(Supplementary) A. Static Phase Selective Convolution

As a follow up on static PSC (sPSC) in Section 3.1, in this section, we present a more details regarding how to systematically split weights between positive and negative pre-activations.

Let $\mathbf{x}^+ \triangleq \text{ReLU}(\mathbf{x})$ and $\mathbf{x}^- \triangleq -\text{ReLU}(-\mathbf{x})$ denote the positive and the negative inputs, respectively. In conventional networks using ReLU as activation functions, only $\mathbf{x}^+$ is selected for further processing of subsequent layers. To improve the expressiveness of the network, it is desirable to also use the native activations $\mathbf{x}^-$. One naive way to incorporate $\mathbf{x}^-$ in the computation is to split the downstream operation into a positive branch and a negative branch, and concatenate their corresponding output:

$$\mathbf{y} = \begin{bmatrix} W^+ & 0 \\ 0 & W^- \end{bmatrix} \begin{bmatrix} \mathbf{x}^+ \\ \mathbf{x}^- \end{bmatrix} = \begin{bmatrix} W^+\mathbf{x}^+ \\ W^-\mathbf{x}^- \end{bmatrix},$$

where W^+ has dimension of $\rho M \times N$ and W^- has dimension of $(1-\rho)M \times N$ so that the overall dimension (number of output nodes in MLP or number of channels in Convnet) of the concatenate output is M. In other words, ρ determines the split of output dimension between the positive branch and the negative branch. The problem, then, is how to find the optimal split. The naive approach of exhaustively search all possible ρ values would be computationally prohibitive. Next, we propose a *gating approach* where ρ is optimized during training and fixed for inference.

For ease of explaining the gating approach, let us rewrite the above Equation as the following, with $\widetilde{W}^+$ and $\widetilde{W}^-$ denoting matrices with dimension $M \times N$.

$$\mathbf{y} = G^+\widetilde{W}^+\mathbf{x}^+ + G^-\widetilde{W}^-\mathbf{x}^-, \tag{19}$$

where $G^- = \text{diag}\{g_1, g_2, \ldots, g_M\}$ and $G^+ = \text{diag}\{1-g_1, \ldots, 1-g_M\}$ denotes the diagonal gating matrix. Each diagonal element g_i is a binary switch that controls whether the i^{th} output dimension is derived from the positive or the negative branch. The value of g_i is in turn defined as $g_i \triangleq \sum_{j \le i} s_j$, where $\mathbf{s} = \{s_1, \ldots, s_M, s_{M+1}\}$ [7] is a one-hot vector whose location of one delimits the boundary of positive and negative branch. We can then train this one-hot vector with standard relaxation methods [19, 28, 12]. Note that the value of $\mathbf{s}$ can be either data-dependent or data-invariant. In the latter case, after training is converged, we can simply remove connections in the two branches that are deactivated and achieve same complexity compared with the positive branch only case.

[7] The size of $\mathbf{s}$ has to be larger than the output dimension by 1 to properly characterize the delimiter of the two branches.

Another way to normalize the number of parameters when using the negative branch is to apply SVD compression [43] on $\widetilde{W}^+$ and $\widetilde{W}^-$. With input dimension of N and output dimension of M, we can decompose each of $\widetilde{W}^+$ and $\widetilde{W}^-$ into the product of two matrices with dimension $M \times r$ and $r \times N$, where $r \le \min\{M, N\}$ denotes the rank after the compression. The detailed procedure is shown below.

$$\begin{aligned} \underbrace{\widetilde{W}^+}_{M\times N} &= \underbrace{U}_{M\times M}\,\underbrace{\Sigma}_{M\times N}\,\underbrace{V^*}_{N\times N} \\ \overset{\text{take } r \text{ largest}}{\underset{\text{singular values}}{\approx}} &\underbrace{U'}_{M\times r}\,\underbrace{\Sigma'}_{r\times r}\,\underbrace{V'^*}_{r\times N} \\ &= \underbrace{U'\Sigma'^{1/2}}_{M\times r}\,\underbrace{\Sigma'^{1/2}V'^*}_{r\times N}. \end{aligned}$$

To equalize the total number of parameters to $N \times M$, we may set the rank to $r = \frac{NM}{2(N+M)}$. This could be deemed undesirable compared with the gating approach (Equation (19)), as the sum degrees of freedom reduces to $2r = NM/(N+M)$ and is strictly less than that of the gating approach, which is $\min\{N, M\}$. We intend to continue developments with static PSC in our future study.

(Supplementary) B. Proof of the L_2-Loss Lower Bound

As a follow up from the discussion in Section 3.1, we provide a proof for the lower bound of $\mathcal{L}(f, g)$ when f is the absolute value function on $[-1, 1]$ and g is the baseline computational model followed by the pooling unit. (See (2) for the definition of the baseline model.) Specifically, g is defined as

$$\begin{aligned} g(x) &\triangleq (w_1 + w_2) \times \text{ReLU}(x) + (b_1 + b_2) \\ &= w \times \text{ReLU}(x) + b, \end{aligned}$$

where $w \triangleq w_1 + w_2$ and $b \triangleq b_1 + b_2$. We have

$$\begin{aligned} \mathcal{L}(f, g) &= \int_0^1 (x - wx - b)^2 dx + \int_{-1}^0 (x + b)^2 dx \\ &= \int_0^1 \Big[\big((1-w)x - b\big)^2 + (x-b)^2\Big] dx. \end{aligned}$$

If we now set

$$\frac{\partial}{\partial w}\mathcal{L}(f, g) = 0, \frac{\partial}{\partial b}\mathcal{L}(f, g) = 0,$$

and differentiate under the integral, we get the system of equations

$$*\frac{2}{3}w + b = \frac{2}{3}, \frac{1}{2}w + 2b = 1,$$

which gives $w = b = \frac{2}{5}$. Therefore

$$\mathcal{L}(f, g) \ge \mathcal{L}(f, g)\Big|_{w=b=\frac{2}{5}} = \frac{2}{15}.$$