DUAL-TREE WAVELET PACKET CNNS FOR IMAGE CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we target an important issue of deep convolutional neural networks (CNNs) – the lack of a mathematical understanding of their properties. We present an explicit formalism that is motivated by the similarities between trained CNN kernels and oriented Gabor filters for addressing this problem. The core idea is to constrain the behavior of convolutional layers by splitting them into a succession of wavelet packet decompositions, which are modulated by freely-trained mixture weights. We evaluate our approach with three variants of wavelet decompositions with the AlexNet architecture for image classification as an example. The first variant relies on the separable wavelet packet transform while the other two implement the 2D dual-tree real and complex wavelet packet transforms, taking advantage of their feature extraction properties such as directional selectivity and shift invariance. Our experiments show that we achieve the accuracy rate of standard AlexNet, but with a significantly lower number of parameters, and an interpretation of the network that is grounded in mathematical theory.

1 INTRODUCTION

Deep convolutional neural networks (CNNs) have dramatically improved state-of-the-art performances in many domains such as speech recognition, visual object recognition or object detection (LeCun et al., 2015). However, they are very resource-intensive and a full mathematical understanding of their properties remains a challenging issue.

On the other hand, in the field of signal processing, wavelet and multi-resolution analysis are built upon a well-established mathematical framework. They have proven to be very efficient in tasks such as signal compression and denoising (Mallat, 2009). Moreover, wavelet filters have been widely used as feature extractors for signal, image and texture classification (Laine & Fan, 1993; Pittner & Kamarthi, 1999; Yen, 2000; Huang & Aviyente, 2008).

While both fields rely on filters to achieve their goals, the two approaches are radically different. In wavelet analysis, filters are specifically designed to meet very restrictive conditions, whereas CNNs use freely trained filters, without any prior assumption on their behavior. Nevertheless, in many computer vision tasks, CNNs tend to learn parameters that are pretty similar to oriented Gabor filters in the first layer (Boureau et al., 2010; Yosinski et al., 2014). This phenomenon suggests that early layers extract general features such as edges or basic shapes, which are independent from the task at hand.

Proposed approach In order to improve our understanding of CNNs, we propose to constrain their behavior by replacing freely-trained filters by a series of discrete wavelet packet decompositions modulated by mixture weights. We therefore introduce prior assumptions to guide learning and reduce the number of trainable parameters in convolution layers, while retaining predictive power.

The main goal of our work is to describe and interpret the observed behavior of CNNs with a sparse model, taking advantage of the feature extraction properties of wavelet packet transforms. By increasing control over the network, we pave the way for future applications in which theoretical guarantees are critical.

In this paper we describe our wavelet packet CNN architectures with a mathematical formulation and introduce an algorithm to visualize the resulting filters. As a proof of concept, we based our experiments on AlexNet (Krizhevsky et al., 2012). Our choice was driven by the large kernels in its first layer and convolution operations performed with a downsampling factor of 4. This allows to perform two levels of wavelet decomposition without any additional transformation, and facilitates visual comparison with our own custom filters. Note however that most CNNs trained on natural image datasets exhibit the same oscillating patterns. We therefore believe that our work could be extended to other architectures with a few adaptations.

Related work In a similar spirit, a few attempts to combine the two research fields have been made in recent years. Wavelet scattering networks (Bruna & Mallat, 2013) compute CNN-like cascading wavelet convolutions to get translation-invariant image representations that are stable to deformation and preserve high-frequency information. They were later adapted to the discrete case using complex oriented wavelet frames (Singh & Kingsbury, 2017). While these networks are designed from scratch and are totally deterministic, other approaches enhance existing networks with wavelet filter preprocessing or embedding. The goal is either to improve classification performance without increasing the network complexity (Chang & Morgan, 2014; Williams & Li, 2016; Fujieda et al., 2017; Williams & Li, 2018; Lu et al., 2018; Luan et al., 2018), or to replace freely-trained layers by more constrained structures implementing spectral filtering. Such models include Gabor filters in parallel to regular trainable weight kernels (Sarwar et al., 2017), wavelet scattering coefficients as the input of a CNN (Oyallon et al., 2018), or linear combinations of discrete cosine transforms (Ulicny et al., 2019).

Our approach falls into this second category, although our design is based upon a different CNN architecture, i.e., AlexNet. To our knowledge we are the first to introduce the dual-tree wavelet packet transform (DT-$\mathbb{C}$WPT) (Bayram & Selesnick, 2008) in such context. Similar to the filters used in the above papers, wavelet packet transforms are well-localized in the frequency domain and share a subsampling factor over the output feature maps. A major advantage with our approach is sparsity: a single vector (called conjugate mirror filter, or CMF) is sufficient to characterize the whole process. Moreover, like Gabor filters, DT-$\mathbb{C}$WPT extracts oriented and shift-invariant features, but achieves this goal with minimal redundancy, while providing an efficient decomposition algorithm based on separable filter banks. Regarding the discrete cosine transform, its complexity is similar to DT-$\mathbb{C}$WPT but lacks orientation properties. Our models therefore provide a sparser description of the observed behavior of convolutional layers. This is a step toward a more complete description of CNNs by using a small number of arbitrary parameters.

2 Background

Notations In this paper, d-dimensional tensors are written with straight bold capital letters: $\mathbf{Z} \in \mathbb{R}^{A_1 \times \cdots \times A_d}$, where A_i denotes the size of $\mathbf{Z}$ along its i-th dimension; the shape of $\mathbf{Z}$ is denoted $\langle \mathbf{Z} \rangle = \begin{pmatrix} A_1 & \dots & A_d \end{pmatrix}^\top$. 2D matrices are written in italic: $\boldsymbol{U} \in \mathbb{R}^{A \times B}$ and 1D vectors in bold lower-case letters: $\boldsymbol{z} \in \mathbb{R}^A$. For the sake of legibility, indices are written between square brackets.

The convolution between two matrices $\boldsymbol{U} \in \mathbb{R}^{A \times B}$ and $\boldsymbol{V} \in \mathbb{R}^{A' \times B'}$ is defined, for all $m \in \{0 \,..\, A + A' - 2\}$ and $n \in \{0 \,..\, B + B' - 2\}$, by $(\boldsymbol{U} * \boldsymbol{V})[m, n] = \sum_{i,j} \boldsymbol{U}[m - i, n - j] \cdot \boldsymbol{V}[i, j]$. Since some indices are negative or bigger than the matrix size, $\boldsymbol{U}$ and $\boldsymbol{V}$ must be extended beyond their limits, either by setting all outside values to zero, or by using a periodic or symmetric pattern. Practical implications of this choice will not be discussed in this paper.

For any $\boldsymbol{U} \in \mathbb{R}^{A \times B}$, $\overline{\boldsymbol{U}}$ denotes the flipped matrix: $\overline{\boldsymbol{U}}[m, n] = \boldsymbol{U}[A - (m + 1), B - (n + 1)]$. The upsampling and downsampling operators are respectively denoted $\uparrow$ and $\downarrow$. For any $\alpha \in \mathbb{N}^*$, $(\boldsymbol{U} \uparrow \alpha)[m, n] = \boldsymbol{U}\left[\frac{m}{\alpha}, \frac{n}{\alpha}\right]$ if both m and n are divisible by α ($= 0$ otherwise), and $(\boldsymbol{U} \downarrow \alpha)[m, n] = \boldsymbol{U}[\alpha m, \alpha n]$. Finally, for any scalar $z \in \mathbb{R}$, we denote $z + \boldsymbol{U} = z\boldsymbol{J} + \boldsymbol{U}$, where $\boldsymbol{J} \in \mathbb{R}^{A \times B}$ denotes the matrix of ones.

Discrete wavelet packet transform (WPT) This is a brief overview on the WPT algorithm (Mallat, 2009), written as a sequence of matrix convolutions. An illustration of the transform is given in Appendix A.1.

We will implicitly build a discrete orthogonal basis in which any matrix $\boldsymbol{X} \in \mathbb{R}^{N \times N}$ can be decomposed. The basis is made of oriented 2D waveforms with high frequency resolution, which is an interesting property for feature extraction. Considering a pair of conjugate mirror filters (CMFs) $\boldsymbol{h}$ and $\boldsymbol{g} \in \mathbb{R}^{\mu}$, we build a separable 2D filter bank, made of one low-pass filter $\boldsymbol{G}^{(0)} = \boldsymbol{h} \cdot \boldsymbol{h}^{\top}$ and three high-pass filters $\boldsymbol{G}^{(1)} = \boldsymbol{h} \cdot \boldsymbol{g}^{\top}$, $\boldsymbol{G}^{(2)} = \boldsymbol{g} \cdot \boldsymbol{h}^{\top}$ and $\boldsymbol{G}^{(3)} = \boldsymbol{g} \cdot \boldsymbol{g}^{\top}$.

We start the decomposition with $\boldsymbol{D}_0^{(0)} = \boldsymbol{X}$. Let us assume that for a given $j \in \mathbb{N}$, the feature maps of wavelet packet coefficients at scale j, denoted $\boldsymbol{D}_j^{(k)}$, have already been computed for any $k \in \left\{0 \,..\, 4^j - 1\right\}$. Then we compute the wavelet packet coefficients at the coarser scale $j+1$ by decomposing each feature map $\boldsymbol{D}_j^{(k)}$ into four smaller submatrices:

$$\forall l \in \{0 \,..\, 3\}\,,\ \boldsymbol{D}_{j+1}^{(4k+l)} = \left(\boldsymbol{D}_j^{(k)} * \overline{\boldsymbol{G}^{(l)}}\right) \downarrow 2\,. \tag{1}$$

At each scale $j > 0$, the set of 4^j matrices $\left\{\boldsymbol{D}_j^{(k)}\right\}$ is a representation of $\boldsymbol{X}$ from which the original signal can be reconstructed. Figure 1 illustrates the WPT resulting filters with $j = 2$.

Dual-tree complex wavelet packet transform (DT-$\mathbb{C}$WPT) WPT has interesting properties such as sparse signal representation and vertical / horizontal feature discrimination. However, it suffers from a lack of shift invariance and a poor directional selectivity. To overcome this, Kingsbury (2001) designed a discrete wavelet transform in which input images are efficiently decomposed in a tight frame of complex oriented waveforms with limited redundancy. It was generalized to the wavelet packet framework by Bayram & Selesnick (2008).

In a nutshell, let us assume that we have decomposed an input matrix $\boldsymbol{X}$ into four WPT representations $\left\{\boldsymbol{D}_{\mathrm{a},\,j}^{(k)}\right\}$, $\left\{\boldsymbol{D}_{\mathrm{b},\,j}^{(k)}\right\}$, $\left\{\boldsymbol{D}_{\mathrm{c},\,j}^{(k)}\right\}$, $\left\{\boldsymbol{D}_{\mathrm{d},\,j}^{(k)}\right\}$. This is achieved by applying expression (1) with four suitable filter banks $\left\{\boldsymbol{G}_{\mathrm{a}}^{(l)}\right\}$, $\left\{\boldsymbol{G}_{\mathrm{b}}^{(l)}\right\}$, $\left\{\boldsymbol{G}_{\mathrm{c}}^{(l)}\right\}$, $\left\{\boldsymbol{G}_{\mathrm{d}}^{(l)}\right\}$. Then we can compute the following complex wavelet packet coefficients $\boldsymbol{E}_j^{\nearrow(k)}$ and $\boldsymbol{E}_j^{\nwarrow(k)}$, for each $k \in \left\{0 \,..\, 4^j - 1\right\}$:

$$\begin{pmatrix} \boldsymbol{E}_j^{\nearrow(k)} \\ \boldsymbol{E}_j^{\nwarrow(k)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & -\boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{D}_{\mathrm{a},\,j}^{(k)} \\ \boldsymbol{D}_{\mathrm{d},\,j}^{(k)} \end{pmatrix} + i \cdot \begin{pmatrix} \boldsymbol{I} & \boldsymbol{I} \\ \boldsymbol{I} & -\boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{D}_{\mathrm{c},\,j}^{(k)} \\ \boldsymbol{D}_{\mathrm{b},\,j}^{(k)} \end{pmatrix}\,. \tag{2}$$

For a given scale $j > 0$, the set of $(2 \cdot 4^j)$ complex matrices $\left\{\boldsymbol{E}_j^{\nearrow(k)},\, \boldsymbol{E}_j^{\nwarrow(k)}\right\}$ constitutes a redundant representation of $\boldsymbol{X}$ from which the original signal can be reconstructed. DT-$\mathbb{C}$WPT is oriented, and nearly shift invariant if we consider the modulus of complex coefficients. Figure 1 illustrates the DT-$\mathbb{C}$WPT resulting filters with $j = 2$.

Dual-tree real wavelet packet transform (DT-$\mathbb{R}$WPT) By computing only the real part of the above coefficients, we get a representation of $\boldsymbol{X}$ in a real tight frame. Like above, DT-$\mathbb{R}$WPT is an oriented transform, but does not possess the shift invariance property. This may have consequences on its predictive power, as will be seen later.

Link with Gabor filters Such as presented above, the wavelet packet transforms compute a full decomposition in what Mallat (2009) calls a pseudo-local cosine basis. The resulting filters have identical window size with a varying number of oscillations within these windows (see Figure 1). Therefore, such wavelet packets share similarities with Gabor filters. However, they offer a competitive advantage: the decomposition is performed efficiently using one or few separable filter banks, which are fully characterized by a single one-dimensional vector.

Convolutional layers Let P denote the number of samples (batch size), K (resp. L) the number of input (resp. output) channels, $(M,\, N)$ the size of input feature maps and $(\mu,\, \nu)$ the kernel size.

A 2D convolutional layer with fixed parameters $s,\, d,\, q \in \mathbb{N}^*$ (stride, dilation factor and number of groups, respectively), weight $\mathbf{W} \in \mathbb{R}^{(K/q) \times L \times \mu \times \nu}$ and bias $\boldsymbol{b} \in \mathbb{R}^L$, transforms any 4D input tensor

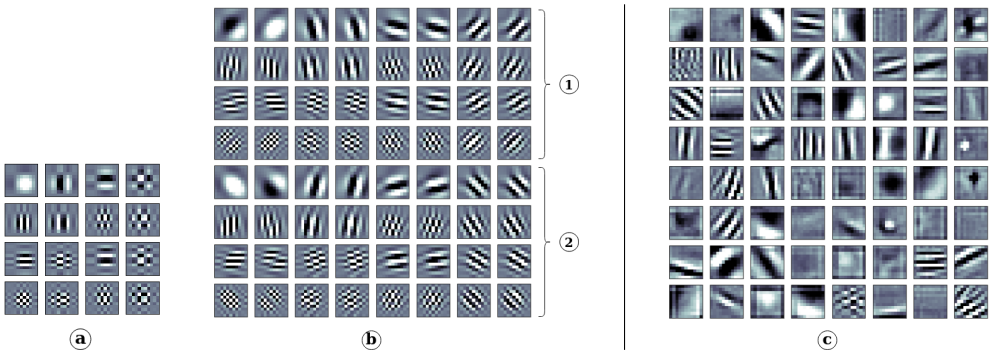

Figure 1: ⓐ-ⓑ Respectively, WPT and DT-$\mathbb{C}$WPT filters for $j = 2$, computed with Q-shift orthogonal CMFs of length 10 (Kingsbury, 2003). The matrices have been cropped to 11×11. ⓑ displays 32 complex filters, alternatively represented by their real and imaginary parts. ① and ② are the filters computing $\boldsymbol{E}_j^{\nearrow(k)}$ and $\boldsymbol{E}_j^{\nwarrow(k)}$, respectively. ⓒ Convolution kernels $\mathbf{W}_{\text{alex}}[0,\, k]$ in AlexNet's first layer. We used a model from the Torchvision package, pretrained on ImageNet.

$\mathbf{X} \in \mathbb{R}^{P \times K \times M \times N}$ into an output tensor $\mathbf{Y} \in \mathbb{R}^{P \times L \times M' \times N'}$, such that

$$\mathbf{Y}[p,\, l] = \boldsymbol{b}[l] + \sum_{k=0}^{K/q-1} \left(\mathbf{X}[p,\, k_0(l) + k] * \left(\overline{\mathbf{W}[k,\, l]} \uparrow d \right) \right) \downarrow s \,, \tag{3}$$

where $k_0(l) = \lfloor lq/L \rfloor \cdot K/q$ denotes the first input channel influencing the l-th output. Note that in expression (3), $\mathbf{Y}[p,\, l]$, $\mathbf{X}[p,\, k_0(l) + k]$ and $\mathbf{W}[k,\, l]$ are 2D matrices while $\boldsymbol{b}[l]$ is a scalar.

Definition 1. We denote $\mathcal{C}_{s,\,d}^{(q)}(\mathbf{W},\, \boldsymbol{b})$ the operator computing (3): $\mathbf{Y} = \mathcal{C}_{s,\,d}^{(q)}(\mathbf{W},\, \boldsymbol{b}) \cdot \mathbf{X}$.

In this paper, we focus on AlexNet's first convolutional layer, which can be represented as a convolution operator $\mathcal{C}_{4,\,1}^{(1)}(\mathbf{W},\, \boldsymbol{b})$, with $\mathbf{W} \in \mathbb{R}^{3 \times 64 \times 11 \times 11}$ and $\boldsymbol{b} \in \mathbb{R}^{64}$. The kernels $\mathbf{W}[0,\, k]$ after training with ImageNet are displayed Figure 1. We can notice oriented oscillating patterns similar to wavelet packet filters.

3 Proposed models

We now introduce several network architectures that are built on standard AlexNet, in which the first 11×11 convolutional layer is replaced by a succession of WPT or DT-($\mathbb{RC}$)WPT decompositions modulated by mixture weights. Each network takes as input a 4D tensor $\mathbf{X} \in \mathbb{R}^{P \times 3 \times 224 \times 224}$, i.e., a set of P images with three input channels (RGB images).

3.1 WPT module

This module computes two successive WPT decompositions (1) for every input channel. Each step $j \in \{0,\, 1\}$ is implemented as a strided convolution operator $\mathcal{C}_{s,\,d}^{(q_j)}(\mathbf{W}_j,\, \mathbf{0})$ (see Definition 1), where $\mathbf{W}_j$ contains the fixed low- and high-pass filters. In this configuration, each input channel is convolved with its own set of filters.

More precisely, we have $(s = 2)$, $(d = 1)$ and $(q_j = K_j)$, where $K_j = (3 \cdot 4^j)$ denotes the number of input channels. $\mathbf{W}_j \in \mathbb{R}^{1 \times (4K_j) \times \mu \times \mu}$ is such that for all $k \in \{0 \,..\, K_j - 1\}$ and $l \in \{0 \,..\, 3\}$, $\mathbf{W}_j[0,\, 4k + l] = \boldsymbol{G}^{(l)}$. The output, denoted $\mathbf{D} \in \mathbb{R}^{P \times 48 \times N' \times N'}$, is such that

$$\mathbf{D} = \mathcal{C}_{2,\,1}^{(12)}(\mathbf{W}_1,\, \mathbf{0}) \cdot \left(\mathcal{C}_{2,\,1}^{(3)}(\mathbf{W}_0,\, \mathbf{0}) \cdot \mathbf{X} \right) \,. \tag{4}$$

Once this is done, we need to modulate the importance of each wavelet packet. Moreover, the number of output channels must be equal to 64 as in standard AlexNet, and every output channel

must be influenced by each input RGB channel. This is achieved with a 1×1 convolutional layer (Lin et al., 2014; Szegedy et al., 2015) placed after the WPT decomposition. Note that this approach was also chosen by Ulicny et al. (2019) in what they call a harmonic block.

Therefore, the final output, denoted $\mathbf{Y}_{\mathrm{wpt}} \in \mathbb{R}^{P \times 64 \times 56 \times 56}$, is such that

$$\mathbf{Y}_{\mathrm{wpt}} = \mathcal{C}_{1,1}^{(1)}(\mathbf{W}_{\mathrm{mix}},\, \boldsymbol{b}_{\mathrm{mix}}) \cdot \mathbf{D}\ , \tag{5}$$

where $\mathbf{W}_{\mathrm{mix}} \in \mathbb{R}^{48 \times 64 \times 1 \times 1}$ and $\boldsymbol{b}_{\mathrm{mix}} \in \mathbb{R}^{64}$ are freely trained. A schematic representation of the WPT module can be found in Figure 2-②. The orange ("FB", a.k.a., filter bank) and green ("Conv") layers compute expressions (4) and (5), respectively.

Number of trainable parameters A WPT module has $|\mathbf{W}_{\mathrm{mix}}| + |\boldsymbol{b}_{\mathrm{mix}}| = 3,136$ trainable parameters, versus $23,296$ for the first convolutional layer in a standard AlexNet.

3.2 DT-($\mathbb{RC}$)WPT MODULES

In DT-$\mathbb{R}$WPT and DT-$\mathbb{C}$WPT modules, respectively two and four suitable filter banks are used on each input channel. The outputs, denoted $\mathbf{E}_{\mathbb{R}} \in \mathbb{R}^{P \times 96 \times N' \times N'}$ and $\mathbf{E}_{\mathbb{C}} \in \mathbb{R}^{P \times 192 \times N' \times N'}$, have the following structure. For any sample $p \in \{0\,..\,P-1\}$,

$$\mathbf{E}_{\mathbb{R}}[p] = \begin{pmatrix} \mathrm{Re}\,\mathbf{E}^{\nearrow}[p] \\ \mathrm{Re}\,\mathbf{E}^{\nwarrow}[p] \end{pmatrix} = \begin{pmatrix} \mathbf{D}_{\mathrm{a}}[p] - \mathbf{D}_{\mathrm{d}}[p] \\ \mathbf{D}_{\mathrm{a}}[p] + \mathbf{D}_{\mathrm{d}}[p] \end{pmatrix}; \quad \mathbf{E}_{\mathbb{C}}[p] = \begin{pmatrix} \mathrm{Re}\,\mathbf{E}^{\nearrow}[p] \\ \mathrm{Re}\,\mathbf{E}^{\nwarrow}[p] \\ \mathrm{Im}\,\mathbf{E}^{\nearrow}[p] \\ \mathrm{Im}\,\mathbf{E}^{\nwarrow}[p] \end{pmatrix} = \begin{pmatrix} \mathbf{D}_{\mathrm{a}}[p] - \mathbf{D}_{\mathrm{d}}[p] \\ \mathbf{D}_{\mathrm{a}}[p] + \mathbf{D}_{\mathrm{d}}[p] \\ \mathbf{D}_{\mathrm{c}}[p] + \mathbf{D}_{\mathrm{b}}[p] \\ \mathbf{D}_{\mathrm{c}}[p] - \mathbf{D}_{\mathrm{b}}[p] \end{pmatrix}, \tag{6}$$

where $\mathbf{D}_{\mathrm{a}}$, $\mathbf{D}_{\mathrm{b}}$, $\mathbf{D}_{\mathrm{c}}$ and $\mathbf{D}_{\mathrm{d}} \in \mathbb{R}^{P \times 48 \times N' \times N'}$ are computed similarly to (4).

Expression (6) is a tensor formulation of (2), where the real and imaginary parts of the complex coefficients are stored separately. As for WPT computed in (4), both DT-$\mathbb{R}$WPT and DT-$\mathbb{C}$WPT can be expressed as a succession of CNN-style convolution operators. This requires a few technicalities that are provided in Appendix A.3.

Again, we placed a 1×1 convolutional layer after the wavelet packet decompositions. The final outputs, denoted $\mathbf{Y}_{\text{dt-}\mathbb{R}\mathrm{wpt}}$ and $\mathbf{Y}_{\text{dt-}\mathbb{C}\mathrm{wpt}} \in \mathbb{R}^{P \times 64 \times N' \times N'}$, are such that

$$\mathbf{Y}_{\text{dt-}\mathbb{R}\mathrm{wpt}} = \mathcal{C}_{1,1}^{(1)}(\mathbf{W}'_{\mathrm{mix}},\, \boldsymbol{b}'_{\mathrm{mix}}) \cdot \mathbf{E}_{\mathbb{R}}; \qquad \mathbf{Y}_{\text{dt-}\mathbb{C}\mathrm{wpt}} = \mathcal{C}_{1,1}^{(1)}(\mathbf{W}''_{\mathrm{mix}},\, \boldsymbol{b}''_{\mathrm{mix}}) \cdot \mathbf{E}_{\mathbb{C}}\ , \tag{7}$$

where $\mathbf{W}'_{\mathrm{mix}} \in \mathbb{R}^{96 \times 64 \times 1 \times 1}$, $\mathbf{W}''_{\mathrm{mix}} \in \mathbb{R}^{192 \times 64 \times 1 \times 1}$, $\boldsymbol{b}'_{\mathrm{mix}}$ and $\boldsymbol{b}''_{\mathrm{mix}} \in \mathbb{R}^{64}$ are freely trained. A schematic representation of both modules can be found in Figure 2-③④. The blue ("$\mp$" and "$\pm$") and green ("Conv") layers compute expressions (6) and (7), respectively.

Number of trainable parameters DT-$\mathbb{R}$WPT and DT-$\mathbb{C}$WPT modules have $6,208$ and $12,352$ trainable parameters, respectively ($23,296$ in a standard AlexNet). We will see in Section 5 how these numbers can be further decreased without degrading the performance of the network.

3.3 KERNEL VISUALIZATION

WPT and DT-($\mathbb{RC}$)WPT modules are designed as a succession of multi-channel convolutional layers. The following proposition states that such cascading layers can be expressed as a single CNN-style convolution operator. It provides an explicit formulation of the resulting hyperparameters – i.e., stride, dilation factor and number of groups describing input-output channel connections – and weight tensor. It takes advantage of the well-known result that two successive convolutions can be written as another convolution with a wider kernel.

Let $\mathcal{C}_{s,1}^{(1)}(\mathbf{W},\, \boldsymbol{b})$, denote an initial convolution operator, with $\mathbf{W} \in \mathbb{R}^{K \times L \times \mu \times \nu}$. We consider a second operator $\mathcal{C}_{t,1}^{(q)}(\mathbf{V},\, \boldsymbol{a})$, with $\mathbf{V} \in \mathbb{R}^{(L/q) \times L' \times \tilde{\mu} \times \tilde{\nu}}$ and $\boldsymbol{a} \in \mathbb{R}^{L'}$ (we assume that both L and L' are divisible by q).

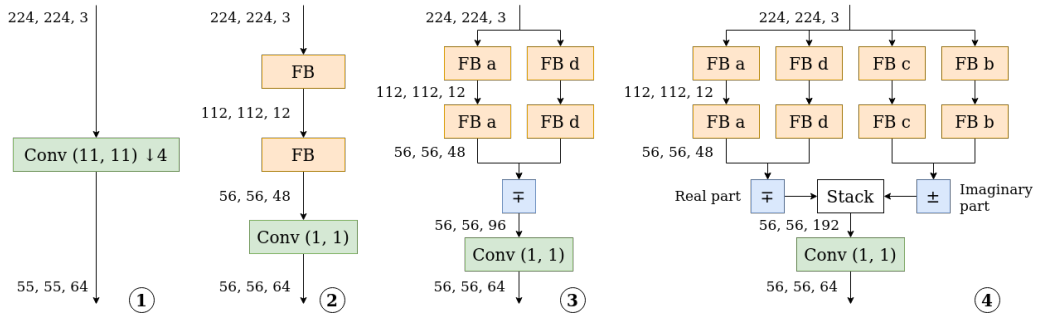

Figure 2: ① First layer of AlexNet; ② WPT module; ③ DT-$\mathbb{R}$WPT module; ④ DT-$\mathbb{C}$WPT module. Only the green layers (Conv) have trainable parameters. The numbers between each layer indicate the height, width and depth (number of channels) of the current feature map tensor.

Proposition 1. *The composition of two strided CNN-style convolution operators such as introduced by Definition 1 can be written as another strided CNN-style convolution operator:*

$$\left(\mathcal{C}_{t,\,1}^{(q)}(\mathbf{V},\,\boldsymbol{a}) \circ \mathcal{C}_{s,\,1}^{(1)}(\mathbf{W},\,\boldsymbol{b})\right) = \mathcal{C}_{s',\,d'}^{(q')}(\mathbf{W}',\,\boldsymbol{b}')\ , \tag{8}$$

with $s' = st$ (stride), $d' = 1$ (dilation) and $q' = 1$ (number of groups). Moreover, the resulting weight $\mathbf{W}'$ is computed using a dilated CNN-style convolution operator:

$$\overline{\mathbf{W}'} = \mathcal{C}_{s_{\mathrm{w}},\,d_{\mathrm{w}}}^{(q_{\mathrm{w}})}(\mathbf{V},\,\mathbf{0}) \cdot \overline{\mathbf{W}}\ , \tag{9}$$

with $(s_{\mathrm{w}} = 1)$, $(d_{\mathrm{w}} = s)$ and $(q_{\mathrm{w}} = q)$. We also have $\boldsymbol{b}' = \boldsymbol{a} + f(\mathbf{V},\,\boldsymbol{b})$, where f is such that $f(\mathbf{V},\,\mathbf{0}) = \mathbf{0}$. An explicit formulation of f is given in the Appendix A.4.

Remark. Proposition 1 is not exactly true when using a symmetric padding (see Section 2 – Notations). Nevertheless, distortion effects only appear at the edges of images; and the property holds everywhere else. We leave the study of the influence of such choice on the network's predictive power for future work.

Proposition 1, whose proof is given in Appendix A.4, shows that the wavelet packet modules compute

$$\mathbf{Y}_{\mathrm{wpt}} = \mathcal{C}_{4,\,1}^{(1)}(\mathbf{W}_{\mathrm{wpt}},\,\boldsymbol{b}_{\mathrm{mix}}) \cdot \mathbf{X}\ ; \tag{10}$$

$$\mathbf{Y}_{\text{dt-}\mathbb{R}\mathrm{wpt}} = \mathcal{C}_{4,\,1}^{(1)}(\mathbf{W}_{\text{dt-}\mathbb{R}\mathrm{wpt}},\,\boldsymbol{b}'_{\mathrm{mix}}) \cdot \mathbf{X}\ ; \quad \mathbf{Y}_{\text{dt-}\mathbb{C}\mathrm{wpt}} = \mathcal{C}_{4,\,1}^{(1)}(\mathbf{W}_{\text{dt-}\mathbb{C}\mathrm{wpt}},\,\boldsymbol{b}''_{\mathrm{mix}}) \cdot \mathbf{X}\ , \tag{11}$$

where $\mathbf{W}_{\mathrm{wpt}}, \mathbf{W}_{\text{dt-}\mathbb{R}\mathrm{wpt}}, \mathbf{W}_{\text{dt-}\mathbb{C}\mathrm{wpt}} \in \mathbb{R}^{3\times 64\times(3\mu-2)\times(3\mu-2)}$ are obtained from (9). As a reminder, μ denotes the size of the CMF. By means of comparison, AlexNet's first layer computes

$$\mathbf{Y}_{\mathrm{alex}} = \mathcal{C}_{4,\,1}^{(1)}(\mathbf{W}_{\mathrm{alex}},\,\boldsymbol{b}_{\mathrm{alex}}) \cdot \mathbf{X}\ , \tag{12}$$

where $\mathbf{W}_{\mathrm{alex}} \in \mathbb{R}^{3\times 64\times 11\times 11}$ and $\boldsymbol{b}_{\mathrm{alex}} \in \mathbb{R}^{64}$. Therefore, all modules are represented as convolution operators with stride $s = 4$, mapping 3 input channels to 64 output channels. Regarding kernel size, it is bigger than 11 as long as $\mu \geq 5$; However, most energy is concentrated in a region the size of which is similar to AlexNet kernels.

A visualization of these kernels after training is given in Figure 3. Training details are provided in Section 4. For the sake of visual comparison, we only displayed center patches of size 11×11 from the original matrices – it turns out that in all our models, between 97% and 99% of their energy (i.e., the squared L^2-norm) is concentrated in these cropped regions. We point out that computing the resulting kernels is used for analysis purpose, but should not be involved in the training process.

4 Experiments

4.1 Implementation details

Wavelet filters For the experiments we used a PyTorch / CUDA implementation. Our wavelet packet modules were designed with a Q-shift orthogonal filter of length 10 (Kingsbury, 2003), which

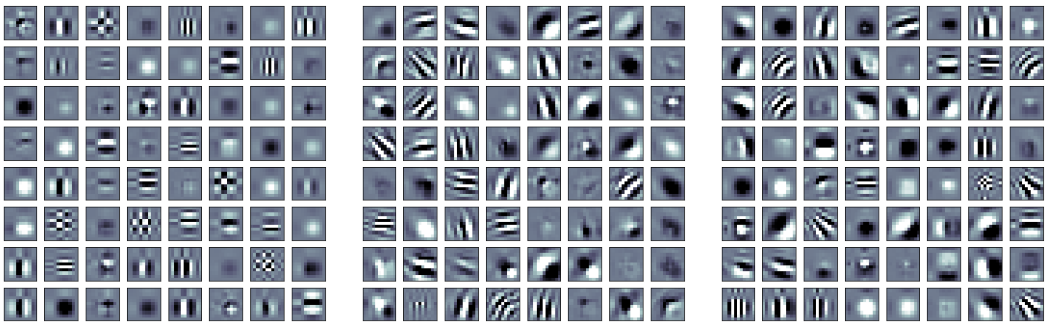

Figure 3: From left to right: $\{\mathbf{W}_{\text{wpt}}[0, k]\}$, $\{\mathbf{W}_{\text{dt-}\mathbb{R}\text{wpt}}[0, k]\}$, $\{\mathbf{W}_{\text{dt-}\mathbb{C}\text{wpt}}[0, k]\}_{k\in\{0..63\}}$ after training with ImageNet ILSVRC2012. All modules are implemented with a Q-shift filter ($\mu = 10$). Whereas the WPT module mainly extracts horizontal and vertical features, many more orientations arise from the dual-tree modules. We can also notice the low-pass filters, which appear as color blobs. The resemblance with AlexNet kernels (see Figure 1) is prominent.

approximately meets the half-sample-shift condition required for the dual-tree transforms. For the sake of setting consistency, we also used this filter for conventional WPT.

Dataset Our models were trained and evaluated on ImageNet ILSVRC2012 dataset (Russakovsky et al., 2015). Since the online evaluation server is no longer available, we set aside $100,000$ images from the training set – 100 per class – in order to create a validation set. We used this subset to compute accuracy rate along the training phase. As for the validation set provided by ImageNet, we turned it into a test set on which our trained models were evaluated.

Training details Each model was trained on a single GPU. The training procedure was inspired from many ILSVRC papers (Krizhevsky et al., 2012; Simonyan & Zisserman, 2015; Szegedy et al., 2015; He et al., 2016). More precisely, it was carried out by optimizing the cross-entropy loss with stochastic gradient descent. For this we fed the network with random batches of 256 images, until we reached 100 cycles through the whole training set (100 epochs, i.e., 461.4K iterations). The momentum was set to 0.9 and weight decay to $5 \cdot 10^{-4}$. As for the learning rate, it was initially set to 10^{-2}, and then decreased by a factor of 10 every 25 epochs.

To reduce overfitting, we followed the data augmentation procedure used in Inception networks (Szegedy et al., 2015). The images are first normalized to a specified mean and standard deviation for each RGB channel. Then they are randomly flipped and cropped from 8% to 100% of their original sizes, with a random aspect ratio varying from $\frac{3}{4}$ to $\frac{4}{3}$, before being resized to 224×224 using a bilinear interpolation.

Model evaluation The test phase was carried out following Krizhevsky et al. (2012). Namely, predictions are made over 10 patches extracted from each input image, and the softmax output vectors are then averaged to get the overall prediction. We used top-1 and top-5 accuracy rate as evaluation metrics.

Comparison with existing models Our models were compared with a standard AlexNet that we trained according to the same procedure. In addition, we wanted to isolate the contribution of wavelet packet modules to the global predictive power. To achieve that we trained an AlexNet in which the first 11×11 convolutional layer was frozen to its initial parameters.

The source code used for our experiments will be published shortly after paper acceptance.

4.2 Results and discussion

The evolution of the loss function and validation error along training is shown in Figure 4. The image classification performance of our trained models, measured as top-1 and top-5 error rates on our custom test set, are displayed in Table 1.

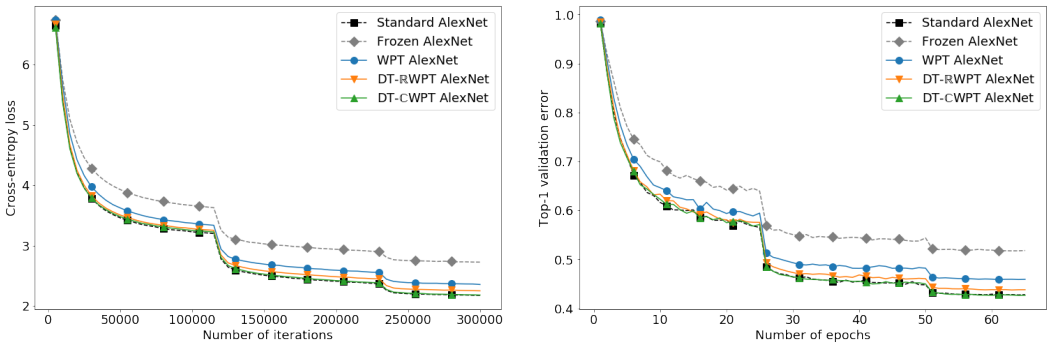

Figure 4: Evolution of the loss function (left) and top-1 validation error (right) during the first 65 training epochs. Validation has been performed by simply resizing the smaller edge of each image to 224 and extracting a single patch of size 224×224 at the center.

Table 1: Comparison of error rates on our custom ImageNet validation and test sets.

Model	**Nb learnt params**	**Top-1** (val)	**Top-5** (val)	**Top-1** (test)	**Top-5** (test)
WPT AlexNet	3.1K	42.2%	20.2%	45.5%	22.9%
DT-ℝWPT AlexNet	6.2K	39.7%	18.2%	43.2%	20.7%
DT-ℂWPT AlexNet	12.4K	38.9%	17.5%	42.3%	20.2%
Standard AlexNet	23.3K	38.9%	17.5%	42.2%	20.0%
Frozen AlexNet	None	47.7%	24.8%	51.1%	27.8%

The DT-ℂWPT model reaches standard AlexNet's accuracy rate with twice less parameters in its first layer. The performances of the WPT and DT-ℝWPT models are slightly below, but still much higher than the frozen version of AlexNet. This result suggests that the predictive power is partly accountable to the wavelet packet modules themselves, and not entirely to the following layers.

By looking at Figure 3, it is easy to explain why conventional WPT has the lowest accuracy of all our models. We identified two main reasons. (1) The filter design makes it impossible to extract oriented features that are neither horizontal nor vertical. Instead, it yields checkerboard patterns, which are useful to catch the remaining information and ensure perfect reconstruction, but fail to process geometric image features like ridges and edges, as pointed out by Selesnick et al. (2005). (2) There is no medium-frequency feature extractor like in AlexNet kernels – black-and-white patches side by side. Such patterns can however be found in dual-tree kernels. If we consider the real and imaginary parts of the complex low-pass filters separately, we actually get one low-pass and one oriented band-pass filter – see Figure 1.

Regarding the DT-ℝWPT model, it performs better than WPT but does not reach the accuracy of DT-ℂWPT or standard AlexNet, despite generating oriented filters. Intuitively, DT-ℂWPT is twice more redundant than DT-ℝWPT, and thus is more likely to extract relevant features for image classification. We propose here a more specific interpretation. In Section 2 we mentioned the shift invariance property of DT-ℂWPT, that neither WPT nor DT-ℝWPT possess. By applying a slight shift to an image, great disturbances can indeed be observed in the matrices of real coefficients (Selesnick et al., 2005). Important features extracted from one image can thus disappear from the other. On the other hand however, the modulus of complex wavelet packet coefficients is nearly shift invariant, meaning that their value is smoothly transferred toward the neighboring pixels in the shift direction. Therefore, any loss of information in their real part is recovered in their imaginary part. The DT-ℂWPT module is capable to extract similar features from two shifted images, which could explain its superior performances.

5 CONCLUSION AND FUTURE WORK

Following an epoch of frantic race toward classification performance, research is more and more focused on understanding learning mechanisms in CNNs. In this perspective, we proposed an architecture in which standard convolutional layers are replaced by dual-tree wavelet packet feature extractors. Our experiments show that such networks can compete with conventional models, providing a sparse description of their behavior.

The DT-$\mathbb{C}$WPT module contains twice less trainable parameters than standard AlexNet's first layer. Future research could be held to further increase the sparsity of our models. (1) Some filters generally extract less information than others, in that the corresponding feature map's energy is much lower. By discarding these feature maps before computing 1×1 convolutions, we could reduce the number of parameters by the same amount. More details about energy distribution is given in Appendix A.2. (2) Feature map combinations may not be equally worth. We could constrain the 1×1 convolutional layer by preventing some scales or orientations from wiring together, or by separating the real and imaginary coefficients.

So far we trained our models with a predefined CMF, but other filters may provide better extraction properties due to a higher frequency localization or number of vanishing moments. One way to address this question could be to let the network learn the optimal filter with a proper regularizer in the loss function. This will be addressed in future work.

We tested our framework on the first layer of AlexNet, because introducing wavelet packet transform into it is quite straightforward. However, the phenomenon of oscillating patterns does not restrict to this particular model, nor is it specific to the sole task of image classification. However, extending our models to a wider range of architectures must be handled carefully to match the desired hyperparameters. This will be tackled in future work, in which we will also benchmark our results with other wavelet CNN approaches.

REFERENCES

Ilker Bayram and Ivan W. Selesnick. On the dual-tree complex wavelet packet and M-band transforms. *IEEE Transactions on Signal Processing*, 2008.

Y-Lan Boureau, Francis Bach, Yann LeCun, and Jean Ponce. Learning mid-level features for recognition. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2559–2566, 2010.

Joan Bruna and Stéphane Mallat. Invariant Scattering Convolution Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1872–1886, 2013.

Shuo-Yiin Chang and Nelson Morgan. Robust CNN-based speech recognition with Gabor filter kernels. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

Shin Fujieda, Kohei Takayama, and Toshiya Hachisuka. Wavelet Convolutional Neural Networks for Texture Classification. 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

Ke Huang and Selin Aviyente. Wavelet feature selection for image classification. *IEEE Transactions on Image Processing*, 2008.

Nick Kingsbury. Complex wavelets for shift invariant analysis and filtering of signals. *Applied and computational harmonic analysis*, 10(3):234–253, 2001.

Nick Kingsbury. Design of Q-shift complex wavelets for image processing using frequency domain energy minimization. In *Proceedings International Conference on Image Processing*, volume 1, pp. I–1013, 2003.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

Andrew Laine and Jian Fan. Texture Classification by Wavelet Packet Signatures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1993.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

Min Lin, Qiang Chen, and Shuicheng Yan. Network In Network. *arXiv:1312.4400 [cs]*, 2014.

Hongya Lu, Haifeng Wang, Qianqian Zhang, Daehan Won, and Sang Won Yoon. A Dual-Tree Complex Wavelet Transform Based Convolutional Neural Network for Human Thyroid Medical Image Segmentation. In *IEEE International Conference on Healthcare Informatics (ICHI)*, pp. 191–198, 2018.

Shangzhen Luan, Chen Chen, Baochang Zhang, Jungong Han, and Jianzhuang Liu. Gabor Convolutional Networks. *IEEE Transactions on Image Processing*, 27(9):4357–4366, 2018.

Stéphane Mallat. *A Wavelet Tour of Signal Processing : The Sparse Way*. Elsevier/Academic Press, 2009.

Edouard Oyallon, Eugene Belilovsky, Sergey Zagoruyko, and Michal Valko. Compressing the Input for CNNs with the First-Order Scattering Transform. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 301–316, 2018.

Stefan Pittner and Sagar V. Kamarthi. Feature extraction from wavelet coefficients for pattern recognition tasks. *IEEE Transactions on pattern analysis and machine intelligence*, 21(1):83–88, 1999.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Syed Shakib Sarwar, Priyadarshini Panda, and Kaushik Roy. Gabor filter assisted energy efficient fast learning Convolutional Neural Networks. In *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 1–6, 2017.

Ivan W. Selesnick, Richard Baraniuk, and Nick Kingsbury. The dual-tree complex wavelet transform. *IEEE Signal Processing Magazine*, 22(6):123–151, 2005.

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*, 2015.

Amarjot Singh and Nick Kingsbury. Dual-Tree wavelet scattering network with parametric log transformation for object classification. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2017.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015.

Matej Ulicny, Vladimir A. Krylov, and Rozenn Dahyot. Harmonic Networks for Image Classification. In *British Machine Vision Conference*, pp. 202, 2019.

Travis Williams and Robert Li. Advanced Image Classification Using Wavelets and Convolutional Neural Networks. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 233–239, 2016.

Travis Williams and Robert Li. Wavelet Pooling for Convolutional Neural Networks. In *International Conference on Learning Representations*, 2018.

Gary G. Yen. Wavelet packet feature extraction for vibration monitoring. *IEEE Transactions on Industrial Electronics*, 2000.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pp. 3320–3328, 2014.

A APPENDIX

A.1 ILLUSTRATION OF WPT AND DT-$\mathbb{C}$WPT

Figures 5 and 6 illustrate WPT and DT-$\mathbb{C}$WPT, respectively. We chose an image from our custom ImageNet test set (one channel only), resized and cropped to 224×224 pixels. We can notice that specific orientations tend to be selected by specific filters, especially in the dual-tree transform. Moreover, some filters seem to extract features of higher energy than others.

A.2 ENERGY DISTRIBUTION OVER FEATURE MAPS

Figure 7 displays the mean energies of the 30 feature maps of high-frequency DT-$\mathbb{C}$WPT coefficients, computed over our custom ImageNet test set. As we can see, the energy distribution is very unbalanced over the different filters.

A.3 IMPLEMENTATION OF THE DT-($\mathbb{R}\mathbb{C}$)WPT MODULES

In this section we show that the dual-tree transforms can be written as a succession of CNN-style convolution operators. We focus on DT-$\mathbb{C}$WPT but it can be easily adapted to DT-$\mathbb{R}$WPT.

The first step is to duplicate each input image four times (one for each filter bank). Given an input tensor $\mathbf{X} \in \mathbb{R}^{P \times 3 \times 224 \times 224}$, this operation can be written as a CNN-style 1×1 convolution operator:

$$\mathbf{X}_0 = \mathcal{C}_{1,\,1}^{(1)}(\mathbf{V}_{\mathrm{dupl}},\, \mathbf{0}) \cdot \mathbf{X}\,, \tag{13}$$

where $\mathbf{X}_0 \in \mathbb{R}^{P \times 12 \times 224 \times 224}$ and $\mathbf{V}_{\mathrm{dupl}} = \begin{pmatrix} \mathbf{I} & \mathbf{I} & \mathbf{I} & \mathbf{I} \end{pmatrix} \in \mathbb{R}^{3 \times 12 \times 1 \times 1}$, with $\mathbf{I} \in \mathbb{R}^{3 \times 3 \times 1 \times 1}$ such that $\mathbf{I}[k,\, l,\, 0,\, 0] = \begin{cases} 1 \text{ if } k = l; \\ 0 \text{ otherwise.} \end{cases}$

Then each level of filter bank decomposition can be summarized into a single CNN-style convolution operator $\mathcal{C}_{s,\,d}^{(q_j)}(\mathbf{W}_j,\, \mathbf{0})$, with $(s = 2)$, $(d = 1)$ and $(q_j = K_j)$, where $K_j = (3 \cdot 4^{(j+1)})$ denotes the number of input channels. For any $j \in \{0,\, 1\}$, the weight tensor $\mathbf{W}_j \in \mathbb{R}^{1 \times (4K_j) \times \mu \times \mu}$ has the following structure:

$$\mathbf{W}_j[0] = \begin{pmatrix} \mathbf{W}_{\mathrm{a},j}[0] \\ \mathbf{W}_{\mathrm{b},j}[0] \\ \mathbf{W}_{\mathrm{c},j}[0] \\ \mathbf{W}_{\mathrm{d},j}[0] \end{pmatrix}\,, \tag{14}$$

where $\mathbf{W}_{\mathrm{a},j}$, $\mathbf{W}_{\mathrm{b},j}$, $\mathbf{W}_{\mathrm{c},j}$ and $\mathbf{W}_{\mathrm{d},j}$ are built similarly to the WPT module, using filter banks $\left\{\boldsymbol{G}_{\mathrm{a}}^{(l)}\right\}$, $\left\{\boldsymbol{G}_{\mathrm{b}}^{(l)}\right\}$, $\left\{\boldsymbol{G}_{\mathrm{c}}^{(l)}\right\}$ and $\left\{\boldsymbol{G}_{\mathrm{d}}^{(l)}\right\}$, respectively. By denoting $\mathbf{D}$ the output of this stage, we get

$$\mathbf{D} = \mathcal{C}_{2,\,1}^{(48)}(\mathbf{W}_1,\, \mathbf{0}) \cdot \left(\mathcal{C}_{2,\,1}^{(12)}(\mathbf{W}_0,\, \mathbf{0}) \cdot \mathbf{X}_0\right)\,. \tag{15}$$

Remark. Note that we have, for all sample $p \in \{0\,..\,P-1\}$,

$$\mathbf{D}[p] = \begin{pmatrix} \mathbf{D}_{\mathrm{a}}[p] \\ \mathbf{D}_{\mathrm{b}}[p] \\ \mathbf{D}_{\mathrm{c}}[p] \\ \mathbf{D}_{\mathrm{d}}[p] \end{pmatrix}\,. \tag{16}$$

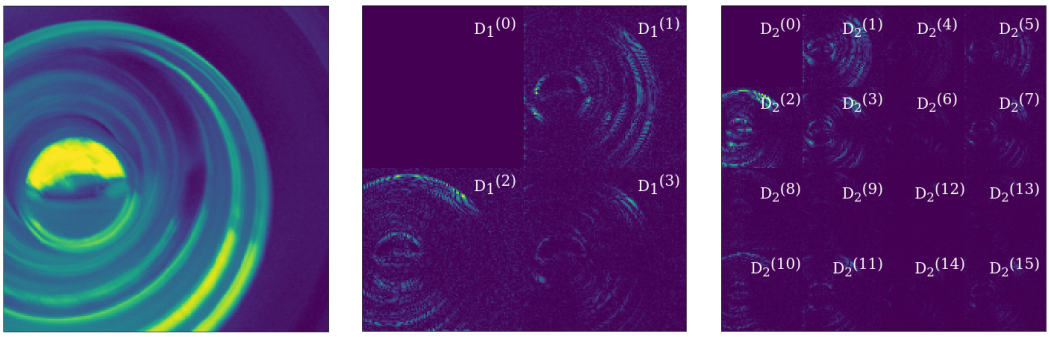

Figure 5: Left: original image from ImageNet2012 ($j = 0$). Middle: WPT with $j = 1$. Right: WPT with $j = 2$. At each step j, the feature maps of wavelet packet coefficients $\boldsymbol{D}_j^{(k)}$ are further decomposed into four smaller submatrices $\boldsymbol{D}_{j+1}^{(4k+l)}$, for $l \in \{0 \,..\, 3\}$, according to (1). To avoid overexposure, scaling coefficients $\boldsymbol{D}_j^{(0)}$ have been set to 0 (top-left images).

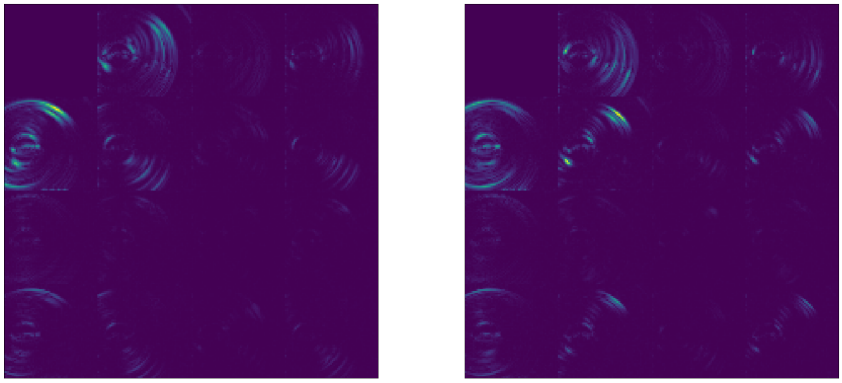

Figure 6: Modulus of the complex DT-$\mathbb{C}$WPT coefficients, computed with $j = 2$: $\left|\boldsymbol{E}_j^{\nearrow(k)}\right|$ (left) and $\left|\boldsymbol{E}_j^{\nwarrow(k)}\right|$ (right), for $k \in \{0 \,..\, 15\}$. Numbering follows the order of Figure 5. To avoid overexposure, scaling coefficients $\boldsymbol{E}_j^{\nearrow(0)}$ and $\boldsymbol{E}_j^{\nwarrow(0)}$ have been set to 0 (top-left images).

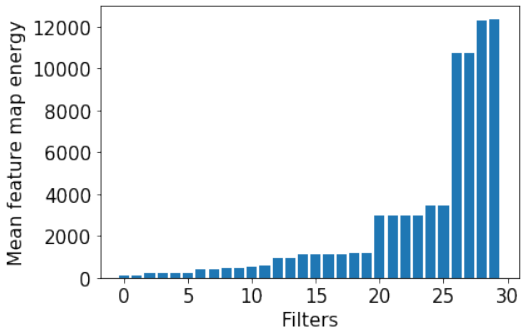

Figure 7: Mean energies of the 30 feature maps of high-frequency DT-$\mathbb{C}$WPT coefficients, computed over our custom ImageNet test set.

Finally, expression (6) is expressed as a CNN-style 1×1 convolution operator:

$$\mathbf{E}_{\mathbb{C}} = \mathcal{C}^{(1)}_{1,\,1}(\mathbf{V}_{\text{combine}},\, \mathbf{0}) \cdot \mathbf{D}\,, \tag{17}$$

where $\mathbf{V}_{\text{combine}} \in \mathbb{R}^{192\times192\times1\times1}$ has the following structure:

$$\mathbf{V}_{\text{combine}} = \begin{pmatrix} \mathbf{I} & \mathbf{O} & \mathbf{O} & -\mathbf{I} \\ \mathbf{O} & \mathbf{I} & \mathbf{I} & \mathbf{O} \\ \mathbf{I} & \mathbf{O} & \mathbf{O} & \mathbf{I} \\ \mathbf{O} & -\mathbf{I} & \mathbf{I} & \mathbf{O} \end{pmatrix}\,, \tag{18}$$

with

- $\mathbf{I} \in \mathbb{R}^{48\times48\times1\times1}$ such that $\mathbf{I}[k,\, l,\, 0,\, 0] = \begin{cases} 1 \text{ if } k = l; \\ 0 \text{ otherwise;} \end{cases}$
- $\mathbf{O} \in \mathbb{R}^{48\times48\times1\times1}$ such that $\mathbf{O}[k,\, l,\, 0,\, 0] = 0$ for all $k,\, l \in \{0\,..\,47\}$.

Note that the two last dimensions of $\mathbf{W}_{\text{combine}}$ have size 1×1; the "convolution" kernels are thus reduced to singleton matrices.

Therefore, expressions (13), (15) and (17) provide a description of DT-$\mathbb{C}$WPT as a succession of CNN-style convolution operators.

A.4 Proof of Proposition 1

Proof. Let $\mathbf{X} \in \mathbb{R}^{P\times K\times N\times N}$ denote an input tensor. Let $\mathbf{Y} \in \mathbb{R}^{P\times L\times M\times M}$ and $\mathbf{Y}' \in \mathbb{R}^{P\times L'\times M'\times M'}$ denote the outputs of the convolution operators, such that

$$\begin{aligned} \mathbf{Y} = \mathcal{C}^{(1)}_{s,\,1}(\mathbf{W},\, \boldsymbol{b}) \cdot \mathbf{X} \quad \text{and} \quad \mathbf{Y}' &= \left(\mathcal{C}^{(q)}_{t,\,1}(\mathbf{V},\, \boldsymbol{a}) \circ \mathcal{C}^{(1)}_{s,\,1}(\mathbf{W},\, \boldsymbol{b})\right) \cdot \mathbf{X} \\ &= \mathcal{C}^{(q)}_{t,\,1}(\mathbf{V},\, \boldsymbol{a}) \left(\mathcal{C}^{(1)}_{s,\,1}(\mathbf{W},\, \boldsymbol{b}) \cdot \mathbf{X}\right) \\ &= \mathcal{C}^{(q)}_{t,\,1}(\mathbf{V},\, \boldsymbol{a}) \cdot \mathbf{Y}\,. \end{aligned} \tag{19}$$

Let $p \in \{0\,..\,P-1\}$. By using (3) and (19), we have, for all $l' \in \{0\,..\,L'-1\}$,

$$\mathbf{Y}'[p,\, l'] = \boldsymbol{a}[l'] + \sum_{l=0}^{L/q-1} \left(\mathbf{Y}\left[p,\, \left\lfloor \frac{l'q}{L'} \right\rfloor \cdot \frac{L}{q} + l\right] * \overline{\mathbf{V}[l,\, l']}\right) \downarrow t\,, \tag{20}$$

and, for all $l \in \{0\,..\,L-1\}$,

$$\mathbf{Y}[p, l] = \boldsymbol{b}[l] + \sum_{k=0}^{K-1} \left(\mathbf{X}[p,\, k] * \overline{\mathbf{W}[k,\, l]}\right) \downarrow s\,. \tag{21}$$

The next steps require the two following lemmas:

Lemma 1. *For all 2D matrices $\boldsymbol{U}$, $\boldsymbol{V}$, and all $b \in \mathbb{R}$,*

$$(b + \boldsymbol{U}) * \boldsymbol{V} = \left(b \cdot \sum_{m,\,n} \boldsymbol{V}[m, n]\right) + (\boldsymbol{U} * \boldsymbol{V})\,. \tag{22}$$

Lemma 2. *For all 2D matrices $\boldsymbol{U}$, $\boldsymbol{V}$, and all integers $s, t \in \mathbb{N}^*$,*

$$((\boldsymbol{U} \downarrow s) * \boldsymbol{V}) \downarrow t = (\boldsymbol{U} * (\boldsymbol{V} \uparrow s)) \downarrow (st)\,, \tag{23}$$

where, as a reminder, $(\boldsymbol{V} \uparrow s)$ denotes the s-dilated matrix.

Then, by plugging (21) into (20) and by using Lemma 1, we get

$$\begin{aligned}\mathbf{Y}'[p, l'] = & \overbrace{\boldsymbol{a}[l'] + \sum_{l=0}^{L/q-1} \left(\boldsymbol{b}\left[\left\lfloor \frac{l'q}{L'} \right\rfloor \cdot \frac{L}{q} + l \right] \cdot \sum_{m,n} \mathbf{V}[l,\, l',\, m,\, n] \right)}^{\boldsymbol{b}'[l']} \\ & + \sum_{l=0}^{L/q-1} \left[\sum_{k=0}^{K-1} \left(\mathbf{X}[p, k] * \overline{\mathbf{W}\left[k,\, \left\lfloor \frac{l'q}{L'} \right\rfloor \cdot \frac{L}{q} + l \right]} \right) \downarrow s * \overline{\mathbf{V}[l, l']} \right] \downarrow t \; ;\end{aligned} \tag{24}$$

and finally, by reversing the 2 sums and using Lemma 2, we get

$$\begin{aligned}\mathbf{Y}'[p, l'] &= \boldsymbol{b}'[l'] + \sum_{k=0}^{K-1} \left(\mathbf{X}[p, k] * \overbrace{\sum_{l=0}^{L/q-1} \left(\overline{\mathbf{W}\left[k,\, \left\lfloor \frac{l'q}{L'} \right\rfloor \cdot \frac{L}{q} + l \right]} * \overline{\mathbf{V}[l, l'] \uparrow s} \right)}^{\overline{\mathbf{W}'[k,l']}} \right) \downarrow (st) \; ; \\ &= \boldsymbol{b}'[l'] + \sum_{k=0}^{K-1} \left(\mathbf{X}[p, k] * \overline{\mathbf{W}'[k, l']} \right) \downarrow (st) \; .\end{aligned} \tag{25}$$

Hence

$$\mathbf{Y}' = \mathcal{C}^{(1)}_{(st),\,1}(\mathbf{W}',\, \boldsymbol{b}') \cdot \mathbf{X} \tag{26}$$

for all $\mathbf{X}$, which proves (8).

By applying the definition of a CNN-style convolution operator (3) to the definition of $\mathbf{W}'[k, l']$ in expression (25), we get

$$\overline{\mathbf{W}'} = \mathcal{C}^{(q)}_{1,\,s}(\mathbf{V},\, \mathbf{0}) \cdot \overline{\mathbf{W}} \; , \tag{27}$$

which proves (9).

Note that the expression of $\boldsymbol{b}'[l']$ defined in (24) can be rewritten in a more concise way:

$$\boldsymbol{b}'[l'] = \boldsymbol{a}[l'] + \left\langle \mathcal{P}^{(q)}_{\lfloor l'q/L' \rfloor} \boldsymbol{b},\; \mathcal{S}_{l'} \mathbf{V} \right\rangle \; , \tag{28}$$

where

- $\left\{ \mathcal{P}^{(q)}_{\gamma} \boldsymbol{b} \right\}_{\gamma \in \{0..q-1\}}$ denotes a partition of even-size slices of $\boldsymbol{b}$. For any $\gamma \in \{0\,..\,q-1\}$,

$$\mathcal{P}^{(q)}_{\gamma} \boldsymbol{b} = \boldsymbol{b}\left[\frac{\gamma K}{q} : \frac{(\gamma+1)K}{q} - 1 \right] \; ; \tag{29}$$

- $\mathcal{S}_l \mathbf{W} \in \mathbb{R}^{K/q}$ denotes the vector such that for any $k \in \{0\,..\,K/q-1\}$,

$$(\mathcal{S}_l \mathbf{W})[k] = \sum_{m,n} \mathbf{W}[k,\, l,\, m,\, n] \; . \tag{30}$$

Let's denote $f(\mathbf{V},\, \boldsymbol{b}) \in \mathbb{R}^{L'}$ such that $f(\mathbf{V},\, \boldsymbol{b})[l'] = \left\langle \mathcal{P}^{(q)}_{\lfloor l'q/L' \rfloor} \boldsymbol{b},\; \mathcal{S}_{l'} \mathbf{W} \right\rangle$. Then we get $f(\mathbf{V},\, \mathbf{0}) = \mathbf{0}$, as stated in Proposition 1.

□

Remark. Operators $\mathcal{P}^{(q)}_{\gamma}$ and $\mathcal{S}_l$ have the advantage of being efficiently computed with libraries such as PyTorch or even NumPy.

A.5 AN ALGORITHM TO COMPUTE THE RESULTING WEIGHT AND BIAS

Proposition 1 provides an iterative algorithm to compute the weight and bias resulting from a succession of CNN-style convolution operators. A special care must be paid to initialization. The number of groups in the first convolution operator must indeed be equal to 1. In order to meet this requirement, we introduce an identity operator $\mathcal{C}_{1,\,1}^{(1)}(\mathbf{I},\,\mathbf{0})$, where $\mathbf{I} \in \mathbb{R}^{K\times K\times 1\times 1}$ is defined by $\mathbf{I}[k,l,0,0] = \begin{cases} 1 \text{ if } k=l; \\ 0 \text{ otherwise.} \end{cases}$.

Proposition 2 (Identity operator)**.** *Let $K,\,L \in \mathbb{N}^*$ and $t,\,q \in \mathbb{N}^*$ such that both K and L are divisible by q. For all $\mathbf{V} \in \mathbb{R}^{(K/q)\times L\times \mu\times \nu}$ and all $\boldsymbol{a} \in \mathbb{R}^L$,*

$$\mathcal{C}_{t,\,1}^{(q)}(\mathbf{V},\,\boldsymbol{a}) = \left(\mathcal{C}_{t,\,1}^{(q)}(\mathbf{V},\,\boldsymbol{a}) \circ \mathcal{C}_{1,\,1}^{(1)}(\mathbf{I},\,\mathbf{0})\right) \,. \tag{31}$$

Proof. It can be easily proven that for all $\mathbf{X} \in \mathbb{R}^{P\times K\times N\times N}$, $\mathcal{C}_{1,\,1}^{(1)}(\mathbf{I},\,\mathbf{0}) \cdot \mathbf{X} = \mathbf{X}$. □

Therefore Proposition 1 can be used on expression (31) in order to initialize the algorithm. The details are given in Algorithm 1.

Algorithm 1 Composition of convolution operators

Require: $K = L_0$ {number of input channels}
Require: $\{(L_1, t_1, g_1, \mathbf{W}_1, \boldsymbol{b}_1), \ldots, (L_R, t_R, q_r, \mathbf{W}_R, \boldsymbol{b}_R)\}$ {list of output channels, strides, groups, weights and bias}
Ensure: $\forall r \in \{1\,..\,R\}$, both L_{r-1} and L_r are divisible by q_r
Ensure: $\forall r \in \{1\,..\,R\},\ \langle \mathbf{W}_r \rangle = \begin{pmatrix} \frac{L_{r-1}}{q_r} & L_r & \mu_r & \nu_r \end{pmatrix}^\top$ and $|\boldsymbol{b}_r| = L_r$
  $\mathbf{W} \leftarrow \mathbf{I} \in \mathbb{R}^{K\times K\times 1\times 1}$ {identity weight}
  $\boldsymbol{b} \leftarrow \mathbf{0}$ {initial bias}
  $s \leftarrow 1$ {initial stride}
  for $r \in \{1\,..\,R\}$ **do**
    $\mathbf{W} \leftarrow \mathcal{C}_{1,\,s}^{(q_r)}(\mathbf{W}_r,\,\mathbf{0}) \cdot \mathbf{W}$ {resulting weight}
    $\boldsymbol{b} \leftarrow \boldsymbol{b}_r + \left(\left\langle \mathcal{P}_{\lfloor l q_r / L_r \rfloor}^{(q_r)} \boldsymbol{b},\ \mathcal{S}_l \mathbf{W}_r \right\rangle\right)_{l\in\{0..L_r\}}$ {resulting bias}
    $s \leftarrow s \times t_r$ {new stride}
  end for
  $\mathbf{W} \leftarrow \overline{\mathbf{W}}$ {flip weight tensor along its 2 last dimensions}
  return $(s,\,\mathbf{W},\,\boldsymbol{b})$ {resulting stride, weight and bias}