Non-local Neural Networks

Xiaolong Wang, Ross Girshick, Abhinav Gupta, Kaiming He

This paper proposes non-local operations for capturing long rage dependencies with neural nets. This is done directly by computing interactions between any two positions regardless of their distance. This is different from a conv layer that only considers a local neighborhood, or a fully connected layer where the weights are learned across examples, and does not depend on the relationship between each input pair.

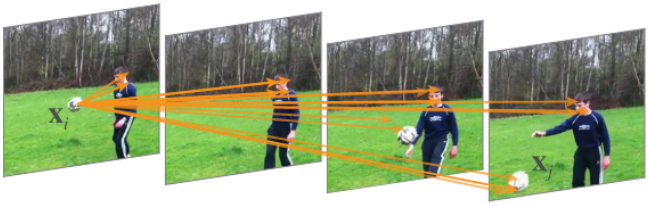

Figure 1. A spacetime ***non-local*** operation in our network trained for video classification in Kinetics. A position $\mathbf{x}_i$'s response is computed by the weighted average of the features of *all* positions $\mathbf{x}_j$ (only the highest weighted ones are shown here). In this example computed by our model, note how it relates the ball in the first frame to the ball in the last two frames. More examples are in Figure 3.

With the non-local operation, the output at each location (either time, space, or spacetime), depends on all of the inputs. Specifically, the output at a location i is the weighted sum over all possible responses between the input I itself and all of the rest of the inputs j, which is computed as the function of the similarity between i and j, and a new representation of the input at position j.

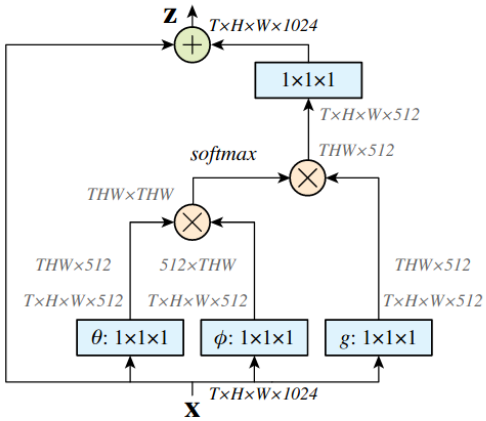

Figure 2. A spacetime **non-local block**. The feature maps are shown as the shape of their tensors, *e.g.*, $T\times H\times W\times 1024$ for 1024 channels (proper reshaping is performed when noted). "$\otimes$" denotes matrix multiplication, and "$\oplus$" denotes element-wise sum. The softmax operation is performed on each row. The blue boxes denote $1\times1\times1$ convolutions. Here we show the embedded Gaussian version, with a bottleneck of 512 channels. The vanilla Gaussian version can be done by removing θ and ϕ, and the dot-product version can be done by replacing softmax with scaling by $1/N$.

Types of the non-local module: In order to aggregate the information between a given input i with respect to the rest of the inputs (j for all j in the inputs). By first fixing the transformation of j as a linear transformation (1x1 or 1x1x1 convolution), we have many different possible choices on how to compute the pairwise relationships:

- Gaussian: exponentiated dot product similarity between the features of a pair (i, j). Normalization is the sum of the relationships.
- Embedded Gaussian: exponentiated dot product similarity between the embedded (two linear transformations) features of a pair (i, j), this can be seen as a case of the standard self-attention. Normalization is the sum of the relationships.
- Dot product: dot product similarity between the embedded features of a pair (i, j). Normalization is the number of pairs.
- Concatenation: ReLu of the projected concatenation of the concatenated features of a pair (i, j). Normalization is the number of pairs.

Model: after defining the non-local module, we can then create a non-local block that uses such a module. The block simply consists of a residual connection, where the output is the sum of the input, and the output of the non-local module after being transformed by a linear layer. This linear layer is initialized as zero so that the non-local block is a simple identity block at the start of the training. The block can then be added at different locations of the network, but the authors propose to add it at the end with downsampled spatial and temporal dimensions to avoid having an excessive cost. One of the main variants of the paper is **C2D**, which is a model composed only of 2D convolutions, 2 3D pooling layers at the start, and non-local blocks.