Konvolution derivations

da1sypetals.iota@gmail.com

1. Notations

- Let $a \in \mathbb{T}(...) \Leftrightarrow a \in \mathbb{R}^{\cdots}$.
- Let $[n] = \{0, 1, ..., n\}$.

2. Convolution

- Taking the easily understandable definition of convolution in CNNs in deep learning, convolution is defined as such:

1. Operands:
   - kernel $k_c \in \mathbb{T}(a_k, a_k, c_{\text{in}})$, where $c = 0, 1, \ldots\ c_{\text{out}} - 1$;
   - input $x \in \mathbb{T}(b, c_{\text{in}}, h, w)$.
2. Operation:

$$\begin{aligned} & y = x * k \\ & \Leftrightarrow y(b, c_o, h, w) = \sum_{c_i \in [c_{\text{in}}]} \sum_{(u,v) \in [a_k]^2} x(b, c_i, h, w) \cdot k_{c_o}(u, v, c_i) \end{aligned}$$

3. Konvolution

- We define the operation of ***konvolution*** as follows:

1. Operands:
   - kernel Φ_c is a matrix of function with shape $(a_k, a_k, d, c_{\text{in}})$, where $c = 0, 1, \ldots\ c_{\text{out}} - 1$;
   - input $x \in \mathbb{T}(b, c_{\text{in}}, h, w)$.
2. Operation:

$$\begin{aligned} & y = x \boxtimes \Phi \\ & \Leftrightarrow y(b, c_o, h, w) = \sum_{c_i \in [c_{\text{in}} - 1]} \sum_{(u,v) \in [a_k - 1]^2} \phi_{c_o}^{(u,v,c_i)}\{x(b, c_i, h, w)\} \end{aligned}$$

where $\phi_{c_o}^{(u,v,c_i)} = \Phi_c(a_k, a_k, d, c_{\text{in}})$. Resembling KAN, we use $\phi(x)$ instead of $w \cdot x$ to operate on an element.

4. Implementations

- Bare implementation include:

1. For each (scalr)element x, we first compute polynomial bases:

$$p(x) \in \mathbb{T}(d+1)$$

2. Then we compute their weighted sum

$$y(b, c_o, h, w) = \sum_{\delta=0}^{d} \sum_{c_i \in [c_{\text{in}}-1]} \sum_{(u,v) \in [a_k-1]^2} x(b, c_i, h, w) \cdot \kappa_{c_o}(\delta, c_i, u, v)$$

- However, the latter operation can be absorbed into convolution operation by:

$$\begin{aligned} y(b, c_o, h, w) &= \sum_{(\delta, c_i) \in [c_{\text{in}}] \times [d]} \sum_{(u,v) \in [a_k]^2} x(b, c_i, h, w) \cdot \kappa_{c_o}(\text{ch}(\delta, c_i), u, v) \\ &= x_{\text{expand}} * \kappa \end{aligned}$$

where $x_{\text{expand}} \in \mathbb{T}(b, c_{\text{in}} \cdot (d+1), h, w)$ and $\text{ch}(\delta, c_i) = \delta \cdot c_{\text{in}} + c_i$.

- By this means, we can make use of the high-performance built-in convolution module in `PyTorch` framework.