Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2023.0322000

A Lattice-structure-based Trainable Orthogonal Wavelet Unit for Image Classification

AN D. LE[1], (Graduate Student Member, IEEE), SHIWEI JIN[1], YOU-SUK BAE[2], and TRUONG Q. NGUYEN[1], (Fellow, IEEE)

[1]Electrical and Computer Engineering Department, University of California San Diego, La Jolla, CA 92093, USA
[2]Department of Computer Engineering, Tech University of Korea, Siheung 15073, Korea

Corresponding author: An D. Le (e-mail: d0le@ucsd.edu).

This research was supported by the Ministry of Science and ICT (MSIT), Korea, under the Innovative Human Resource Development for Local Intellectualization support program (ITP-2023-2020-0-01741) supervised by the Institute for Information & communications Technology Planning & Evaluation (IITP)

ABSTRACT This work introduces Orthogonal Lattice Universal Wavelet Unit, a novel trainable wavelet unit to enhance image classification and anomaly detection in convolutional neural networks by reducing information loss during pooling. The unit employs an orthogonal lattice structure, relaxing the zero-at-π condition and decreasing the number of trainable wavelet coefficients. This innovation is a key novelty of the work. The unit modifies convolution, pooling, and down-sampling operations. Implemented in residual neural networks with 18 layers, it improved detection accuracy on CIFAR10 (by 2.67%), ImageNet1K (by 1.85%), and the Describable Textures dataset (by 9.52%), showcasing its advantages in detecting detailed features. Similar gains were seen in the implementations for residual neural networks with 34 layers and 50 layers. For anomaly detection in hazelnut images on the MVTec Anomaly Detection dataset, the proposed method achieved a segmentation area under the receiver operating characteristic curve of 97.15% and better anomaly localization. The method excels in detecting detailed features, despite increased trainable parameters from using one-layer fully convolutional networks for feature combination.

INDEX TERMS Anomaly detection, Computer vision, Discrete wavelet transforms, Feature extraction, Image processing, Image recognition, Machine learning, Supervised learning, Wavelet coefficients, Wavelet transform.

I. INTRODUCTION

Max pooling and average pooling are common downsampling functions in convolutional neural networks (CNNs) for computer vision applications. The max pooling method selects the pixel with the largest value for the down-sampled feature map, and the average pooling method averages the pixels in the kernel, smoothing out the feature map. Both of these conventional pooling methods degrade details in the down-sampled feature maps because both functions operate as low-pass filters. Max pooling and average pooling methods are also deterministic processes with vanishing effects on feature maps in deeper layers [1]. Because these downsampling methods do not perform filtering, feature maps in common CNN architectures, such as VGG [2], DenseNet [3], Mobilenets [4, 5], and ResNets [6] are also affected by aliasing among low-frequency and high-frequency components. These effects compromise the performance of models working on textural data, such as in the Describable Textures Dataset (DTD) [7]. Detailed features are also needed for detecting objects like insects [8, 9] due to their intricate and textural characteristics [10, 11].

Fig. 1 presents several examples where high-frequency or detail image parts hold important information. The CIFAR10 [12] sample shown in Fig. 1 has most information concentrated in the low-frequency region or the approximation component. In contrast, all other samples from ImageNet1K [13], MVTec AD [14, 15], and DTD [7] have information in both low (approximation) and high (detail) frequency regions, demonstrating the importance of both low-frequency and high-frequency information in images.

Several methods have been proposed [16–18] to deal with the drawbacks of conventional pooling methods. In [16], the AVG-TopK pooling model was proposed, which takes K pixels with dominant values and averages them, while a universal pooling method was proposed in [17], which uses a linear combination of image features, where average pooling, max pooling, and stride pooling functions are special cases. The AVG-MAX VPB pooling module was proposed in [18]. In

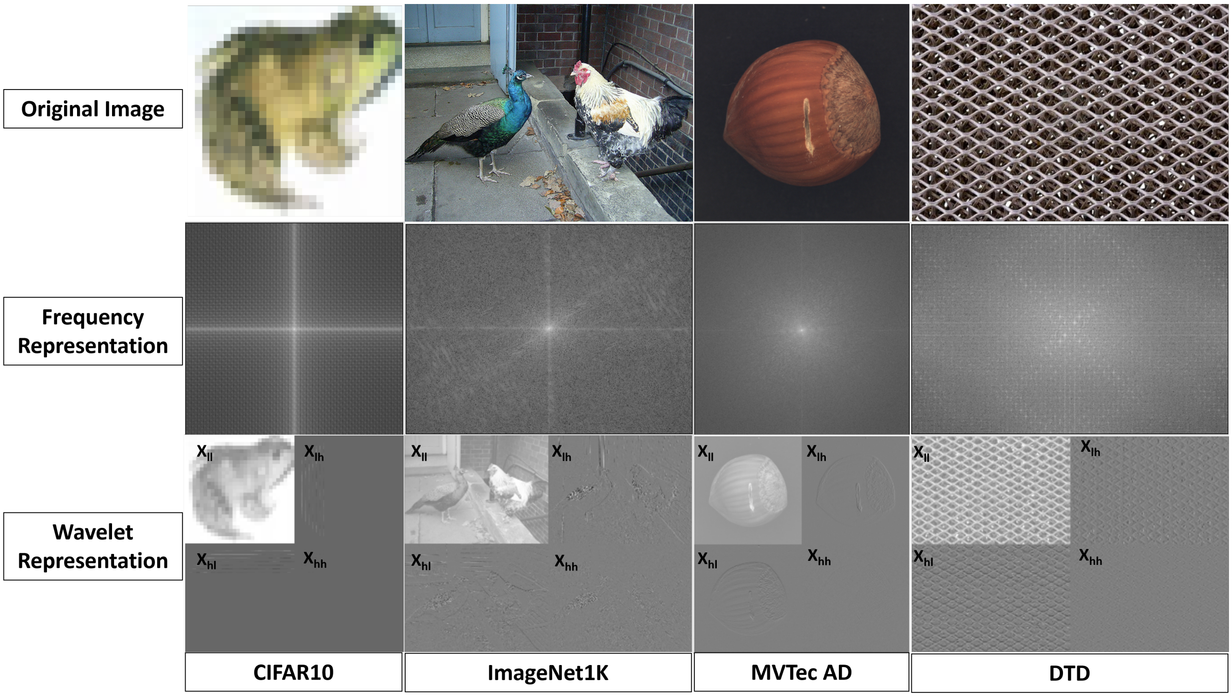

FIGURE 1. From left to right, Haar-wavelet and frequency representations of sample images from CIFAR10 (first column), ImageNet1K (second column), MVTec AD (third column), and DTD (fourth column). The original images (top row) are shown with their frequency representations (middle row) and wavelet representations (bottom row). X_{ll}, X_{lh}, X_{hl}, and X_{hh} show the coarse approximation and details wavelet representations.

that approach, max pooling and average pooling are applied in the vertical and horizontal directions. Spectral pooling-based methods proposed in [19, 20] pool the features from the frequency domain to avoid aliasing, while other methods utilize the wavelet basis to develop alternative pooling approaches [21–23]. Spectral-pooling-based approaches mainly utilize the low-pass frequency information in pooling and discard the high-frequency parts. Li et al. [21] only used the approximation or low-pass information in the pooling. In contrast, the work in [22] places more attention and constraints on the detail, or high-frequency information.

The aforementioned works utilize either max pooling, which may lead to aliasing, or average pooling, a low-pass filter method that results in the loss of detailed features in deeper layers. Additionally, previously proposed wavelet and frequency-based approaches only employ either low-pass or high-pass features. Moreover, the filter coefficients in these prior wavelet-based methods are fixed. There are also a limited number of works attempting to make wavelets trainable, as seen in [24, 25]. The work in [25] introduced a wavelet loss function that applies constraints to low-pass and high-pass filters while maintaining reversibility and symmetry conditions on the trainable filters' coefficients. In contrast, [24] introduced the relaxation of the perfect reconstruction constraint, along with a perfect reconstruction loss function, to train the filter bank's coefficients.

In the new proposed approach, we leverage both low-pass and high-pass features in the network and allow the filter coefficients in the wavelet decomposition to be tunable. We introduce a new learnable orthogonal wavelet unit using a lattice structure, named the Orthogonal Lattice Universal Wavelet Unit (Orthogonal-LatticeUwU), where the lattice coefficients of the lattice structure are optimized to increase the classification performance. The use of lattice coefficients instead of filter coefficients will also reduce the number of trainable coefficients in the filters. These aspects also differentiate our work from other related studies, highlighting its novelty. The decomposed wavelet components are combined to generate the optimal feature maps for the CNN by using a one-layer Fully Convolutional Network (FCN). The goal of this implementation is to fully utilize both low-frequency and high-frequency features from an optimized filter bank. The proposed Orthogonal-LatticeUwU unit is used to generate optimal feature maps for the downsampling and pooling functions to achieve better performance. In addition, Orthogonal-LatticeUwU followed by non-stride convolutional layers can replace the traditional stride convolutional layers and further enhance CNN performance. The proposed units in our work enable the wavelet to be specifically trainable during CNN training, a novel aspect that, to our knowledge, has not been explored in previous research. We apply the proposed units on ResNet18-based architecture then compare our proposed method with wavelet-based variants, such as WaveCNet [21], Wavelet-Attention CNN [22], Convolutional Wavelet Neural Network (CWNN) [23], Learnable Discrete Wavelet Pooling (LDW-Pooling) [25], wavelet unit with perfect reconstruction relaxation (PR-relaxation) [24], and spectrum-based approaches such as SpectralPooling [19] and DiffStride [20] on

CIFAR10 [12], ImageNet1K [13], and DTD [7] datasets.We then incorporate the proposed units into anomaly detection and segmentation tasks in the hazelnut category group of the industrial inspection MVTec Anomaly Detection (MVTec AD) dataset [14, 15]. The key elements of the proposed new method are as follows:

- We propose Orthogonal-LatticeUwU, a learnable orthogonal wavelet unit based on the lattice structure to improve classification performance. As orthogonal filter banks and their lattice structure are used, perfect reconstruction is maintained and the number of parameters is minimized.
- To achieve better classification performance, we relax the smoothness constraints (zeros at π) on the filter.
- The lattice coefficients are initialized using a synthesis procedure given the existing wavelet filter coefficients.
- The proposed method is implemented and tested on a wide range of image classification datasets, achieving excellent performance.
- The proposed unit is also used in anomaly detection.

In this paper, related works on conventional pooling methods and frequency and wavelet-based approaches are first discussed. Subsequently, the proposed Orthogonal-LatticeUwU unit, its theory and implementation are presented. The proposed unit is evaluated for image classification and anomaly detection tasks on ImageNet1K [13], CIFAR10 [12], DTD [7], and MVTec AD [14, 15] datasets. The results and performance of the proposed method are also illustrated and compared with other related approaches.

II. RELATED WORKS

A. CONVENTIONAL POOLINGS

Both average and max pooling methods [26] compute local statistics and are typically followed by a non-strided convolution to downsample and extract the feature maps in a CNN architecture. Max pooling, a prevalent downsampling method, retains the maximal values, thus preserving prominent features [27]. In contrast, average pooling averages values over feature maps, producing a smoothing effect; this method was notably used in LeNet [28]. However, both of these downsampling methods utilize only a subset of the image features, potentially discarding vital information. Furthermore, downsampling without filtering can result in aliasing between low-pass and high-pass components in the feature maps, violating the sampling theory [29]. In addition, it has been reported that max pooling can degrade object structures in deep networks [21].

B. FREQUENCY DOMAIN-BASED APPROACHES

To address the aliasing problem inherent in max pooling and average pooling methods, spectral pooling was introduced [19]. This method uses the discrete Fourier Transform (DFT) technique to leverage the frequency domain during the pooling process. Essentially, spectral pooling operates as a low-pass filter that truncates the frequency representation of the input, retaining only the lower frequencies to mitigate aliasing [29]. Spectral pooling was first presented in [19]. However, it was highlighted in [20] that spectral pooling was non-differentiable with respect to its strides. As a result, the number of strides must be predefined as a hyper-parameter for each downsampling layer.

The study in [20] tackled the challenge of determining stride parameters by introducing DiffStride. This method autonomously determines the number of strides in spectral poolings using backpropagation. Because spectral pooling performs cropping in the Fourier domain, DiffStride determines the optimal cropping box size via backpropagation. Despite improvements, both spectral pooling and DiffStride still omit detailed information from the feature map, which might reside in the truncated high-pass frequency. In our work, we address this issue by including both high-pass and low-pass information in the proposed unit.

C. WAVELET DOMAIN-BASED APPROACHES

Discrete wavelet transform (DWT)-based methods have emerged as an alternative approach for harnessing the wavelet domain in CNNs. By leveraging DWT or fast wavelet transform (FWT), these wavelet-centric pooling techniques enable CNN models to operate on downsampled features in the wavelet domain. Using DWT minimizes artifacts typically seen in neighborhood reduction techniques like max pooling. Importantly, feature map components decomposed from DWT can reconstruct the input without aliasing, and the components can also be selected to form a feature map. The use of DWT in neural networks as a pooling function was pioneered in [30], and it has since been incorporated into models for image classification tasks [21–23].

However, most studies have used only a subset of the decomposed wavelet components. For instance, [30] employed a second-order wavelet decomposition for pooling but reconstructed the image features with only the second-order wavelet sub-bands. Similarly, [21] revealed that WaveCNet, applied on the ImageNet1K dataset, predominantly used the approximation of the first-order decomposition for building feature maps. The idea of incorporating decomposed detail components to assemble sub-sampled feature maps was proposed in [22, 23]. Specifically, Wavelet-Attention CNN [22], tested on CIFAR10 and CIFAR100 datasets, found vertical and horizontal details to formulate an attention map, which was then overlaid on a feature map crafted from the DWT approximation component. Alternatively, [23] described the Convolutional-Wavelet Neural Network (CWNN) and its application to SAR images, using dual-tree complex wavelet transformation (DT-CWT) and averaged decomposed components for downsampling. The methodologies in [23] were integrated into a ResNet18 architecture for image classification on the ImageNet1K dataset, with performance benchmarks presented in [21]. In addition, there are only a few studies that have tried to make wavelets trainable, such as those by [24] and [25]. The research by [25] presented a wavelet loss function that enforces constraints on low-pass and high-

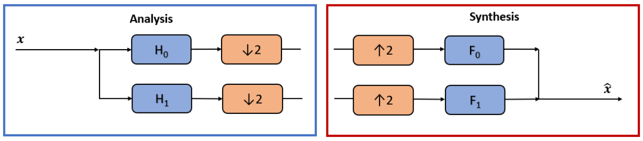

FIGURE 2. Analysis (left) and synthesis (right) parts of a two-channel filter bank architecture is illustrated. A signal can be decomposed into smaller components in the analysis part and reconstructed with the decomposed components in the synthesis part.

pass filters while preserving reversibility and symmetry in the trainable filters' coefficients. On the other hand, [24] introduced a method that relaxes the perfect reconstruction constraint and employs a perfect reconstruction loss function to train the filter bank's coefficients.

In this work, we introduce Orthogonal-LatticeUwU with tunable wavelet coefficients to improve the feature extraction process. To validate our proposed approach, we implement and test Orthogonal-LatticeUwU units on a wide range of datasets, including CIFAR10 [12], ImageNet1K [13], and DTD [7].

III. PROPOSED METHOD

We introduce Orthogonal-LatticeUwU, an orthogonal wavelet unit with tunable coefficients based on the lattice structure. Orthogonal-LatticeUwU units can be used as down-sampling and pooling units. Additionally, by pairing a non-stride convolution layer with Orthogonal-LatticeUwU units, it's possible to replace a stride convolution layer and still retain the detail components in the convolution output.

A. ORTHOGONAL-LATTICEUWU: AN INTEGRATION WITH LEARNABLE ORTHOGONAL WAVELET UNIT

Orthogonal-LatticeUwU is a unit that leverages both low-frequency and high-frequency components from a DWT analysis to find the optimal feature map. Instead of utilizing predefined wavelets, Orthogonal-LatticeUwU is characterized by trainable coefficients and is constructed using a lattice structure. In addition, the perfect reconstruction characteristic of DWT can be achieved through the analysis and synthesis components of a filter bank. A visualization of the filter bank structure is provided in Fig. 2, where the analysis and synthesis parts of the filter bank are shown in the blue and red rectangular boxes, respectively. For the analysis component, $\mathbf{H}_0$ and $\mathbf{H}_1$ are low-pass and high-pass filters, correspondingly. Conversely, $\mathbf{F}_0$ and $\mathbf{F}_1$ are low-pass and high-pass filters for the synthesis part, respectively. To achieve perfect reconstruction, the aliasing cancellation condition must be fulfilled, and there should be no distortion in the reconstructed signal. To satisfy the alias cancellation condition, given $\mathbf{h}_0 = [h(0), h(1), ..., h(N-1)]$ as coefficients of $\mathbf{H}_0$ with N taps, the coefficients of the other filters in the orthogonal filter bank can be deduced through sign alternating flip, order flip, and alternating signs relations [31], which can be expressed as follows:

$$\begin{cases} \textbf{Order Flip:} & \mathbf{f}_0(n) = \mathbf{h}_0(N-1-n) \\ \textbf{Sign Alternating Flip:} & \mathbf{h}_1(n) = (-1)^n \mathbf{h}_0(N-1-n) \\ \textbf{Alternating Sign:} & \mathbf{f}_1(n) = -(-1)^n \mathbf{h}_0(n), \end{cases} \quad (1)$$

where $\mathbf{f}_0$, $\mathbf{h}_1$, and $\mathbf{f}_1$ are filter coefficients of $\mathbf{F}_0$, $\mathbf{H}_1$, and $\mathbf{F}_1$, respectively. From the relations presented in Eq. (1), the filter bank satisfies the anti-aliasing condition. Moreover, with the aliasing cancellation condition, filter coefficients $\mathbf{h}_0$ of $\mathbf{H}_0$ is designed, which reduces the number of parameters needed for the analysis portion of a classification model. Then, in to find a $\mathbf{H}_0$ that ensures no distortion in the reconstructed signal, one approach is to impose the orthogonal structure in the filter by building it with lattice blocks [31], which can be expressed as follows:

$$\begin{aligned} \begin{bmatrix} \mathbf{H}_0(z) \\ \mathbf{H}_1(z) \end{bmatrix} &= \begin{bmatrix} \mathbf{H}_0(z) \\ -z^{-(N-1)}\mathbf{H}_0(-z^{-1}) \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \mathbf{R}_K \Lambda(z^2) \cdots \mathbf{R}_1 \Lambda(z^2) \mathbf{R}_0 \begin{bmatrix} 1 \\ z^{-1} \end{bmatrix}, \end{aligned} \quad (2)$$

where $\mathbf{R}_k$ is a rotation matrix constructing the filter with $k = 0, \cdots, K$. The delay matrices within the filter are represented by $\mathbf{\Lambda}(z^2)$. In addition, N is the order of the filter which can be defined as $N = 2K + 1$. In this work, to ensure the half-band condition, we use rotation matrices, which inherently are orthogonal matrices. Hence, $\mathbf{R}_k$ and $\mathbf{\Lambda}(z)$ can be mathematically expressed as follows:

$$\mathbf{R}_k = \begin{bmatrix} cos(\theta_k) & sin(\theta_k) \\ -sin(\theta_k) & cos(\theta_k) \end{bmatrix} = \begin{bmatrix} c_k & s_k \\ -s_k & c_k \end{bmatrix}. \quad (3)$$

$$\mathbf{\Lambda}(z) = \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix}. \quad (4)$$

In Eq. (3), θ_k is a rotation angle determining the coefficients of the wavelet filter bank with $k = 0, ..., K$. These rotation angles in the rotation matrices, where either their rows or columns are orthonormal to each other, are also termed lattice coefficients and determine the coefficients of the filter bank's filters. This orthonormality consequently ensures that the filters, which result from the multiplication of rotation and delay matrices, maintain orthogonality.

Instead of training the filter coefficients directly, the proposed method integrates these lattice coefficients as trainable parameters for the wavelet unit. With lattice coefficients, the filter bank is orthogonally structured and thus satisfies perfect reconstruction. Moreover, the number of lattice coefficients is $(K + 1)$, which is approximately half the number of filter coefficients N. This helps to reduce the number of trainable coefficients. In this work, the initialization of the lattice wavelet unit with trainable lattice coefficients needs two crucial subroutine procedures: analysis and synthesis.

- The **analysis procedure** computes the filter coefficients given the lattice coefficients.
- The **synthesis procedure** computes the lattice coefficients given the filter coefficients.

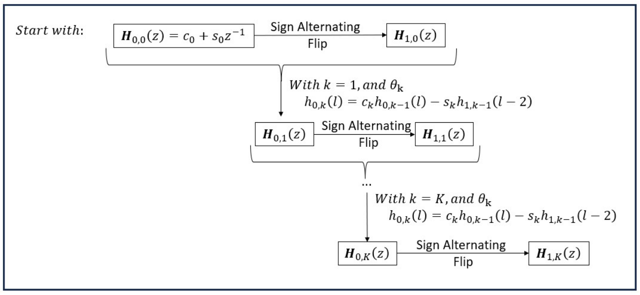
Start with:
$\boldsymbol{H}_{0,0}(z) = c_0 + s_0 z^{-1}$
Sign Alternating Flip
$\boldsymbol{H}_{1,0}(z)$
$With\ k = 1, and\ \theta_k$
$h_{0,k}(l) = c_k h_{0,k-1}(l) - s_k h_{1,k-1}(l-2)$
$\boldsymbol{H}_{0,1}(z)$
Sign Alternating Flip
$\boldsymbol{H}_{1,1}(z)$
...
$With\ k = K, and\ \theta_k$
$h_{0,k}(l) = c_k h_{0,k-1}(l) - s_k h_{1,k-1}(l-2)$
$\boldsymbol{H}_{0,K}(z)$
Sign Alternating Flip
$\boldsymbol{H}_{1,K}(z)$

FIGURE 3. Analysis Procedure diagram. In this diagram, Sign Alternating Flip can be implemented with $h_{1,k}(l) = (-1)^l h_{0,k}(N-1-l)$, and l is the coefficient index of the filter, which decreases by two after each iteration.

In the following section, we will derive analysis and synthesis procedures for image detection. Since the process is iterative (from stage k to $k+1$ for analysis and vice versa for synthesis), we include the stage index as part of filter notations. Specifically, $\mathbf{H}_{0,k}(z)$ and $\mathbf{H}_{1,k}(z)$ correspond to the low-pass and high-pass filters of the filter bank, constructed with $k+1$ lattice coefficients. The detailed analysis procedure is as follows:

$$\begin{bmatrix} \mathbf{H}_{0,k}(z) \\ -\mathbf{H}_{1,k}(z) \end{bmatrix} = \mathbf{R}_k \boldsymbol{\Lambda}(z^2) \mathbf{R}_{k-1} \boldsymbol{\Lambda}(z^2) \ldots \mathbf{R}_0 \begin{bmatrix} 1 \\ z^{-1} \end{bmatrix} \tag{5}$$

$$\leftrightarrow \begin{bmatrix} \mathbf{H}_{0,k}(z) \\ -\mathbf{H}_{1,k}(z) \end{bmatrix} = \mathbf{R}_k \boldsymbol{\Lambda}(z^2) \begin{bmatrix} \mathbf{H}_{0,k-1}(z) \\ -\mathbf{H}_{1,k-1}(z) \end{bmatrix} \tag{6}$$

$$\leftrightarrow \begin{bmatrix} \mathbf{H}_{0,k}(z) \\ -\mathbf{H}_{1,k}(z) \end{bmatrix} = \begin{bmatrix} c_k & s_k \\ -s_k & c_k \end{bmatrix} \begin{bmatrix} \mathbf{H}_{0,k-1}(z) \\ -z^{-2}\mathbf{H}_{1,k-1}(z) \end{bmatrix} \tag{7}$$

$$\leftrightarrow \begin{bmatrix} \mathbf{H}_{0,k}(z) \\ \mathbf{H}_{1,k}(z) \end{bmatrix} = \begin{bmatrix} c_k \mathbf{H}_{0,k-1}(z) - z^{-2} s_k \mathbf{H}_{1,k-1}(z) \\ s_k \mathbf{H}_{0,k-1}(z) + z^{-2} c_k \mathbf{H}_{1,k-1}(z) \end{bmatrix}. \tag{8}$$

As shown in Eq. (8), $\mathbf{H}_{0,k}(z)$ and $\mathbf{H}_{1,k}(z)$ are updated based on $\mathbf{H}_{0,k-1}(z)$ and $\mathbf{H}_{1,k-1}(z)$ from the previous stage and the rotation matrix with the lattice coefficient θ_k. In addition, for the case $k = 0$, we have:

$$\begin{bmatrix} \mathbf{H}_{0,0}(z) \\ \mathbf{H}_{1,0}(z) \end{bmatrix} = \begin{bmatrix} c_0 + z^{-1} s_0 \\ s_0 - z^{-1} c_0 \end{bmatrix}. \tag{9}$$

Algorithm 1 Analysis Procedure

Input: θ_k (trainable lattice coefficients for k from 0 to K)
Output: $H_{0,K}$ and $H_{1,K}$

```
for $k \leftarrow 0$ to $K$ do
    if $k = 0$ then
        $H_{0,0}(z) \leftarrow cos(\theta_0) + z^{-1} sin(\theta_0)$
        $H_{1,0}(z) \leftarrow sin(\theta_0) - z^{-1} cos(\theta_0)$
    else
        $H_{0,k}(z) \leftarrow cos(\theta_k) H_{0,k}(z) - z^{-2} sin(\theta_k) H_{1,k}$
        $H_{1,k}(z) \leftarrow -z^{-(2k+1)} H_{0,k}(-z^{-1})$
    end if
end for
return $H_{0,K}$ and $H_{1,K}$
```

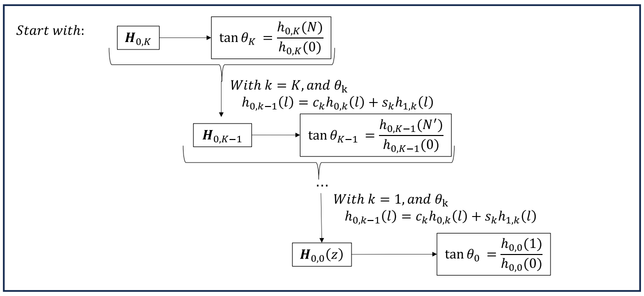
Start with:
$\boldsymbol{H}_{0,K}$
$\tan\theta_K = \frac{h_{0,K}(N)}{h_{0,K}(0)}$
$With\ k = K, and\ \theta_k$
$h_{0,k-1}(l) = c_k h_{0,k}(l) + s_k h_{1,k}(l)$
$\boldsymbol{H}_{0,K-1}$
$\tan\theta_{K-1} = \frac{h_{0,K-1}(N')}{h_{0,K-1}(0)}$
...
$With\ k = 1, and\ \theta_k$
$h_{0,k-1}(l) = c_k h_{0,k}(l) + s_k h_{1,k}(l)$
$\boldsymbol{H}_{0,0}(z)$
$\tan\theta_0 = \frac{h_{0,0}(1)}{h_{0,0}(0)}$

FIGURE 4. Synthesis Procedure diagram. In the diagram, l is the coefficient index of the filter, which decreases by two after each iteration.

With Eq. (8), Eq. (9) and the lattice coefficients, $\mathbf{H}_{0,k}(z)$ and $\mathbf{H}_{1,k}(z)$ can be found for $k = 0, \ldots, K$. Fig. 3 further illustrates this analysis procedure. In addition, the implementation of the analysis procedure is illustrated through a pseudo-code shown in Algorithm 1.

In the analysis procedure, $\mathbf{H}_{0,k}(z)$ and $\mathbf{H}_{1,k}(z)$ can be represented with tunable lattice coefficients θ_k. Hence, we need to derive a synthesis procedure to find the lattice coefficients θ_k of the predefined wavelets from their filter coefficients. In our approach, we use the Daubechies and Symlet wavelets as initial filters, as they are orthogonal wavelets and filter banks. Given $H_{0,k}(z)$ in Eq. (8), we can find the lattice coefficient as follows:

$$\begin{cases} z^{-L} : h_{0,k}(M) = s_k h_{0,k-1}(0) \\ z^0 : h_{0,k}(0) = c_k h_{0,k-1}(0) \end{cases} \tag{10}$$

$$\leftrightarrow \begin{cases} \mathrm{h}_{0,k-1}(0) = h_{0,k}(M)/s_k \\ \mathrm{h}_{0,k-1}(0) = h_{0,k}(0)/c_k \end{cases} \tag{11}$$

$$\leftrightarrow tan\theta_k = \frac{h_{0,k}(M)}{h_{0,k}(0)}, \tag{12}$$

where $M = (2k+1)$ is the order of the $H_{0,k}(z)$ filter. In addition, from Eq. (8), we also have the following:

$$\mathbf{H}_{0,k-1}(z) = c_k H_{0,k}(z) + s_k H_{1,k}(z). \tag{13}$$

From Eq. (12) and Eq. (13), we can find the lattice coefficients from a given set of filter coefficients. The synthesis

Algorithm 2 Synthesis Procedure

Input: $h_{0,l}$ (wavelet coefficients of $H_{0,K}$ with l from 0 to K)
Output: θ_k (lattice coefficient values for k from 0 to K)

```
for $k \leftarrow 0$ to $K$ do
    $H_{1,k}(z) \leftarrow -z^{-(2k+1)} H_{0,k}(-z^{-1})$
    $\theta_k \leftarrow arctan(\frac{h_{0,k}(2k+1)}{h_{0,k}(0)})$
    for $l \leftarrow 0$ to $2k-1$ do
        $h_{0,k-1}(l) \leftarrow cos(\theta_k) h_{0,k}(l) + sin(\theta_k) h_{1,k}(l)$
    end for
end for
return $\theta_k$ (lattice coefficient values for $k$ from 0 to $K$)
```

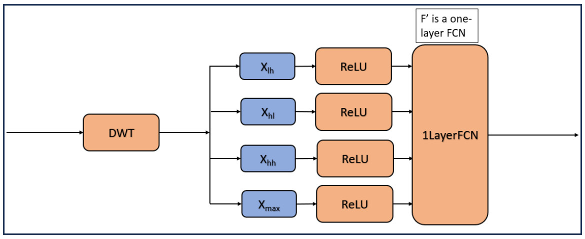

FIGURE 5. Diagram of low-pass and high-pass component implementation. The signal goes from left to right. The results from DWT go to ReLU functions to become the inputs of a one-layer FCN. Because an FCN can take inputs of arbitrary sizes, the one-layer FCN can read the decomposed components and finetune the trainable coefficients to optimally combine the decomposed components. The fine-tuned one-layer FCN combines the inputs to find the optimal feature map.

procedure is demonstrated in Fig. 4. In addition, the implementation of the synthesis procedure is illustrated through a pseudo-code shown in Algorithm 2.

In this work, we fine tune the lattice coefficients in our proposed units from the filter coefficients of Haar, Daubechies, and Symlet wavelets. As these predefined wavelets are designed for signal representation and have maximal smoothness in the basis function, we relax this condition to improve classification performance. As only one level of decomposition is used in this work, smoothness of basis function is not needed. Furthermore, since orthogonal filter banks and their lattice structure are used, we minimize the number of parameters in the Orthogonal-LatticeUwU and maintain their perfect reconstruction. In the initialization step of the Orthogonal-LatticeUwU method, we first employ the synthesis procedure to extract lattice coefficients from the filter coefficients of the predefined wavelets. Subsequently, the analysis procedure is utilized to determine the filter coefficients.

B. 2D IMPLEMENTATION

From the low-pass and high-pass filters $H_{0,K}$ and $H_{1,K}$, represented with the trainable lattice coefficients found with the synthesis and analysis procedures, we compute the high-pass and low-pass filter matrices, denoted as $\mathbf{H}$ and $\mathbf{L}$. These matrices are used to find the approximation $\mathbf{X}_{ll}$ as well as the detail components $\mathbf{X}_{lh}$, $\mathbf{X}_{hl}$, and $\mathbf{X}_{hh}$. The computation of $\mathbf{L}$ can be mathematically described as follows:

$$\mathbf{L} = \mathbf{D}\widehat{\mathbf{H}}, \tag{14}$$

where $\mathbf{D}$ is the downsampling matrix and $\widehat{\mathbf{H}}$ is a Toeplitz matrix with filter coefficients of $\mathbf{H}_0(z)$. $\mathbf{H}$ has a similar form as $\mathbf{L}$ with filter coefficients of $\mathbf{H}_0(z^{-1})$. Using $\mathbf{H}$ and $\mathbf{L}$, $\mathbf{X}_{ll}$, $\mathbf{X}_{lh}$, $\mathbf{X}_{hl}$, and $\mathbf{X}_{hh}$ are computed as follows:

$$\begin{aligned} \mathbf{X}_{ll} &= \mathbf{L}\mathbf{X}\mathbf{L}^T, \quad \mathbf{X}_{lh} = \mathbf{H}\mathbf{X}\mathbf{L}^T, \\ \mathbf{X}_{hl} &= \mathbf{L}\mathbf{X}\mathbf{H}^T, \quad \mathbf{X}_{hh} = \mathbf{H}\mathbf{X}\mathbf{H}^T. \end{aligned} \tag{15}$$

C. ONE-LAYER FCN IN THE COMBINATION OF LOW-PASS AND HIGH-PASS COMPONENTS

In this work, we use a one-layer Fully Convolutional Network (FCN) to combine features from the sub-sample low-pass and high-pass components extracted via the Discrete Wavelet Transform (DWT). Nevertheless, since one-layer FCNs apply a weight to every feature to find the optimal feature map, this increases the number of trainable parameters and the computational complexity of the unit.

Conventional max pooling methods keep the most prominent features in a sub-sampled feature map after the pooling process and discard the high-pass features, which may contain critical information. To strike a balance, we harness both the approximation (the low-pass component) and the details (the high-pass components) from DWT results. This is achieved by applying a one-layer FCN to find the optimal feature map with the following decomposed components as inputs: $\mathbf{X}_{ll}$, $\mathbf{X}_{lh}$, $\mathbf{X}_{hl}$, and $\mathbf{X}_{hh}$ from DWT.

With the stated motivation, we design the proposed unit with tunable parameters by applying a one-layer FCN. Hence, the final feature map is a combination of the approximated features and vertical, horizontal, and diagonal detailed features from DWT. The tunable parameters enable us to find an optimal feature map based on the low-pass features and vertical, horizontal, and diagonal detailed features. Therefore, the weights for the combination are fine-tuned and optimized through back propagation during the training process. The unit can be mathematically expressed as follows:

$$\begin{aligned} X_p = F'(&ReLU(\mathbf{X}_{ll}), ReLU(\mathbf{X}_{lh}), \\ &ReLU(\mathbf{X}_{hl}), ReLU(\mathbf{X}_{hh})), \end{aligned} \tag{16}$$

where F' is the one-layer FCN with tunable weights. Because an FCN can take inputs of arbitrary size [32], the one-layer FCN can read the decomposed components and fine tune the trainable coefficients to optimally combine the decomposed components. As shown in Eq. (16), the decomposed detail components via DWT along with the prominent feature map are first independently processed with rectified linear unit (ReLU) functions. The implementation of the component combination is shown in Fig. 5.

D. IMPLEMENTATION IN CNN ARCHITECTURES

We integrate Orthogonal-LatticeUwU units into a ResNet architecture. For the downsampling and pooling layers, we use the decomposed components along with the max-pooling results as inputs for a one-layer FCN. In addition, we substitute the two-stride convolution with a non-stride convolution

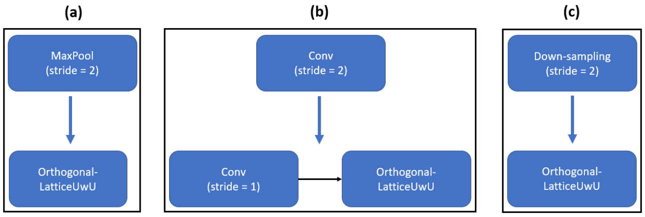

FIGURE 6. Implementation of Orthogonal-LatticeUwU in CNN architecture. The diagram shows how Orthogonal-LatticeUwU can be used to replace max-pooling (a), stride-convolution (b), and downsampling (c) functions in CNNs.

block followed by the proposed units. The general implementation of the proposed unit is shown in Fig. 6. In this work, as we mainly apply Orthogonal-LatticeUwU in a ResNet architecture. The implementation of Orthogonal-LatticeUwU in a ResNet block is illustrated in Fig. 7.

We implement the proposed method on the ResNet family of architectures and then train and test it on CIFAR10 [12], ImageNet1K [13], and DTD [7] datasets. We train our proposed units on ResNet18 with the proposed units for wavelets with 2, 4, 6 and 8 coefficients. To initialize Orthogonal-LatticeUwU's trainable parameters, we use Haar coefficients for two taps, Daubechies (DB) 2 for four taps, DB3 for six taps, and DB4 and Symlet4 for eight taps [33]. These wavelet types are the two classes of orthogonal wavelets and they are used to initialize the lattice coefficients. These initializing coefficients and the optimized ones are shown in Table 1. As shown in Table 1, most of the coefficients have changed after optimization, except for the Haar wavelet. In parallel, we also train the network with the corresponding WaveCNet [21] models. For CIFAR10 and ImageNet1K, the best results are then compared with the reported performances of the baseline ResNet18, WaveCNet ResNet18, CWNN-ResNet18 in [21], SpectralPool-ResNet18 and DiffStride-ResNet18 from [20], PR-relaxation ResNet18 from [24], and WaveletAttention CNN ResNet18 (WA-CNN-ResNet18) described in [22]. We broaden our evaluation to ResNet34 and ResNet50 architectures on the DTD and CIFAR10 datasets. For ResNet50 on CIFAR10, we also include the LDW-Pooling ResNet50 performance reported in [25]. Then, we apply the proposed units in the encoder of CFLOW-AD [34] pipeline for anomaly detection in the hazelnut category on MVTec AD [14, 15].

TABLE 1. Coefficients of Haar, Daubechies, and Symlets (third column) and the optimized filters (fourth column). D and S denote Daubechies and Symlets, respectively.

N		Index	Original	Optimized
2	Haar	0	0.7071	0.7071
		1	0.7071	0.7071
4	D	0	0.4830	0.5000
		1	0.8365	0.8660
		2	0.2241	0.9659
		3	-0.1294	-0.2588
6	D	0	0.3327	0.3812
		1	0.8069	0.9245
		2	0.4599	0.8777
		3	-0.1350	-0.4793
		4	-0.0854	0.9944
		5	0.0352	0.1053
8	D	0	0.2304	0.3067
		1	0.7148	0.9518
		2	0.6309	0.7767
		3	-0.0280	-0.6299
		4	-0.1870	0.9680
		5	0.0308	0.2510
		6	0.0329	0.9989
		7	-0.0106	-0.0460
	S	0	0.0322	0.9313
		1	-0.0126	-0.3643
		2	-0.0992	0.1420
		3	0.2979	0.9899
		4	0.8037	0.6227
		5	0.4976	0.7824
		6	-0.0296	0.3914
		7	-0.0758	-0.9202

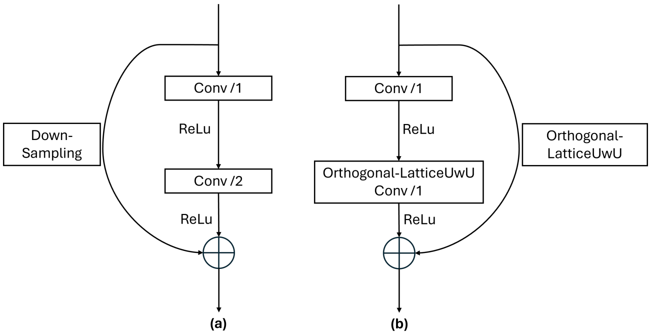

FIGURE 7. Implementation of the proposed Orthogonal-LatticeUwU unit in ResNet architecture. Diagram (a) shows the original ResNet block [6], while diagram (b) demonstrates the ResNet block implemented with the proposed Orthogonal-LatticeUwU.

IV. EXPERIMENTS

In this work, we design Orthogonal-LatticeUwU units with tunable coefficients so that we can relax the zero-at-π condition hold of predefined wavelets, such as Daubechies and Symlets. We first examine this effect with pole-zero plots of the filters from the original DB2 and from Orthogonal-LatticeUwU trained with DB2 initialization. The pole-zero and frequency-response plots of the original DB2 filter are shown in Fig. 8, and the same plots of the Orthogonal-LatticeUwU unit trained with DB2 initialization are illustrated in Fig. 9. As it can be observed in Fig. 8 and Fig. 9, the tunable coefficients of Orthogonal-LatticeUwU relax the zero-at-π condition after training when compared to the plot of the original DB2 wavelet. We then applied the Orthogonal-LatticeUwU units in the ResNet18 architecture and examined the effects by comparing performance between baseline and WaveCNet on CIFAR10 [12], ImageNet1K [13], and DTD [7] datasets. The CIFAR10 dataset comprises 60,000 32 x 32 color images distributed across 10 classes, with 6,000 images per class. The training set consists of 50,000 images, while the test set comprises 10,000 images. On the other hand, the ImageNet1K dataset is significantly larger, containing over 1,000 object classes. It includes 1,281,167 training images, 50,000 validation images, and 100,000 test images. Additionally, the average image resolution in ImageNet1K is 469 x 387 pixels. Furthermore, the DTD dataset serves as a texture database, containing 5,640 images across 47 categories, each with 120 images. In DTD, an equal number of images are allocated for training, validation, and test sets within each category. The image sizes in the DTD dataset range between 300 x 300 and 640 x 640 pixels. For Orthogonal-LatticeUwU, the 2, 4, 6, and 8-tap units with tunable coefficients are initialized with Haar, DB2, DB3, and DB4 wavelet coefficients, respectively. Additionally, Sym4 coefficients were used to train Orthogonal-LatticeUwU 8-tap units. Then, the best ResNet18-based models integrated with Orthogonal-LatticeUwU were selected and compared with other reported models on CIFAR10 and

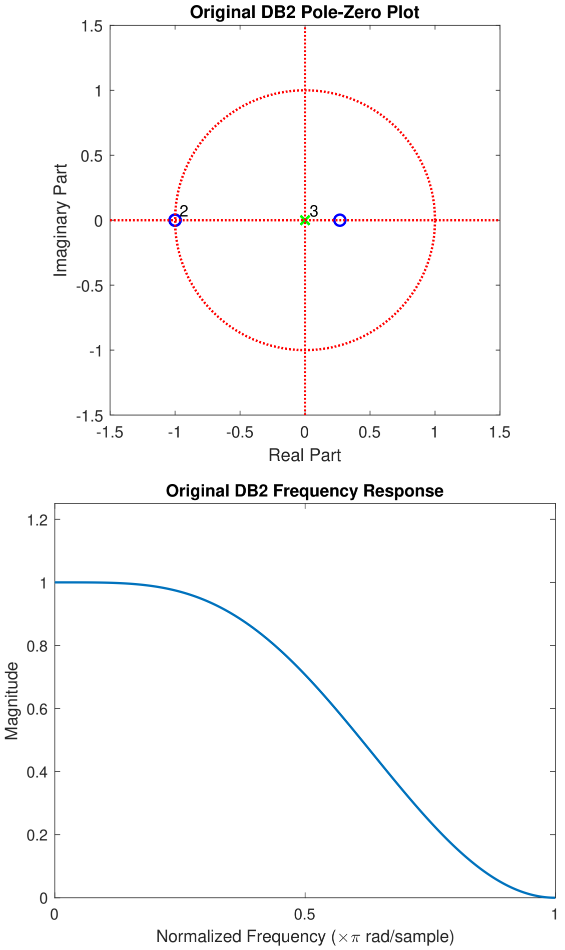

FIGURE 8. Pole-zero and frequency response plots of the original DB2 are shown in the top and bottom sub-figures, respectively. The pole-zero plot of Original DB2 with the maximum number of zeros at π, and the frequency response plot shows a low-pass filter response.

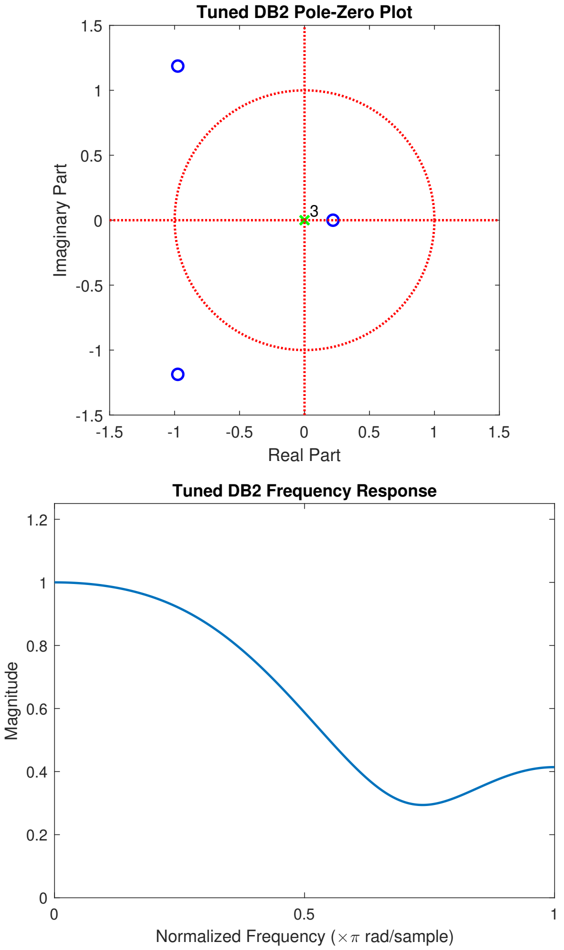

FIGURE 9. Pole-zero and frequency response plots of the Orthogonal-LatticeUwU with DB2 initialization are shown in the top and bottom sub-figures, respectively. The pole-zero plot of the unit shows the relaxation of the zero-at-π condition, and the frequency-response shows the change of the filter's response.

ImageNet1K. Furthermore, we extend our study to ResNet34 and ResNet50 for DTD and CIFAR10 to see if improvements seen in ResNet18 models can still be observed. To train the ResNet-family-based CNN models, we set up the same training procedure used in [6, 21] with random translation and rotation as augmentation techniques and a cross-entropy loss function. We also train the models from scratch by using stochastic gradient descent (SGD) with a batch size of 256 and an initial learning rate of 0.1. Finally, we apply our best Orthogonal-LatticeUwU ResNet18 units as encoders for the CFLOW-AD pipeline to detect anomalies in hazelnut images from the MVTec AD dataset [14, 15]. We evaluate the performance of the Orthogonal-LatticeUwU encoders against baseline and WaveCNet ResNet18 encoders based on detection and segmentation area under the curve (AUC) metrics and their corresponding heat maps. In this study, we conducted model implementations, including our own models, models from other studies, and baseline models, on a GTX 1080 TI machine for both training and testing phases.

A. IMAGE CLASSIFICATION WITH RESNET18-BASED ARCHITECTURE

In this section, we show the performance of Orthogonal-LatticeUwU implementation with CNNs as demonstrated in Fig. 6 on CIFAR10 [12], ImageNet1K [13], and DTD [7] for the classification task. CIFAR10 is a set of images with low resolution; ImageNet1K is a dataset of high-resolution images, and DTD focuses on high-resolution images that include large amounts of textures and details. In this experiment, the proposed units are applied on the ResNet18 architecture and work as pooling, down-sampling, and stride-convolution functions. For Orthogonal-LatticeUwU, the 2, 4, 6, and 8-tap units with tunable coefficients are trained with the coefficients initialized with Haar, DB2, DB3, and DB4 wavelet coefficients. In addition, Sym4 coefficients are used to initialize Orthogonal-LatticeUwU 8-tap. With trained models integrated with our units, the performance of the various Orthogonal-LatticeUwU models is compared with the baseline and WaveCNet models.

TABLE 2. Accuracy of Orthogonal-LatticeUwU ResNet 18 models with different number of taps and initialization-wavelet types trained on CIFAR10.

Wavelet	Accuracy(%)	
None (Baseline)	92.44	
	Orthogonal-LatticeUwU ResNet18 (ours)	WaveCNet ResNet18 [21]
2-tap Haar	**94.97**	94.76
4-tap DB2	**95.05**	94.93
6-tap DB3	**95.03**	94.56
8-tap DB4	**95.11**	93.81
8-tap Sym4	**94.99**	94.84

For the image classification task in this study, we use accuracy as the main evaluation metric. The accuracy metric can be defined as follows:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}. \tag{17}$$

The comparison shows that the proposed units improve the baseline's performance. The accuracy from proposed units, in most cases, is better than the accuracy from WaveCNet models. Then, the best performance of Orthogonal-LatticeUwU is compared with the other reported performances of Wavelet-Attention CNN ResNet18 [22] (WA-CNN-ResNet18), SpectralPool-ResNet18, DiffStride-ResNet18 [20], Convolutional Wavelet Neural Network ResNet18 (CWNN-ResNet18) [21], wavelet unit with perfect reconstruction relaxation ResNet18 (PR-relaxation ResNet18) [24] on CIFAR10 and ImageNet1K datasets. As we could not find other reported works using ResNet18-based architecture on DTD, we do not make the same comparison analysis on the DTD set. Comparison results show that models with the proposed Orthogonal-LatticeUwU achieve the best performance.

1) On CIFAR10 with low-resolution images

In this experiment, the models are trained on CIFAR10 for 200 epochs with randomly horizontally flipped and randomly shifted images. The same training pipeline and setup used in other methods is also used in the comparison. In general, the proposed Orthogonal-LatticeUwU method shows an improvement over the baseline ResNet18 modelthat is comparable to or better than WaveCNet. As shown in Table 2, all of the Orthogonal-LatticeUwU models outperform the baseline by at least 2.53%. In addition, the Orthogonal-LatticeUwU models outperform the WaveCNet models by an additional 0.21–1.3%. The best performing Orthogonal-LatticeUwU model is the Orthogonal-LatticeUwU with 8-Tap DB4 ResNet18 model, which achieved 95.11% accuracy, whereas the best performance observed by WaveCNet is 94.93% accuracy achieved with the WaveCNet DB2 ResNet model.

2) On ImageNet1K with high-resolution images

In this experiment, the models are trained on the ImageNet1K dataset for 90 epochs with randomly horizontally flipped and randomly cropped images. The same training pipeline and setup that was used in other approaches is also used in the comparison and analysis. In general, the proposed Orthogonal-LatticeUwU methods show a clear improvement to the baseline and achieve a competitive performance to the WaveCNet models. As shown in Table 3, the best performance comes from the Orthogonal-LatticeUwU 2-tap Haar ResNet18 model, which showed accuracy of 71.61%. In addition, the performance results from Orthogonal-LatticeUwU and WaveCNet are comparable to each other and bring consistent improvements to the baseline in all cases.

TABLE 3. Accuracy of Orthogonal-LatticeUwU with different number of taps and initialization-wavelet types for ResNet18 trained on ImageNet1K.

Wavelet	Accuracy(%)	
None (Baseline)	69.76	
	Orthogonal-LatticeUwU ResNet18 (ours)	WaveCNet ResNet18 [21]
2-tap Haar	**71.61**	71.47
4-tap DB2	71.30	**71.48**
6-tap DB3	70.80	**71.08**
8-tap DB4	**70.63**	70.35
8-tap Symlet4	71.38	**71.42**

TABLE 4. Accuracy of Orthogonal-LatticeUwU with different number of taps and initialization-wavelet types for DTD.

Wavelet	Accuracy(%)	
None (Baseline)	33.99	
	Orthogonal-LatticeUwU ResNet18 (ours)	WaveCNet ResNet18 [21]
2-tap Haar	**40.37**	25.53
4-tap DB2	**43.51**	23.62
6-tap DB3	**40.59**	24.36
8-tap DB4	**40.80**	26.91
8-tap Symlet4	**40.16**	35.27

3) On DTD with high-resolution textural images

On DTD, the models are trained for 700 epochs in total with randomly horizontally flipped and randomly cropped images. In this experiment, the proposed Orthogonal-LatticeUwU method shows a clear improvement to the baseline model and achieves outstanding performance compared to the WaveCNet models. As shown in Table 4, Orthogonal-LatticeUwU consistently achieves improvement compared with the baseline model.

Results from the CIFAR10, ImageNet1K, and DTD experiments show that the Orthogonal-LatticeUwU models achieve higher performance improvement over the baseline than WaveCNet models when implemented in a ResNet18 architecture, with the best model from Orthogonal-LatticeUwU outperforming the best model from WaveCNet in all three experiments. In addition, significant improvements in the performance of the Orthogonal-LatticeUwU models were seen in the DTD experiments, when compared to the performance of WaveCNet models. This may be explained by Orthogonal-LatticeUwU's utilization of both low-pass and high-pass components when processing DTD images, which are rich in details and textural features, whereas the WaveCNet models only use low-pass features.

In these experiments, models were implemented on a GTX 1080 TI machine. Table 5 provides information on the average inference time of the proposed methods, WaveCNet models, and the baseline on DTD dataset with a batch size of 256. Based on the average inference time reported on DTD, the inference time values of the proposed methods are comparable with the baseline's inference time. However, the numbers of parameters associated with the proposed method is a potential drawback. On DTD, the number of parameters in the baseline ResNet18 is 11,200,623, and the number of parameters in Orthogonal-LatticeUwU ResNet18 is 21,022,845. This higher number is attributed to the utilization of FCN for every pixel in the feature maps during the feature optimization process.

4) Comparison with other approaches on CIFAR10 and ImageNet1K

In this section, we select the Orthogonal-LatticeUwU models with the best performance shown in Table 2 and compare them with the reported performance of Wavelet-Attention CNN ResNet18 [22] (WA-CNN-ResNet18), SpectralPool-ResNet18, and DiffStride-ResNet18 from [20]. The best performance results on CIFAR10 are shown in Table 6. As shown in Table 6, the Orthogonal-LatticeUwU 8-TapDB4 ResNet18 model achieves better performance than other spectrum and wavelet-based approaches, except for the PR-relaxation Symlet2 ResNet18, to which our method has comparable performance. For the ImageNet1K experiment, the best results from Table 3 are compared with the reported performance of SpectralPool-ResNet18 and DiffStride-ResNet18 from [20], the best performance from WaveCNet-ResNet18-DB2 [21], and CWNN-ResNet18 [21], as shown in Table 7. Results show that the Orthogonal-LatticeUwU 2-tap Haar ResNet18 model achieves better performance than what has been reported for other approaches.

B. EXTENSION STUDY FOR IMAGE CLASSIFICATION WITH RESNET34 AND RESNET50 ARCHITECTURES ON CIFAR10 AND DTD

On DTD, the proposed Orthgonal-LatticeUwU unit is applied to ResNet34 and ResNet50 and compared with the corresponding WaveCNet network for 2-tap and 4-tap cases with Haar and DB2, respectively. Performance results are shown in Table 8. From Table 8, Orthogonal-LatticeUwU achieves the best performance on DTD with a clear and significant improvement to the baseline in both 2-tap and 4-tap experiments. Orthogonal-LatticeUwU is also applied to ResNet34 and ResNet50 and trained on the CIFAR10 dataset. The performance of the models is compared to that of other approaches and the baseline model. In addition, the reported performance of LDW-Pooling ResNet50 on CIFAR10 [25] is also included for the comparison. The performance results of these models are shown in Table 9. A drop in performance in all models is observed in the DTD study when the depth of the network is increased. This is because there is a lack of data in the training set for DTD. In the case of CIFAR10, with the ResNet34 architecture, Orthogonal-LatticeUwU has better accuracy than WaveCNet in both 2-tap and 4-tap cases. With ResNet50, Orthogonal-LatticeUwU's improvement to the baseline is also higher than that of WaveCNet for the 2-tap and 4-tap cases.

TABLE 5. Average Inference time of Orthogonal-LatticeUwU ResNet18, WaveCNet ResNet18, and the baseline model with different numbers of taps on DTD dataset for a batch size of 256 on a GTX 1080 TI machine.

Wavelet	**Inference Time (second)**	
None (Baseline)	0.021	
	Orthogonal-LatticeUwU ResNet18 (ours)	WaveCNet ResNet18 [21]
2-tap Haar	0.023	0.032
4-tap DB2	0.021	0.030
6-tap DB3	0.022	0.047
8-tap DB4	0.029	0.026
8-tap Symlet4	0.036	0.021

TABLE 6. Accuracy of the best Orthogonal-LatticeUwU models compared to other approaches with ResNet18 architecture on CIFAR10.

Models	**Accuracy(%)**
baseline-ResNet18 [6]	92.44
SpectralPool-ResNet18 [19]	92.50 (+0.06)
DiffStride-ResNet18 [20]	92.90 (+0.46)
WA-CNN-ResNet18 [22]	92.57 (+0.13)
WaveCNet-ResNet18 sym4 [21]	94.84 (+2.40)
PR-relaxation Symlet2 ResNet18 [24]	**95.13 (+2.69)**
Orthogonal-LatticeUwU 8-tap DB4 ResNet18 (ours)	95.11 (+2.67)

TABLE 7. Accuracy of the best Orthogonal-LatticeUwU models compared to other approaches with ResNet18 architecture on ImageNet1K.

Models	**Accuracy(%)**
baseline-ResNet18 [6]	69.76
SpectralPool-ResNet18 [19]	69.93 (+0.17)
DiffStride-ResNet18 [20]	69.72 (-0.04)
WaveCNet-ResNet18-DB2 [21]	71.48 (+1.72)
CWNN-ResNet18 [23]	70.06 (+0.3)
Orthogonal-LatticeUwU 2-tap Haar ResNet18 (ours)	**71.61 (+1.85)**

TABLE 8. Accuracy of Orthogonal-LatticeUwU with different number of taps and initialization-wavelet types on ResNet34 and ResNet50 for DTD.

Wavelet	**ResNet34**	
None (Baseline)	24.47%	
	Orthogonal-LatticeUwU (ours)	WaveCNet [21]
2-tap Haar	**41.49%**	32.39%
4-tap DB2	**38.88%**	32.23%
Wavelet	**ResNet50**	
None (Baseline)	20.74%	
	Orthogonal-LatticeUwU (ours)	WaveCNet [21]
2-tap Haar	**37.34%**	19.15%
4-tap DB2	**35.90%**	30.21%

TABLE 9. Accuracy of Orthogonal-LatticeUwU with different number of taps and initialization-wavelet types on ResNet34 and ResNet50 for CIFAR10.

Wavelet	**ResNet34**	
None (Baseline)	94.33%	
	Orthogonal-LatticeUwU (ours)	WaveCNet [21]
2-tap Haar	**95.44%**	95.07%
4-tap DB2	**95.61%**	95.12%
Wavelet	**ResNet50**	
None (Baseline)	94.09%	
LDW-Pooling ResNet50 [25]	92.13%	
	Orthogonal-LatticeUwU (ours)	WaveCNet [21]
2-tap Haar	**94.43%**	94.31%
4-tap DB2	**94.60%**	94.09%

C. AS THE ENCODER OF CFLOW-AD ON MVTEC AD (HAZELNUT)

In this experiment, we evaluate the proposed method alongside other models using the MVTec AD dataset. As previ-

TABLE 10. Segmentation and Detection AUROCs of CFLOW-AD pipeline with the Orthogonal-LatticeUwU, WaveCNet, and baseline ResNet18 encoders for hazelnut category in MVTec AD.

Models	Segmentation AUROC	Detection AUROC
Baseline ResNet18 CFLOW-AD	96.45%	**92.46**%
Orthogonal-LatticeUwU 2-tap Haar ResNet18 CFLOW-AD	**97.15**%	89.57%
WaveCNet Sym4 ResNet18 CFLOW-AD	96.20%	87.61%

ously mentioned, MVTec AD is specifically designed for bench-marking anomaly detection methods, with a focus on industrial inspection [14, 15]. In this experiment, the area under the receiver operating characteristic (AUROC) curve is utilized to assess the performance of the models both at the image-level (for anomaly detection) and at the pixel-level segmentation (for localization). ResNet18 with the proposed unit is used as the encoder in the CFLOW-AD [34] pipeline for the anomaly detection task on hazelnut images from MVTec AD [14, 15], which shows a comparable performance to the baseline ResNet18 along with the WaveCNet ResNet18 encoders. The models are evaluated with segmentation and detection AUROCs, shown in Table 10. For the result, the CFLOW pipeline with the Orthogonal-LatticeUwU 4-Tap ResNet18 encoder has the best segmentation performance and the second-best detection result. Defect detection result examples are also visualized in Fig. 10. From Fig. 10, we can see that the heat maps from Orthogonal-LatticeUwU show a more localized result than baseline and WaveCNet.

V. CONCLUSION

We develop Orthogonal-LatticeUwU, a wavelet unit with a built-in learnable orthogonal wavelet unit constructed with the lattice structure. With the tunable coefficients, we relax the zero-at-π condition, which enforces a maximum number of zeros at π. The decomposed components from Orthogonal-LatticeUwU are used as inputs for a one-layer FCN to find the optimal feature map for CNNs. The proposed technique is implemented on ResNet family architectures, which achieve competitive performance on CIFAR10 and a noticeable improvement on ImageNet1K and DTD. The proposed method is also used in the ResNet18 encoder of the CFLOW-AD pipeline for the anomaly detection task on hazelnut objects, which also shows a promising performance improvement, as well as more accurate and localized heat maps. The results from the proposed method implemented across various datasets demonstrate its competitive performance on normal images and excellent performance on images with textural and detailed features. This advantage has been shown to benefit the detection of anomaly patterns and features in manufactured products. In addition, the proposed method also has competitive inference time compared to the baseline. Note that the proposed methods are currently used in image classification and anomaly detection tasks. In future work, we plan to extend this work in other tasks, such as detection and segmentation. The feature optimization process in our future work will also be simplified to reduce the number of trainable parameters. Instead of using a one-layer FCN, which applies a weight for each feature, we will apply weights only to the four decomposed components, reducing computational complexity. In the next study, we will also relax the orthogonality constraint by using a biorthogonal lattice structure, which still maintains the perfect reconstruction constraint and allows more freedom to fine-tune the wavelet coefficients.

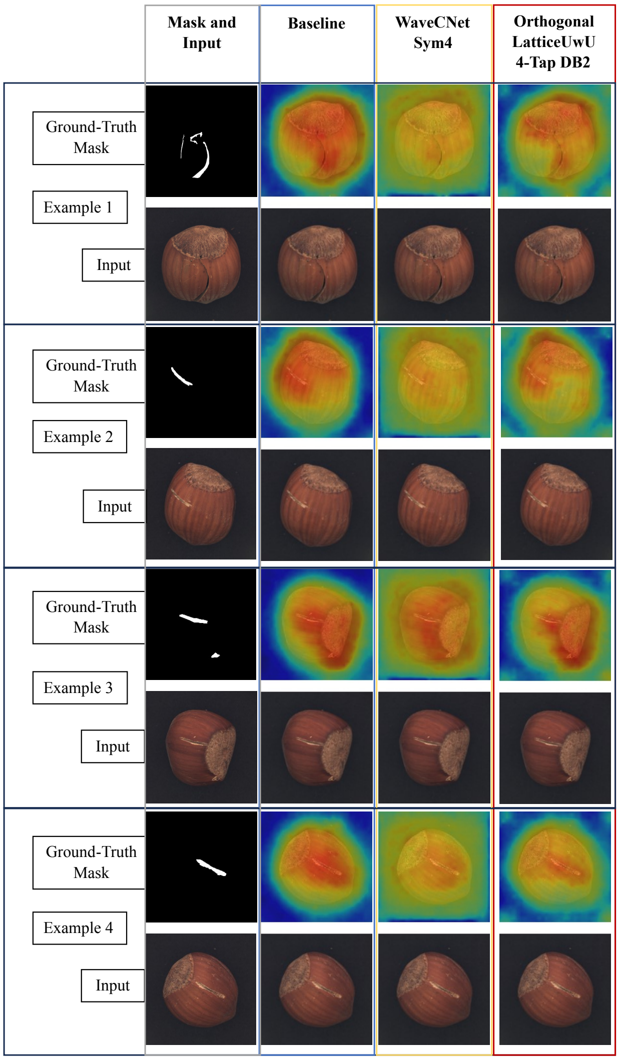

FIGURE 10. Anomaly detection for hazelnut objects in the MVTec AD dataset with four examples. From left to right: the first column shows the mask and input of each example, and the second, third, fourth, and fifth columns show the corresponding heat-maps and input images from baseline, WaveCNet sym4, and Orthogonal-LatticeUwU 4-tap DB2, respectively.

REFERENCES

[1] A. Zafar, M. Aamir, N. Mohd Nawi, A. Arshad, S. Riaz, A. Alruban, A. K. Dutta, and S. Almotairi, "A comparison of pooling methods for convolutional neural networks," *Applied Sciences*, vol. 12, no. 17, 2022.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[3] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2017.

[4] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural

networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.

[5] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[7] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi, "Describing textures in the wild," in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[8] D. A. Pham, A. D. Le, D. T. Pham, and H. B. Vo, "Alerttrap: On designing an edge-computing remote insect monitoring system," in *2021 8th NAFOSTED Conference on Information and Computer Science (NICS)*, pp. 323–328, 2021.

[9] Q. M. Nguyen, D. A. Pham, D. T. Pham, A. D. Le, N. Q. H. Vo, and H. B. Vo, "Smarttrap: An on-field insect monitoring system empowered by edge computing capabilities," in *2023 RIVF International Conference on Computing and Communication Technologies (RIVF)*, pp. 77–82, 2023.

[10] A. Cannet, C. Simon-chane, A. Histace, M. Akhoundi, O. Romain, M. Souchaud, P. Jacob, D. Sereno, P. Bousses, and D. Sereno, "An annotated wing interferential pattern dataset of dipteran insects of medical interest for deep learning," *Scientific Data*, vol. 11, Jan 2024.

[11] N. Chitsaz, R. Marian, and J. Chahl, "Experimental method for 3d reconstruction of odonata wings (methodology and dataset)," *PLOS ONE*, vol. 15, Apr 2020.

[12] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[13] 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," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[14] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, "Mvtec ad — a comprehensive real-world dataset for unsupervised anomaly detection," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9584–9592, 2019.

[15] P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, and C. Steger, "The mvtec anomaly detection dataset: A comprehensive real-world dataset for unsupervised anomaly detection," *International Journal of Computer Vision*, vol. 129, no. 4, p. 1038–1059, 2021.

[16] C. Özdemir, "Avg-topk: A new pooling method for convolutional neural networks," *Expert Systems with Applications*, vol. 223, p. 119892, 2023.

[17] J. Hyun, H. Seong, and E. Kim, "Universal pooling – a new pooling method for convolutional neural networks," *Expert Systems with Applications*, vol. 180, p. 115084, 2021.

[18] E. A. Mohamed, T. Gaber, O. Karam, and E. A. Rashed, "A novel cnn pooling layer for breast cancer segmentation and classification from thermograms," *PLOS ONE*, vol. 17, Oct 2022.

[19] O. Rippel, J. Snoek, and R. P. Adams, "Spectral representations for convolutional neural networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, (Cambridge, MA, USA), p. 2449–2457, MIT Press, 2015.

[20] R. Riad, O. Teboul, D. Grangier, and N. Zeghidour, "Learning strides in convolutional neural networks," *ICLR*, 2022.

[21] Q. Li, L. Shen, S. Guo, and Z. Lai, "Wavelet integrated cnns for noise-robust image classification," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7243–7252, 2020.

[22] X. Zhao, P. Huang, and X. Shu, "Wavelet-attention cnn for image classification," *Multimedia Syst.*, vol. 28, p. 915–924, jun 2022.

[23] Y. Duan, F. Liu, L. Jiao, P. Zhao, and L. Zhang, "Sar image segmentation based on convolutional-wavelet neural network and markov random field," *Pattern Recognition*, vol. 64, pp. 255–267, 2017.

[24] A. D. Le, S. Jin, Y. S. Bae, and T. Nguyen, "A novel learnable orthogonal wavelet unit neural network with perfection reconstruction constraint relaxation for image classification," in *2023 IEEE International Conference on Visual Communications and Image Processing (VCIP)*, pp. 1–5, 2023.

[25] J.-W. Hsieh, M.-C. Chang, B. Wang, P.-Y. Chen, L. Ke, and S. Lyu, "Learnable discrete wavelet pooling (ldw-pooling) for convolutional networks," in *British Machine Vision Conference*, 2021.

[26] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, (Madison, WI, USA), p. 111–118, Omnipress, 2010.

[27] J. Weng, N. Ahuja, and T. Huang, "Cresceptron: a self-organizing neural network which grows adaptively," in *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, vol. 1, pp. 576–581 vol.1, 1992.

[28] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[29] R. Zhang, "Making convolutional networks shift-invariant again," in *ICML*, 2019.

[30] T. Williams and R. Li, "Wavelet pooling for convolutional neural networks," in *International Conference on Learning Representations*, 2018.

[31] G. Strang and T. Q. Nguyen, "Wavelets and filter banks," 1996.

[32] O. Matan, C. J. C. Burges, Y. LeCun, and J. S. Denker, "Multi-digit recognition using a space displacement neural network," in *Neural Information Processing Systems*, 1991.

[33] I. Daubechies, *Ten Lectures on Wavelets*. CBMS-NSF Regional Conference Series in Applied Mathematics, Society for Industrial and Applied Mathematics, 1992.

[34] D. Gudovskiy, S. Ishizaka, and K. Kozuka, "CFLOW-AD: Real-time unsupervised anomaly detection with localization via conditional normalizing flows," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 98–107, January 2022.

AN D. LE received the B.Eng. degree in Electrical Engineering and Information Technology from Frankfurt University of Applied Science, Frankfurt am Main, Germany, and Vietnamese German University, Vietnam, in 2017 and the M.S. degree in Mechatronics and Sensor Systems Technology from Karlsruhe University of Applied Science, Karlsruhe, Germany, and Vietnamese German University, Vietnam in 2019. He was a recipient of the DAAD Sur-place scholarship in Vietnam, 2017, of the DAAD scholarship in Karlsruhe, Germany, in 2018, and of Vingroup Graduate Scholarship, in 2020.

He is currently pursuing the Ph.D. degree with the Electrical and Computer Engineering Department, University of California in San Diego. His research interests include image processing, signal processing, and machine learning.

SHIWEI JIN received the B.S. in optical science and engineering from Zhejiang University, Hangzhou, China, in 2018, and the M.S. in electrical and computer engineering from University of California, San Diego, CA, USA, in 2020.

Currently he is working towards the Ph.D. in the Electrical and Computer Engineering Department, University of California, San Diego. He is supervised by Prof. Truong Q. Nguyen at the Video Processing Lab. His research interests include 2D/3D feature matching and gaze estimation.

YOU-SUK BAE received his BA and the MS degrees from Seoul National University both in Electrical Engineering and Ph.D. degrees from Korea Advanced Institute of Science & Technology (KAIST) in Electrical Engineering

Dr. You-Suk Bae is a Research Fellow at Korea Electronics Technology Institute (KETI) and Professor of Department of Computer Engineering at Tech University of Korea (TUK). Before joining the KPU, he was a research at the US National Institute of Standards and Technology (NIST) and a principal engineer at Samsung Electronics. His research specialties are pattern recognition, biometircs, and optical information processing. He has been involved in various projects such as 3-D imaging system and storage system development, biometiric recognition system development and improvement, and the next generation of laser and LED display development utilizing MEMS.

TRUONG Q. NGUYEN (Fellow, IEEE) is currently a Professor with the ECE Department, UC San Diego. His current research interests include 3D video processing and communications and their efficient implementation. He is the coauthor (with Prof. Gilbert Strang) of a popular textbook, Wavelets and Filter Banks (WellesleyCambridge Press, 1997), and the author of several matlab-based toolboxes on image compression, electrocardiogram compression, and filter bank design. He has over 400 publications. He received the IEEE TRANSACTION ON SIGNAL PROCESSING Paper Award (Image and Multidimensional Processing area) for the article he co-wrote with Prof. P. P. Vaidyanathan on linear-phase perfect-reconstruction filter banks (1992). He received the NSF Career Award in 1995 and is currently the Series Editor (Digital Signal Processing) for Academic Press. He served as the Associate Editor for the IEEE TRANSACTION ON SIGNAL PROCESSING, from 1994 to 1996, the SIGNAL PROCESSING LETTERS, from 2001 to 2003, the IEEE TRANSACTION ON CIRCUITS AND SYSTEMS, from 1996 to 1997, 2001 to 2004, and the IEEE TRANSACTION ON IMAGE PROCESSING, from 2004 to 2005.