

Poly-NL: Linear Complexity Non-local Layers with Polynomials

Francesca Babiloni¹, Ioannis Marras¹, Filippos Kokkinos³, Jiankang Deng², Grigorios Chrysos⁴,
Stefanos Zafeiriou²

¹Huawei Noah’s Ark Lab ²Imperial College London ³University College London
⁴École Polytechnique Fédérale de Lausanne

Abstract

Spatial self-attention layers, in the form of Non-Local blocks, introduce long-range dependencies in Convolutional Neural Networks by computing pairwise similarities among all possible positions. Such pairwise functions underpin the effectiveness of non-local layers, but also determine a complexity that scales quadratically with respect to the input size both in space and time. This is a severely limiting factor that practically hinders the applicability of non-local blocks to even moderately sized inputs. Previous works focused on reducing the complexity by modifying the underlying matrix operations, however in this work we aim to retain full expressiveness of non-local layers while keeping complexity linear. We overcome the efficiency limitation of non-local blocks by framing them as special cases of 3rd order polynomial functions. This fact enables us to formulate novel fast Non-Local blocks, capable of reducing the complexity from quadratic to linear with no loss in performance, by replacing any direct computation of pairwise similarities with element-wise multiplications. The proposed method, which we dub as “Poly-NL”, is competitive with state-of-the-art performance across image recognition, instance segmentation, and face detection tasks, while having considerably less computational overhead.

1. Introduction

Convolutional Neural Networks (CNNs) have led to a revolution in machine learning, and, specifically, are currently the undisputed state of the art in computer vision on various tasks. Nonetheless, CNNs, even if composed by a deep stack of convolutional operators, have a limited receptive field [32], which makes the crucial long-range dependencies hard to capture.

Recent work on spatial self-attention ameliorated this issue with a novel set of modules for neural networks [50, 45]. These blocks extract non-local interactions among all spa-

tial positions of the input and weight them with a set of learnable parameters. Passing through a Non-local block, each input position takes into account the contribution of all the others, scaled by their similarity with a given reference. These blocks introduce the possibility to reason about the whole space in one glance and make non-local behavior easier to be captured by the network. Inserting Non-local blocks in neural architectures has been proven very effective [2, 14, 49, 37, 38, 36], but the computation of a similarity score for each pair of points scales quadratically with the number of spatial positions. As such, the expensive computational and storage complexity makes non-local blocks impractical to compute even upon moderately sized input.

Recent works tackle such limitation via an efficient computation of the similarity matrix [56, 31, 42] but miss to provide a theoretical overview of the Non-local block formulation. In this work, we build upon the aforementioned line of research, and revisit Non-local layers under the lens of polynomials, framing them as special cases of 3^{rd} order polynomials. Powered by this intuition, we derive an efficient version of Non-local neural networks, *Poly-NL* which takes into account long-range dependencies without the need to compute explicitly any pairwise similarity. *Poly-NL* layers perform computations using the same set of interactions as the Non-local block of [50], and at the same time reduce the overall complexity drastically from $O(N^2)$ to $O(N)$ with no loss in performance.

In this work, we link polynomials and the Non-Local layer. Our goal is to efficiently extract high-order interactions from the input while capturing long-range spatial dependencies. Thus, our contribution can be summarized as follows:

- We bridge the formulations between high-order polynomials and non-local attention. In particular, we prove that self-attention (in the form of Non-local blocks) can be seen as a particular case of general 3^{rd} order polynomials.

- We propose "Poly-NL" a novel building block for neural networks, which can be seen as polynomials of the input matrix. In particular, we propose an alternative Non-local block that reduce complexity from quadratic to linear with respect to the spatial dimensions.
- We showcase the efficiency and the effectiveness of our method across a range of tasks: image recognition, instance segmentation, and face detection.

2. Related Work

Multiplicative interactions [21] can be found at the core of various machine learning models such as Bilinear layers, LSTM, and Higher-order Boltzmann machines. In LSTM [19, 24], element-wise products are used to fuse representations. In Bilinear layers [44, 6, 30, 53] feature maps of different networks get bilinearly combined together to capture pairwise interactions. In k^{th} -order Boltzmann machines [34, 35, 40] k^{th} order multiplicative interactions are used to define the energy function. These *high order interactions* capture the many possible ways in which the output can depend on the input. More recently, Π -nets [12] use polynomial expansions as a function approximator, replacing traditional activation functions with polynomials of the input vectors and use tensor decompositions [23] to reduce the number of learnable parameters.

Multiplicative interactions are also crucial in the context of self-attention. Self-attention methods have been proposed as mechanisms to self-recalibrate feature maps and have been used either as replacement or addition to traditional residual blocks [18]. Complementary to our work, some of these methods accumulate contextual information into lightweight global-descriptors, either extrapolating a single scalar for each spatial position [46], channel [20, 4], channel and position [51] or region of space [26].

Closer to our work, it is the idea of modeling non-local long-range dependencies among spatial positions. While this is not new in computer vision [3, 25, 13] it is relatively recent in the context of neural networks architecture, in the form of "Non-local" attention modules, capable to attend at the same time every element of the input [50, 45]. Examples of successful use of these modules can be found in natural language processing as well as in computer vision, where some form of self-attention has been used to achieve state-of-the-art performance in various problems as translation [37], question answering [36], classification [38, 2], segmentation [47, 5], and video processing [49], among others. While some works focused on extending the scope of Non-local blocks, by capturing channels' correlations [1, 54, 15] or considering multiple resolutions of the image [33, 14], recent work has sparked a discussion on the scalability of these modules, and on how to overcome their intrinsic efficiency limitations [43].

Existing solutions focus on increasing the efficiency of the similarity operator, for example by reducing the number of positions attended [10, 55] or using low dimensional latent spaces [11, 56, 8, 48] or on the computation order of the matrix-formula [42, 22]. In this work, we propose an alternative solution to this problem. We introduce a faster reformulation of the Non-Local block by framing non-local dependencies as 3^{rd} order interactions. Our method can extract non-local dependencies using no matrix multiplication computed along the spatial dimension.

3. Non-locality and high-order interactions

We start by introducing notation and background, then proceed in formalizing the concept of 3^{rd} order interactions. Our goal is to accelerate Non-local blocks in a principled manner without losing the rich, long-range interactions that have proven successful in practice.

3.1. Background

Notation. We follow the notation of Kolda *et al.* as in [23]. Vectors are denoted as lower-case bold letters (e.g. \mathbf{x}) and matrices as upper-case bold letters (e.g. \mathbf{X}). The element (i, j) of a matrix \mathbf{X} can be indicated as $x_{(i,j)}$. Tensors are identified with bold Euler script letters (e.g. \mathcal{X}). The order of a tensor is the number of dimensions, also known as way or mode. Hadamard products are indicated using the symbol " \odot ". Given two tensors, we define their double-dot product as the tensor contraction with respect to the last two indices of the first one and the first two indices of the second one, identified with the bullet " \bullet " symbol. In the case of a tensor $\mathcal{W} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_{N-1} \times I_N}$ and a matrix $\mathbf{X} \in \mathbb{R}^{I_{N-1} \times I_N}$ their double-dot product is a tensor of order $N - 2$, i.e. $\mathcal{Y} = \mathcal{W} \bullet \mathbf{X}$ of dimension $I_1 \times I_2 \cdots \times I_{N-2}$. Specifically, in element-wise form, such double-dot product reads

$$y_{(i_1, \dots, i_{N-2})} = \sum_{i_n=1}^{I_N} \sum_{i_{n-1}=1}^{I_{N-1}} w_{(i_1, \dots, i_{N-2}, i_{n-1}, i_n)} x_{(i_{n-1}, i_n)}.$$

Non-local Block. A generic self-attention block for neural networks highlights relevant interactions in a feature map using a function g , designed to manipulate the input, and a function f , in charge of extracting similarities from it. In [50], the authors introduce the "Non-local block", a self-attention block used to highlight non-local long-range dependencies in the input. It operates on a folded feature map $\mathbf{X} \in \mathbb{R}^{N \times C}$ of N spatial positions and C channels and outputs a matrix \mathbf{Z} of the same dimensionality

$$\mathbf{Z} = \mathbf{Y} + \mathbf{X} = f(\mathbf{X})g(\mathbf{X}) + \mathbf{X} \quad (1)$$

where $f: \mathbb{R}^{N \times C} \rightarrow \mathbb{R}^{N \times N}$ is a pairwise function that calculates similarity for each pair of spatial positions, and

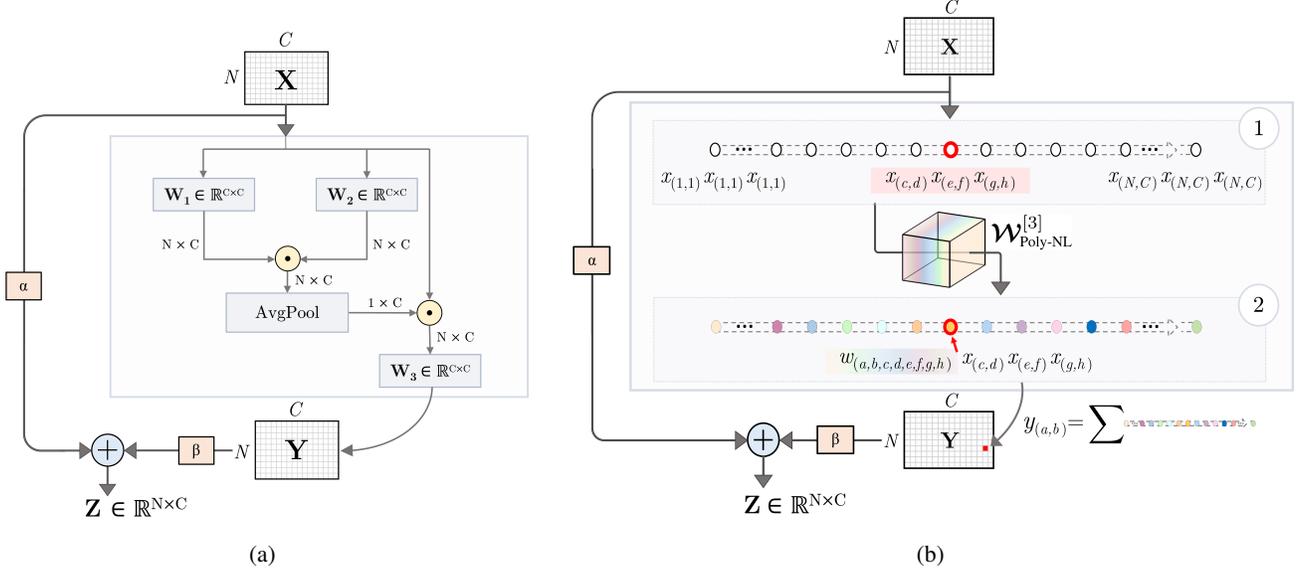


Figure 1: **Two views of the Poly-NL block.** **a)** Poly-NL as a non-local self-attention block for neural networks. The symbol \odot denotes Hadamard products. Gray boxes represent convolutions of kernel size 1 and an averaging function over the rows. The output of the average pooling undergoes an expansion before the Hadamard multiplication. **b)** Poly-NL as a 3^{rd} order polynomial module for neural networks. In the first box the space of 3^{rd} order interactions is represented as a line of $(NC)^3$ white dots, containing all possible triplets. The learnable parameters of $\mathcal{W}_{\text{Poly-NL}}^{[3]} \in \mathbb{R}^{N \times C \times N \times C \times N \times C \times N \times C}$ weight each triplet $x_{(c,d)}x_{(e,f)}x_{(g,h)}$ by its importance $w_{(a,b,c,d,e,f,g,h)}$. This is depicted in the second box as a line of colored dots. The output element $y_{(a,b)}$ is the weighted summation of every triplet.

$g: \mathbb{R}^{N \times C} \rightarrow \mathbb{R}^{N \times C}$, which has the form of a unary function computing a new representation for the input. In the case where $g(\mathbf{X})$ is a linear embedding and $f(\mathbf{X})$ is an embedded dot-product, the contribution of the self-attention to the output can be written as

$$\mathbf{Y}^{NL} = (\mathbf{X}\mathbf{W}_\theta\mathbf{W}_\phi^\top\mathbf{X}^\top)(\mathbf{X}\mathbf{W}_g) = \mathbf{X}\mathbf{W}_f\mathbf{X}^\top\mathbf{X}\mathbf{W}_g \quad (2)$$

where $\mathbf{W}_\theta, \mathbf{W}_\phi, \mathbf{W}_g$ are matrices of learnable parameters of dimension $C \times C$. To produce the output \mathbf{Y} , the Non-local block computes the dot-product between a similarity matrix $(\mathbf{X}\mathbf{W}_f\mathbf{X}^\top) \in \mathbb{R}^{N \times N}$ and an embedding of the input $(\mathbf{X}\mathbf{W}_g) \in \mathbb{R}^{N \times C}$. This matrix multiplication recalibrates the features of the n^{th} position via aggregating information from all the others. The pairwise function provides the similarity weights for the contribution of each position and uses a matrix multiplication along the N dimension. Such matrix multiplication on the N dimension is at the core of the non-local processing but introduces a quadratic term in computation that makes the complexity of this module equal to $O(N^2)$.

Polynomials for Neural Networks. Recently in [12], the authors adopted polynomials as layers of neural networks. We follow their formulation of a polynomial function $P: \mathbb{R}^{N \times C} \rightarrow \mathbb{R}^{N \times C}$ such that $\mathbf{Y} = P(\mathbf{X})$, in which each element of the output matrix is expressed as a polynomial of all the input elements $x_{(i,j)}$. The output of the layer

is formed as

$$\mathbf{Y} = P(\mathbf{X}) = \sum_{d=1}^D \mathcal{W}^{[d]} \prod_{j=1}^d \bullet \mathbf{X} + \mathbf{W}^{[0]} \quad (3)$$

where D is the order of the polynomial, $\mathcal{W}^{[d]}$ is the tensor of learnable parameters associated with a specific order d , and $\mathbf{W}^{[0]}$ is a bias matrix of learnable parameters. The order of the tensor $\mathcal{W}^{[d]}$ increases exponentially for higher-order polynomial terms, i.e. $\mathcal{W}^{[d]} \in \mathbb{R}^{N \times C \times \prod_{j=1}^d (N \times C)}$.

3.2. 3^{rd} order interactions

To present our method, we start by describing 3^{rd} order interactions terms for a feature map $\mathcal{X} \in \mathbb{R}^{H \times W \times C}$. We consider its folding $\mathbf{X} \in \mathbb{R}^{N \times C}$, where the spatial dimensions have been grouped together $N = H \times W$. To capture all potential 3^{rd} order dependencies between \mathbf{X} 's elements, we consider their linear combination weighted with a set of learnable parameters. In other words, we isolate the 3^{rd} order term ($D = 3$) of Eq. (3) by assuming $\mathcal{W}^{[0]} = 0, \mathcal{W}^{[1]} = 0, \mathcal{W}^{[2]} = 0$. Under the aforementioned assumptions, Eq. (3) becomes

$$\mathbf{Y} = (((\mathcal{W}^{[3]} \bullet \mathbf{X}) \bullet \mathbf{X}) \bullet \mathbf{X}) \quad (4)$$

where $\mathcal{W}^{[3]}$ is a tensor of order 8 and dimension $\mathbb{R}^{N \times C \times N \times C \times N \times C \times N \times C}$. We can find all possible 3^{rd}

order interactions, i.e. the multiplication of all possible triplets of the input’s elements summed together, clearly highlighted in its element-wise formula

$$y_{(a,b)} = \sum_{c,e,g} \sum_{d,f,h}^N w_{3_{(a,b,c,d,e,f,g,h)}} x_{(c,d)} x_{(e,f)} x_{(g,h)} \quad (5)$$

As depicted in Eq. (5), in a 3^{rd} order polynomial each element of the output matrix, $y_{(a,b)}$, benefits from the contributions of every possible triplet $x_{(c,d)}x_{(e,f)}x_{(g,h)}$, each weighted by its unique importance $w_{3_{(a,b,c,d,e,f,g,h)}}$. The use of $\mathcal{W}^{[3]}$ in its most general form would allow taking into account every possible pattern in the input but, at the same time, it would increase the number of parameters exponentially. A well-known problem in higher-order models [40, 12] is the number of parameters considered, which tends to be the most expensive part of their implementation. The number of the parameters to determine in Eq. (3) depends on the order of the polynomial and, even without considering orders lower than D , the parameters required are $(NC)^D$ (for instance, the use of $D = 3$ on an input 1024×196 will introduce nearly extra 10^{21} parameters).

Different approaches can be considered for reducing the number of parameters, for example by taking into account prior knowledge about the task or the nature of the input data [23, 35]. One approach to reduce the complexity is by selecting only a limited subsets of all the possible combinations $x_{(c,d)}x_{(e,f)}x_{(g,h)}$ exploiting a particular structure of the tensor $\mathcal{W}^{[3]}$. For example, assigning the same weight to a group of triplets will guarantee the same contribution for each of them, or imposing some zero-weights on one subsets of triplet will cancel their impact on the output $y_{(a,b)}$. These choices can be expressed in a formal way, which makes the format of the $\mathcal{W}^{[3]}$ tensor sparse, because some of the dimensions are constrained to be diagonal, or low-rank, since repeated values are used along some dimensions. The central idea of this paper is to factor the interaction tensor $\mathcal{W}^{[3]}$ in a particular way, to extract only a minimal subset of 3^{rd} order interactions from the input data. In other words, the choice of such tensor allows its replacement with matrices of smaller size, implemented using only pre-existing building blocks for neural networks.

4. Method

In this section, we characterize the set of 3^{rd} order interactions associated with non-local dependencies and propose a method that accesses them without the expensive computation of a similarity matrix. The proposed method “Poly-NL” is a novel non-local block, capable of selecting the same interactions as a Non-local layer at a fraction of its original computational cost in both space and time.

4.1. Poly-NL layer

As described in Section 3.1, a major drawback of the Non-local block is its complexity, which depends on the number of spatial positions as $O(N^2)$. To address this drawback we propose Poly-NL, a non-local spatial self-attention module that avoids any matrix multiplications along dimension N . Poly-NL takes in input a matrix $\mathbf{X} \in \mathbb{R}^{N \times C}$ and outputs a matrix of the same dimensionality \mathbf{Z} , that can be computed as $\mathbf{Z} = \alpha\mathbf{X} + \beta\mathbf{Y}^{\text{Poly-NL}}$, with α and β are learnable scalars. The matrix $\mathbf{Y}^{\text{Poly-NL}}$ is the core of the Poly-NL layer and can be written as follows

$$\mathbf{Y}^{\text{Poly-NL}} = (\Phi(\mathbf{X}\mathbf{W}_1 \odot \mathbf{X}\mathbf{W}_2) \odot \mathbf{X})\mathbf{W}_3, \quad (6)$$

where $\Phi: \mathbb{R}^{N \times C} \rightarrow \mathbb{R}^{N \times C}$ is an average pooling on spatial positions followed by an expand function, $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3 \in \mathbb{R}^{C \times C}$ are matrices of learnable parameters and \odot indicates element-wise multiplication. A visual depiction of the module is presented in Figure 1, Poly-NL is a layer that scales linearly with the dimension N (i.e. has a complexity of $O(N)$).

Noteworthy, Poly-NL extracts the same set of dependencies as the Non-local block but learns a different set of weights to process them. In order to connect the two formulations, we describe the set of interactions captured by these two blocks. The set of spatial interactions associated with Poly-NL is clearly highlighted in its element-wise formula

$$y_{(a,b)}^{\text{Poly-NL}} = \sum_{d,f,h}^C \sum_e^N \frac{1}{N} w_{1_{(h,d)}} w_{2_{(f,d)}} w_{3_{(d,b)}} x_{(a,d)} x_{(e,f)} x_{(e,h)} \quad (7)$$

In Poly-NL, each element $y_{(a,b)}^{\text{Poly-NL}}$ of the output matrix is computed using the contribution of a set of triplets $x_{(a,d)}x_{(e,f)}x_{(e,h)}$, weighted using the learnable parameters $w_{1_{(h,d)}} w_{2_{(f,d)}} w_{3_{(d,b)}}$.

Analogously, we highlight the set of interaction captured by the non-local module of Eq. (2), by writing its element-wise formulation

$$y_{(a,b)}^{\text{NL}} = \sum_{d,f,h}^C \sum_e^N w_{f_{(d,f)}} w_{g_{(h,b)}} x_{(a,d)} x_{(e,f)} x_{(e,h)}. \quad (8)$$

In the Non-local block, each element of the output matrix $y_{(a,b)}^{\text{NL}}$ is computed using the contribution of a set of triplets $x_{(a,d)}x_{(e,f)}x_{(e,h)}$, weighted using the learnable parameters $w_{f_{(d,f)}} w_{g_{(h,b)}}$. As visible from the comparison between the two formulas, Poly-NL and Non-Local block modules are closely connected. They both access the same set of triplets and optimize through backpropagation a set of learnable weights. Nevertheless, the two modules differ considerably in terms of computational efficiency. Poly-NL does not need to explicitly compute any pairwise-function and can be therefore viewed as a linear complexity alternative to the Non-Local blocks.

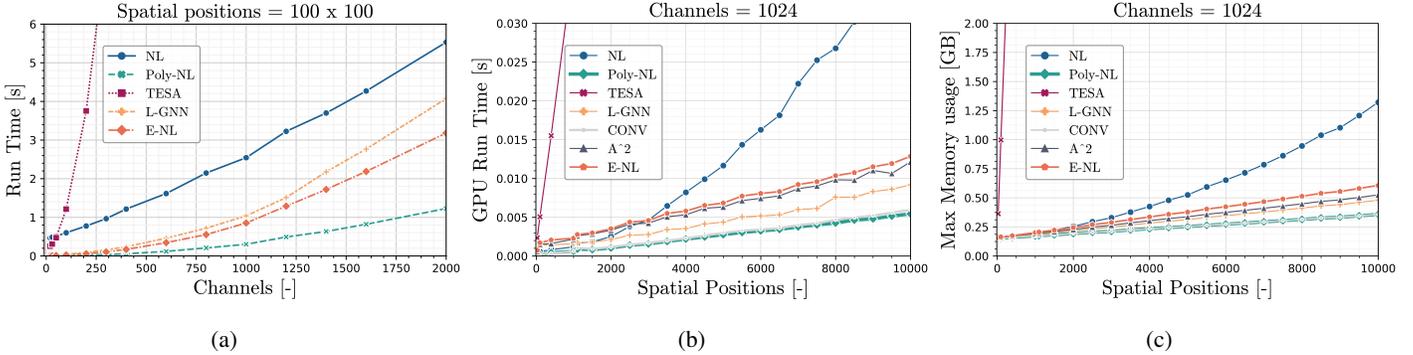


Figure 2: **Runtime and Peak memory consumption** performance comparison between Poly-NL and other non-local methods executed on a CPU Intel(R) Core(TM) i9-9900X CPU (a) and a RTX2080 GPU (b,c). Poly-NL exhibits lower computational overhead than competing methods, which is of importance with an increasing number of spatial positions.

| Method | AP_{box} | AP_{box50} | AP_{box75} | AP_{mask} | AP_{mask50} | AP_{mask75} |
|----------------|-------------|--------------|--------------|-------------|---------------|---------------|
| MaskR-CNN | 37.9 | 59.2 | 41.0 | 34.6 | 56.0 | 36.9 |
| + Non-local | 38.8 | 60.6 | 42.0 | <u>35.4</u> | <u>57.3</u> | <u>37.7</u> |
| + TESA | 39.5 | 60.9 | 43.1 | 35.4 | 57.2 | 37.5 |
| + Latent-GNN | 38.9 | 60.4 | 42.4 | 35.3 | 57.3 | 37.4 |
| + Efficient-NL | 38.9 | 60.3 | 42.2 | 35.4 | 57.2 | 37.7 |
| + Poly-NL | <u>39.2</u> | <u>60.8</u> | 42.2 | 35.4 | 57.4 | 37.6 |

Table 1: **Results** of Poly-NL and other Non-Local methods on Instance Segmentation-COCO.

Interestingly, both of these blocks are also special cases of 3^{rd} order polynomials of Eq. (4). The outputs of these blocks \mathbf{Y}^{NL} and $\mathbf{Y}^{Poly-NL}$, can be equivalently computed using Eq. (4), in which $\mathcal{W}_{NL}^{[3]}$ and $\mathcal{W}_{Poly-NL}^{[3]}$ are block-sparse, low-rank, and can be decomposed through smaller matrices (i.e. $\mathbf{W}_g \mathbf{W}_f$ and $\mathbf{W}_1 \mathbf{W}_2 \mathbf{W}_3$, respectively).

4.2. Relation with other Non-local blocks

The idea of decomposing higher-order tensors in smaller matrices is not new [9, 35], but can be used to cast a new light on a series of popular self-attention models. Besides Non-local block and Poly-NL, other popular non-local variants can be framed as special cases of 3^{rd} order polynomials [1, 56, 42]. We compare Poly-NL to these methods and discuss the advantages of our formulation in terms of computational efficiency.

Figure 2 depicts the complexity overhead of various spatial Non-local blocks for different sizes of the input matrix \mathbf{X} . In the visualization, we examine both the number of spatial positions (Figures 2b and 2c) and the number of channels (as in Figure 2a). The charts examine the performance of five different methods (TESA [1], NL [50], L-GNN [56], E-NL [42] and Poly-NL) and a showcase how the proposed solution is able to process inputs size otherwise unmanageable by other formulations. We report computational time on CPU (Figure 2a) and GPU (Figure 2b) as measure of time complexity and peak memory usage on GPU as indi-

cator of space complexity (Figure 2c). To ease comparisons among methods, we include as baseline a regular convolution layer (CONV), where no attention mechanism is used. All benchmarks were executed considering a single layer of each method on identical hardware, under comparable implementations, and hyper-parameters (please check additional material for full-nets run-times and implementation details). For each method, the values shown in the charts are the median of 20 runs.

The TESA block of [1] proposes to integrate spatial correlations together with channels’ dependencies by computing six matrix multiplications on the three different matrixizations of the input tensor \mathcal{X} . This procedure increments the patterns captured by the self-attention but it is burdened by a very high computational complexity of $O(N^2)$. The Latent-GNN block of [56], given the input matrix $\mathbf{X} \in \mathbb{R}^{N \times C}$, proposes to use a latent representation $N \times d$ to extract long-range dependencies with $O(Nd^2)$ complexity. The block uses matrix multiplications to compute a low-rank matrix $d \times d$, which captures latent space interactions, and a matrix $d \times C$, which captures its relation with the input channels. This method has a computational complexity that is linear with respect to the number of spatial positions N but depends on the choice of the hyperparameter d and a sequence of matrix dot-product multiplications to compute the output. Lastly, the “Efficient Non-local block” [42] proposes to compute the original formula of Eq. (2) from right

| Method | AP_{box} | AP_{box50} | AP_{box75} | AP_{mask} | AP_{mask50} | AP_{mask75} |
|-------------|-------------|--------------|--------------|-------------|---------------|---------------|
| MaskR-CNN | 37.9 | 59.2 | 41.0 | 34.6 | 56.0 | 36.9 |
| w/ Poly-NL | | | | | | |
| + on Res3 | 38.6 | 60.1 | 41.5 | 35.2 | 56.9 | 37.4 |
| + on Res4 | <u>39.2</u> | <u>60.8</u> | <u>42.2</u> | <u>35.4</u> | <u>57.4</u> | <u>37.6</u> |
| + on Res5 | 38.7 | 60.6 | 41.9 | 35.2 | 57.2 | 37.3 |
| + on Res345 | 39.8 | 61.7 | 43.2 | 36.0 | 58.4 | 38.3 |

Table 2: **Ablation study** of Poly-NL placement in MaskR-CNN for Instance Segmentation. Adding Poly-NL on different ResNet blocks yields changes in performance. An application of Poly-NL on all ResNet blocks provides the best results when compared to a sole application on a single block.

| Method | Top-1 | Top-5 |
|----------------|--------------|--------------|
| ResNet-50 | 75.62 | 92.68 |
| + Non-local | 76.09 | 93.00 |
| + TESA | 76.49 | <u>93.05</u> |
| + Latent-GNN | 75.28 | 92.33 |
| + Efficient-NL | 75.86 | 93.02 |
| + Poly-NL | <u>76.30</u> | 93.06 |

(a) **Imagenet**

| Method | <i>Easy</i> | <i>Medium</i> | <i>Hard</i> |
|----------------|--------------|---------------|--------------|
| ResNet-50 | 95.49 | 94.85 | 89.87 |
| + Non-local | 95.88 | 95.14 | 91.94 |
| + TESA | <u>96.22</u> | <u>95.61</u> | <u>92.58</u> |
| + Latent-GNN | 96.00 | 95.31 | 92.49 |
| + Efficient-NL | 96.06 | 95.42 | 92.55 |
| + Poly-NL | 96.37 | 95.71 | 92.76 |

(b) **Face Detection**

Table 3: **Results** of Non-Local variants for image classification on ImageNet and face detection on WIDER FACE.

to left. This procedure avoids the computation of pairwise-spatial similarities and makes the complexity linear with respect to N , but it still requires computing a sequence of two matrix dot-products multiplications to extract the output.

As displayed in Figures 2b and 2c, increasing the number of spatial positions greatly impacts efficiency. Run times of TESA and NL, which both depend quadratically on N , quickly become impractical, even in cases where the input dimension is small. Efficient methods (E-NL, L-GNN, Poly-NL) scale better with increasing spatial positions. Nonetheless, our method holds a competitive advantage across all figures, due to its lack of any matrix dot-product multiplication on the spatial dimension N . As shown in Figure 2a, the number of channels impact linearly the runtime performance of most methods, with the notable exception of TESA. Even in this case, our proposed method is performing significantly better than competing methods especially when the number of channels becomes significant.

As visible on the figures, Poly-NL consistently outperforms existing competitors, and has an efficiency on par with a regular convolution layer (CONV) since by design it avoids the explicit computation of any attention matrix.

5. Experiments

We evaluate the proposed method on three different tasks: object detection and instance segmentation on COCO [29], image classification on ImageNet [39], and

face detection on the WIDER FACE dataset [52]. We provide empirical evidence that Poly-NL outperforms previously proposed Non-local neural networks while maintaining an optimal trade-off between efficiency and performance.

5.1. Instance Segmentation on MS COCO

We tested our method on object detection and instance segmentation, where the network processes an image and produces a pixel-pixel mask that identifies both the category and the instance for each object. We use the MS-COCO 2017 dataset [29], composed by 118k images as training set, 5k as validation set and 20k as test set, and the Mask R-CNN baseline of [17]. For all the experiments, we report the standard metrics of Average Precision AP , AP_{50} , and AP_{75} for both bounding boxes and segmentation masks. The Mask R-CNN architecture is composed of a ResNet-FPN backbone for feature extraction followed by a stage that predicts class and box offsets. We trained with 8 Tesla V-100 GPUs and 2 images per GPU (effective batch size 16) using random horizontal flip as augmentation during training. We use an SGD solver with weight decay of 0.0001, momentum of 0.90, and an initial learning rate of 0.02. All models are trained for 26 epochs with learning rate steps are executed at epoch 16 and 22 with gamma 0.1. We used as backbones ResNet-50 [18] architectures pre-trained on Imagenet.

Following prior work, we modify the Mask R-CNN backbone by adding one non-local layer right before the last

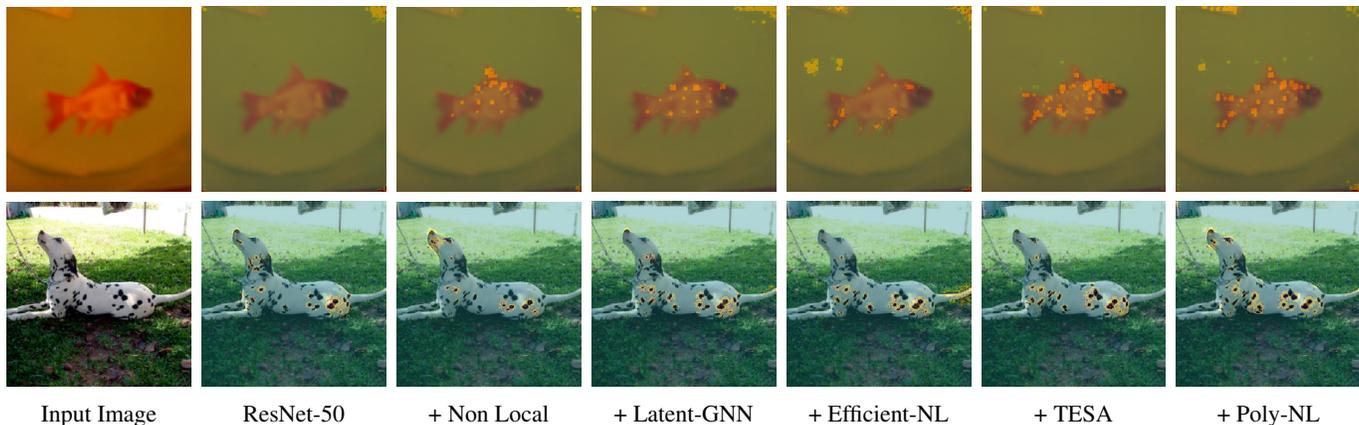


Figure 3: **Class saliency maps** of different methods. Grad-Cam [41] evaluates regions of the image which correspond to the class of interest. The use of non-local blocks helps to discriminate classes’ characteristics.

residual block of Res4. This procedure highlights the capacity of the self-attention to boost features’ representation and improve the quality of the candidate object bounding boxes. We compare our method against four different spatial self-attention layers, the original Non-local block of [50], the efficient Latent-GNN variant of [56], the Efficient-NL of [42] and the recently proposed TESA [1]. For fair comparison, we report the results from our training, achieved using public available source codes and hyperparameters as provided by the respective authors.

Quantitative results are summarized in Table 1. When compared to the best performing method, TESA [1], Poly-NL exhibits identical performance in AP_{mask} and slightly lower accuracy for AP_{box} . However, we note that our proposed method is nearly $\times 10$ faster to compute than TESA at the given resolution. Moreover, compared to the Non-local layer [50] and its efficient variants Latent-GNN [56] and Efficient-Net, our method improves performance by 0.3% \uparrow in AP_{box} while keeping linear computational complexity.

We ablate the location we insert the proposed layer in MaskR-CNN and present our findings in Table 2. We find that employing self-attention on any block of the ResNet backbone improves considerably the performance in both detection and segmentation. It appears that Res4 is the optimal block to insert Poly-NL into, since the numerical improvement across all metrics is consistent. At the same time, Table 2 shows that a combination of all ResNet blocks leads to the best performance (at most 1.2% \uparrow in AP_{box} and 1.5% \uparrow in AP_{mask}). Although having a self-attention block at Res4 is preferable, the contribution of attention on multiple blocks outperforms the usage on a single module. These results suggest how complementary attention patterns can be captured at different network stages.

5.2. Face Detection on WIDER FACE

We also apply our model to the task of face detection on the WIDER FACE dataset [52], which consists of 32,203 images and 393,703 face bounding boxes (40% training, 10% validation, and 50% testing) with a high degree of variability in scale, pose, expression, occlusion, and illumination. Compared to COCO [29], WIDER FACE [52] contains more tiny and dense detection objects (*i.e.* faces). 51% of objects from COCO [29] have the relative scale to the image below 0.11, while for a similar proportion, 55% of faces in WIDER FACE are less than 0.02. In addition, 1% of images in COCO have more than 30 objects, while there are 8% images contains more than 30 faces in WIDER FACE and many images even include more than 150 faces. Based on the detection rate of EdgeBox [58], three levels of difficulty (*i.e.* Easy, Medium, and Hard) are defined by incrementally incorporating hard samples. By using the evaluation metric of IoU 0.5, we compare the Average Precision (AP) of the proposed method and other baselines on Easy, Medium and Hard subsets, respectively.

Our experiments are implemented with PyTorch based on open source mmdetection [7]. Inspired by RetinaNet [28], we choose ResNet-50 [18] as backbone and Feature Pyramid Network (FPN) [27] as neck to construct the feature extractor. The losses of classification and regression branches are focal loss [28] and DIoU loss [57], respectively. Following [50], we insert one Poly-NL block right before the last residual block of $c4$. To detect tiny faces, we tile three scales of anchors over each level of the FPN. The aspect ratio is set as 1.3 and the IoU threshold for positive sampling matching is 0.35. For augmentation during training, square patches are cropped and resized to 640×640 from the original image with a random scale. Then, photometric distortion and random horizontal flip with the probability of 0.5 are applied. We train the model by using SGD

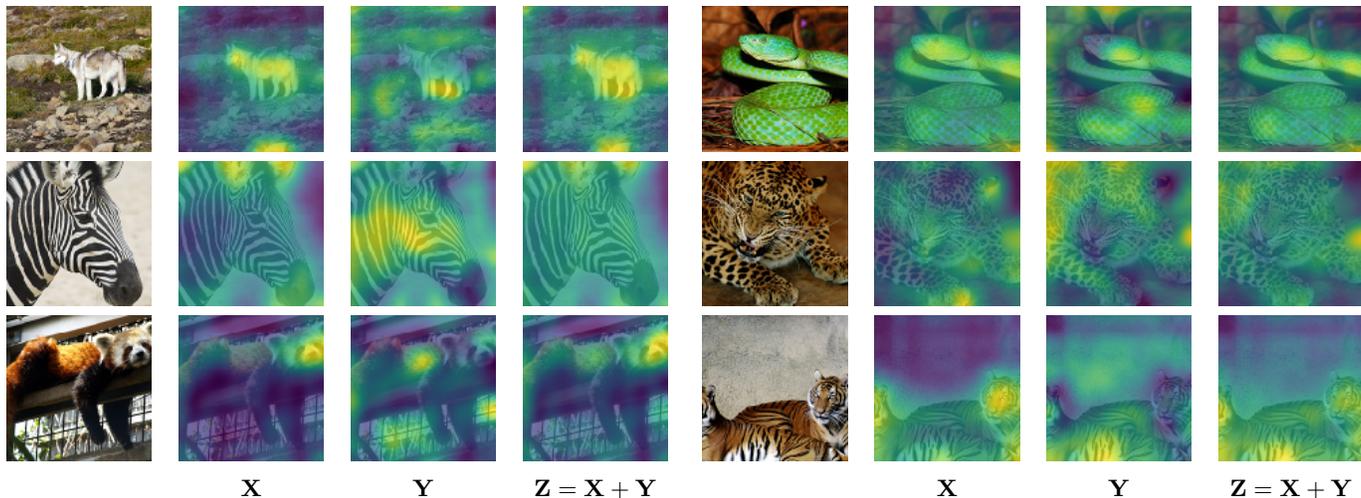


Figure 4: **Non-local dependencies** captured by Poly-NL. The norm of the extracted features per spatial location is visualized over the input image. The attention contribution Y learns patterns complementary to those captured by the input X . The summation of the aforementioned quantities merges together the contribution of short and long-range spatial dependencies.

optimizer (momentum 0.9, weight decay $5e-4$) with a batch size of 8×8 on 8 Tesla V100 GPUs. The initial learning rate is set to 0.001, linearly warms up for the first 3 epochs, and decays by a factor of 10 at 250-th epoch and 350-th epoch. All the models are trained with 400 epochs from scratch without any pre-training. During testing, we only employ single scale inference with the short and long edge of image bounded by 1100 and 1650, respectively. As shown in Table 3b, all attention modules can significantly improve the performance of face detection on WIDER FACE, indicating the effectiveness of context modeling (*i.e.* capturing long-range dependencies among pixels). Besides, the proposed Poly-NL module consistently outperforms all other non-local layers across the three levels of difficulty, achieving the mAP of 92.76% on the hard subset while being considerably faster than all competing methods.

5.3. Classification on Imagenet

We evaluated our method on the task of large-scale image classification, using Imagenet dataset [39], counting 1.28M training images of 1000 classes. For all the experiments, we modify a ResNet-50 architecture [18] by inserting a self-attention module and then train from scratch with 8 GPUs for 90 epochs, using a batch size of 256 and an SGD optimizer with an initial learning rate of 0.1 and weight-decay as described in [16]. Quantitative results are reported in table (3a) and show the Top-1 and the Top-5 accuracy for the evaluated methods. It is apparent that also in the classification task, where the goal is to provide a summary of the input, reasoning about spatial dependencies benefits greatly the accuracy. Poly-NL achieves the best performance on Top-5 accuracy, and on Top-1, outperforms significantly

all other Non-Local neural networks with the exception of TESA [1], which is very computationally demanding.

Beyond quantitative results, Fig. 3 illustrates qualitative differences between different non-local variants. The visualization is produced with Gradient-weighted Class Activation Mapping (Grad-CAM) [41], a technique that highlights areas of high importance for image classification tasks. Our proposed method more accurately captures global context and salient features when compared to other Non-Local methods.

Fig. 4 visualize the contribution of Poly-NL to the input’s representation. The figure overlays the norm of different features on top of the input image. We report the input’s feature map X , the contribution of attention module Y and their summation Z , *i.e.* output of the Poly-NL module. It is apparent that Poly-NL learns to contextualize the visual clues of the input with non-local dependencies. Our self-attention module can effectively recognize patterns that are complementary to those captured in the input and makes the features map aware of long-range dependencies.

6. Conclusions

In this work, we cast the recently proposed Non-local block as a 3rd order polynomial in the form of multiplicative interactions between spatial locations on a grid. Based on this fact, we propose a novel and fast embodiment of Non-local layers named Poly-NL that can capture long-range dependencies with a complexity that scales linearly with the size of the input in both space and time. Poly-NL consistently outperforms other non-local networks on image recognition, instance segmentation, and face detection.

References

- [1] Francesca Babiloni, Ioannis Marras, Gregory Slabaugh, and Stefanos Zafeiriou. Tesa: Tensor element self-attention via matricization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13945–13954, 2020. 2, 5, 7, 8
- [2] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V. Le. Attention augmented convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 1, 2
- [3] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65. IEEE, 2005. 2
- [4] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 2
- [5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020. 2
- [6] Joao Carreira, Rui Caseiro, Jorge Batista, and Cristian Sminchisescu. Semantic segmentation with second-order pooling. In *European Conference on Computer Vision*, pages 430–443. Springer, 2012. 2
- [7] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv:1906.07155*, 2019. 7
- [8] Yunpeng Chen, Marcus Rohrbach, Zhicheng Yan, Yan Shuicheng, Jiashi Feng, and Yannis Kalantidis. Graph-based global reasoning networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 433–442, 2019. 2
- [9] Eric C Chi and Tamara G Kolda. On tensors, sparsity, and nonnegative factorizations. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1272–1299, 2012. 5
- [10] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019. 2
- [11] Krzysztof Marcin Choromanski, Valerii Likhoshesterov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarrlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021. 2
- [12] Grigorios G Chrysos, Stylianos Moschoglou, Giorgos Bouritsas, Yannis Panagakis, Jiankang Deng, and Stefanos Zafeiriou. P-nets: Deep polynomial neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7325–7335, 2020. 2, 3, 4
- [13] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095, 2007. 2
- [14] Tao Dai, Jianrui Cai, Yongbing Zhang, Shu-Tao Xia, and Lei Zhang. Second-order attention network for single image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11065–11074, 2019. 1, 2
- [15] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019. 2
- [16] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large mini-batch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 8
- [17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 6
- [18] 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*, pages 770–778, 2016. 2, 6, 7, 8
- [19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2
- [20] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 2
- [21] Siddhant M Jayakumar, Wojciech M Czarnecki, Jacob Menick, Jonathan Schwarz, Jack Rae, Simon Osindero, Yee Whye Teh, Tim Harley, and Razvan Pascanu. Multiplicative interactions and where to find them. In *International Conference on Learning Representations*, 2019. 2
- [22] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020. 2
- [23] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009. 2, 4
- [24] Ben Krause, Liang Lu, Iain Murray, and Steve Renals. Multiplicative lstm for sequence modelling. *arXiv preprint arXiv:1609.07959*, 2016. 2
- [25] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289. Morgan Kaufmann Publishers Inc., 2001. 2
- [26] Xiang Li, Xiaolin Hu, and Jian Yang. Spatial group-wise enhance: Improving semantic feature learning in convolutional networks. *arXiv preprint arXiv:1905.09646*, 2019. 2
- [27] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 7

- [28] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 7
- [29] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 6, 7
- [30] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1457, 2015. 2
- [31] Chih-Ting Liu, Chih-Wei Wu, Yu-Chiang Frank Wang, and Shao-Yi Chien. Spatially and temporally efficient non-local attention network for video-based person re-identification. In *British Machine Vision Conference*, 2019. 1
- [32] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. 1
- [33] Yiqun Mei, Yuchen Fan, Yulun Zhang, Jiahui Yu, Yuqian Zhou, Ding Liu, Yun Fu, Thomas S Huang, and Honghui Shi. Pyramid attention networks for image restoration. *arXiv preprint arXiv:2004.13824*, 2020. 2
- [34] Roland Memisevic and Geoffrey Hinton. Unsupervised learning of image transformations. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. 2
- [35] Roland Memisevic and Geoffrey E Hinton. Learning to represent spatial transformations with factored higher-order boltzmann machines. *Neural computation*, 22(6):1473–1492, 2010. 2, 4, 5
- [36] Abdelrahman Mohamed, Dmytro Okhonko, and Luke Zettlemoyer. Transformers with convolutional context for asr. *arXiv preprint arXiv:1904.11660*, 2019. 1, 2
- [37] Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation (WMT)*, 2018. 1, 2
- [38] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 1, 2
- [39] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 6, 8
- [40] Terrence J Sejnowski. Higher-order boltzmann machines. In *AIP Conference Proceedings*, volume 151, pages 398–403. American Institute of Physics, 1986. 2, 4
- [41] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017. 7, 8
- [42] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3531–3539, 2021. 1, 2, 5, 7
- [43] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*, 2020. 2
- [44] Joshua B Tenenbaum and William T Freeman. Separating style and content with bilinear models. *Neural computation*, 12(6):1247–1283, 2000. 2
- [45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. 1, 2
- [46] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2017. 2
- [47] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *European Conference on Computer Vision*, pages 108–126. Springer, 2020. 2
- [48] Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020. 2
- [49] Xintao Wang, Kelvin CK Chan, Ke Yu, Chao Dong, and Chen Change Loy. Edvr: Video restoration with enhanced deformable convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 1, 2
- [50] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018. 1, 2, 5, 7
- [51] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 2
- [52] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. Wider face: A face detection benchmark. In *CVPR*, 2016. 6, 7
- [53] Chaojian Yu, Xinyi Zhao, Qi Zheng, Peng Zhang, and Xinge You. Hierarchical bilinear pooling for fine-grained visual recognition. In *Proceedings of the European conference on computer vision (ECCV)*, pages 574–589, 2018. 2
- [54] Kaiyu Yue, Ming Sun, Yuchen Yuan, Feng Zhou, Errui Ding, and Fuxin Xu. Compact generalized non-local network. In *NeurIPS*, pages 6511–6520, 2018. 2
- [55] Li Zhang, Dan Xu, Anurag Arnab, and Philip HS Torr. Dynamic graph message passing networks. In *Proceedings of*

the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 3726–3735, 2020. [2](#)

- [56] Songyang Zhang, Xuming He, and Shipeng Yan. Latent-gnn: Learning efficient non-local relations for visual recognition. In *International Conference on Machine Learning*, pages 7374–7383, 2019. [1](#), [2](#), [5](#), [7](#)
- [57] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-iou loss: Faster and better learning for bounding box regression. In *AAAI*, 2020. [7](#)
- [58] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. [7](#)