Deep Polynomial Neural Networks

Grigorios G. Chrysos,   Stylianos Moschoglou,   Giorgos Bouritsas,
Jiankang Deng,   Yannis Panagakis,   Stefanos Zafeiriou

Abstract—Deep Convolutional Neural Networks (DCNNs) are currently the method of choice both for generative, as well as for discriminative learning in computer vision and machine learning. The success of DCNNs can be attributed to the careful selection of their building blocks (e.g., residual blocks, rectifiers, sophisticated normalization schemes, to mention but a few). In this paper, we propose Π-Nets, a new class of DCNNs. Π-Nets are polynomial neural networks, i.e., the output is a high-order polynomial of the input. The unknown parameters, which are naturally represented by high-order tensors, are estimated through a collective tensor factorization with factors sharing. We introduce three tensor decompositions that significantly reduce the number of parameters and show how they can be efficiently implemented by hierarchical neural networks. We empirically demonstrate that Π-Nets are very expressive and they even produce good results without the use of non-linear activation functions in a large battery of tasks and signals, i.e., images, graphs, and audio. When used in conjunction with activation functions, Π-Nets produce state-of-the-art results in three challenging tasks, i.e. image generation, face verification and 3D mesh representation learning.

Index Terms—Polynomial neural networks, tensor decompositions, high-order polynomials, generative models, discriminative models, face verification

✦

1 INTRODUCTION

Deep Convolutional Neural Networks (DCNNs) [1], [2] have demonstrated impressive results in a number of tasks the last few years [2], [3], [4]. Arguably, the careful selection of architectural pipelines, e.g. skip connections [5], normalization schemes [6] etc., is significant, however the core structure relies on compositional functions of linear and nonlinear operators. Both theoretical [7], [8] and empirical studies reveal the limitations of the existing structure.

Recent empirical [9] and theoretical [10] results support that multiplicative interactions expand the classes of functions that can be approximated. Motivated by these findings, we study a new class of function approximators, which we coin $\Pi-$nets, where the output is a polynomial function of the input. Specifically, we model a vector-valued function $\boldsymbol{G}(\boldsymbol{z}) : \mathbb{R}^d \to \mathbb{R}^o$ by a high-order multivariate polynomial of the input $\boldsymbol{z}$, whose unknown parameters are naturally represented by high-order tensors. The number of parameters required to accommodate all higher-order correlations of the input explodes with the desired order of the polynomial. To that end, we cast polynomial parameters estimation as a coupled tensor factorization [11] that jointly factorizes all the polynomial parameters tensors. We introduce three joint decompositions with shared factors and exhibit the resulting hierarchical structures (i.e., architectures of neural networks).

In our preliminary works [12], [13], we introduced the concept of higher-order expansions for both generative and discriminative networks. In this work, our improvements are threefold. The concepts and the motivation behind each model are elaborated; the new intuitions will enable practitioners to devise new models tailored to their specific tasks. In addition, we extend the experimental results, e.g. include experiment in the challenging task of face verification and identification. Lastly, we conduct a thorough discussion on

• *GC, SM, GB, JD, SZ are with the Department of Computing, Imperial College London, SW7 2AZ, UK. YP is with the Department of Informatics and Telecommunications , University of Athens, GR.*
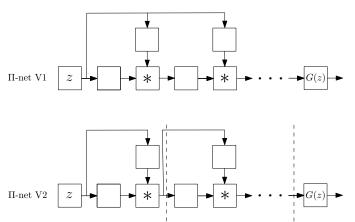*Corresponding author's e-mail: g.chrysos@imperial.ac.uk*

Fig. 1: In this paper we introduce a class of networks called $\Pi-$nets, where the output is a polynomial of the input. The input in this case, $\boldsymbol{z}$, can be either the latent space of Generative Adversarial Network for a generative task or an image in the case of a discriminative task. Our polynomial networks can be easily implemented.

several challenging topics that require further work on this new class of neural networks.

In particular, the paper bears the following contributions:

- A new family of neural networks (called $\Pi-$nets) where the output is a high-order polynomial of the input is introduced. To avoid the combinatorial explosion in the number of parameters of polynomial activation functions [14], our $\Pi-$nets cast polynomial parameters estimation as a coupled tensor factorization with shared factors (please see Fig. 1 for an indicative schematic representation).
- The proposed architectures are applied in a) generative models such as GANs, and b) discriminative networks. Additionally, the polynomial architectures are used to learn high-dimensional distributions without non-linear activation functions.
- We convert state-of-the-art baselines using the proposed

$\Pi-$nets and show how they can largely improve the expressivity of the baseline. We demonstrate it conclusively in a battery of tasks (i.e., generation, classification and face verification/identification). Our architectures are applicable to many different signals (e.g. images, meshes, and audio) and outperform the prior art.

The rest of the paper is organized as follows: Sec. 2 summarizes the related work. In Sec. 3 we introduce the polynomial networks and showcase the resulting architectures for three decompositions. Experiments with generative and discriminative models without using activation functions are conducted in Sec. 4, while the bulk of the experiments is conducted in Sec. 5. The existing limitations and future directions of the polynomial networks are discussed in Sec. 6, while Sec. 7 concludes the paper.

2 RELATED WORK

Expressivity of (deep) neural networks: The last few years, (deep) neural networks have been applied to a wide range of applications with impressive results. The performance boost can be attributed to a host of factors including: a) the availability of massive datasets [15], [16], b) the machine learning libraries [17], [18] running on massively parallel hardware, c) training improvements. The training improvements include a) optimizer improvement [19], [20], b) augmented capacity of the network [21], c) regularization tricks [6], [22], [23], [24]. However, the paradigm for each layer remains largely unchanged for several decades: each layer is composed of a linear transformation and an element-wise activation function. Despite the variety of linear transformations [1], [2], [25] and activation functions [26], [27] being used, the effort to extend this paradigm has not drawn much attention to date.

Recently, hierarchical models have exhibited stellar performance in learning expressive generative models [9], [28], [29]. For instance, the recent BigGAN [28] performs a hierarchical composition through skip connections from the noise $\boldsymbol{z}$ to multiple resolutions of the generator. A similar idea emerged in StyleGAN [9], which is an improvement over the Progressive Growing of GANs (ProGAN) [30]. As ProGAN, StyleGAN is a highly-engineered network that achieves compelling results on synthesized 2D images. In order to provide an explanation on the improvements of StyleGAN over ProGAN, the authors adopt arguments from the style transfer literature [31]. We believe that these improvements can be better explained under the light of our proposed polynomial function approximation. Despite the hierarchical composition proposed in these works, we present an intuitive and mathematically elaborate method to achieve a more precise approximation with a polynomial expansion. We also demonstrate that such a polynomial expansion can be used in both image generation (as in [9], [28]), image classification, and graph representation learning.

Polynomial networks: Polynomial relationships have been investigated in two specific categories of networks: a) self-organizing networks with hard-coded feature selection, b) pi-sigma networks.

The idea of learnable polynomial features can be traced back to Group Method of Data Handling (GMDH) [32][1]. GMDH learns partial descriptors that capture quadratic correlations between two predefined input elements. In [34], more input elements are allowed, while higher-order polynomials are used. The input to each partial descriptor is predefined (subset of the input elements), which

---

1. This is often referred to as the first deep neural network [33].

does not allow the method to scale to high-dimensional data with complex correlations.

Shin *et al*. [35] introduce the pi-sigma network, which is a neural network with a single hidden layer. Multiple affine transformations of the data are learned; a product unit multiplies all the features to obtain the output. Improvements in the pi-sigma network include regularization for training in [36] or using multiple product units to obtain the output in [37]. The pi-sigma network is extended in sigma-pi-sigma neural network (SPSNN) [38]. The idea of SPSNN relies on summing different pi-sigma networks to obtain each output. SPSNN also uses a predefined basis (overlapping rectangular pulses) on each pi-sigma sub-network to filter the input features. Even though such networks use polynomial features or products, they do not scale well in high-dimensional signals. In addition, their experimental evaluation is conducted only on signals with known ground-truth distributions (and with up to 3 dimensional input/output), unlike the modern generative models where only a finite number of samples from high-dimensional ground-truth distributions is available.

Another instance of such polynomial networks is through multiplicative interactions. Recently, there is a surge of methods [39], [40], [41] reporting superior performance through multiplicative interactions. The work of [10] provides a theoretical understanding on why such connections might be beneficial. The aforementioned works model interactions of second or third order. Polynomial networks can be seen as high-order generalizations of such multiplicative interactions [10], [39], [40].

3 METHOD

Notation[2]: Tensors are symbolized by calligraphic letters, e.g., $\boldsymbol{\mathcal{X}}$, while matrices (vectors) are denoted by uppercase (lowercase) boldface letters e.g., $\boldsymbol{X}$, $(\boldsymbol{x})$. The *mode-m vector product* of $\boldsymbol{\mathcal{X}}$ with a vector $\boldsymbol{u} \in \mathbb{R}^{I_m}$ is denoted by $\boldsymbol{\mathcal{X}} \times_m \boldsymbol{u}$. A core tool in our analysis is the CP decomposition [42]. By considering the mode-1 unfolding of an Mth-order tensor $\boldsymbol{\mathcal{X}}$, the CP decomposition can be written in matrix form as [42]: $\boldsymbol{X}_{(1)} \doteq \boldsymbol{U}_{[1]} \left(\odot_{m=M}^{2} \boldsymbol{U}_{[m]} \right)^{T}$ where $\{\boldsymbol{U}_{[m]}\}_{m=1}^{M}$ are the factor matrices.

We want to learn a function approximator where each element of the output x_j, with $j \in [1, o]$, is expressed as a polynomial[3] of all the input elements z_i, with $i \in [1, d]$. That is, we want to learn a function $G : \mathbb{R}^d \rightarrow \mathbb{R}^o$ of order $N \in \mathbb{N}$, such that:

$$
\begin{aligned}
x_j = G(\boldsymbol{z})_j = \beta_j + \boldsymbol{w}_j^{[1]^T} \boldsymbol{z} + \boldsymbol{z}^T \boldsymbol{W}_j^{[2]} \boldsymbol{z} + \\
\boldsymbol{\mathcal{W}}_j^{[3]} \times_1 \boldsymbol{z} \times_2 \boldsymbol{z} \times_3 \boldsymbol{z} + \cdots + \boldsymbol{\mathcal{W}}_j^{[N]} \prod_{n=1}^{N} \times_n \boldsymbol{z}
\end{aligned}
\tag{1}
$$

where $\beta_j \in \mathbb{R}$, and $\left\{ \boldsymbol{\mathcal{W}}_j^{[n]} \in \mathbb{R}^{\prod_{m=1}^{n} \times_m d} \right\}_{n=1}^{N}$ are parameters for approximating the output x_j. The correlations (of the input elements z_i) up to N^{th} order emerge in (1). A more compact expression of (1) is obtained by vectorizing the outputs:

$$
\boldsymbol{x} = G(\boldsymbol{z}) = \sum_{n=1}^{N} \left(\boldsymbol{\mathcal{W}}^{[n]} \prod_{j=2}^{n+1} \times_j \boldsymbol{z} \right) + \boldsymbol{\beta}
\tag{2}
$$

---

2. A detailed tensor notation is deferred to the supplementary.

3. The theorem of [43] guarantees that any smooth function can be approximated by a polynomial. The approximation of multivariate functions is covered by an extension of the Weierstrass theorem, e.g., in [44] (pg 19).

TABLE 1: Nomenclature

Symbol	Dimension(s)	Definition
n, N	$\mathbb{N}$	Polynomial term order, total approximation order.
k	$\mathbb{N}$	Rank of the decompositions.
$\boldsymbol{z}$	$\mathbb{R}^d$	Input to the polynomial approximator.
$\boldsymbol{C}, \boldsymbol{\beta}$	$\mathbb{R}^{o \times k}, \mathbb{R}^o$	Parameters in all decompositions.
$\boldsymbol{A}_{[n]}, \boldsymbol{S}_{[n]}, \boldsymbol{B}_{[n]}$	$\mathbb{R}^{d \times k}, \mathbb{R}^{k \times k}, \mathbb{R}^{\omega \times k}$	Matrix parameters in the hierarchical decomposition.
$\odot, *$	-	Khatri-Rao product, Hadamard product.

TABLE 2: Single polynomial models (Sec. 3.1)

Name	Schematic	Recursive eq.
CCP	Fig. 2	(4)
NCP	Fig. 3	(7)
NCP-Skip	Fig. 4	(8)

where $\boldsymbol{\beta} \in \mathbb{R}^o$ and $\left\{ \boldsymbol{\mathcal{W}}^{[n]} \in \mathbb{R}^{o \times \prod_{m=1}^{n} \times_m d} \right\}_{n=1}^{N}$ are the learnable parameters. This form of (2) allows us to approximate any smooth function (for large N), however the parameters grow with $\mathcal{O}(d^N)$.

A variety of methods, such as pruning [45], [46], special linear operators [47] with reduced parameters, parameter sharing/prediction [48], [49], can be employed to reduce the parameters. The aforementioned approaches are post-processing techniques, i.e., given a (pre-trained) network, they reduce the parameters of the specific network. Instead, we design two principled ways which allow an efficient implementation. The first method relies on performing an off-the-shelf tensor decomposition on (2), while the second considers the final polynomial as the product of lower-degree polynomials.

3.1 Single polynomial

A tensor decomposition on the parameters is a natural way to reduce the parameters and to implement (2) with a neural network. Below, we demonstrate how three such decompositions result in novel architectures for a neural network training. The main symbols are summarized in Table 1, while the equivalence between the recursive relationship and the polynomial is analyzed in the supplementary.

Model 1: CCP (Coupled CP decomposition)

Instead of factorizing each parameter tensor $\boldsymbol{\mathcal{W}}^{[n]}$ individually we propose to jointly factorize all the parameter tensors using a coupled CP decomposition [42] with a specific pattern of factor sharing. To illustrate the factorization, we assume a third order approximation ($N = 3$), and then provide the recursive relationship that can scale to arbitrary expansion.

Let us assume that the parameter tensors admit the following coupled CP decomposition with the factors corresponding to lower-order levels of approximation being shared across all parameters tensors. That is:

- Let $\boldsymbol{W}^{[1]} = \boldsymbol{C}\boldsymbol{U}_{[1]}^T$, be the parameters for first level of approximation.
- Let $\boldsymbol{\mathcal{W}}^{[2]}$ being a superposition of of two weights tensors, namely $\boldsymbol{\mathcal{W}}^{[2]} = \boldsymbol{\mathcal{W}}_{1:2}^{[2]} + \boldsymbol{\mathcal{W}}_{1:3}^{[2]}$, with $\boldsymbol{\mathcal{W}}_{i:j}^{[2]}$ denoting parameters associated with the second order interactions across the ith and j^{th} order of approximation. By enforcing the CP decomposition of the above tensors to share the factor with tensors corresponding to lower-order of approximation we obtain in matrix form: $\boldsymbol{W}_{(1)}^{[2]} = \boldsymbol{C}(\boldsymbol{U}_{[3]} \odot \boldsymbol{U}_{[1]})^T + \boldsymbol{C}(\boldsymbol{U}_{[2]} \odot \boldsymbol{U}_{[1]})^T$.
- Similarly, we enforce the third-order parameters tensor to admit the following CP decomposition (in matrix form) $\boldsymbol{W}_{(1)}^{[3]} = \boldsymbol{C}(\boldsymbol{U}_{[3]} \odot \boldsymbol{U}_{[2]} \odot \boldsymbol{U}_{[1]})^T$. Note that all but the $\boldsymbol{U}_{[3]}$ factor matrices are shared in the factorization of tensors capturing polynomial parameters for the first and second order of approximation.

The parameters are $\boldsymbol{C} \in \mathbb{R}^{o \times k}, \boldsymbol{U}_{[m]} \in \mathbb{R}^{d \times k}$ for $m = 1, 2, 3$. Then, (2) for $N = 3$ is written as:

$$
\begin{aligned}
G(\boldsymbol{z}) = {} & \boldsymbol{\beta} + \boldsymbol{C}\boldsymbol{U}_{[1]}^T \boldsymbol{z} + \boldsymbol{C}\left(\boldsymbol{U}_{[3]} \odot \boldsymbol{U}_{[1]}\right)^T (\boldsymbol{z} \odot \boldsymbol{z}) + \\
& \boldsymbol{C}\left(\boldsymbol{U}_{[2]} \odot \boldsymbol{U}_{[1]}\right)^T (\boldsymbol{z} \odot \boldsymbol{z}) + \\
& \boldsymbol{C}\left(\boldsymbol{U}_{[3]} \odot \boldsymbol{U}_{[2]} \odot \boldsymbol{U}_{[1]}\right)^T (\boldsymbol{z} \odot \boldsymbol{z} \odot \boldsymbol{z})
\end{aligned} \tag{3}
$$

Using the Lemma 1 (provided in the supplementary), we can transform the (3) into a neural network as depicted in Fig. 2.

The CCP factorization generalizes to N^{th} order expansion. The recursive relationship for the Nth order approximation is:

$$
\boldsymbol{x}_n = \left(\boldsymbol{U}_{[n]}^T \boldsymbol{z}\right) * \boldsymbol{x}_{n-1} + \boldsymbol{x}_{n-1} \tag{4}
$$

for $n = 2, \ldots, N$ with $\boldsymbol{x}_1 = \boldsymbol{U}_{[1]}^T \boldsymbol{z}$ and $\boldsymbol{x} = \boldsymbol{C}\boldsymbol{x}_N + \boldsymbol{\beta}$. The parameters $\boldsymbol{C} \in \mathbb{R}^{o \times k}, \boldsymbol{U}_{[n]} \in \mathbb{R}^{d \times k}$ for $n = 1, \ldots, N$ are learnable.
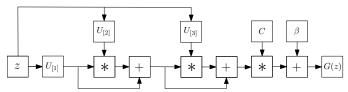

Fig. 2: Schematic illustration of the CCP (for third order approximation). Symbol $*$ refers to the Hadamard product.

Model 2: NCP (Nested coupled CP decomposition)

Instead of explicitly separating the interactions between layers, we can utilize a joint hierarchical decomposition on the polynomial parameters. Let us first introduce learnable hyper-parameters $\left\{\boldsymbol{b}_{[n]} \in \mathbb{R}^\omega\right\}_{n=1}^{N}$, which act as scaling factors for each parameter tensor. Therefore, we modify (2) to:

$$
G(\boldsymbol{z}) = \sum_{n=1}^{N} \left(\boldsymbol{\mathcal{W}}^{[n]} \times_2 \boldsymbol{b}_{[N+1-n]} \prod_{j=3}^{n+2} \times_j \boldsymbol{z} \right) + \boldsymbol{\beta}, \tag{5}
$$

with $\left\{\boldsymbol{\mathcal{W}}^{[n]} \in \mathbb{R}^{o \times \omega \times \prod_{m=1}^{n} \times_m d}\right\}_{n=1}^{N}$. Similarly to CCP, we demonstrate the decomposition assuming a third order approximation ($N = 3$), and then provide the general recursive relationship.

To estimate the parameters (in $N = 3$ expansion) we jointly factorize all parameter tensors by employing nested CP decomposition with parameter sharing as follows (in matrix form):

- First order parameters : $\boldsymbol{W}_{(1)}^{[1]} = \boldsymbol{C}(\boldsymbol{A}_{[3]} \odot \boldsymbol{B}_{[3]})^T$.
- Second order parameters:
$$
\boldsymbol{W}_{(1)}^{[2]} = \boldsymbol{C}\left\{ \boldsymbol{A}_{[3]} \odot \left[\left(\boldsymbol{A}_{[2]} \odot \boldsymbol{B}_{[2]} \right) \boldsymbol{S}_{[3]} \right] \right\}^T .
$$

- Third order parameters:

$$\boldsymbol{W}^{[3]}_{(1)} = \boldsymbol{C}\Bigg\{\boldsymbol{A}_{[3]} \;\odot\; \Bigg[\Big(\boldsymbol{A}_{[2]} \;\odot\; \Big\{\Big(\boldsymbol{A}_{[1]} \;\odot\; \boldsymbol{B}_{[1]}\Big)\boldsymbol{S}_{[2]}\Big\}\Big)\boldsymbol{S}_{[3]}\Bigg]^{T}\Bigg\}$$

with $\boldsymbol{C} \in \mathbb{R}^{o \times k}, \boldsymbol{A}_{[n]} \in \mathbb{R}^{d \times k}, \boldsymbol{S}_{[n]} \in \mathbb{R}^{k \times k}, \boldsymbol{B}_{[n]} \in \mathbb{R}^{\omega \times k}$ for $n = 1, \ldots, N$. Altogether, (5) for $N = 3$ is written as:

$$G(\boldsymbol{z}) = \boldsymbol{\beta} + \boldsymbol{C}(\boldsymbol{A}_{[3]} \odot \boldsymbol{B}_{[3]})^{T}(\boldsymbol{z} \odot \boldsymbol{b}_{[3]}) +$$
$$\boldsymbol{C}\Bigg\{\boldsymbol{A}_{[3]} \odot \Bigg[\Big(\boldsymbol{A}_{[2]} \odot \boldsymbol{B}_{[2]}\Big)\boldsymbol{S}_{[3]}\Bigg]\Bigg\}^{T}\Big(\boldsymbol{z} \odot \boldsymbol{z} \odot \boldsymbol{b}_{[2]}\Big) + \quad (6)$$
$$\boldsymbol{C}\Bigg\{\boldsymbol{A}_{[3]} \odot \Bigg[\Big(\boldsymbol{A}_{[2]} \odot \Big\{\Big(\boldsymbol{A}_{[1]} \odot \boldsymbol{B}_{[1]}\Big)\boldsymbol{S}_{[2]}\Big\}\Big)\boldsymbol{S}_{[3]}\Bigg]\Bigg\}^{T}\boldsymbol{\mu}$$

with $\boldsymbol{\mu} = \Big(\boldsymbol{z} \odot \boldsymbol{z} \odot \boldsymbol{z} \odot \boldsymbol{b}_{[1]}\Big)$. Using Lemma1 and further algebraic operations (see Sec. 3.2 in the supplementary), (6) can be implemented by a neural network as depicted in Fig. 3.

The recursive relationship for N^{th} order approximation is defined as:

$$\boldsymbol{x}_n = \Big(\boldsymbol{A}^{T}_{[n]}\boldsymbol{z}\Big) * \Big(\boldsymbol{S}^{T}_{[n]}\boldsymbol{x}_{n-1} + \boldsymbol{B}^{T}_{[n]}\boldsymbol{b}_{[n]}\Big) \quad (7)$$

for $n = 2, \ldots, N$ with $\boldsymbol{x}_1 = \Big(\boldsymbol{A}^{T}_{[1]}\boldsymbol{z}\Big) * \Big(\boldsymbol{B}^{T}_{[1]}\boldsymbol{b}_{[1]}\Big)$ and $\boldsymbol{x} = \boldsymbol{C}\boldsymbol{x}_N + \boldsymbol{\beta}$. The parameters $\boldsymbol{C} \in \mathbb{R}^{o \times k}, \boldsymbol{A}_{[n]} \in \mathbb{R}^{d \times k}, \boldsymbol{S}_{[n]} \in \mathbb{R}^{k \times k}, \boldsymbol{B}_{[n]} \in \mathbb{R}^{\omega \times k}, \boldsymbol{b}_{[n]} \in \mathbb{R}^{\omega}$ for $n = 1, \ldots, N$, are learnable.
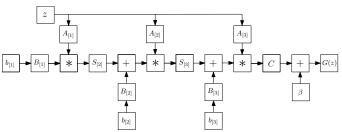
Fig. 3: Schematic illustration of the NCP (for third order approximation). Symbol $*$ refers to the Hadamard product.

Model 3: NCP-Skip (Nested coupled CP decomposition with skip)

The expressiveness of NCP can be further extended using a skip connection (motivated by CCP). The new model uses a nested coupled decomposition and has the following recursive expression:

$$\boldsymbol{x}_n = \Big(\boldsymbol{A}^{T}_{[n]}\boldsymbol{z}\Big) * \Big(\boldsymbol{S}^{T}_{[n]}\boldsymbol{x}_{n-1} + \boldsymbol{B}^{T}_{[n]}\boldsymbol{b}_{[n]}\Big) + \boldsymbol{x}_{n-1} \quad (8)$$

for $n = 2, \ldots, N$ with $\boldsymbol{x}_1 = \Big(\boldsymbol{A}^{T}_{[1]}\boldsymbol{z}\Big) * \Big(\boldsymbol{B}^{T}_{[1]}\boldsymbol{b}_{[1]}\Big)$ and $\boldsymbol{x} = \boldsymbol{C}\boldsymbol{x}_N + \boldsymbol{\beta}$. The learnable parameters are the same as in NCP, however the difference in the recursive form results in a different polynomial expansion and thus architecture.

Comparison between the models

All three models (see Table 2 for names and schematics) are based on a polynomial expansion, however their recursive forms and employed decompositions differ.

CCP is a straightforward coupled decomposition and is a proof of concept that polynomials can learn high-dimensional distributions. NCP illustrates how to convert a popular CNN/linear model of the form $\boldsymbol{x}_k = \boldsymbol{S}^{T}_{[k]}\boldsymbol{x}_{k-1} + \boldsymbol{b}_{[k]}$ to a polynomial (i.e., $\boldsymbol{x}_k = (\boldsymbol{A}^{T}_{[k]}\boldsymbol{z}) * (\boldsymbol{S}^{T}_{[k]}\boldsymbol{x}_{k-1} + \boldsymbol{b}_{[k]})$). Similarly, NCP-Skip
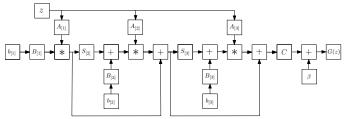
Fig. 4: Schematic illustration of the NCP-Skip (for third order approximation). The difference from Fig. 3 is the skip connections added in this model.

demonstrates how a residual network can be transformed into a polynomial. In Sec. 4.2, an experimental comparison is conducted; all three models are assessed on image generation. In the remainder of the paper, for comparison purposes we use the NCP by default for the image generation and NCP-Skip for the image classification. In all cases, to mitigate stability issues that might emerge during training, we employ certain normalization schemes that constrain the magnitude of the gradients.

3.2 Product of polynomials

Instead of using a single polynomial, we express the function approximation as a product of polynomials. The product is implemented as successive polynomials where the output of the ith polynomial is used as the input for the $(i + 1)^{th}$ polynomial. The concept is visually depicted in Fig. 5; each polynomial expresses a second order expansion. Stacking N such polynomials results in an overall order of 2^N. Trivially, if the approximation of each polynomial is B and we stack N such polynomials, the total order is B^N. The product does not necessarily demand the same order in each polynomial, the expressivity and the expansion order of each polynomial can be different and dependent on the task, e.g., for generative tasks that the resolution increases progressively, the expansion order could increase in the last polynomials. In all cases, the final order will be the product of each polynomial power.

There are two main benefits of the product over the single polynomial: a) it allows using different decompositions (e.g., like in Sec. 3.1) and expressive power for each polynomial; b) it requires much fewer parameters for achieving the same order of approximation. Given the benefits of the product of polynomials, we assume below that a product of polynomials is used, unless explicitly mentioned otherwise. The respective model of product polynomials is called ProdPoly.

3.3 Task-dependent input/output

The aforementioned polynomials are a function $\boldsymbol{x} = G(\boldsymbol{z})$, where the input/output are task-dependent. For a generative task, e.g., learning a decoder, the input $\boldsymbol{z}$ is typically some low-dimensional noise, while the output is a high-dimensional signal, e.g., an image. For a discriminative task the input $\boldsymbol{z}$ is an image; for a domain adaptation task the signal $\boldsymbol{z}$ denotes the source domain and $\boldsymbol{x}$ the target domain.

4 PROOF OF CONCEPT

In this section, we conduct motivational experiments in both generative and discriminative tasks to demonstrate the expressivity of $\Pi-$nets. Specifically, the networks are implemented **without activation functions**, i.e., only linear operations (e.g., convolutions) and Hadamard products are used. In this setting, the output is linear or multi-linear with respect to the parameters.
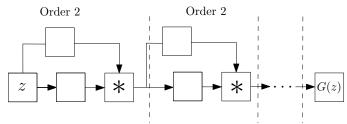
Fig. 5: Abstract illustration of the ProdPoly. The input variable z on the left is the input to a 2nd order expansion; the output of this is used as the input for the next polynomial (also with a 2nd order expansion) and so on. If we use N such polynomials, the final output $G(z)$ expresses a 2^N order expansion. In addition to the high order of approximation, the benefit of using the product of polynomials is that the model is flexible, in the sense that each polynomial can be implemented as a different decomposition of Sec. 3.1.

4.1 Generation without activation functions

One of the most popular generative models is Generative Adversarial Nets (GANs) [52]. GANs typically consist of two deep networks, namely a generator G and a discriminator D. G is a decoder, which receives as input a random noise vector $z \in \mathbb{R}^d$ and outputs a sample $x = G(z)$. D receives as input both $G(z)$ and real samples and tries to differentiate the fake and the real samples. During training, both G and D compete against each other. We design a GAN, where the generator is implemented as a product of polynomials (using the NCP), while the discriminator of [4] is used. The activation functions in-between the layers are removed[4]. In the image-related tasks, we add a single activation function (hyperbolic tangent) in the output space of the generator.

Qualitative results on surface generation: The approximation power of a single polynomial is scrutinized in four different manifolds in 2D/3D. The generator does not have a single activation function. In Fig. 6, both the target manifolds and the synthesized samples are visualized. Notice that Gabriel's Horn (i.e., the second from the left) has a parametric form $[x, \alpha \cdot \frac{\cos t}{x}, \alpha \cdot \frac{\sin t}{x}]$ for $t \in [0, 160\pi]$ and $x \in [1, 4]$. The dependence on both the sinusoidal and the function $\frac{1}{x}$ poses challenges for a polynomial expansion. However, we demonstrate that a single polynomial can approximate the distribution.

Qualitative results on image generation: Three experiments are conducted with a polynomial generator (Fashion-Mnist, Color-Mnist[4] and YaleB). We perform a linear interpolation in the latent space when trained with Fashion-Mnist [50] and with YaleB [51] and visualize the results in Figs. 7. Note that the linear interpolation generates plausible images and traverses through i) different categories, e.g., trousers to sneakers or trousers to t-shirts, ii) different colors and digits, iii) extreme illuminations in parts of the face.

Quantitative results on image generation: CIFAR10 [53] is used for the evaluation. The generator of [4] with three residual blocks is used; we omit the activation functions in the generator. That is the 'Orig' baseline; then we use NCP to convert this generator into a single polynomial (called PolyGAN). If we replace the Hadamard operator with concatenation, we obtain the 'Concat' method; while by adding FC polynomials to PolyGAN, we convert

4. Additional details are deferred to the supplementary material.

it into a product of polynomials. We note that 'Orig' and 'Concat' are designed to work along with non-linearities, however we add them as a reference metric.

The Inception score (IS) and Frechet Inception Distance (FID) are reported in Tab. 3. The single polynomial, i.e., PolyGAN, demonstrates the benefits of polynomial expansion; when combined with additional polynomials to increase the expansion order, we obtain scores that have never emerged for generators without activation functions.

4.2 Model comparison

The three different decompositions of Sec. 3.1 are experimentally scrutinized. Each model is implemented as a product of polynomials. The base for each respective model is one of the three decompositions, i.e., CCP, NCP, NCP-Skip. To evaluate the expressivity of each model, we train each without activation functions, i.e., similarly to the previous paragraph.

The three models are originally compared in fashion image generation. The outcomes, visualized in Fig. 8, demonstrate similar generation properties.

In Fig. 9, samples of the three models (in the product of polynomials case) are synthesized; all three models can generate faces without activation functions. The three models share similar generation quality.

4.3 Classification without activation functions

The performance of polynomial networks without activation functions is also assessed in a classification setting. We use ResNet without activation functions for classification. Residual Network (ResNet) [5], [40] and its variants [3], [54], [55], [56], [57] have been applied to diverse tasks including object detection and image generation [4], [58], [59]. The core component of ResNet is the residual block; the tth residual block is expressed as $z_{t+1} = z_t + Cz_t$ for input z_t.

We modify each residual block to express a higher order interaction, which can be achieved with NCP-Skip. The output of each residual block is the input for the next residual block, which makes our ResNet a product of polynomials. We conduct a classification experiment with CIFAR10 [53] (10 classes) and CIFAR100 [60] (100 classes). Each residual block is modified in two ways: a) all the activation functions are removed, b) it is converted into an ith order expansion with $i \in [2, 5]$. The second order expansion (for the tth residual block) is expressed as $z_{t+1} = z_t + Cz_t + \left(Cz_t\right) * z_t$; higher orders are constructed similarly by performing a Hadamard product of the last term with z_t (e.g., for third order expansion it would be $z_{t+1} = z_t + Cz_t + \left(Cz_t\right) * z_t + \left(Cz_t\right) * z_t * z_t$). The following two variations are evaluated: a) a single residual block is used in each 'group layer', b) two blocks are used per 'group layer'. The latter variation is equivalent to ResNet18 without activations.

Each method is trained for 120 epochs with batch size 128. The SGD optimizer is used with initial learning rate of 0.1. The learning rate is multiplied with a factor of 0.1 in epochs 40, 60, 80, 100. Each experiment is repeated 10 times; the mean accuracy is reported in Fig. 10 and Table 5. We note that the same trends emerge in both datasets[5]. The performance remains similar

5. The performance of the baselines, i.e., ResNet18 **without** activation functions, is 0.391 and 0.168 for CIFAR10 and CIFAR100 respectively. However, we emphasize that the original ResNet was not designed to work without activation functions. The performance of ResNet18 in CIFAR10 and CIFAR100 **with** activation functions is 0.945 and 0.769 respectively.

(a) Target manifolds   (b) Synthesized samples

Fig. 6: Target vs synthesized manifolds using generators **without any activation functions.** Each generator is a single polynomial (Sec. 3.1).
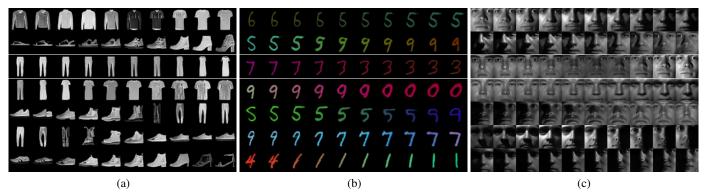

(a)   (b)   (c)

Fig. 7: Linear interpolation in the latent space of ProdPoly when trained on a) fashion images [50], b) colored MNIST, c) facial images [51]. Note that the generator does not include any activation functions in between the linear blocks (Sec. 4.1). All the images are synthesized; the image on the leftmost column is the source, while the one in the rightmost is the target synthesized image.

TABLE 3: IS/FID scores on CIFAR10 [53] generation without activation functions.

	Image generation without activation functions on CIFAR10			
Model	Unsupervised		Conditional	
	IS (↑)	FID (↓)	IS (↑)	FID (↓)
Orig	3.25 ± 0.28	211.33 ± 12.47	4.47 ± 0.21	156.67 ± 12.29
Concat	3.17 ± 0.27	233.13 ± 12.47	2.47 ± 0.19	192.08 ± 13.49
PolyGAN	5.52 ± 0.18	78.23 ± 7.23	6.43 ± 0.11	53.50 ± 2.71
ProdPoly	$\mathbf{6.95 \pm 0.18}$	$\mathbf{40.45 \pm 1.40}$	$\mathbf{7.50 \pm 0.13}$	$\mathbf{36.77 \pm 1.85}$

TABLE 4: Training accuracy in the CIFAR10 classification.

Order	bl1111	bl2222	Order	bl1111	bl2222
2	0.99	1.00	4	0.98	0.99
3	0.98	0.99	5	0.96	0.98

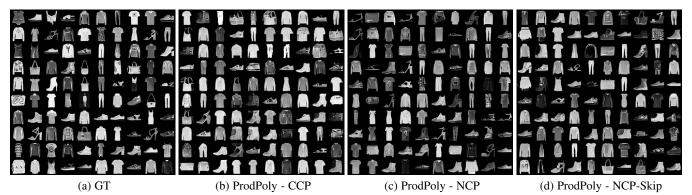(a) GT   (b) ProdPoly - CCP   (c) ProdPoly - NCP   (d) ProdPoly - NCP-Skip

Fig. 8: Comparison of the proposed models in fashion image [50] generation without activation functions.

irrespective of the amount of residual blocks in the group layer. The performance is affected by the order of the expansion, i.e., higher orders cause a decrease in the accuracy. Our conjecture is that this can be partially attributed to overfitting (note that a 3rd order expansion for the 2222 block - in total 8 res. units - yields a polynomial of 3^8 power). The training accuracy of the CIFAR10 experiment in Table 4 is $> 95\%$. Nevertheless, in all cases without activations the accuracy is close to the original ResNet18 with activation functions[5].

5 EXPERIMENTS

We conduct four experiments against state-of-the-art models in three diverse tasks: image generation, image classification, face verification/identification and graph representation learning. In each case, the baseline considered is converted into an instance of our family of Π-nets and the two models are compared.

5.1 Image generation

The robustness of ProdPoly in image generation is assessed in two different architectures/datasets below.

(a) GT   (b) ProdPoly - CCP   (c) ProdPoly - NCP   (d) ProdPoly - NCP-Skip
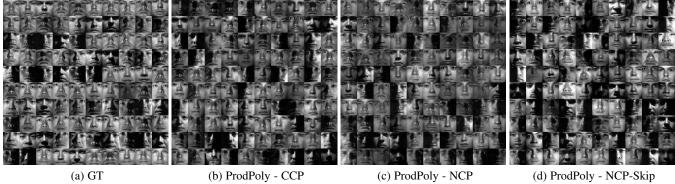
Fig. 9: Comparison of the proposed models in facial image [51] generation without activation functions.
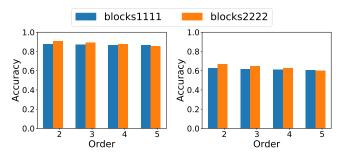

Fig. 10: Image classification accuracy without using activation functions in the residual blocks[5]. The schematic on the left is on CIFAR10 classification, while the one on the right is on CIFAR100 classification.

TABLE 5: Quantitative results on classification without the activation functions. The symbol N abbreviates the order of expansion in each residual block, while 'Acc' abbreviates the accuracy.

# blocks	N	Acc CIFAR10	Acc CIFAR100
[1, 1, 1, 1]	2	0.876 ± 0.003	0.626 ± 0.004
	3	0.870 ± 0.003	0.616 ± 0.003
	4	0.868 ± 0.002	0.609 ± 0.002
	5	0.864 ± 0.001	0.606 ± 0.003
[2, 2, 2, 2]	2	0.907 ± 0.003	0.667 ± 0.003
	3	0.891 ± 0.001	0.648 ± 0.002
	4	0.877 ± 0.003	0.626 ± 0.004
	5	0.856 ± 0.006	0.598 ± 0.007

SNGAN on CIFAR10: In the first experiment, the architecture of SNGAN [4] is selected as a strong baseline on CIFAR10 [53]. The baseline includes 3 residual blocks in the generator and the discriminator.

The generator is converted into a Π-net, where each residual block is a single order of the polynomial. We implement two versions, one with a single polynomial (NCP) and one with product of polynomials (where each polynomial uses NCP). In our implementation $\boldsymbol{A}_{[n]}$ is a thin FC layer, $(\boldsymbol{B}_{[n]})^T \boldsymbol{b}_{[n]}$ is a bias vector and $\boldsymbol{S}_{[n]}$ is the transformation of the residual block. Other than the aforementioned modifications, the hyper-parameters (e.g., discriminator, learning rate, optimization details) are kept the same as in [4].

Each network was run for 10 times and the mean and variance are reported. The popular Inception Score (IS) [61] and the Frechet Inception Distance (FID) [62] are used for quantitative evaluation. Both scores extract feature representations from a pre-trained classifier (the Inception network [63]).

TABLE 6: IS/FID scores on CIFAR10 [53] generation. The scores of [58], [59] are added from the respective papers as using similar residual based generators. The scores of [64], [65], [66] represent alternative generative models. ProdPoly outperforms the compared methods in both metrics.

Image generation on CIFAR10		
Model	IS ($\uparrow$)	FID ($\downarrow$)
SNGAN	8.06 ± 0.10	19.06 ± 0.50
NCP(Sec. 3.1)	8.30 ± 0.09	17.65 ± 0.76
ProdPoly	$\mathbf{8.49 \pm 0.11}$	$\mathbf{16.79 \pm 0.81}$
CSGAN- [58]	7.90 ± 0.09	-
WGAN-GP- [59]	7.86 ± 0.08	-
CQFG- [66]	8.10	18.60
EBM [64]	6.78	38.2
GLANN [65]	-	46.5 ± 0.20

The quantitative results are summarized in Table 6. In addition to SNGAN and our two variations with polynomials, we have added the scores of [58], [59], [64], [65], [66] as reported in the respective papers. Note that the single polynomial already outperforms the baseline, while the ProdPoly boosts the performance further and achieves a substantial improvement over the original SNGAN.

StyleGAN on FFHQ: StyleGAN [9] is the state-of-the-art architecture in image generation. The generator is composed of two parts, namely: (a) the mapping network, composed of 8 FC layers, and (b) the synthesis network, which is based on ProGAN [30] and progressively learns to synthesize high quality images. The sampled noise is transformed by the mapping network and the resulting vector is then used for the synthesis network. As discussed in the introduction, StyleGAN is already an instance of the Π-net family, due to AdaIN. Specifically, the kth AdaIN layer is $\boldsymbol{h}_k = (\boldsymbol{A}_k^T \boldsymbol{w}) * n(c(\boldsymbol{h}_{k-1}))$, where n is a normalization, c the convolution operator and $\boldsymbol{w}$ is the transformed noise $\boldsymbol{w} = MLP(\boldsymbol{z})$ (mapping network). This is equivalent to our NCP model by setting $\boldsymbol{S}_{[k]}^T$ as the convolution operator.

In this experiment we illustrate how simple modifications, using our family of products of polynomials, further improve the representation power. We make a minimal modification in the mapping network, while fixing the rest of the hyper-parameters. In particular, we convert the mapping network into a polynomial (specifically a NCP), which makes the generator a product of two polynomials.

The Flickr-Faces-HQ Dataset (FFHQ) dataset [9] which includes $70,000$ images of high-resolution faces is used. All the images are resized to 256×256. The best FID scores of the two methods (in 256×256 resolution) are **6.82** for ours and

Fig. 11: Samples synthesized from ProdPoly (trained on FFHQ).

7.15 for the original StyleGAN, respectively. That is, our method improves the results by 5%. Synthesized samples of our approach are visualized in Fig. 11.

5.2 Classification

We perform two experiments on classification: a) audio classification, b) image classification.

Audio classification: The goal of this experiment is twofold: a) to evaluate ResNet on a distribution that differs from that of natural images, b) to validate whether higher-order blocks make the model more expressive. The core assumption is that we can increase the expressivity of our model, or equivalently we can use fewer residual blocks of higher-order to achieve performance similar to the baseline.

The performance of ResNet is evaluated on the Speech Commands dataset [67]. The dataset includes $60,000$ audio files; each audio contains a single word of a duration of one second. There are 35 different words (classes) with each word having $1,500 - 4,100$ recordings. Every audio file is converted into a mel-spectrogram of resolution 32×32.

The baseline is a ResNet34 architecture; we use second-order residual blocks to build the Prodpoly-ResNet to match the performance of the baseline. The quantitative results are added in Table 7. The two models share the same accuracy, however Prodpoly-ResNet includes 38% fewer parameters. This result validates our assumption that our model is more expressive and with even fewer parameters, it can achieve the same performance.

TABLE 7: Speech classification with ResNet. The accuracy of the compared methods is similar, but Prodpoly-ResNet has 38% fewer parameters. The symbol '# par' abbreviates the number of parameters (in millions).

Speech Commands classification with ResNet			
Model	# blocks	# par	Accuracy
ResNet34	$[3, 4, 6, 3]$	21.3	0.951 ± 0.002
Prodpoly-ResNet	$[3, 3, 3, 2]$	**13.2**	0.951 ± 0.002

Classification on CIFAR: The expressivity of the polynomial networks is also assessed on CIFAR10 classification. That is, we can reduce the number of residual blocks (of higher-order) to achieve the same performance as vanilla residual blocks.
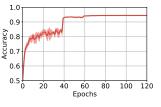
We select the ResNet18 and ResNet34 as baselines, while the rest training details are similar to Sec. 4.3. Prodpoly-ResNet is implemented by employing second-order residual blocks.
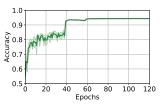
In Table 8 the two different ResNet baselines are compared against Prodpoly-ResNet on CIFAR10; the respective Prodpoly-ResNet models have the same accuracy. However, each Prodpoly-ResNet has $\sim 40\%$ less parameters than the respective baseline. In addition, we visualize the test accuracy for ResNet18 and the respective Prodpoly-ResNet in Fig. 12. The test error of the two models is similar throughout the training.

The same experiment is repeated on CIFAR100 with ResNet34 as the baseline. Table 9 exhibits a similar pattern. That is, the test

TABLE 8: Image classification on CIFAR10 with ResNet. The # abbreviates 'number of', while the parameters are measured in millions. The term 'block' abbreviates a 'residual block'. Note that each baseline, e.g. ResNet18, has the same performance with the respective Prodpoly-ResNet, but significantly more parameters.

CIFAR10 classification with ResNet			
Model	# blocks	# params (M)	Accuracy
ResNet18	$[2, 2, 2, 2]$	11.2	0.945 ± 0.000
Prodpoly-ResNet	$[2, 2, 1, 1]$	**6.0**	0.945 ± 0.001
ResNet34	$[3, 4, 6, 3]$	21.3	0.948 ± 0.001
Prodpoly-ResNet	$[3, 3, 2, 2]$	**13.0**	0.949 ± 0.002

(a) ResNet18   (b) Prodpoly-ResNet

Fig. 12: The test accuracy of (a) ResNet18 and (b) the respective Prodpoly-ResNet are plotted (CIFAR10 training). The two models perform similarly throughout the training, while ours has 46% less parameters. The width of the highlighted region denotes the standard deviation of each model.

accuracy of ResNet34 and Prodpoly-ResNet is similar, however Prodpoly-ResNet has $\sim 30\%$ less parameters.

TABLE 9: CIFAR100 classification with ResNet. The accuracy of the compared methods is similar, but Prodpoly-ResNet has 30% less parameters.

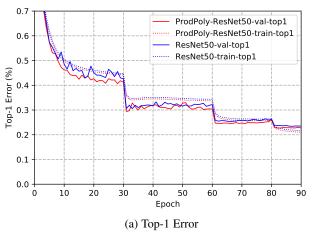| CIFAR100 classification with ResNet			
Model	# blocks	# params (M)	Accuracy
ResNet34	$[3, 4, 6, 3]$	21.3	0.769 ± 0.003
Prodpoly-ResNet	$[3, 4, 3, 2]$	**14.7**	0.769 ± 0.001

Classification on ImageNet: We perform a large-scale classification experiment on ImageNet [68]. To stabilize the training, the second order of each residual block is normalized with a hyperbolic tangent unit. SGD with momentum 0.9, weight decay 10^{-4} and a mini-batch size of 512 is used. The initial learning rate is set to 0.2 and decreased by a factor of 10 at $30, 60$, and 80 epochs. Models are trained for 90 epochs from scratch, using linear warm-up of the learning rate during first five epochs according to [69].

The Top-1 and Top-5 error throughout the training is visualized in Fig. 13, while the validation results are added in Table 10. For a fair comparison, we report the results from our training in both the original ResNet and Prodpoly-ResNet[6]. Prodpoly-ResNet consistently improves the performance with a negligible increase in computational complexity and model size. Remarkably, Prodpoly-ResNet50 achieves a single-crop Top-1 validation error of 22.827% and Top-5 validation error of 6.431%, exceeding ResNet50 by 0.719% and 0.473%, respectively.

TABLE 10: ImageNet classification results of ResNet50 and the proposed Prodpoly-ResNet.

Model	Top-1 error (%)	Top-5 error (%)
ResNet50	23.546	6.904
Prodpoly-ResNet50	22.827 $(\downarrow 0.719)$	6.431 $(\downarrow 0.473)$

6. The performance of the original ResNet [5] is inferior to the one reported here and in [70].
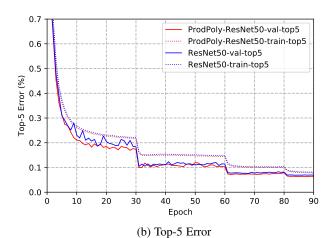
(a) Top-1 Error

(b) Top-5 Error

Fig. 13: Top-1 and Top-5 error curves on the ImageNet dataset.

5.3 Face verification and identification

We scrutinize the performance of the Π-nets on the challenging task of face recognition. The architecture of the current state-of-the-art method of ArcFace [71] is a ResNet, which we can convert into a polynomial network using the NCP-Skip.

Training Data: The data of MS1M-RetinaFace dataset [72], [73] consist the training images; all face images inside MS1M-RetinaFace are pre-processed to the size of 112×112 based on the five facial landmarks predicted by RetinaFace [74]. In total, there are 5.1M images of 93K identities.

Testing Data: The performance is compared on widely used face verification data-sets (e.g., LFW [75], CFP [76], AgeDB [77], CPLFW [78], CALFW [79] and RFW [80]). Besides, we also extensively test the proposed method on large-scale benchmarks (e.g., IJB-B [81], IJB-C [82] and MegaFace [83]); the fundamental statistics of all the datasets are summarized in Table 11. During testing, we only keep the feature embedding network without the fully connected layer and extract the 512-D features for each normalized face. To get the embedding features for templates (e.g., IJB-B [81] and IJB-C [82]), we simply calculate the feature center of all images from the template or all frames from the video.

Training Details: For the baseline embedding network, we employ the widely used CNN architecture, ResNet50. Specifically, we follow [71] to set the feature scale s to 64 and choose the angular margin m at 0.5. The batch size is set to 512 with momentum 0.9 and weight decay $5e-4$, while we decrease the learning rate in iterations 100K, 160K, 220K. The training finishes after 30 epochs and is trained on 8 NVIDIA 2080ti (11GB) GPUs.

The baseline residual block is converted into a second-order residual block to build the Prodpoly-ResNet, while we keep all the other settings exactly the same as the baseline. For the baseline ResNet50, the model size is 175MB and the computational complexity is 12.6G FLOPS. For Prodpoly-ResNet50, the model size is 182MB and the computational complexity is 13.2G FLOPS. **Results on LFW, CFP-FF, CFP-FP, CPLFW, AgeDB-30, CALFW and RFW.** LFW [75] contains 13,233 web-collected images from 5,749 different identities, with limited variations in pose, age, expression and illuminations. CFP [76] consists of collected images of celebrities in frontal and profile views. On CFP, there are two evaluation protocols: CFP-Frontal-Frontal and CFP-Frontal-Profile. CFP-Frontal-Profile is very challenging as

TABLE 11: Face datasets for training and testing. "(P)" and "(G)" refer to the probe and gallery set, respectively.

Datasets	#Identity	#Image
MS1MV2	93K	5.1M
LFW [75]	5,749	13,233
CFP [76]	500	7,000
AgeDB [77]	568	16,488
CPLFW [78]	5,749	11,652
CALFW [79]	5,749	12,174
RFW [80]	11,430	40,607
RFW-Caucasian [80]	2,959	10,196
RFW-Indian [80]	2,984	10,308
RFW-Asian [80]	2,492	9,688
RFW-African [80]	2,995	10,415
MegaFace [83]	530 (P)	1M (G)
IJB-B [81]	1,845	76.8K
IJB-C [82]	3,531	148.8K

the pose gap within positive pairs is around $90°$. CPLFW [78] was collected by crowd-sourcing efforts to seek the pictures of people in LFW with pose gap as large as possible from the Internet. CALFW [79] is similar to CALFW, but from the perspective of age difference. AgeDB [77] contains manually annotated images. In this paper, we use the evaluation protocol with 30 years gap [71]. RFW [80] is a benchmark for measuring racial bias, which consists of four test subsets, namely Caucasian, Indian, Asian and African. The quantitative results of the comparisons are exhibited in Table 12. On LFW and CFP-FP, the results of ResNet50 and Prodpoly-ResNet50 are similar to face verification on semi-frontal faces is saturated. Nevertheless, Prodpoly-ResNet50 significantly outperforms ResNet50 on CFP-FP, CPLFW, AgeDB-30, CALFW and RFW, indicating that the proposed method can enhance the robustness of the embedding features under pose variations, age variations and racial variations.

Results on IJB-B and IJB-C. The IJB-B dataset [81] contains $1,845$ subjects with 21.8K still images and 55K frames from $7,011$ videos. The IJB-C dataset [81] is a further extension of IJB-B, having $3,531$ subjects with 31.3K still images and 117.5K frames from $11,779$ videos. On IJB-B and IJB-C datasets, there are two evaluation protocols, 1:1 verification and 1:N identification.

In Figure 14, ROC curves of ResNet50 and Prodpoly-ResNet50 under 1:1 verification protocol on IJB-B and IJB-C is plotted. On IJB-B, there are $12,115$ templates with $10,270$ genuine matches

TABLE 12: Verification performance (%) of ResNet50 and the proposed Prodpoly-ResNet50 on LFW, CFP-FF, CFP-FP, CPLFW, AgeDB-30, CALFW and RFW (Caucasian, Indian, Asian and African).

Method	ResNet50	Prodpoly-ResNet50
LFW	99.733 ± 0.309	$\mathbf{99.833} \pm 0.211$ (↑ 0.100)
CFP-FF	99.871 ± 0.135	$\mathbf{99.886} \pm 0.178$ (↑ 0.015)
CFP-FP	98.800 ± 0.249	$\mathbf{98.986} \pm 0.274$ (↑ 0.186)
CPLFW	92.433 ± 1.245	$\mathbf{93.317} \pm 1.343$ (↑ 0.884)
AgeDB-30	98.233 ± 0.655	$\mathbf{98.467} \pm 0.623$ (↑ 0.234)
CALFW	95.917 ± 1.209	$\mathbf{96.233} \pm 1.114$ (↑ 0.316)
RFW-Caucasian	99.333 ± 0.307	$\mathbf{99.700} \pm 0.100$ (↑ 0.367)
RFW-Indian	98.567 ± 0.507	$\mathbf{99.300} \pm 0.296$ (↑ 0.733)
RFW-Asian	98.333 ± 0.435	$\mathbf{98.950} \pm 0.350$ (↑ 0.617)
RFW-African	98.650 ± 0.329	$\mathbf{99.417} \pm 0.227$ (↑ 0.767)

(a) CMC

(b) ROC

Fig. 15: CMC and ROC curves of ResNet50 and the proposed Prodpoly-ResNet50 on MegaFace. Results are evaluated on the refined MegaFace dataset [71].

and 8M impostor matches. On IJB-C, there are $23,124$ templates with $19,557$ genuine matches and $15,639$K impostor matches. The proposed method surpasses the baseline by a clear margin. The comparison of TAR in Table 13 illustrates that Prodpoly-ResNet50 improves the TAR (@FAR=1e-4) by 0.46% and 0.41% on IJB-B and IJB-C, respectively.

Table 14 compares ResNet50 and Prodpoly-ResNet50 under the 1:N end-to-end mixed protocol, which contains both still images and full-motion videos. On IJB-B, there are $10,270$ probe templates containing $60,758$ still images and video frames. On IJB-C, there are $19,593$ probe templates containing $127,152$ still images and video frames. Prodpoly-ResNet50 outperforms ResNet50 by 0.23% and 0.34% on IJB-B and IJB-C rank-1 face identification.
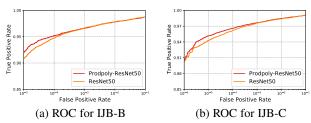

(a) ROC for IJB-B

(b) ROC for IJB-C

Fig. 14: ROC curves of ResNet50 and Prodpoly-ResNet50 under 1:1 verification protocol on the IJB-B and IJB-C dataset.

Results on MegaFace. The MegaFace dataset [83] includes 1M images of 690K different individuals as the gallery set and 100K photos of 530 unique individuals from FaceScrub [84] as the probe set. On MegaFace, there are two testing protocols (e.g., identification and verification). Table 15 show the identification and verification results on MegaFace dataset. In particular, the proposed Prodpoly-ResNet50 achieve 0.50% improvement at the Rank-1@1e6 identification rate and 0.31% improvement at the verification TPR@FAR=1e-6 rate over the baseline ResNet50. In Figure 15, Prodpoly-ResNet50 shows superiority over ResNet50 and forms an upper envelope under both identification and verification scenarios.

5.4 3D Mesh representation learning

Below, we evaluate higher order correlations in graph related tasks. We experiment with 3D deformable meshes of fixed topology [85], i.e., the connectivity of the graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ remains the same and each different shape is defined as a different signal $\boldsymbol{x}$ on the vertices of the graph: $\boldsymbol{x} : \mathcal{V} \to \mathbb{R}^d$. As in the previous experiments, we extend a state-of-the-art operator, namely spiral convolutions
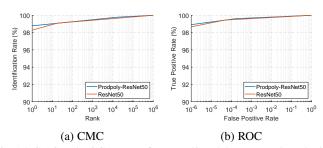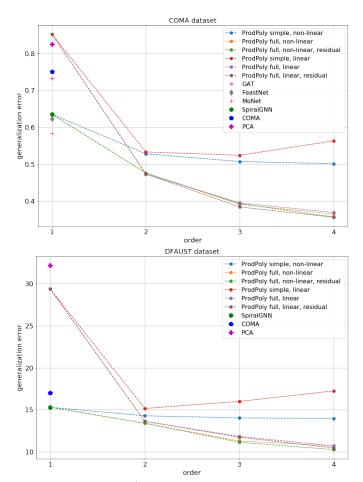

Fig. 16: ProdPoly vs 1st order graph learnable operators for mesh autoencoding. Note that even without using activation functions the proposed methods significantly improve upon the state-of-the-art.

[86], with the ProdPoly formulation and test our method on the task of autoencoding 3D shapes. We use the existing architecture and hyper-parameters of [86], thus showing that ProdPoly can be used as a plug-and-play operator to existing models, turning the aforementioned one into a Spiral Π-Net. Our implementation uses a product of polynomials (referred as *ProdPoly full*), where each layer is a Nth order polynomial instantiated as a specific case of (7) or (8):

NCP: $\boldsymbol{x}_n = \left(\boldsymbol{A}_{[n]}^T \boldsymbol{x}_1 \right) * \left(\boldsymbol{S}_{[n]}^T \boldsymbol{x}_{n-1} \right) + \boldsymbol{A}_{[n]}^T \boldsymbol{x}_1$

NCP-Skip: $\boldsymbol{x}_n = \left(\boldsymbol{A}_{[n]}^T \boldsymbol{x}_1 \right) * \left(\boldsymbol{S}_{[n]}^T \boldsymbol{x}_{n-1} \right) + \boldsymbol{A}_{[n]}^T \boldsymbol{x}_1 + \boldsymbol{x}_{n-1}$, $\boldsymbol{x} = \boldsymbol{x}_N + \boldsymbol{\beta}$, where $\boldsymbol{A}_{[n]}, \boldsymbol{S}_{[n]}$ are spiral convolutions written

TABLE 13: 1:1 **verification TAR** on the IJB-B and IJB-C datasets.

Methods (%)	IJB-B				IJB-C			
	FAR=1e−6	FAR=1e−5	FAR=1e−4	FAR=1e−3	FAR=1e−6	FAR=1e−5	FAR=1e−4	FAR=1e−3
ResNet50	37.28	90.73	94.73	96.63	90.47	94.28	96.17	97.57
Prodpoly-ResNet50	**43.46** (↑ 6.18)	**91.95** (↑ 1.22)	**95.19** (↑ 0.46)	**96.67** (↑ 0.04)	**90.77** (↑ 0.30)	**95.16** (↑ 0.88)	**96.58** (↑ 0.41)	**97.66** (↑ 0.09)

TABLE 14: **1:N (mixed media) Identification** on the IJB-B and IJB-C datasets. False positive identification rate (FPIR) is the proportion of non-mated searches returning any (1 or more) candidates at or above a threshold.

Methods (%)	IJB-B				IJB-C			
	FPIR=0.01	FPIR=0.1	Rank 1	Rank 5	FPIR=0.01	FPIR=0.1	Rank 1	Rank 5
ResNet50	84.70	94.01	95.29	97.14	92.87	95.28	96.52	97.69
Prodpoly-ResNet50	**85.58** (↑ 0.88)	**94.69** (↑ 0.68)	**95.52** (↑ 0.23)	**97.16** (↑ 0.02)	**93.60** (↑ 0.73)	**95.93** (↑ 0.65)	**96.86** (↑ 0.34)	**97.79** (↑ 0.10)

TABLE 15: Face identification and verification evaluation of ResNet50 and the proposed Prodpoly-ResNet50 on MegaFace Challenge1 using FaceScrub as the probe set. "Id" refers to the rank-1 face identification accuracy with 1M distractors, and "Ver" refers to the face verification TAR at 10^{-6} FAR. Results are evaluated on the refined MegaFace dataset [71].

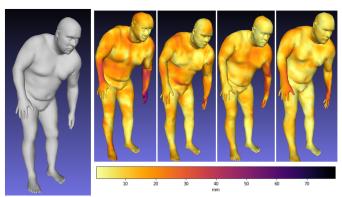| Methods	Id (%)	Ver (%)
ResNet50	98.28	98.64
Prodpoly-ResNet50	**98.78** (↑ 0.50)	**98.95** (↑ 0.31)

Fig. 17: Color coding of the per vertex reconstruction error on an exemplary human body mesh. From left to right: ground truth mesh, 1st order SpiralGNN, 2nd, 3rd and 4th order Spiral ProdPoly.

in matrix form, $\boldsymbol{\beta}$ is a bias vector, $\boldsymbol{x}_1$, $\boldsymbol{x}$ is the input (which is equal to the output of the previous layer) and the output of the layer respectively. Stability of the optimization is ensured by applying vertex-wise instance normalization on the 2nd order term of the recursive formulation.

Additionally, we evaluate our formulation with a simpler model (*ProdPoly simple*) that allows for an attractive trade-off between increased expressivity and constrained parameter budget. In specific, we can create higher-order polynomials without adding new blocks in the original architecture as follows:

$$\boldsymbol{x}_N = \sum_{n=2}^{N} \underbrace{\left(\boldsymbol{S}^T\boldsymbol{x}_1\right) * \left(\boldsymbol{S}^T\boldsymbol{x}_1\right) \cdots \left(\boldsymbol{S}^T\boldsymbol{x}_1\right)}_{\text{n times}} + \boldsymbol{S}^T\boldsymbol{x}_1 + \boldsymbol{\beta}.$$

We use the same normalization scheme as before, by independently normalizing each higher order term. Note that here we only use one learnable operator $\boldsymbol{S}$ (spiral convolution) per layer. It is interesting to notice that this model can be also re-interpreted as a learnable polynomial activation function as in [14], which is a specific case of ProdPoly. As evidence from the experiments suggests, such polynomial activation functions lead to increased expressivity per se, but are less expressive when compared to richer multiplicative interactions as introduced by our NCP and NCP-skip models.

In Fig. 16, we compare the reconstruction error of the proposed method to the baseline spiral convolutions along with other popular graph learnable operators, i.e. the Graph Attention Network (GAT) [87], FeastNet [88], Mixture model CNNs (MoNet) [89], Convolutional Mesh Autoencoders (COMA) [85] which are based on the spectral graph filters of ChebNet [90], as well as with Principal Component Analysis (PCA), which is quite popular in shape analysis applications [91]. The evaluation is performed on two popular 3D deformable shape benchmarks, COMA [85] and DFAUST [92], that depict facial expressions and body poses respectively. Π−nets outperform all published methods even when discarding the activation functions across the entire network. Similar patterns emerge in both datasets: NCP and NCP-Skip behave similarly regardless of the absence of activation functions or not, leading to an increased performance when the order of the polynomial increases, i.e. generalization improves along with

expressivity. Moreover, the simple model provides a boost in performance as well, although we observe a decrease for the 3rd and 4th order of the linear model, which might be attributed to overfitting (similarly to the linear experiments in Sec. 4.3). Thus, we showcase that expressivity improves by seamlessly converting the existing architecture to a polynomial, without having to increase the depth or width of the architecture as frequently done by ML practitioners, and with small sacrifices in terms of inference time and parameter count.

Finally, in Fig. 17 we assess how the order of the polynomial qualitatively reflects in the reconstruction of an exemplary mesh. In particular, we color code the per vertex reconstruction error on the reconstructed meshes (right) and compare them with the input (left). Notice that the overall shape resembles the input more as we increase the order of the polynomial (especially in the head), while body parts with strong articulations (e.g. hands) are reconstructed with higher fidelity.

6 FUTURE DIRECTIONS

The new class of Π−nets has strong experimental results and few empirical theoretical results already. We expect in the following years new works that improve our results and extend our formulation. To that end, we summarize below several fundamental topics that are open for interested practitioners.

A core topic is the theoretical properties of Π−nets. That includes the expressivity of this class of neural networks. The recent study of [10] proves that second-order correlations are beneficial, however the expressivity on higher-order polynomials remains yet to be studied. The generalization of the Π−nets is also crucial. In our evaluation without activation functions, we

noticed that polynomials might be prone to overfitting (e.g., in the classification setting of Sec. 4.3). When we add the non-linear activations we did not observe such a consistent pattern.

Reducing the network redundancy is also an exciting topic. The theoretical properties of multiplicative interactions along with our experiments, exhibit how polynomial neural networks can be used to reduce the network redundancy. Additional post-processing techniques, such as pruning, or exploiting tools from the tensor methods, such as low-rank constraints, might be beneficial in this context.

7 CONCLUSION

In this work, we have introduced a new class of DCNNs, called Π-Nets, that perform function approximation using a polynomial neural network. Our Π-Nets can be efficiently implemented via a special kind of skip connections that lead to high-order polynomials, naturally expressed with tensorial factors. The proposed formulation extends the standard compositional paradigm of overlaying linear operations with activation functions. We motivate our method by a sequence of experiments without activation functions that showcase the expressive power of polynomials, and demonstrate that Π-Nets are effective in both discriminative, as well as generative tasks. Trivially modifying state-of-the-art architectures in image generation, image and audio classification, face verification/identification as well as mesh representation learning, the performance consistently improves.

8 ACKNOWLEDGEMENTS

REFERENCES

[1] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. 1, 2
[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems (NeurIPS)*, 2012, pp. 1097–1105. 1, 2
[3] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4700–4708. 1, 5
[4] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *International Conference on Learning Representations (ICLR)*, 2018. 1, 5, 7
[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. 1, 5, 8
[6] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning (ICML)*, 2015. 1, 2
[7] S. Arora, N. Cohen, N. Golowich, and W. Hu, "A convergence analysis of gradient descent for deep linear neural networks," in *International Conference on Learning Representations (ICLR)*, 2019. 1
[8] K. Ji and Y. Liang, "Minimax estimation of neural net distance," in *Advances in neural information processing systems (NeurIPS)*, 2018, pp. 3845–3854. 1
[9] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 7
[10] 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 (ICLR)*, 2020. 1, 2, 11
[11] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, 2017. 1
[12] G. Chrysos, S. Moschoglou, Y. Panagakis, and S. Zafeiriou, "Polygan: High-order polynomial generators," *arXiv preprint arXiv:1908.06571*, 2019. 1
[13] G. Chrysos, S. Moschoglou, G. Bouritsas, Y. Panagakis, J. Deng, and S. Zafeiriou, "π−nets: Deep polynomial neural networks," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1
[14] J. Kileel, M. Trager, and J. Bruna, "On the expressive power of deep polynomial neural networks," in *Advances in neural information processing systems (NeurIPS)*, 2019. 1, 11
[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248–255. 2
[16] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *International Conference on Computer Vision (ICCV)*, 2015, pp. 3730–3738. 2
[17] S. Tokui, K. Oono, S. Hido, and J. Clayton, "Chainer: a next-generation open source framework for deep learning," in *NeurIPS Workshops*, 2015. 2
[18] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NeurIPS Workshops*, 2017. 2
[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015. 2
[20] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," in *International Conference on Learning Representations (ICLR)*, 2018. 2
[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 2015. 2
[22] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010, pp. 249–256. 2
[23] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," in *International Conference on Learning Representations (ICLR)*, 2014. 2
[24] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016. 2
[25] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980. 2
[26] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017. 2
[27] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814. 2
[28] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," in *International Conference on Learning Representations (ICLR)*, 2019. 2
[29] S. Zhao, J. Song, and S. Ermon, "Learning hierarchical features from deep generative models," in *International Conference on Machine Learning (ICML)*, 2017, pp. 4091–4099. 2
[30] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," in *International Conference on Learning Representations (ICLR)*, 2018. 2, 7
[31] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *International Conference on Computer Vision (ICCV)*, 2017, pp. 1501–1510. 2
[32] A. G. Ivakhnenko, "Polynomial theory of complex systems," *transactions on Systems, Man, and Cybernetics*, no. 4, pp. 364–378, 1971. 2
[33] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015. 2
[34] S.-K. Oh, W. Pedrycz, and B.-J. Park, "Polynomial neural networks architecture: analysis and design," *Computers & Electrical Engineering*, vol. 29, no. 6, pp. 703–725, 2003. 2
[35] Y. Shin and J. Ghosh, "The pi-sigma network: An efficient higher-order neural network for pattern classification and function approximation," in *International Joint Conference on Neural Networks*, vol. 1, 1991, pp. 13–18. 2

[36] Y. Xiong, W. Wu, X. Kang, and C. Zhang, "Training pi-sigma network by online gradient algorithm with penalty for small weight update," *Neural computation*, vol. 19, no. 12, pp. 3356–3368, 2007. 2

[37] C. Voutriaridis, Y. S. Boutalis, and B. G. Mertzios, "Ridge polynomial networks in pattern recognition," in *EURASIP Conference focused on Video/Image Processing and Multimedia Communications*, vol. 2, 2003, pp. 519–524. 2

[38] C.-K. Li, "A sigma-pi-sigma neural network (spsnn)," *Neural Processing Letters*, vol. 17, no. 1, pp. 1–19, 2003. 2

[39] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *International Conference on Learning Representations (ICLR)*, 2015. 2

[40] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015. 2, 5

[41] S. Reed, K. Sohn, Y. Zhang, and H. Lee, "Learning to disentangle factors of variation with manifold interaction," in *International Conference on Machine Learning (ICML)*, 2014, pp. 1431–1439. 2

[42] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009. 2, 3

[43] M. H. Stone, "The generalized weierstrass approximation theorem," *Mathematics Magazine*, vol. 21, no. 5, pp. 237–254, 1948. 2

[44] S. Nikol'skii, *Analysis III: Spaces of Differentiable Functions*, ser. Encyclopaedia of Mathematical Sciences. Springer Berlin Heidelberg, 2013. 2

[45] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *International Conference on Learning Representations (ICLR)*, 2019. 3

[46] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems (NeurIPS)*, 2015, pp. 1135–1143. 3

[47] C. Ding, S. Liao, Y. Wang, Z. Li, N. Liu, Y. Zhuo, C. Wang, X. Qian, Y. Bai, G. Yuan *et al.*, "Circnn: accelerating and compressing deep neural networks using block-circulant weight matrices," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, 2017, pp. 395–408. 3

[48] C. Yunpeng, J. Xiaojie, K. Bingyi, F. Jiashi, and Y. Shuicheng, "Sharing residual units through collective tensor factorization in deep neural networks," in *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2018. 3

[49] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. De Freitas, "Predicting parameters in deep learning," in *Advances in neural information processing systems (NeurIPS)*, 2013, pp. 2148–2156. 3

[50] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017. 5, 6

[51] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose," *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, no. 6, pp. 643–660, 2001. 5, 6, 7

[52] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems (NeurIPS)*, 2014. 5

[53] A. Krizhevsky, V. Nair, and G. Hinton, "The cifar-10 dataset," *online: http://www. cs. toronto. edu/kriz/cifar. html*, vol. 55, 2014. 5, 6, 7

[54] W. Wang, X. Li, J. Yang, and T. Lu, "Mixed link networks," in *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2018. 5

[55] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500. 5

[56] K. Zhang, M. Sun, T. X. Han, X. Yuan, L. Guo, and T. Liu, "Residual networks of residual networks: Multilevel residual networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 6, pp. 1303–1314, 2017. 5

[57] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016. 5

[58] G. L. Grinblat, L. C. Uzal, and P. M. Granitto, "Class-splitting generative adversarial networks," *arXiv preprint arXiv:1709.07359*, 2017. 5, 7

[59] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in neural information processing systems (NeurIPS)*, 2017, pp. 5767–5777. 5, 7

[60] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-100 (canadian institute for advanced research)." [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html 5

[61] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in neural information processing systems (NeurIPS)*, 2016, pp. 2234–2242. 7

[62] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in neural information processing systems (NeurIPS)*, 2017, pp. 6626–6637. 7

[63] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9. 7

[64] Y. Du and I. Mordatch, "Implicit generation and generalization in energy-based models," in *Advances in neural information processing systems (NeurIPS)*, 2019. 7

[65] Y. Hoshen, K. Li, and J. Malik, "Non-adversarial image synthesis with generative latent nearest neighbors," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5811–5819. 7

[66] T. Lucas, K. Shmelkov, K. Alahari, C. Schmid, and J. Verbeek, "Adversarial training of partially invertible variational autoencoders," *arXiv preprint arXiv:1901.01091*, 2019. 7

[67] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018. 8

[68] 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 (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. 8

[69] 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:1706.02677*, 2017. 8

[70] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7132–7141. 8

[71] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4690–4699. 9, 10, 11

[72] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao, "Ms-celeb-1m: A dataset and benchmark for large-scale face recognition," in *European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 87–102. 9

[73] J. Deng, J. Guo, D. Zhang, Y. Deng, X. Lu, and S. Shi, "Lightweight face recognition challenge," in *CVPRW*, 2019, pp. 0–0. 9

[74] J. Deng, J. Guo, Y. Zhou, J. Yu, I. Kotsia, and S. Zafeiriou, "Retinaface: Single-stage dense face localisation in the wild," *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 9

[75] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments," 2008. 9

[76] S. Sengupta, J.-C. Chen, C. Castillo, V. M. Patel, R. Chellappa, and D. W. Jacobs, "Frontal to profile face verification in the wild," in *WACV*, 2016. 9

[77] S. Moschoglou, A. Papaioannou, C. Sagonas, J. Deng, I. Kotsia, and S. Zafeiriou, "Agedb: The first manually collected in-the-wild age database," in *CVPR Workshop*, 2017. 9

[78] T. Zheng and W. Deng, "Cross-pose lfw: A database for studying cross-pose face recognition in unconstrained environments," *Technical Report*, 2018. 9

[79] T. Zheng, W. Deng, and J. Hu, "Cross-age lfw: A database for studying cross-age face recognition in unconstrained environments," *arXiv:1708.08197*, 2017. 9

[80] M. Wang, W. Deng, J. Hu, X. Tao, and Y. Huang, "Racial faces in the wild: Reducing racial bias by information maximization adaptation network," in *ICCV*, 2019. 9

[81] C. Whitelam, E. Taborsky, A. Blanton, B. Maze, J. C. Adams, T. Miller, N. D. Kalka, A. K. Jain, J. A. Duncan, and K. Allen, "Iarpa janus benchmark–b face dataset." in *CVPR Workshop*, 2017. 9

[82] B. Maze, J. Adams, J. A. Duncan, N. Kalka, T. Miller, C. Otto, A. K. Jain, W. T. Niggel, J. Anderson, and J. Cheney, "Iarpa janus benchmark–c: Face dataset and protocol," in *ICB*, 2018. 9

[83] I. Kemelmacher-Shlizerman, S. M. Seitz, D. Miller, and E. Brossard, "The megaface benchmark: 1 million faces for recognition at scale," in *CVPR*, 2016. 9, 10

[84] H.-W. Ng and S. Winkler, "A data-driven approach to cleaning large face datasets," in *ICIP*, 2014. 10

[85] A. Ranjan, T. Bolkart, S. Sanyal, and M. J. Black, "Generating 3d faces using convolutional mesh autoencoders," in *European Conference on Computer Vision (ECCV)*, 2018, pp. 704–720. 10, 11

[86] G. Bouritsas, S. Bokhnyak, S. Ploumpis, M. Bronstein, and S. Zafeiriou, "Neural 3d morphable models: Spiral convolutional networks for 3d shape representation learning and generation," in *International Conference on Computer Vision (ICCV)*, 2019. 10

[87] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *International Conference on Learning Representations (ICLR)*, 2018. 11

[88] N. Verma, E. Boyer, and J. Verbeek, "Feastnet: Feature-steered graph convolutions for 3d shape analysis," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 11

[89] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 11

[90] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems (NeurIPS)*, 2016. 11

[91] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3d faces," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, 1999. 11

[92] F. Bogo, J. Romero, G. Pons-Moll, and M. J. Black, "Dynamic faust: Registering human bodies in motion," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 11

Jiankang Deng is a Ph.D. candidate 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. He is in the project of EPSRC FACER2VM (Face Matching for Automatic Identity Retrieval, Recognition, Verification and Management). His Ph.D. research topic is face analysis (face detection, face alignment, face recognition and face generation). During his PhD studies, he has organised the Menpo 2D Challenge (CVPR 2017), the Menpo 3D Challenge (ICCV 2017) and Lightweight Face Recognition Challenge (ICCV 2019). He also won many academic challenges, such as ILSVRC Object Detection and Tracking 2017, Activity-Net Untrimmed Video Classification 2017, iQIYI Celebrity Video Identification Challenge 2018, Disguised Face Recognition Challenge 2019. 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.

Grigorios G. Chrysos is a fourth year PhD student in Imperial College London working with Stefanos Zafeiriou. Previously, he graduated from National Technical University of Athens (2014). He has published his work on deformable models in prestigious journals (T-PAMI, IJCV, T-IP), while he has co-organised workshops for deformable models, e.g. 2D/3D facial landmark tracking, in CVPR/ICCV. He is a reviewer in prestigious journals including T-PAMI, IJCV. Currently, his primary research interest is on machine learning, including generative models, tensor decompositions and modelling high dimensional distributions; his recent work has been published in top tier conferences (CVPR, ICLR).

Stylianos Moschoglou received his Diploma/MEng in Electrical and Computer Engineering from Aristotle University of Thessaloniki, Greece, in 2014. In 2015-16, he pursued an MSc in Computing (specialisation Artificial Intelligence) at Imperial College London, U.K., where he completed his project under the supervision of Dr. Stefanos Zafeiriou. He is currently a PhD student at the Department of Computing, Imperial College London, under the supervision of Dr. Stefanos Zafeiriou. His interests lie within the area of Machine Learning and in particular in Generative Adversarial Networks and Component Analysis.

Giorgos Bouritsas is a 2nd year PhD student in Imperial College London working with Prof. Michael Bronstein and Prof. Stefanos Zafeiriou. Giorgos graduated from National Technical University of Athens (NTUA) with an MEng Diploma in Electrical and Computer Engineering in 2017. Before starting his PhD he spent time as a visiting researcher at the Universitat Politècnica de Catalunya (UPC), Barcelona, and as a research associate at the National Center for Scientific Research "Demokritos", Athens conducting research on computer vision and machine learning. His current research interests lie within the field of on non-Euclidean deep learning, graph neural networks, graph signal processing & network science, and he is particularly focused on the expressivity of graph neural networks and generative models for non-Euclidean data.

Yannis Panagakis is an Associate Professor of machine learning and signal processing at the University of Athens. His research interests lie in machine learning and its interface with signal processing, high-dimensional statistics, and computational optimization. Specifically, Yannis is working on models and algorithms for robust and efficient learning from high-dimensional data and signals representing audio, visual, affective, and social information. He has been awarded the prestigious Marie-Curie Fellowship, among various scholarships and awards for his studies and research. He co-organized the BMVC 2017 conference and several workshops and special sessions in top venues such as ICCV. He received his PhD and MSc degrees from the Department of Informatics, Aristotle University of Thessaloniki and his BSc degree in Informatics and Telecommunication from the University of Athens, Greece.

Stefanos Zafeiriou (M'09) is a Reader 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 10000 citations to his work, h-index 51.