LDW-Pooling: Learnable Discrete Wavelet Pooling for Convolutional Networks

Bor-Shiun Wang[1]
eddiewang.ai08@nycu.edu.tw
Jun-Wei Hsieh[1]
jwhsieh@nctu.edu.tw
Ping-Yang Chen[1]
pingyang.cs08g@nctu.edu.tw
Ming-Ching Chang[3]
mchang2@albany.edu
Lipeng Ke[4]
lipengke@buffalo.edu
Siwei Lyu[4]
siweilyu@buffalo.edu

[1] National Yang Ming Chiao Tung University, Taiwan
[2] University at Albany, State University of New York, USA
[3] University at Buffalo, State University of New York, USA

Abstract

Pooling is a simple but widely used layer in modern deep CNN architectures for feature aggregation and extraction. Typical CNN design focuses on the conv layers and activation functions, while leaving the pooling layers with fewer options. We introduce the Learning Discrete Wavelet Pooling (LDW-Pooling) that can be applied universally to replace standard pooling operations to better extract features with improved accuracy and efficiency. Motivated from the wavelet theory, we adopt the low-pass (L) and high-pass (H) filters horizontally and vertically for pooling on a 2D feature map. Feature signals are decomposed into four (LL, LH, HL, HH) subbands to retain features better and avoid information dropping. The wavelet transform ensures features after pooling can be fully preserved and recovered. We next adopt an energy-based attention learning to fine-select crucial and representative features. LDW-Pooling is effective and efficient when compared with other state-of-the-art pooling techniques such as WaveletPooling and LiftPooling. Extensive experimental validation shows that LDW-Pooling can be applied to a wide range of standard CNN architectures and consistently outperform standard (max, mean, mixed, and stochastic) pooling operations.

1 Introduction

Convolutional Neural Networks (CNNs) have been widely used in many domains, including object detection, segmentation, and recognition. In the design of CNN architecture, downsampling layers with pooling and stride-convolutions are common operations that can effectively aggregate features. Standard pooling methods such as simple average pooling, max/min pooling are widely used for multiple purposes in CNN, to (1) fine-select important features, (2) aggregate spatially local features, and (3) retain important features for efficient processing in subsequent layers. Although being widely used, there are several drawbacks of

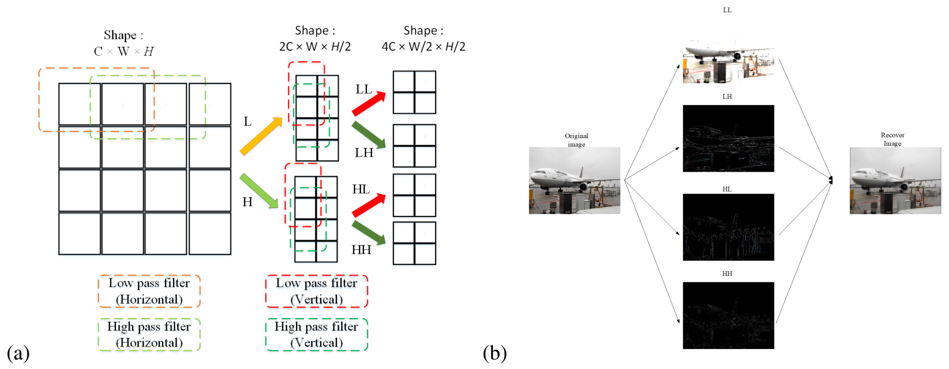
Shape :
C × W × H
Shape :
2C × W × H/2
Shape :
4C × W/2 × H/2
L
H
LL
LH
HL
HH
Low pass filter (Horizontal)
High pass filter (Horizontal)
Low pass filter (Vertical)
High pass filter (Vertical)
(a)
LL
LH
HL
HH
Original image
Recover Image
(b)

Figure 1: **Overview of the proposed LDW-Pooling method.** (a) Given an input signal of C channels of width W $\times$ height H, we use high-pass (H) and low-pass (L) filters in the size of $1 \times K$ (horizontally) and $K \times 1$ (vertically) to decompose each 2D feature map into four subbands, LL, HL, LH, HH. (b) Reconstructed result after LDW-Pooling decomposition and recovering by UNet with PSNR=33.1.

the simple pooling methods. First, over-aggressive filtering inevitably results in information loss and degraded performance. Second, the simple reduction of spatial resolution breaks *shift-invariance* [22]. Third, pooling operations are generally not *invertible* — upsampling the down-sampled feature maps cannot recover the lost information. Finally, feature reduction can be pushed to the extreme, that the feature vector of a small object can reduce into a single-pixel or disappear in the deeper layers, thus hinders accurate discrimination. To this end, many recent designs of more sophisticated but effective pooling methods are proposed [18, 24]. However, due to the broad applicability, the potential of a better pooling mechanism is yet to be thoroughly explored. This paper investigates a bidirectional pooling mechanism that can preserve details when down-sampling the feature maps and retain finer spatial information to recover detailed up-sampled feature maps.

Max pooling is the most widely used downsampling operation in popular CNN architectures, where the strongest activation within each neighborhood is kept in the filtering scope. However, such strongest activations may not always be the best choice for pooling. In [20], a stochastic pooling technique is proposed to ensure that non-maximal activations can also be utilized. To address the shift-invariance problem, an anti-aliasing filter with a smoothing kernel is adopted in [22] to improve ImageNet classification. In general, excessive pooling can degrade object detection or recognition accuracy due to the inability to preserve discriminative features or details. In [4], the Local Importance-based Pooling (LIP) outperformed hand-crafted pooling layers by a large margin, with a penalty of lower efficiency. In [3], an adaptive pooling technique can adjust the level of details to preserve. However, this adaptive technique needs a parallel implementation for efficiency consideration.

In this work, we propose a *learnable* discrete wavelet (LDW) pooling technique to better aggregate local features, such that lost information can be better recovered. Moreover, in LDW-Pooling, we decompose 2-D convolutions into the $1 \times K$ and $K \times 1$ kernels to efficiently extract horizontal and vertical features with a learnable attention mechanism. Figure 1 shows the overview of the proposed LDW-Pooling method.

The main contributions of this paper are summarized in the following:

- We propose a new LDW-Pooling scheme by decomposing the input signal into four discrete wavelet subbands (LL, LH, HL, HH) without information loss. With a new learnable energy attention design, LDW-Pooling enables accurate and fast pooling that can be combined and used in mainstream CNN applications.

- The *reversibility* of LDW-Pooling is well-suited for fine-grained object classification for objects in both small and large sizes with higher accuracy.
- The LDW-Pooling performs 2-D convolution by decomposing it into two 2 runs of 1-D convolutions. It is thus more efficient than traditional pooling methods. It can be easily integrated with different backbone thanks to this efficient and reversible design.
- LDW-Pooling outperform the state-of-the-art classification method on CIFAR-10, CIFAR-100, and ImageNet datasets in terms of accuracy and efficiency. Results also show great generalization ability on various backbones.

2 Related Works

Pooling in CNN. In CNN architectures, pooling is a widely used technique to improve the feature extraction efficiency and robustness to input variations. Two most popular pooling methods are average pooling and max pooling. The former is equivalent to blurred-downsampling and the latter is generally more effective and popularly used in many CNN backbones. Although being simple and effective, these standard pooling methods lose details in downsampling [19, 20]. To maintain structured details, the detail-preserving pooling (DPP) in [14] can magnify spatial changes and thus preserves some structural details.

Learning based pooling. To make the pooling learnable, [11] learns adapted pooling on complex and variable patterns via a linear combination of max and average pooling. A stochastic pooling was proposed in [20] for solving the over-fitting problem in CNN architectures by using a multinomial distribution to aggregate activations within a region. Similarly, in [21], S3 pooling was proposed to embed randomness into a multinomial probabilistic model, to select local activations with proper weights. A Gaussian model is modified in [17] for probabilistic pooling with trainable parameters, where local activations can be aggregated based on their local statistics. Toward trainable pooling operation, a global feature guided flexible pooling in [8] selects the pooling function based on the maximum entropy principle.

Wavelet related pooling methods. It is known that standard (max and average) poolings can drop details in downsampling, especially for important details that are with less intensity than the insignificant ones [19]. Several pooling approaches tried to preserve details by incorporate wavelet representations into CNNs. A wavelet pooling algorithm in [18] decomposes input signals to different subbands using wavelet transform, such that the feature contents can more accurately represented with fewer artifacts. The first layer of ResNet is modified in [12] with a wavelet scattering network to maintain performances using a smaller number of pooling parameters. In [2], Haar wavelet CNNs are combined with a multi-resolution analysis for texture classification and image annotation.

Feature-preserving and invertible pooling. The Lifting Scheme within the CNN networks in [3] improves texture and object classification with less parameters. Their framework focuses on building an interpretable network by integrating multiresolution analysis, rather than pooling. The BlurPool with a smoothing kernel in [22] can reduce aliasing and improve ImageNet classification accuracy, however the pooling still results in detail lost. In the above wavelet-based pooling techniques or multi-resolution analysis, Haar wavelets are used for band decomposition. Thus in these methods, the wavelet filter are handcrafted, with fixed values, and thus not adaptive to handle data-dependent tasks. These pooling methods not invertible (that is, the lost information in the downsampling process cannot be recovered). Inspired by the Lifting Scheme [16], LiftPooling [24] allows the CNN architecture to be invertible and thus can achieve better robustness against input corruptions and perturbations. The LiftDownPool module was created to decompose the input into four sub-bands with the Haar wavelet. Next, another LiftUpPool module performs upsampling from these four sub-bands without losing information. The parameter space of LiftPooling is high, since all subbands must be used to achieve the best performance. Similar to other wavelet-based pooling methods, the Haar wavelet filters used for generating the four bands (LL, LH, HL, and HH) are handcrafted and not learnable from data.

Compared to all surveyed methods, in this paper, we propose a learnable pooling scheme based on discrete wavelets that is *invertible, adaptive, and efficient*. The proposed LDW-Pooling learns the wavelet filter parameters from data for subband decomposition and recovery. Since the filters are learned from image contents, it is adaptive and potentially perform better for image classification and recognition. With a new energy attention learning, our method enables accurate and fast pooling that is suitable for CNN classification of objects of various sizes. It is computational efficient due to the decomposition of the 2-D convolution into two 1-D convolution operations.

3 Methods

3.1 Discrete Wavelet Pooling

We employ the learnable discrete wavelet pooling to extract features and better retain them with less computation. Let $X \in R^{H\times W}$ denote an input feature map. The local pooling operation on X can be defined as:

$$Y_{(x',y')} = \sum_{(\Delta x_i,\Delta y_i)\in\Omega} w_i\, X_{(x+\Delta x_i,y+\Delta y_i)} \qquad s.t. \quad \sum_i (w_i)^2 = 1,\ w_i \geq 0, \tag{1}$$

where Ω indicates a kernel region, and w_i indicates the weighting parameter at each position. The division $(\frac{x}{x'},\frac{y}{y'})$ stands for the stride factor, *i.e.*, $x = 2x'$ and $y = 2y'$ for 2×2 stride. For the case of average pooling, Y represents the low-frequency part of X; and the high-frequency part of X is discarded (in other words, the details of X are missing after pooling). Max pooling keeps the maximal activation and thus performs generally better than average pooling on feature selection. However all non-maximum signals are discarded after max pooling. The widely used down-sampling scheme of *convolution with stride 2* in CNN also results in information loss. An intuitive motivation of our work is to develop a pooling operation that can extract and retain both low and high frequency parts of X. Inspired by the wavelet theory and the strength of invertability of wavelet transform, our wavelet based pooling is invertible in nature. With the new design of learning wavelet parameters from data, our LDW-Pooling can excel in feature-preserving and computational efficiency.

Let W^L and W^H denote the low-pass and high-pass wavelet kernel weights, respectively. According to the property of wavelet transform, W^H and W^L should satisfy the constraints:

$$\sum_{i\in\mathbf{K}} \left(W^L(i)\right)^2 = 1 \quad \text{and} \quad \sum_{i\in\mathbf{K}} W^H(i) = 0, \tag{2}$$

where $\mathbf{K}$ represents a 1-D kernel of size $K = |\mathbf{K}|$. As shown in Figure 1, the two horizontal poolings are first performed using W^L and W^H respectively, and then two vertical pooling are performed subsequently. The horizontal pooling decomposes the incoming map along the X-direction to low-frequency signal $Y^L_{(x',y)}$ and high-frequency signal $Y^H_{(x',y)}$:

$$Y^{L}_{(x',y)} = \sum_{i \in \mathbf{K}} W^{L}(i)\, X_{(x+i,y)}, \qquad (3) \qquad\qquad Y^{H}_{(x',y)} = \sum_{i \in \mathbf{K}} W^{H}(i)\, X_{(x+i,y)}. \qquad (4)$$

Assume there are C channels of input feature maps. After the horizontal pooling, there are $2C$ channels of resulting feature maps with half sizes along the X direction.

Next, two vertical pooling operations continue on $Y^{L}_{(x',y)}$ and $Y^{H}_{(x',y)}$, respectively. Regarding $Y^{L}_{(x',y)}$, we obtain two subbands:

$$Y^{LL}_{(x',y')} = \sum_{j \in \mathbf{K}} W^{L}(j)\, Y^{L}_{(x',y+j)}, \qquad (5) \qquad\qquad Y^{LH}_{(x',y')} = \sum_{j \in \mathbf{K}} W^{H}(j)\, Y^{L}_{(x',y+j)}. \qquad (6)$$

In addition, for $Y^{H}_{(x',y)}$, we obtain the other two subbands $Y^{HL}_{(x',y')}$ and $Y^{HH}_{(x',y')}$ are obtained:

$$Y^{HL}_{(x',y')} = \sum_{j \in \mathbf{K}} W^{L}(j)\, Y^{H}_{(x',y+j)}. \qquad (7) \qquad\qquad Y^{HH}_{(x',y')} = \sum_{j \in \mathbf{K}} W^{H}(j)\, Y^{H}_{(x',y+j)}. \qquad (8)$$

Note that only 1-D convolutions are used to obtain the pooling results $Y^{LL}_{(x',y')}$, $Y^{LH}_{(x',y')}$, $Y^{HL}_{(x',y')}$, and $Y^{HH}_{(x',y')}$ as well. The time complexity for performing Eqs.(3)-(8) is $\mathcal{O}(KWH)$. In comparison, standard pooling techniques such max pooling or average pooling on conv stride 2 are all based on a 2-D kernel. Their complexity is $\mathcal{O}(K^2WH)$, and the output signal is not invertible.

For the best of our knowledge, all other existing wavelet-based pooling techniques [2, 3, 12, 18, 24] perform signal decomposition based on a fixed Haar wavelets design: $W^L = \{\frac{1}{2}, \frac{1}{2}\}$ and $W^H = \{\frac{1}{2}, \frac{-1}{2}\}$. Such handcrafted wavelet parameters are not learnable nor adaptive to application tasks. Another drawback of these methods are that, after each pooling operation, the number of feature channels increases to become quadruple. Their parameter spaces are higher, since all subbands shall be used to achieve the desired CNN performance. In comparison to these method, our proposed LWD-Pooling adopts a learnable scheme to learn filter parameters from data for subband decomposition. Such learned 1-D wavelet parameters can better recover pooled signals with improved efficacy and efficiency, and the total parameters are greatly reduced.

3.2 Learning Discrete Wavelet Pooling Filter Parameters

We next describe the design of the loss function for data-driven learning of discrete wavelet (DW) filter parameters. To prevent the filter weights from approaching mostly zero during the training process, we enforce the following constraints in the loss design.

For the low-pass DW filter, the square sum of filter weights should be equal to 1 according to Eq. (2). Since the filters are to be learned from data, the values of the low-pass DW filter can be distinct to gain better performance. And we empirically confirmed such proposition. Let W^L denotes the low-pass DW filter weights. According to this constraint from Eq. (2) during training, we introduce the following loss term for the low-pass DW filters:

$$L_{Low} = \left[\sum_{i}^{K} W^{L}(i)^2 - 1\right]^2 + \left[\sum_{i}^{K} \left(W^{L}(i)\right) - \sqrt{2}\right]^2. \qquad (9)$$

This constraint minimizes the squared differences between the sum of low-pass wavelet weights and $\sqrt{2}$. For the high-pass DW filters, similarly, direct summation of high-pass

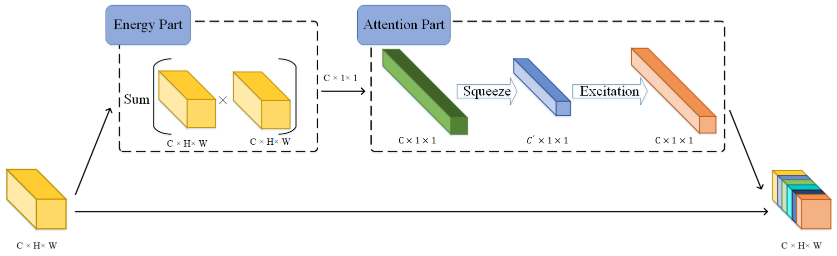

Figure 2: The **energy attention mechanism** for feature enhancement consists of two parts. The "Energy" part calculates feature energy in each channel as the sum of squared feature values. The "Attention" part learns which channel is important via squeeze-and-excitation [6].

filter weights in training may result in all zeros (as a trivial solution that is not desirable). Since the sum of high-pass filter weights must be zero, we introduce the following constraint L_{High} in the loss term for the high-pass DW filters:

$$L_{High} = \left(\sum_{i}^{K} W^H(i)^2 - 1 \right)^2 + \left(\sum_{i}^{K} W^H(i) \right)^2 , \tag{10}$$

where W^H denote the high-pass filter weight. The first term of Eq. (10) enforce W^H to become unit vectors.

To retain the *reversibility* of DW pooling, the energy of W^L and W^H should be equal to 1. Thus, we introduce the following reversibility loss term:

$$L_{Reverse} = \left[\left(\sum_{i}^{K} W^L(i)^2 + \sum_{i}^{K} W^H(i)^2 \right) - 2 \right]^2 . \tag{11}$$

Finally, to make the symmetrical filters, we introduce the following symmetrical loss term:

$$L_{Sym} = \sum_{i}^{\lfloor K/2 \rfloor} \left[W^L(i) - W^L(K-i) \right]^2 + \sum_{i}^{\lfloor K/2 \rfloor} \left[W^H(i) - W^H(K-i) \right]^2 . \tag{12}$$

The four constraints in Eqs. (9)-(12) jointly ensure: (1) the sum of low-pass filter weights to be $\sqrt{2}$, (2) the sum of high-pass filter weights to be 0, (3) the reversibility of pooling, and (4) the symmetric of the filter. We sum up all three terms in combination:

$$L_{Wavelet} = (L_{Low} + L_{High} + L_{Reverse} + L_{Sym}). \tag{13}$$

For training the CNN classification model, we use cross-entropy loss which is commonly used in the classification task,

$$L_{CE} = \sum_{b}^{B} [\hat{y_b} \log(y_b) + (1 - \hat{y_b}) \log(1 - y_b)], \tag{14}$$

where B denote the batch size, and $\hat{y_b}$ and y_b denote the ground-truth label and model predicted probability, respectively. The final total loss used for training is the sum of the cross-entropy loss $L_{Wavelet}$ and the constraint terms $L_{Wavelet}$:

$$L_{total} = L_{CE} + L_{Wavelet}. \tag{15}$$

3.3 Energy-based Attention Learning

To finish feature aggregation from the discrete wavelet pooling in § 3.1 so that features can be better preserved, we perform an energy based attention learning that can enhance the

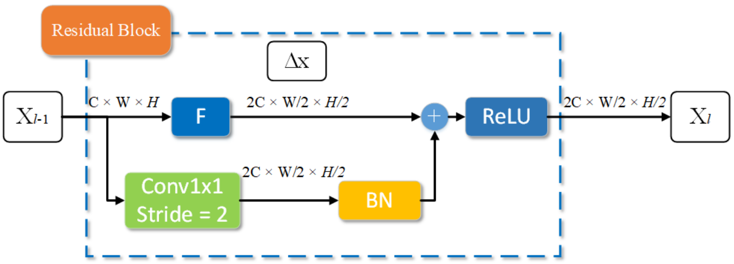

Figure 3: **Original Bottleneck.** *BN* and *ReLU* denote to batch normalize and ReLU function. With several convolutions to extract the feature to get Δx. The residual branch simply get features from X_{t-1} with 1×1 convolution with 2-stride if needed.

extracted feature with crucial information. Figure 2 illustrates such energy-based attention learning mechanism. In contrast to the use of average in computing the feature map, the energy design aims to keep high activations. In standard average pooling, such high activations will be smoothed by the average operation when it surrounds by low activation, which is a drawback that leads to the missing of crucial information after pooling.

In our energy-based design, we introduce a square operation in energy calculation, such that low activations that are possibly with negative values can become positive after taking square. This way, the energy-based learning can possibly attend to low input activations during the training. This design can essentially resolve the difficulty of learning important low activation signals, while preventing the high activations to be smoothed by low activations.

Energy definition. Given an input feature map X of size $W \times H$, let X^c denote the feature map of the c^{th} feature channel. The energy E^c of the feature map X^c for channel c is calculated as the sum of the square of all feature values in the channel:

$$E^c = \sum_x^W \sum_y^H \left(X^c_{(x,y)} \right)^2 . \tag{16}$$

Note that the output Y of feature maps (Eqs.(3)-(8)) at current layer will be the input X of next layer. The energy mechanism reinforces all the activations no mater it responses high or low to the filter. To eliminate the impact of image brightness (whose value is usually greater than other features), we perform batch normalization before calculating the energy.

Attention calculation. In contrast to the simple use of global average to summarize the feature information, we use the energy calculated from Eq. (16) to represent the channel information. Our approach is inspired from the Squeeze-and-Excitation Net (SENet) [6]. Let SE denote the Squeeze-and-Excitation without average pooling. The Squeeze-and-Excitation with energy S_e is calculated as:

$$S_e = SE\left(E^1, ..., E^c, ..., E^C\right), \tag{17}$$

where C is the total number of channels. The feature channels $\{X^c, c = 1, ...C\}$ can then be weighted based on S_e.

3.4 LDW-Pooling for Residual Net

We describe how to apply LDW-Pooling to the widely used Residual Net [5]. In Residual Net, the down-sample operators including max pooling and 2-stride convolution, as in Figure 3. The time complexity to perform the residual block is $\mathcal{O}(\frac{1}{4}CK^2WH)$, where the kernel size for convolution is $K \times K$.

To apply LDW-pooling to the Residual Net, we first use a 1×1 convolution with stride

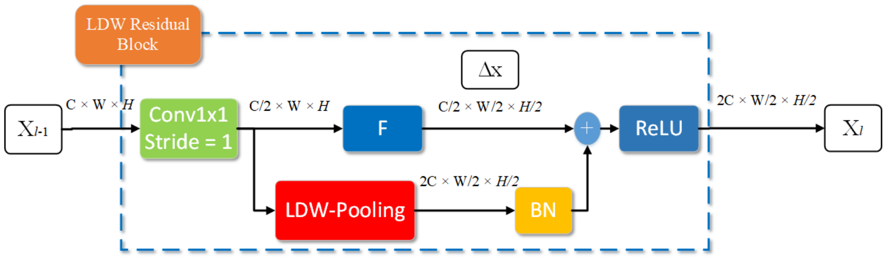

Figure 4: **LDW-Pooling in Bottleneck.** With LDW-Pooling on X_{t-1} at residual branch. We slightly change the output of $Conv1 \times 1$ with the channel number equal to the input channel. This makes the output channel of LDW-Pooling correspond to the original output.

2 to reduce the number of channels from C to $\frac{C}{4}$. Then, the time complexity to perform the residual block is thus $\mathcal{O}(\frac{1}{16}CK^2WH)$. Figure 4 shows the details of how our LDW-pooling is applied to the bottleneck of Residual Net. The LDW-pooling is added to the skip connection. The number of feature channels fed to the LDW-pooling is $C/4$ and becomes C after pooling. After integration, the output dimension of this bottleneck is $C \times \frac{W}{2} \times \frac{H}{2}$. The LDW-Pooling performs 2-D convolution by decomposing it into two 2 runs of 1-D convolutions and results in the time complexity $\mathcal{O}(\frac{1}{2}CKWH)$. It is thus more efficient than the original design of Residual Net. LDW-Pooling can be easily integrated with different backbones in a similar way to leverage such efficient design.

Table 1: Ablation study of our LDW-pool on CIFAR-10 [10] and CIFAR-100 [9]

CIFAR-10			CIFAR-100		
Model	Pooling	Mean Test Error	Model	Pooling	Mean Test Error
ResNet50	Original	8.72%	ResNet50	Original	31.89%
	Ours	**7.87%**		Ours	**29.75%**
VGG13	Original	9.63%	VGG13	Original	32.06%
	Ours	**8.98%**		Ours	**30.50%**
VGG16	Original	8.48%	VGG16	Original	30.73%
	Ours	**7.94%**		Ours	**29.87%**

4 Experimental Results

Implementation details. We trained several CNN models (ResNet and VGG with various depth) with LDW-Pooling with batch size 64 for experimental evaluation. The learning rate starts from 10^{-4} with a decrease by the rate of 0.1 for every 100 epochs. Each model is trained for 400 epoch using the Adam optimizer with weight decay 10^{-4}.

4.1 Results

Results on CIFAR-10. The dataset consists of 50,000 training data and 10,000 testing data in 10 classes of low resolution (28×28), so we perform data augmentation on training set by re-scaling the images into 300×300 then random cropping into 224×224, and random flipping horizontally or vertically. For testing, we simply re-size the images into 224×224. As shown in Table 1 (left) our proposed pooling surpass three popular backbones.

Results on CIFAR-100. The CIFAR-100 dataset is similar to CIFAR-10, except that the number of the class is 100 classes and 600 images per class. There are 500 and 100 images per class for the training and testing set, respectively. We perform the same pre-processing as in CIFAR-10. As shown in Table 1 (right), that LDW-Pooling has sigfinicant performance gain over all the baselines due to the property of feature preserving.

Results on PASCAL-VOC12. The PASCAL-VOC12 is a semantic segmentation dataset that contains 20 foreground object classes and one background class. The augmented version of PASCAL-VOC12 which contains 10582 training images and 1449 validation images is used for training and validation. We consider the basic UNet model which downsamples with maximum pooling and upsamples with de-convolution as the benchmark. We change the maximum pooling and de-convolution with LDW-Pooling and reverse LDW-Pooling for the comparison. The performance is measured in terms of pixel mean-intersection-over-union(mIoU) across 21 classes as shown in Table 4. The reversibility of our method makes better segmentation results gained than the original Unet. Figure 1(b) shows the reconstructed result after LDW-Pooling decomposition and recovering by UNet with PSNR=33.1.

Results on ImageNet. ImageNet consists of 1,000 categories ranging from 732 to 1300 images per class in the training set and 50 images per class in the validation set. We performed the same pre-processing as done in Section 4.1 for CIFAR-10. Table 2 shows the comparison of LDW-Pooling with other state-of-the-art pooling methods. For all the categories except ResNet 18, our method outperforms other methods. For ResNet18, fewer convolution layers might cause few semantic features to be extracted for LDW-pooling.

4.2 Ablation Study

To show the effectiveness of our method, we conduct ablation study on several settings.

LDW-Pooling and Energy Attention on classification is reported with CIFAR-10/100 in Table 3. It is clear that the accuracy improvement is majorly caused by the LDW-Pooling technique rather than energy attention.

LDW-Pooling vs i-RevNet [7]: Table 5 shows that comparing with SoTA reversible pooling method, *i.e.*, i-RevNet [7]. Our LDW-pooling method can gain better classification accuracy on both CIFAR-10 and CIFAR-100 dataset. Since i-RevNet [7] focuses only on the design of filters with good reversibility rather than good feature extraction.

Pretrained Filter Weights: The pretrained weights were obtained by randomly generating filters which satisfy wavelet constraints before training. Clearly, better pooling filters for target tasks can be derived via the loss $L_{wavelet}$. More weights on $L_{Wavelet}$ will enhance the reversibility with higher PSNR if the target task is "reconstruction" or "segmentation". For the classification task, L_{CE} is a necessary term. Table 6 shows the accuracy of classification can be further improved if the term $L_{Wavelet}$ is added. If the loss $L_{Wavelet}$ is added more, more classification accuracy can be gained. However, when the reversibility is higher, the effect of L_{CE} will decrease and result in accuracy degradation. Thus, the weights for L_{CE} and $L_{Wavelet}$ are equally set (see Eq.(15)). The "pretrain" option means initial weights are obtained by randomly generating filters which satisfy wavelet constraints before training.

5 Conclusion

This paper presented the LDW-Pooling, an effective learnable pooling method that can swap and replace the widely-used pooling methods in CNN with many desirable advantages. The discrete wavelet transform properties ensure the invertibility of the extracted features after pooling. The weights of the wavelet kernel can be learned from training data via an energy-based attention mechanism to better perform in each application-dependent scenario. The 1-D linear complexity is computationally efficient when compared with other

Table 2: Comparisons among LDW-Pooling and SoTA pooling methods based on ImageNet [1]. Symbol ⋆ denotes the re-evaluated scores by running source codes from original authors. Other scores were cited from [24].

	ResNet18		Renst50		MobileNet-V2	
	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
Skip [15]	30.22	10.23	24.31	7.34	28.66	9.70
MaxPool	28.60	9.77	24.26	7.22	28.65	9.82
AveragePool	28.03	9.55	24.40	7.35	28.32	9.72
S3Pool [21]	33.91	13.09	27.98	9.34	40.56	17.91
WaveletPool [18]	30.33	10.82	24.43	7.36	29.27	10.26
BlurPool [22]⋆	29.88	10.58	24.60	7.73	30.58	11.26
DPP [14]⋆	29.12	10.21	24.62	7.49	29.85	10.53
SpectralPool [13]	28.69	9.87	24.81	7.57	33.38	12.56
GatedPool [11]	27.78	9.44	23.79	7.06	28.94	9.90
MixedPool [11]	27.76	9.50	24.08	7.32	29.00	9.97
GGFGP [8]⋆	26.88	8.66	22.76	6.34	28.42	9.59
GaussPool [17]⋆	26.58	8.86	22.95	6.30	27.13	8.92
LiftDownPool [23]	**25.80**	8.14	22.36	6.11	26.09	8.22
LDW-Pool	26.10	**8.01**	**22.10**	**5.88**	**25.97**	**7.98**

Table 3: Ablation study of LDW-Pooling and Energy Attention.

Model	LDW-Pooling	Energy Attention	CIFAR-10 Mean Test Error	CIFAR-100 Mean Test Error
ResNet50	V	V	7.87%	29.75%
	V		7.98%	29.83%
			8.72%	31.89%

Table 4: Ablation study of reversibility with UNet on VOC12 dataset.

VOC12 (21 classes)		
Model	Pooling	mIoU
Unet	Original	47.83
	Ours	49.02

Table 5: Comparisons between our method and i-RevNet[7].

Model	Pooling	CIFAR-10 Mean Test Error	CIFAR-100 Mean Test Error
ResNet50	Our	7.87%	29.75%
	PS	8.77%	32.02%

Table 6: Ablation study of the effect of $L_{Wavelet}$.

Pretrain	L_wavelet	Mean Test Error
V	V	7.87%
V		7.98%
	V	9.13%
		9.94%

wavelet-based pooling methods. Experimental evaluations show that LDW-Pooling outperforms other wavelet-based pooling approaches as well as standard pooling methods.

Future work includes further application and evaluation of the LDW-Pooling on other computer vision tasks such as object detection with segmentation, as well as application to non-vision tasks including NLP and others.

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

[2] Shin Fujieda, Kohei Takayama, and Toshiya Hachisuka. Wavelet convolutional neural networks. *CoRR*, abs/1805.08620, 2018. URL http://arxiv.org/abs/1805.08620.

[3] Shin Fujieda, Flavio Prieto Ortiz, Kohei Takayama, and Kohei Takayama. Deep adaptive wavelet network. In *WACV*, 2020.

[4] Ziteng Gao, Limin Wang, and Gangshan Wu. LIP: Local importance-based pooling. In *ICCV*, 2019.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, June 2016.

[6] Hu, Jie, Shen, Li, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.

[7] Jorn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. i-revnet: Deep invertible networks. In *ICLR*, 2018.

[8] Takumi Kobayashi. Global feature guided local pooling. In *ICCV*, 2019.

[9] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research). 2009. URL http://www.cs.toronto.edu/~kriz/cifar.htm.

[10] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. CIFAR-10 dataset, 2009. http://www.cs.toronto.edu/~kriz/cifar.html.

[11] Chen-Yu Lee, Patrick W Gallagher, and Zhuowen Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *AISTATS*, 2016.

[12] Edouard Oyallon, Eugene Belilovsky, and Sergey Zagoruyko. Scaling the scattering transform: Deep hybrid networks. In *ICCV*, 2017.

[13] Oren Rippel, Jasper Snoek, and Ryan P Adams. Spectral representations for convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, 2015.

[14] Faraz Saeedan, Nicolas Weber, Michael Goesele, and Stefan Roth. Detail-preserving pooling in deep networks. In *CVPR*, 2018.

[15] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity:the all convolutional net. In *ICLR*, 2015.

[16] Wim Sweldens. The lifting scheme: A construction of second generation wavelets. In *SIAM Journal on Mathematical Analysis*, 1998.

[17] Kobayashi Takumi. Gaussian-based pooling for convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[18] Travis Williams and Robert Li. Wavelet pooling for convolutional neural networks. In *ICLR*, 2018.

[19] Dingjun Yu, Hanli Wang, Peiqiu Chen, and Zhihua Wei. Mixed pooling for convolutional neural networks. In *Rough Sets and Knowledge Technology*, 2014.

[20] Matthew D. Zeiler and Rob Fergus. Stochastic pooling for regularization of deep convolutional neural networks. In *ICLR*, 2013.

[21] Shuangfei Zhai, Hui Wu, Abhishek Kumar, Yu Cheng, Yongxi Lu, Zhongfei (Mark) Zhang, and Rogerio Feris. S3Pool: Pooling with stochastic spatial sampling. In *CVPR*, 2017.

[22] Richard Zhang. Making convolutional networks shift-invariant again. In *ICML*, 2019.

[23] Jiaojiao Zhao and Cees G. M. Snoek. Liftpool: Bidirectional convnet pooling. In *ICLR*, 2021. URL https://openreview.net/forum?id=kE3vd639uRW.

[24] Jiaojiao Zhao and Cees G.M. Snoek. Liftpool : Bidirectional convnet pooling. In *ICLR*, 2021.