Accelerating Convolutional Neural Networks via Activation Map Compression Supplementary Material

Georgios Georgiadis
Samsung Semiconductor, Inc.
g.geor@samsung.com

In the supplementary material, we discuss (1) the parameter choice of the regularization parameter α_l for sparsification for each layer l of a network (Sec. 1) and (2) present the exponential-Golomb algorithm for completeness (Sec. 2).

1. Regularization parameter

We present the regularization parameters per layer for each network in Tables 1, 2, 3. While we finely-tuned the regularization parameter of each layer for the smaller networks (LeNet-5, MobileNet-V1), we chose a global regularization parameter for the bigger ones (Inception-V3, ResNet-18, ResNet-34). Finely-tuning the regularization of each layer can yield further sparsity gains, but also adds an intensive search over the parameter space. Instead, for the bigger networks, we showed that even one global regularization parameter per network is sufficient to successfully sparsify a model. For layers not shown in the tables, the regularization parameter was set to 0. Note that we never regularize the final layer L of each network.

LeNet-5	conv1	conv2	fc1
α_l	0.25×10^{-5}	2.00×10^{-5}	5.00×10^{-5}

Table 1: LeNet-5 [3] regularization parameters.

MobileNet-V1	Conv/s2	Conv dw/s1 Conv / s1	Conv dw/s2 Conv / s1	Conv dw/s1 Conv / s1	Conv dw/s2 Conv / s1	Conv dw/s1 Conv / s1	Conv dw/s2 Conv / s1	5× Conv dw / s1 Conv / s1	Conv dw/s2 Conv / s1	Conv dw/s2 Conv / s1
α_l	15×10^{-8}	15×10^{-8}	15×10^{-8}	15×10^{-8}	1×10^{-8}	1×10^{-8}	1×10^{-8}	1×10^{-8}	2×10^{-8}	2×10^{-8}

Table 2: MobileNet-V1 [2] regularization parameters.

Model	**Variant**	α_l $(l = 1, \ldots, L-1)$
Inception-V3	Sparse	1×10^{-8}
	Sparse_v2	1×10^{-7}
ResNet-18	Sparse	1×10^{-8}
	Sparse_v2	1×10^{-7}
ResNet-34	Sparse	1×10^{-8}
	Sparse_v2	5×10^{-8}

Table 3: Inception-V3 [4] and the ResNet-18/34 [1] regularization parameters.

2. Exponential-Golomb

We provide pseudo-code for the encoding and decoding algorithms of exponential-Golomb [5] in Alg. 1.

Algorithm 1 Exponential-Golomb

```
Input: Non-negative integer x, Order k
Output: Bitstream y
function encode_exp_Golomb (x, k)
{
  If k == 0:
    y = encode_exp_Golomb_0_order(x)
  Else:
    q = floor(x/2^k)
    q_c = encode_exp_Golomb_0_order(q)
    r = x mod 2^k
    r_c = to_binary(r, k) // to_binary(r, k) converts r into binary using k bits.
    y = concatenate(q_c, r_c)
  Return y
}
Input: Bitstream x, Order k
Output: Non-negative integer y
function decode_exp_Golomb (x)
{
  If k == 0:
    y, l = decode_exp_Golomb_0_order(x)
  Else:
    q, l = decode_exp_Golomb_0_order(x)
    r = int(x[l : l + k])
    y = q × (2^k) + r
  Return y
}
Input: Non-negative integer x
Output: Bitstream y
function encode_exp_Golomb_0_order (x)
{
  q = to_binary(x + 1)
  q_len = length(q)
  p = "0" * (q_len − 1) // replicates "0" q_len − 1 times.
  y = concatenate(p, q)
  Return y
}
Input: Bitstream x
Output: Non-negative integer y, Non-negative integer l
function decode_exp_Golomb_0_order (x)
{
  p = count_consecutive_zeros_from_start(x) // consecutive zeros of x before the first "1".
  y = int(x[p : 2 × p + 1]) − 1
  l = 2 × p + 1
  Return y, l
}
//The notation x[a : b], follows the Python rules, i.e. selects characters in the range [a, b)
```

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1

[2] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1

[3] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998. 1

[4] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1

[5] Jukka Teuhola. A compression method for clustered bit-vectors. *Information processing letters*, 1978. 2