UDC 004.89

# ВЕЙВЛЕТНЫЕ НЕЙРОННЫЕ СЕТИ

**И. Боровский**

*Белорусский государственный университет, Беларусь, Минск,*
*iljabarouski@gmail.com*

Вейвлеты нашли применение в обработке сигналов и анализе изображений. Они позволяют извлекать важную информацию из сигнала, формируя его аппроксимацию и детализационные компоненты, каждая из которых вдвое короче исходного сигнала. Затем возможно восстановление исходного сигнала.
Тем не менее, вейвлет-преобразование редко используется в нейронных сетях из-за сложности реализации. Мы предлагаем обучаемые модули на PyTorch для вейвлет-преобразований и функций потерь, которые могут применяться для построения специализированных вейвлетов.

***Ключевые слова:*** вейвлеты; дискретное вейвлет-преобразование; обратное дискретное вейвлет-преобразование; DWT; IDWT; нейронные сети.

# WAVELET NEURAL NETWORKS

**I. Baroŭski**

*Belarusian State University, Belarus, Minsk, iljabarouski@gmail.com*

Wavelets have found usage in signal processing and image analysis. They can extract important information from the signal, producing its approximation and details components both half the length of the original signal. Then the original signal can be reconstructed. Nevertheless, wavelet transform is not widely used in neural networks because of complications in implementation. We propose trainable PyTorch modules for wavelet transforms and loss functions which can be used for construction of specific wavelets.

***Keywords***: wavelets; discrete wavelet transform; inverse discrete wavelet transform; DWT; IDWT; neural networks.

## Introduction

Wavelet transform is a signal analysis technique. While Fourier transform analyses a signal in the frequency domain, Gabor transform implies Fourier analysis to sliding windows of a signal giving analysis in both time and frequency domains, wavelet transform analyses a signal in time and frequency domains without limitations of windowing. It projects the initial signal into two subspaces formed by bases functions: the first one contains the approximation

of the signal and the second contains the details. Then the analysis can be performed on approximation, dividing it into two components once again. The process is called multiresolution analysis. [1]

In practice, wavelet transform is performed via convolution with two tied kernels called scaling and wavelet coefficients (low-pass and high-pass filters), where wavelet coefficients are computed from scaling ones. Stride in convolution operation is equal to two which gives us the ability to compress the signal by the factor of two while preserving signal features. Moreover, unlike regular convolution and pooling layers, wavelet transform with kernel size $4k+2$ is invertible in our implementation.

## Method

We propose the trainable wavelet transform blocks based on convolution scheme and differentiable padding block for proper size reduction. The kernels can be trained via special weight regularization loss functions, based on wavelet constraints, or be provided by the user either with pre-computed kernels or with wavelet name from the PyWavelets library.

## Loss Functions

First, we will describe orthonormal wavelets. As said above, used convolution kernels are tied: the wavelet kernel $\{g_i\}_{i=0}^{N-1}$ is obtained from the scaling one $\{h_i\}_{i=0}^{N-1}$: $g_i = (-1)^i h_{N-1-i}$. Wavelets kernels should obey the following conditions: admissibility (1), orthogonality (2) and regularity (3). [2]

$$\sum_{k=0}^{N-1} h_k = \sqrt{2} \tag{1}$$

$$\sum_{k=0}^{N-1-2n} h_k h_{k+2n} = \delta_n \ \forall n = 0, \ldots, N/2-1 \tag{2}$$

$$\sum_{k=0}^{N-1} (-1)^k k^{p-1} h_k = \sum_{k=0}^{N-1} k^{p-1} g_k = 0 \ \forall p = 1, \ldots, N/2 \tag{3}$$

From these conditions one can construct loss functions: admissibility (4), orthogonality (5) and regularity (6) losses. To be properly minimized, a loss function must have a lower bound. To obtain this behavior, each component is squared. In regularity condition, the constant multiplier in the left-hand side was dropped, but it is used in the loss function to stabilize the optimization process as the function grows rapidly, and the constant naturally reduces it.

$$\mathcal{L}_{WN} = \left( \sum_{k=0}^{N-1} h_k - \sqrt{2} \right)^2 \tag{4}$$

$$\mathcal{L}_{O} = \left( \sum_{k=0}^{N-1} h_k^2 - 1 \right)^2 + \sum_{n=1}^{N/2-1} \left( \sum_{k=0}^{N-1-2n} h_k h_{k+2n} \right)^2 \tag{5}$$

$$\mathcal{L}_{R} = \sum_{p=1}^{P} \left( \frac{\sqrt{2}}{2^P} \sum_{k=0}^{N-1} k^{p-1} g_k \right)^2 \tag{6}$$

Even having the stabilization multiplier, the regularity loss still grows very fast and destabilizes the gradient. The following training technique is proposed. The transform is trained to have only $P$ first vanishing moments, i.e., the regularity loss has $P$ components. If more moments are needed, the component for $P+1$ moment is added to the loss, the optimizer is set to the initial options and the training is proceeded.

In case of biorthogonal wavelets we have scaling and wavelet kernels $\{h_i\}_{i=0}^{N-1}$ and $\{g_i\}_{i=0}^{N-1}$ for analysis, synthesis scaling filter $\{\tilde{h}_i = (-1)^{(i+1)} g_{N-1-i}\}_{i=0}^{N-1}$ and synthesis wavelet filter $\{\tilde{g}_i = (-1)^i h_{N-1-i}\}_{i=0}^{N-1}$. These kernels should obey the following conditions: admissibility (7), orthogonality (8) and regularity (9). [3]

$$\sum_{k=0}^{N-1} h_k = \sqrt{2}, \sum_{k=0}^{N-1} \tilde{h}_k = \sqrt{2} \tag{7}$$

$$\sum_{k=0}^{N-1-2n} h_k \tilde{h}_{k+2n} = \delta_n \ \forall n = 0, \ldots, N/2-1 \tag{8}$$

$$\sum_{k=0}^{N-1} k^{p-1} g_k = 0, \sum_{k=0}^{N-1} k^{p-1} \tilde{g}_k = 0 \ \forall p = 1, \ldots, N/2 \tag{9}$$

These conditions give us the following loss functions constructed with the same procedure described above: admissibility (10), orthogonality (11) and regularity (12) losses.

$$\mathcal{L}_{WN} = \left( \sum_{k=0}^{N-1} h_k - \sqrt{2} \right)^2 + \left( \sum_{k=0}^{N-1} \tilde{h}_k - \sqrt{2} \right)^2 \tag{10}$$

$$\mathcal{L}_{O} = \left( \sum_{k=0}^{N-1} h_k \tilde{h}_k - 1 \right)^2 + \sum_{n=1}^{N/2-1} \left( \sum_{k=0}^{N-1-2n} h_k \tilde{h}_{k+2n} \right)^2 \tag{11}$$

$$\mathcal{L}_R = \sum_{p=1}^{P}\left(\frac{\sqrt{2}}{2^P}\sum_{k=0}^{N-1}k^{p-1}g_k\right)^2 + \sum_{p=1}^{P}\left(\frac{\sqrt{2}}{2^P}\sum_{k=0}^{N-1}k^{p-1}\tilde{g}_k\right)^2 \tag{12}$$

The three loss functions are combined in a single wavelet loss function:

$$\mathcal{L}_{WL} = \lambda_1\mathcal{L}_{WN} + \lambda_2\mathcal{L}_O + \lambda_3\mathcal{L}_R$$

where $\lambda_1, \lambda_2, \lambda_3$ are weights, each equal to one by default in our implementation, $\mathcal{L}_{WN}, \mathcal{L}_O, \mathcal{L}_R$ are (4), (5), (6) for orthonormal wavelets and (10), (11), (12) for biorthogonal wavelets.

The proposed loss functions can be used as weight regularization for wavelet blocks as part of the entire training process to obtain specific behaviors unique to the data and the problem being solved.

**Padding**

The parameters of the convolution operation should be carefully designed to obey the relation (13).

$$\frac{W - K + 2P}{S} + 1 = C \tag{13}$$

The parameters are input length $W$, length of kernel $K$, padding $P$ for each side, stride $S$ and desired output length $C$. That means $K$ must be even. We implement several extension methods:

- "constant": $0\,0\,|\,x_1 \ldots x_n\,|\,0\,0$;
- "circular": $x_{n-1}\,x_n\,|\,x_1 \ldots x_n\,|\,x_1\,x_2$;
- "replicate": $x_1\,x_1\,|\,x_1 \ldots x_n\,|\,x_n\,x_n$;
- "reflect": $x_3\,x_2\,|\,x_1 \ldots x_n\,|\,x_{n-1}\,x_{n-2}$;
- "antireflect": $(2x_1 - x_3)\,(2x_1 - x_2)\,|\,x_1 \ldots x_n\,|\,(2x_n - x_{n-1})\,(2x_n - x_{n-2})$.

"circular" is the default method used in wavelet analysis and provides the perfect reconstruction, while "antireflect" happens to be the most natural one.

**Wavelet Transform Blocks**

The main proposed blocks perform one- and two-dimensional multilevel wavelet transforms, requiring only the size of a kernel being even (for these blocks $W = 2C$, $S = 2$, so according to relation (13), $P = (K-2)/2$). In one-dimensional case convolution with scaling kernel produces approximation and convolution with wavelet kernel produces details, then on the next level analysis can be performed once again on approximation. In two-dimensional

case one-dimensional analysis is first performed on image's rows and then on columns of both approximation and details from the previous step producing four components. The next level analysis can be performed on the approximation of approximation from the previous level. The most natural "antireflect" padding is the default one, because it is better suited for windowed signals.

The one- and two-dimensional inverse wavelet transform (synthesis) blocks are provided for kernels of length $4k+2$ (the condition is obtained from the relation (13) with $W=C$, $S=1$, the size of the synthesis kernel is half the size of the analysis one and the requirement of $P$ to be an integer). The parameters of the inverse transform blocks should be explicitly specified either by providing wavelet name from the PyWavelets library or the kernels obtained from computing the output of the analysis block with "return_filters" flag on. These blocks are also convolutional, but the stride is one and kernels used are computed from provided one's by dividing them into kernels with two output channels: one kernel's channel contains reversed weights from odd positions and the other — reversed weights from even positions. In one-dimensional case the result of convolution of approximation and details with corresponding odd kernels are summed up and placed on odd positions in the resulting signal, even positions are formed by the same process. In two-dimensional case the analysis is reversed in the same way: from final four components approximation and details are obtained, and then the original signal is reconstructed from them.

## Conclusion

The novel trainable blocks for one- and two-dimensional multi-channel wavelet transform were introduced. The inverse wavelet transform blocks for wavelets with kernels of size $4k+2$ implemented. The wavelet blocks may be trained via proposed loss functions or used with user-defined wavelets. The package is posted on PyPi: https://pypi.org/project/waveletnn. Source code is available at https://github.com/Scurrra/WaveletNN-PyTorch.

## References

1. *Amara Graps.* An Introduction to Wavelets. IEEE Comp. Sci. Engi.. 1995.

2. *Ingrid Daubechies.* Orthonormal bases of compactly supported wavelets II: variations on a theme. Siam Journal on Mathematical Analysis. 1993. № 24. С. 499-519.

3. *A. Cohen, Ingrid Daubechies, and J.-C. Feauveau.* Biorthogonal bases of compactly supported wavelets. Communications on Pure and Applied Mathematics. 1992. № 45(5). С. 485-560.