The Journal of Open Source Software

TFDWT: Fast Discrete Wavelet Transform TensorFlow Layers

Kishore K. Tarafdar and **Vikram M. Gadre**

Department of Electrical Engineering, Indian Institute of Technology Bombay, India

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

1 Summary

TFDWT is an open-source Python library that allows the construction of TensorFlow Layers for Fast Discrete Wavelet Transform (DWT) and Inverse Discrete Wavelet Transform (IDWT) in end-to-end backpropagation learning networks. These layers facilitate the construction of multilevel DWT filter banks and Wavelet Packet Transform (WPT) filter banks for a spatial-frequency representation of the inputs and features in shallow or deep networks. A multiresolution signal representation using a multi-rate discrete wavelet system creates enriched joint natural-frequency representations. The discrete wavelet system partitions the frequency plane into subbands using orthogonal dilated and translated lowpass scaling and highpass wavelet function. A realization of a fast discrete wavelet system is a two-band perfect reconstruction multi-rate filter bank with FIR filters corresponding to the impulse responses of the scaling and wavelet function with downsampling and upsampling operations. A filter bank for a higher dimensional input is a seamless extension by successive separable circular convolutions across each independent axis. The command `pip install TFDWT` installs the latest version of the package.

2 Statement of need

In machine or deep learning, an efficient multiresolution representation of data often helps to build economical and explainable models. The Wavelet toolbox (Misiti et al., 1996) by MathWorks is a proprietary software that has served the requirements for D-dimensional wavelet transforms in the MATLAB environment for a few decades. Several open-source packages are now available for 1D and 2D DWT in Python. Pywavelets (Lee et al., 2019) is a D-dimensional wavelet transform library in Python that works with Numpy (Harris et al., 2020) arrays. However, it is challenging to directly use Pywavelets with the symbolic tensors in TensorFlow (Developers, 2022) layers and CUDA (Fatica, 2008). WaveTF (Versaci, 2021) is a solution for constructing 1D and 2D DWT layers in TensorFlow but is limited to only Haar and Db2 wavelets. The package tensorflow-wavelets (Leiderman, 2025) supports multiple wavelets, but it has a minor bug in perfect reconstruction due to the padding and boundary effects in processing the finite-length inputs. In Pytorch (Imambi et al., 2021), the pytorch-wavelets (Cotter, 2023) package allows the construction of 1D and 2D DWT layers. However, there are limited libraries for 3D and higher transforms with a wide range of wavelet families for Graphics Processing Unit (GPU) computations.

For a D-dimensional wavelet $\psi \in L^2(\mathbb{R})^{\mathsf{D}}$, a discrete wavelet system defined by $\{\psi_{m,\boldsymbol{p}} : m \in \mathbb{Z}, \boldsymbol{p} \in \mathbb{Z}^{\mathsf{D}}\}$ forms an orthonormal basis in $\psi \in L^2(\mathbb{R})^{\mathsf{D}}$, where $\psi_{m,\boldsymbol{p}}(\boldsymbol{x}) := 2^m \psi(2^m \boldsymbol{x} - \boldsymbol{p})$. Then, by definition the DWT of $\boldsymbol{x} \in \mathbb{Z}^{\mathsf{D}}$ is $\boldsymbol{x} \mapsto (\langle \boldsymbol{x}, \psi_{m,\boldsymbol{p}} \rangle)_{m,\boldsymbol{p}}$, where m is the dilation parameter and $\boldsymbol{p}$ is the shift or translation parameter. The TFDWT Python package is a simple standalone DWT and IDWT library with minimal dependencies that allow computation with symbolic tensors and CUDA. This release supports up to 3D forward and inverse transforms with

various orthogonal and biorthogonal wavelet families. A seamless package upgrade for higher dimensional DWT is possible by separable transforms. The boundary effects are taken care of with cyclic convolutions instead of padding. The package supports orthogonal and biorthogonal wavelets of different families having impulse responses of diverse lengths. In this paper, we defined the platform-independent, underlying mathematics for realizing fast DWT and IDWT layers with filter bank structures having FIR filters, downsamplers and upsamplers. Although our realization of the discrete wavelet system is in TensorFlow 2, a seamless reproduction of the computations is possible in other deep learning frameworks.

3 Discrete wavelet system for sequences

A discrete wavelet system has a lowpass scaling function and a highpass wavelet. The realization of a discrete wavelet system with a continuous one-dimensional (D=1) mother wavelet $\psi \in L^2(\mathbb{R})$ is by using the impulse responses of the scaling function and the wavelet as Finite Impulse Response (FIR) filters $g[n]$ and $h[n]$. Figure 1 shows wavelets and scaling functions of the analysis and synthesis bank of bior3.1 wavelet and their corresponding impulse responses are shown in Figure 2. These FIR filters are the building blocks of a two-band perfect reconstruction filter bank for realizing fast discrete wavelet systems. Figure 3 shows a two-band perfect reconstruction filter bank that operates on one-dimensional inputs, i.e., sequences in $l^2(\mathbb{Z})$. The analysis and synthesis bank in orthogonal wavelet systems have identical lowpass and highpass FIR filters but differ in biorthogonal wavelet systems. In biorthogonal wavelet filter banks, $\tilde{g}[n]$ and $\tilde{h}[n]$ are the lowpass and highpass filters of the synthesis bank. The only difference in the biorthogonal families like bior and rbio is the interchange of the analysis and synthesis scaling and wavelets functions, for example, in bior3.1 and rbio3.1.

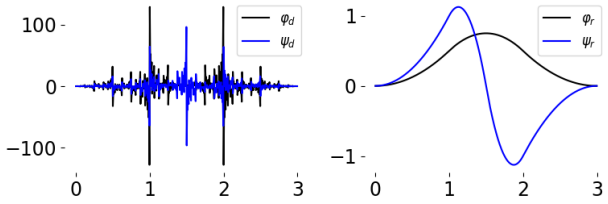

Figure 1: Wavelets and scaling functions of bior3.1 analysis (left) and synthesis (right).

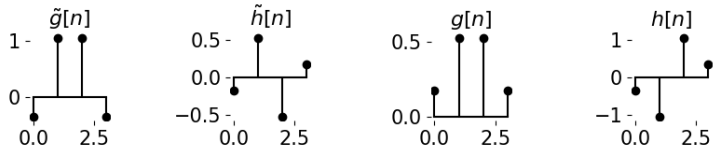

Figure 2: Impulse responses of different bior3.1 analysis (left two) and synthesis (right two) lowpass and highpass FIR filters.

3.1 Circular convolution operators

The four matrices in the two band perfect reconstruction filter bank in Figure 3 are — (i) $\boldsymbol{G}$ is lowpass analysis matrix, (ii) $\boldsymbol{H}$ is highpass analysis matrix, (iii) $\tilde{\boldsymbol{G}}$ is lowpass synthesis matrix and (iv) $\tilde{\boldsymbol{H}}$ is highpass synthesis matrix. These matrices are operators for circular convolution, constructed by circular shifts of the corresponding FIR filters $g[n-k]$, $h[n-k]$, $\tilde{g}[n-k]$ and $\tilde{h}[n-k]$, where $g = \tilde{g}$ and $h = \tilde{h}$ for orthogonal wavelets.

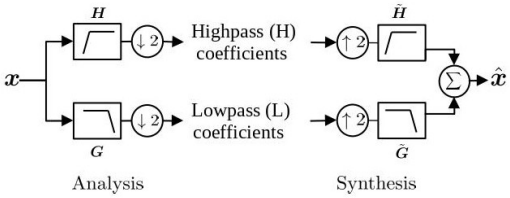

Figure 3: Two band perfect reconstruction filter bank.

3.1.1 Analysis matrix

For a lowpass analysis FIR filter of length L and a input sequence of length N, the circular convolution operator is a matrix $\boldsymbol{G}$ of shape $N \times N$. A downsampling by two $f_{(\downarrow 2)}$ of the output the convolution is equivalent to getting rid of the even rows of $\boldsymbol{G}$ to give $\frac{N}{2} \times N$ operator $f_{(\downarrow 2)}\boldsymbol{G}$. Similarly, for the highpass analysis FIR filter of same wavelet with convolution and downsampling is a $\frac{N}{2} \times N$ operator $f_{(\downarrow 2)}\boldsymbol{H}$. The analysis matrix is,

$$\boldsymbol{A} = \begin{bmatrix} f_{(\downarrow 2)}\boldsymbol{G} \\ f_{(\downarrow 2)}\boldsymbol{H} \end{bmatrix}_{N \times N}$$

where $\boldsymbol{A}$ is a $N \times N$ decimated circular convolution operator formed by combining the lowpass and highpass decimated operators].

3.1.2 Synthesis matrix

The synthesis matrix $\boldsymbol{S}$ is another $N \times N$ decimated circular convolution operator as given by equation [eq:SynthesisMatrix],

$$\boldsymbol{S} = \begin{bmatrix} f_{(\downarrow 2)}\tilde{\boldsymbol{G}} \\ f_{(\downarrow 2)}\tilde{\boldsymbol{H}} \end{bmatrix}^{T}_{N \times N}$$

where $\tilde{\boldsymbol{G}}$ and $\tilde{\boldsymbol{H}}$ are matrices formed by the lowpass and highpass synthesis FIR filters. The above is a general representation for both orthogonal and biorthogonal wavelets families where for orthogonal wavelets, $\tilde{\boldsymbol{G}} = \boldsymbol{G}$, $\tilde{\boldsymbol{H}} = \boldsymbol{H}$ and thus $\boldsymbol{S} = \boldsymbol{A}^T$.

A two-band perfect reconstruction discrete wavelet system for one-dimensional inputs is given by the analysis equation and the synthesis equation,

$$\boldsymbol{q} = \text{DWT}(\boldsymbol{x}) \text{ or, } \boldsymbol{q} = \left(\boldsymbol{A}\boldsymbol{x}^T\right)^T \text{—Analysis}$$

$$\boldsymbol{x} = \text{IDWT}(\boldsymbol{q}) \text{ or, } \boldsymbol{x} = \left(\boldsymbol{S}\boldsymbol{q}^T\right)^T \text{—Synthesis}$$

where $\boldsymbol{A}$ and $\boldsymbol{S}$ are analysis and synthesis matrices, $\boldsymbol{x}$ is a input sequence and $\boldsymbol{q}$ has a distinct lowpass and a highpass subband.

Example 1. Given, a sequence $\boldsymbol{x} \in \mathbb{R}^8$ or $N = 8$ and FIR filters length $L = 6$.

$$\text{LPF and downsampling } f_{(\downarrow 2)}\boldsymbol{G} = \begin{bmatrix} g_1 & g_0 & 0 & 0 & g_5 & g_4 & g_3 & g_2 \\ g_3 & g_2 & g_1 & g_0 & 0 & 0 & g_5 & g_4 \\ g_5 & g_4 & g_3 & g_2 & g_1 & g_0 & 0 & 0 \\ 0 & 0 & g_5 & g_4 & g_3 & g_2 & g_1 & g_0 \end{bmatrix}_{\frac{N}{2} \times N}$$

$$\text{HPF and downsampling } f_{(\downarrow 2)}\boldsymbol{H} = \begin{bmatrix} h_1 & h_0 & 0 & 0 & h_5 & h_4 & h_3 & h_2 \\ h_3 & h_2 & h_1 & h_0 & 0 & 0 & h_5 & h_4 \\ h_5 & h_4 & h_3 & h_2 & h_1 & h_0 & 0 & 0 \\ 0 & 0 & h_5 & h_4 & h_3 & h_2 & h_1 & h_0 \end{bmatrix}_{\frac{N}{2}\times N}$$

$$\text{Analysis matrix is } \boldsymbol{A} = \begin{bmatrix} g_1 & g_0 & 0 & 0 & g_5 & g_4 & g_3 & g_2 \\ g_3 & g_2 & g_1 & g_0 & 0 & 0 & g_5 & g_4 \\ g_5 & g_4 & g_3 & g_2 & g_1 & g_0 & 0 & 0 \\ 0 & 0 & g_5 & g_4 & g_3 & g_2 & g_1 & g_0 \\ h_1 & h_0 & 0 & 0 & h_5 & h_4 & h_3 & h_2 \\ h_3 & h_2 & h_1 & h_0 & 0 & 0 & h_5 & h_4 \\ h_5 & h_4 & h_3 & h_2 & h_1 & h_0 & 0 & 0 \\ 0 & 0 & h_5 & h_4 & h_3 & h_2 & h_1 & h_0 \end{bmatrix}_{N\times N}$$

Similarly,

$$\text{Synthesis matrix is } \boldsymbol{S} = \begin{bmatrix} \tilde{g}_1 & \tilde{g}_0 & 0 & 0 & \tilde{g}_5 & \tilde{g}_4 & \tilde{g}_3 & \tilde{g}_2 \\ \tilde{g}_3 & \tilde{g}_2 & \tilde{g}_1 & \tilde{g}_0 & 0 & 0 & \tilde{g}_5 & \tilde{g}_4 \\ \tilde{g}_5 & \tilde{g}_4 & \tilde{g}_3 & \tilde{g}_2 & \tilde{g}_1 & \tilde{g}_0 & 0 & 0 \\ 0 & 0 & \tilde{g}_5 & \tilde{g}_4 & \tilde{g}_3 & \tilde{g}_2 & \tilde{g}_1 & \tilde{g}_0 \\ \tilde{h}_1 & \tilde{h}_0 & 0 & 0 & \tilde{h}_5 & \tilde{h}_4 & \tilde{h}_3 & \tilde{h}_2 \\ \tilde{h}_3 & \tilde{h}_2 & \tilde{h}_1 & \tilde{h}_0 & 0 & 0 & \tilde{h}_5 & \tilde{h}_4 \\ \tilde{h}_5 & \tilde{h}_4 & \tilde{h}_3 & \tilde{h}_2 & \tilde{h}_1 & \tilde{h}_0 & 0 & 0 \\ 0 & 0 & \tilde{h}_5 & \tilde{h}_4 & \tilde{h}_3 & \tilde{h}_2 & \tilde{h}_1 & \tilde{h}_0 \end{bmatrix}_{N\times N}$$

The DWT of $\boldsymbol{x}$ produces subbbands,

$$\boldsymbol{q} = \text{DWT}(\boldsymbol{x}) \text{ or, } \boldsymbol{q} = \boldsymbol{A}\boldsymbol{x}$$

Perfect reconstruction,

$$\boldsymbol{x} = \text{IDWT}(\boldsymbol{q}) \text{ or, } \boldsymbol{x} = \boldsymbol{S}\boldsymbol{q}$$

3.2 DWT 1D layer

A DWT 1Dayer operates on input tensors of shape (batch, length, channels) and produces an output of shape $(\text{batch}, \text{length}/2, 2\times \text{channels})$. as described in Algorithm 1a.

Algorithm 1a —

1. Input $\boldsymbol{X}$ of shape (batch, length, channels).
2. Generate analysis matrix $\boldsymbol{A}$ using length of input.
3. For each batched channel $\boldsymbol{x} \in \boldsymbol{X}$ of shape (batch, length): $\boldsymbol{q}_c = \boldsymbol{A}\boldsymbol{x}^T$
4. Stacking for all c channels: $\boldsymbol{Q} := (\boldsymbol{q}_c)_{\forall c}$ to a shape (batch, length, channels).
5. Group subbands and return an output $\boldsymbol{Q}^{(\text{grouped})}$ of shape $(\text{batch}, \text{length}/2, , 2\times\text{channels})$

```
# Grouping two subbands in DWT 1D
mid = int(Q.shape[1]/2)
L = Q[:,:mid,:]
H = Q[:,mid:,:]
out = Concatenate([L, H], axis=-1)
```

3.3 IDWT 1D layer

An IDWT 1D layer operates on input tensors of shape $(\text{batch}, \text{length}/2, 2 \times \text{channels})$ and produces an output of shape (batch, length, channels) as described in Algorithm 1b.

Algorithm 1b —

1. Input $\boldsymbol{Q}^{(\text{grouped})}$ of shape $(\text{batch}, \text{length}/2, 2 \times \text{channels})$
2. Ungroup the subbands to get $\boldsymbol{Q}$ of shape (batch, length, channels)
3. Generate synthesis matrix $\boldsymbol{S}$ using length of $\boldsymbol{Q}$.
4. For each batched channel $\boldsymbol{q} \in \boldsymbol{Q}$ of shape (batch, length): $\boldsymbol{x} = \boldsymbol{S}\boldsymbol{q}^T$, i.e., a perfect reconstruction, where, $\boldsymbol{S} = \boldsymbol{A}^T$ for orthogonal wavelets
5. Layer output (perfect reconstruction): $\boldsymbol{X} := (\boldsymbol{x}_c)_{\forall c}$ is of shape (batch, length, channels)

4 Higher dimensional discrete wavelet systems

In sequences (1D), the DWT 1D applies along the only independent variable. To achieve higher dimensional DWT, the same DWT 1D is successively applied separably to all the independent variables. For example, in an image (2D) the DWT 2D is a row-wise DWT 1D followed by a column-wise DWT 1D. Similarly, the reconstruction is column-wise IDWT 1D followed by a row-wise IDWT 1D.

4.1 Two-dimensional discrete wavelet system

The pixel values in an image are a function of two independent spatial axes. A DWT 2D filter bank is a separable transform with row-wise filtering followed by column-wise filtering that yields four subbands - LL, LH, HL and HH. A two-dimensional discrete wavelet system is,

$$\boldsymbol{q} = \text{DWT}(\boldsymbol{x}) := \boldsymbol{A}(\boldsymbol{A}\boldsymbol{x}_{021})^T_{021} \qquad \text{—Analysis}$$

$$\boldsymbol{x} = \text{IDWT}(\boldsymbol{q}) := \boldsymbol{S}(\boldsymbol{S}\left[\boldsymbol{A}(\boldsymbol{A}\boldsymbol{x}^T_{021})^T_{021}\right]^T_{021})^T_{021} \quad \text{—Synthesis}$$

where, $\boldsymbol{A}$ and $\boldsymbol{S}$ are the same analysis and synthesis matrices defined for one-dimensional wavelet system. Figure 4 shows an $N \times N$ image and its four localized spatial-frequency subbands after DWT. Here, the low-frequency band is LL, and the other three are high-frequency subbands representing horizontal, vertical and diagonal features. Figure 5 illustrates a separable 2D DWT perfect reconstruction filter bank. The 2D layers operate on batched, multichannel tensors of shape (batch, height, width, channels), where each image is of shape height and width. Figure 6 illustrates input, output and perfect reconstruction by DWT 2D and IDWT 2D layers. The Multiresolution Encoder-Decoder Convolutional Neural Network in (Tarafdar et al., 2025) uses these forward and inverse transform layers.

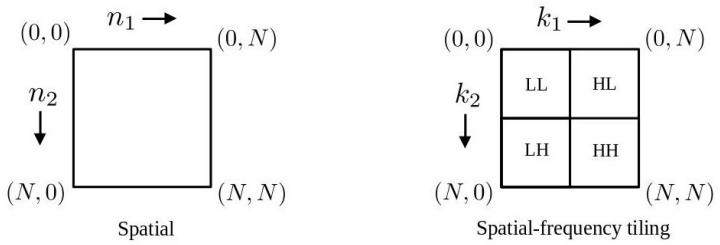

Figure 4: Natural domain (left) and spatial-frequency tiling (right) after a DWT 2D.

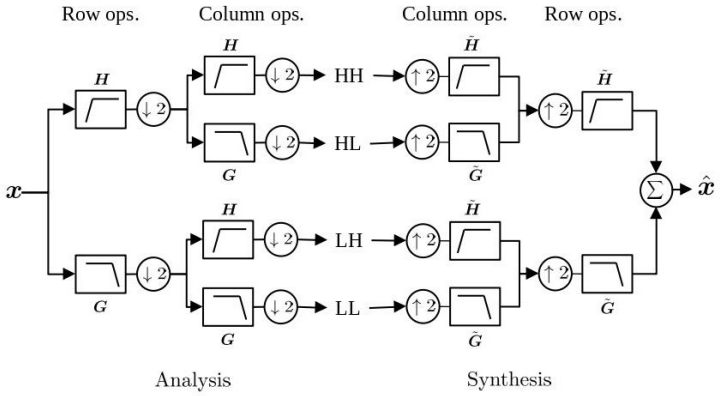

Figure 5: Separable DWT 2D perfect reconstruction filter bank.

4.1.1 DWT 2D layer

A DWT 2D layer operates on input tensors of shape (batch, height, width, channels) and produces an output of shape (batch, height$/2$, width$/2$, $4\times$channels) as described in Algorithm 2a.

***Algorithm 2a* —**

1. Input $\boldsymbol{X}$ of shape (batch, height, width, channels).
2. Generate analysis matrix $\boldsymbol{A}$ using height and width of input.
3. For each batched channel $\boldsymbol{x}_c \in \boldsymbol{X}$ of shape (batch, height, width):
   (omitting suffix c in $\boldsymbol{x}$ below for simplicity of notation)
   1. Row-wise batch DWT 1D: $\boldsymbol{A}\boldsymbol{x}_{021}^T := \text{Einsum}(ij, bjk \to bik)$
   2. Column-wise batch DWT 1D: $\boldsymbol{A}(\boldsymbol{A}\boldsymbol{x}_{021}^T)_{021}^T := \text{Einsum}(ij, bjk \to bik)$ Or, equivalently, DWT of a batched channel $\boldsymbol{x}$ is,

$$\boldsymbol{q}_c = \text{DWT}(\boldsymbol{x}) \text{ or, } \boldsymbol{q}_c = \boldsymbol{A}(\boldsymbol{A}\boldsymbol{x}_{021})_{021}^T$$

   where, the suffix 021 in $\boldsymbol{x}_{021}^T$ denotes permutation of axis, i.e., transpose.
4. Stacking for all c channels: $\boldsymbol{Q} := (\boldsymbol{q}_c)_{\forall c}$ to a shape (batch, height, width, channels).
5. Group subbands and return an output $\boldsymbol{Q}^{(\text{grouped})}$ of shape (batch, height$/2$, width$/2$, $4\times$ channels)

```
# Grouping four subbands in DWT 2D
mid = int(Q.shape[1]/2)
LL = Q[:,:mid,:mid,:]
LH = Q[:,mid:,:mid,:]
HL = Q[:,:mid,mid:,:]
HH = Q[:,mid:,mid:,:]
output = Concatenate([LL, LH, HL, HH], axis=-1)
```

4.1.2 IDWT 2D layer

An IDWT 2D layer operates on input tensors of shape (batch, height$/2$, width$/2$, $4\times$channels) and produces an output of shape (batch, height, width, channels) as described in Algorithm 2b.

***Algorithm 2b* —**

1. Input $\boldsymbol{Q}^{(\text{grouped})}$ of shape (batch, height$/2$, width$/2$, $4\times$ channels)

2. Ungroup the subbands to get $\boldsymbol{Q}$ of shape (batch, height, width, channels)
3. Generate synthesis matrix $\boldsymbol{S}$ using height and width of $\boldsymbol{Q}$.
4. For each batched channel $\boldsymbol{q}_c \in \boldsymbol{Q}$ of shape (batch, height, width):
   (omitting suffix c in $\boldsymbol{q}$ below for simplicity of notation)
   1. Row-wise batch IDWT 1D: $\boldsymbol{S}\boldsymbol{q}^T_{021} := \text{Einsum}(ij, bjk \to bik)$
   2. Column wise batch IDWT 1D: $\boldsymbol{S}(\boldsymbol{S}\boldsymbol{q}^T_{021})^T_{021} := \text{Einsum}(ij, bjk \to bik)$ or equivalently, a perfect reconstruction,

$$\boldsymbol{x} = \text{IDWT}(\boldsymbol{q}) \text{ or, } \boldsymbol{x} = \boldsymbol{S}(\boldsymbol{S}\boldsymbol{q}^T_{021})^T_{021}$$

   where, the suffix 021 in $\boldsymbol{x}^T_{021}$ denotes permutation of axis, i.e., transpose and $\boldsymbol{S} = \boldsymbol{A}^T$ for orthogonal wavelets
5. Layer output : $\boldsymbol{X} := (\boldsymbol{x}_c)_{\forall c}$ is of shape (batch, height, width, channels)
   (Perfect reconstruction)

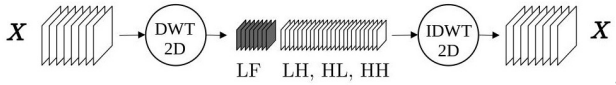

Figure 6: DWT decomposition and perfect reconstruction of a multichannel image tensor.

4.2 Three-dimensional discrete wavelet system

A three-dimensional (3D) discrete wavelet system for a 3D input $\boldsymbol{x}$ is given by,

$$\boldsymbol{q} = \text{DWT}(\boldsymbol{x}) := [\boldsymbol{A}(\boldsymbol{A}(\boldsymbol{A}\boldsymbol{x}^T_{0213})^T_{0213})^T_{0132}]^T_{0132} \quad \text{—Analysis}$$

$$\boldsymbol{x} = \text{IDWT}(\boldsymbol{q}) := (\boldsymbol{S}(\boldsymbol{S}(\boldsymbol{S}[\boldsymbol{A}(\boldsymbol{A}(\boldsymbol{A}\boldsymbol{x}^T_{0213})^T_{0213})^T_{0132}]^T_{0132}{}^T_{0132})^T_{0132})^T_{0213})_{0213} \quad \text{—Synthesis}$$

where, $\boldsymbol{A}$ and $\boldsymbol{S}$ are the same analysis and synthesis matrices as defined for one-dimensional wavelet system. The DWT 3D and IDWT 3D layers operate on batched, multichannel tensors of shape (batch, height, width, depth, channels).

4.2.1 DWT 3D layer

A DWT 3D layer operates on input tensors of shape (batch, height, width, depth, channels) and produces an output of shape (batch, height$/2$, width$/2$, depth$/2$, $8\times$channels) as described in Algorithm 3a.

***Algorithm 3a* —**

1. Input $\boldsymbol{X}$ of shape (batch, height, width, depth, channels).
2. Generate analysis matrix $\boldsymbol{A}$ using height, width and depthof input.
3. For each batched channel $\boldsymbol{x}_c \in \boldsymbol{X}$ of shape (batch, height, width, depth):
   (omitting suffix c in $\boldsymbol{x}$ below for simplicity of notation)
   1. Row-wise batch DWT 1D: $\boldsymbol{A}\boldsymbol{x}^T_{0213} := \text{Einsum}(ij, bjkl \to bikl)$
   2. Column-wise batch DWT 1D: $\boldsymbol{A}\left(\boldsymbol{A}\boldsymbol{x}^T_{0213}\right)^T_{0213} := \text{Einsum}(ij, bjkl \to bikl)$
   3. Depth-wise batch IDWT 1D: $\boldsymbol{A}(\boldsymbol{A}(\boldsymbol{A}\boldsymbol{x}^T_{0213})^T_{0213})^T_{0132} := \text{Einsum}(ik, bjkl \to bjil)$
   Therefore, DWT of $\boldsymbol{x}$ yield coefficients:

$$\boldsymbol{q}_c := \text{DWT}\left(\boldsymbol{x}\right) = [\boldsymbol{A}(\boldsymbol{A}(\boldsymbol{A}\boldsymbol{x}^T_{0213})^T_{0213})^T_{0132}]^T_{0132}$$

4. Stacking for all c channels as $\boldsymbol{Q} := (\boldsymbol{q}_c)_{\forall c}$ to a shape (batch, height, width, depth, channels).
5. Group subbands and return an output $\boldsymbol{Q}^{(\text{grouped})}$ of shape (batch, height/2, width/2, depth/2, 8×channels).

```
# Grouping Eight subbands in 3D DWT
mid = int(Q.shape[2]/2)
LLL = Q[:,:mid,:mid,:mid,:]
LLH = Q[:,mid:,:mid,:mid,:]
LHL = Q[:,:mid,mid:,:mid,:]
LHH = Q[:,mid:,mid:,:mid,:]
HLL = Q[:,:mid,:mid,mid:,:]
HLH = Q[:,mid:,mid:,mid:,:]
HHL = Q[:,:mid,mid:,mid:,:]
HHH = Q[:,_mid:,mid:,mid:,:]
output = Concatenate([LLL, LLH, LHL, LHH, HLL, HLH, HHL, HHH], axis=-1)
```

4.2.2 IDWT 3D layer

An IDWT 3D layer operates on input tensors of shape $(\text{batch}, \text{height}/2, \text{width}/2, \text{depth}/2, 8 \times \text{channels})$ and produces an output of shape (batch, height, width, depth, channels) as described in Algorithm 3b.

Algorithm 3b —

1. Input $\boldsymbol{Q}^{(\text{grouped})}$ of shape $(\text{batch}, \text{height}/2, \text{width}/2, \text{depth}/2, 8 \times \text{channels})$
2. Ungroup to get $\boldsymbol{Q}$ of shape (batch, height, width, depth, channels)
3. Generate synthesis matrix $\boldsymbol{S}$ using height, width and depth of $\boldsymbol{Q}$.
4. For each batched channel $\boldsymbol{q}_c \in \boldsymbol{Q}$ of shape (batch, height, width, depth): (omitting suffix c in $\boldsymbol{q}$ below for simplicity of notation)
   1. Row-wise batch IDWT 1D: $\boldsymbol{S}\boldsymbol{q}^T_{0132} := \text{Einsum}(ik, bjkl \to bjil)$
   2. Column-wise batch IDWT 1D: $\boldsymbol{S}(\boldsymbol{S}\boldsymbol{q}^T_{0132})_{0132} := \text{Einsum}(ij, bjkl \to bikl)$
   3. Depth-wise batch IDWT 1D: $\boldsymbol{S}(\boldsymbol{S}(\boldsymbol{S}\boldsymbol{q}^T_{0132})^T_{0132})^T_{0213} := \text{Einsum}(ij, bjkl \to bikl)$ or equivalently, a perfect reconstruction,

$$\boldsymbol{x}_c := \text{IDWT}\,(\boldsymbol{q}) = \left[\boldsymbol{S}(\boldsymbol{S}(\boldsymbol{S}\boldsymbol{q}^T_{0132})^T_{0132})^T_{0213}\right]^T_{0213}$$

   where, $\boldsymbol{S} = \boldsymbol{A}^T$ for orthogonal wavelets.
5. Layer output : $\boldsymbol{X} := (\boldsymbol{x}_c)_{\forall c}$ is of shape (batch, height, width, depth, channels) (Perfect reconstruction)

In general, a seamless realization of fast D-dimensional DWT and IDWT is possible by extending the above separable method to all the independent N axes one after the other. The number of subbands will be equal to 2^{D} for a D dimensional DWT. For example, sequences $(\text{D} = 1)$ yield two subbands, images $(\text{D} = 2)$ yield four subbands, three-dimensional inputs $(\text{D} = 3)$ with voxels yield eight subbands etc.

5 Multilevel wavelet filter banks

The above-discussed DWT and IDWT layers are building blocks in constructing multilevel DWT filter banks. Figure 7 shows the partitioning of the 1D frequency axis and tiling of the 2D frequency plane using a level-4 1D and 2D DWT. The multilevel DWT successively decomposes the low-frequency feature. If the high-frequency features are also decomposed successively, then we get a Wavelet Packet Transform (WPT) filter bank.

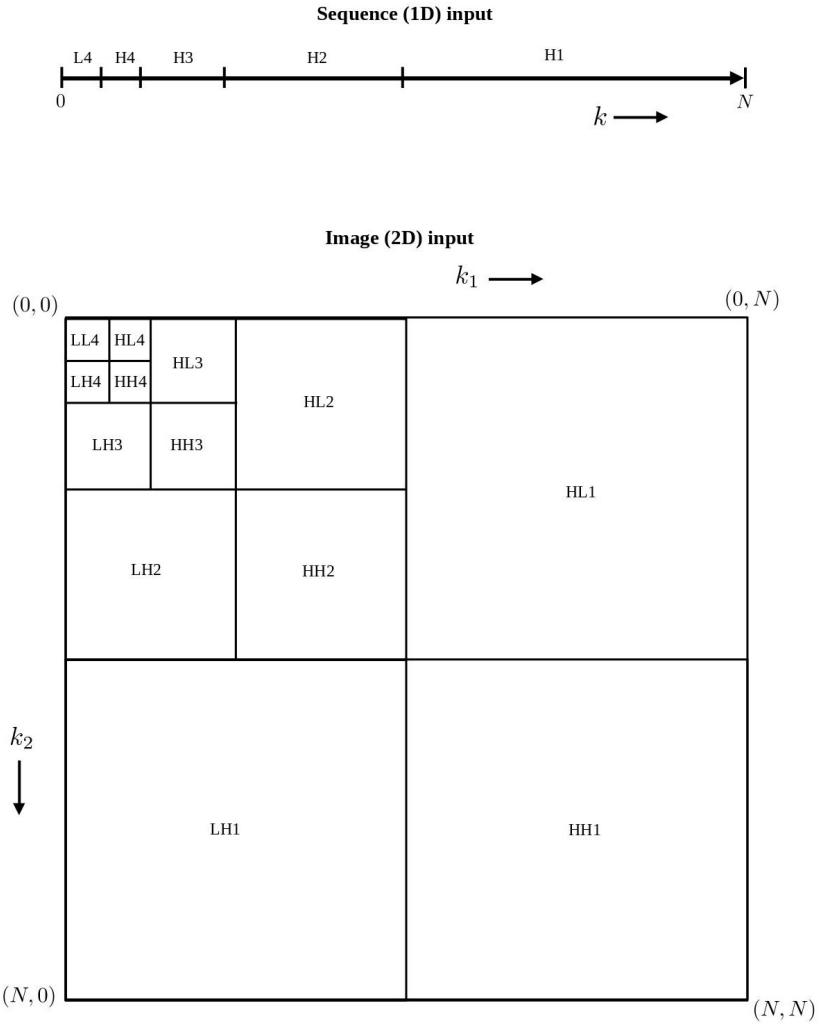

Figure 7: Spatial-frequency tiling by DWT level 4 decomposition of a sequence of length N (top) and image of shape Nimes N (bottom).

References

Cotter, F. (2023). *Pytorch-wavelets: A port of the DTCWT toolbox to run on pytorch*. https://pypi.org/project/pytorch-wavelets/

Developers, T. (2022). TensorFlow. *Zenodo*.

Fatica, M. (2008). CUDA toolkit and libraries. *2008 IEEE Hot Chips 20 Symposium (HCS)*, 1–22.

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Imambi, S., Prakash, K. B., & Kanagachidambaresan, G. (2021). PyTorch. *Programming with TensorFlow: Solution for Edge Computing Applications*, 87–104.

Lee, G., Gommers, R., Waselewski, F., Wohlfahrt, K., & O'Leary, A. (2019). PyWavelets: A

python package for wavelet analysis. *Journal of Open Source Software*, *4*(36), 1237.

Leiderman, T. (2025). *Tensorflow-wavelets: Tensorflow wavelet layers*. https://pypi.org/project/tensorflow-wavelets/

Misiti, M., Misiti, Y., Oppenheim, G., & Poggi, J.-M. (1996). Wavelet toolbox. *The MathWorks Inc., Natick, MA*, *15*, 21.

Tarafdar, K. K., Meher, A., Shah, M., Saifee, Q., Kumar, D., Nimkar, A. V., Jayasundar, R., & Gadre, V. M. (2025). Multiresolution encoder-decoder convolutional neural network for magnetic resonance image segmentation. *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5.

Versaci, F. (2021). WaveTF: A fast 2D wavelet transform for machine learning in keras. *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part i*, 605–618.