From CNNs to Shift-Invariant Twin Models Based on Complex Wavelets
Supplementary Material

Hubert Leterme*†, Kévin Polisano‡, Valérie Perrier‡, Karteek Alahari†
*Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France
†Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LJK, 38000 Grenoble, France
‡Univ. Grenoble Alpes, CNRS, Grenoble INP, LJK, 38000 Grenoble, France
E-mail: hubert.leterme@unicaen.fr

To avoid confusion with respect to the main paper, sections, figures and tables are identified with capital letters. Moreover, equations are numbered following a lower-case Roman style.

CONTENTS

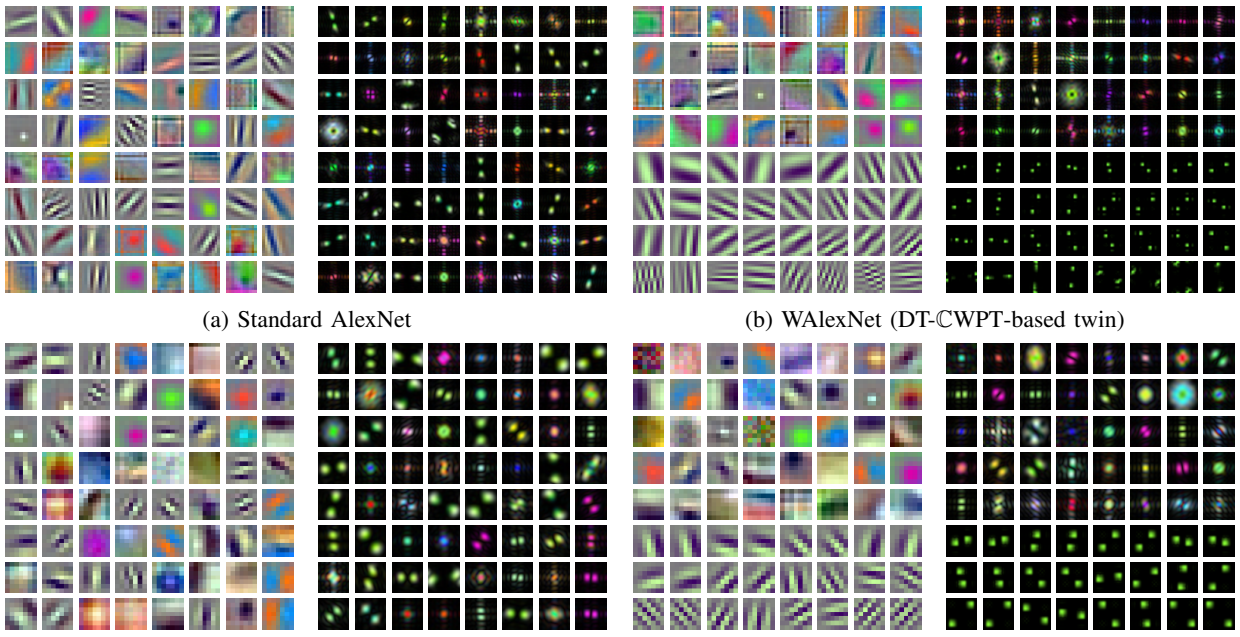
A. DESIGN OF WCNNS: GENERAL ARCHITECTURE

In this section, we provide complements to the description of the mathematical twin (WCNN) introduced in Sections II-C and II-D.

We assume, without loss of generality, that $K = 3$ (RGB input images). The numbers L_{free} and L_{gabor} of freely-trained and Gabor channels are empirically determined from the trained CNNs (see Figs. Aa and Ac). In a twin WCNN architecture, the two groups of output channels are organized such that $\mathcal{F} = \{1 \,..\, L_{\text{free}}\}$ and $\mathcal{G} = \{(L_{\text{free}} + 1) \,..\, L\}$. The first L_{free} channels, which are outside the scope of our approach, remain freely-trained, like in the standard architecture. Regarding the L_{gabor} remaining channels (Gabor channels), the convolution kernels V_{lk} with $l \in \mathcal{G}$ are constrained to satisfy the following requirements. First, all three RGB input channels are processed with the same filter, up to a multiplicative constant. More formally, there exists a *luminance* weight vector $\boldsymbol{\mu} := (\mu_1, \mu_2, \mu_3)^\top$, with $\mu_k \in [0, 1]$ and $\sum_{k=1}^{3} \mu_k = 1$, such that,

$$\forall k \in \{1 \,..\, 3\}\,,\ \mathrm{V}_{lk} = \mu_k \widetilde{\mathrm{V}}_l, \tag{i}$$

where $\widetilde{\mathrm{V}}_l := \sum_{k=1}^{3} \mathrm{V}_{lk}$ denotes the mean kernel. Furthermore, $\widetilde{\mathrm{V}}_l$ must be band-pass and oriented (Gabor-like filter). The following paragraphs explain how these two constraints are implemented in our WCNN architecture.



(a) Standard AlexNet (b) WAlexNet (DT-ℂWPT-based twin)

(c) Standard ResNet-34 (d) WResNet-34 (DT-ℂWPT-based twin)

Fig. A. Convolution kernels $\mathbf{V} \in \mathcal{S}^{64\times 3}$ for the models based on AlexNet and ResNet-34, after training with ImageNet. Each image represents a 3D filter $(\mathrm{V}_{lk})_{k\in\{1..3\}}$, for any output channel $l \in \{1\,..\,64\}$. For our DT-ℂWPT-based twin architecture (Figs. Ab and Ad), the $L_{\text{free}} := 32$ or 40 first kernels are freely-trained, whereas the remaining $L_{\text{gabor}} := 32$ or 24 kernels are constrained to be monochrome, band-pass and oriented. Left: representation in the spatial domain; right: corresponding power spectra.

Monochrome Filters

Expression (i) is actually a property of standard CNNs: the oriented band-pass RGB kernels generally appear monochrome (see kernel visualization of freely-trained CNNs in Figs. Aa and Ac). In WCNNs, this constraint is implemented with a trainable 1×1 convolution layer [1], parameterized by $\boldsymbol{\mu}$, computing the following luminance image:

$$\mathrm{X}^{\text{lum}} := \sum_{k=1}^{3} \mu_k \mathrm{X}_k. \tag{ii}$$

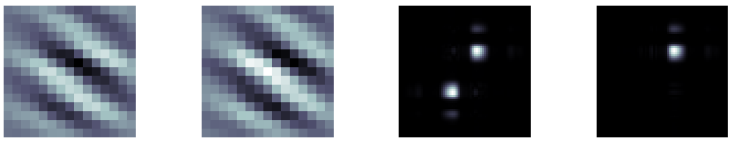
This constraint can be relaxed by authorizing a specific luminance vector $\boldsymbol{\mu}_l$ for each Gabor channel $l \in \mathcal{G}$. Numerical experiments on such models are left for future work.

Gabor-Like Kernels

To guarantee the Gabor-like property on $\widetilde{\mathrm{V}}_l$, we implemented DT-ℂWPT, which is achieved through a series of subsampled convolutions. The number of decomposition stages $J \in \mathbb{N} \setminus \{0\}$ was chosen such that $m = 2^{J-1}$, where, as a reminder, m denotes the subsampling factor as introduced in (5). DT-ℂWPT generates a set of filters $\big(\mathrm{W}^{\text{dt}}_{k'}\big)_{k'\in\{1..4\times 4^J\}}$, which tiles the Fourier domain $[-\pi,\,\pi]^2$ into 4×4^J overlapping square windows. Their real and imaginary parts approximately form a 2D Hilbert transform pair. Figure B illustrates such a convolution filter.

The WCNN architecture is designed such that, for any Gabor channel $l \in \mathcal{G}$, $\widetilde{\mathrm{V}}_l$ is the real part of one such filter:

$$\exists k' \in \big\{1\,..\,4\times 4^J\big\} : \widetilde{\mathrm{V}}_l = \mathrm{Re}\big(\mathrm{W}^{\text{dt}}_{k'}\big). \tag{iii}$$



(a) (b) (c) (d)

Fig. B. (a), (b): Real and imaginary parts of a Gabor-like convolution kernel $\mathrm{W}_{lk} := \mathrm{V}_{lk} + i\mathcal{H}(\mathrm{V}_{lk})$, forming a 2D Hilbert transform pair. (c), (d): Power spectra (energy of the Fourier transform) of V_{lk} and W_{lk}, respectively.

The output Y_l introduced in (5) then becomes

$$\mathrm{Y}_l = \left(\mathrm{X}^{\mathrm{lum}} \star \widetilde{\mathrm{V}}_l\right) \downarrow 2^{J-1}. \tag{iv}$$

To summarize, a WCNN substitutes the freely-trained convolution (5) with a combination of (ii) and (iv), for any Gabor output channels $l \in \mathcal{G}$. This combination is wrapped into a *wavelet block*, also referred to as WBlock. Technical details about its exact design are provided in Section B. Note that the Fourier resolution of V_{lk} increases with the subsampling factor m. This property is consistent with what is observed in freely-trained CNNs: in AlexNet, where $m = 4$, the Gabor-like filters are more localized in frequency (and less spatially localized) than in ResNet, where $m = 2$.

Visual representations of the kernels $\mathbf{V} \in \mathcal{S}^{L \times K}$, with $K = 3$ and $L = 64$, for the WCNN architectures based on AlexNet and ResNet-34, referred to as WAlexNet and WResNet-34, are provided in Figs. Ab and Ad, respectively.

Stabilized WCNNs

Using the principles presented in Section II-B of the main paper, we replace $\mathbb{R}$Max (10) by $\mathbb{C}$Mod (12) for all Gabor channels $l \in \mathcal{G}$. In the corresponding model, referred to as $\mathbb{C}$WCNN, the wavelet block is replaced by a *complex wavelet block* ($\mathbb{C}$WBlock), in which (iv) becomes

$$\mathrm{Z}_l = \left(\mathrm{X}^{\mathrm{lum}} \star \widetilde{\mathrm{W}}_l\right) \downarrow 2^{J}, \tag{v}$$

where $\widetilde{\mathrm{W}}_l$ is obtained by considering both real and imaginary parts of the DT-$\mathbb{C}$WPT filter:

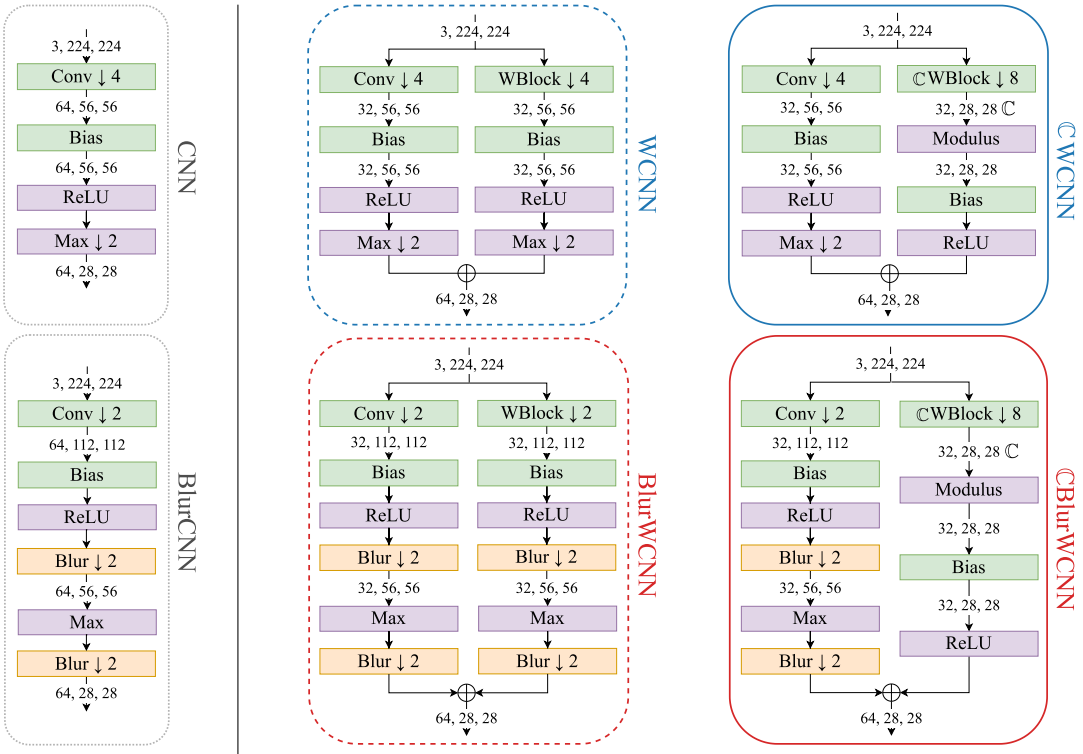
$$\widetilde{\mathrm{W}}_l := \mathrm{W}^{\mathrm{dt}}_{k'}, \tag{vi}$$

where k' has been introduced in (iii). Then, a modulus operator is applied to Z_l, which yields $\mathrm{Y}^{\mathrm{mod}}_l$ such as defined in (12), with $\mathrm{W}_{lk} := \mu_k \widetilde{\mathrm{W}}_l$ for any RGB channel $k \in \{1\,..\,3\}$. Finally, we apply a bias and ReLU to $\mathrm{Y}^{\mathrm{mod}}_l$, following (11).

A schematic representation of WAlexNet and its stabilized version, referred to as $\mathbb{C}$WAlexNet, is provided in Fig. C (top part). Following Section II-D, the WCNN and $\mathbb{C}$WCNN architectures built upon blurpooled AlexNet, referred to as BlurWAlexNet and $\mathbb{C}$BlurWAlexNet, respectively, are represented in the same figure (bottom part). Note that, for a fair comparison, all three models use blur pooling in the freely-trained channels as well as deeper layers; only the Gabor channels are modified.

B. Filter Selection and Sparse Regularization

We explained that, for each Gabor channel $l \in \mathcal{G}$, the average kernel $\widetilde{\mathrm{V}}_l$ is the real part of a DT-$\mathbb{C}$WPT filter, as written in (iii). We now explain how the filter selection is done; in other words, how k' is chosen among $\left\{1\,..\,4 \times 4^J\right\}$. Since input images are real-valued, we restrict to the filters with bandwidth located in the half-plane of positive x-values. For the sake of concision, we denote by $K_{\mathrm{dt}} := 2 \times 4^J$ the number of such filters.



(a) AlexNet (b) WAlexNet (baseline) (c) ℂ**WAlexNet** (proposed approach)

Fig. C. First layers of AlexNet and its variants, corresponding to a convolution layer followed by ReLU and max pooling (1). The models are framed according to the same colors and line styles as in Fig. 1 (main paper). The green modules are the ones containing trainable parameters; the orange and purple modules represent static linear and nonlinear operators, respectively. The numbers between each module represent the depth (number of channels), height and width of each output. Fig. Ca: freely-trained models. Top: standard AlexNet. Bottom: Zhang's "blurpooled" AlexNet. Fig. Cb: mathematical twins (WAlexNet) reproducing the behavior of standard (top) and blurpooled (bottom) AlexNet. The left side of each diagram corresponds to the $L_{\text{free}} := 32$ freely-trained output channels, whereas the right side displays the $L_{\text{gabor}} := 32$ remaining channels, where freely-trained convolutions have been replaced by a wavelet block (WBlock) as described in Section A. Fig. Cc: ℂMod-based WAlexNet, where WBlock has been replaced by ℂWBlock, and max pooling by a modulus. The bias and ReLU are placed after the modulus, following (2). In the bottom models, we compare Zhang's antialiasing approach (Fig. Cb) with ours (Fig. Cc) in the Gabor channels.

For any RGB image $\mathbf{X} \in \mathcal{S}^3$, a luminance image $\mathrm{X}^{\mathrm{lum}} \in \mathcal{S}$ is computed following (ii), using a 1×1 convolution layer. Then, DT-ℂWPT is performed on $\mathrm{X}^{\mathrm{lum}}$. We denote by $\mathbf{D} := (\mathrm{D}_k)_{k \in \{1..K_{\mathrm{dt}}\}}$ the tensor containing the real part of the DT-ℂWPT feature maps:

$$\mathrm{D}_k = \left(\mathrm{X}^{\mathrm{lum}} \star \mathrm{Re}\, \mathrm{W}_k^{(J)}\right) \downarrow 2^{J-1}. \tag{vii}$$

For the sake of computational efficiency, DT-ℂWPT is performed with a succession of subsampled separable convolutions and linear combinations of real-valued wavelet packet feature maps [2]. To match the subsampling factor $m := 2^{J-1}$ of the standard model, the last decomposition stage is performed without subsampling.

Filter Selection

The number of dual-tree feature maps K_{dt} may be greater than the number of Gabor channels L_{gabor}. In that case, we therefore want to select filters that contribute the most to the network's predictive power.

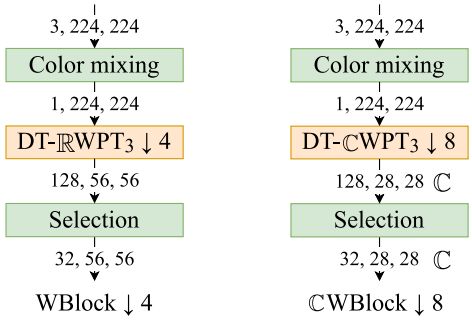


Fig. D. Detail of a wavelet block with $J = 3$ as in AlexNet, in its $\mathbb{R}$Max (left) and $\mathbb{C}$Mod (right) versions. DT-$\mathbb{R}$WPT corresponds to the real part of DT-$\mathbb{C}$WPT.

First, the low-frequency feature maps D_0 and $\mathrm{D}_{(4^J+1)}$ are discarded. Then, a subset of $K'_{\mathrm{dt}} < K_{\mathrm{dt}}$ feature maps is manually selected and permuted in order to form clusters in the Fourier domain. Considering a (truncated) permutation matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{K'_{\mathrm{dt}} \times K_{\mathrm{dt}}}$, the output of this transformation, denoted by $\mathbf{D}' \in \mathcal{S}^{K'_{\mathrm{dt}}}$, is defined by:

$$\mathbf{D}' := \boldsymbol{\Sigma}\, \mathbf{D}. \tag{viii}$$

The feature maps $\mathbf{D}'$ are then sliced into Q groups of channels $\mathbf{D}^{(q)} \in \mathcal{S}^{K_q}$, each of them corresponding to a cluster of band-pass dual-tree filters with neighboring frequencies and orientations. On the other hand, the output of the wavelet block, $\mathbf{Y}^{\mathrm{gabor}} := (\mathrm{Y}_l)_{l \in \{L_{\mathrm{free}}+1..L\}} \in \mathcal{S}^{L_{\mathrm{gabor}}}$, where Y_l has been introduced in (5), is also sliced into Q groups of channels $\mathbf{Y}^{(q)} \in \mathcal{S}^{L_q}$. Then, for each group $q \in \{1\,..\,Q\}$, an affine mapping between $\mathbf{D}^{(q)}$ and $\mathbf{Y}^{(q)}$ is performed. It is characterized by a trainable matrix $\boldsymbol{A}^{(q)} := \left(\boldsymbol{\alpha}_1^{(q)}, \cdots, \boldsymbol{\alpha}_{L_q}^{(q)}\right)^\top \in \mathbb{R}^{L_q \times K_q}$ such that, for any $l \in \{1\,..\,L_q\}$,

$$\mathrm{Y}_l^{(q)} := \boldsymbol{\alpha}_l^{(q)\top} \cdot \mathbf{D}^{(q)}. \tag{ix}$$

As in the color mixing stage, this operation is implemented as a 1×1 convolution layer.

A schematic representation of the real- and complex-valued wavelet blocks can be found in Fig. D.

Sparse Regularization

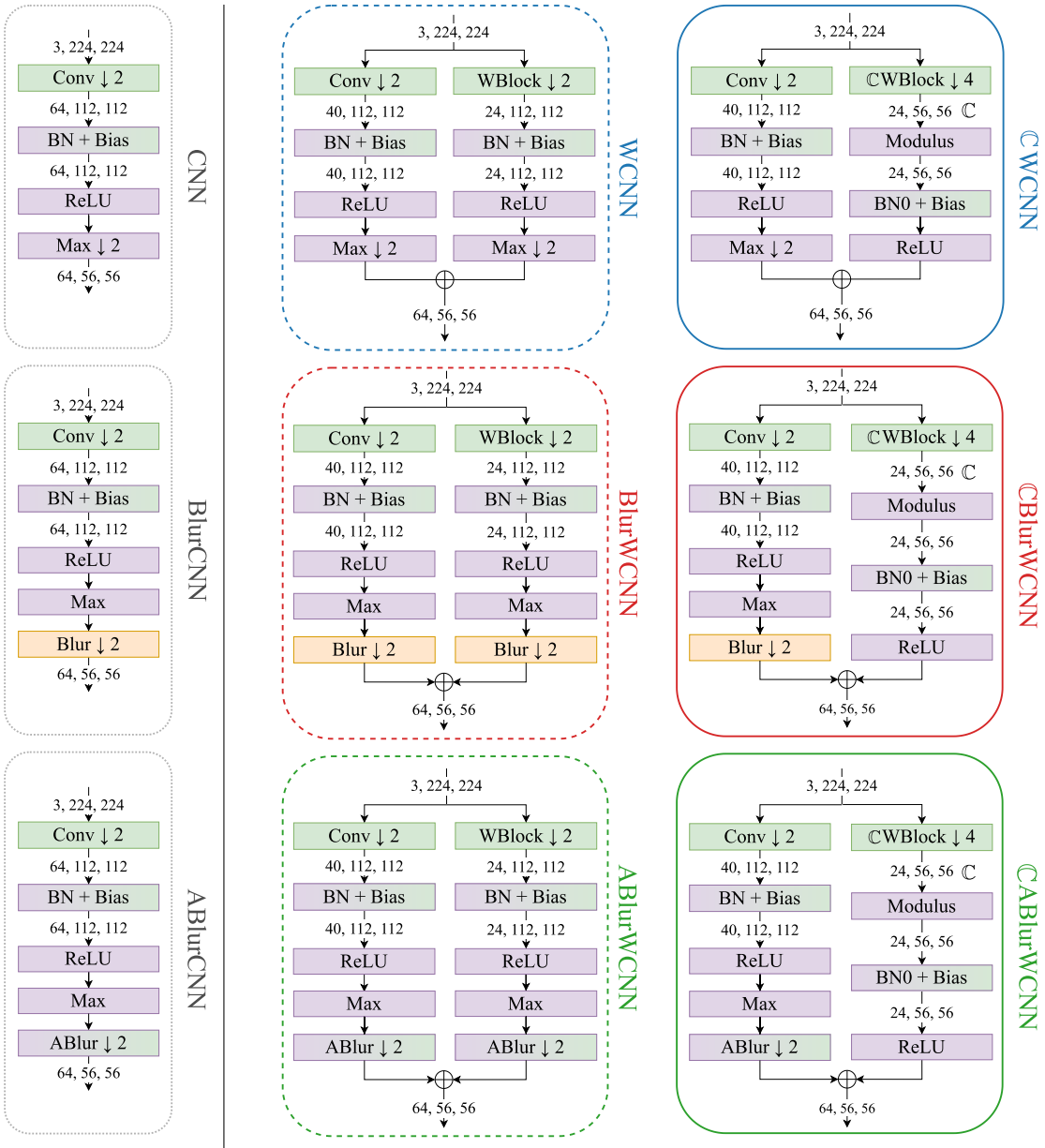
For any group $q \in \{1\,..\,Q\}$ and output channel $l \in \{1\,..\,L_q\}$, we want the model to select one and only one wavelet packet feature map within the q-th group. In other words, each row vector $\boldsymbol{\alpha}_l^{(q)} := \left(\alpha_{l,\,1}^{(q)}, \cdots, \alpha_{l,\,K_q}^{(q)}\right)^\top$ of $\boldsymbol{A}^{(q)}$ contains no more than one nonzero element, such that (ix) becomes

$$\mathrm{Y}_l^{(q)} = \alpha_{lk}^{(q)} \mathrm{D}_k^{(q)} \tag{x}$$

for some (unknown) value of $k \in \{1\,..\,K_q\}$. To enforce this property during training, we add a mixed-norm l^1/l^∞-regularizer [3] to the loss function to penalize non-sparse feature map mixing as follows:

$$\mathcal{L} := \mathcal{L}_0 + \sum_{q=1}^{Q} \lambda_q \sum_{l=1}^{L_q} \left(\frac{\left\|\boldsymbol{\alpha}_l^{(q)}\right\|_1}{\left\|\boldsymbol{\alpha}_l^{(q)}\right\|_\infty} - 1 \right), \tag{xi}$$

where $\mathcal{L}_0$ denotes the standard cross-entropy loss and $\boldsymbol{\lambda} \in \mathbb{R}^Q$ denotes a vector of regularization hyperparameters. Note that the unit bias in (xi) serves for interpretability of the regularized loss ($\mathcal{L} = \mathcal{L}_0$ in the desired configuration) but has no impact on training.



(a) ResNet (b) WResNet (baseline) (c) ℂWResNet (proposed approach)

Fig. E. First layers of ResNet and its variants, corresponding to a convolution layer followed by ReLU and max pooling. The bias module from Fig. C has been replaced by an affine batch normalization layer ("BN → Bias", or "BN0 → Bias" when placed after Modulus—see Section C). Top: ResNet without blur pooling. Middle: Zhang's "blurpooled" models [4]. Bottom: Zou et al.'s approach, using adaptive blur pooling [5].

C. Adaptation to ResNet: Batch Normalization

In many architectures including ResNet, the bias is computed after an operation called *batch normalization* (BN) [6]. In this context, the first layers have the following structure:

$$\text{Conv} \to \text{Sub} \to \text{BN} \to \text{Bias} \to \text{ReLU} \to \text{MaxPool}. \tag{xii}$$

As shown hereafter, the $\mathbb{R}$Max-$\mathbb{C}$Mod substitution yields, analogously to (2),

$$\mathbb{C}\text{Conv} \to \text{Sub} \to \text{Modulus} \to \text{BN0} \to \text{Bias} \to \text{ReLU}, \tag{xiii}$$

where BN0 refers to a special type of batch normalization without mean centering. A schematic representation of the DT-$\mathbb{C}$WPT-based ResNet architecture and its variants is provided in Fig. E.

A BN layer is parameterized by trainable weight and bias vectors, respectively denoted by $\boldsymbol{a}$ and $\boldsymbol{b} \in \mathbb{R}^L$. In the remaining of the section, we consider input images $\mathbf{X}$ as a stack of discrete stochastic processes. Then, expression (6) is replaced by

$$\mathrm{A}_l := \mathrm{MaxPool}\left\{ \mathrm{ReLU}\left(a_l \cdot \frac{\mathrm{Y}_l - \mathbb{E}_m[\mathrm{Y}_l]}{\sqrt{\mathbb{V}_m[\mathrm{Y}_l] + \varepsilon}} + b_l \right) \right\}, \tag{xiv}$$

with Y_l satisfying (5) (output of the first convolution layer). In the above expression, we have introduced $\mathbb{E}_m(\mathrm{Y}_l) \in \mathbb{R}$ and $\mathbb{V}_m(\mathrm{Y}_l) \in \mathbb{R}_+$, which respectively denote the mean expected value and variance of $\mathrm{Y}_l[\boldsymbol{n}]$, for indices $\boldsymbol{n}$ contained in the support of Y_l, denoted by $\mathrm{supp}(\mathrm{Y}_l)$. Let us denote by $N \in \mathbb{N} \setminus \{0\}$ the support size of input images. Therefore, if the filter's support size N_{filt} is much smaller that N, then $\mathrm{supp}(\mathrm{Y}_l)$ is roughly of size N/m. We thus define the above quantities as follows:

$$\mathbb{E}_m[\mathrm{Y}_l] := \frac{m^2}{N^2} \sum_{\boldsymbol{n} \in \mathbb{Z}^2} \mathbb{E}[\mathrm{Y}_l[\boldsymbol{n}]]; \tag{xv}$$

$$\mathbb{V}_m[\mathrm{Y}_l] := \frac{m^2}{N^2} \sum_{\boldsymbol{n} \in \mathbb{Z}^2} \mathbb{V}[\mathrm{Y}_l[\boldsymbol{n}]]. \tag{xvi}$$

In practice, estimators are computed over a minibatch of images, hence the layer's denomination. Besides, $\varepsilon > 0$ is a small constant added to the denominator for numerical stability. For the sake of concision, we now assume that $\boldsymbol{a} = \mathbf{1}$. Extensions to other multiplicative factors is straightforward.

Let $l \in \mathcal{G}$ denote a Gabor channel. Then, recall that Y_l satisfies (iv) (output of the WBlock), with

$$\widetilde{\mathrm{V}}_l := \mathrm{Re}\, \widetilde{\mathrm{W}}_l, \tag{xvii}$$

where $\widetilde{\mathrm{W}}_l$ denotes one of the Gabor-like filters spawned by DT-$\mathbb{C}$WPT. The following proposition states that, if the kernel's bandwidth is small enough, then the output of the convolution layer sums to zero.

Proposition 1: We assume that the Fourier transform of $\widetilde{\mathrm{W}}_l$ is supported in a region of size $\kappa \times \kappa$ which does not contain the origin (Gabor-like filter). If, moreover, $\kappa \leq \frac{2\pi}{m}$, then

$$\sum_{\boldsymbol{n} \in \mathbb{Z}^2} \mathrm{Y}_l[\boldsymbol{n}] = 0. \tag{xviii}$$

Proof: This proposition takes advantage of Shannon's sampling theorem. A similar reasoning can be found in the proof of Theorem 2.9 in [7]. ∎

In practice, the power spectrum of DT-$\mathbb{C}$WPT filters cannot be exactly zero on regions with nonzero measure, since they are finitely supported. However, we can reasonably assume that it is concentrated within a region of size $\pi/2^{J-1} = \pi/m$. Therefore, since we have discarded low-pass filters, the conditions of Proposition 1 are approximately met for $\widetilde{\mathrm{W}}_l$.

We now assume that (xviii) is satisfied. Moreover, we assume that $\mathbb{E}[\mathrm{Y}_l[\boldsymbol{n}]]$ is constant for any $\boldsymbol{n} \in \mathrm{supp}(\mathrm{Y}_l)$. Aside from boundary effects, this is true if $\mathbb{E}[\mathrm{X}^{\mathrm{lum}}[\boldsymbol{n}]]$ is constant for any $\boldsymbol{n} \in \mathrm{supp}(\mathrm{X}^{\mathrm{lum}})$.[1] We then get, for any $\boldsymbol{n} \in \mathbb{Z}^2$, $\mathbb{E}[\mathrm{Y}_l[\boldsymbol{n}]] = 0$. Therefore, interchanging max pooling and ReLU yields the normalized version of (9):

$$\mathrm{A}_l^{\max} = \mathrm{ReLU}\left(\frac{\mathrm{Y}_l^{\max}}{\sqrt{\mathbb{E}_m[\mathrm{Y}_l^2] + \varepsilon}} + b_l\right). \tag{xix}$$

As in Section II-B, we replace $\mathrm{Y}_l^{\max}$ by $\mathrm{Y}_l^{\mathrm{mod}}$ for any Gabor channel $l \in \mathcal{G}$, which yields the normalized version of (11):

$$\mathrm{A}_l^{\mathrm{mod}} := \mathrm{ReLU}\left(\frac{\mathrm{Y}_l^{\mathrm{mod}}}{\sqrt{\mathbb{E}_m[\mathrm{Y}_l^2] + \varepsilon}} + b_l\right). \tag{xx}$$

Implementing (xx) as a deep learning architecture is cumbersome because Y_l needs to be explicitly computed and kept in memory, in addition to $\mathrm{Y}_l^{\mathrm{mod}}$. Instead, we want to express the second-order moment $\mathbb{E}_m[\mathrm{Y}_l^2]$ (in the denominator) as a function of $\mathrm{Y}_l^{\mathrm{mod}}$. To this end, we state the following proposition.

Proposition 2: If we restrict the conditions of Proposition 1 to $\kappa \leq \pi/m$, we have

$$\|\mathrm{Y}_l\|_2^2 = 2\left\|\mathrm{Y}_l^{\mathrm{mod}}\right\|_2^2. \tag{xxi}$$

Proof: This result, once again, takes advantage of Shannon's sampling theorem. The proof of our Proposition 2.10 in [7] is based on similar arguments. ■

As for Proposition 1, the conditions of Proposition 2 are approximately met. We therefore assume that (xxi) is satisfied, and (xx) becomes

$$\mathrm{A}_l^{\mathrm{mod}} := \mathrm{ReLU}\left(\frac{\mathrm{Y}_l^{\mathrm{mod}}}{\sqrt{\frac{1}{2}\mathbb{E}_{2m}[{\mathrm{Y}_l^{\mathrm{mod}}}^2] + \varepsilon}} + b_l\right). \tag{xxii}$$

In the case of ResNet, the bias layer (Bias) is therefore preceded by a batch normalization layer without mean centering satisfying (xxii), which we call BN0. The second-order moment of $\mathrm{Y}_l^{\mathrm{mod}}$ is computed on feature maps which are twice smaller than Y_l in both directions (hence the index "$2m$" in (xxii)), which is the subsampling factor for the $\mathbb{C}$Mod operator.

D. Implementation Details

In this section, we provide further information that complements the experimental details presented in Section III-A of the main paper.

Subsampling Factor and Decomposition Depth

As explained in Section II-C, the decomposition depth J is chosen such that $m = 2^{J-1}$ (subsampling factor). Since $m = 4$ in AlexNet and 2 in ResNet, we get $J = 3$ and 2, respectively (see Table A). Therefore, the number of dual-tree filters $K_{\mathrm{dt}} := 2 \times 4^J$ is equal to 128 and 32, respectively.

[1]This property is a rough approximation for images of natural scenes or man-made objects. In practice, the main subject is generally located at the center, the sky at the top, *etc*. These are sources of variability for color and luminance distributions across images, as discussed by Torralba and Oliva [8].

	WAlexNet	**WResNet**
m (subsampling factor)	4	2
J (decomposition depth)	3	2
L_{free}, L_{gabor} (output channels)	32, 32	40, 24

TABLE A. Experimental settings for our WCNN twin models.

Number of Freely-Trained and Gabor Channels

The split L_{free}-L_{gabor} between the freely-trained and Gabor channels, provided in the last row of Table A, have been empirically determined from the standard models. More specifically, considering standard AlexNet and ResNet-34 trained on ImageNet (see Figs. Aa and Ac, respectively), we determined the characteristics of each convolution kernel: frequency, orientation, and coherence index (which indicates whether an orientation is clearly defined). This was done by computing the *tensor structure* [9]. Then, by applying proper thresholds, we isolated the Gabor-like kernels from the others, yielding the approximate values of L_{free} and L_{gabor}. Furthermore, this procedure allowed us to draw a rough estimate of the distribution of the Gabor-like filters in the Fourier domain, which was helpful to design the mapping scheme shown in Fig. F, as explained below.

Filter Selection and Grouping

We then manually selected $K'_{\text{dt}} < K_{\text{dt}}$ filters, used in (viii). In particular, we removed the two low-pass filters, which are outside the scope of our theoretical study. Besides, for computational reasons, in WAlexNet we removed 32 "extremely" high-frequency filters which are clearly absent from the standard model (see Fig. Fa). Finally, in WResNet we removed the 14 filters whose bandwidths outreach the boundaries of the Fourier domain $[-\pi, \pi]^2$ (see Fig. Fb). These filters indeed have a poorly-defined orientation, since a small fraction of their energy is located at the far end of the Fourier domain [10, see Fig. 1, "Proposed DT-$\mathbb{C}$WPT"]. Therefore, they somewhat exhibit a checkerboard pattern.[2]

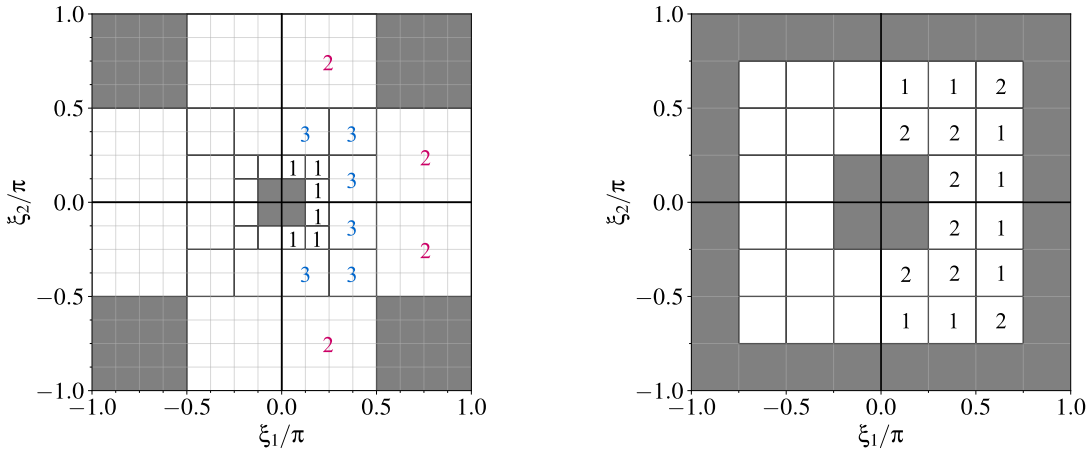
As explained in Section B, once the DT-$\mathbb{C}$WPT feature maps have been manually selected, the output $\mathbf{D}' \in \mathcal{S}^{K'_{\text{dt}}}$ is sliced into Q groups of channels $\mathbf{D}^{(q)} \in \mathcal{S}^{K_q}$. For each group q, a depthwise linear mapping from $\mathbf{D}^{(q)}$ to a bunch of output channels $\mathbf{Y}^{(q)} \in \mathcal{S}^{L_q}$ is performed. Finally, the wavelet block's output feature maps $\mathbf{Y}^{\text{gabor}} \in \mathcal{S}^{L_{\text{gabor}}}$ are obtained by concatenating the outputs $\mathbf{Y}^{(q)}$ depthwise, for any $q \in \{1\,..\,Q\}$. Figure F shows how the above grouping is made, and how many output channels L_q each group q is assigned to.

During training, the above process aims at selecting one single DT-$\mathbb{C}$WPT feature map among each group. This is achieved through mixed-norm l^∞/l^1 regularization, as introduced in (xi). The regularization hyperparameters λ_q have been chosen empirically. If they are too small, then regularization will not be effective. On the contrary, if they are too large, then the regularization term will become predominant, forcing the trainable parameter vector $\boldsymbol{\alpha}_l^{(q)}$ to randomly collapse to 0 except for one element. The chosen values of λ_q are displayed in Table B.

Benchmark against Blur-Pooling-based Approaches

As mentioned in Section II-D, we compare blur-pooling-based antialiasing approach with ours. To apply static or adaptive blur pooling to the WCNNs, we proceed as follows. Following Zhang's implementation, the wavelet block is not antialiased if $m = 2$ as in ResNet, for computational reasons. However, when $m = 4$ as in AlexNet, a blur pooling layer is placed after ReLU, and the wavelet block's subsampling factor is divided by 2. Moreover, max pooling is replaced by max-blur pooling. The size of the blurring filters is set to 3, as recommended by Zhang [4].

[2]Note that the same procedure could have been applied to WAlexNet, but it was deemed unnecessary because the boundary filters were spontaneously discarded during training.



(a) WAlexNet ($J = 3$) (b) WResNet ($J = 2$)

Fig. F. Mapping scheme from DT-$\mathbb{C}$WPT feature maps $\mathbf{D} \in \mathcal{S}^{K_{\mathrm{dt}}}$ to the wavelet block's output $\mathbf{Y}^{\mathrm{gabor}} \in \mathcal{S}^{L_{\mathrm{gabor}}}$. Each wavelet feature map is symbolized by a small square in the Fourier domain, where its energy is mainly located. The gray areas show the feature maps which have been manually removed. Elsewhere, each group of feature maps $\mathbf{D}^{(q)} \in \mathcal{S}^{K_q}$ is symbolized by a dark frame—in (b), K_q is always equal to 1. For each group $q \in \{1\,..\,Q\}$, a number indicates how many output channels L_q are assigned to it. The colored numbers in (a) refer to groups on which we have applied l^∞/l^1-regularization. Note that, when inputs are real-valued, only the half-plane of positive x-values is considered.

Model	Filt. frequency	Reg. param.
WAlexNet	$[\pi/8,\ \pi/4[$	–
	$[\pi/4,\ \pi/2[$	$4.1 \cdot 10^{-3}$
	$[\pi/2,\ \pi[$	$3.2 \cdot 10^{-4}$
WResNet	any	–

TABLE B. Regularization hyperparameters λ_q for each group q of DT-$\mathbb{C}$WPT feature maps. The groups with only one feature map do not need any regularization since this feature map is automatically selected. The second and third rows of WAlexNet correspond to the blue and magenta groups in Fig. Fa, respectively.

E. Accuracy vs Consistency: Additional Plots

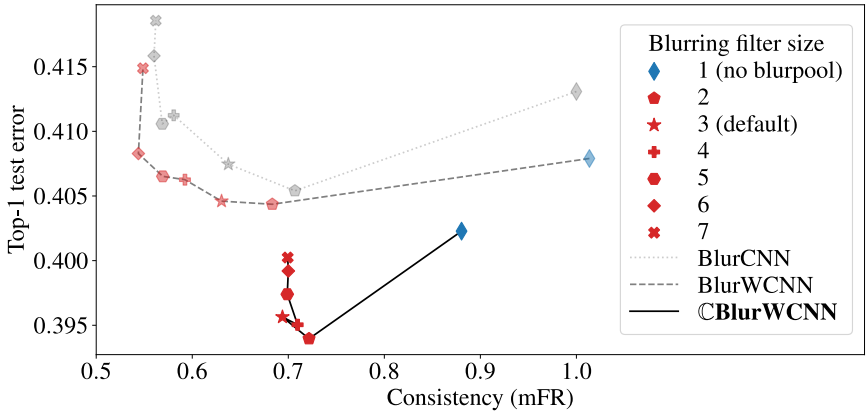
Figure G shows the relationship between consistency and prediction accuracy of AlexNet and ResNet-based models on ImageNet, for different filter sizes ranging from 1 (no blur pooling) to 7 (heavy loss of high-frequency information). The data for AlexNet on the validation set are displayed in the main document, Fig. 2. As recommended by Zhang [4], the optimal trade-off is generally achieved when the blurring filter size is equal to 3. Moreover, in either case, at equivalent level of consistency, replacing blur pooling by our $\mathbb{C}$Mod-based antialiasing approach in the Gabor channels increases accuracy.

F. Computational cost

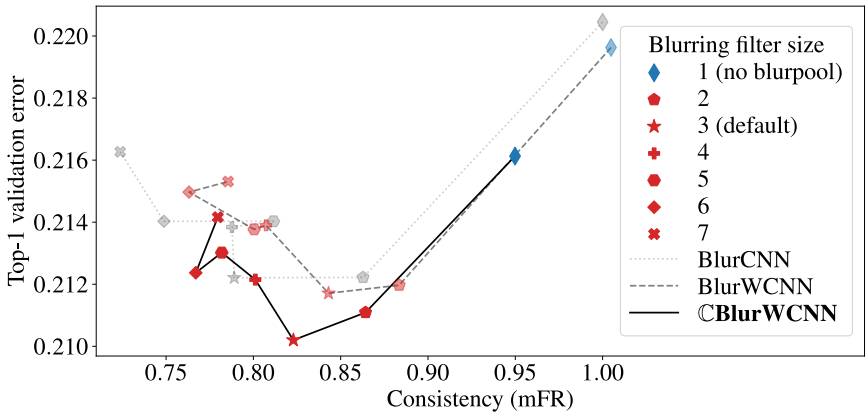
This section provides technical details about our estimation of the computational cost (FLOPs), such as reported in Table III, for *one input image* and *one Gabor channel*. This metric was estimated in the case of standard 2D convolutions.

Average Computation Time per Operation

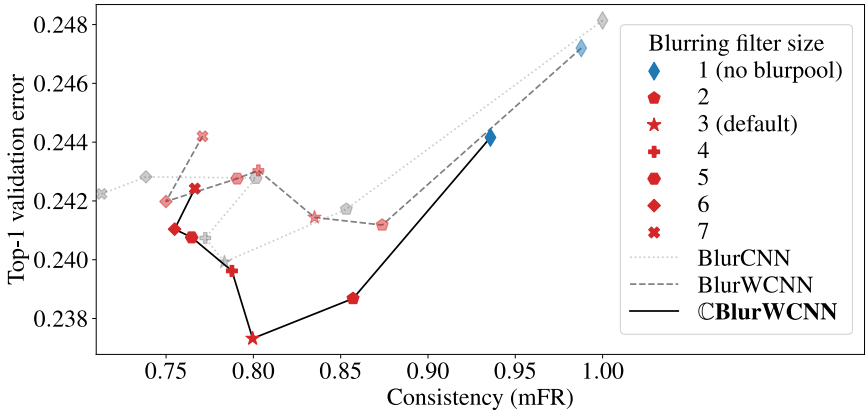
The following values have been determined experimentally using PyTorch (CPU computations). They have been normalized with respect to the computation time of an addition.



(a) AlexNet, test set (50K images)



(b) ResNet-34, validation set (100K images)



(c) ResNet-34, test set (50K images)

Fig. G. Classification accuracy (ten-crops) vs consistency, measuring the stability of predictions to small input shifts (the lower the better for both axes). The metrics have been computed on ImageNet-1K, on both validation set (100K images set aside from the training set) and test set (50K images provided as a separate dataset). For each model (BlurCNN, BlurWCNN and ℂBlurWCNN), we increased the blurring filter size from 1 (*i.e.*, no blur pooling) to 7. The blue diamonds (no blur pooling) and red stars (blur pooling with filters of size 3) correspond to the models for which evaluation metrics have been reported in Table I (models trained after 90 epochs).

$$
\begin{aligned}
t_{\mathrm{s}} &= 1.0 \quad \text{(addition);} \\
t_{\mathrm{p}} &= 1.0 \quad \text{(multiplication);} \\
t_{\mathrm{e}} &= 0.75 \quad \text{(exponential);} \\
t_{\mathrm{mod}} &= 3.5 \quad \text{(modulus);} \\
t_{\mathrm{relu}} &= 0.75 \quad \text{(ReLU);} \\
t_{\mathrm{max}} &= 12.0 \quad \text{(max pooling).}
\end{aligned}
$$

Computational Cost per Layer

In the following paragraphs, $L \in \mathbb{N} \setminus \{0\}$ denotes the number of output channels (depth) and $N' \in \mathbb{N} \setminus \{0\}$ denotes the size of output feature maps (height and width). However, note that N' is not necessary the same for all layers. For instance, in standard ResNet, the output of the first convolution layer is of size $N' = 112$, whereas the output of the subsequent max pooling layer is of size $N' = 56$. For each type of layer, we calculate the number of FLOPs required to produce a single output channel $l \in \{1 \,..\, L\}$. Moreover, we assume, without loss of generality, that the model processes one input image at a time.

a) Convolution Layers: Inputs of size $(K \times N \times N)$ (input channels, height and width); outputs of size $(L \times N' \times N')$. For each output unit, a convolution layer with kernels of size $(N_{\mathrm{filt}} \times N_{\mathrm{filt}})$ requires KN_{filt}^2 multiplications and $KN_{\mathrm{filt}}^2 - 1$ additions. Therefore, the computational cost per output channel is equal to

$$
T_{\mathrm{conv}} = {N'}^2 \left((KN_{\mathrm{filt}}^2 - 1) \cdot t_{\mathrm{s}} + KN_{\mathrm{filt}}^2 \cdot t_{\mathrm{p}} \right). \tag{xxiii}
$$

b) Complex Convolution Layers: Inputs of size $(K \times N \times N)$; complex-valued outputs of size $(L \times N' \times N')$. For each output unit, a complex-valued convolution layer requires $2 \times KN_{\mathrm{filt}}^2$ multiplications and $2 \times (KN_{\mathrm{filt}}^2 - 1)$ additions. Computational cost per output channel:

$$
T_{\mathbb{C}\,\mathrm{conv}} = 2{N'}^2 \left((KN_{\mathrm{filt}}^2 - 1) \cdot t_{\mathrm{s}} + KN_{\mathrm{filt}}^2 \cdot t_{\mathrm{p}} \right). \tag{xxiv}
$$

Note that, in our implementations, the complex-valued convolution layers are less expensive than the real-valued ones, because the output size N' is twice smaller, due to the larger subsampling factor.

c) Bias and ReLU: Inputs and outputs of size $(L \times N' \times N')$. One evaluation for each output unit:

$$
T_{\mathrm{bias}} = {N'}^2 \, t_{\mathrm{s}} \qquad \text{and} \qquad T_{\mathrm{relu}} = {N'}^2 \, t_{\mathrm{relu}}. \tag{xxv}
$$

d) Max Pooling: Outputs of size $(L \times N' \times N')$, with N' depending on whether subsampling is performed at this stage (no subsampling when followed by a blur pooling layer). One evaluation for each output unit:

$$
T_{\mathrm{max}} = {N'}^2 \, t_{\mathrm{max}}. \tag{xxvi}
$$

e) Modulus Pooling: Complex-valued inputs and real-valued outputs of size $(L \times N' \times N')$. One evaluation for each output unit:

$$
T_{\mathrm{mod}} = {N'}^2 \, t_{\mathrm{mod}}. \tag{xxvii}
$$

f) Batch Normalization: Inputs and outputs of size $(L \times N' \times N')$. A batch normalization (BN) layer, described in (xiv), can be split into several stages.

(1) Mean: N'^2 additions.
(2) Standard deviation: N'^2 multiplications, N'^2 additions (second moment), N'^2 additions (subtract squared mean).
(3) Final value: N'^2 additions (subtract mean), $2N'^2$ multiplications (divide by standard deviation and multiplicative coefficient).

Overall, the computational cost per image and output channel of a BN layer is equal to

$$T_{\mathrm{bn}} = N'^2 \left(4\, t_{\mathrm{s}} + 3\, t_{\mathrm{p}}\right). \tag{xxviii}$$

g) Static Blur Pooling: Inputs of size $(L \times 2N' \times 2N')$; outputs of size $(L \times N' \times N')$. For each output unit, a static blur pooling layer [4] with filters of size $(N_{\mathrm{b}} \times N_{\mathrm{b}})$ requires N_{b}^2 multiplications and $N_{\mathrm{b}}^2 - 1$ additions. The computational cost per output channel is therfore equal to

$$T_{\mathrm{blur}} = N'^2 \left((N_{\mathrm{b}}^2 - 1) \cdot t_{\mathrm{s}} + N_{\mathrm{b}}^2 \cdot t_{\mathrm{p}}\right). \tag{xxix}$$

h) Adaptive Blur Pooling: Inputs of size $(L \times 2N' \times 2N')$; outputs of size $(L \times N' \times N')$. An adaptive blur pooling layer [5] with filters of size $(N_{\mathrm{b}} \times N_{\mathrm{b}})$ splits the L output channels into $Q := L/L_{\mathrm{g}}$ groups of L_{g} channels that share the same blurring filters. The adaptive blur pooling layer can be decomposed into the following stages.

(1) Generation of blurring filters using a convolution layer with trainable kernels of size $(N_{\mathrm{b}} \times N_{\mathrm{b}})$: inputs of size $(L \times 2N' \times 2N')$, outputs of size $(QN_{\mathrm{b}}^2 \times N' \times N')$. For each output unit, this stage requires LN_{b}^2 multiplications and $LN_{\mathrm{b}}^2 - 1$ additions. The computational cost divided by the number L of channels is therefore equal to

$$T_{\mathrm{conv\,ablur}} = N'^2 \, \frac{N_{\mathrm{b}}^2}{L_{\mathrm{g}}} \left((LN_{\mathrm{b}}^2 - 1) \cdot t_{\mathrm{s}} + LN_{\mathrm{b}}^2 \cdot t_{\mathrm{p}}\right). \tag{xxx}$$

Note that, despite being expressed on a per-channel basis, the above computational cost depends on the number L of output channels. This is due to the asymptotic complexity of this stage in $O(L^2)$.

(2) Batch normalization, inputs and outputs of size $(QN_{\mathrm{b}}^2 \times N' \times N')$:

$$T_{\mathrm{bn\,ablur}} = N'^2 \, \frac{N_{\mathrm{b}}^2}{L_{\mathrm{g}}} \left(4\, t_{\mathrm{s}} + 3\, t_{\mathrm{p}}\right). \tag{xxxi}$$

(3) Softmax along the depthwise dimension:

$$T_{\mathrm{sftmx\,ablur}} = N'^2 \, \frac{N_{\mathrm{b}}^2}{L_{\mathrm{g}}} (t_{\mathrm{e}} + t_{\mathrm{s}} + t_{\mathrm{p}}). \tag{xxxii}$$

(4) Blur pooling of input feature maps, using the filter generated at stages (1)–(3): inputs of size $(L \times 2N' \times 2N')$, outputs of size $(L \times N' \times N')$. The computational cost per output channel is identical to the static blur pooling layer, even though the weights may vary across channels and spatial locations:

$$T_{\mathrm{blur}} = N'^2 \left((N_{\mathrm{b}}^2 - 1) \cdot t_{\mathrm{s}} + N_{\mathrm{b}}^2 \cdot t_{\mathrm{p}}\right). \tag{xxxiii}$$

Overall, the computational cost of an adaptive blur pooling layer per input image and output channel is equal to

$$T_{\mathrm{ablur}} = N'^2 \, \frac{N_{\mathrm{b}}^2}{L_{\mathrm{g}}} \left[\left((L+1)N_{\mathrm{b}}^2 + 3\right) \cdot t_{\mathrm{s}} + \left((L+1)N_{\mathrm{b}}^2 + 4\right) \cdot t_{\mathrm{p}} + t_{\mathrm{e}}\right]. \tag{xxxiv}$$

We notice that an adaptive blur pooling layer has an asymptotic complexity in $O(N_{\mathrm{b}}^4)$, versus $O(N_{\mathrm{b}}^2)$ for static blur pooling.

Application to AlexNet- and ResNet-based Models

Since they are normalized by the computational cost of standard models, the FLOPs reported in Table III only depend on the size of the convolution kernels and blur pooling filters, respectively denoted by N_{filt} and $N_{\mathrm{b}} \in \mathbb{N} \setminus \{0\}$. In addition, the computational cost of the adaptive blur pooling layer depend on the number of output channels L as well as the number of output channels per group L_{g}.

In practice, N_{filt} is respectively equal to 11 and 7 for AlexNet- and ResNet-based models. Moreover, $N_{\mathrm{b}} = 3$, $L = 64$ and $L_{\mathrm{g}} = 8$. Actually, the computational cost is largely determined by the convolution layers, including step (1) of adaptive blur pooling.

G. Memory Footprint

This section provides technical details about our estimation of the memory footprint for *one input image* and *one output channel*, such as reported in Table III. This metric is generally difficult to estimate, and is very implementation-dependent. Hereafter, we consider the size of the output tensors, as well as intermediate tensors saved by `torch.autograd` for the backward pass. However, we didn't take into account the tensors containing the trainable parameters. To get the size of intermediate tensors, we used the Python package PyTorchViz.[3] These tensors are saved according to the following rules.

- Convolution (Conv), batch normalization (BN), Bias, max pooling (MaxPool or Max), blur pooling (BlurPool), and Modulus: the input tensors are saved, not the output. When Bias follows Conv or BN, no intermediate tensor is saved.
- ReLU, Softmax: the output tensors are saved, not the input.
- If an intermediate tensor is saved at both the output of a layer and the input of the next layer, its memory is not duplicated. An exception is Modulus, which stores the input feature maps as complex numbers.
- MaxPool or Max: a tensor of indices is kept in memory, indicating the position of the maximum values. The tensors are stored as 64-bit integers, so they weight twice as much as conventional float-32 tensors.
- BN: four 1D tensors of length L are kept in memory: computed mean and variance, and running mean and variance. For BN0 (xxii), where the variance is not computed, only two tensors are kept in memory.

In the following paragraphs, we denote by L the number of output channels, N the size of input images (height and width), m the subsampling factor of the baseline models (4 for AlexNet, 2 for ResNet), N_{b} the blurring filter size (set to 3 in practice). For each model, a table contains the size of all saved intermediate or output tensors. For example, the values associated to "Layer1 $\rightarrow$ Layer2" correspond to the depth (number of channel), height and width of the intermediate tensor between Layer1 and Layer2.

AlexNet-based Models

a) No Antialiasing: Conv $\rightarrow$ Bias $\rightarrow$ ReLU $\rightarrow$ MaxPool.

ReLU $\rightarrow$ MaxPool	L	$\frac{N}{m}$	$\frac{N}{m}$
MaxPool $\rightarrow$ *output*	L	$\frac{N}{2m}$	$\frac{N}{2m}$
MaxPool indices ($\times 2$)	L	$\frac{N}{2m}$	$\frac{N}{2m}$

Then, the memory footprint for each output channel is equal to

$$\implies S_{\mathrm{std}} = \frac{7}{4}\frac{N^2}{m^2}.$$

[3] https://github.com/szagoruyko/pytorchviz

b) Static Blur Pooling: Conv → Bias → ReLU → BlurPool → Max → BlurPool.

ReLU → BlurPool	L	$\frac{2N}{m}$	$\frac{2N}{m}$
BlurPool → Max	L	$\frac{N}{m}$	$\frac{N}{m}$
Max → BlurPool	L	$\frac{N}{m}$	$\frac{N}{m}$
Max indices (×2)	L	$\frac{N}{m}$	$\frac{N}{m}$
BlurPool → *output*	L	$\frac{N}{2m}$	$\frac{N}{2m}$

$$\implies S_{\text{blur}} = \frac{33}{4}\frac{N^2}{m^2}.$$

c) ℂMod-based Approach: ℂConv → Modulus → Bias → ReLU.

ℂConv → Modulus	$2L$	$\frac{N}{2m}$	$\frac{N}{2m}$
Modulus → Bias	L	$\frac{N}{2m}$	$\frac{N}{2m}$
ReLU → *output*	L	$\frac{N}{2m}$	$\frac{N}{2m}$

$$\implies S_{\text{mod}} = \frac{N^2}{m^2}.$$

ResNet-based Models

a) No Antialiasing: Conv → BN → Bias → ReLU → MaxPool.

Conv → BN	L	$\frac{N}{m}$	$\frac{N}{m}$
BN metrics	$4L$	–	–
ReLU → MaxPool	L	$\frac{N}{m}$	$\frac{N}{m}$
MaxPool → *output*	L	$\frac{N}{2m}$	$\frac{N}{2m}$
MaxPool indices (×2)	L	$\frac{N}{2m}$	$\frac{N}{2m}$

$$\implies S_{\text{std}} = \frac{11}{4}\frac{N^2}{m^2} + 4 \approx \frac{11}{4}\frac{N^2}{m^2}.$$

b) Static Blur Pooling: Conv → BN → Bias → ReLU → Max → BlurPool.

Conv → BN	L	$\frac{N}{m}$	$\frac{N}{m}$
BN metrics	$4L$	–	–
ReLU → Max	L	$\frac{N}{m}$	$\frac{N}{m}$
Max → BlurPool	L	$\frac{N}{m}$	$\frac{N}{m}$
Max indices (×2)	L	$\frac{N}{m}$	$\frac{N}{m}$
BlurPool → *output*	L	$\frac{N}{2m}$	$\frac{N}{2m}$

$$\implies S_{\text{blur}} = \frac{21}{4}\frac{N^2}{m^2} + 4 \approx \frac{21}{4}\frac{N^2}{m^2}.$$

c) Adaptive Blur Pooling: Conv → BN → Bias → ReLU → Max → ABlurPool.

Conv $\rightarrow$ BN	L	$\frac{N}{m}$	$\frac{N}{m}$
BN metrics	$4L$	–	–
ReLU $\rightarrow$ Max	L	$\frac{N}{m}$	$\frac{N}{m}$
Max $\rightarrow$ ABlurPool	L	$\frac{N}{m}$	$\frac{N}{m}$
Max indices ($\times 2$)	L	$\frac{N}{m}$	$\frac{N}{m}$
ABlurPool $\rightarrow$ *output*	L	$\frac{N}{2m}$	$\frac{N}{2m}$
Generate adaptive blurring filter			
Conv $\rightarrow$ BN $\rightarrow$ Bias $\rightarrow$ Softmax			
Conv $\rightarrow$ BN	$\frac{LN_{\mathrm{b}}^2}{L_{\mathrm{g}}}$	$\frac{N}{2m}$	$\frac{N}{2m}$
BN metrics	$4\frac{LN_{\mathrm{b}}^2}{L_{\mathrm{g}}}$	–	–
Softmax $\rightarrow$ *output*	$\frac{LN_{\mathrm{b}}^2}{L_{\mathrm{g}}}$	$\frac{N}{2m}$	$\frac{N}{2m}$

$$\implies S_{\mathrm{ablur}} = \frac{21}{4}\frac{N^2}{m^2} + 4 + \frac{N_{\mathrm{b}}^2}{L_{\mathrm{g}}}\left(\frac{N^2}{2m^2} + 4\right) \approx \frac{21}{4}\frac{N^2}{m^2} + \frac{N_{\mathrm{b}}^2}{L_{\mathrm{g}}}\frac{N^2}{2m^2}.$$

d) $\mathbb{C}$Mod-based Approach: $\mathbb{C}$Conv $\rightarrow$ Modulus $\rightarrow$ BN0 $\rightarrow$ Bias $\rightarrow$ ReLU.

$\mathbb{C}$Conv $\rightarrow$ Modulus	$2L$	$\frac{N}{2m}$	$\frac{N}{2m}$
Modulus $\rightarrow$ BN0	L	$\frac{N}{2m}$	$\frac{N}{2m}$
BN0 metrics	$2L$	–	–
ReLU $\rightarrow$ *output*	L	$\frac{N}{2m}$	$\frac{N}{2m}$

$$\implies S_{\mathrm{mod}} = \frac{N^2}{m^2} + 2 \approx \frac{N^2}{m^2}.$$

REFERENCES

[1] M. Lin, Q. Chen, and S. Yan, "Network In Network," *arXiv:1312.4400 [cs]*, 2014.

[2] I. W. Selesnick, R. Baraniuk, and N. Kingsbury, "The dual-tree complex wavelet transform," *IEEE Signal Processing Magazine*, vol. 22, no. 6, pp. 123–151, Nov. 2005.

[3] J. Liu and J. Ye, "Efficient L1/Lq Norm Regularization," *arXiv:1009.4766*, Sep. 2010.

[4] R. Zhang, "Making Convolutional Networks Shift-Invariant Again," in *ICML*, 2019.

[5] X. Zou, F. Xiao, Z. Yu, Y. Li, and Y. J. Lee, "Delving Deeper into Anti-Aliasing in ConvNets," *IJCV*, vol. 131, no. 1, pp. 67–81, Jan. 2023.

[6] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proc. 32nd International Conference on Machine Learning*. PMLR, Jun. 2015, pp. 448–456.

[7] H. Leterme, K. Polisano, V. Perrier, and K. Alahari, "On the Shift Invariance of Max Pooling Feature Maps in Convolutional Neural Networks," *arXiv:2104.05704*, Oct. 2023.

[8] A. Torralba and A. Oliva, "Statistics of natural image categories," *Network: Computation in Neural Systems*, vol. 14, no. 3, pp. 391–412, Jan. 2003.

[9] B. Jahne, *Practical Handbook on Image Processing for Scientific and Technical Applications*. CRC Press, 2004.

[10] I. Bayram and I. W. Selesnick, "On the Dual-Tree Complex Wavelet Packet and M-Band Transforms," *IEEE Transactions on Signal Processing*, vol. 56, no. 6, pp. 2298–2310, Jun. 2008.