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

Hubert Leterme, Kévin Polisano, Valérie Perrier, Karteek Alahari

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

Hubert Leterme*†, Kévin Polisano‡, Valérie Perrier‡, and 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

***Abstract*—We propose a novel method to increase shift invariance and prediction accuracy in convolutional neural networks. Specifically, we replace the first-layer combination "real-valued convolutions → max pooling" (RMax) by "complex-valued convolutions → modulus" (CMod), which is stable to translations, or shifts. To justify our approach, we claim that CMod and RMax produce comparable outputs when the convolution kernel is band-pass and oriented (Gabor-like filter). In this context, CMod can therefore be considered as a stable alternative to RMax. To enforce this property, we constrain the convolution kernels to adopt such a Gabor-like structure. The corresponding architecture is called mathematical twin, because it employs a well-defined mathematical operator to mimic the behavior of the original, freely-trained model. Our approach achieves superior accuracy on ImageNet and CIFAR-10 classification tasks, compared to prior methods based on low-pass filtering. Arguably, our approach's emphasis on retaining high-frequency details contributes to a better balance between shift invariance and information preservation, resulting in improved performance. Furthermore, it has a lower computational cost and memory footprint than concurrent work, making it a promising solution for practical implementation.**

***Index Terms*—deep learning, image processing, shift invariance, max pooling, dual-tree complex wavelet packet transform, aliasing**

I. Introduction

Over the past decade, some progress has been made on understanding the strengths and limitations of convolutional neural networks (CNNs) for computer vision [1], [2]. The ability of CNNs to embed input images into a feature space with linearly separable decision regions is a key factor to achieve high classification accuracy. An important property to reach this linear separability is the ability to discard or minimize non-discriminative image components. In particular, feature vectors are expected to be stable with respect to translations [2]. However, subsampling operations, typically found in convolution and pooling layers, are an important source of instability—a phenomenon known as aliasing [3]. A few approaches have attempted to address this issue.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program Investissement d'avenir, as well as the ANR grant MIAI (ANR-19-P3IA-0003). Most of the computations presented in this paper were performed using the GRICAD infrastructure (https://gricad.univ-grenoble-alpes.fr), which is supported by Grenoble research communities.

Blurpooled CNNs: Zhang [4] proposed to apply a low-pass *blurring* filter before each subsampling operation in CNNs. Specifically, 1) max pooling layers (Max → Sub)[1] are replaced by max-blur pooling (Max → Blur → Sub); 2) convolution layers followed by ReLU (Conv → Sub → ReLU) are blurred before subsampling (Conv → ReLU → Blur → Sub).[2] The combination Blur → Sub is referred to as *blur pooling*. This approach follows a well-known practice called *antialiasing*, which involves low-pass filtering a high-frequency signal before subsampling, in order to avoid artifacts in reconstruction. Their approach improved the shift invariance as well as the accuracy of CNNs trained on ImageNet and CIFAR-10 datasets. However, this was achieved with a significant loss of information.

A question then arises: is it possible to design a non-destructive method, and if so, does it further improve accuracy? In a more recent work, Zou et al. [5] tackled this question through an adaptive antialiasing approach, called *adaptive blur pooling*. Albeit achieving higher prediction accuracy, adaptive blur pooling requires additional memory, computational resources, and trainable parameters.

Proposed Approach: In this paper, we propose an alternative approach based on complex-valued convolutions, extracting high-frequency features that are stable to translations. We observed improved accuracy for ImageNet and CIFAR-10 classification, compared to the two antialiasing methods based on blur pooling [4], [5]. Furthermore, our approach offers significant advantages in terms of computational efficiency and memory usage, and does not induce any additional training, unlike adaptive blur pooling.

Our proposed method replaces the first layers of a CNN: Conv → Sub → Bias → ReLU → MaxPool, which can provably be rewritten as

$$\text{Conv} \to \text{Sub} \to \text{MaxPool} \to \text{Bias} \to \text{ReLU}, \tag{1}$$

by the following combination:

$$\mathbb{C}\text{Conv} \to \text{Sub} \to \text{Modulus} \to \text{Bias} \to \text{ReLU}, \tag{2}$$

where $\mathbb{C}$Conv denotes a convolution operator with a complex-valued kernel, whose real and imaginary parts approximately

[1] Sub and Conv stand for "subsampling" and "convolution," respectively.

[2] ReLU is computed before blurring; otherwise the network would simply perform on low-resolution images.

form a 2D Hilbert transform pair [6]. From (1) and (2), we introduce the two following operators:

$$\mathbb{R}\text{Max} \ : \ \text{Conv} \rightarrow \text{Sub} \rightarrow \text{MaxPool}; \tag{3}$$

$$\mathbb{C}\text{Mod} \ : \ \mathbb{C}\text{Conv} \rightarrow \text{Sub} \rightarrow \text{Modulus}. \tag{4}$$

Our method is motivated by the following theoretical claim. In a recent preprint [7], we proved that 1) $\mathbb{C}$Mod is nearly invariant to translations, if the convolution kernel is band-pass and clearly oriented; 2) $\mathbb{R}$Max and $\mathbb{C}$Mod produce comparable outputs, except for some filter frequencies regularly scattered across the Fourier domain. We then combined these two properties to establish a stability metric for $\mathbb{R}$Max as a function of the convolution kernel's frequency vector. This work was essentially theoretical, with limited experiments conducted on a deterministic model solely based on the dual-tree complex wavelet packet transform (DT-$\mathbb{C}$WPT). However, it lacked applications to tasks such as image classification. Building upon this theoretical study, in this paper, we consider the $\mathbb{C}$Mod operator as a proxy for $\mathbb{R}$Max, extracting comparable, yet more stable features.

In compliance with the theory, the $\mathbb{R}$Max-$\mathbb{C}$Mod substitution is only applied to the output channels associated with oriented band-pass filters, referred to as *Gabor-like kernels*. This kind of structure is known to arise spontaneously in the first layer of CNNs trained on image datasets such as ImageNet [8]. In this paper, we enforce this property by applying additional constraints to the original model. Specifically, a predefined number of convolution kernels are guided to adopt Gabor-like structures, instead of letting the network learn them from scratch. For this purpose, we rely on the dual-tree complex wavelet packet transform (DT-$\mathbb{C}$WPT) [9]. Throughout the paper, we refer to this constrained model as a *mathematical twin*, because it employs a well-defined mathematical operator to mimic the behavior of the original model. In this context, replacing $\mathbb{R}$Max by $\mathbb{C}$Mod is straightforward, since the complex-valued filters are provided by DT-$\mathbb{C}$WPT.

Other Related Work: Chaman and Dokmanic [10] reached perfect shift invariance by using an adaptive, input-dependent subsampling grid, whereas previous models rely on fixed grids. Although this method satisfied shift invariance for integer-pixel translations, it did not address the problem of shift instability for fractional-pixel translations, and therefore falls outside the scope of this paper.

Another aspect of shift invariance in CNNs is related to boundary effects. The fact that CNNs can encode the absolute position of an object in the image by exploiting boundary effects was discovered independently by Islam et al. [11], and Kayhan and Gemert [12]. This phenomenon is left outside the scope of our paper. Finally, [13], [14] studied the impact of pretraining on shift invariance and generalizability to out-of-distribution data, without modifying the network architecture.

II. Proposed Approach

We first describe the general principles of our approach based on complex convolutions. We then present the mathematical twin based on DT-$\mathbb{C}$WPT, and explain how our method has been benchmarked against blur-pooling-based antialiased models.

We represent feature maps with straight capital letters: $\mathrm{X} \in \mathcal{S}$, where $\mathcal{S}$ denotes the space of square-summable 2D sequences. Indexing is denoted by square brackets: for any 2D index $\boldsymbol{n} \in \mathbb{Z}^2$, $\mathrm{X}[\boldsymbol{n}] \in \mathbb{R}$ or $\mathbb{C}$. The cross-correlation between X and $\mathrm{V} \in \mathcal{S}$ is defined by $(\mathrm{X} \star \mathrm{V})[\boldsymbol{n}] := \sum_{\boldsymbol{k}\in\mathbb{Z}^2} \mathrm{X}[\boldsymbol{n}+\boldsymbol{k}]\,\mathrm{V}[\boldsymbol{k}]$. The down arrow refers to subsampling: for any $m \in \mathbb{N}^*$, $(\mathrm{X} \downarrow m)[\boldsymbol{n}] := \mathrm{X}[m\boldsymbol{n}]$.

A. Standard Architectures

A convolution layer with K input channels, L output channels and subsampling factor $m \in \mathbb{N}\backslash\{0\}$ is parameterized by a weight tensor $\mathbf{V} := (\mathrm{V}_{lk})_{l\in\{1..L\},\,k\in\{1..K\}} \in \mathcal{S}^{L\times K}$. For any multichannel input $\mathbf{X} := (\mathrm{X}_k)_{k\in\{1..K\}} \in \mathcal{S}^K$, the corresponding output $\mathbf{Y} := (\mathrm{Y}_l)_{l\in\{1..L\}} \in \mathcal{S}^L$ is defined such that, for any output channel $l \in \{1\,..\,L\}$,

$$\mathrm{Y}_l := \sum_{k=1}^{K} (\mathrm{X}_k \star \mathrm{V}_{lk}) \downarrow m. \tag{5}$$

For instance, in AlexNet and ResNet, $K = 3$ (RGB input images), $L = 64$, and $m = 4$ and 2, respectively. Next, a bias $\boldsymbol{b} := (b_1, \cdots, b_L)^\top \in \mathbb{R}^L$ is applied to $\mathbf{Y}$, which is then transformed through nonlinear ReLU and max pooling operators. The activated outputs satisfy

$$\mathrm{A}_l^{\max} := \mathrm{MaxPool}\left(\mathrm{ReLU}(\mathrm{Y}_l + b_l)\right), \tag{6}$$

where we have defined, for any $\mathrm{Y} \in \mathcal{S}$ and any $\boldsymbol{n} \in \mathbb{Z}^2$,

$$\mathrm{ReLU}(\mathrm{Y})[\boldsymbol{n}] := \max(0,\, \mathrm{Y}[\boldsymbol{n}]); \tag{7}$$

$$\mathrm{MaxPool}(\mathrm{Y})[\boldsymbol{n}] := \max_{\|\boldsymbol{k}\|_\infty \leq 1} \mathrm{Y}[2\boldsymbol{n}+\boldsymbol{k}]. \tag{8}$$

B. Core Principle of our Approach

We consider the first convolution layer of a CNN, as described in (5). As widely discussed in the literature [8], after training with ImageNet, a certain number of convolution kernels V_{lk} spontaneously take the appearance of oriented waveforms with well-defined frequency and orientation (Gabor-like kernels). A visual representation of trained convolution kernels is provided in Fig. 1. In the present paper, we refer to these specific output channels $l \in \mathcal{G} \subset \{1\,..\,L\}$ as *Gabor channels*. The main idea is to substitute, for any $l \in \mathcal{G}$, $\mathbb{R}$Max by $\mathbb{C}$Mod, as explained hereafter. Following (1), expression (6) can be rewritten

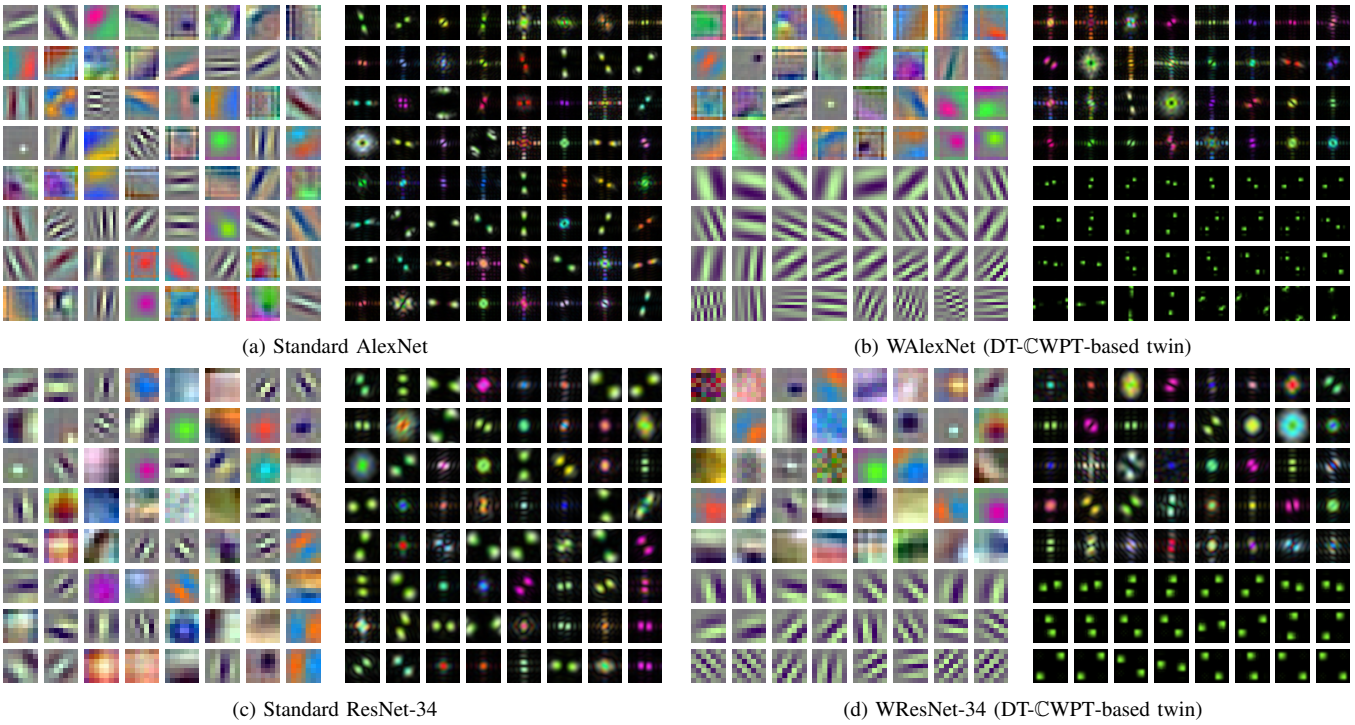
$$\mathrm{A}_l^{\max} = \mathrm{ReLU}\big(\mathrm{Y}_l^{\max} + b_l\big), \tag{9}$$

where $\mathrm{Y}_l^{\max}$ is the output of an $\mathbb{R}$Max operator as introduced in (3). More formally,

$$\mathrm{Y}_l^{\max} := \mathrm{MaxPool}\left(\sum_{k=1}^{K} (\mathrm{X}_k \star \mathrm{V}_{lk}) \downarrow m\right). \tag{10}$$

Then, following (2), the $\mathbb{R}$Max-$\mathbb{C}$Mod substitution yields

$$\mathrm{A}_l^{\mathrm{mod}} = \mathrm{ReLU}\big(\mathrm{Y}_l^{\mathrm{mod}} + b_l\big), \tag{11}$$



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

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

Fig. 1. 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. 1b and 1d), the $L_{\mathrm{free}} := 32$ or 40 first kernels are freely-trained, whereas the remaining $L_{\mathrm{gabor}} := 32$ or 24 kernels are constrained to be monochrome, band-pass and oriented. Left: representation in the spatial domain; right: corresponding power spectra.

where $\mathrm{Y}_l^{\mathrm{mod}}$ is the output of a ℂMod operator (4), satisfying

$$\mathrm{Y}_l^{\mathrm{mod}} := \left| \sum_{k=1}^{K} (\mathrm{X}_k \star \mathrm{W}_{lk}) \downarrow (2m) \right|. \tag{12}$$

In the above expression, W_{lk} is a complex-valued analytic kernel defined as $\mathrm{W}_{lk} := \mathrm{V}_{lk} + i\mathcal{H}(\mathrm{V}_{lk})$, where $\mathcal{H}$ denotes the two-dimensional *Hilbert transform* as introduced by Havlicek et al. [6]. The Hilbert transform is designed such that the Fourier transform of W_{lk} is entirely supported in the half-plane of nonnegative x-values. Therefore, since V_{lk} has a well-defined frequency and orientation, the energy of W_{lk} is concentrated within a small window in the Fourier domain. Due to this property, the modulus operator provides a smooth envelope for complex-valued cross-correlations with W_{lk} [15]. This leads to the output $\mathrm{Y}_l^{\mathrm{mod}}$ (12) being nearly invariant to translations. Additionally, the subsampling factor in (12) is twice that in (10), to account for the factor-2 subsampling achieved through max pooling (8).

C. Wavelet-Based Twin Models (WCNNs)

As explained in Section II-B, introducing an imaginary part to the Gabor-like convolution kernels improves shift invariance. Our method therefore restricts to the Gabor channels $l \in \mathcal{G} \subset \{1\,..\,L\}$. However, $\mathcal{G}$ is unknown a priori: for a given output channel $l \in \{1\,..\,L\}$, whether V_{lk} will become band-pass and oriented after training is unpredictable. Thus, we need a way to automatically separate the set $\mathcal{G}$ of Gabor channels from the set of remaining channels, denoted by $\mathcal{F} := \{1\,..\,L\} \setminus \mathcal{G}$. To this end, we built "mathematical twins" of standard CNNs, based on the dual-tree wavelet packet transform (DT-ℂWPT). These models, which we call WCNNs, reproduce the behavior of freely-trained architectures with a higher degree of control and fewer trainable parameters. In short, the two groups of output channels are organized such that $\mathcal{F} = \{1\,..\,L_{\mathrm{free}}\}$ and $\mathcal{G} = \{(L_{\mathrm{free}}+1)\,..\,L\}$. The first L_{free} channels, which are outside the scope of our approach, remain freely-trained as in the standard architecture. The remaining $L_{\mathrm{gabor}} := 1 - L_{\mathrm{free}}$ channels are constrained to adopt a Gabor-like structure with deterministic frequencies and orientations, through the implementation of DT-ℂWPT. Using the principles introduced in Section II-B, we then replace ℝMax (10) by ℂMod (12) for all Gabor channels $l \in \mathcal{G}$. The corresponding models are referred to as ℂWCNNs. A detailed description of WCNNs and ℂWCNNs is provided in Appendix A, together with schematic representations.

D. WCNNs with Blur Pooling

We benchmark our approach against the antialiasing methods proposed by Zhang [4] and Zou et al. [5]. To this end, we first consider a WCNN antialiased with static or adaptive blur pooling, respectively referred to as BlurWCNN and ABlurWCNN. Then, we substitute the blurpooled Gabor channels with our own ℂMod-based approach. The corresponding models are respectively referred to as ℂBlurWCNN and

ℂABlurWCNN. A schematic representation of BlurWAlexNet and ℂBlurWAlexNet can be found in Fig. 5.

III. Experiments

To ensure reproducibility, we have released the code associated with our study on GitHub.[3]

A. Experiment Details

ImageNet: We built our WCNN and ℂWCNN twin models based on AlexNet [16] and ResNet-34 [17]. The hyperparameter L_{free} was manually chosen based on empirical observations (32 for AlexNet and 40 for ResNet-34). Besides, DT-ℂWPT decompositions were performed with Q-shift orthogonal filters of length 10 as introduced by Kingsbury [18]. More details can be found in Appendix D.

Zhang's static blur pooling approach has been tested on both AlexNet and ResNet, whereas Zou et al.'s adaptive approach has only been tested on ResNet. The latter was indeed not implemented on AlexNet in the original paper, and we were unable to adapt it to this architecture.

Our models were trained on the ImageNet ILSVRC2012 dataset [19], following the standard procedure provided by the PyTorch library [20].[4] Moreover, we set aside 100K images from the training set—100 per class—in order to compute the top-1 error rate after each training epoch ("validation set").

CIFAR-10: We also trained ResNet-18- and ResNet-34-based models on the CIFAR-10 dataset. Training was performed on 300 epochs, with an initial learning rate set to 0.1, decreased by a factor of 10 every 100 epochs. We set aside 5 000 images out of 50K to compute accuracy during the training phase.

B. Evaluation Metrics

Classification Accuracy: Classification accuracy was computed on the ImageNet test set (50K images). We followed the *ten-crops* procedure [16]: predictions are made over 10 patches extracted from each input image, and the softmax outputs are averaged to get the overall prediction. We also considered center crops of size 224 for *one-crop* evaluation. In both cases, we used top-1-5 error rates. For CIFAR-10 evaluation (10K images in the test set), we measured the top-1 error rate with one- and ten-crops.

Measuring Shift Invariance: For each image in the ImageNet evaluation set, we extracted several patches of size 224, each of which being shifted by 0.5 pixel along a given axis. We then compared their outputs in order to measure the model's robustness to shifts. This was done by computing the Kullback-Leibler (KL) divergence between output vectors—which, under certain hypotheses, can be interpreted as probability distributions [21, pp. 205-206]. This metric is intended for visual representation (see Fig. 2).

In addition, we measured the mean flip rate (mFR) between predictions [22], as done by Zhang [4] in its blurpooled models. For each direction (vertical, horizontal and diagonal), we measured the mean frequency upon which two shifted input images yield different top-1 predictions, for shift distances varying from 1 to 8 pixels. We then normalized the results with respect to AlexNet's mFR, and averaged over the three directions. This metric is also referred to as *consistency*.

[3]https://github.com/hubert-leterme/wcnn

[4]PyTorch "examples" repository available at https://github.com/pytorch/examples/tree/main/imagenet

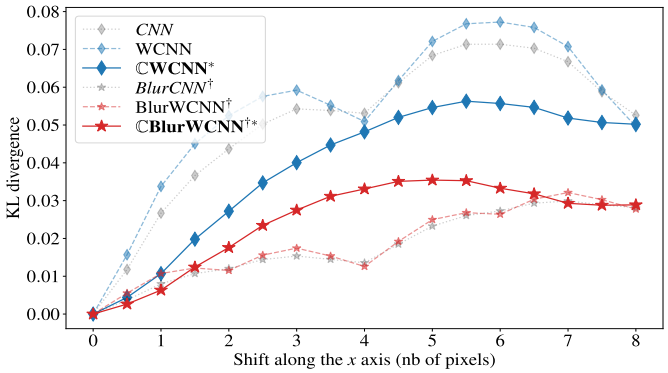


Fig. 2. AlexNet-based models: mean KL divergence between the outputs of shifted images. Legend: †blur pooling; *ℂMod-based approach (ours).

We repeated the procedure for the models trained on CIFAR-10. This time, we extracted patches of size 32×32 from the evaluation set, and computed mFR for shifts varying from 1 to 4 pixels. Normalization was performed with respect to ResNet-18's mFR.

C. Results and Discussion

Validation and Test Accuracy: Error rates of AlexNet- and ResNet-based architectures, computed on the test sets, are provided in Table I for ImageNet and Table II for CIFAR-10.

When trained on ImageNet, our ℂMod-based approach significantly outperforms the baselines for AlexNet: ℂWCNN vs WCNN, and ℂBlurWCNN vs BlurWCNN. Positive results are also obtained for ResNet-based models trained on ImageNet. However, adaptive blur pooling, when applied to the Gabor channels (ABlurWCNN), yields similar or marginally higher accuracy than our approach (ℂABlurWCNN). Nevertheless, our method is computationally more efficient, requires less memory (see "Computational Resources" below for more details), and does not demand additional training, unlike adaptive blur pooling. On the other hand, when trained on CIFAR-10, our approach systematically yields the lowest error rates.

Shift Invariance (KL Divergence): The mean KL divergence between the outputs of shifted images are plotted in Fig. 2 for AlexNet trained on ImageNet. The mean flip rate for shifted inputs (consistency) is reported in Table I for ImageNet (AlexNet and ResNet-34) and Table II for CIFAR-10 (ResNet-18 and 34).

In models without blur pooling (blue curves), the ℝMax-ℂMod substitution greatly reduces first-layer instabilities, resulting in a flattened curve and avoiding the "bumps" observed for non-stabilized models. On the other hand, when applied to the blurpooled models (red curves), the ℝMax-ℂMod substitution actually tends to degrade shift invariance, as evidenced by the bell-shaped curve. Nevertheless, the corresponding classifier is significantly more accurate, as shown in

TABLE I
EVALUATION METRICS ON IMAGENET (%): THE LOWER THE BETTER

Model	One-crop		Ten-crops		Shifts
	top-1	top-5	top-1	top-5	mFR
	AlexNet				
CNN	*45.3*	*22.2*	*41.3*	*19.3*	*100.0*
WCNN	44.9	21.8	40.8	19.0	101.4
ℂWCNN*	**44.3**	**21.3**	**40.2**	**18.5**	**88.0**
BlurCNN†	*44.4*	*21.6*	*40.7*	*18.7*	*63.8*
BlurWCNN†	44.3	21.4	40.5	18.5	**63.1**
ℂBlurWCNN†*	**43.3**	**20.5**	**39.6**	**17.9**	69.4
	ResNet-34				
CNN	*27.6*	*9.2*	*24.8*	*7.7*	*78.1*
WCNN	27.4	9.2	24.7	7.6	77.2
ℂWCNN*	**27.2**	**9.0**	**24.4**	**7.4**	**73.1**
BlurCNN†	*26.7*	*8.6*	*24.0*	*7.2*	*61.2*
BlurWCNN†	26.7	8.6	24.1	7.3	65.2
ℂBlurWCNN†*	**26.5**	**8.4**	**23.7**	**7.0**	**62.5**
ABlurCNN‡	*26.1*	*8.3*	*23.5*	*7.0*	*60.8*
ABlurWCNN‡	**26.0**	**8.2**	**23.6**	**6.9**	**62.1**
ℂABlurWCNN‡*	26.1	**8.2**	23.7	7.0	63.1

†static and ‡adaptive blur pooling; *ℂMod-based approach (ours)

TABLE II
EVALUATION METRICS ON CIFAR-10 (%): THE LOWER THE BETTER

Model	ResNet-18			ResNet-34		
	1crp	10crp	shifts	1crp	10crps	shifts
CNN	*14.9*	*10.8*	*100.0*	*15.2*	*10.9*	*100.3*
WCNN	14.2	10.3	92.4	14.5	10.5	99.2
ℂWCNN*	**13.8**	**9.6**	**88.8**	**12.9**	**9.2**	**93.0**
BlurCNN†	*14.2*	*10.4*	*87.7*	*15.7*	*11.6*	*88.2*
BlurWCNN†	13.1	9.7	**84.6**	13.2	9.9	85.6
ℂBlurWCNN†*	**12.3**	**8.9**	85.7	**12.4**	**9.1**	**83.7**
ABlurCNN‡	*14.6*	*11.0*	*90.9*	*16.3*	*12.8*	*91.9*
ABlurWCNN‡	14.5	11.0	86.5	14.0	10.4	93.3
ℂABlurWCNN‡*	**12.8**	**9.7**	**81.7**	**12.8**	**9.2**	**86.6**

1crp and *10crp*: top-1 error rate using one- and ten-crops methods
shifts: mFR measuring consistency
†static and ‡adaptive blur pooling; *ℂMod-based approach (ours)

Table I. This is not surprising, as our approach prioritizes the conservation of high-frequency details, which are important for classification. An extreme reduction of shift variance using a large blur pooling filter would indeed result in a significant loss of accuracy. Therefore, our work achieves a better tradeoff between shift invariance and information preservation.

To gain further insights into this phenomenon, we conducted experiments by varying the size of the blurring filters. Figure 3 shows the relationship between consistency and prediction accuracy on ImageNet (custom validation set), for AlexNet-based models with different blurring filter sizes ranging from 1 (no blur pooling) to 7 (heavy loss of high-frequency information). Additional plots are provided in Appendix E, for the test set as well as ResNet-based models. We find that a near-optimal trade-off is achieved when the filter size is set to 2 or 3. Furthermore, at equivalent consistency levels, ℂBlurWCNN (our approach) outperforms BlurWCNN in terms of accuracy.

As a side note, because shift invariance is desirable for a wide range of tasks and datasets, embedding this property into CNNs may improve generalizability and avoid overfitting.

Computational Resources: Table III displays the computational resources and memory footprint required for each method, per Gabor channel. The values are normalized relative to non-stabilized AlexNet or ResNet. The metrics are, on the one hand, the FLOPs necessary for computing $\mathrm{Y}_l^{\max}$ (10) or $\mathrm{Y}_l^{\mathrm{mod}}$ (12), and, on the other hand, the size of the intermediate and output tensors saved by PyTorch for the backward pass. More details are provided in Appendix F.

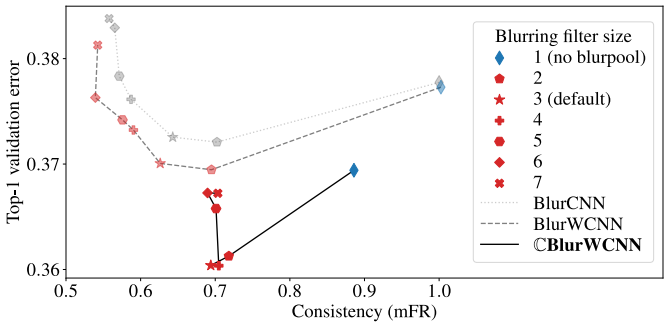


Fig. 3. Classification accuracy (ten-crops) vs consistency, measuring the stability of predictions to small input shifts, for AlexNet-based models (the lower the better for both axes). For each of the three architectures, we increased the blurring filter size from 1 (*i.e.*, no blur pooling) to 7. The 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).

TABLE III
COMPUTATIONAL COST AND MEMORY FOOTPRINT

Method	Computational cost		Memory footprint	
	AlexNet	ResNet	AlexNet	ResNet
No antialiasing (ref)	*1.0*	*1.0*	*1.0*	*1.0*
BlurPool [4]	4.0	1.0	4.7	1.9
ABlurPool [5]	–	2.1	–	2.0
ℂMod (ours)	**0.5**	**0.5**	**0.6**	**0.4**

The observed improvements are mainly due to the larger stride (*i.e.*, subsampling factor) in the first layer, allowing for smaller intermediate feature maps.

IV. CONCLUSION

The mathematical twins introduced in this paper serve as a proof of concept for our ℂMod-based approach. However, its range of application extends well beyond DT-ℂWPT filters. It is important to note that such initial layers play a critical role in CNNs by extracting low-level geometric features such as edges, corners or textures. Therefore, a specific attention is required for their design. In contrast, deeper layers are more focused on capturing high-level structures that conventional image processing tools are poorly suited for [23].

Furthermore, our approach has potential for broader applicability beyond CNNs. There is a growing interest in using self-attention mechanisms in computer vision [24] to capture complex, long-range dependencies among image representations. Recent work on vision transformers has proposed using the first layers of a CNN as a "convolutional token embedding" [25]–[27], effectively reintroducing inductive biases to the architecture, such as locality and weight sharing. By applying our method to this embedding, we can potentially provide self-attention modules with shift-invariant inputs. This could be beneficial in improving the performance of vision transformers, especially when the amount of available data is limited.

APPENDIX 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. 1a and 1c). 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,

$$\forall k \in \{1 \,..\, 3\}\,,\ \mathrm{V}_{lk} = \mu_k \widetilde{\mathrm{V}}_l, \tag{13}$$

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. Monochrome Filters

Expression (13) 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. 1a and 1c). In WCNNs, this constraint is implemented with a trainable 1×1 convolution layer [28], parameterized by $\boldsymbol{\mu}$, computing the following luminance image:

$$\mathrm{X}^{\text{lum}} := \sum_{k=1}^{3} \mu_k \mathrm{X}_k. \tag{14}$$

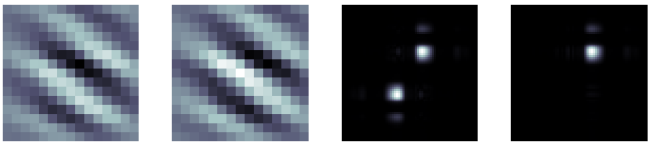
This constraint can be relaxed by authorizing a specific luminance vector $\boldsymbol{\mu}_l$ for each Gabor channel $l \in \mathcal{G}$. Numerical experiments on such models are left for future work.

B. Gabor-Like Kernels

To guarantee the Gabor-like property on $\widetilde{\mathrm{V}}_l$, we implemented DT-$\mathbb{C}$WPT, 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-$\mathbb{C}$WPT 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 4 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:

$$\exists k' \in \big\{1 \,..\, 4 \times 4^J\big\} : \widetilde{\mathrm{V}}_l = \mathrm{Re}\big(\mathrm{W}^{\text{dt}}_{k'}\big). \tag{15}$$



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

Fig. 4. (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

$$\mathrm{Y}_l = \big(\mathrm{X}^{\text{lum}} \star \widetilde{\mathrm{V}}_l\big) \downarrow 2^{J-1}. \tag{16}$$

To summarize, a WCNN substitutes the freely-trained convolution (5) with a combination of (14) and (16), 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. 1b and 1d, respectively.

C. 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 (16) becomes

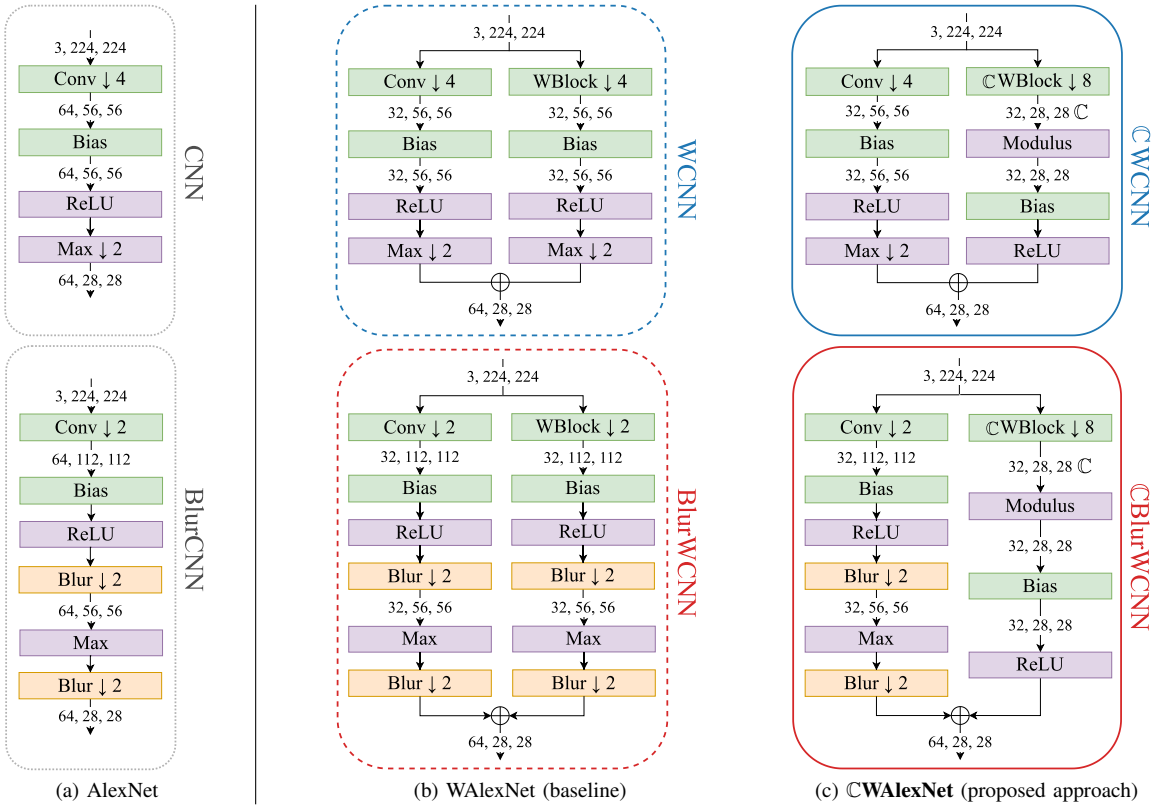
$$\mathrm{Z}_l = \big(\mathrm{X}^{\text{lum}} \star \widetilde{\mathrm{W}}_l\big) \downarrow 2^{J}, \tag{17}$$

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

$$\widetilde{\mathrm{W}}_l := \mathrm{W}^{\text{dt}}_{k'}, \tag{18}$$

where k' has been introduced in (15). Then, a modulus operator is applied to Z_l, which yields $\mathrm{Y}^{\text{mod}}_l$ 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 $\mathrm{Y}^{\text{mod}}_l$, following (11).

A schematic representation of WAlexNet and its stabilized version, referred to as $\mathbb{C}$WAlexNet, is provided in Fig. 5 (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.



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

Fig. 5. 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. 2 (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. 5a: freely-trained models. Top: standard AlexNet. Bottom: Zhang's "blurpooled" AlexNet. Fig. 5b: 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. 5c: 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. 5b) with ours (Fig. 5c) in the Gabor channels.

Appendix 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-ℂWPT filter, as written in (15). We now explain how the filter selection is done; in other words, how k' is chosen among $\{1\,..\,4 \times 4^J\}$. 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 $K_{\text{dt}} := 2 \times 4^J$ the number of such filters.

For any RGB image $\mathbf{X} \in \mathcal{S}^3$, a luminance image $\mathrm{X}^{\text{lum}} \in \mathcal{S}$ is computed following (14), using a 1×1 convolution layer. Then, DT-ℂWPT is performed on X^{lum}. We denote by $\mathbf{D} := (\mathrm{D}_k)_{k \in \{1..K_{\text{dt}}\}}$ the tensor containing the real part of the DT-ℂWPT feature maps:

$$\mathrm{D}_k = \left(\mathrm{X}^{\text{lum}} \star \operatorname{Re} \mathrm{W}_k^{(J)}\right) \downarrow 2^{J-1}. \tag{19}$$

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 [29]. To match the subsampling factor $m := 2^{J-1}$ of the standard model, the last decomposition stage is performed without subsampling.

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

$$\mathbf{D}' := \boldsymbol{\Sigma}\, \mathbf{D}. \tag{20}$$

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, $\mathbf{Y}^{\text{gabor}} := (\mathrm{Y}_l)_{l \in \{L_{\text{free}}+1..L\}} \in \mathcal{S}^{L_{\text{gabor}}}$, where Y_l has been introduced in (5), is also sliced into Q groups of channels

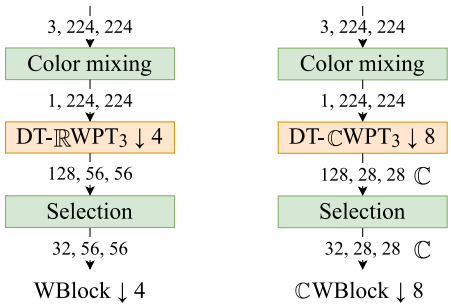


Fig. 6. Detail of a wavelet block with $J = 3$ as in AlexNet, in its $\mathbb{R}$Max (left) and $\mathbb{C}$Mod (right) versions. DT-$\mathbb{R}$WPT corresponds to the real part of DT-$\mathbb{C}$WPT.

$\mathbf{Y}^{(q)} \in \mathcal{S}^{L_q}$. 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 $l \in \{1\,..\,L_q\}$,

$$\mathrm{Y}_l^{(q)} := \boldsymbol{\alpha}_l^{(q)\top} \cdot \mathbf{D}^{(q)}. \tag{21}$$

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. 6.

B. 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 (21) becomes

$$\mathrm{Y}_l^{(q)} = \alpha_{lk}^{(q)} \mathrm{D}_k^{(q)} \tag{22}$$

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 [30] to the loss function to penalize non-sparse feature map mixing as follows:

$$\mathcal{L} := \mathcal{L}_0 + \sum_{q=1}^{Q} \lambda_q \sum_{l=1}^{L_q} \left(\frac{\left\|\boldsymbol{\alpha}_l^{(q)}\right\|_1}{\left\|\boldsymbol{\alpha}_l^{(q)}\right\|_\infty} - 1 \right), \tag{23}$$

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 (23) serves for interpretability of the regularized loss ($\mathcal{L} = \mathcal{L}_0$ in the desired configuration) but has no impact on training.

Appendix C
Adaptation to ResNet: Batch Normalization

In many architectures including ResNet, the bias is computed after an operation called *batch normalization* (BN) [31]. In this context, the first layers have the following structure:

$$\text{Conv} \to \text{Sub} \to \text{BN} \to \text{Bias} \to \text{ReLU} \to \text{MaxPool}. \tag{24}$$

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

$$\mathbb{C}\text{Conv} \to \text{Sub} \to \text{Modulus} \to \text{BN0} \to \text{Bias} \to \text{ReLU}, \tag{25}$$

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. 7.

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

$$\mathrm{A}_l := \mathrm{MaxPool}\left\{ \mathrm{ReLU}\left(a_l \cdot \frac{\mathrm{Y}_l - \mathbb{E}_m[\mathrm{Y}_l]}{\sqrt{\mathbb{V}_m[\mathrm{Y}_l] + \varepsilon}} + b_l \right) \right\}, \tag{26}$$

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 $N \in \mathbb{N} \backslash \{0\}$ 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:

$$\mathbb{E}_m[\mathrm{Y}_l] := \frac{m^2}{N^2} \sum_{\boldsymbol{n} \in \mathbb{Z}^2} \mathbb{E}[\mathrm{Y}_l[\boldsymbol{n}]]; \tag{27}$$

$$\mathbb{V}_m[\mathrm{Y}_l] := \frac{m^2}{N^2} \sum_{\boldsymbol{n} \in \mathbb{Z}^2} \mathbb{V}[\mathrm{Y}_l[\boldsymbol{n}]]. \tag{28}$$

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 (16) (output of the WBlock), with

$$\widetilde{\mathrm{V}}_l := \mathrm{Re}\, \widetilde{\mathrm{W}}_l, \tag{29}$$

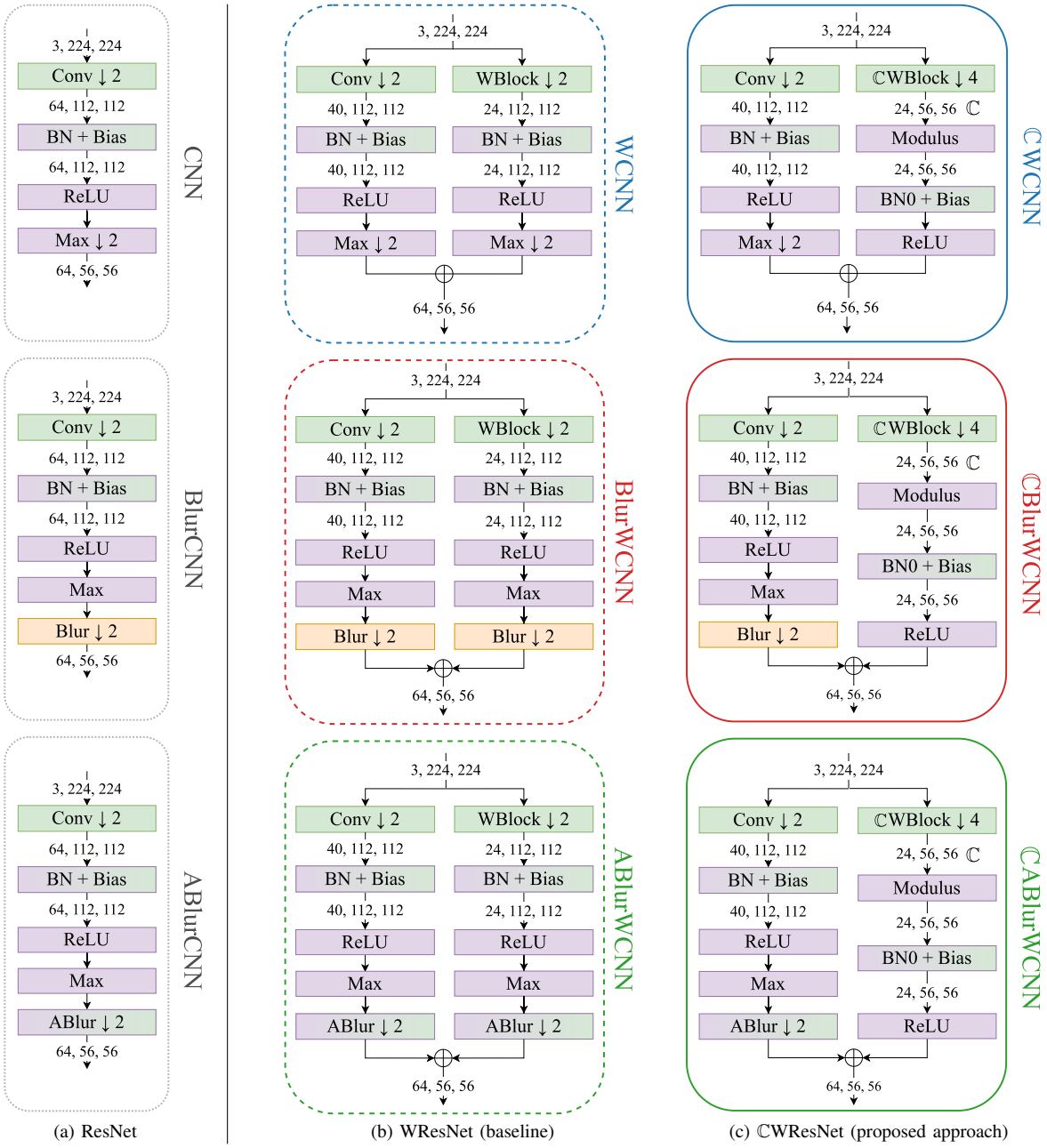
where $\widetilde{\mathrm{W}}_l$ denotes one of the Gabor-like filters spawned by DT-$\mathbb{C}$WPT. The following proposition states that, if the kernel's bandwidth is small enough, then the output of the convolution layer sums to zero.

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

$$\sum_{\boldsymbol{n} \in \mathbb{Z}^2} \mathrm{Y}_l[\boldsymbol{n}] = 0. \tag{30}$$

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



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

Fig. 7. First layers of ResNet and its variants, corresponding to a convolution layer followed by ReLU and max pooling. The bias module from Fig. 5 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].

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 (30) 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 $\boldsymbol{n} \in \mathrm{supp}(\mathrm{X}^{\mathrm{lum}})$. 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 in [32].

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):

$$\mathrm{A}_l^{\max} = \mathrm{ReLU}\left(\frac{\mathrm{Y}_l^{\max}}{\sqrt{\mathbb{E}_m[\mathrm{Y}_l^2] + \varepsilon}} + b_l\right). \tag{31}$$

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):

$$\mathrm{A}_l^{\mathrm{mod}} := \mathrm{ReLU}\left(\frac{\mathrm{Y}_l^{\mathrm{mod}}}{\sqrt{\mathbb{E}_m[\mathrm{Y}_l^2] + \varepsilon}} + b_l\right). \tag{32}$$

Implementing (32) 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

$$\|\mathrm{Y}_l\|_2^2 = 2\left\|\mathrm{Y}_l^{\mathrm{mod}}\right\|_2^2. \tag{33}$$

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 (33) is satisfied, and (32) becomes

$$\mathrm{A}_l^{\mathrm{mod}} := \mathrm{ReLU}\left(\frac{\mathrm{Y}_l^{\mathrm{mod}}}{\sqrt{\frac{1}{2}\mathbb{E}_{2m}[{\mathrm{Y}_l^{\mathrm{mod}}}^2] + \varepsilon}} + b_l\right). \tag{34}$$

In the case of ResNet, the bias layer (Bias) is therefore preceded by a batch normalization layer without mean centering satisfying (34), 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 (34)), which is the subsampling factor for the $\mathbb{C}$Mod operator.

APPENDIX D
IMPLEMENTATION DETAILS

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

A. 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 IV). Therefore, the number of dual-tree filters $K_{\mathrm{dt}} := 2 \times 4^J$ is equal to 128 and 32, respectively.

B. 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 IV, have been empirically determined from the standard models. More specifically, considering standard AlexNet and ResNet-34 trained on ImageNet (see Figs. 1a and 1c, 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* [33]. 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. 8, as explained below.

TABLE IV
EXPERIMENTAL SETTINGS FOR OUR TWIN MODELS

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

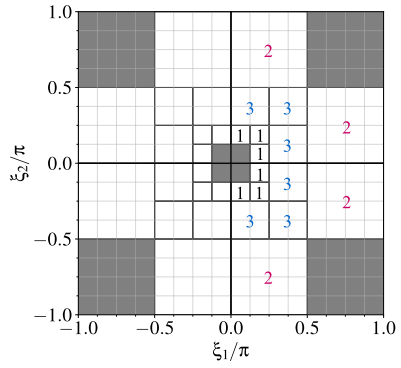
C. Filter Selection and Grouping

We then manually selected $K'_{\mathrm{dt}} < K_{\mathrm{dt}}$ filters, used in (20). 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. 8a). Finally, in WResNet we removed the 14 filters whose bandwidths outreach the boundaries of the Fourier domain $[-\pi, \pi]^2$ (see Fig. 8b). 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 [9, see Fig. 1, "Proposed DT-$\mathbb{C}$WPT"]. Therefore, they somewhat exhibit a checkerboard pattern.[5]

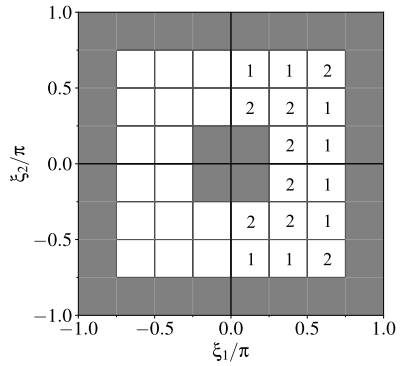
As explained in Section B, once the DT-$\mathbb{C}$WPT feature maps have been manually selected, the output $\mathbf{D'} \in \mathcal{S}^{K'_{\mathrm{dt}}}$ is sliced into Q groups of channels $\mathbf{D}^{(q)} \in \mathcal{S}^{K_q}$. For each group q, a depthwise linear mapping from $\mathbf{D}^{(q)}$ to a bunch of output channels $\mathbf{Y}^{(q)} \in \mathcal{S}^{L_q}$ is performed. Finally, the wavelet block's output feature maps $\mathbf{Y}^{\mathrm{gabor}} \in \mathcal{S}^{L_{\mathrm{gabor}}}$ are obtained by concatenating the outputs $\mathbf{Y}^{(q)}$ depthwise, for any $q \in \{1\,..\,Q\}$. Figure 8 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 (23). 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 V, 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. 8a, respectively.

[5]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 ($J = 2$)

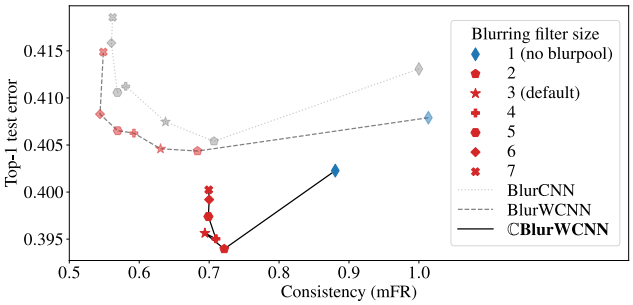
Fig. 8. Mapping scheme from DT-ℂWPT feature maps $\mathbf{D} \in \mathcal{S}^{K_{\mathrm{dt}}}$ to the wavelet block's output $\mathbf{Y}^{\mathrm{gabor}} \in \mathcal{S}^{L_{\mathrm{gabor}}}$. Each wavelet feature map is symbolized by a small square in the Fourier domain, where its energy is mainly 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.

TABLE V
REGULARIZATION HYPERPARAMETERS

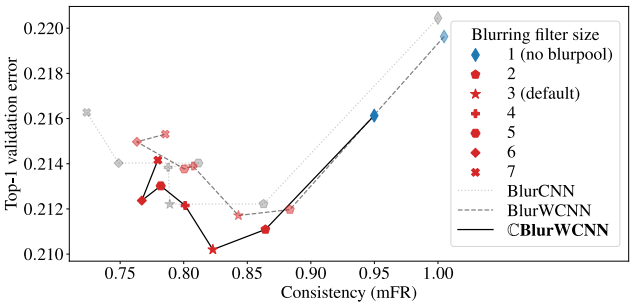
Model	Filt. frequency	Reg. param.
WAlexNet	$[\pi/8,\ \pi/4[$	–
	$[\pi/4,\ \pi/2[$	$4.1 \cdot 10^{-3}$
	$[\pi/2,\ \pi[$	$3.2 \cdot 10^{-4}$
WResNet	any	–

D. Benchmark against Blur-Pooling-based Approaches

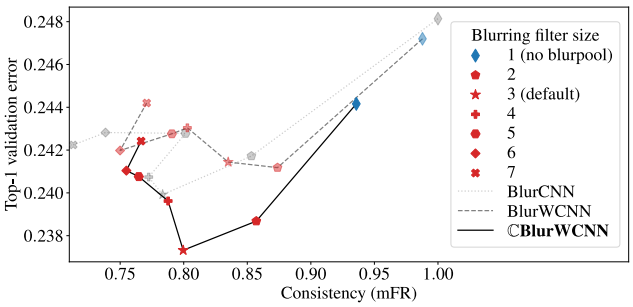
As mentioned in Section II-D, we compare blur-pooling-based antialiasing approach with ours. To apply static or adaptive blur pooling to the WCNNs, we proceed as follows. Following Zhang's implementation, the wavelet block is not antialiased if $m = 2$ as in ResNet, for computational reasons. However, 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].



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



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



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

Fig. 9. 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).

APPENDIX E
ACCURACY VS CONSISTENCY: ADDITIONAL PLOTS

Figure 9 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. 3. 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 ℂMod-based antialiasing approach in the Gabor channels increases accuracy.

APPENDIX 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.

A. 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.

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

B. Computational Cost per Layer

In the following paragraphs, $L \in \mathbb{N}\backslash\{0\}$ 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

$$T_\mathrm{conv} = {N'}^2 \left((KN_\mathrm{filt}^2 - 1) \cdot t_\mathrm{s} + KN_\mathrm{filt}^2 \cdot t_\mathrm{p} \right). \tag{35}$$

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:

$$T_{\mathbb{C}\,\mathrm{conv}} = 2{N'}^2 \left((KN_\mathrm{filt}^2 - 1) \cdot t_\mathrm{s} + KN_\mathrm{filt}^2 \cdot t_\mathrm{p} \right). \tag{36}$$

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:

$$T_\mathrm{bias} = {N'}^2\, t_\mathrm{s} \qquad \text{and} \qquad T_\mathrm{relu} = {N'}^2\, t_\mathrm{relu}. \tag{37}$$

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:

$$T_\mathrm{max} = {N'}^2\, t_\mathrm{max}. \tag{38}$$

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

$$T_\mathrm{mod} = {N'}^2\, t_\mathrm{mod}. \tag{39}$$

f) Batch Normalization: Inputs and outputs of size $(L \times N' \times N')$. A batch normalization (BN) layer, described in (26), 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), $2{N'}^2$ multiplications (divide by standard deviation and multiplicative coefficient).

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

$$T_\mathrm{bn} = {N'}^2 \left(4\, t_\mathrm{s} + 3\, t_\mathrm{p} \right). \tag{40}$$

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

$$T_\mathrm{blur} = {N'}^2 \left((N_\mathrm{b}^2 - 1) \cdot t_\mathrm{s} + N_\mathrm{b}^2 \cdot t_\mathrm{p} \right). \tag{41}$$

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
$$T_{\mathrm{conv\,ablur}} = {N'}^2\, \frac{N_\mathrm{b}^2}{L_\mathrm{g}} \left((LN_\mathrm{b}^2 - 1) \cdot t_\mathrm{s} + LN_\mathrm{b}^2 \cdot t_\mathrm{p} \right). \tag{42}$$
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 $(QN_\mathrm{b}^2 \times N' \times N')$:
$$T_{\mathrm{bn\,ablur}} = {N'}^2\, \frac{N_\mathrm{b}^2}{L_\mathrm{g}} \left(4\, t_\mathrm{s} + 3\, t_\mathrm{p} \right). \tag{43}$$

3) Softmax along the depthwise dimension:

$$T_{\mathrm{sftmx\,ablur}} = N'^2\,\frac{N_\mathrm{b}^2}{L_\mathrm{g}}(t_\mathrm{e} + t_\mathrm{s} + t_\mathrm{p}). \tag{44}$$

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:

$$T_\mathrm{blur} = N'^2\left((N_\mathrm{b}^2 - 1)\cdot t_\mathrm{s} + N_\mathrm{b}^2 \cdot t_\mathrm{p}\right). \tag{45}$$

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

$$T_\mathrm{ablur} = N'^2\,\frac{N_\mathrm{b}^2}{L_\mathrm{g}}\left[\left((L+1)N_\mathrm{b}^2 + 3\right)\cdot t_\mathrm{s} + \left((L+1)N_\mathrm{b}^2 + 4\right)\cdot t_\mathrm{p} + t_\mathrm{e}\right]. \tag{46}$$

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.

C. 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 $N_\mathrm{b} \in \mathbb{N}\backslash\{0\}$. 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.

Appendix 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.[6] 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 (34), 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.

A. 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	$\frac{N}{2m}$	$\frac{N}{2m}$

The memory footprint for each output channel is equal to

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

b) Static Blur Pooling:

Conv $\rightarrow$ Bias $\rightarrow$ ReLU $\rightarrow$ BlurPool $\rightarrow$ Max $\rightarrow$ BlurPool.

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

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

c) $\mathbb{C}$Mod-based Approach:

$\mathbb{C}$Conv $\rightarrow$ Modulus $\rightarrow$ Bias $\rightarrow$ ReLU.

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

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

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

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

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

d) $\mathbb{C}$Mod-based Approach:

$\mathbb{C}$Conv → Modulus → BN0 → Bias → ReLU.

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

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

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] T. Wiatowski and H. Bölcskei, "A Mathematical Theory of Deep Convolutional Neural Networks for Feature Extraction," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1845–1866, Mar. 2018.

[3] A. Azulay and Y. Weiss, "Why do deep convolutional networks generalize so poorly to small image transformations?" *JMLR*, vol. 20, no. 184, pp. 1–25, 2019.

[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] J. Havlicek, J. Havlicek, and A. Bovik, "The analytic image," in *ICIP*, 1997.

[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] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *NeurIPS*, 2014.

[9] 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.

[10] A. Chaman and I. Dokmanic, "Truly Shift-Invariant Convolutional Neural Networks," in *CVPR*, 2021.

[11] M. A. Islam, S. Jia, and N. D. B. Bruce, "How Much Position Information Do Convolutional Neural Networks Encode?" in *ICLR*, 2020.

[12] O. S. Kayhan and J. C. van Gemert, "On Translation Invariance in CNNs: Convolutional Layers Can Exploit Absolute Spatial Location," in *CVPR*, 2020.

[13] V. Biscione and J. S. Bowers, "Convolutional Neural Networks Are Not Invariant to Translation, but They Can Learn to Be," *Journal of Machine Learning Research*, vol. 22, no. 229, pp. 1–28, 2021.

[14] H. Kvinge, T. Emerson, G. Jorgenson, S. Vasquez, T. Doster, and J. Lew, "In What Ways Are Deep Neural Networks Invariant and How Should We Measure This?" in *NeurIPS*, 2022.

[15] N. Kingsbury and J. Magarey, "Wavelet Transforms in Image Processing," in *Signal Analysis and Prediction*, ser. Applied and Numerical Harmonic Analysis. Birkhäuser, 1998, pp. 27–46.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[18] N. Kingsbury, "Design of Q-shift complex wavelets for image processing using frequency domain energy minimization," in *ICIP*, 2003.

[19] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *IJCV*, vol. 115, no. 3, pp. 211–252, Apr. 2015.

[20] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NeurIPS*, 2017.

[21] C. M. Bishop and T. M. Mitchell, *Pattern Recognition and Machine Learning*. Springer, 2014.

[22] D. Hendrycks and T. Dietterich, "Benchmarking Neural Network Robustness to Common Corruptions and Perturbations," in *ICLR*, 2019.

[23] E. Oyallon, E. Belilovsky, and S. Zagoruyko, "Scaling the Scattering Transform: Deep Hybrid Networks," in *ICCV*, 2017.

[24] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *ICLR*, 2021.

[25] A. Hassani, S. Walton, N. Shah, A. Abuduweili, J. Li, and H. Shi, "Escaping the Big Data Paradigm with Compact Transformers," *arXiv:2104.05704*, Jun. 2022.

[26] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, and L. Zhang, "CvT: Introducing Convolutions to Vision Transformers," in *ICCV*, 2021.

[27] K. Yuan, S. Guo, Z. Liu, A. Zhou, F. Yu, and W. Wu, "Incorporating Convolution Designs Into Visual Transformers," in *ICCV*, 2021.

[28] M. Lin, Q. Chen, and S. Yan, "Network In Network," *arXiv:1312.4400 [cs]*, 2014.
[29] 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.
[30] J. Liu and J. Ye, "Efficient L1/Lq Norm Regularization," *arXiv:1009.4766*, Sep. 2010.
[31] 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.
[32] A. Torralba and A. Oliva, "Statistics of natural image categories," *Network: Computation in Neural Systems*, vol. 14, no. 3, pp. 391–412, Jan. 2003.
[33] B. Jahne, *Practical Handbook on Image Processing for Scientific and Technical Applications*. CRC Press, 2004.