Appendix: An Empirical Study of Spatial Attention Mechanisms in Deep Networks

Xizhou Zhu\textsuperscript{1,2}\textsuperscript{*}, Dazhi Cheng\textsuperscript{2}\textsuperscript{*}, Zheng Zhang\textsuperscript{2}\textsuperscript{*}, Stephen Lin\textsuperscript{2}, Jifeng Dai\textsuperscript{2}

\textsuperscript{1}University of Science and Technology of China
\textsuperscript{2}Microsoft Research Asia

\{v-dachen, zhez, stevelin, jifdai\}@microsoft.com

Incorporating attention modules into deep networks

For the object detection and semantic segmentation tasks, ResNet-50 \cite{he2016deep} is chosen as the backbone and just the self-attention mechanism is involved. The Transformer attention module is incorporated by applying it on the $3 \times 3$ convolution output in the residual block. For insertion into a pre-trained model without breaking the initial behavior, the Transformer attention module includes a residual connection, and its output is multiplied by a learnable scalar initialized to zero, as in \cite{vaswani2017attention}. The manner of incorporating dynamic convolution is the same. To exploit deformable convolution, the $3 \times 3$ regular convolution in the residual block is replaced by its deformable counterpart. The resulting architecture is called “Attended Residual Block”, shown in Figure 2 (a).

In the neuron machine translation (NMT) task, the network architecture follows the Transformer base model \cite{vaswani2017attention}, where both self-attention and encoder-decoder attention mechanisms are involved. Different from the original paper, we update the absolute position embedding in the Transformer attention module by the latest relative position version as in Eq. 2. Because both deformable convolution and dynamic convolution capture self-attention, they are added to only the blocks capturing self-attention in Transformer. For dynamic convolution, we replace the Transformer attention module by dynamic convolution directly, as in \cite{zhu2019deformable}. The architecture is shown in Figure 2 (b). For its deformable convolution counterpart, because the Transformer model does not utilize any spatial convolution (with kernel size larger than 1), we insert the deformable convolution unit (with kernel size of 3) prior to the input of the Transformer attention module. The resulting architecture is called “Transformer + Deformable”, shown in Figure 2 (c).

Experimental settings

For the object detection task, experiments are implemented based on the open source mmdetection \cite{mmdetection} code base. Anchors of 5 scales and 3 aspect ratios are utilized. 2k and 1k region proposals are generated at a non-maximum suppression threshold of 0.7 at training and inference respectively. In SGD training, 256 anchor boxes (of positive-negative ratio 1:1) and 512 region proposals (of positive-negative ratio 1:3) are sampled for backpropagating their gradients. In our experiments, the networks are trained on 8 GPUs with 2 images per GPU for 12 epochs. The learning rate is initialized to 0.02 and is divided by 10 at the 8-th and the 11-th epochs. The weight decay and the momentum parameters are set to $10^{-4}$ and 0.9, respectively.

For the semantic segmentation task, in SGD training, the training images are augmented by randomly scaling (from 0.7 to 2.0), randomly cropping (size of $769 \times 769$ pixels) and random flipping horizontally. The hyper parameter setting follows \cite{zhu2019deformable}. In our experiments, the networks are trained on 8 GPUs with 1 image per GPU for 60k iterations. The “poly” learning rate policy is employed, where the initial learning rate is set as 0.005 and multiplied by $\left(1 - \frac{\text{iter}}{\text{iter}_{\text{max}}}\right)^{0.9}$. Synchronized Batch Normalization \cite{wu2018group} is placed after every newly added layer with learnable weights. The weight decay and the momentum parameters are set as $10^{-4}$ and 0.9, respectively.

For the neuron machine translation (NMT) task, we use the fairseq \cite{ott2019fairseq} code base for our experiments. The hyper parameters follows the original setting in \cite{vaswani2017attention}. We used the Adam optimizer \cite{kingma2014adam} with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. In our experiments, the networks are trained on 8 GPUs for 100k iterations. Each training batch contained a set of sentence pairs containing approximately 30k source tokens and 30k target tokens. The initial learning rate is set as $10^{-7}$ and linearly increased to 0.001 after $\text{iter}_{\text{warmup}} = 4000$ iterations, and then multiplied by

\*Equal contribution. \textsuperscript{†}This work was done when Xizhou Zhu and Dazhi Cheng were interns at Microsoft Research Asia.
\[
\text{iter}^{-0.5}
\]. No weight decay is adopted. During training, label smoothing [7] of value 0.1 is employed.

References


