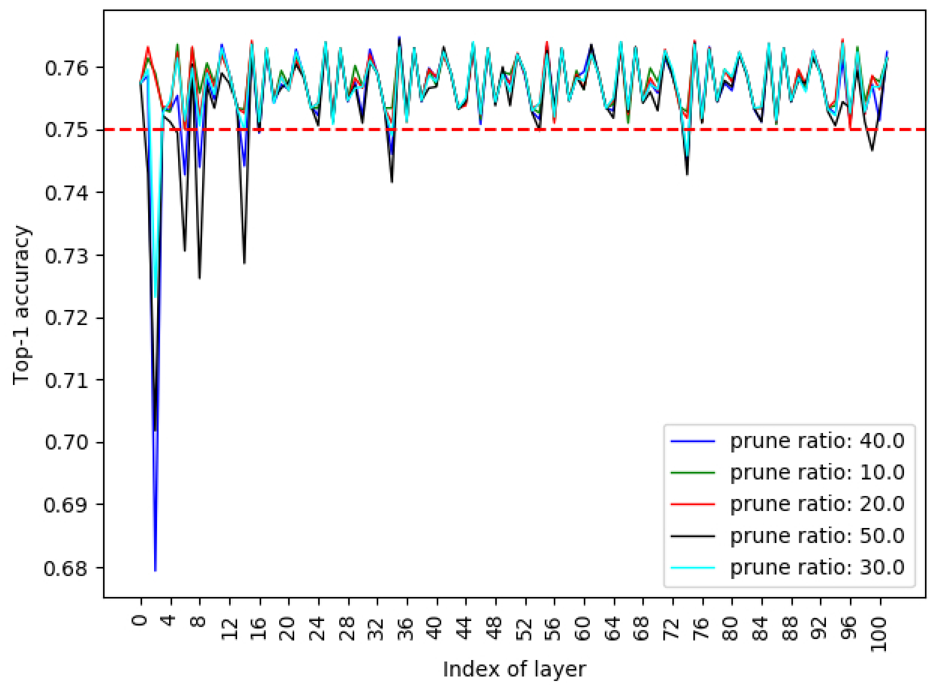

Figure 2: The curve of sensitive analysis result of prune ratio in $10\% \sim 50\%$, other prune ratios are omitted for clarity.

Finetuning with mask. In the last, we finetune the model with the generated masks for several epoch to recoup the loss in accuracy. Throughout the finetuning, the pruned parameters are never to be updated.

Notably, we remove the last dense procedure of DSD, since it brings marginal accuracy improvements and make model lost sparsity.

Following the above processes, we are able to prune model at about $30\% \sim 50\%$ **sparsity with neglectable loss in accuracy.**

3 Quantization

Quantization is an another essential procedure which can largely accelerate inference.

Quantization for a floating-point vector X is able to be formulated:

$$x_{int} = clamp((round(\frac{x}{s}), INT_{min}, INT_{max})), x \in X$$

where x refers to an element in vector X, s refer to step of two adjacent fixed-point numbers, and INT_{min} and INT_{max} refer to corresponding lower bound and upper bound that fixed-point number can reach. Given a fixed-point number, the quantization process is to find the optimal step to minimize the error.

There are three different quantization approach used here:

- Symmetric KL-divergence based quantization
- Asymmetric KL-divergence based quantization
- Symmetric MaxValue quantization

3.1 Symmetric KL-divergence based quantization

Symmetric KL-divergence based quantization is used for the activation which has negative responses (e.g. input, convolutional output without ReLU activation). The diagram is show in Fig. 3.

Pseudo code of calculating step

limitation: precision <= 10

```
SOURCE_BINS = 2048
TARGET_BINS = 2^(precison-1)
bin[0], ... , bin[2047] = GenerateHist(fabs(Input))

For i in range(TARGET_BINS, SOURCE_BINS):
    reference_dist_P = [bin[0] , ..., bin[i-1]]
    outliers_count = sum(bin[i] , bin[i+1] , ... , bin[SOURCE_BINS-1])
    reference_distribution_P[ i-1 ] += outliers_count
    P /= sum(P) # normalize distribution P
    candidate_dist_Q = quantize[bin[0], ... , bin[i-1]] into TARGET_BINS
        levels
    expand candidate_distribution_Q to 'i' bins
    Q /= sum(Q) # normalize distribution Q
    divergence[i] = KL_divergence(reference_dist_P, candidate_dist_Q)
End For

Find the minimal KL_divergence called minKL
Find the max index 'h' for which divergence[k] <= tolerance * minKL
s = (h + 0.5) * (width of a source Tbin) / TARGET_BINS
return s
```

Hyperparameter

Precision : Specify the fixed-point number, such as int8 means precision=8.

Tolerance: Specify scale factor for minimal kl-divergence to find better step. We noticed that taking the step of minimal kl-divergence often obtain inferior result and enlarge this step can get better performance. So we introduce *tolerance* to relax the objective. Instead, we choose the max step in which kl-divergences less then *tolerance* * (minimal kl-divergence). In particular, we found that setting *tolerance* = 1.3 can achieve a good accuracy.

If a *precision* is given, the INT_{min} and INT_{max} can be calculated using the following formulation:

$$INT_{min} = -2^{precision-1}$$
$$INT_{max} = 2^{precision-1} - 1$$

3.2 Asymmetric KL-divergence based quantization

It is almost the same as Symmetric KL-divergence based quantization that customized for layer only has positive responses (e.g. ReLU or Sigmoid output). The diagram is exhibited in Fig. 4. In this case, the lower bound and upper bound are defined:

$$INT_{min} = 0$$
$$INT_{max} = 2^{precision} - 1$$

3.3 Symmetric MaxValue quantization

The lower bound and upper bound formulation is the same as Symmetric KL-divergence based quantization. And the step calculation are formulated as: $s = max(fabs(X))/(2^{precision-1} - 1)$.

3.4 Accumulation in FP16

For common implementation of quantized matrix dot product, the accumulation part is still in FP32 or INT32 to maintain precision. In order to accelerate accumulation process, we find that FP16 is accurate enough in modern neural networks with normalization technology (e.g. BatchNorm, LayerNorm). In the competition, we implement convolution with GEMM by calling `cublasGemmEx` function with FP16 dataType and FP16 computeType in NVIDIA cublas engine.

3.5 Quantization in competition

In the competition, we use *fake quantization* throughout inference since lack of standard software support. Actually, we take floating-point number $x_{quant} = x_{int} * s$ as quantized number and use standard floating-point library to calculate. As mentioned above, we perform conversion after each convolution. If it followed by a ReLU or Sigmoid activation layer whose output are all non-negative, Asymmetric KL-divergence based quantization will be applied. Otherwise, Symmetric KL-divergence based quantization will be taken. As for convolutional kernels, we perform Symmetric MaxValue quantization. Notably, it is essential to quantize convolutional kernels in one layer separately which can largely reduce the accuracy loss (e.g. If the shape of convolutional kernels is $[c_{out}, c_{in}, k, k]$, c_{out} number of individual step need to be calculated). Besides, there are several special cases need to be pointed out:

- In residual-like block, we do not perform any fixed-point quantization closely followed the last convolution in residual branch. Instead, we quantize the feature maps after element-wise summation.
- In SE-module, we do not perform any fixed-point quantization on all feature maps as well. For the sake of reducing parameter storage, we conduct Symmetric MaxValue quantization for parameters of two FC layers.

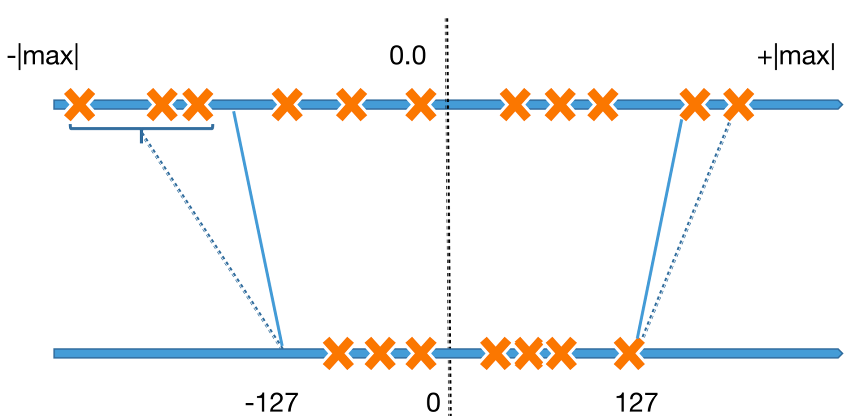

Figure 3: The diagram of symmetric KL-divergence based quantization while fixed-point number is 8.

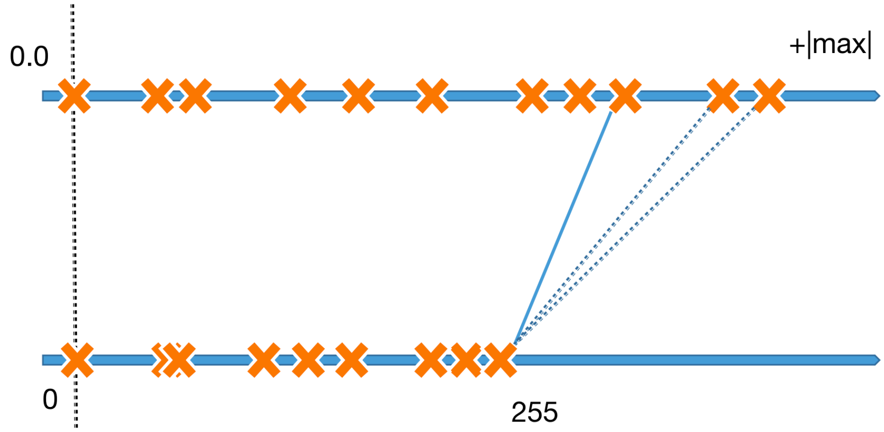

Figure 4: The diagram of asymmetric KL-divergence based quantization while fixed-point number is 8.

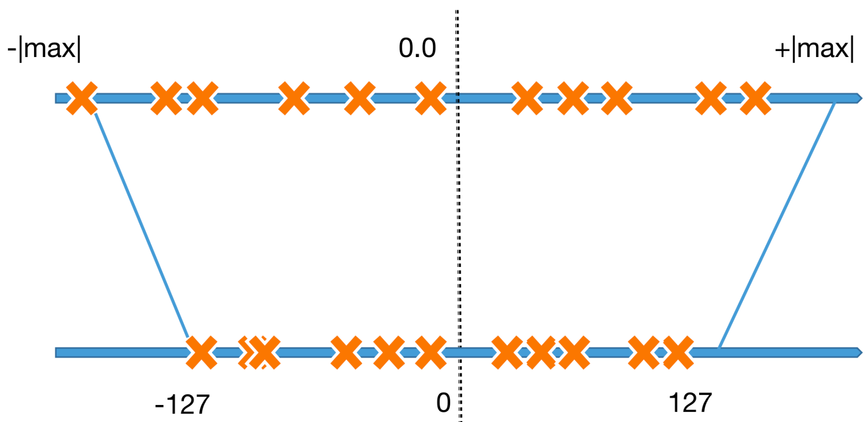

Figure 5: The diagram of asymmetric KL-divergence based quantization while fixed-point number is 8.

4 Training & Testing

4.1 Training

Our training schedule consists of five sequential steps that are training model from scratch, pruning model, finetuning pruned model, quantizing model and finetuning quantized model. The training flowchart is showed in Fig. 7.

Training from scratch

During training on ImageNet, we follow standard practice and perform data augmentation with random-size cropping [5] to 224×224 and random horizontal flipping. The mean channel subtraction are performed on input images. The model is optimized by synchronous SGD with momentum 0.9 and a mini-batch size of 256. The initial learning rate is set to 0.1 and is adjusted according to cosine annealing strategy [4]. In addition, a warmup strategy is used in the beginning 5 epoch where the learning rate increases linearly from 0 to 0.1. The model is trained for 300 epoch from scratch with weight decay 4e-5. Label-smoothing regularization [6] is used.

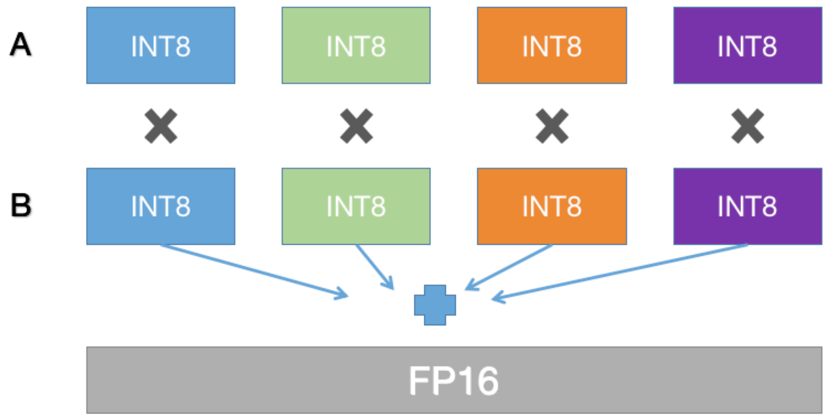

Figure 6: The diagram of fp16 accumulation matrix dot product

Pruning model and finetuning pruned model

We take sensitive analysis refer to Sec. 2 to prune previous well-trained model. In order to compensate the accuracy loss, we finetune the pruned model for 40 epochs with initial learning rate 1e-3 and decay by a factor of 0.1 at epoch 20. Other training settings are the same as training from scratch.

Quantizing model and finetuning quantized model

We randomly take 1000 images from training set as calibration set to calculate the appropriate steps of activation which need to be quantized. For the entry model, all the precision is set to 8 and the tolerance is set to 1.3. After that, we also require finetuning the quantized model to improve accuracy. We continue finetune it for 20 epochs with initial learning rate 5e-4, decayed by a factor of 0.2 at epoch 10.

4.2 Testing

When testing, we apply a centre crop evaluation on the validation set, where 224×224 pixels are cropped from each image whose shorter edge is first resized to 256. In the entry model, we eventually obtain 75.2% top-1 accuracy.

5 Scoring

The final score consists of two parts: (a) Parameter Storage (b) Math Operations. We implemented a tool for easily scoring. Once a model definition and sparsity file are given, the score can be figured out automatically.

5.1 Parameter Storage

In the entry model, we perform linear combination for all BatchNorm layers, integrating these parameters into their former convolutional layers or fully-connected layers. Here, we replace the last fully-connected layer (i.e. classifier layer) and fully-connected layers in SE-module with 1×1 convolutional layers. Consequently, all the convolutional kernels form the whole parameter set. And the size of parameter storage is affected by the degree of sparsity and fixed-point number of parameters. More concretely, the pseudo code of figuring out storage size is as follows:

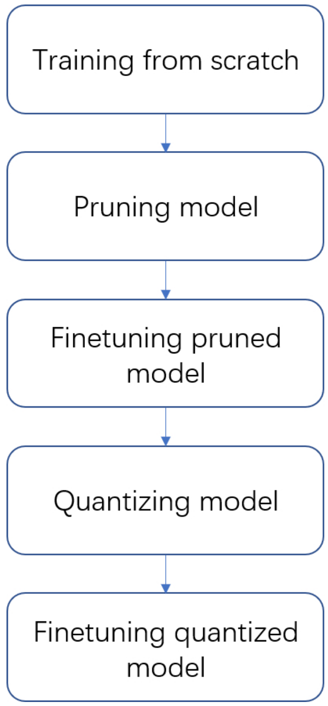

Figure 7: The training flow of our model

```
weight_count = c_out x c_in x k x k
storage = weight_count * weight_precision * (1 - layer_sparsity)
storage += weight_count  // binary mask storage
if use_bias:
    storage += output_size * bias_precision;
storage /= 32
```

5.2 Math Operations

Math Operations consists of multiplication operations and addition operations. According to the competition rule, we demonstrate the process of calculating math operations in a convolution as follows:

```
mul_bit = max(input_precision, weight_precision);
add_bit = accumulation_precision
vector_length = c_in * k * k * (1 - layer_sparsity);
output_count = c_out * h_out * w_out
mul_bitops = vector_length * output_count * mul_bit;
add_bitops = (vector_length - 1) * output_count * add_bit;
if use_bias
    add_bitops += output_count * add_bit;
math_ops = (mul_bitops + add_bitops) / 32
```

Other operation's flops calculation are almost the same to Convolution Example.

5.3 Score report

The score of entry model is **0.25097**. The parameter storage and math operations are as follows:

- Parameter storage: 0.825353 M
- Mul operations: 52.1957 M
- Add operations: 101.488 M

More details of scoring each layer are exhibited in Appendix B.

References

[1] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *ICLR*, 2019. 1

[2] Song Han, Jeff Pool, Sharan Narang, Huizi Mao, Enhao Gong, Shijian Tang, Erich Elsen, Peter Vajda, Manohar Paluri, John Tran, Bryan Catanzaro, and William J. Dally. Dsd: Dense-sparse-dense training for deep neural networks. In *ICLR*, 2017. 1

[3] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018. 1

[4] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *CoRR*, 2016. 5

[5] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 5

[6] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 5

[7] Mingxing Tan and Quoc V. Le. Mixconv: Mixed depthwise convolutional kernels. *arXiv preprint arXiv:1907.09595*, 2019. 1

A Appendix: Model Schema

The Fig. 8 shows the details of the entry model.

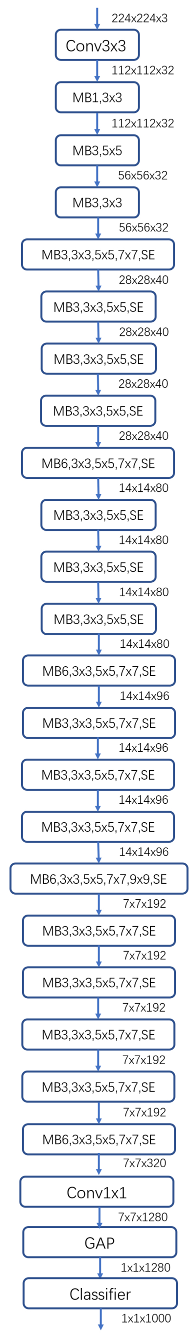

Figure 8: The overall schema of ProfitableNet.

B Scoring Details

The table 1 shows the details of scoring for each layer.

Table 1: score details

layername	type	input1	input2	weight	sparsity	mul	add	mulops(M)	addops(M)	storage(K)
conv1/ 3x3_s2	Conv	8	-	8	0.45	8	16	44.96	89.92	5.18
conv1/3x3_s2/ relu	ReLU	8	-	-	-	8	-	3.21	-	-
conv2_0/ 3x3_dwconv	Conv	8	-	8	0.00	8	16	28.90	57.80	2.82
conv2_0/3x3_dwconv/ relu	ReLU	8	-	-	-	8	-	3.21	-	-
conv2_0/ 1x1_decrease	Conv	8	-	8	0.00	8	16	51.38	102.76	4.35
conv3_0/ 1x1_increase	Conv	8	-	8	0.45	8	16	38.54	77.07	4.92
conv3_0/1x1_increase/ relu	ReLU	8	-	-	-	8	-	4.82	-	-
conv3_0/ 3x3_dwconv	Conv	8	-	8	0.00	8	16	30.11	60.21	10.37
conv3_0/3x3_dwconv/ relu	ReLU	8	-	-	-	8	-	1.20	-	-
conv3_0/ 1x1_decrease	Conv	8	-	8	0.00	8	16	38.54	77.07	12.80
conv3_1/ 1x1_increase	Conv	8	-	8	0.45	8	16	40.94	81.89	18.12
conv3_1/1x1_increase/ relu	ReLU	8	-	-	-	8	-	2.41	-	-
conv3_1/ 3x3_dwconv	Conv	8	-	8	0.00	8	16	21.68	43.35	8.45
conv3_1/3x3_dwconv/ relu	ReLU	8	-	-	-	8	-	2.41	-	-
conv3_1/ 1x1_decrease	Conv	8	-	8	0.45	8	16	41.75	83.49	17.10
conv3_1/ elt_sum	Elt	8	16	-	-	-	16	-	1.61	-
conv4_0/ 1x1_increase	Conv	8	-	8	0.45	8	16	40.94	81.89	18.12
conv4_0/1x1_increase/ relu	ReLU	8	-	-	-	8	-	2.41	-	-
conv4_0/3x3_dwconv/ slice_0	Conv	8	-	8	0.00	8	16	1.81	3.61	2.82
conv4_0/3x3_dwconv/ slice_0/relu	ReLU	8	-	-	-	8	-	0.20	-	-
conv4_0/3x3_dwconv/ slice_1	Conv	8	-	8	0.00	8	16	5.02	10.04	6.91
conv4_0/3x3_dwconv/ slice_1/relu	ReLU	8	-	-	-	8	-	0.20	-	-
conv4_0/3x3_dwconv/ slice_2	Conv	8	-	8	0.00	8	16	9.83	19.67	13.06
conv4_0/3x3_dwconv/ slice_2/relu	ReLU	8	-	-	-	8	-	0.20	-	-
conv4_0/3x3_dwconv/ SE_global_pool	Pool	8	-	-	-	8	16	0.00	1.20	-
conv4_0/3x3_dwconv/ SE_fc1	Conv	16	-	8	0.00	16	16	0.01	0.01	4.61
conv4_0/3x3_dwconv/ SE_fc1/relu	ReLU	16	-	-	-	16	-	0.00	-	-
conv4_0/3x3_dwconv/ SE_fc2	Conv	16	-	8	0.00	16	16	0.01	0.01	4.61
conv4_0/3x3_dwconv/ SE_weights	Sigmoid	16	-	-	-	16	16	0.00	0.00	-
conv4_0/3x3_dwconv/ scale	Scale	8	16	-	-	16	-	1.20	-	-
conv4_0/ 1x1_decrease	Conv	8	-	8	0.00	8	16	24.08	48.17	31.36
conv4_1/ 1x1_increase	Conv	8	-	8	0.45	8	16	16.56	33.12	27.84
conv4_1/1x1_increase/ relu	ReLU	8	-	-	-	8	-	0.75	-	-
conv4_1/3x3_dwconv/ slice_0	Conv	8	-	8	0.40	8	16	1.88	3.76	4.09
conv4_1/3x3_dwconv/ slice_0/relu	ReLU	8	-	-	-	8	-	0.38	-	-

layername	type	input1	input2	weight	sparsity	mul	add	mulops(M)	addops(M)	storage(K)
conv4_1/3x3_dwconv/ slice_1	Conv	8	-	8	0.40	8	16	5.64	11.29	9.66
conv4_1/3x3_dwconv/ slice_1/relu	ReLU	8	-	-	-	8	-	0.38	-	-
conv4_1/3x3_dwconv/ SE_global_pool	Pool	8	-	-	-	8	16	0.00	1.50	-
conv4_1/3x3_dwconv/ SE_fc1	Conv	16	-	8	0.00	16	16	0.01	0.01	6.72
conv4_1/3x3_dwconv/ SE_fc1/relu	ReLU	16	-	-	-	16	-	0.00	-	-
conv4_1/3x3_dwconv/ SE_fc2	Conv	16	-	8	0.00	16	16	0.01	0.01	6.72
conv4_1/3x3_dwconv/ SE_weights	Sigmoid	16	-	-	-	16	16	0.00	0.00	-
conv4_1/3x3_dwconv/ scale	Scale	8	16	-	-	16	-	1.51	-	-
conv4_1/ 1x1_decrease	Conv	8	-	8	0.45	8	16	16.56	33.12	26.56
conv4_1/ elt_sum	Elt	8	16	-	-	-	16	-	0.50	-
conv4_2/ 1x1_increase	Conv	8	-	8	0.45	8	16	16.56	33.12	27.84
conv4_2/1x1_increase/ relu	ReLU	8	-	-	-	8	-	0.75	-	-
conv4_2/3x3_dwconv/ slice_0	Conv	8	-	8	0.40	8	16	1.88	3.76	4.09
conv4_2/3x3_dwconv/ slice_0/relu	ReLU	8	-	-	-	8	-	0.38	-	-
conv4_2/3x3_dwconv/ slice_1	Conv	8	-	8	0.40	8	16	5.64	11.29	9.66
conv4_2/3x3_dwconv/ slice_1/relu	ReLU	8	-	-	-	8	-	0.38	-	-
conv4_2/3x3_dwconv/ SE_global_pool	Pool	8	-	-	-	8	16	0.00	1.50	-
conv4_2/3x3_dwconv/ SE_fc1	Conv	16	-	8	0.00	16	16	0.01	0.01	6.72
conv4_2/3x3_dwconv/ SE_fc1/relu	ReLU	16	-	-	-	16	-	0.00	-	-
conv4_2/3x3_dwconv/ SE_fc2	Conv	16	-	8	0.00	16	16	0.01	0.01	6.72
conv4_2/3x3_dwconv/ SE_weights	Sigmoid	16	-	-	-	16	16	0.00	0.00	-
conv4_2/3x3_dwconv/ scale	Scale	8	16	-	-	16	-	1.51	-	-
conv4_2/ 1x1_decrease	Conv	8	-	8	0.45	8	16	16.56	33.12	26.56
conv4_2/ elt_sum	Elt	8	16	-	-	-	16	-	0.50	-
conv4_3/ 1x1_increase	Conv	8	-	8	0.45	8	16	16.56	33.12	27.84
conv4_3/1x1_increase/ relu	ReLU	8	-	-	-	8	-	0.75	-	-
conv4_3/3x3_dwconv/ slice_0	Conv	8	-	8	0.40	8	16	1.88	3.76	4.09
conv4_3/3x3_dwconv/ slice_0/relu	ReLU	8	-	-	-	8	-	0.38	-	-
conv4_3/3x3_dwconv/ slice_1	Conv	8	-	8	0.40	8	16	5.64	11.29	9.66
conv4_3/3x3_dwconv/ slice_1/relu	ReLU	8	-	-	-	8	-	0.38	-	-
conv4_3/3x3_dwconv/ SE_global_pool	Pool	8	-	-	-	8	16	0.00	1.50	-
conv4_3/3x3_dwconv/ SE_fc1	Conv	16	-	8	0.00	16	16	0.01	0.01	6.72
conv4_3/3x3_dwconv/ SE_fc1/relu	ReLU	16	-	-	-	16	-	0.00	-	-
conv4_3/3x3_dwconv/ SE_fc2	Conv	16	-	8	0.00	16	16	0.01	0.01	6.72

layername	type	input1	input2	weight	sparsity	mul	add	mulops(M)	addops(M)	storage(K)
conv4_3/3x3_dwconv/ SE_weights	Sigmoid	16	-	-	-	16	16	0.00	0.00	-
conv4_3/3x3_dwconv/ scale	Scale	8	16	-	-	16	-	1.51	-	-
conv4_3/ 1x1_decrease	Conv	8	-	8	0.45	8	16	16.56	33.12	26.56
conv4_3/ elt_sum	Elt	8	16	-	-	-	16	-	0.50	-
conv5_0/ 1x1_increase	Conv	8	-	8	0.45	8	16	33.12	66.23	55.68
conv5_0/1x1_increase/ relu	ReLU	8	-	-	-	8	-	1.51	-	-
conv5_0/3x3_dwconv/ slice_0	Conv	8	-	8	0.00	8	16	1.13	2.26	7.04
conv5_0/3x3_dwconv/ slice_0/relu	ReLU	8	-	-	-	8	-	0.13	-	-
conv5_0/3x3_dwconv/ slice_1	Conv	8	-	8	0.00	8	16	3.14	6.27	17.28
conv5_0/3x3_dwconv/ slice_1/relu	ReLU	8	-	-	-	8	-	0.13	-	-
conv5_0/3x3_dwconv/ slice_2	Conv	8	-	8	0.00	8	16	6.15	12.29	32.64
conv5_0/3x3_dwconv/ slice_2/relu	ReLU	8	-	-	-	8	-	0.13	-	-
conv5_0/3x3_dwconv/ SE_global_pool	Pool	8	-	-	-	8	16	0.00	0.75	-
conv5_0/3x3_dwconv/ SE_fc1	Conv	16	-	8	0.00	16	16	0.06	0.06	28.80
conv5_0/3x3_dwconv/ SE_fc1/relu	ReLU	16	-	-	-	16	-	0.00	-	-
conv5_0/3x3_dwconv/ SE_fc2	Conv	16	-	8	0.00	16	16	0.06	0.05	28.80
conv5_0/3x3_dwconv/ SE_weights	Sigmoid	16	-	-	-	16	16	0.01	0.00	-
conv5_0/3x3_dwconv/ scale	Scale	8	16	-	-	16	-	0.75	-	-
conv5_0/ 1x1_decrease	Conv	8	-	8	0.00	8	16	30.11	60.21	154.88
conv5_1/ 1x1_increase	Conv	8	-	8	0.45	8	16	16.56	33.12	107.52
conv5_1/1x1_increase/ relu	ReLU	8	-	-	-	8	-	0.38	-	-
conv5_1/3x3_dwconv/ slice_0	Conv	8	-	8	0.40	8	16	0.94	1.88	8.18
conv5_1/3x3_dwconv/ slice_0/relu	ReLU	8	-	-	-	8	-	0.19	-	-
conv5_1/3x3_dwconv/ slice_1	Conv	8	-	8	0.40	8	16	2.82	5.64	19.32
conv5_1/3x3_dwconv/ slice_1/relu	ReLU	8	-	-	-	8	-	0.19	-	-
conv5_1/3x3_dwconv/ SE_global_pool	Pool	8	-	-	-	8	16	0.00	0.75	-
conv5_1/3x3_dwconv/ SE_fc1	Conv	16	-	8	0.00	16	16	0.06	0.06	28.80
conv5_1/3x3_dwconv/ SE_fc1/relu	ReLU	16	-	-	-	16	-	0.00	-	-
conv5_1/3x3_dwconv/ SE_fc2	Conv	16	-	8	0.00	16	16	0.06	0.05	28.80
conv5_1/3x3_dwconv/ SE_weights	Sigmoid	16	-	-	-	16	16	0.01	0.00	-
conv5_1/3x3_dwconv/ scale	Scale	8	16	-	-	16	-	0.75	-	-
conv5_1/ 1x1_decrease	Conv	8	-	8	0.45	8	16	16.56	33.12	104.96
conv5_1/ elt_sum	Elt	8	16	-	-	-	16	-	0.25	-
conv5_2/ 1x1_increase	Conv	8	-	8	0.45	8	16	16.56	33.12	107.52
conv5_2/1x1_increase/ relu	ReLU	8	-	-	-	8	-	0.38	-	-

layername	type	input1	input2	weight	sparsity	mul	add	mulops(M)	addops(M)	storage(K)
conv5_2/3x3_dwconv/ slice_0	Conv	8	-	8	0.40	8	16	0.94	1.88	8.18
conv5_2/3x3_dwconv/ slice_0/relu	ReLU	8	-	-	-	8	-	0.19	-	-
conv5_2/3x3_dwconv/ slice_1	Conv	8	-	8	0.40	8	16	2.82	5.64	19.32
conv5_2/3x3_dwconv/ slice_1/relu	ReLU	8	-	-	-	8	-	0.19	-	-
conv5_2/3x3_dwconv/ SE_global_pool	Pool	8	-	-	-	8	16	0.00	0.75	-
conv5_2/3x3_dwconv/ SE_fc1	Conv	16	-	8	0.00	16	16	0.06	0.06	28.80
conv5_2/3x3_dwconv/ SE_fc1/relu	ReLU	16	-	-	-	16	-	0.00	-	-
conv5_2/3x3_dwconv/ SE_fc2	Conv	16	-	8	0.00	16	16	0.06	0.05	28.80
conv5_2/3x3_dwconv/ SE_weights	Sigmoid	16	-	-	-	16	16	0.01	0.00	-
conv5_2/3x3_dwconv/ scale	Scale	8	16	-	-	16	-	0.75	-	-
conv5_2/ 1x1_decrease	Conv	8	-	8	0.45	8	16	16.56	33.12	104.96
conv5_2/ elt_sum	Elt	8	16	-	-	-	16	-	0.25	-
conv5_3/ 1x1_increase	Conv	8	-	8	0.45	8	16	16.56	33.12	107.52
conv5_3/1x1_increase/ relu	ReLU	8	-	-	-	8	-	0.38	-	-
conv5_3/3x3_dwconv/ slice_0	Conv	8	-	8	0.40	8	16	0.94	1.88	8.18
conv5_3/3x3_dwconv/ slice_0/relu	ReLU	8	-	-	-	8	-	0.19	-	-
conv5_3/3x3_dwconv/ slice_1	Conv	8	-	8	0.40	8	16	2.82	5.64	19.32
conv5_3/3x3_dwconv/ slice_1/relu	ReLU	8	-	-	-	8	-	0.19	-	-
conv5_3/3x3_dwconv/ SE_global_pool	Pool	8	-	-	-	8	16	0.00	0.75	-
conv5_3/3x3_dwconv/ SE_fc1	Conv	16	-	8	0.00	16	16	0.06	0.06	28.80
conv5_3/3x3_dwconv/ SE_fc1/relu	ReLU	16	-	-	-	16	-	0.00	-	-
conv5_3/3x3_dwconv/ SE_fc2	Conv	16	-	8	0.00	16	16	0.06	0.05	28.80
conv5_3/3x3_dwconv/ SE_weights	Sigmoid	16	-	-	-	16	16	0.01	0.00	-
conv5_3/3x3_dwconv/ scale	Scale	8	16	-	-	16	-	0.75	-	-
conv5_3/ 1x1_decrease	Conv	8	-	8	0.45	8	16	16.56	33.12	104.96
conv5_3/ elt_sum	Elt	8	16	-	-	-	16	-	0.25	-
conv6_0/ 1x1_increase	Conv	8	-	8	0.45	8	16	33.12	66.23	215.04
conv6_0/1x1_increase/ relu	ReLU	8	-	-	-	8	-	0.75	-	-
conv6_0/3x3_dwconv/ slice_0	Conv	8	-	8	0.00	8	16	2.26	4.52	14.08
conv6_0/3x3_dwconv/ slice_0/relu	ReLU	8	-	-	-	8	-	0.25	-	-
conv6_0/3x3_dwconv/ slice_1	Conv	8	-	8	0.00	8	16	6.27	12.54	34.56
conv6_0/3x3_dwconv/ slice_1/relu	ReLU	8	-	-	-	8	-	0.25	-	-
conv6_0/3x3_dwconv/ slice_2	Conv	8	-	8	0.00	8	16	12.29	24.59	65.28
conv6_0/3x3_dwconv/ slice_2/relu	ReLU	8	-	-	-	8	-	0.25	-	-

layername	type	input1	input2	weight	sparsity	mul	add	mulops(M)	addops(M)	storage(K)
conv6_0/3x3_dwconv/ SE_global_pool	Pool	8	-	-	-	8	16	0.00	1.50	-
conv6_0/3x3_dwconv/ SE_fc1	Conv	16	-	8	0.00	16	16	0.23	0.23	115.20
conv6_0/3x3_dwconv/ SE_fc1/relu	ReLU	16	-	-	-	16	-	0.00	-	-
conv6_0/3x3_dwconv/ SE_fc2	Conv	16	-	8	0.00	16	16	0.23	0.22	115.20
conv6_0/3x3_dwconv/ SE_weights	Sigmoid	16	-	-	-	16	16	0.02	0.01	-
conv6_0/3x3_dwconv/ scale	Scale	8	16	-	-	16	-	1.51	-	-
conv6_0/ 1x1_decrease	Conv	8	-	8	0.00	8	16	72.25	144.51	370.18
conv6_1/ 1x1_increase	Conv	8	-	8	0.45	8	16	23.48	46.96	153.91
conv6_1/1x1_increase/ relu	ReLU	8	-	-	-	8	-	0.45	-	-
conv6_1/3x3_dwconv/ slice_0	Conv	8	-	8	0.40	8	16	0.75	1.51	6.55
conv6_1/3x3_dwconv/ slice_0/relu	ReLU	8	-	-	-	8	-	0.15	-	-
conv6_1/3x3_dwconv/ slice_1	Conv	8	-	8	0.40	8	16	2.26	4.52	15.46
conv6_1/3x3_dwconv/ slice_1/relu	ReLU	8	-	-	-	8	-	0.15	-	-
conv6_1/3x3_dwconv/ slice_2	Conv	8	-	8	0.40	8	16	4.37	8.73	28.82
conv6_1/3x3_dwconv/ slice_2/relu	ReLU	8	-	-	-	8	-	0.15	-	-
conv6_1/3x3_dwconv/ SE_global_pool	Pool	8	-	-	-	8	16	0.00	0.90	-
conv6_1/3x3_dwconv/ SE_fc1	Conv	16	-	8	0.00	16	16	0.08	0.08	41.47
conv6_1/3x3_dwconv/ SE_fc1/relu	ReLU	16	-	-	-	16	-	0.00	-	-
conv6_1/3x3_dwconv/ SE_fc2	Conv	16	-	8	0.00	16	16	0.08	0.08	41.47
conv6_1/3x3_dwconv/ SE_weights	Sigmoid	16	-	-	-	16	16	0.01	0.00	-
conv6_1/3x3_dwconv/ scale	Scale	8	16	-	-	16	-	0.90	-	-
conv6_1/ 1x1_decrease	Conv	8	-	8	0.45	8	16	23.78	47.57	150.84
conv6_1/ elt_sum	Elt	8	16	-	-	-	16	-	0.30	-
conv6_2/ 1x1_increase	Conv	8	-	8	0.45	8	16	23.48	46.96	153.91
conv6_2/1x1_increase/ relu	ReLU	8	-	-	-	8	-	0.45	-	-
conv6_2/3x3_dwconv/ slice_0	Conv	8	-	8	0.40	8	16	0.75	1.51	6.55
conv6_2/3x3_dwconv/ slice_0/relu	ReLU	8	-	-	-	8	-	0.15	-	-
conv6_2/3x3_dwconv/ slice_1	Conv	8	-	8	0.40	8	16	2.26	4.52	15.46
conv6_2/3x3_dwconv/ slice_1/relu	ReLU	8	-	-	-	8	-	0.15	-	-
conv6_2/3x3_dwconv/ slice_2	Conv	8	-	8	0.40	8	16	4.37	8.73	28.82
conv6_2/3x3_dwconv/ slice_2/relu	ReLU	8	-	-	-	8	-	0.15	-	-
conv6_2/3x3_dwconv/ SE_global_pool	Pool	8	-	-	-	8	16	0.00	0.90	-
conv6_2/3x3_dwconv/ SE_fc1	Conv	16	-	8	0.00	16	16	0.08	0.08	41.47

layername	type	input1	input2	weight	sparsity	mul	add	mulops(M)	addops(M)	storage(K)
conv6_2/3x3_dwconv/ SE_fc1/relu	ReLU	16	-	-	-	16	-	0.00	-	-
conv6_2/3x3_dwconv/ SE_fc2	Conv	16	-	8	0.00	16	16	0.08	0.08	41.47
conv6_2/3x3_dwconv/ SE_weights	Sigmoid	16	-	-	-	16	16	0.01	0.00	-
conv6_2/3x3_dwconv/ scale	Scale	8	16	-	-	16	-	0.90	-	-
conv6_2/ 1x1_decrease	Conv	8	-	8	0.45	8	16	23.78	47.57	150.84
conv6_2/ elt_sum	Elt	8	16	-	-	-	16	-	0.30	-
conv6_3/ 1x1_increase	Conv	8	-	8	0.45	8	16	23.48	46.96	153.91
conv6_3/1x1_increase/ relu	ReLU	8	-	-	-	8	-	0.45	-	-
conv6_3/3x3_dwconv/ slice_0	Conv	8	-	8	0.40	8	16	0.75	1.51	6.55
conv6_3/3x3_dwconv/ slice_0/relu	ReLU	8	-	-	-	8	-	0.15	-	-
conv6_3/3x3_dwconv/ slice_1	Conv	8	-	8	0.40	8	16	2.26	4.52	15.46
conv6_3/3x3_dwconv/ slice_1/relu	ReLU	8	-	-	-	8	-	0.15	-	-
conv6_3/3x3_dwconv/ slice_2	Conv	8	-	8	0.40	8	16	4.37	8.73	28.82
conv6_3/3x3_dwconv/ slice_2/relu	ReLU	8	-	-	-	8	-	0.15	-	-
conv6_3/3x3_dwconv/ SE_global_pool	Pool	8	-	-	-	8	16	0.00	0.90	-
conv6_3/3x3_dwconv/ SE_fc1	Conv	16	-	8	0.00	16	16	0.08	0.08	41.47
conv6_3/3x3_dwconv/ SE_fc1/relu	ReLU	16	-	-	-	16	-	0.00	-	-
conv6_3/3x3_dwconv/ SE_fc2	Conv	16	-	8	0.00	16	16	0.08	0.08	41.47
conv6_3/3x3_dwconv/ SE_weights	Sigmoid	16	-	-	-	16	16	0.01	0.00	-
conv6_3/3x3_dwconv/ scale	Scale	8	16	-	-	16	-	0.90	-	-
conv6_3/ 1x1_decrease	Conv	8	-	8	0.45	8	16	23.78	47.57	150.84
conv6_3/ elt_sum	Elt	8	16	-	-	-	16	-	0.30	-
conv7_0/ 1x1_increase	Conv	8	-	8	0.45	8	16	46.96	93.93	307.81
conv7_0/1x1_increase/ relu	ReLU	8	-	-	-	8	-	0.90	-	-
conv7_0/3x3_dwconv/ slice_0	Conv	8	-	8	0.00	8	16	0.51	1.02	12.67
conv7_0/3x3_dwconv/ slice_0/relu	ReLU	8	-	-	-	8	-	0.06	-	-
conv7_0/3x3_dwconv/ slice_1	Conv	8	-	8	0.00	8	16	1.41	2.82	31.10
conv7_0/3x3_dwconv/ slice_1/relu	ReLU	8	-	-	-	8	-	0.06	-	-
conv7_0/3x3_dwconv/ slice_2	Conv	8	-	8	0.00	8	16	2.77	5.53	58.75
conv7_0/3x3_dwconv/ slice_2/relu	ReLU	8	-	-	-	8	-	0.06	-	-
conv7_0/3x3_dwconv/ slice_3	Conv	8	-	8	0.00	8	16	4.57	9.14	95.62
conv7_0/3x3_dwconv/ slice_3/relu	ReLU	8	-	-	-	8	-	0.06	-	-
conv7_0/3x3_dwconv/ SE_global_pool	Pool	8	-	-	-	8	16	0.00	0.44	-
conv7_0/3x3_dwconv/ SE_fc1	Conv	16	-	8	0.00	16	16	0.33	0.33	165.89

layername	type	input1	input2	weight	sparsity	mul	add	mulops(M)	addops(M)	storage(K)
conv7_0/3x3_dwconv/ SE_fc1/relu	ReLU	16	-	-	-	16	-	0.00	-	-
conv7_0/3x3_dwconv/ SE_fc2	Conv	16	-	8	0.00	16	16	0.33	0.32	165.89
conv7_0/3x3_dwconv/ SE_weights	Sigmoid	16	-	-	-	16	16	0.02	0.01	-
conv7_0/3x3_dwconv/ scale	Scale	8	16	-	-	16	-	0.45	-	-
conv7_0/ 1x1_decrease	Conv	8	-	8	0.00	8	16	43.35	86.70	887.81
conv7_1/ 1x1_increase	Conv	8	-	8	0.45	8	16	23.71	47.42	606.41
conv7_1/1x1_increase/ relu	ReLU	8	-	-	-	8	-	0.23	-	-
conv7_1/3x3_dwconv/ slice_0	Conv	8	-	8	0.40	8	16	0.38	0.75	13.09
conv7_1/3x3_dwconv/ slice_0/relu	ReLU	8	-	-	-	8	-	0.08	-	-
conv7_1/3x3_dwconv/ slice_1	Conv	8	-	8	0.40	8	16	1.13	2.26	30.91
conv7_1/3x3_dwconv/ slice_1/relu	ReLU	8	-	-	-	8	-	0.08	-	-
conv7_1/3x3_dwconv/ slice_2	Conv	8	-	8	0.40	8	16	2.18	4.37	57.64
conv7_1/3x3_dwconv/ slice_2/relu	ReLU	8	-	-	-	8	-	0.08	-	-
conv7_1/3x3_dwconv/ SE_global_pool	Pool	8	-	-	-	8	16	0.00	0.44	-
conv7_1/3x3_dwconv/ SE_fc1	Conv	16	-	8	0.00	16	16	0.33	0.33	165.89
conv7_1/3x3_dwconv/ SE_fc1/relu	ReLU	16	-	-	-	16	-	0.00	-	-
conv7_1/3x3_dwconv/ SE_fc2	Conv	16	-	8	0.00	16	16	0.33	0.32	165.89
conv7_1/3x3_dwconv/ SE_weights	Sigmoid	16	-	-	-	16	16	0.02	0.01	-
conv7_1/3x3_dwconv/ scale	Scale	8	16	-	-	16	-	0.45	-	-
conv7_1/ 1x1_decrease	Conv	8	-	8	0.45	8	16	23.78	47.57	600.27
conv7_1/ elt_sum	Elt	8	16	-	-	-	16	-	0.15	-
conv7_2/ 1x1_increase	Conv	8	-	8	0.45	8	16	23.71	47.42	606.41
conv7_2/1x1_increase/ relu	ReLU	8	-	-	-	8	-	0.23	-	-
conv7_2/3x3_dwconv/ slice_0	Conv	8	-	8	0.40	8	16	0.38	0.75	13.09
conv7_2/3x3_dwconv/ slice_0/relu	ReLU	8	-	-	-	8	-	0.08	-	-
conv7_2/3x3_dwconv/ slice_1	Conv	8	-	8	0.40	8	16	1.13	2.26	30.91
conv7_2/3x3_dwconv/ slice_1/relu	ReLU	8	-	-	-	8	-	0.08	-	-
conv7_2/3x3_dwconv/ slice_2	Conv	8	-	8	0.40	8	16	2.18	4.37	57.64
conv7_2/3x3_dwconv/ slice_2/relu	ReLU	8	-	-	-	8	-	0.08	-	-
conv7_2/3x3_dwconv/ SE_global_pool	Pool	8	-	-	-	8	16	0.00	0.44	-
conv7_2/3x3_dwconv/ SE_fc1	Conv	16	-	8	0.00	16	16	0.33	0.33	165.89
conv7_2/3x3_dwconv/ SE_fc1/relu	ReLU	16	-	-	-	16	-	0.00	-	-
conv7_2/3x3_dwconv/ SE_fc2	Conv	16	-	8	0.00	16	16	0.33	0.32	165.89

layername	type	input1	input2	weight	sparsity	mul	add	mulops(M)	addops(M)	storage(K)
conv7_2/3x3_dwconv/ SE_weights	Sigmoid	16	-	-	-	16	16	0.02	0.01	-
conv7_2/3x3_dwconv/ scale	Scale	8	16	-	-	16	-	0.45	-	-
conv7_2/ 1x1_decrease	Conv	8	-	8	0.45	8	16	23.78	47.57	600.27
conv7_2/ elt_sum	Elt	8	16	-	-	-	16	-	0.15	-
conv7_3/ 1x1_increase	Conv	8	-	8	0.45	8	16	23.71	47.42	606.41
conv7_3/1x1_increase/ relu	ReLU	8	-	-	-	8	-	0.23	-	-
conv7_3/3x3_dwconv/ slice_0	Conv	8	-	8	0.40	8	16	0.38	0.75	13.09
conv7_3/3x3_dwconv/ slice_0/relu	ReLU	8	-	-	-	8	-	0.08	-	-
conv7_3/3x3_dwconv/ slice_1	Conv	8	-	8	0.40	8	16	1.13	2.26	30.91
conv7_3/3x3_dwconv/ slice_1/relu	ReLU	8	-	-	-	8	-	0.08	-	-
conv7_3/3x3_dwconv/ slice_2	Conv	8	-	8	0.40	8	16	2.18	4.37	57.64
conv7_3/3x3_dwconv/ slice_2/relu	ReLU	8	-	-	-	8	-	0.08	-	-
conv7_3/3x3_dwconv/ SE_global_pool	Pool	8	-	-	-	8	16	0.00	0.44	-
conv7_3/3x3_dwconv/ SE_fc1	Conv	16	-	8	0.00	16	16	0.33	0.33	165.89
conv7_3/3x3_dwconv/ SE_fc1/relu	ReLU	16	-	-	-	16	-	0.00	-	-
conv7_3/3x3_dwconv/ SE_fc2	Conv	16	-	8	0.00	16	16	0.33	0.32	165.89
conv7_3/3x3_dwconv/ SE_weights	Sigmoid	16	-	-	-	16	16	0.02	0.01	-
conv7_3/3x3_dwconv/ scale	Scale	8	16	-	-	16	-	0.45	-	-
conv7_3/ 1x1_decrease	Conv	8	-	8	0.45	8	16	23.78	47.57	600.27
conv7_3/ elt_sum	Elt	8	16	-	-	-	16	-	0.15	-
conv7_4/ 1x1_increase	Conv	8	-	8	0.45	8	16	23.71	47.42	606.41
conv7_4/1x1_increase/ relu	ReLU	8	-	-	-	8	-	0.23	-	-
conv7_4/3x3_dwconv/ slice_0	Conv	8	-	8	0.40	8	16	0.38	0.75	13.09
conv7_4/3x3_dwconv/ slice_0/relu	ReLU	8	-	-	-	8	-	0.08	-	-
conv7_4/3x3_dwconv/ slice_1	Conv	8	-	8	0.40	8	16	1.13	2.26	30.91
conv7_4/3x3_dwconv/ slice_1/relu	ReLU	8	-	-	-	8	-	0.08	-	-
conv7_4/3x3_dwconv/ slice_2	Conv	8	-	8	0.40	8	16	2.18	4.37	57.64
conv7_4/3x3_dwconv/ slice_2/relu	ReLU	8	-	-	-	8	-	0.08	-	-
conv7_4/3x3_dwconv/ SE_global_pool	Pool	8	-	-	-	8	16	0.00	0.44	-
conv7_4/3x3_dwconv/ SE_fc1	Conv	16	-	8	0.00	16	16	0.33	0.33	165.89
conv7_4/3x3_dwconv/ SE_fc1/relu	ReLU	16	-	-	-	16	-	0.00	-	-
conv7_4/3x3_dwconv/ SE_fc2	Conv	16	-	8	0.00	16	16	0.33	0.32	165.89
conv7_4/3x3_dwconv/ SE_weights	Sigmoid	16	-	-	-	16	16	0.02	0.01	-
conv7_4/3x3_dwconv/ scale	Scale	8	16	-	-	16	-	0.45	-	-

layername	type	input1	input2	weight	sparsity	mul	add	mulops(M)	addops(M)	storage(K)
conv7_4/ 1x1_decrease	Conv	8	-	8	0.45	8	16	23.78	47.57	600.27
conv7_4/ elt_sum	Elt	8	16	-	-	-	16	-	0.15	-
conv8_0/ 1x1_increase	Conv	8	-	8	0.45	8	16	47.42	94.83	1212.83
conv8_0/1x1_increase/ relu	ReLU	8	-	-	-	8	-	0.45	-	-
conv8_0/3x3_dwconv/ slice_0	Conv	8	-	8	0.00	8	16	1.35	2.71	33.79
conv8_0/3x3_dwconv/ slice_0/relu	ReLU	8	-	-	-	8	-	0.15	-	-
conv8_0/3x3_dwconv/ slice_1	Conv	8	-	8	0.00	8	16	3.76	7.53	82.94
conv8_0/3x3_dwconv/ slice_1/relu	ReLU	8	-	-	-	8	-	0.15	-	-
conv8_0/3x3_dwconv/ slice_2	Conv	8	-	8	0.00	8	16	7.38	14.75	156.67
conv8_0/3x3_dwconv/ slice_2/relu	ReLU	8	-	-	-	8	-	0.15	-	-
conv8_0/3x3_dwconv/ SE_global_pool	Pool	8	-	-	-	8	16	0.01	0.88	-
conv8_0/3x3_dwconv/ SE_fc1	Conv	16	-	8	0.00	16	16	1.33	1.33	663.55
conv8_0/3x3_dwconv/ SE_fc1/relu	ReLU	16	-	-	-	16	-	0.00	-	-
conv8_0/3x3_dwconv/ SE_fc2	Conv	16	-	8	0.00	16	16	1.33	1.31	663.55
conv8_0/3x3_dwconv/ SE_weights	Sigmoid	16	-	-	-	16	16	0.04	0.02	-
conv8_0/3x3_dwconv/ scale	Scale	8	16	-	-	16	-	0.90	-	-
conv8_0/ 1x1_decrease	Conv	8	-	8	0.00	8	16	144.51	289.01	2954.24
conv9/1x1	Conv	8	-	8	0.50	8	16	80.28	160.56	2068.48
conv9/1x1/ relu	ReLU	8	-	-	-	8	-	0.50	-	-
avepool/ 7x7	Pool	8	-	-	-	8	16	0.01	0.98	-
classifier	Conv	8	-	8	0.50	8	16	5.12	10.24	6416.00