Factorized Dynamic Fully-Connected Layers for Neural Networks

Francesca Babiloni[1,2], Thomas Tanay[1], Jiankang Deng[1,2], Matteo Maggioni[1], Stefanos Zafeiriou[2]
[1]Huawei, Noah's Ark Lab [2]Imperial College London
{f.babiloni22, j.deng16, s.zafeiriou}@imperial.ac.uk {thomas.tanay, matteo.maggioni}@huawei.com

Abstract

The design of neural network layers plays a crucial role in determining the efficiency and performance of various computer vision tasks. However, most existing layers compromise between fast feature extraction and reasoning abilities, resulting in suboptimal outcomes. In this paper, we propose a novel and efficient operator for representation learning that can dynamically adjust to the underlying data structure. We introduce a general Dynamic Fully-Connected (DFC) layer, a non-linear extension of a Fully-Connected layer that has a learnable receptive field, is instance-adaptive, and spatially aware. We propose to use CP decomposition to reduce the complexity of the DFC layer without compromising its expressivity. Then, we leverage Summed Area Tables and Modulation to create an adaptive receptive field that can process the input with constant complexity. We evaluate the effectiveness of our method on image classification and other downstream vision tasks using both hierarchical and isotropic architectures. Our results demonstrate that our method outperforms other commonly used layers by a significant margin while keeping a fixed computational budget, therefore establishing a new strategy to efficiently design neural architectures that can capture the multi-scale features of the input without increasing complexity.

1. Introduction

Deep learning models have achieved remarkable results on various computer vision (CV) tasks, such as image classification, object detection, and instance segmentation [33, 9, 46]. However, these models often require a large number of parameters and computations, which limits their applicability in real-world scenarios. Moreover, existing models often rely on fixed or predefined layer types, such as convolution or fully-connected layers, that do not adapt to the input data structure, resulting in suboptimal feature extraction and reasoning. Thus, there is still a need for designing efficient and adaptive neural network layers that can capture rich features of the input without increasing complexity and can serve as basic blocks for different CV applications.

In this paper, we address this problem by proposing a new Dynamic Fully-Connected (DFC) layer, a non-linear extension of a Fully-Connected layer that has an adaptive receptive field, is spatially aware, and instance-adaptive. The DFC layer generates its weights dynamically as a function of the input, allowing it to adjust its receptive field size and shape according to the input data structure. This makes the DFC layer more expressive and flexible than traditional layer types, such as convolution or fully-connected layers, that have fixed or predefined receptive fields. Moreover, the DFC layer is spatially adaptive, meaning that it does not share weights across spatial positions, allowing it to capture local variations and dependencies in the input. The DFC layer is also instance-adaptive, meaning that it processes each input differently, allowing it to handle diverse and heterogeneous data. However, the DFC layer has a high complexity due to its dense weight tensor, which makes it impractical to use in neural network architectures. To overcome this limitation and reduce the complexity of the DFC layer without compromising its expressivity, we leverage CANDECOMP/PARAFAC (CP) decomposition [22]. CP decomposition is a technique for low-rank approximation of multi-dimensional arrays that can be used to reparameterize neural network layers in order to speed up their inference [37, 14]. In this paper, we use CP decomposition to factorize the weight tensor of the DFC layer into a product of low-rank matrices, which reduces the number of parameters and computations required by the DFC layer. Leveraging Summed Area Tables [6, 44], we also propose some modifications to the CP decomposition to make it more efficient and suitable for cases where the spatial size of the data is large or unknown.We demonstrate the effectiveness and generality of our method on the ImageNet dataset and downstream vision tasks, achieving high-quality results with fewer parameters and computations than traditional methods. We also investigate how our factorized DFC performs under different resource constraints, such as a fixed (predefined) computational budget, and analyze the trade-off between its efficiency and accuracy. In summary, our approach introduces a novel operator for representation learning that dynamically adjusts to the input, enabling the

development of computer vision models that are both efficient and high-performing. This approach has the potential to serve as a backbone for a wide range of computer vision applications that require high-level feature extraction and reasoning. The main contributions of this paper are:

- We propose a new Dynamic Fully-Connected (DFC) layer, a non-linear extension of a Fully-Connected layer that has an adaptive receptive field, is instance-adaptive, and spatially aware.
- We show how to use CP decomposition to reduce the complexity of the DFC layer without compromising its expressivity, and how to modify it for cases where the spatial size of the data is large or unknown. We use the CP-Decomposed DFC layer as a basic block to build neural network architectures for image classification, object detection, and segmentation.
- We demonstrate the effectiveness and generality of our method on the ImageNet dataset and two downstream vision tasks under different resource constraints, such as computational memory and different architectural macro-designs, outperforming static traditional layers by a large margin under the same parameters and computational requirements.

2. Related Work

Reducing Complexity of Neural Networks. In this work, we focus on the development of efficient neural network design. Related but orthogonal research focuses on improving inference time of *pretrained* Deep Learning models. Such techniques reduce or optimize memory requirement, energy consumption, and number of operations of a given network without significantly decreasing its accuracy. Interesting research trends in this area include: parameter quantization [11], network pruning [29], network architecture search [10], and knowledge distillation [12].

Tensor Decomposition for Neural Networks. Tensor Decomposition is an active area of research dedicated to the study of low-rank approximation for multi-dimensional arrays and has applications in a variety of fields, ranging from psychology to computer vision [22, 37]. Tensor decomposition techniques have been used to reparameterize neural network layers in order to speed up their inference [5, 4, 34].[25] and [36] used CP Decomposition to speed up spatial static convolutional and FC layers. [23] extended this idea to spatio-temporal static convolutional kernels. Differently from these works, we focus on non-linear dynamic layers and extend this trend of research by investigating a tensor decomposition for a "*Dynamic* Fully-Connected" layer.

Tensor Notation for Neural Networks. Einstein notation provides an intuitive notation for tensor manipulations. In machine learning, it can be used as an alternative to tensor algebra [37, 14]. Recently, Einstein notation has gained traction as a practical way to improve code readability [41, 40] and enable efficient tensor calculus [24]. Here we use the Einstein notation as a way to compare building blocks for neural networks.

Dynamic Neural Network Layers. The idea of using a layer whose weights are adaptive to the input can be traced back to early CNNs using max-pooling [19]. Dynamic convolutions emerged multiple times in the context of low-level [20, 35, 48] as well as high-level vision [13, 3, 47, 27]. The dynamic component is also a neglected feature of attention mechanisms [43, 17, 1, 21], and we identify it here as the key to unlocking non-linear behavior.

Summed Area Tables for Neural Networks. Summed Area Tables (SAT) is an established algorithm in computer vision [6, 44] that is able to provide the sum of values within an arbitrary subset of a grid in constant time. Recently, SAT has been used to accelerate large-kernel convolution in a dense prediction network for Human Pose Estimation [50] and dynamic large-kernel convolutions in language tasks [32]. In the context of Neural network design, SAT enabled fast computation of a linearized attention variant [51], a parameter-free method to adapt the size of the area to attend [28]. Related work on Pooling based neural networks [49] shows how the summation operation can be used to build a competitive backbone for CV. Here, we leverage SAT to achieve an efficient CP Decomposition for DFC.

3. Einstein Notation for Neural Networks

Neural networks – and deep learning architectures in particular – are commonly built as a sequence of tensor operations interleaved by point-wise non-linearities. These critical components extract various meaningful information from the input, with different operators often presented in the literature as new "building blocks" or "layers" for neural architectures, such as "MLP", "Convolution", "Residual Block", "Dense Block", "Deformable Conv", "Attention", "Dynamic-Conv", etc. In this section, we present a general form of a neural network layer and showcase how the Einstein summation convention can be used as an alternative, short-hand, and self-contained way to represent and relate building blocks for neural networks.

3.1. Background

Einstein notation. In the rest of the paper, we adopt the notation of [24]. Tensors are denoted with uppercase letters and indices to the dimensions of the tensors are denoted in lowercase subscripts. For instance $X_{ijk} \in \mathbb{R}^{I \times J \times K}$ is a three-dimensional tensor of size $I \times J \times K$ with three modes (or dimensions) indexed by $i \in [1, I]$, $j \in [1, J]$, and $k \in [1, K]$. Using the Einstein nota-

tion, any multiplication among tensors can be written as: $C_{s_3} = \sum_{(s_1 \cup s_2) \setminus s_3} A_{s_1} B_{s_2}$ where s_1, s_2, and s_3 are the index sets of the left argument, the right argument, and the result tensor, respectively. The summation is only relevant for inner products and is made explicit by underlining tensor indexes. As a representative example to illustrate our notation, we review a set of common operations among tensors. Given the tensors of order two $Y, X \in \mathbb{R}^{I \times J}$, their Hadamard product can be written as $Z_{ij} = X_{ij} Y_{ij}$ and is equivalent to the algebraic notation $Z = X \odot Y$. Similarly, their matrix-product can be written as $Z_{ij} = X_{i\underline{k}} Y_{\underline{k}j}$ and is equivalent to the algebraic notation $Z = XY^\top$. Given the tensors of order one $Y \in \mathbb{R}^I$ and $X \in \mathbb{R}^J$, their outer product creates a tensor $Z \in \mathbb{R}^{I \times J}$ as $Z_{ij} = X_i Y_j$. It is equivalent to the algebraic expression $Z = Y^\top X$. When using a chained sequence of operations, we use the $\rfloor$ symbol to write each intermediate result.

CP Decomposition. The CP Decomposition also referred to as CANDECOMP/PARAFAC or polyadic decomposition, is used to express the factorization of a multi-dimensional tensor as a linear combination of components with rank one, and thus generalizes the concept of matrix singular value decomposition (SVD) to tensors [22]. For example, let $X_{ijk} \in \mathbb{R}^{I \times J \times K}$ be a three-dimensional tensor, then we can define the CP Decomposition $X_{ijk} \approx U^1_{i\underline{r}} U^2_{j\underline{r}} U^3_{k\underline{r}}$ as the approximation of the original tensor from a set of three-factor matrices $[U^1_{ar}, U^2_{br}, U^3_{cr}]$. The rank of the tensor X_{ijk} is defined as the smallest number of R components needed to generate equality in the CP Decomposition. Note that we call a CP Decomposition canonical whenever R is equal to the rank of X_{ijk}.

3.2. The Dynamic Fully-Connected Layer

A *neural network layer* is a function f that takes as input a tensor X_{mc} composed of $m \in [1, M]$ spatial positions with $c \in [1, C]$ features (or channels) and produces as output a tensor Y_{nd} composed of $n \in [1, N]$ output spatial positions with $d \in [1, D]$ channels:

$$Y_{nd} = f(X_{mc}) \tag{1}$$

In the following, we start by considering the special case where f is a linear function before introducing the more general dynamic fully-connected layer.

Linear Layers allocate a set of learnable parameters that are allowed to change during training. At inference time, weights are static or fixed (i.e. independent from the input data). These operators compute the output as a linear combination of the elements of the input tensor. The most general instantiation of a *linear* neural network layer is the Fully-Connected layer (FC) [42]:

$$Y_{nd} = X_{\underline{mc}} W_{\underline{mn}\underline{c}d}, \tag{2}$$

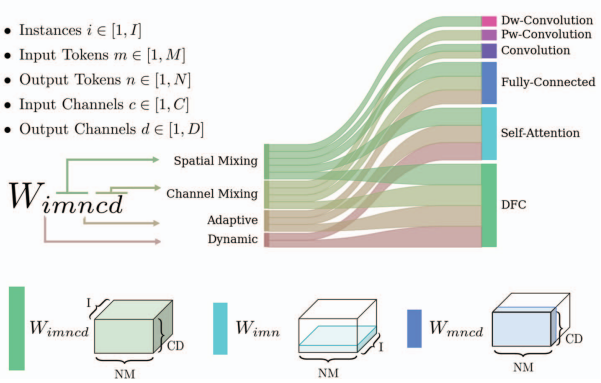


Figure 1: **Overview of Building Blocks Characteristics.** The tensor W_{imncd} is a representation of a general neural network layer and each of its dimensions is associated with one characteristic that can be used to describe existing building blocks. For example, a Fully-Connected layer has Spatial and Channel mixing, but is not Dynamic. DFC incorporates all possible characteristics in its formulation. Incorporating additional characteristics in a layer has the side effect of increasing the dimensions of the underlying parameter tensors.

parametrized by a four-dimensional weight tensor W_{mncd} mixing all the available information of the input. FC is *spatially adaptive*, i.e. weights are not shared across spatial positions and have a complexity of $\mathcal{O}(M \cdot N \cdot C \cdot D)$. Convolutions [26], Point-wise (Pw) and Depth-wise (Dw) Convolutions [16] are special cases of FC that use the priors on weight sharing and local processing to reduce complexity.

Dynamic Layers for neural networks adapt their response to the input at inference time. Their output depends on a static tensor of learnable parameters as well as a set of "instance-adaptive" weights, usually created via the use of a function $g(X)$ of the input tensor. These layers compute the output as a non-linear combination of the elements of the input tensor. The Dynamic Fully-Connected layer (Dynamic-FC or DFC) is a non-linear generalization of the FC layer that can be obtained by turning the weight tensor into a function g of the input: $W_{mncd} = g(X_{mc})$. To illustrate that the tensor W_{mncd} is not constant anymore but the result of a dynamic construction mechanism, we now consider a "batch" of input instances X_{imc} created as a stack of I inputs $[X^1_{mc}, X^2_{mc}, ..., X^I_{mc}]$ and the corresponding batch of output instances Y_{ind} both indexed by the new instance dimension $i \in [1, I]$:

$$\begin{cases} Y_{ind} = X_{i\underline{mc}} W_{i\underline{mn}\underline{c}d} \\ W_{imncd} = [g(X^1_{mc}), \ldots, g(X^I_{mc})]. \end{cases} \tag{3}$$

As in the FC layer, the DFC generates the output by mixing all spatial and channel information. On top of that, DFC is *dynamic* or *instance-adaptive*, i.e. it processes each input differently. Note the relationship between FC and

DFC: i) every instance of FC is also an instance of DFC (i.e. DFC where function g is constant) and ii) there are instances of DFC that are not FC (i.e. DFC where function g is non-constant). Therefore FC is a special case of DFC. As visible in Figure 1, DFC represents a general way to leverage the complete range of interactions of the input and serves as a generalization of existing layers. However, the usage of the DFC is severely limited in practice because of the dense structure of its weights and its high complexity, which is equal to the complexity of the FC layer plus the complexity of the function g. Interestingly, a number of well-known neural network layers can be framed as simplified cases of DFC. That this is the case for Self-Attention [43, 45], Dynamic Convolution [47, 17], Deformable Convolution [7, 52].

4. Factorized Dynamic Fully-Connected Layer

The use of a parameters tensor W_{imncd} makes the DFC block general, but also its computation heavy to the point of being unattainable in practice. Next, we propose to use CP Decomposition as a means to decrease DFC complexity. Specifically, we propose to factorize the weights of the DFC layer through its CP Decomposition:

$$W_{imncd} = U^1_{i\underline{r}} U^2_{m\underline{r}} U^3_{n\underline{r}} U^4_{c\underline{r}} U^5_{d\underline{r}} + \epsilon_{imncd} \tag{4}$$

which represents the full tensor W_{imncd} as a linear combination of lower-dimensional factor matrices plus an approximation error ϵ_{imncd} dependent on the choice of R. Typically, lower R implies larger errors, while for $R \geq rank(W_{imncd})$ the error is zero, and the CP Decomposition is exact. This approach makes it possible to use these layers efficiently in a neural network without using any strong prior on their weights. Finally, we can define the DFC CP Decomposition by replacing the DFC weights of Eq. (3) with those factorized in Eq. (4) and rearranging terms as:

$$Y_{ind} = X_{imn\underline{c}} U^4_{\underline{c}r} \Big]_{im\underline{n}nr} U^2_{\underline{m}r} \Big]_{inr} U^3_{nr} \Big]_{inr} U^1_{ir} \Big]_{in\underline{r}} U^5_{d\underline{r}} \tag{5}$$

where X_{imnc} is the result of unfolding the input X_{imc} with a global receptive field of size M for all N output positions[1]. Equation (5) acts as a low-complexity substitute of (3) .

Replacing DFC layers with the CP Decomposition of (5) reduces drastically the memory needed to store the weights from $\mathcal{O}(M \cdot N \cdot C \cdot D)$ to $\mathcal{O}(L \cdot R)$, $L = max(M,N,C,D)$. It also reduces its computational complexity as the sum of its sequence of operations: $\mathcal{O}(M \cdot C \cdot R)$ + $\mathcal{O}(M \cdot N \cdot R)$ + $\mathcal{O}(N \cdot R)$ + $\mathcal{O}(R)$ + $\mathcal{O}(N \cdot D \cdot R)$, plus the complexity of the function used to create matrix U^1_{ir}. Moreover, it can be easily learned end-to-end as a sequence of fully differentiable building blocks for neural networks.

[1]We define as "unfolding with receptive field K" the operation of rearranging the input as a collection of N sliding patches of size K.

However, in cases where the spatial size is large compared to the number of channels (i.e. $L = M$ or $L = N$), the factor matrices U^2_{mr} and U^3_{nr} cannot be implemented as matrices of learnable parameters without acting as computation bottlenecks. In fact, they require a number of parameters directly proportional to the number of input and output spatial positions. For these reasons, we propose a set of modifications to Eq. (5) that allow us to approximate DFC behaviour in cases where the spatial size of the data is not known in advance and has to fit a low memory footprint. First, we make the complexity of the spatial operator U^2_{mr} independent from the number of input spatial position M by leveraging Summed-Area Table (SAT). SAT, also known as an integral image, is a data structure that can be used to perform fast image filtering [6, 44]. It enables the computation of pooling operations on a receptive field of arbitrary size with a constant computational cost and can be used to implement a pooling operation on a *learned* receptive field [50]. In practice, we propose to decompose the contribution of the factor matrix U^2_{mr} as follows:

$$U^2_{mr} = P_{mr\underline{g}} E_{r\underline{g}} \tag{6}$$

where P_{mrg} is a collection of G fully differentiable pooling layers with a learnable receptive field, and E_{rg} is the set of learnable weights used to combine their contribution. The advantages are two-fold: i) the model is able to actively learn the optimal receptive field, opting for global or local reasoning for the task at hand, and ii) spatial mixing can be performed at a constant computational cost even when the receptive field is global.
Second, we avoid the need to instantiate a parameter for every n-th output spatial position via the use of hypernetwork.

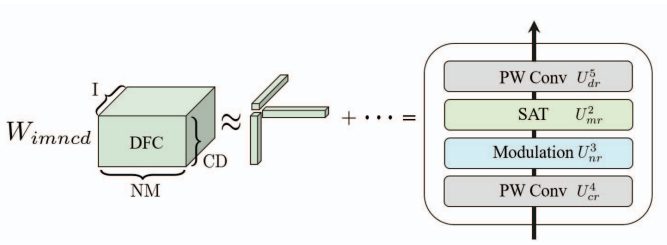


Figure 2: **Overview of our method.** We consider $M = N, C = D, R << C, M > C$. The DFC block uses a W_{imncd} tensor and has, therefore, a complexity that scales quadratically with both the number of spatial positions and the number of channels $\mathcal{O}(N^2 \cdot C^2)$. On the contrary, our Factorized DFC layer approximates the DFC as a sequence of smaller blocks, achieving linear complexity: $\mathcal{O}(N \cdot C)$. This is achievable thanks to the use of SAT, which mixes all spatial elements with a complexity that is independent of the size of the input image.

```python
# Factorized DFC Eq.8
def gate(x):
  x_downscale = downscale(x)
  mask = pw_conv(x_downscale, in_ch=R, out_ch=R//4)
  mask = gelu(mask)
  mask = pw_conv(mask, in_ch=R//4, out_ch=R)
  mask = mask.broadcast(inr)
  return x * upscale(mask) + x

def dfc_decomposition(x, C, D, R, N, G):
  I,N,C = x.shape                              # X_{inc}
  x_4   = pw_conv(x, in_ch=C, out_ch=R)        # U^{4}_{cr}
  x_13  = gate(x_4)                            # U^{13}_{inr}
  x_2   = dynamic_pool(x_13, kernel=N, groups=G) # U^{2}_{mr}
  x_5   = pw_conv(x_2, in_ch=R, out_ch=D)      # U^{5}_{dr}
  return gelu(x_5) + x
```

Listing 1: Pseudo-code for our Factorized DFC Layer. N refers to the number of spatial positions, R the dimension of CP decomposition, C input channels, D output channels . Our block also takes in input the number G of dynamic pooling to learn, and achieves global spatial-mixing by using the `gate` modulation function followed by the `dynamic_pool` function which is in practice implemented with sum-area tables (SAT).

Specifically, we propose to combine the effect of the two gating modules U^1_{ir} and U^3_{nr} into a single operator as

$$U^{13}_{inr} = U^1_{ir} U^3_{nr} = \phi(X_{imr}), \tag{7}$$

where the function ϕ, parametrized via a small CNN, generates dynamic and spatially adaptive weights conditioned on the input X_{imc}. Note that, to further limit complexity, the input of ϕ can be downsampled to a pre-defined fixed size, and then the output can be upsampled to match the original resolution, e.g., by interpolation[2]. As a result, the complexity of this spatial adaptive operation is again constant with respect to the number of output spatial positions N.

We obtain the final formulation by replacing Eq. (6) and Eq. (7) in Eq. (5) as:

$$Y_{ind} = X_{imn\underline{c}}\, U^4_{\underline{c}r}\Big]_{im\underline{n}r} P_{\underline{m}rg}\Big]_{inr\underline{g}} E_{r\underline{g}}\Big]_{inr} U^{13}_{inr}\Big]_{in\underline{r}} U^5_{d\underline{r}} \tag{8}$$

This formulation, which is a CP Decomposition for a DFC block with two extra assumptions on its factor matrices, comes with several desirable properties. Firstly, Eq. (8) can be applied to inputs of arbitrary resolutions without compromises on the size of the receptive field. Secondly, approximate the Eq. (3) behaviour reducing its complexity from $\mathcal{O}(M \cdot N \cdot C \cdot D)$ to $\approx \mathcal{O}(N \cdot C \cdot R)$ (where, potentially, $R << C$). Moreover, its memory footprint is drastically reduced when compared with Eq. (5). Eq. (8) achieves global reasoning with a complexity *and parameter count* independent of the number of spatial positions, cutting the allocated number of learnable parameters from $\mathcal{O}(M+N)$ of Eq. (5) to a constant $\mathcal{O}(1)$. Figure 2 shows an overview of our method and the Listing 1 presents pseudo-code to describe its implementation. In our experiments, we add a residual connection after our block to ease convergence. In our experimental sections, to ease comparisons we use a DFC weight tensor W_{inmcd} with the same number of input and output channels $C = D$ and implement the function ϕ as a stack of two point-wise convolutions. We set the rank of the CP decomposition (R) to be fixed to a quarter of the original dimensionality, and the G number of fully differentiable pooling groups to be 12, which empirically showed a good trade off between complexity and performance.

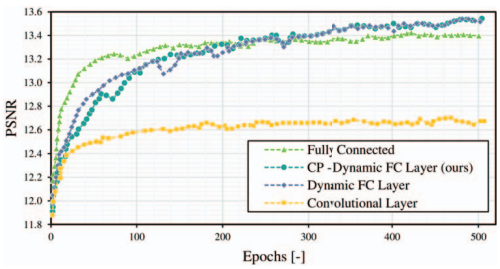
[2]The resizing functions are assumed to be absorbed into ϕ for the sake of notation simplicity.

4.1. Illustrative Example

One of the goals of a Dynamic Fully-Connected Layer is to equip a model with the capability to reason about the whole input representation at one glance, discovering non-linear interactions among input elements. We first test the ability of our Factorized DFC variant to approximate the full DFC weight tensor W_{imncd} via CP decomposition in a controlled scenario. We used the MNIST dataset and a four-layer fully convolutional encoder-decoder architecture. To test the ability of our method to make use of available but scattered information, we attempted the reconstruction of an image given its shuffled version, by designing a "puzzle reconstruction on MNIST" experiment. To obtain an input puzzle, each image is split into 16 tiles of equal size. These tiles are then randomly rotated and mirrored before being stitched back together. Input and ground truth (GT) samples can be seen in Figure 4. Between the encoder and the decoder part of the network, the DFC module integrates non-linear global information about the input tensor. We start by analyzing the effect of the two DFC layers of Eq. (3) and (8). To highlight the effect of global reasoning in the latent space, we use as a baseline a standard Convolution (Conv) layer, which has a complexity similar to our method. To highlight the benefit of non-linear processing in the latent space, we use as a baseline a Fully-Connected layer (FC), which uses a static set of weights to process the input globally but linearly. Figures 3 and 4 show an overview of our comparisons. The Conv baseline is limited to processing the input locally and performs worse than models trained with global reasoning. The FC layer cannot adapt its behavior to the input data and thus has lower performance than the DFC variants. As visible, our CP-Decomposed DFC can approximate global non-linear behavior and is qualitatively on par with the DFC layer. Nevertheless, while a standard DFC has a complexity $\mathcal{O}(N^2 \cdot C^2)$, our version achieves the same results with a linear $\mathcal{O}(N \cdot C \cdot R)$ complexity.

5. Experiments

In the previous sections, we have provided a new perspective on the formulation of neural network layers showcasing how the CP decomposition can be used to reduce the complexity of a DFC layer while keeping its inductive biases of i) adaptive receptive field ii) dependence on the input



Layer	Weights Form	$\mathcal{O}()$	Adaptivity i, n	Receptive Field
Dynamic FC	W_{inmcd}	$\mathcal{O}(N^2 \cdot C^2)$	i, n	Global
Fully Connected	W_{mncd}	$\mathcal{O}(N^2 \cdot C^2)$	n	Global
Convolutional	W_{kcd}	$\mathcal{O}(N \cdot K \cdot C^2)$	-	Local
Dynamic FC - CP	$\approx W_{inmcd}$	$\mathcal{O}(N \cdot C \cdot R)$	i, n	Adaptive

Overview of different Layers

Figure 3: **Puzzle Reconstruction on MNIST. Comparisons among Layers of Neural Networks (left)**. PSNR validation curves show how our CP-DFC method is capable to approximate a general DFC layer performance. **Overview of Layers (right)**. Methods are described by complexity and flagged with an i if dynamic, and with an n if spatially-adaptive. M, N indicates input and output spatial sizes, C, D input and output channels, K convolutional kernel size, R the CP decomposition size. We assume: $M = N$, $C = D$, $K < N$, $R < C$.

Input | **Convolutional** W_{kcd} | **FC Layer** W_{nmcd} | **DFC-CP Layer** $\approx W_{inmcd}$ | **DFC Layer** W_{inmcd} | **GT**

Figure 4: **Qualitative Comparison on MNIST dataset.** Augmenting the size of the weights tensor boosts output quality. Our DFC variant (DFC-CP) uses an adaptive (potentially global) receptive field and dynamic weights prediction. It is able to generate clean outputs and sharp digits and performs on par with the computationally expensive DFC layer, while The use CP decomposition makes its complexity comparable to a Convolutional layer .

and iii) spatial adaptivity. In this section, we investigate the use of our DFC layer as a basic building block for neural network architectures applied to computer vision problems. Further, in our analysis, we emphasize which aspects of the DFC inductive biases are relevant to process vision data.

5.1. Large Scale Classification

We evaluate our framework on the image classification task using the ImageNet dataset [8], a large-scale and diverse benchmark that requires high-level semantic understanding and robustness to variations. We hypothesize that our framework can achieve comparable or better performance than existing methods with fewer parameters and computations, thanks to the joint use of CP decomposition and DFC inductive biases. To test this hypothesis, we contrast the performance of CP-Decomposed DFC layers with CP-Decomposed Convolutional [25] ($\approx W_{kncd}$) and Fully-Connected [36] ($\approx W_{mncd}$) layers. Moreover, we highlight the contribution of SAT by providing a comparison with a CP-Decomposed Convolution variant that uses Pooling ($\approx P_{kncd}$). Note that differently from the small-scale experiment presented in the previous section, a comparison with a standard DFC layer on real computer vision applications is not feasible, since its high computational complexity prevents the network to fit even a standard desktop GPU memory. Overall, these are natural baselines to compare with, as they represent different trade-offs between expressiveness and efficiency in neural network design. Our goal is to demonstrate how our framework can simplify neural network architectures without extensive hyperparameter tuning or architectural modifications. To this end, we keep the same architecture and training setup across all experiments, allowing us to isolate the effect of different layer types on the final outcome. Finally, we investigate two macro-design choices: the hierarchical aspect and the size of the architecture, which influence the capacity and scalability of neural networks. We aim to show how our framework can adapt to different settings with minimal changes. We follow [49] for training experimental setup and hyperparameters choices. We used the ImageNet 1K datasets as input images of size 224×224. We train each model under the same setup using 300 epochs, AdamW optimizer, cosine schedule, and learning rate of $1e^{-3}$. We used the standard data augmentations strategies of CutMix, MixUP, CutOut, and RandAugment. For all our experiments, we used 8 GPUs and an effective batch size of 1024. We refer to the original paper for more in-depth details. As performance metrics, we report results for the original ImageNet test set as well as two additional test sets as a measure of overfitting: the cleaned-up ReaL validation set [2] (Real) and ImageNet-V2 [39] (V2). As a measure of space complexity, we report parameter count in millions P(M), while for time complexity we report Giga Floating Point operations per second F(G).

Layer		Complexity		Classification			
Type	Weights	P(M)	F(G)	T1	T5	v2	Real
DFC-CP	$\approx W_{imncd}$	15	2.3	**80.9**	**95.5**	**69.6**	**86.6**
		28	4.5	**82.0**	**95.6**	**70.6**	**86.7**
FC -CP	$\approx W_{mncd}$	15	2.4	78.5	93.2	66.5	85.0
		28	4.5	80.7	95.2	69.1	86.0
Conv-CP	$\approx W_{kcd}$	15	2.4	78.9	94.4	67.6	85.2
		28	4.5	80.9	95.1	69.2	86.0
Conv-CP	$\approx P_{kcd}$	15	2.4	78.5	94.0	67.0	84.8
		28	4.5	80.6	95.0	68.8	85.8

Table 1: **Performance comparison on Hierarchical Architecture**, where a multi-scale representation of the input is obtained with subsequent downsampling in 4 stages. DFC outperforms competitors.

Layer		Complexity		Classification			
Type	Weights	P(M)	F(G)	T1	T5	v2	Real
DFC-CP	$\approx W_{imncd}$	5	1.0	**71.9**	**90.9**	**59.7**	**79.9**
		20	3.8	**79.7**	**95.0**	**68.8**	**86.0**
FC -CP	$\approx W_{mncd}$	5	1.0	68.8	89.2	56.6	76.9
		20	3.8	77.2	93.5	64.8	83.6
Conv-CP	$\approx W_{kcd}$	5	1.0	69.4	89.5	56.7	77.5
		20	3.8	77.0	93.3	65.4	83.7
Conv-CP	$\approx P_{kcd}$	5	1.0	63.5	85.6	50.6	71.3
		20	3.8	73.0	91.1	59.3	80.1

Table 2: **Performance comparison on Isotropic Architecture**, where no downsampling layer is used. P(M) millions of parameters. F(G) Giga FLOPS . DFC shows consistent performance .

Architecture	C,D	R	T1	T5
Iso-DFC	768	384	77.7	93.8
Iso-DFC	768	192	71.9	90.9
Iso-DFC	768	96	62.6	83.7
Iso-DFC	768	48	47.5	73.5

Table 3: **Ablation on R (the CP decomposition rank)** . We use a small Isotropic net . Lowering R yields higher approximation error and lower performance .

Hierarchical Architecture First, we focus on computer vision networks with a hierarchical structure (that is with a multi-scale representation of the input obtained with subsequent downsampling stages). Specifically, we use a 4-stage network composed by 2:2:6:2 layers with 64 initial channels. After each stage, the network uses a factor x2 of spatial reduction with average pooling and x2 channels increment. We build each stage as a stack of layers and GeLU non-linearities. We made this macro-design choice with fairness in mind, since CNNs, used as baselines in our comparisons, are limited to local processing and traditionally rely on hierarchical networks to augment their effective receptive field. We construct two model variants using the 15M and 28M parameters. Results are reported in Table 1. We observe that the DFC-CP layers consistently outperform the other layer types across all metrics and complexity levels, achieving up to 2.5% higher top-1 accuracy, 1.2% higher top-5 accuracy, 3.0% higher v2 accuracy, and 1.6% higher real accuracy than the best competing layer type. Metrics on the three test-set show a similar trend, providing evidence that, despite its soft inductive biases, an architecture equipped with DFC layers does not tend to overfit. This demonstrates the effectiveness of the DFC inductive biases in capturing the multi-scale features of the input given a fixed computational budget.

Isotropic Architecture We examine the capability of our DFC block design to generalize to isotropic architectures, where no downsampling layers are used across the model. In this set of experiments, features are kept at the same 14×14 resolution and dimensionality throughout the entire network. We construct two isotropic models using $R = 192$ and $R = 384$ feature dimensions. Depths are set at 12 to match the number of layers in the hierarchical version of our model. Results are reported in Table 2 and show how DFC layers are able to outperform FC layers (non-dynamic) and Convolutional layers (non-dynamic, local) by a large margin, replicating a trend observed in the hierarchical case. Interestingly, FC global receptive field does benefit performance. This result, together with a direct comparison between DFC and FC, suggests the benefit of using adaptive receptive fields, that can perform both local or global reasoning depending on the task at hand. Moreover, compared to the best competing layer type, the DFC-CP layers achieve a boost in performance of up to 3.6% higher top-1 accuracy, 1.8% higher top-5 accuracy, 5.3% higher v2 accuracy, and 3.1% higher real accuracy, providing further evidence that DFC design is competitive in different network designs.

Rank R As described in the method section, the CP-Decomposed DFC layer acts as an approximation for the non-linear extension of FC layers. Therefore, a natural question is to assess the change in the performance of our network when a different fraction of the original channel is used as R. In this ablation, we train from scratch the same an Isotropic network allowing a fixed higher or lower number of channels in the latent space. Table 3 illustrates the performance of the same model with different R and the same $C = D$. It is evident that, as R decreases, the performance degrades. In fact, in these cases, the decomposition is not able to approximate properly the weight tensor, and thus converges to a suboptimal solution.

Comparison with other Dynamic Layers Lastly, we compare our method with other dynamic layers, by substituting our spatial processing module (modulation + SAT) with other well-known spatial operators that provide instance-adaptive response: Self-Attention [43], Linear Attention [21], and Dynamic-Convolution [27, 47]. We use the small hierarchical architecture for our comparison. As visible in Table 6, our method achieves the best efficiency performance trade-off. Moreover, thanks to the use of modulation and SAT, is the only method that processes spatial information with a complexity that is *independent from the size of the input image*.

Dynamic Layer	Complexity	P(M)	F(G)	T1	T5
Dynamic Convolution	$\mathcal{O}(K \cdot N \cdot C)$	17	2.6	80.4	95.3
Self-Attention	$\mathcal{O}(N^2 \cdot C)$	17	2.5	**80.9**	**95.5**
Linear Self-Attention	$\mathcal{O}(N \cdot C^2)$	17	2.4	79.3	94.9
Modulation+SAT (Ours)	$\mathcal{O}(C)$	**15**	**2.3**	**80.9**	**95.5**

Table 6: **Comparisons with other dynamic layers**. N spatial size, C number of channels, K kernel size. Our method achieves the best performance and lowest complexity.

Layer		Complexity		Detection			Segmentation		
Type	Weights	P(M)	F(G)	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
DFC-CP	$\approx W_{imncd}$	15	2.3	**38.8**	**58.3**	**41.2**	**21.9**	**42.1**	**51.6**
		28	4.5	**41.2**	**61.4**	**44.0**	**24.1**	**44.6**	**55.0**
FC -CP	$\approx W_{mncd}$	15	2.4	-	-	-	-	-	-
		28	4.5	-	-	-	-	-	-
Conv-CP	$\approx W_{kcd}$	15	2.4	37.3	56.9	39.6	19.9	40.8	49.4
		28	4.5	40.5	60.5	43.4	23.3	44.2	53.8
Conv-CP	$\approx P_{kcd}$	15	2.4	36.7	56.8	39.0	19.8	39.8	49.0
		28	4.5	39.0	59.5	41.5	22.8	42.1	51.1

Table 4: **Detection and Segmentation using RetinaNet**. Under the same complexity and training strategy, our method outperforms others by a large margin thanks to the use of dynamic weights, the generation of a spatially adaptive response, and the leveraging of an adaptive receptive field.

Layer		Complexity		Detection			Segmentation		
Type	Weights	P(M)	F(G)	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
DFC-CP	$\approx W_{imncd}$	15	2.3	**40.1**	**61.4**	**43.8**	**37.1**	**58.6**	**39.6**
		28	4.5	**42.4**	**63.6**	**46.7**	**38.7**	**60.5**	**41.6**
FC -CP	$\approx W_{mncd}$	15	2.4	-	-	-	-	-	-
		28	4.5	-	-	-	-	-	-
Conv-CP	$\approx W_{kcd}$	15	2.4	38.7	60.1	41.9	35.8	57.0	38.2
		28	4.5	41.5	63.0	45.6	38.2	60.2	41.0
Conv-CP	$\approx P_{kcd}$	15	2.4	38.0	59.5	41.3	35.5	56.6	37.6
		28	4.5	40.7	62.6	44.4	37.3	59.7	39.8

Table 5: **Detection and Segmentation using Mask-RCNN**. A stack of DFC layers serves as effective features extractor for a given computational budget. Note that FC layers cannot process input of different sizes and cannot be used as a backbone for downstream tasks.

5.2. Downstream Vision Tasks

We show that our method can serve as a backbone for different computer vision applications that require high-level feature extraction and reasoning. To this end, we use the hierarchical models pre-trained on ImageNet 1K and test them on the dense prediction tasks of object detection and segmentation on the COCO dataset [31]. Our goal is to demonstrate the use of DFC layers as extractors of pixel-level features for downstream tasks. For a fair comparison, we compare the backbones under the same training setup, macro design choices, and computational budget, and report the best performance among three runs with different random initialization for each method. Moreover, to test their generalization ability, we use two types of widely used detectors that represent different detection frameworks: RetinaNet [30], a single-stage detector, and Mask R-CNN [15], a two-stage detector. We train on the train2017 split and test on the 5K validation images of val2017. We employ a 12-epoch training schedule and use AdamW optimizer with an initial learning rate and weight decay of $1e^{-4}$, batch size of 16. We fix the short size of the image to 800 pixels during testing. We train on 8 GPUs and report performance for the best epoch. Results are shown in Table 4 and Table 5. Consistently with the classification results, our method significantly outperforms its counterparts. Compared with a traditional CP-Decomposed Conv, our method achieves a substantial improvement of $+1.5$ AP and $+0.7$ AP for a backbone of 15 and 28 millions of parameters respectively. Compared to the architecture using FC layers, our method can handle inputs of varying dimensions and can be used as a backbone for the tasks. Furthermore, the results show a consistent trend across the two tables, indicating the generalization ability of our method across detectors. Additionally, we observe that the performance variation across runs is no larger than 0.2 AP, demonstrating the robustness of our results. We show that DFC layers can replace traditional convolution layers with the same complexity while achieving better performance, dynamic weight generation, spatially adaptive response, and global reasoning by design.

6. Conclusion

In this paper, we introduce a new Dynamic Fully-Connected layer, a non-linear extension of a Fully-Connected layer that has an adaptive receptive field, is instance-adaptive, and spatially aware. Starting from our analysis of deep learning layers using Einstein notation and CP decomposition, we design a new factorized operator for neural networks that can dynamically adjust the receptive field size in constant time, resulting in significant complexity reduction without compromising expressivity. In the task of image classification with a fixed computational budget measured in Floating Point Operations, our operator outperforms traditional static layers by almost 2%, with a trend consistent across architectures and sizes. It also achieves the same performance and lower complexity when compared to other dynamic layers. Moreover, when used to extract feature representation for the downstream tasks of detection and segmentation, it improved results up to 3% without increasing the parameter count. These results demonstrate that our method is able to learn a data representation superior to traditional layers without increasing the underlying complexity of the architecture, thus establishing a novel paradigm for neural network design.

Despite these promising initial results, the performance of neural networks on any given task depends not only on the choice of its building blocks, but also on other factors such as the architectural macro-designs (e.g. how to combine building blocks sequentially, the optimal depth-width ratio, the patch generation strategy), the training techniques (e.g. data augmentation, optimization method) and implementation strategy for the DFC layer (e.g. alternative factorization, implementation of gating mechanisms). This exploration is left for future work. Next, we plan to release our code on Pytorch [38] and MindSpore [18] deep learning computing libraries and to explore the relationship between our framework, the CP-Decomposed variant of the DFC layer, and other dynamic layers.

References

[1] Alaaeldin Ali, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al. Xcit: Cross-covariance image transformers. *Advances in neural information processing systems*, 34, 2021. 2

[2] Lucas Beyer, Olivier J Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aäron van den Oord. Are we done with imagenet? *arXiv preprint arXiv:2006.07159*, 2020. 6

[3] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11030–11039, 2020. 2

[4] Grigorios G Chrysos, Markos Georgopoulos, Jiankang Deng, Jean Kossaifi, Yannis Panagakis, and Anima Anandkumar. Augmenting deep classifiers with polynomial neural networks. In *European Conference on Computer Vision*, pages 692–716. Springer, 2022. 2

[5] Grigorios G Chrysos, Stylianos Moschoglou, Giorgos Bouritsas, Jiankang Deng, Yannis Panagakis, and Stefanos Zafeiriou. Deep polynomial neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4021–4034, 2021. 2

[6] Franklin C Crow. Summed-area tables for texture mapping. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 207–212, 1984. 1, 2, 4

[7] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017. 4

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1

[10] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019. 2

[11] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pages 291–326. Chapman and Hall/CRC, 2022. 2

[12] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021. 2

[13] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. 2

[14] Kohei Hayashi, Taiki Yamaguchi, Yohei Sugawara, and Shin-ichi Maeda. Exploring unexplored tensor network decompositions for convolutional neural networks. *Advances in Neural Information Processing Systems*, 32, 2019. 1, 2

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 8

[16] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 3

[17] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 2, 4

[18] Ltd. Huawei Technologies Co. Huawei mindspore ai development framework. In *Artificial Intelligence Technology*, pages 137–162. Springer, 2022. 8

[19] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th international conference on computer vision*, pages 2146–2153. IEEE, 2009. 2

[20] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. *Advances in neural information processing systems*, 29, 2016. 2

[21] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020. 2, 7

[22] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009. 1, 2, 3

[23] Jean Kossaifi, Antoine Toisoul, Adrian Bulat, Yannis Panagakis, Timothy M Hospedales, and Maja Pantic. Factorized higher-order cnns with an application to spatio-temporal emotion estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6060–6069, 2020. 2

[24] Sören Laue, Matthias Mitterreiter, and Joachim Giesen. A simple and efficient tensor calculus. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4527–4534, 2020. 2

[25] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014. 2, 6

[26] Yann LeCun et al. Generalization and network design strategies. *Connectionism in perspective*, 19(143-155):18, 1989. 3

[27] Duo Li, Jie Hu, Changhu Wang, Xiangtai Li, Qi She, Lei Zhu, Tong Zhang, and Qifeng Chen. Involution: Inverting the inherence of convolution for visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12321–12330, 2021. 2, 7

[28] Yang Li, Lukasz Kaiser, Samy Bengio, and Si Si. Area attention. In *International Conference on Machine Learning*, pages 3846–3855. PMLR, 2019. 2

[29] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021. 2

[30] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 8

[31] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 8

[32] Vasileios Lioutas and Yuhong Guo. Time-aware large kernel convolutions. In *International Conference on Machine Learning*, pages 6172–6183. PMLR, 2020. 2

[33] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022. 1

[34] Xindian Ma, Peng Zhang, Shuai Zhang, Nan Duan, Yuexian Hou, Ming Zhou, and Dawei Song. A tensorized transformer for language modeling. *Advances in neural information processing systems*, 32, 2019. 2

[35] Ben Mildenhall, Jonathan T Barron, Jiawen Chen, Dillon Sharlet, Ren Ng, and Robert Carroll. Burst denoising with kernel prediction networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2502–2510, 2018. 2

[36] Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. *Advances in neural information processing systems*, 28, 2015. 2, 6

[37] Yannis Panagakis, Jean Kossaifi, Grigorios G Chrysos, James Oldfield, Mihalis A Nicolaou, Anima Anandkumar, and Stefanos Zafeiriou. Tensor methods in computer vision and deep learning. *Proceedings of the IEEE*, 109(5):863–890, 2021. 1, 2

[38] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 8

[39] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pages 5389–5400. PMLR, 2019. 6

[40] Tim Rocktäschel. Einsum is all you need - einstein summation in deep learning, 2018. 2

[41] Alex Rogozhnikov. Einops: Clear and reliable tensor manipulations with einstein-like notation. In *International Conference on Learning Representations*, 2021. 2

[42] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958. 3

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2, 4, 7

[44] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004. 1, 2, 4

[45] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. 4

[46] Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021. 1

[47] Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*, 2019. 2, 4, 7

[48] Zhihao Xia, Federico Perazzi, Michaël Gharbi, Kalyan Sunkavalli, and Ayan Chakrabarti. Basis prediction networks for effective burst denoising with large kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11844–11853, 2020. 2

[49] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10819–10829, 2022. 2, 6

[50] Linguang Zhang, Maciej Halber, and Szymon Rusinkiewicz. Accelerating large-kernel convolution using summed-area tables. *arXiv preprint arXiv:1906.11367*, 2019. 2, 4

[51] Lin Zheng, Huijie Pan, and Lingpeng Kong. Ripple attention for visual perception with sub-quadratic complexity. In *International Conference on Machine Learning*, pages 26993–27010. PMLR, 2022. 2

[52] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9308–9316, 2019. 4