Wavelet Integrated CNNs for Noise-Robust Image Classification

Qiufu Li[1,2] Linlin Shen *[1,2] Sheng Guo[3] Zhihui Lai[1,2]
[1]Computer Vision Institute, School of Computer Science & Software Engineering, Shenzhen University
[2]Shenzhen Institute of Artificial Intelligence & Robotics for Society [3]Malong Technologies
{liqiufu,llshen}@szu.edu.cn, sheng@malong.com, lai_zhi_hui@163.com

Abstract

Convolutional Neural Networks (CNNs) are generally prone to noise interruptions, i.e., small image noise can cause drastic changes in the output. To suppress the noise effect to the final predication, we enhance CNNs by replacing max-pooling, strided-convolution, and average-pooling with Discrete Wavelet Transform (DWT). We present general DWT and Inverse DWT (IDWT) layers applicable to various wavelets like Haar, Daubechies, and Cohen, etc., and design wavelet integrated CNNs (WaveCNets) using these layers for image classification. In WaveCNets, feature maps are decomposed into the low-frequency and high-frequency components during the down-sampling. The low-frequency component stores main information including the basic object structures, which is transmitted into the subsequent layers to extract robust high-level features. The high-frequency components, containing most of the data noise, are dropped during inference to improve the noise-robustness of the WaveCNets. Our experimental results on ImageNet and ImageNet-C (the noisy version of ImageNet) show that WaveCNets, the wavelet integrated versions of VGG, ResNets, and DenseNet, achieve higher accuracy and better noise-robustness than their vanilla versions.

1. Introduction

Drastic changes due to small variations of the input can emerge in the output of a well-trained convolutional neural network (CNN) for image classification [13, 36, 12]. Particularly, the CNN is associated with weak noise-robustness [15]. Random noise of data is mostly high-frequency components. In the field of signal processing, transforming the data into different frequency intervals, and denoising the components in the high-frequency intervals, is an effective way to denoise it [9, 10]. The transformation, such as Discrete Wavelet Transform (DWT) [26], consists of filtering and down-sampling. However, the commonly used CNN architectures (VGG, ResNets, and DenseNet, etc.) do not perform filtering before the feature map down-sampling.

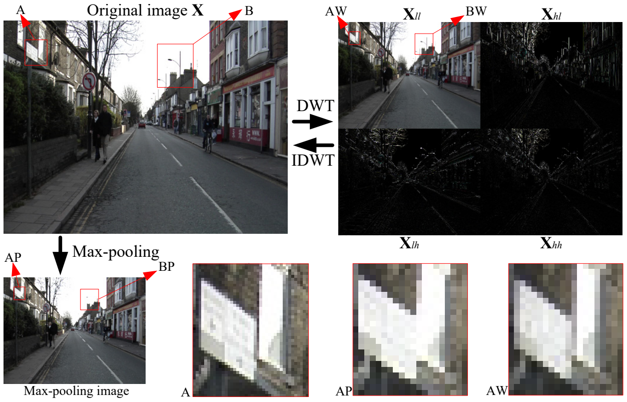


Figure 1. Comparison of max-pooling and wavelet transforms. Max-pooling is a commonly used down-sampling operation in the deep networks, which could easily breaks the basic object structures. Discrete Wavelet Transform (DWT) decomposes an image $\mathbf{X}$ into its low-frequency component $\mathbf{X}_{ll}$ and high-frequency components $\mathbf{X}_{lh}, \mathbf{X}_{hl}, \mathbf{X}_{hh}$. While $\mathbf{X}_{lh}, \mathbf{X}_{hl}, \mathbf{X}_{hh}$ represent image details including most of the noise, $\mathbf{X}_{ll}$ is a low resolution version of the data, where the basic object structures are represented. In the figures, the window boundary in area A (AP) and the poles in area B (BP) are broken by max-pooling, while the principal features of these objects are kept in the DWT output (AW and BW).

Without the filtering, down-sampling may result in the aliasing among low-frequency and high-frequency components [28, 42]. In particular, noise in the high-frequency components could be down-sampled into the following feature maps, and degrade the noise-robustness of the CNNs. Meanwhile, the basic object structures presented in the low-frequency component could be broken, as Fig. 1 shows.

In this paper, to suppress the noise effect to the final predication and increase the classification accuracy, we integrate wavelet into commonly used CNN architectures. We firstly transform DWT and Inverse DWT (IDWT) as general network layers in PyTorch [29]. Then, we design wavelet integrated convolutional network (WaveCNet), by replacing the commonly used down-sampling with DWT. During down-

*Corresponding Author: Linlin Shen.

sampling, WaveCNet eliminates the high-frequency components of the feature maps to increase the noise-robustness of the CNNs, and then extracts high-level features from the low-frequency component for better classification accuracy. Using ImageNet [8] and ImageNet-C [15], we evaluate WaveCNets in terms of classification accuracy and noise-robustness, when various wavelets and various CNN architectures are used. At last, we explore the application of DWT/IDWT layer in image segmentation. In summary:

1. We present general DWT/IDWT layer applicable to various wavelets, which could be used to design end-to-end wavelet integrated deep networks.
2. We design WaveCNets by replacing existing down-sampling operations with DWT to improve the classification accuracy and noise-robustness of CNNs.
3. We evaluate WaveCNets on ImageNet, and achieve increased accuracy and better noise-robustness.
4. The proposed DWT/IDWT layer is further integrated into SegNet [2] to improve the segmentation performance of encoder-decoder networks.

2. Related works

2.1. Noise-robustness

When the input image is changed, the output of CNN can be significantly different, regardless of whether the change can be easily perceived by human or not [13, 12, 21, 36]. While the changes may result from various factors, such as shift [42, 25], rotation [5], noise [36], blur [15], manual attack [13], etc., we focus on the robustness of CNNs to the common noise. A high-level representation guided denoiser is designed in [21] to denoise the contaminated image before inputting it into the CNN, which may complicate the whole deep network structure. In [36], the authors propose denoising block for CNNs to denoise the feature map and suppress the effect of noise on the final prediction. However, the authors design their denoising block using the spacial filtering, such as Gaussian filtering, mean filtering, and median filtering, etc., which do denoising in the whole frequency domain and may break basic object structure contained in the low-frequency component. Therefore, their denoising block requires a residual structure for the CNN to converge. Recently, a benchmark evaluating CNN performance on noisy images is proposed in [15]. Our WaveCNets will be evaluated using this benchmark.

The recent studies show that ImageNet-trained CNNs prefer to extract features from object textures sensitive to noise [3, 12]. Stylized ImageNet [12] is proposed via stylizing ImageNet images with style transfer to enable the CNNs to extract more robust features from object structures. The noise could be enlarged as the feature maps flow through layers in the CNNs [21, 36], resulting in the final wrong predictions. These issues may be related to the down-sampling operations ignoring the classic sampling theorem.

2.2. Down-sampling

For local connectivity and weight sharing, researchers introduce into deep networks various down-sampling operations, such as max-pooling, average-pooling, mixed-pooling, stochastic pooling, and strided-convolution, etc. While max-pooling and average-pooling are simple and effective, they can erase or dilute details from images [38, 40]. Although mixed-pooling [38] and stochastic pooling [40] are introduced to address these issues, max-pooling, average-pooling, and strided-convolution are still the most widely used operations in CNNs [14, 16, 31, 33].

These down-sampling operations usually ignore the classic sampling theorem [1, 42], which could break object structures and accumulate noise. Fig. 1 shows a max-pooling example. Anti-aliased CNNs [42] integrate the classic anti-aliasing filtering with the down-sampling. The author is surprised at the increased classification accuracy and better noise-robustness. Compared to the anti-aliased CNNs, our WaveCNets are significantly different in two aspects: (1) While Max operation is still used in anti-aliased CNNs, WaveCNets do not require such operation. (2) The low-pass filters used in anti-aliased CNNs are empirically designed based on the row vectors of Pascal's triangle, which is ad hoc and no theoretical justifications are given. As no up-sampling operation, i.e., reconstruction, of the low-pass filter is available, the anti-aliased U-Net [42] has to apply the same filtering after normal up-sampling to achieve the anti-aliasing effect. In comparison, our WaveCNets are justified by the well defined wavelet theory [6, 26]. Both down-sampling and up-sampling can be replaced by DWT and IDWT, respectively.

In deep networks for image-to-image translation tasks, the up-sampling operations, such as transposed convolution in U-Net [30] and max-unpooling in SegNet [2], are widely applied to upgrade the feature map resolution. Due to the absence of the strict mathematical terms, these up-sampling operations can not precisely recover the original data. They do not perform well in the restoration of image details.

2.3. Wavelets

Wavelets [6, 26] are powerful time-frequency analysis tools, which have wide applications in signal processing. While Discrete Wavelet Transform (DWT) decompose a data into various components in different frequency intervals, Inverse DWT (IDWT) could reconstruct the data using the DWT output. DWT could be applied for anti-aliasing in signal processing, and we will explore its application in deep networks. IDWT could be used for detail restoration in image-to-image tasks.

Wavelet has been combined with neural network for function approximation [41], signal representation and classification [34]. In these early works, the authors apply shallow networks to search the optimal wavelet in wavelet parameter domain. Recently, this method is utilized with deeper network for image classification, but the network is difficult to train because of the significant amount of computational cost [7]. ScatNet [5] cascades wavelet transform with nonlinear modulus and average pooling, to extract a translation invariant feature robust to deformations and preserve high-frequency information for image classification. The authors introduce ScatNet when they explore from mathematical and algorithmic perspective how to design the optimal deep network. Compared with the CNNs of the same period, ScatNet gets better performance on the handwritten digit recognition and texture discrimination tasks. However, due to the strict mathematical assumptions, ScatNet can not be easily transferred to other tasks.

In deep learning, wavelets commonly play the roles of image preprocessing or postprocessing [17, 23, 32, 39]. Meanwhile, researchers try to introduce wavelet transforms into the design of deep networks in various tasks [22, 35, 11, 37], by taking wavelet transforms as sampling operations. Multi-level Wavelet CNN (MWCNN) proposed in [22] integrates Wavelet Package Transform (WPT) into the deep network for image restoration. MWCNN concatenates the low-frequency and high-frequency components of the input feature map, and processes them in a unified way, while the data distribution in these components significantly differs from each other. Convolutional-Wavelet Neural Network (CWNN) proposed in [11] applies dual-tree complex wavelet transform (DT-CWT) to suppress the noise and keep the structures for extracting robust features from SAR images. The architecture of CWNN contains only two convolution layers. While DT-CWT is redundant, CWNN takes as its down-sampling output the average value of the multiple components output from DT-CWT. Wavelet pooling proposed in [35] is designed using a two-level DWT. Its back-propagation performs a one-level DWT and a two-level IDWT, which does not follow the mathematical principle of gradient. The authors test their method on various dataset (MNIST [20], CIFAR-10 [18], SHVN [27], and KDEF [24]). However, their network architectures contain only four or five convolutional layers. The authors do not study systematically the potential of the method on standard image dataset like ImageNet [8]. Recently, the application of wavelet transform in image style transfer is studied in [37]. In above works, the authors evaluate their methods with only one or two wavelets, due to the absence of the general wavelet transform layers.

3. Our method

Our method is trying to apply wavelet transforms to improve the down-sampling operations in deep networks. We firstly design the general DWT and IDWT layers.

3.1. DWT and IDWT layers

The key issues in designs of DWT and IDWT layers are the data forward and backward propagations. Although the following analysis is for orthogonal wavelet and 1D signal, it can be generalized to other wavelets and 2D/3D signal with only slight changes.

Forward propagation For a 1D signal $\mathbf{s} = \{s_j\}_{j\in\mathbb{Z}}$, DWT decomposes it into its low-frequency component $\mathbf{s}_1 = \{s_{1k}\}_{k\in\mathbb{Z}}$ and high-frequency component $\mathbf{d}_1 = \{d_{1k}\}_{k\in\mathbb{Z}}$, where

$$\begin{cases} s_{1k} = \sum_j l_{j-2k} s_j, \\ d_{1k} = \sum_j h_{j-2k} s_j, \end{cases} \tag{1}$$

and $\mathbf{l} = \{l_k\}_{k\in\mathbb{Z}}, \mathbf{h} = \{h_k\}_{k\in\mathbb{Z}}$ are the low-pass and high-pass filters of an orthogonal wavelet. According to Eq. (1), DWT consists of filtering and down-sampling.

Using IDWT, one can reconstruct $\mathbf{s}$ from $\mathbf{s}_1, \mathbf{d}_1$, where

$$s_j = \sum_k \left(l_{j-2k} s_{1k} + h_{j-2k} d_{1k} \right). \tag{2}$$

In expressions with matrices and vectors, Eq. (1) and Eq. (2) can be rewritten as

$$\mathbf{s}_1 = \mathbf{L}\mathbf{s}, \quad \mathbf{d}_1 = \mathbf{H}\mathbf{s}, \tag{3}$$

$$\mathbf{s} = \mathbf{L}^T \mathbf{s}_1 + \mathbf{H}^T \mathbf{d}_1, \tag{4}$$

where

$$\mathbf{L} = \begin{pmatrix} \cdots & \cdots & \cdots & & & & \\ \cdots & l_{-1} & l_0 & l_1 & \cdots & & \\ & & \cdots & l_{-1} & l_0 & l_1 & \cdots \\ & & & & & \cdots & \cdots \end{pmatrix}, \tag{5}$$

$$\mathbf{H} = \begin{pmatrix} \cdots & \cdots & \cdots & & & & \\ \cdots & h_{-1} & h_0 & h_1 & \cdots & & \\ & & \cdots & h_{-1} & h_0 & h_1 & \cdots \\ & & & & & \cdots & \cdots \end{pmatrix}. \tag{6}$$

For 2D signal $\mathbf{X}$, the DWT usually do 1D DWT on its every row and column, i.e.,

$$\mathbf{X}_{ll} = \mathbf{L}\mathbf{X}\mathbf{L}^T, \tag{7}$$

$$\mathbf{X}_{lh} = \mathbf{H}\mathbf{X}\mathbf{L}^T, \tag{8}$$

$$\mathbf{X}_{hl} = \mathbf{L}\mathbf{X}\mathbf{H}^T, \tag{9}$$

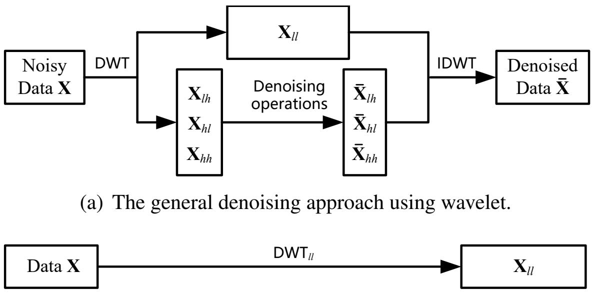
$$\mathbf{X}_{hh} = \mathbf{H}\mathbf{X}\mathbf{H}^T, \tag{10}$$

and the corresponding IDWT is implemented with

$$\mathbf{X} = \mathbf{L}^T \mathbf{X}_{ll} \mathbf{L} + \mathbf{H}^T \mathbf{X}_{lh} \mathbf{L} + \mathbf{L}^T \mathbf{X}_{hl} \mathbf{H} + \mathbf{H}^T \mathbf{X}_{hh} \mathbf{H}. \tag{11}$$

Backward propagation For the backward propagation of DWT, we start from Eq. (3) and differentiate it,

$$\frac{\partial \mathbf{s}_1}{\partial \mathbf{s}} = \mathbf{L}^T, \quad \frac{\partial \mathbf{d}_1}{\partial \mathbf{s}} = \mathbf{H}^T. \tag{12}$$



(a) The general denoising approach using wavelet.

(b) The simplest wavelet based "denoising" method, DWT$_{ll}$.

Figure 2. The general denoising approach based on wavelet transforms and the one used in WaveCNet.

Similarly, for the back propagation of the 1D IDWT, differentiate Eq. (4),

$$\frac{\partial \mathbf{s}}{\partial \mathbf{s}_1} = \mathbf{L}, \quad \frac{\partial \mathbf{s}}{\partial \mathbf{d}_1} = \mathbf{H}. \tag{13}$$

The forward and backward propagations of 2D/3D DWT and IDWT are slightly more complicated, but similar to that of 1D DWT and IDWT. In practice, we choose the wavelets with finite filters, for example, Haar wavelet with $\mathbf{l} = \frac{1}{\sqrt{2}}\{1,1\}$ and $\mathbf{h} = \frac{1}{\sqrt{2}}\{1,-1\}$. For finite signal $\mathbf{s} \in \mathbb{R}^N$ and $\mathbf{X} \in \mathbb{R}^{N\times N}$, the $\mathbf{L},\mathbf{H}$ are truncated to be the size of $\lfloor \frac{N}{2} \rfloor \times N$. We transform 1D/2D/3D DWT and IDWT as network layers in PyTorch. In the layers, we do DWT and IDWT channel by channel for multi-channel data.

3.2. WaveCNets

Given a noisy 2D data $\mathbf{X}$, the random noise mostly show up in its high-frequency components. Therefore, as Fig. 2(a) shows, the general wavelet based denoising [9, 10] consists of three steps: (1) decompose the noisy data $\mathbf{X}$ using DWT into low-frequency component $\mathbf{X}_{ll}$ and high-frequency components $\mathbf{X}_{lh}, \mathbf{X}_{hl}, \mathbf{X}_{hh}$, (2) filter the high-frequency components, (3) reconstruct the data with the processed components using IDWT.

In this paper, we choose the simplest wavelet based "denoising", i.e., dropping the high-frequency components, as Fig. 2(b) shows. DWT$_{ll}$ denotes the transform mapping the feature maps to the low-frequency component. We design WaveCNets by replacing the commonly used down-sampling with DWT$_{ll}$. As Fig. 3 shows, in WaveCNets, max-pooling and average-pooling are directly replaced by DWT$_{ll}$, while strided-convolution is upgrated using convolution with stride of 1 followed by DWT$_{ll}$, i.e.,

$$\mathrm{MaxPool}_{s=2} \rightarrow \mathrm{DWT}_{ll}, \tag{14}$$

$$\mathrm{Conv}_{s=2} \rightarrow \mathrm{DWT}_{ll} \circ \mathrm{Conv}_{s=1}, \tag{15}$$

$$\mathrm{AvgPool}_{s=2} \rightarrow \mathrm{DWT}_{ll}, \tag{16}$$

where "MaxPool$_s$", "Conv$_s$" and "AvgPool$_s$" denote the max-pooling, strided-convolution, and average-pooling with stride s, respectively.

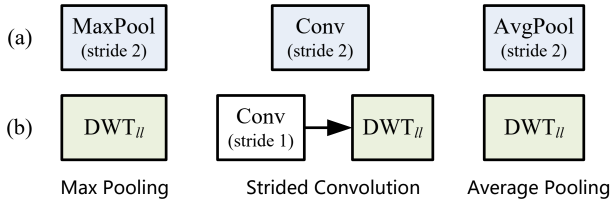


Figure 3. (a) Baseline, the down-sampling operations in deep networks. (b) Wavelet integrated down-sampling in WaveCNets.

While DWT$_{ll}$ halves the size of the feature maps, it removes their high-frequency components and denoises them. The output of DWT$_{ll}$, i.e., the low-frequency component, saves the main information of the feature map to extract the identifiable features. During down-sampling of WaveCNets, DWT$_{ll}$ could resist the noise propagation in the deep networks and helps to maintain the basic object structure in the feature maps. Therefore, DWT$_{ll}$ would accelerate the training of deep networks and lead to better noise-robustness and increased classification accuracy.

4. Experiments

The commonly used CNN architectures for image classification, such as VGG [33], ResNets [14], DenseNet [16], compose of various max-pooling, average-pooling, and strided-convolution. By upgrading the down-sampling with Eqs. (14) - (16), we create WaveCNets, including WVGG16bn, WResNets, WDenseNet121. Compared with the original CNNs, WaveCNets do not employ additional learnable parameters. We evaluate their classification accuracies and noise-robustness using ImageNet [8] and ImageNet-C [15]. At last, we explore the potential of wavelet integrated deep networks for image segmentation.

4.1. ImageNet classification

ImageNet contains 1.2M training and 50K validation images from 1000 categories. On the training set, we train WaveCNets when various wavelets are used, with the standard training protocols from the publicly available PyTorch [29] repository. Table 1 presents the top-1 accuracy of WaveCNets on ImageNet validation set, where "haar", "dbx", and "chx.y" denote the Haar wavelet, Daubechies wavelet with approximation order x, and Cohen wavelet with orders (x, y). The length of the wavelet filters increase as the orders increase. While Haar and Cohen wavelets are symmetric, Daubechies are not.

In Table 1, parenthesized numbers are accuracy difference compared with the baseline results. The baseline results, i.e., the results of the original CNNs, are sourced from the official PyTorch [29]. For all CNN architectures, Haar and Cohen wavelets improve their classification ac-

Table 1. Top-1 accuracy of WaveCNets on ImageNet validation set.

Wavelet		WVGG16bn	WResNet18	WResNet34	WResNet50	WResNet101	WDenseNet121
None (baseline)*		73.37	69.76	73.30	76.15	77.37	74.65
Haar		74.10 (+0.73)	71.47 (+1.71)	74.35 (+1.05)	**76.89 (+0.74)**	78.23 (+0.86)	75.27 (+0.62)
Cohen	ch2.2	74.31 (+0.94)	**71.62 (+1.86)**	74.33 (+1.03)	76.41 (+0.26)	78.34 (+0.97)	75.36 (+0.71)
	ch3.3	**74.40 (+1.03)**	71.55 (+1.79)	74.51 (+1.21)	76.71 (+0.56)	**78.51 (+1.14)**	**75.44 (+0.79)**
	ch4.4	74.02 (+0.65)	71.52 (+1.76)	**74.61 (+1.31)**	76.56 (+0.41)	78.47 (+1.10)	75.29 (+0.64)
	ch5.5	73.67 (+0.30)	71.26 (+1.50)	74.34 (+1.04)	76.51 (+0.36)	78.39 (+1.02)	75.01 (+0.36)
Daubechies	db2	74.08 (+0.71)	71.48 (+1.72)	74.30 (+1.00)	76.27 (+0.12)	78.29 (+0.92)	75.08 (+0.43)
	db3		71.08 (+1.32)	74.11 (+0.81)	76.38 (+0.23)		
	db4		70.35 (+0.59)	73.53 (+0.23)	75.65 (−0.50)		
	db5		69.54 (−0.22)	73.41 (+0.11)	74.90 (−1.25)		
	db6		68.74 (−1.02)	72.68 (−0.62)	73.95 (−2.20)		

* corresponding to the results of original CNNs, i.e., VGG16bn, ResNets, DenseNet121.

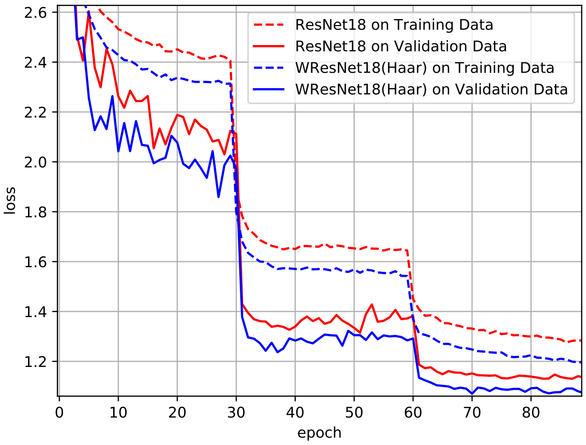


Figure 4. The loss of ResNet18 and WResNet18(Haar).

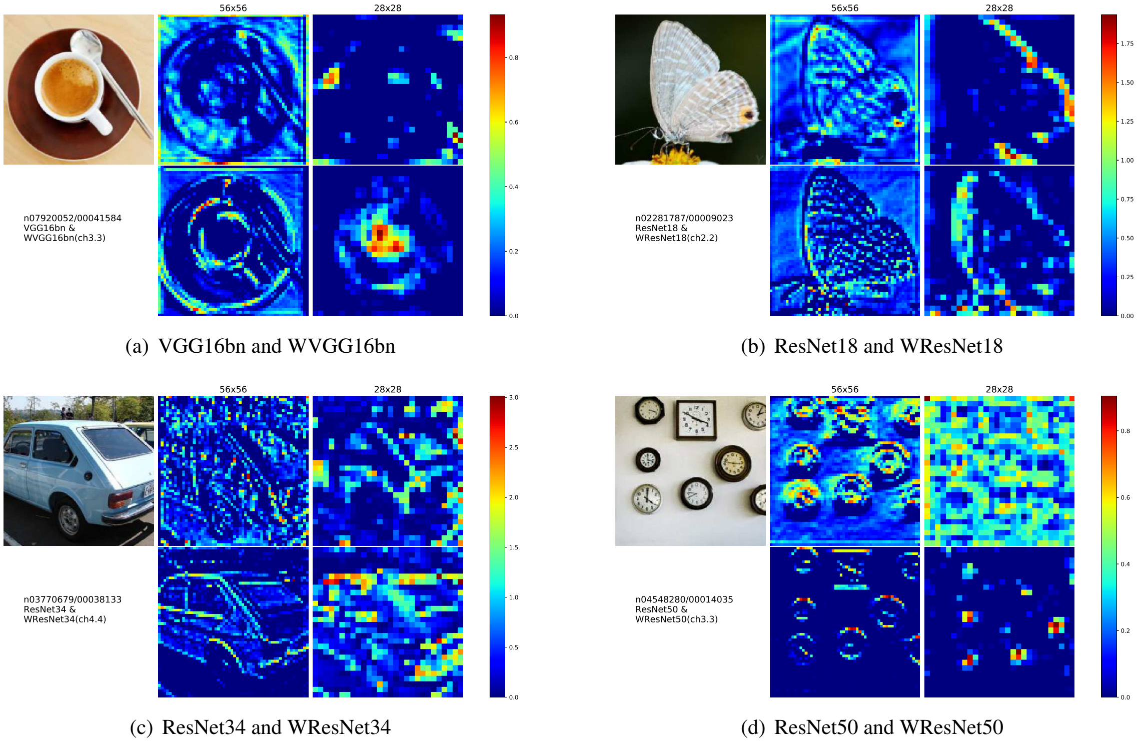
curacy, although the best wavelet varies with CNN. For example, Haar and Cohen wavelets improve the accuracy of ResNet18 by 1.50% to 1.86%. However, the performance of asymmetric Daubechies wavelet gets worse as the approximation order increases. Daubechies wavelets with shorter filters ("db2" and "db3") could improve the CNN accuracy, while that with longer filters ("db5" and "db6") may reduce the accuracy. For example, the top-1 accuracy of WResNet18 decreases from 71.48% to 68.74%. We conclude that the symmetric wavelets perform better than asymmetric ones in image classification. That is the reason why we do not train WVGG16bn, WResNet101, W-DenseNet121 with "db3", "db4", "db5", "db6".

We retrain ResNet18 using the standard ImageNet classification training repository in PyTorch. In Fig. 4, we compare the losses of ResNet18 and WResNet18(Haar) during the training procedure. Fig. 4 adopts red dashed and green dashed lines to denote the train losses of ResNet18 and WResNet18(Haar), respectively. Throughout the whole training procedure, the training loss of WResNet18(Haar) is about 0.08 lower than that of ResNet18, when the two networks employ the same amount of learnable parameters. This suggests that wavelet accelerates the training of ResNet18 architecture. On the validation set, WResNet18 loss (green solid line) is also always lower than ResNet18 loss (red solid line), which lead to the increase of final classification accuracy by 1.71%.

Fig. 5 presents four example feature maps of well trained CNNs and WaveCNets. In each subfigure, the top row shows the input image with size of 224 × 224 from ImageNet validation set and the two feature maps produced by original CNN, while the bottom row shows the related information (image, CNN and WaveCNet names) and feature maps produced by the WaveCNet. The two feature maps are captured from the 16th output channel of the final layer in the network blocks with tensor size of 56 × 56 (middle) and 28 × 28 (right), respectively. The feature maps have been enlarged for better illustration.

From Fig. 5, one can find that the backgrounds of the feature map produced by WaveCNets are cleaner than that produced by CNNs, and the object structures in the former are more complete than that in the latter. For example, in the top row of Fig. 5(d), the clock boundary in the ResNet50 feature map with size of 56 × 56 are fuzzy, and the basic structures of clocks have been totally broken by strong noise in the feature map with size of 28 × 28. In the second row, the backgrounds of feature maps produced by WResNet50(ch3.3) are very clean, and it is easy to figure out the clock structures in the feature map with size of 56 × 56 and the clock areas in the feature map with size of 28 × 28. The above observations illustrate that the down-sampling operations could cause noise accumulation and break the basic object structures during CNN inference, while DWT in WaveCNets relieves these drawbacks. We believe that this is the reason why WaveCNets converge faster in training and ultimately achieve better classification accuracy.

In [42], the author is surprised at the increased classification accuracy of CNNs after filtering is integrated into the down-sampling. In [12], the authors show that "ImageNet-trained CNNs are strongly biased towards recognising textures rather than shapes". Our experimental results suggest that this may be sourced from the commonly used down-sampling operations, which tend to break the object structures and accumulate noise in the feature maps.



(a) VGG16bn and WVGG16bn

(b) ResNet18 and WResNet18

(c) ResNet34 and WResNet34

(d) ResNet50 and WResNet50

Figure 5. The feature maps of CNNs (top) and WaveCNets (bottom).

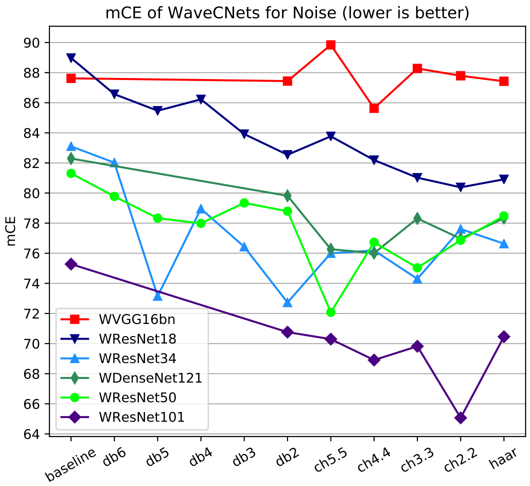


Figure 6. The noise mCE of WaveCNets.

4.2. Noise-robustness

In [15], the authors corrupt the ImageNet validation set using 15 visual corruptions with five severity levels, to create ImageNet-C and test the robustness of ImageNet-trained classifiers to the input corruptions. The 15 corruptions are sourced from four categories, i.e., noise (Gaussian noise, shot noise, impulse noise), blur (defocus blur, frosted glass blur, motion blur, zoom blur), weather (snow, frost, fog, brightness), and digital (contrast, elastic, pixelate, JPEG-compression). $E_{s,c}^{f}$ denotes the top-1 error of a trained classifier f on corruption type c at severity level s. The authors present the Corruption Error CE_c^f, computed with

$$\mathrm{CE}_c^f = \sum_{s=1}^{5} E_{s,c}^{f} \Bigg/ \sum_{s=1}^{5} E_{s,c}^{\mathrm{AlexNet}}, \tag{17}$$

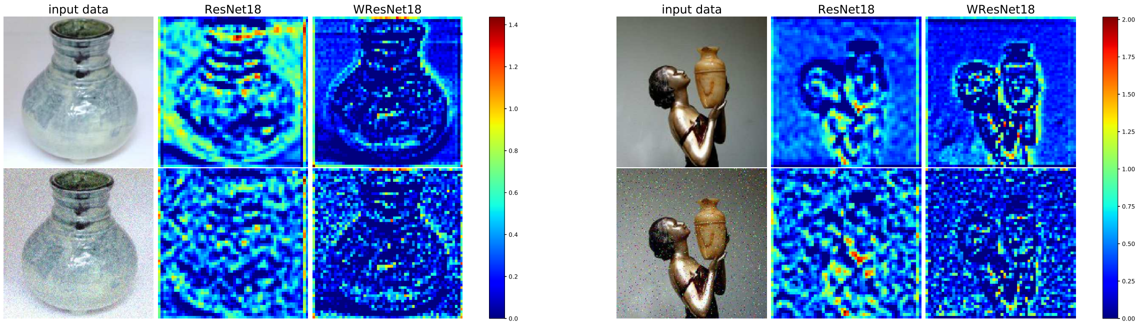
to evaluate the performance of a trained classifier f. In Eq. (17), the authors normalize the error using the top-1 error of AlexNet [19] to adjust the difference of various corruptions.

In this section, we use the noise part (750K images, 50K × 3 × 5) of ImageNet-C and

$$\mathrm{mCE}_{\text{noise}}^f = \frac{1}{3}\left(\mathrm{CE}_{\text{Gaussian}}^f + \mathrm{CE}_{\text{shot}}^f + \mathrm{CE}_{\text{impulse}}^f\right) \tag{18}$$

to evaluate the noise-robustness of WaveCNet f.

We test the top-1 errors of WaveCNets and AlexNet on each noise corruption type c at each level of severity s, when WaveCNets and AlexNet are trained on the clean ImageNet training set. Then, we compute $\mathrm{mCE}_{\text{noise}}^{\text{WaveCNet}}$ according to Eqs. (17) and (18). In Fig. 6, we show the noise mCEs of WaveCNets for different network architectures and various wavelets. The "baseline" corresponds to the noise mCEs of original CNN architectures, while "dbx", "chx.y" and "haar" correspond to the mCEs of WaveCNets with different wavelets. Except VGG16bn, our method



(a) Gaussian noise

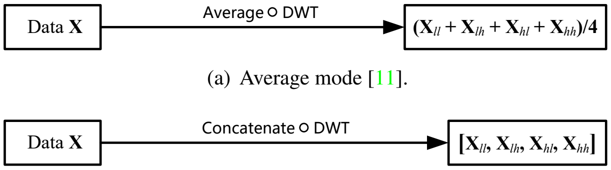
(b) Impulse noise

Figure 7. The feature maps sourced from clean (top) and noisy (bottom) images.

obviously increase the noise-robustness of the CNN architectures for image classification. For example, the noise mCE of ResNet18 (with navy blue color and down triangle marker in Fig. 6) decreases from 88.97 (“baseline”) to 80.38 (“ch2.2”). One can find that the all wavelets including “db5” and “db6” improve the noise-robustness of ResNet18, ResNet34, and ResNet50, although the classification accuracy of the WResNets with “db5” and “db6” for the clean images may be lower than that of the original ResNets. It means that our methods indeed increase the noise-robustness of these network architectures.

Fig. 7 shows two example feature maps for well trained ResNet18 and WResNet18 with noisy images as input. In every subfigure, the first row shows the clean image with size of 224 × 224 from ImageNet validation set and feature maps generated by ResNet18 and WResNet18(ch2.2), respectively. The second row shows the image added with Gaussian or impulse noise and the feature maps generated by the two networks. These feature maps are captured from the 16th output channel of the last layer in the network blocks with tensor size of 56 × 56. From the two examples, one can find that it is difficult for the original CNN to suppress noise, while WaveCNet could suppress the noise and maintain the object structure during its inference. For example, in Fig. 7(a), the bottle structures in the two feature maps generated by ResNet18 and WResNet18(ch2.2) are complete, when the clean porcelain bottle image is fed into the networks. However, after the image is corrupted by Gaussian noise, the ResNet18 feature map contains very strong noise and the bottle structure vanishes, while the basic structure could still be observed from the WResNet18 feature map. This advantage improves the robustness of WaveCNets against different noise.

The noise-robustness of VGG16bn is inferior to that of ResNet34, although they achieve similar accuracy (73.37% and 73.30%). Our method can not significantly improve the noise-robustness of VGG16bn, although it can increase the accuracy by 1.03%. It means that the VGG16bn may be not a proper architecture in terms of noise-robustness.


Data X
Average ∘ DWT
$(\mathbf{X}_{ll} + \mathbf{X}_{lh} + \mathbf{X}_{hl} + \mathbf{X}_{hh})/4$
Data X
Concatenate ∘ DWT
$[\mathbf{X}_{ll}, \mathbf{X}_{lh}, \mathbf{X}_{hl}, \mathbf{X}_{hh}]$

(a) Average mode [11].

(b) Concatenation mode [22].

Figure 8. Wavelet integrated down-sampling in various modes.

4.3. Comparison with other wavelet based down-sampling

Different with our DWT based down-sampling (Fig. 2(b)), there are other wavelet integrated down-sampling modes in literatures. In [11], the authors adopt as down-sampling output the average value of the multiple components of wavelet transform, as Fig. 8(a) shows. In [22], the authors concatenate all the components output from DWT, and process them in a unified way, as Fig. 8(b) shows.

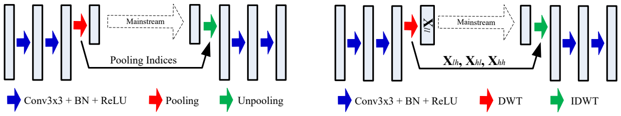
Here, taking ResNet18 as backbone, we compare our wavelet integrated down-sampling with the previous approaches, in terms of classification accuracy and noise-robustness. We rebuild ResNet18 using the three down-sampling modes shown in Fig. 2(b) and Fig. 8, and denote them as WResNet18, WResNet18_A, and WResNet18_C, respectively. We train them on ImageNet when various wavelets are used. Table 2 shows the accuracy on ImageNet and the noise mCEs on the ImageNet-C. Generally, the networks using wavelet based down-sampling achieve better accuracy and noise mCE than that of original network, ResNet18 (69.76% accuracy and 88.97 mCE).

Similar to WResNet18, the number of parameters of WResNet18_A is the same with that of original ResNet18. However, the added high-frequency components in the feature maps damage the information contained in the low-frequency component, because of the high-frequency noise. WResNet18_A performs the worst among the networks using wavelet based down-sampling.

Due to the tensor concatenation, WResNet18_C employs

Table 2. Comparison with other wavelet based down-sampling.

Network	Top-1 Accuracy (higher is better)						Params.
	haar	ch2.2	ch3.3	ch4.4	ch5.5	db2	
WResNet18	71.47	71.62	71.55	71.52	71.26	71.48	**11.69**M
WResNet18_A [11]	70.06	69.24	69.91	69.98	70.31	70.52	11.69M
WResNet18_C [22]	**71.94**	**71.75**	**71.66**	**71.99**	**72.03**	**71.88**	21.62M
WResNet34	74.35	74.33	74.51	74.61	74.34	74.30	21.80M
	Noise mCE (lower is better)						
WResNet18	**80.91**	**80.38**	**81.02**	82.19	83.77	82.54	
WResNet18_A [11]	83.17	86.02	86.07	85.22	82.96	84.01	
WResNet18_C [22]	81.79	83.67	83.51	**82.13**	**82.60**	**80.11**	
WResNet34	76.64	77.61	74.30	76.19	76.00	72.73	



(a) SegNet (b) WaveUNets

Figure 9. Down-sampling and up-sampling used in SegNet and WaveUNet.

much more parameters (21.62×10^6) than WResNet18 and WResNet18_A (11.69×10^6). WResNet18_C thus increase the accuracy of WResNet18 by 0.11% to 0.77%, when various wavelets are used. However, due to the included noise, the concatenation does not evidently improve the noise-robustness. In addition, the amount of parameters for WResNet18_C is almost the same with that for WResNet34 (21.80×10^6), while the accuracy and noise mCE of WResNet34 are obviously superior to that of WResNet18_C.

4.4. Image segmentation

The main contributions of our method are the DWT and IDWT layers. IDWT is a useful up-sampling approach to recover the data details. With IDWT, WaveCNets can be easily transferred to image-to-image translation tasks. We now test their applications in semantic image segmentation.

To restore details in image segmentation, we design WaveUNets by replacing the max-pooling and max-unpooling in SegNet [2] with DWT and IDWT. SegNet adopts encoder-decoder architecture and uses VGG16bn as its encoder backbone. In its decoder, SegNet recovers the feature map resolution using max-unpooling, as Fig. 9(a) shows. While max-unpooling only recover very limited details, IDWT can recover most of the data details. In the encoder, WaveUNets decompose the feature maps into various frequency components, as Fig. 9(b) shows. While the low-frequency components are used to extract high-level features, the high-frequency components are stored and transmitted to the decoder for resolution restoration with IDWT.

We evaluate WaveUNets and SegNet using CamVid [4] dataset. CamVid contains 701 road scene images (367, 101, and 233 for the training, validation, and test). In [2], the authors train the SegNet using an extended CamVid training set containing 3433 images, which achieved 60.10% mIoU on the CamVid test set. We train SegNet and WaveUNet with various wavelets using only 367 CamVid training images. Table 3 shows the mIoU on the CamVid test set. Our WaveUNets get higher mIoU and achieve the best result (64.23%) with Haar wavelet.

Table 3. Results on CamVid test set.

Network	SegNet		Our WaveUNets					
	[2]	Ours	haar	ch2.2	ch3.3	ch4.4	ch5.5	db2
mIoU	60.10	57.89	**64.23**	63.35	62.90	63.76	63.61	63.78

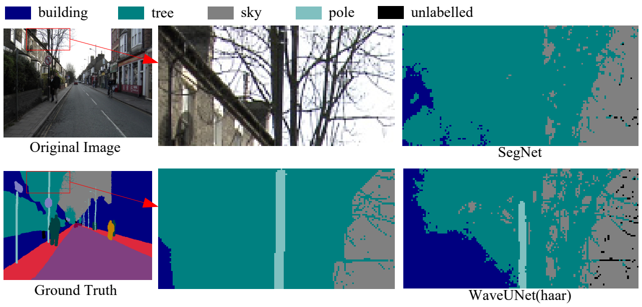


Figure 10. Comparison of SegNet and WaveUNet segmentations.

In Fig. 10, we present a visual example for SegNet and WaveUNet segmentations. Fig. 10 shows the example image, its manual annotation, a region consisting of “building”, “tree”, “sky” and “pole”, and the segmentation results achieved using SegNet and WaveUNet. The region has been enlarged with colored segmentation results for better illustration. From the figure, one can find in the segmentation result that WaveUNet keeps the basic structure of “tree”, “pole”, and “building” and restores the object details, such as the “tree” branches and the “pole”. The segmentation result of WaveUNet matches the image region much better than that of SegNet, even corrects the annotation noise about “building” and “tree” in the ground truth.

5. Conclusions

We transform Discrete Wavelet Transform (DWT) and Inverse DWT (IDWT) into general network layers, and design wavelet integrated convolutional networks (WaveCNets) for image classification. Being able to well keep object structures and suppress data noise during network inference, WaveCNets achieve higher image classification accuracy and better noise-robustness for various commonly used network architectures.

Acknowledgments

The work was supported by the Natural Science Foundation of China under grants no. 61672357, 91959108 and U1713214, and the Science and Technology Project of Guangdong Province under grant no. 2018A050501014.

References

[1] Aharon Azulay and Yair Weiss. Why do deep convolutional networks generalize so poorly to small image transformations? *arXiv preprint arXiv:1805.12177*, 2018. 2

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on PAMI*, 39(12):2481–2495, 2017. 2, 8

[3] Wieland Brendel and Matthias Bethge. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. *arXiv preprint arXiv:1904.00760*, 2019. 2

[4] Gabriel J Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009. 8

[5] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE transactions on PAMI*, 35(8):1872–1886, 2013. 2, 3

[6] Ingrid Daubechies. *Ten lectures on wavelets*, volume 61. Siam, 1992. 2

[7] DDN De Silva, HWMK Vithanage, KSD Fernando, and ITS Piyatilake. Multi-path learnable wavelet neural network for image classification. *arXiv preprint arXiv:1908.09775*, 2019. 3

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 2, 3, 4

[9] David L Donoho. De-noising by soft-thresholding. *IEEE transactions on information theory*, 41(3):613–627, 1995. 1, 4

[10] David L Donoho and Jain M Johnstone. Ideal spatial adaptation by wavelet shrinkage. *biometrika*, 81(3):425–455, 1994. 1, 4

[11] Yiping Duan, Fang Liu, Licheng Jiao, Peng Zhao, and Lu Zhang. Sar image segmentation based on convolutional-wavelet neural network and markov random field. *Pattern Recognition*, 64:255–267, 2017. 3, 7, 8

[12] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018. 1, 2, 5

[13] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 1, 2

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2, 4

[15] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019. 1, 2, 4, 6

[16] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 2, 4

[17] Huaibo Huang, Ran He, Zhenan Sun, and Tieniu Tan. Wavelet-srnet: A wavelet-based cnn for multi-scale face super resolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1689–1697, 2017. 3

[18] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 3

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 6

[20] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 3

[21] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1778–1787, 2018. 2

[22] Pengju Liu, Hongzhi Zhang, Kai Zhang, Liang Lin, and Wangmeng Zuo. Multi-level wavelet-cnn for image restoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 773–782, 2018. 3, 7, 8

[23] Yunfan Liu, Qi Li, and Zhenan Sun. Attribute enhanced face aging with wavelet-based generative adversarial networks. *arXiv preprint arXiv:1809.06647*, 2018. 3

[24] Daniel Lundqvist, Anders Flykt, and Arne Öhman. The karolinska directed emotional faces (kdef). *CD ROM from Department of Clinical Neuroscience, Psychology section, Karolinska Institutet*, 91:630, 1998. 3

[25] Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional kernel networks. In *Advances in neural information processing systems*, pages 2627–2635, 2014. 2

[26] Stephane G Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7):674–693, 1989. 1, 2

[27] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 3

[28] Harry Nyquist. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, 47(2):617–644, 1928. 1

[29] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 1, 4

[30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on MICCAI*, pages 234–241. Springer, 2015. 2

[31] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE*

Conference on Computer Vision and Pattern Recognition, pages 4510–4520, 2018. 2

[32] Behrouz Alizadeh Savareh, Hassan Emami, Mohamadreza Hajiabadi, Seyed Majid Azimi, and Mahyar Ghafoori. Wavelet-enhanced convolutional neural network: a new idea in a deep learning paradigm. *Biomedical Engineering/Biomedizinische Technik*, 64(2):195–205, 2019. 3

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2, 4

[34] Harold H Szu, Brian A Telfer, and Shubha L Kadambe. Neural network adaptive wavelets for signal representation and classification. *Optical Engineering*, 31(9):1907–1917, 1992. 3

[35] Travis Williams and Robert Li. Wavelet pooling for convolutional neural networks. In *International Conference on Learning Representations*, 2018. 3

[36] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 501–509, 2019. 1, 2

[37] Jaejun Yoo, Youngjung Uh, Sanghyuk Chun, Byeongkyu Kang, and Jung-Woo Ha. Photorealistic style transfer via wavelet transforms. *arXiv preprint arXiv:1903.09760*, 2019. 3

[38] Dingjun Yu, Hanli Wang, Peiqiu Chen, and Zhihua Wei. Mixed pooling for convolutional neural networks. In *International Conference on Rough Sets and Knowledge Technology*, pages 364–375. Springer, 2014. 2

[39] Binhang Yuan, Chen Wang, Fei Jiang, Mingsheng Long, Philip S Yu, and Yuan Liu. Waveletfcnn: A deep time series classification model for wind turbine blade icing detection. *arXiv preprint arXiv:1902.05625*, 2019. 3

[40] Matthew D Zeiler and Rob Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*, 2013. 2

[41] Qinghua Zhang and Albert Benveniste. Wavelet networks. *IEEE transactions on Neural Networks*, 3(6):889–898, 1992. 3

[42] Richard Zhang. Making convolutional networks shift-invariant again. *arXiv preprint arXiv:1904.11486*, 2019. 1, 2, 5