Linear Complexity Self-Attention with 3^{rd} Order Polynomials

Francesca Babiloni, Ioannis Marras, Jiankang Deng, Filippos Kokkinos, Matteo Maggioni, Grigorios Chrysos, Philip Torr and Stefanos Zafeiriou

Abstract—Self-attention mechanisms and non-local blocks have become crucial building blocks for state-of-the-art neural architectures thanks to their unparalleled ability in capturing long-range dependencies in the input. However their cost is quadratic with the number of spatial positions hence making their use impractical in many real case applications. In this work, we analyze these methods through a polynomial lens, and we show that self-attention can be seen as a special case of a 3rd order polynomial. Within this polynomial framework, we are able to design polynomial operators capable of accessing the same data pattern of non-local and self-attention blocks while reducing the complexity from quadratic to linear. As a result, we propose two modules (Poly-NL and Poly-SA) that can be used as "drop-in" replacements for more-complex non-local and self-attention layers in state-of-the-art CNNs and ViT architectures. Our modules can achieve comparable, if not better, performance across a wide range of computer vision tasks while keeping a complexity equivalent to a standard linear layer.

Index Terms—self-attention, non-local blocks, transformers, polynomial expansion, neural networks

1 Introduction

Convolutional Neural Networks (CNNs) are often at the core of state-of-the-art methods in computer vision. However, CNNs suffers from limited receptive field, even in deep architectures, because interactions between input features decay exponentially with their distance [1].

Self-attention (SA) [2] and the non-local block (NL) [3] have been proposed as techniques to extract long-range dependencies from the input in a position-independent manner. Specifically, SA adaptively modulates any given (reference) position using contributions of all the other positions scaled by their pairwise similarity with the reference. This has been shown to be an effective strategy to enrich, or even completely replace, standard convolutional layers in a wide range of architectures and domains [4], [5], [6], [7], [8], [9], [10]. However, the effectiveness of these layers relies on computing similarities between every pair of positions. This operation has a quadratic cost (with respect to the input-dimension size) and thus is prohibitive to compute when input resolution is large and/or computational resources are limited (e.g., on edge devices).

We establish a new link between the self-attention/non-local block and polynomial expansions. This new perspective enables us to design novel operators that are able to capture *the same correlations* as the non-local block of [3] while reducing the computational cost of these blocks from quadratic to linear without shrinking the receptive field as done in previous works [11], [12], [13].

- *F. Babiloni, I.Marras, J.Deng, F.Kokkinos, M.Maggioni and S. Zafeiriou are with Huawei Technologies Ltd, Noah's Ark Lab. E-mail: francesca.babiloni@huawei.com*
- *F. Kokkinos is with University College London (UCL)*
- *G.Chrysos is with the Department of Electrical Engineering, Ecole Polytechnique Federale de Lausanne (EPFL)*
- *P. Torr is with the Department of Engineering Science, University of Oxford*
- *S. Zafeiriou and F.Babiloni are with Imperial College London*

In our preliminary work [14], we frame the NL block as a special case of 3^{rd} order polynomials, and introduced *Poly-NL*. In this work, we perform three significant extensions: a) we propose *Poly-SA*, a multi-head self-attention layer that extends our *Poly-NL* to fit Vision Transformer (ViT) architectures, which are widely used across a range of tasks [15], [10], [16]; b) We provide new insights into the link between SA/NL and the polynomial expansions (i.e. Sec. 2), as well as polynomial networks in general; c) We conduct a thorough evaluation with the newly introduced proposed Poly-SA, which highlights its ability to work as low-complexity alternative to standard attention blocks in several ViT backbones. In addition to these extensions, we d) provide additional visualizations of the long-term dependencies captured by our blocks which are able to intuitively illustrate the inner workings of our method, and e) we extend the text to discuss limitations for the interested practitioner. In summary, our contributions can be summarized as follows:

- We bridge the formulations of high-order polynomials and 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* and *Poly-SA*, two novel building blocks for neural networks which can replace standard NL/SA reducing the complexity from quadratic to linear.
- We showcase the efficiency and the effectiveness of our blocks in both CNN and ViT architectures across a wide range of tasks: image recognition, instance segmentation, jigsaw puzzle reconstruction, and face detection.

2 Related Work

2.1 Multiplicative Interactions

Multiplicative interactions [17], [18] are essential to various machine learning models such as LSTM, Bilinear layers, and Higher-order Boltzmann machines. In LSTM [19], [20], element-wise products are used to fuse representations. In Bilinear layers [21], [22], [23], [24] feature maps of different networks get bilinearly combined together to capture pairwise interactions. In k^{th}-order Boltzmann machines [25], [26], [27], k^{th} order multiplicative interactions are used to define the energy function. These *high order interactions* capture diverse interactions between the input elements. More recently, $\Pi-$nets [28] use polynomial expansions as a function approximator using tensor decompositions [29] to reduce the number of learnable parameters. A number of works have demonstrated the separation of polynomial networks from regular neural networks and their benefits with respect to expressivity [30], [17], interpretability [31], learning high frequency functions [32], and extrapolation [33].

2.2 Attention

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 [34]. Complementary to our work, some of these methods accumulate contextual information into lightweight global-descriptors extrapolating a single scalar for each spatial position [35], channel [36], [37], channel and position [38], or region of space [39]. Capturing long-range spatial dependencies among spatial positions is a long-standing problem in computer vision [40], [41], [42]. However, it had received little attention until recently in the context of neural network architectures [3], [2]. Spatial self-attention modules for neural networks leverage long-range dependencies of the input and have been used in natural language processing as well as in computer vision to achieve state-of-the-art performance in various problems such as translation [7], question answering [9], classification [8], [4], segmentation [43], [44], [45], and video processing [6], among others. Some works focused on extending the scope of attention by capturing channel correlations [46], [47], [48] or considering multiple resolutions of the image [49], [5]. Other influential works conducted in the context of architecture design proved how spatial-self-attention networks (Transformers) represent a suitable alternative to CNN in vision [50], [51], [52], [53], [54].

2.3 Reducing Complexity

Despite the undiscussed success of self-attention, recent works sparked a discussion on its scalability, and on how to overcome its intrinsic efficiency limitations [55]. Existing solutions focus on increasing the efficiency of the similarity operator, for example by reducing the number of positions attended [56] or using low dimensional latent spaces [57], [58]. Linear-Attention (LA) [59] replaces the softmax-attention in transformer architectures with a feature-map dot product while XCiT [60] replaces the computation of the standard attention matrix with a cross-covariance alternative. These methods linearize complexity by computing pairwise relations between features instead of spatial positions. A similar idea in the context of non-local block can be found in Double Attention network [61] and, more recently, in Efficient Attention [62]. In a similar spirit, LatentGNN [63] introduces an additional interaction operation in the latent space, representing non-local relations via a mixture of low-rank kernel matrices. Different strategies investigate alternative kernels for transformer architectures [64] and propose alternatives of linear complexity by using random features [65], [66] or kernel learning [67].Lastly, Attention Free Transformers [68] uses element-wise multiplication to calculate attention in transformer architectures. In contrast to previous works, we propose a linear alternative of the non-local block by framing non-local dependencies as 3^{rd} order interactions.

3 Capturing High-order Interactions in Neural Networks

We start by introducing notation and background, then proceed in formalizing the concept of 3^{rd} order interactions. Our goal is to frame spatial-attention blocks and long-range interactions as a polynomial expansion.

3.1 Polynomials for Neural Networks.

We follow the notation of Kolda *et al*. [29]. 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 at position (i, j) of a matrix $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$ can be indicated as $x_{(i,j)}$. Tensors are identified with bold Euler script letters (e.g. $\boldsymbol{\mathcal{X}}$). 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. For instance, the double-dot product between a tensor $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_{N-1} \times I_N}$ and a matrix $\mathbf{X} \in \mathbb{R}^{I_{N-1} \times I_N}$ is a tensor of order N - 2, i.e. $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{W}} \bullet \mathbf{X} \in \mathbb{R}^{I_1 \cdots \times I_{N-2}}$. Specifically, the element-wise form of the double-dot product is expressed as:

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

In [28], the authors adopted polynomials as layers of neural networks. Intuitively each element of the output depends on all the elements of the input through a polynomial function. Formally, the output of the layer is defined as

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

where $\mathbf{X}$ and $\mathbf{Y}$ are the input and output matrices both having size $I_1 \times I_2$, P is a polynomial function of order D, $\boldsymbol{\mathcal{W}}^{[d]} \in \mathbb{R}^{I_1 \times I_2 \times \prod_{j=1}^{d} (I_1 \times I_2)}$ is a tensor of learnable parameters associated with a specific order $d \in \{1, \ldots, D\}$, and $\mathbf{W}^{[0]}$ is a bias matrix of learnable parameters. Note that the

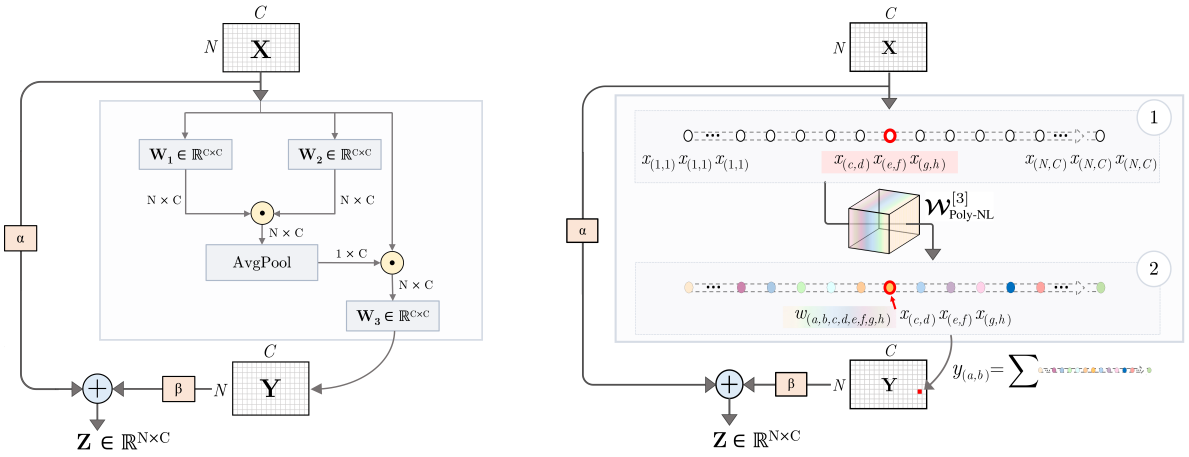


Fig. 1: **Two views of the Poly-NL block. a)** Poly-NL as a non-local self-attention block for neural networks. Gray boxes represent convolutions of kernel size 1 and an averaging function over the rows. The output of the average pooling undergoes an expansion before the Hadamard multiplication. **b)** Poly-NL as a 3rd order polynomial module for neural networks. In the first box the space of 3rd order interactions is represented as a line of $(NC)^3$ white dots, containing all possible triplets. The learnable parameters of $\boldsymbol{\mathcal{W}}^{[3]}_{\text{Poly-NL}} \in \mathbb{R}^{N\times C\times N\times C\times N\times C\times N\times C}$ 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. Poly-NL focuses only on a small subset of all order interactions (e.g. 0.025%), which is equivalent to assuming a portion of $\boldsymbol{\mathcal{W}}^{[3]}_{\text{Poly-NL}}$ values equal to zero.

size of the parameter tensor $\boldsymbol{\mathcal{W}}^{[d]}$ increases exponentially with the order d of the polynomial.

3.2 Higher Order Interactions

To provide some relevant background, we start by describing higher-order interactions terms for a feature map $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{H\times W\times C}$, where H, W, and C correspond to the height, the width and the number of channels for the given input tensor. We consider its folding $\mathbf{X} \in \mathbb{R}^{N\times C}$, with vectorized spatial dimensions of size $N = HW$. A 2nd order polynomial building block for neural networks captures pairwise dependencies among the elements of $\mathbf{X}$, by considering their linear combination weighted with a set of learnable parameters. The general equation for a 2nd order block can be derived by isolating the 2nd order term ($D = 2$) of Eq. (1)

$$\mathbf{Y} = \left(\left(\boldsymbol{\mathcal{W}}^{[2]} \bullet \mathbf{X}\right) \bullet \mathbf{X}\right), \tag{2}$$

where $\boldsymbol{\mathcal{W}}^{[2]}$ is a tensor of order 6 and dimension $\mathbb{R}^{N\times C\times N\times C\times N\times C}$. Note that Eq. (1) acts as a generalization of a single linear layer, as visible from its element-wise equation:

$$y_{(a,b)} = \sum_{c,e}^{N}\sum_{d,f}^{C} w_{2_{(a,b,c,d,e,f)}} x_{(c,d)}x_{(e,f)}. \tag{3}$$

Analogously, to capture all 3rd order dependencies between elements of $\mathbf{X}$, we isolate the 3rd order term ($D = 3$) of Eq. (1) by assuming $\boldsymbol{\mathcal{W}}^{[d]} = \mathbf{0}$ for $d \in \{0, 1, 2\}$. As a result, we obtain a simplified formulation for Eq. (1) as

$$\mathbf{Y} = \left(\left(\left(\boldsymbol{\mathcal{W}}^{[3]} \bullet \mathbf{X}\right) \bullet \mathbf{X}\right) \bullet \mathbf{X}\right), \tag{4}$$

where $\boldsymbol{\mathcal{W}}^{[3]}$ is a tensor of order 8 and dimension $\mathbb{R}^{N\times C\times N\times C\times N\times C\times N\times C}$. Similarly to the 2nd order case, its element-wise form is defined as

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

which clearly highlights how Eq. (5) includes multiplication of all possible triplets of the input elements summed together, *i.e.*. , all possible 3rd order interactions. As can be seen from Eq. (5), in a 3rd 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 $w_{3_{(a,b,c,d,e,f,g,h)}}$, where spatial indexes a, c, e, g are ranging from 1 to N and channels indexes b, d, f, h are ranging from 1 to C. The use of $\boldsymbol{\mathcal{W}}^{[3]}$ in its most general form allows to take into account every possible interaction in the input but, at the same time, it exponentially increases the number of parameters. A major issue when scaling to higher orders is the exponential growth in the number of parameters and computational cost. In our case, the number of parameters in Eq. (1) depends on the order D of the polynomial and, even without considering orders lower than D, the parameters required are $(NC)^{D+1}$ (for instance, the use of $D = 3$ on an input 1024×196 will introduce approximately extra 10^{21} parameters). The number of parameters can be reduced by taking into account prior knowledge about the task or the nature of the input data [29], [26]. That is, we can select only a limited subset of all the possible combinations $x_{(c,d)}x_{(e,f)}x_{(g,h)}$ exploiting a particular structure of the tensor $\boldsymbol{\mathcal{W}}^{[3]}$. For example, assigning the same weight to a group of triplets will guarantee each of them to have same contribution on the output, while setting their weight to zero will cancel their impact altogether. The central idea of this paper is to factor the

interaction tensor $\mathcal{W}^{[3]}$ in a particular way, and extract only a minimal subset of 3rd order interactions from the input data. Practically, we replace the interaction tensor with matrices of smaller size using efficient operators commonly used in neural networks.

4 METHOD

In this section, we formally describe the proposed non-local module, called "Poly-NL", which is a novel operator capturing the same interactions as the non-local block [3] at a fraction of the computational cost in both space and time. Next, we adapt its formulation for Vision Transformers architectures :"Poly-SA" . Specifically, we first characterize the specific set of 3rd order interactions associated with non-local dependencies. Then, we define a method capable of accessing them without the need to compute the expensive pairwise similarity matrix. Lastly, we align the formulation to self-attention for transformers and extend it to allow spatial adaptivity.

4.1 Poly-NL layer

In [3], the authors introduce the "Non-local block", a learnable layer used to extract long-range dependencies in the input. This block 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

$$\mathbf{Z} = \mathbf{Y} + \mathbf{X} = f(\mathbf{X})g(\mathbf{X}) + \mathbf{X} \tag{6}$$

where $f\colon \mathbb{R}^{N \times C} \rightarrow \mathbb{R}^{N \times N}$ is a pairwise function that calculates the similarity for each pair of spatial positions, and $g\colon \mathbb{R}^{N \times C} \rightarrow \mathbb{R}^{N \times C}$, is a unary projection function computing a new representation for the input. In the case where $g(\mathbf{X})$ is a linear embedding and $f(\mathbf{X})$ is a dot-product, the formulation of the Non-local block can be defined as

$$\mathbf{Y}^{NL} = \left(\mathbf{X}\mathbf{W}_\theta \mathbf{W}_\phi^\top \mathbf{X}^\top\right)(\mathbf{X}\mathbf{W}_g) = \mathbf{X}\mathbf{W}_f \mathbf{X}^\top \mathbf{X}\mathbf{W}_g, \tag{7}$$

where $\mathbf{W}_\theta, \mathbf{W}_\phi, \mathbf{W}_g$ are learnable parameters of dimension $C \times C$. Formally, the dependencies singled-out by Eq. (7) are visible by writing the element-wise formulation as:

$$y^{\text{NL}}_{(a,b)} = \sum_{e}^{N} \sum_{d,f,h}^{C} w_{f_{(d,f)}} w_{g_{(h,b)}} x_{(a,d)} x_{(e,f)} x_{(e,h)}, \tag{8}$$

where the scalars $w_{f_{(d,f)}}$ and $w_{g_{(h,b)}}$ identify the elements of the matrices $\mathbf{W}_f$ and $\mathbf{W}_g$ at given indexes d, f, h, b.

Note that the Non-local block in Eq. (7) can be viewed as a special case of 3rd order polynomials defined in Eq. (4) and can be alternatively computed using a special tensor of parameters $\mathcal{W}^{[3]}_{\text{NL}}$ block-sparse, low-rank, and decomposed through the $\mathbf{W}_g$ and $\mathbf{W}_f$ matrices. In particular, Eq. (5) coincides with Eq. (8) upon choosing weights $w^{\text{NL}}_{3_{(a,b,c,d,e,f,g,h)}}$ null for every c different from a, g different from e, and equal to the Non-local block weights in the remaining cases.

The major drawback of this module is its complexity. The Non-local block generates the output $\mathbf{Y}$ by computing the dot-product between a similarity matrix $(\mathbf{X}\mathbf{W}_f\mathbf{X}^\top) \in \mathbb{R}^{N \times N}$ and the embedded input $(\mathbf{X}\mathbf{W}_g) \in \mathbb{R}^{N \times C}$. This matrix multiplication recalibrates the features at all input positions by 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. The matrix multiplication on the N dimension is at the core of the non-local processing but it introduces a quadratic term in computation that makes the complexity of this module equal to $O(N^2)$.

To address this drawback we propose Poly-NL, a non-local module that does not need any matrix multiplications along the spatial dimension N. Analogously to Eq. (6), 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 the additional α and β learnable scalars. The matrix $\mathbf{Y}^{\text{Poly-NL}}$ is the core of the Poly-NL layer and can be written as follows

$$\mathbf{Y}^{\text{Poly-NL}} = \Big(\Psi\big(\mathbf{X}\mathbf{W}_1 \odot \mathbf{X}\mathbf{W}_2\big) \odot \mathbf{X}\Big)\mathbf{W}_3, \tag{9}$$

where $\Psi\colon \mathbb{R}^{N \times C} \rightarrow \mathbb{R}^{N \times C}$ is an average pooling followed by an expansion function on the spatial positions, $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3 \in \mathbb{R}^{C \times C}$ are matrices of learnable parameters and $\odot$ indicates an element-wise multiplication.

The set of spatial interactions associated with Poly-NL is clearly highlighted in its element-wise formula as follows:

$$y^{\text{Poly-NL}}_{(a,b)} = \sum_{d,f,h}^{C} \sum_{e}^{N} w'_{1_{(h,d)}} w_{2_{(f,d)}} w_{3_{(d,b)}} x_{(a,d)} x_{(e,f)} x_{(e,h)}, \tag{10}$$

where the values $w'_{1_{(h,d)}}$ refer to the parameters of the matrix $\mathbf{W}'_1 = \mathbf{W}_1/N$. As visible from the comparison between the two element-wise formulas, Poly-NL and Non-Local block modules are closely connected. In the Non-local block, each element of the output matrix $y^{\text{NL}}_{(a,b)}$ is computed using the contribution of a set of triplets $x_{(a,d)}x_{(e,f)}x_{(e,h)}$, weighted using the learnable parameters $w_{f_{(d,f)}} w_{g_{(h,b)}}$. In Poly-NL, each element of the output matrix is computed using the contribution of *the exact same set of triplets*, weighted using a different set of learnable parameters. Moreover, Poly-NL is a special case of 3rd order polynomials and can be computed as in Eq. (4), upon choosing a $\mathcal{W}^{[3]}_{\text{Poly-NL}}$ block-sparse and decomposed by the matrices $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$. 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. A diagram of the proposed module is presented in Figure 1.

4.2 Poly-SA layer

In this section, we discuss and extend our polynomial non-local module from a transformer perspective. As introduced in [2] a self-attention block shares the same underlying attention mechanism with the non-local layer but differs in the choice of embedding matrices and the similarity function. The output $\mathbf{Y}$ of a self-attention block is computed as follows:

$$\mathbf{Y} = \sigma\left(\mathbf{Q}\mathbf{K}^\top/\sqrt{C_q}\right)\mathbf{V}. \tag{11}$$

In particular, a self-attention block takes as input a feature map $\mathbf{X} \in \mathbb{R}^{N \times C}$, of N tokens and C channels. The input is projected into keys $\mathbf{K} = \mathbf{X}\mathbf{W}_k$, queries $\mathbf{Q} = \mathbf{X}\mathbf{W}_q$ and values $\mathbf{V} = \mathbf{X}\mathbf{W}_v$ by the learnable parameters $\mathbf{W}_k \in \mathbb{R}^{C \times C_q}$, $\mathbf{W}_q \in \mathbb{R}^{C \times C_q}$, $\mathbf{W}_v \in \mathbb{R}^{C \times C_q}$. In a standard self-attention

layer, the keys and queries are used to compute an attention matrix and a softmax normalization function σ is used to obtain the weights on the values. Analogously to the non-local block, the complexity of Eq. (11) is quadratic with respect to the input size N and the matrix $\sigma\left(\mathbf{Q}\mathbf{K}^{\top}/\sqrt{C_q}\right) \in \mathbb{R}^{N\times N}$ holds the similarity among every possible pair of spatial positions $\mathbf{x_i}\mathbf{x_j}$ computed with an asymmetric softmax attention kernel. In the remainder of this work, we omit the channel-wise renormalization term $1/\sqrt{C_q}$ since we can equivalently renormalize input keys and queries. The element-wise form of Eq. (11) reads:

$$y_{(a,b)} = \sum_{e}^{N} \sigma\left(\sum_{d,f,g}^{C} w_{q_{(f,g)}} w_{k_{(d,g)}} x_{(a,d)} x_{(e,f)}\right) \sum_{h}^{C} x_{(e,h)} w_{v_{(h,b)}}. \tag{12}$$

In order to build a formulation that can consider the same set of triplets as the original self-attention, while having linear complexity with respect to both input dimensions, we proceed in two consecutive steps. Firstly, we start by viewing the attention mechanism of Eq. (11) through kernelization. We follow the established literature of [64], [59], [65], [66], [69] and consider a generalized self-attention

$$\mathbf{Y}^{SA} = \mathbf{A}\mathbf{V} = \left(\phi\left(\mathbf{Q}\right)\phi\left(\mathbf{K}\right)^{\top}\right)\mathbf{V}, \tag{13}$$

where the softmax attention matrix of the traditional self-attention is replaced with a generic similarity matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$. Specifically, here we consider $\mathbf{A}$ as a linear dot product of the rows in $\mathbf{Q}$ and $\mathbf{K}$ mapped via function ϕ designed to generate positive similarity measures. The associative property of matrix multiplications can be used to decrease the complexity of Eq. (13) :

$$\mathbf{Y}^{SA} = \left(\phi\left(\mathbf{Q}\right)\phi\left(\mathbf{K}\right)^{\top}\right)\mathbf{V} = \phi\left(\mathbf{Q}\right)\left(\phi\left(\mathbf{K}\right)^{\top}\mathbf{V}\right) = \phi\left(\mathbf{Q}\right)\mathbf{B}, \tag{14}$$

where the matrix $\mathbf{B} \in \mathbb{R}^{C_q\times C_q}$ characterizes the relationships among channels. This approximation makes the self-attention computation linear with respect to N, but also comes with the disadvantage of losing spatial adaptivity. In fact, Eq. (14) replaces a spatial mixing layer with an additional channel mixing block, that can be seen as a dynamic linear layer with weighting matrix $\mathbf{B}$ generated on the fly. This makes the response of all the layers in the ViT identical no matter the spatial position considered. It other words, Eq. (14) has two main characteristics: i) adapts the response of the layer depending on the input $\mathbf{X}$ ii) mixes all channels contribution together. While the first property allows for a switch from static to dynamic neural network [70], the second one is potentially redundant: ViT architectures already deploy a set of linear layers in their MLPs. Therefore, as second step to reduce complexity, we propose to consider only the diagonal of $\mathbf{B}$, that estimates the "importance" of each channel, and use this information to rescale the input:

$$\mathbf{Y}^{Poly-SA} = \phi\left(\mathbf{Q}\right) Diag(\mathbf{B}). \tag{15}$$

This layer mantains the dynamic aspect of Eq. (14), but reduces extra computations. Moreover, Eq. (15) has a direct link with the Poly-NL formulation. Similarly to Eq. (9), can be computed via average-pooling and element-wise multiplications only:

$$\mathbf{Y}^{Poly-SA} = \phi\left(\mathbf{Q}\right) \odot \Psi\Big(\phi\left(\mathbf{K}\right) \odot \mathbf{V}\Big), \tag{16}$$

where we consider ϕ to be the identity function and therefore the similarity kernel to be a linear kernel. Under this assumption, the element-wise form for Eq. (16) reads:

$$y^{\text{Poly-SA}}_{(a,b)} = \sum_{e}^{N}\sum_{d}^{C}\sum_{h,f}^{C} w_{k_{(d,b)}} w_{q_{(f,b)}} w_{v_{(h,b)}} x_{(e,f)} x_{(e,h)} x_{(a,d)}. \tag{17}$$

From the above equations, it is easy to recognize this formulation as a third-order polynomial block of linear complexity with respect to N and C. Furthermore, the dependencies captured by this module closely mimic those considered in the original self-attention: a side-by-side comparison of Eq. (12) and Eq. (17) shows how, for both modules, the output is computed by considering a weighted sum of the $x_{(e,f)}x_{(e,h)}x_{(a,d)}$ interactions of the input. Nevertheless, due to its diagonal form, the block of Eq. (16) scales every spatial position equally and, to be used as an efficient alternative to self-attention in ViT, still lacks a way to re-introduce spatial adaptivity in its formulation without losing its low runtime and memory requirements. We propose to overcome this problem by using two vectors of learnable parameters, $\mathbf{p}_1 \in R^N$ and $\mathbf{p}_2 \in R^N$, to automatically adjust the attention quantities to each spatial position. Lastly, to avoid convergence problems during training, we wrap the value of the dynamic weights with a sigma normalization function. In conclusion, the equation for Poly-SA reads

$$\mathbf{Y}^{Poly-SA} = \mathbf{Q} \odot \mathbf{p_1}\sigma\left(\mathbf{p_2}^{\top}\left(\mathbf{K} \odot \mathbf{V}\right)\right). \tag{18}$$

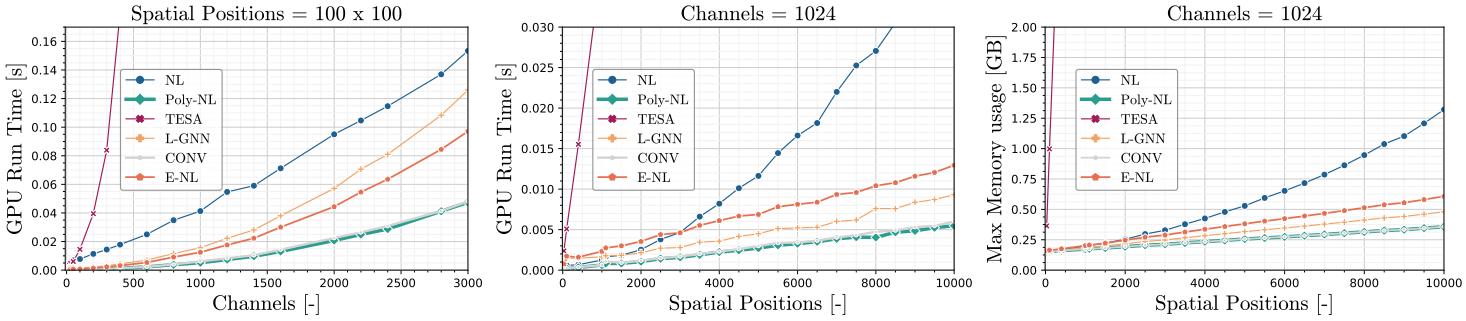
Compared to Poly-NL, Poly-SA maintains the embeddings as proposed in the original self-attention block, it keeps characteristics of spatial adaptivity, and use a σ normalization functions to ensure training stability. Therefore, its equation can be naturally extended to the multi-head case described in [2]. Specifically, to create the multi-head extensions, the self-attention formula of Poly-SA is first applied in parallel h times, then the output of each attention head is concatenated along the channel dimension (resulting in feature dimension hC_v), and finally the concatenated tensor is projected by $\mathbf{W}_{out} \in \mathbb{R}^{hC_v\times C}$ to produce the output $\mathbf{Z} = [\mathbf{Y}_0, \ldots, \mathbf{Y}_h]\mathbf{W}_{out} \in \mathbb{R}^{N\times C}$ having same dimensionality as the input .

5 Experiments

In this section, we demonstrate the ability of our modules to work as a drop-in replacement in existing computer-vision networks, while leaving the exploration of new transformer architectures to future work. Specifically, we showcase the capacity of Poly-NL and Poly-SA to enrich features on established CNN and ViT backbones.

5.1 Poly-NL for CNNs

We evaluate the proposed Poly-NL on three different tasks: object detection and instance segmentation on COCO [71], image classification on ImageNet [72], and face detection



(a) (b) (c)

Fig. 2: **Runtime and Peak memory consumption** performance comparison between Poly-NL and other non-local methods executed on a RTX2080 GPU. Poly-NL exhibits lower computational overhead than competing methods, which is of importance with an increasing number of spatial positions or channels.

Method	Att-F (G)	Att-P (M)	Top-1	Top-5
ResNet-50	0.0	0.0	75.62	92.68
+ NL Block	0.47	2.1	76.09	93.00
+ TESA	0.94	1.0	**76.49**	93.05
+ LatentGNN	0.13	0.6	75.28	92.33
+ EA	**0.11**	**0.5**	75.86	93.02
+ Poly-NL	0.14	0.7	76.30	**93.06**

(a) **ImageNet**

Method	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
MaskR-CNN	37.9	59.2	41.0	34.6	56.0	36.9
+ NL Block	38.8	60.6	42.0	35.4	57.3	**37.7**
+ TESA	**39.5**	**60.9**	**43.1**	35.4	57.2	37.5
+ LatentGNN	38.9	60.4	42.4	35.3	57.3	37.4
+ EA	38.9	60.3	42.2	35.4	57.2	37.7
+ Poly-NL	39.2	60.8	42.2	**35.4**	**57.4**	37.6

(b) **COCO**

TABLE 1: **Results of non-local variants** for image classification on ImageNet and instance segmentation on COCO. Performance metrics are reported next to FLOPS count (Att-F) and parameters count (Att-P) for each attention module, computed considering an input of size $14 \times 14 \times 1024$ and using the fvcore package.

on the WIDER FACE dataset [73]. We provide empirical evidence that Poly-NL outperforms previously proposed non-local blocks for CNNs while maintaining an optimal trade-off between efficiency and performance.

5.1.1 *Efficiency*

We examine the performance of five different layers (TESA [46], NL [3], LatentGNN [63], EA [62] and Poly-NL) to showcase how the proposed solution is able to process inputs with size unmanageable by other formulations. Figure 2 depicts the complexity overhead of various 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). We report the runtime on GPU[1] (Figure 2a, 2b) as a measure of time complexity and the peak memory usage on GPU as indicator of space complexity (Figure 2c). To highlight the impact of computing the non-local interactions, we also include a baseline layer of similar number of parameters (CONV), where no attention mechanism is used. Specifically, we pass the input through the same convolution layers used in a non-local block, but avoid the computation of the attention formula. All benchmarks were executed on single individual block for each method on an identical hardware, under comparable implementations and hyper-parameters. In particular, we highlight the efficiency trends of different long-range interactions computations. In particular, to isolate the contribution of the attention mechanism in the overall computations, we consider for all methods a single block with a single head and a channel reduction factor equal to 1. For each method, the values shown in the charts are the median of 20 runs. Here, the median is used to avoid the effect of potential outliers in the estimate. As can be seen from Figures 2b and 2c, increasing the number of spatial positions greatly impacts efficiency. Runtimes of TESA and NL, which both depend quadratically on the number of spatial positions N, quickly become impractical, even when N is relatively small. Efficient methods (EA, LatentGNN, Poly-NL) scale better with increasing N. Nonetheless, our method holds a competitive advantage in all cases, due to its lack of any matrix dot-product multiplications. As shown in Figure 2a, the number of channels is linearly proportional to the runtime performance of most methods, with the notable exception of TESA. However note that the proposed Poly-NL is considerably faster than the compared methods especially when the number of channels becomes significant. Finally, we highlight how Poly-NL is able to retain access to third order interactions with a complexity on par with the convolutional block (CONV) since by design it extracts the set of non-local interactions by avoiding the explicit computation of any attention matrix.

1. Test executed on an RTX2080 GPU

5.1.2 *Classification*

We evaluated our method on large-scale image classification, using ImageNet dataset [72], counting 1.28M training images split into 1000 classes. For all the experiments, we modify a ResNet-50 architecture [34] by inserting a non-local module at stage Res4 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 [74]. We compare our method against four different spatial non-local layers, the original non-

local block of [3], the efficient LatentGNN variant of [63], the Efficient Attention of [62] and the recently proposed TESA [46]. For the NL block we set the number of channels for the embedding matrices to be half of the input channels. Similarly, for TESA we a use a channel reduction factor of 2. For the LatentGNN block we use 2 latent kernels, latent dimension equal 100 and a channel reduction factor of 8. For EA we use a channel reduction factor of 8, a head count of 1 and softmax normalization. For Poly-NL we use a channel reduction factor of 4 by placing the block between a convolutional bottleneck and a convolutional expansion layers. Quantitative results are reported in Table (1a) and show the Top-1 and the Top-5 accuracy for the compared methods. Beyond the performance metrics, we also report the number of parameters and FLOPS count for the evaluated methods using the publicly available fvcore [2] package and assuming input of $N = 14$ and $C = 1024$. 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 [46], which is however computationally very demanding.

5.1.3 Instance Segmentation

We tested our method on object detection and instance segmentation, where the network processes an image and produces a pixel-wise mask that identifies both the category and the instance for each object. We use the Mask R-CNN baseline of [75] trained on MS-COCO 2017 dataset [71], which consists of 118k images as training set, 5k as validation set, and 20k as test set. 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 used as backbones ResNet-50 [34] architectures pre-trained on ImageNet [72]. 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. In all the experiments, we report the standard metrics of Average Precision AP, AP_{50}, and AP_{75} for both bounding boxes and segmentation masks.

Following prior work, we modify the Mask R-CNN backbone by adding one non-local layer right before the last residual block of Res4. This procedure highlights the ability of non-local blocks to boost features representation and consequently improve the quality of the candidate object bounding boxes. We compare our method against four different spatial non-local layers, the original non-local block of [3], the efficient LatentGNN variant of [63], the Efficient Attention of [62] and the recently proposed TESA [46]. For a fair comparison, we report the results from our training, achieved using public available source codes and hyper-parameters as provided by the respective authors. Quantitative results are summarized in Table 1b. When compared to the best performing method, TESA [46], 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 [3] and its efficient variants LatentGNN [63] and Efficient-Net [62], our method improves performance by 0.3% $\uparrow$ in AP_{box} while keeping linear computational complexity.

2. https://github.com/facebookresearch/fvcore

5.1.4 $2^{\mathbf{nd}}$ and $3^{\mathbf{rd}}$ Order Methods

In this section we briefly discuss the link between our polynomial framework for attention and popular building blocks for neural networks that can be framed as 2^{nd} order polynomial blocks. We discuss how the use of 2^{nd} order interactions could boost results on instance segmentation as well as classification but leave a more thorough exploration of this link to future work.

As visible in the element-wise formula of Eq. (3), the use of a block including all possible set of interactions introduces an intractably high amount of parameters and, for this reason, no existing layer can possibly implement such complete formulation. Nevertheless, there are some building blocks which use a low-rank parameter tensor $\boldsymbol{\mathcal{W}}^{[2]}$ decomposed through smaller matrices, and thus can be framed as special cases of this general formula. Specifically, we focus on two blocks closest to our work: Squeeze and Excitation [36] (SE) and the self-attention block proposed Global Context Network [37] (GC). Differently from Poly-NL, which is based on 3^{rd} order interactions, these blocks process only 2^{nd} order interactions. While 2^{nd} order methods avoid the quadratic complexity of non-local layers by design, they consider a smaller set of interactions (i.e. $\sum_{c,e}^{N}\sum_{d,f}^{C} x_{(c,d)}x_{(e,f)}$). To highlight the importance of higher-level interactions, we provide an ablation on instance segmentation on COCO in Table 2a. We followed the protocol described in Section 5.1.3, and modify a ResNet-50 backbone by adding one extra block at stage Res4 and ablate on various possible choices. We consider SE and GC blocks, together with a 2^{nd} adaptation of our method (as defined in Eq. (9)) obtained by replacing $(\mathbf{XW_1} \odot \mathbf{XW_2})$ by only $\mathbf{XW_1}$ (*Ours-2nd ord.*). This change makes our block close to SE, with the difference in the use of non-linearities and pooling functions. It is evident that in this task, where non-local patterns are crucial, substituting Poly-NL with 2^{nd} order methods causes a drop in performance. As visible from Table 2a, our 2^{nd} order results are on par with SE, but lower than Poly-NL, thus highlighting the importance of transitioning to 3^{rd} order interactions. Further, Poly-NL maintains better performance when compared with GC, which uses the same contribution for every position and thus lose part of the full interaction patterns. In contrast to this method, Poly-NL retains access to every triplet of the original NL providing in return better performance.

Moreover, we showcase how 2^{nd} and 3^{rd} order interactions can be used jointly to build more discriminative features. A lot of possible choices exists of 2^{nd} order blocks, and the best possible combination of 2^{nd} and 3^{rd} blocks is left to future work. Here, for the sake of simplicity, we choose a formulation $\mathbf{XW_p} * \mathbf{X}$ equivalent to ProdPoly block introduced in [76]. We follow the design of the classification experiment of Section 5.1.2 and test the contribution of each layer separately as well as their impact combined together.

Method	AP_{box}	AP_{box50}	AP_{box75}	AP_{mask}	AP_{mask50}	AP_{mask75}
MaskR-CNN (R50)	37.9	59.2	41.0	34.6	56.0	36.9
+SE	38.1	59.5	40.8	34.8	56.3	36.8
+*Ours-2nd ord.*	38.1	59.3	41.3	34.7	56.1	36.9
+GC	38.8	60.3	42.1	35.2	56.9	37.4
+ Poly-NL	**39.2**	**60.8**	**42.2**	**35.4**	**57.4**	**37.6**

(a) **Instance Segmentation (COCO)**

Method	Top-1	Top-5
ResNet-50	75.62	92.68
+ Poly-NL	76.30	93.06
+ ProdPoly	76.24	93.16
+ both	**76.57**	**93.25**

(b) **Classification (ImageNet)**

TABLE 2: **Comparison** between Poly-NL and various 2^{nd} order methods on Instance Segmentation (COCO) 2a. Poly-NL outperforms competitors thanks to the use of $3rd$ order interactions. On the right 2b, the combinations of Poly-NL and a 2^{nd} order method (Pi-Nets) improve results on ImageNet .

As visible from the results in Table 2b, the use of both 2^{nd} and 3^{rd} order interactions improve performance over the baseline, but it is their combination that achieves the best results, with a 0.95% $\uparrow$ in $Top-1$ and 0.57% $\uparrow$ in $Top-5$. Note that the increase in network complexity is negligible in this case.

5.2 Poly-SA for Transformers

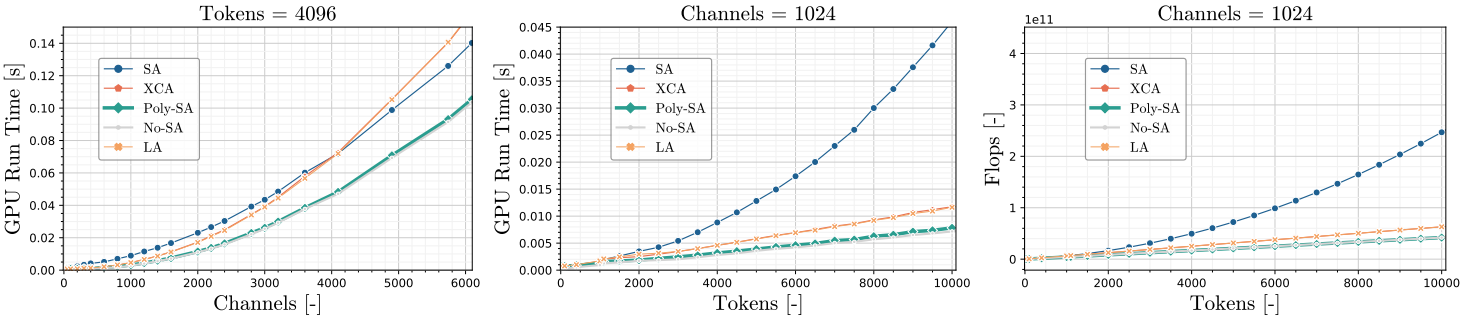
In a transformer architecture, MLP-based neural networks are equipped with self-attention blocks, capable of routing information among distant tokens and integrating long-range interactions in their output. In this section, we use the proposed Poly-SA to replace the standard (and more expensive) formulation of self-attention in various transformer architectures. We discuss the differences with respect to standard self-attention in term of efficiency and present its application on two different computer vision tasks.

5.2.1 Efficiency

Next, we discuss the time-complexity of Poly-SA and compare its efficiency with the standard multi-head self-attention SA [2], the efficient variant XCA [60] and LA [59], as these are closely related to our work. As a measure of time-complexity, we report runtime on GPU[1] together with floating-point operations per second, both expressed as a function of input sizes. Similar to the previous section, we define the absolute lower complexity bound in this scenario by reporting the complexity of a linear layer of comparable size (No-SA). In this scenario, XCA, LA and, Poly-SA keep a linear trend with respect to the input size, while a standard SA mechanism scales its complexity quadratically. Nonetheless, as can be seen from Figures 3c and 3b, Poly-SA outperforms other efficient variants by avoiding the need to access elements outside the cross-covariance diagonal and keeps a complexity on par with No-SA, the baseline linear layer where no attention mechanism is used. In addition, we showcase the scalability of our method by analyzing runtime on GPU as a function of the number of channels. As apparent in Figure 3a, while XCA and LA outperform standard SA only for $C < N$, the performance of Poly-SA is still adjacent to the No-SA lower bound even for a high number of channels. Nonetheless, compared to an MLP, Poly-SA is still able to access the same 3^{rd} order dependencies as a traditional multi-head self-attention.

5.2.2 Classification

We evaluated our Poly-SA block on a set of different state-of-the-art Vision Transformers. As a setup, we used the large-scale classification task on the ImageNet dataset, consisting of 1.3M training images, 50K validation images, and 1K object classes. We considered two well-known isotropic architectures with no downsampling layers and three hierarchical Vision transformer architectures. Concretely, as isotropic networks, we considered the traditional ViT [50] trained as in [50] (i.e. DeiT) and the XCiT architecture of [60]. As hierarchical networks, we selected two popular 4-stages Transformers, the Swin architecture of [51] and the MetaFormer architecture of [77]. Lastly, we considered the popular 3-stages Transformer of CvT [52]. For every network, we replace all the self-attention (or cross-covariance self-attention) with our Poly-SA block, leaving the remaining architecture unchanged. We used the publicly available code to replicate the original training setup. For CvT, we considered a stride equal to 1 for the Convolutional Projection VK. For DeiT and Swin Transformers, we use a 3x3 depthwise convolution as projection layer to mix heads content together. In the Metaformer architecture SA is used only in the last two stages. To ensure a fair comparison, we also implement Poly-SA only on the last two stages in our variant. We refer to the original papers for the rest of the architectural hyperparameters and training setup details. Table 3 reports architecture size, type of the attention block, parameters count of the architecture and FLOPS used to compute the attention blocks. It also reports performance for all the evaluated methods in terms of Top1 accuracy and shows how, without any hyperparameter and macro-design change, Poly-SA is capable of working as a drop-in replacement of traditional blocks in a variety of different cases. As apparent from the table, Poly-SA reduces the complexity without drastically compromising performance in all five architectures. It saves up to 58% of Flops with less than 2% drop in performance, showing consistent results among the evaluated architectures. Next, we evaluate the capacity of our method to provide consistent results across datasets and architectural sizes. To do so, we fix an architecture design and evaluate our method on the classification task using both the small scale dataset of CIFAR-100 and the large scale dataset of ImageNet. We select the state-of-the-art architecture XCiT [60] as baseline comparison. This transformer variant builds on top of the CaiT [53] network, and integrates convolution, layer-scale and class-attention modules to the traditional ViT [10]. Moreover, it replaces traditional self-attention with a cross-covariance self-attention, which is closely related to our work. As suggested by the authors, we followed the training setup as in DeiT [50] and trained for 400 epochs with the AdamW optimizer. We refer to the original paper for a detailed de-



(a) (b) (c)

Fig. 3: **Runtime and Flops** comparison between Poly-SA and two other self-attention method executed on a RTX2080 GPU. Poly-SA exhibits lower computational overhead than competing methods, with a complexity comparable to a linear layer using no attention mechanism.

Arch	Method	Size	Param (M)	Att-Flops (G)		Accuracy (T1)	
Swin	SA	Tiny	28.3	2.05		81.28	
Swin	Poly-SA	Tiny	26.1	1.60	**(-22%)**	80.66	**(-1.0%)**
XCiT	XCA	Tiny	6.7	0.57		75.66	
XCiT	Poly-SA	Tiny	6.7	0.39	**(-31%)**	75.21	**(-0.6%)**
MetaF.	SA	Small	16.5	0.74		80.97	
MetaF.	Poly-SA	Small	15.4	0.46	**(-38%)**	79.57	**(-1.7%)**
DeiT	SA	Small	5.7	0.52		72.16	
DeiT	Poly-SA	Small	5.4	0.28	**(-46%)**	71.48	**(-1.0%)**
CvT	SA	Tiny	20.0	3.50		82.3	
CvT	Poly-SA	Tiny	20.0	1.50	**(-58%)**	81.2	**(-1.3%)**

TABLE 3: **Classification Results** of Poly-SA in ViT for Large Scale Classification on Imagenet Dataset. Poly-SA is tested as a replacement for self-attention (SA) and cross-covariance attention (XCA) in five different transformer architectures: Swin Transformer [51], XCiT [60], DeiT [50], MetaFormer [77] (MetaF.), CvT [52]. Top1 accuracy is reported next to parameters count for each architecture (Params) and Flops count for the attention modules (Att-Flops). Poly-SA substantially reduces complexity of attention block in various ViT architecture while keeping performance close to baseline.

Method	Arch	Size	Top-1	Top-5
XCA	XCiT	Nano	68.30	88.92
Poly-SA	XCiT	Nano	68.52	88.88
XCA	XCiT	Tiny	75.66	92.99
Poly-SA	XCiT	Tiny	75.21	92.82
XCA	XCiT	Small	82.11	95.89
Poly-SA	XCiT	Small	81.50	95.77
XCA	XCiT	Medium	82.52	95.82
Poly-SA	XCiT	Medium	82.19	95.79

(a) **ImageNet**

Method	Arch	Size	Top-1	Top-5
XCA	XCiT	Nano	71.72	92.78
Poly-SA	XCiT	Nano	71.27	92.50
XCA	XCiT	Tiny	76.16	94.57
Poly-SA	XCiT	Tiny	75.56	94.07
XCA	XCiT	Small	81.89	95.83
Poly-SA	XCiT	Small	81.21	95.53
XCA	XCiT	Medium	81.53	95.45
Poly-SA	XCiT	Medium	80.70	95.29

(b) **CIFAR-100**

TABLE 4: **Classification Results** of Poly-SA tested as a replacement for cross-covariance attention (XCA) in XCiT transformer architecture [60] of various sizes on two different computer vision datasets. Poly-SA achieves comparable performance to baseline while reducing complexity of attention mechanism by up to 58%.

scription of this architecture and the related training setup. We ablate four different network sizes for XCiT and replace the original XCA block with the proposed Poly-SA, keeping the remaining architecture unchanged. We experiment with the "Nano" size (associated with 12 layers 128 channels and 4 attention heads), the "Tiny" size (with 12 layers 192 channels and 4 attention heads), "Small" (12 layers 384 channels and 8 attention heads) and "Medium" (24 layers 512 channels and 8 attention heads). In addition, we experiment with the CIFAR-100 dataset with a network of similar structure where the number of layers fixed at 6 and the number of channels is: Nano=128, Tiny = 192, Small=384 and Medium=512. The results for CIFAR-100 and ImageNet are reported on Table 4b and Table 4a, respectively. As visible from the results, despite using only the information on the main diagonal, Poly-SA still maintains performances comparable with XCA, and exhibits consistent performance across datasets and network sizes.

6 Discussion

This work is the first to link self-attention layers and polynomial blocks. We have provided a comprehensive theoretical framework for linear complexity attention and extensive empirical evidence that shows how the proposed Poly-NL

and Poly-SA modules can replace more-complex non-local and self-attention layers. Nevertheless, extensions to our work are still possible. In this section we briefly discuss potential directions, hoping to open a new interesting line of research around this topic.

Firstly, the use of different types of interactions within the self-attention formula is a promising direction. The spatial self-attention blocks only include NC^3 triplets which is actually a small subset of all the N^3C^3 possible 3^{rd} order interactions. For example, even for inputs with small size $N = 8^2$, the percentage of utilized triplets would be less than 0.025% of the total. Among these, many might not be very informative, so the question becomes how to efficiently extract the most meaningful interactions from this exponential space. Moreover, our experiments show how the inclusion of interactions of order different than the 3^{rd} can lead to performance improvement. This idea could be fully developed by exploring how different orders of interactions interact with each other and whether or not their best configuration is related to the specific task at hand. We also want to use Poly-NL on MindSpore (https://www.mindspore.cn/), which is a new deep learning computing framework. This exploration is left for future work. Second, spatial self-attention only involves one variant of 3^{rd} order polynomials. Future work could explore its relations with other instances of the general formula or even investigate strategies to automatically decompose the tensor of parameters $\boldsymbol{\mathcal{W}}^{[3]}$ into tractable and meaningful factorizations. Lastly, at their core, polynomials compute non-linear dependencies through multiplicative interactions. A complete analysis of the relation between non-linear interactions and non-linear activations for deep learning remains an interesting topic to explore.

7 Conclusion

In this work, we cast the non-local block as a 3^{rd} order polynomial in the form of multiplicative interactions between spatial locations of the input. Based on this fact, we propose a novel and fast embodiment of non-local layers named Poly-NL which is able to capture long-range dependencies equivalently to NL with a complexity that scales linearly with the size of the input in both computational complexity and memory requirements. Then, we extend our formulation to fit in Transformers literature and propose a multi-head self-attention module named Poly-SA. Poly-NL consistently outperforms other non-local modules on image recognition, instance segmentation, and face detection. Poly-SA achieves an equivalent, if not better, performance than traditional attention modules for ViT on image recognition and jigsaw puzzle reconstruction, while significantly reducing complexity up to the point of being equivalent to a standard linear layer. We expect our proposed framework to make a difference on applications where reducing FLOPs count is of critical importance, such as edge-devices or large-scale training with low ecological impact.

References

[1] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[3] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7794–7803.

[4] I. Bello, B. Zoph, A. Vaswani, J. Shlens, and Q. V. Le, "Attention augmented convolutional networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[5] T. Dai, J. Cai, Y. Zhang, S.-T. Xia, and L. Zhang, "Second-order attention network for single image super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 065–11 074.

[6] X. Wang, K. C. Chan, K. Yu, C. Dong, and C. Change Loy, "Edvr: Video restoration with enhanced deformable convolutional networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[7] M. Ott, S. Edunov, D. Grangier, and M. Auli, "Scaling neural machine translation," in *Proceedings of the Third Conference on Machine Translation (WMT)*, 2018.

[8] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens, "Stand-alone self-attention in vision models," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.

[9] A. Mohamed, D. Okhonko, and L. Zettlemoyer, "Transformers with convolutional context for asr," *arXiv preprint arXiv:1904.11660*, 2019.

[10] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[11] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[12] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," *arXiv preprint arXiv:1904.10509*, 2019.

[13] N. Kitaev, Ł. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," *arXiv preprint arXiv:2001.04451*, 2020.

[14] F. Babiloni, I. Marras, F. Kokkinos, J. Deng, G. Chrysos, and S. Zafeiriou, "Poly-nl: Linear complexity non-local layers with 3rd order polynomials," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 518–10 528.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[16] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9650–9660.

[17] S. M. Jayakumar, W. M. Czarnecki, J. Menick, J. Schwarz, J. Rae, S. Osindero, Y. W. Teh, T. Harley, and R. Pascanu, "Multiplicative interactions and where to find them," in *International Conference on Learning Representations*, 2019.

[18] G. Chrysos, M. Georgopoulos, J. Deng, J. Kossaifi, Y. Panagakis, and A. Anandkumar, "Augmenting deep classifiers with polynomial neural networks," in *European Conference on Computer Vision*, 2022.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] B. Krause, L. Lu, I. Murray, and S. Renals, "Multiplicative lstm for sequence modelling," *arXiv preprint arXiv:1609.07959*, 2016.

[21] J. B. Tenenbaum and W. T. Freeman, "Separating style and content with bilinear models," *Neural computation*, vol. 12, no. 6, pp. 1247–1283, 2000.

[22] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu, "Semantic segmentation with second-order pooling," in *European Conference on Computer Vision*. Springer, 2012, pp. 430–443.

[23] T.-Y. Lin, A. RoyChowdhury, and S. Maji, "Bilinear cnn models for fine-grained visual recognition," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1449–1457.

[24] C. Yu, X. Zhao, Q. Zheng, P. Zhang, and X. You, "Hierarchical bilinear pooling for fine-grained visual recognition," in *Proceedings*

of the European conference on computer vision (ECCV), 2018, pp. 574–589.

[25] R. Memisevic and G. Hinton, "Unsupervised learning of image transformations," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–8.

[26] R. Memisevic and G. E. Hinton, "Learning to represent spatial transformations with factored higher-order boltzmann machines," *Neural computation*, vol. 22, no. 6, pp. 1473–1492, 2010.

[27] T. J. Sejnowski, "Higher-order boltzmann machines," in *AIP Conference Proceedings*, vol. 151. American Institute of Physics, 1986, pp. 398–403.

[28] G. G. Chrysos, S. Moschoglou, G. Bouritsas, J. Deng, Y. Panagakis, and S. P. Zafeiriou, "Deep polynomial neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[29] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[30] F.-L. Fan, M. Li, F. Wang, R. Lai, and G. Wang, "Expressivity and trainability of quadratic networks," *arXiv preprint arXiv:2110.06081*, 2021.

[31] A. Dubey, F. Radenovic, and D. Mahajan, "Scalable interpretability via polynomials," in *Advances in neural information processing systems*, 2022.

[32] M. Choraria, L. Dadi, G. Chrysos, J. Mairal, and V. Cevher, "The spectral bias of polynomial neural networks," in *International Conference on Learning Representations*, 2022.

[33] Y. Wu, Z. Zhu, F. Liu, G. G. Chrysos, and V. Cevher, "Extrapolation and spectral bias of neural nets with hadamard product: a polynomial net study," in *Advances in neural information processing systems*, 2022.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[35] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3156–3164.

[36] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[37] Y. Cao, J. Xu, S. Lin, F. Wei, and H. Hu, "Gcnet: Non-local networks meet squeeze-excitation networks and beyond," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[38] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[39] X. Li, X. Hu, and J. Yang, "Spatial group-wise enhance: Improving semantic feature learning in convolutional networks," *arXiv preprint arXiv:1905.09646*, 2019.

[40] A. Buades, B. 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)*, vol. 2. IEEE, 2005, pp. 60–65.

[41] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the Eighteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., 2001, pp. 282–289.

[42] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Transactions on image processing*, vol. 16, no. 8, pp. 2080–2095, 2007.

[43] H. Wang, Y. Zhu, B. Green, H. Adam, A. Yuille, and L.-C. Chen, "Axial-deeplab: Stand-alone axial-attention for panoptic segmentation," in *European Conference on Computer Vision*. Springer, 2020, pp. 108–126.

[44] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European Conference on Computer Vision*. Springer, 2020, pp. 213–229.

[45] L. Ke, Y.-W. Tai, and C.-K. Tang, "Deep occlusion-aware instance segmentation with overlapping bilayers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4019–4028.

[46] F. Babiloni, I. Marras, G. Slabaugh, and S. Zafeiriou, "Tesa: Tensor element self-attention via matricization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 945–13 954.

[47] K. Yue, M. Sun, Y. Yuan, F. Zhou, E. Ding, and F. Xu, "Compact generalized non-local network," in *NeurIPS*, 2018, pp. 6511–6520.

[48] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3146–3154.

[49] Y. Mei, Y. Fan, Y. Zhang, J. Yu, Y. Zhou, D. Liu, Y. Fu, T. S. Huang, and H. Shi, "Pyramid attention networks for image restoration," *arXiv preprint arXiv:2004.13824*, 2020.

[50] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International Conference on Machine Learning*. PMLR, 2021, pp. 10 347–10 357.

[51] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," *arXiv preprint arXiv:2103.14030*, 2021.

[52] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, and L. Zhang, "Cvt: Introducing convolutions to vision transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 22–31.

[53] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou, "Going deeper with image transformers," *ICCV*, 2021.

[54] D. Zhou, Y. Shi, B. Kang, W. Yu, Z. Jiang, Y. Li, X. Jin, Q. Hou, and J. Feng, "Refiner: Refining self-attention for vision transformers," *arXiv preprint arXiv:2106.03714*, 2021.

[55] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *arXiv preprint arXiv:2009.06732*, 2020.

[56] L. Zhang, D. Xu, A. Arnab, and P. H. Torr, "Dynamic graph message passing networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3726–3735.

[57] Y. Chen, M. Rohrbach, Z. Yan, Y. Shuicheng, J. Feng, and Y. Kalantidis, "Graph-based global reasoning networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 433–442.

[58] S. Wang, B. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," *arXiv preprint arXiv:2006.04768*, 2020.

[59] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are rnns: Fast autoregressive transformers with linear attention," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5156–5165.

[60] A. El-Nouby, H. Touvron, M. Caron, P. Bojanowski, M. Douze, A. Joulin, I. Laptev, N. Neverova, G. Synnaeve, J. Verbeek *et al.*, "Xcit: Cross-covariance image transformers," *preprint*, 2021.

[61] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng, "a^2-nets: Double attention networks," *arXiv preprint arXiv:1810.11579*, 2018.

[62] Z. Shen, M. Zhang, H. Zhao, S. Yi, and H. Li, "Efficient attention: Attention with linear complexities," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 3531–3539.

[63] S. Zhang, X. He, and S. Yan, "Latentgnn: Learning efficient non-local relations for visual recognition," in *International Conference on Machine Learning*, 2019, pp. 7374–7383.

[64] Y.-H. H. Tsai, S. Bai, M. Yamada, L.-P. Morency, and R. Salakhutdinov, "Transformer dissection: A unified understanding of transformer's attention via the lens of kernel," *arXiv preprint arXiv:1908.11775*, 2019.

[65] K. M. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Q. Davis, A. Mohiuddin, L. Kaiser, D. B. Belanger, L. J. Colwell, and A. Weller, "Rethinking attention with performers," in *International Conference on Learning Representations*, 2021.

[66] H. Peng, N. Pappas, D. Yogatama, R. Schwartz, N. A. Smith, and L. Kong, "Random feature attention," *arXiv preprint arXiv:2103.02143*, 2021.

[67] S. P. Chowdhury, A. Solomou, A. Dubey, and M. Sachan, "On learning the transformer kernel," *arXiv preprint arXiv:2110.08323*, 2021.

[68] S. Zhai, W. Talbott, N. Srivastava, C. Huang, H. Goh, R. Zhang, and J. Susskind, "An attention free transformer," *arXiv preprint arXiv:2105.14103*, 2021.

[69] Z. Qin, W. Sun, H. Deng, D. Li, Y. Wei, B. Lv, J. Yan, L. Kong, and Y. Zhong, "cosformer: Rethinking softmax in attention," *arXiv preprint arXiv:2202.08791*, 2022.

[70] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," *arXiv preprint arXiv:1609.09106*, 2016.

[71] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.
[72] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.
[73] S. Yang, P. Luo, C.-C. Loy, and X. Tang, "Wider face: A face detection benchmark," in *CVPR*, 2016.
[74] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.
[75] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.
[76] G. G. Chrysos, S. Moschoglou, G. Bouritsas, Y. Panagakis, J. Deng, and S. Zafeiriou, "P-nets: Deep polynomial neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7325–7335.
[77] W. Yu, M. Luo, P. Zhou, C. Si, Y. Zhou, X. Wang, J. Feng, and S. Yan, "Metaformer is actually what you need for vision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 819–10 829.

Francesca Babiloni is a Ph.D. student at Imperial College London (ICL), supervised by Prof. Stefanos Zafeiriou. She received her master's degree from the University of Rome la Sapienza and worked at the KU Leuven University on Life-Long Learning. She is currently working at Huawei London R&D in the Computer Vision Team on low-complexity neural networks and deep learning for edge devices. Her research interest include machine learning and its application in computer vision.

Ioannis Marras received the BS degree from the Department of informatics of Aristotle University of Thessaloniki in 2005. From March 2005 until January 2010 was working as a Research Assistant, Teaching Assistant and Development engineer in the Artificial Intelligence Information Analysis (AIIA) laboratory at the Aristotle University of Thessaloniki, Greece, as well as in Informatics and Telematics Institute (CERTH), Greece. Since February 2011, he is working as Researcher in the Department of Computing at Imperial College studying dynamic 3D facial expression modelling/recognition and 2D/3D object tracking. From 2018 to 2021, he worked as researcher for Huawei London R&D on mobile devices applications.

Jiankang Deng received his Ph.D. in the Intelligent Behaviour Understanding Group (IBUG) at Imperial College London (ICL), supervised by Stefanos Zafeiriou and funded by the Imperial President's PhD Scholarships. His research interest focuses on face analysis (face detection, face alignment, face recognition and face generation). He is a reviewer in prestigious computer vision journals and conferences including T-PAMI, IJCV, CVPR, ICCV and ECCV. He is the main contributor of the widely used open-source platform Insightface.

Filippos Kokkinos is a Ph.D. student at UCL under the supervision of Prof. Iasonas Kokkinos. Before that, He was a member of the Computational Imaging Group (CIG) at Skoltech working on learnable regularizers. Since 2018, he is working on the development of unsupervised 3D reconstruction from single frames and videos, developing solution for incorporation of structured layers inside deep neural networks. He is research interest include machine learning, 3D reconstruction, computational photography, and inverse imaging problems.

Matteo Maggioni received B.Sc. and M.Sc. degrees in computer science from Politecnico di Milano (Italy), a Ph.D. degree in image processing from Tampere University of Technology (Finland), and a Post-Doc with the EEE Department in Imperial College London (UK). He is currently a research scientist with Huawei R&D (London, UK). His research interests include signal and image processing, computational photography, machine learning, and deep neural networks for computer vision.

Grigorios G. Chrysos is a Post-doctoral researcher at Ecole Polytechnique Federale de Lausanne (EPFL) following the completion of his PhD at Imperial College London (2020). Previously, he graduated from National Technical University of Athens with a Diploma/MEng in Electrical and Computer Engineering (2014). He has co-organized workshops in top conferences (CVPR, ICCV). He also organized a tutorial on polynomial nets (CVPR'22). His research interests lie in machine learning and its interface with computer vision. In particular, he is working on generative models, tensor decompositions and modelling high dimensional distributions; his recent work has been published in top-tier conferences (CVPR, ICML, ICLR, NeurIPS) and prestigious journals (T-PAMI, IJCV, T-IP). Grigorios has been recognized as an outstanding reviewer in top-tier conferences (ICML'21, ICLR'22, ICML'22, NeurIPS'22).

Philip Torr Philip H.S. Torr received the PhD degree from Oxford University, U.K. After working for another three years at Oxford as a research fellow, he worked for six years in Microsoft Research, first in Redmond, then in Cambridge, founding the vision side of the Machine Learning and Perception Group. He then became a professor in Computer Vision and Machine Learning at Oxford Brookes University, U.K. He is currently a professor at Oxford University.

Stefanos Zafeiriou (M'09) is a Professor in Machine Learning and Computer Vision with the Department of Computing, Imperial College London, U.K, and a Distinguishing Research Fellow with University of Oulu. He was a recipient of the Prestigious Junior Research Fellowships from Imperial College London in 2011 to start his own independent research group. He was the recipient of the President's Medal for Excellence in Research Supervision for 2016. He currently serves as an Associate Editor of the IEEE Transactions on Affective Computing and Computer Vision and Image Understanding journal. He has been a Guest Editor of over six journal special issues and co-organised over 13 workshops/special sessions on specialised computer vision topics in top venues, such as CVPR/FG/ICCV/ECCV. He has co-authored over 55 journal papers mainly on novel statistical machine learning methodologies applied to computer vision problems, such as 2-D/3-D face analysis, deformable object fitting and tracking, published in the most prestigious journals in his field of research, such as the IEEE T-PAMI, the International Journal of Computer Vision, the IEEE T-IP, the IEEE T-NNLS, the IEEE T-VCG, and the IEEE T-IFS, and many papers in top conferences. He has more than 14,000 citations to his work, h-index 57.