# Chapter 1

# Learning in the Wavelet Domain

In this chapter we move away from the ScatterNet ideas from the previous chapters and instead look at using the wavelet domain as a new space in which to learn. In particular the ScatterNet, and even the learnable ScatterNet proposed in the previous chapter, are built around taking complex magnitudes of the highpass wavelets. This inherently builds invariance to shifts but at the cost of making things smoother. In many ways this was beneficial, as it allowed us to subsample the output and we saw that the scattering layers worked well right before the downsampling stages of a CNN. However, we would now like to explore if it is possible and at all beneficial to learn with wavelets without taking the complex magnitude. This means that the frequency support of our activations will remain in the same space in the Fourier domain.

The inspiration to this chapter is the hope that learning in the frequency/wavelet domain may afford simpler filters than learning in the pixel domain. A classic example of this is the first layer filters in AlexNet shown in Figure 1.1. These are a collection of lowpass (with colour) and oriented highpass filters, each of which would only require a few parameters if coded as subband gains.

Our experiments show that

As neural network training involves presenting thousands of training samples, we want our layer to be fast. To achieve this we would ideally choose to use a critically sampled filter bank implementation. The fast 2-D Discrete Wavelet Transform (DWT) is a possible option, but it has two drawbacks: it has poor directional selectivity and any alteration of wavelet coefficients will cause the aliasing cancelling properties of the reconstructed signal to disappear. Instead we choose to use the Dual-Tree Complex Wavelet Transform ($DTCWT$) [3] as at the expense of limited redundancy (4:1), it enables us to have better directional selectivity, and allows us to modify the wavelet coefficients and still have minimal aliasing terms when we reconstruct [4].

section 1.4 of the paper describes the implementation details of our design, and section 1.8 describes the experiments and results we have done so far.

## 1.1 Related Work

### 1.1.1 Wavelets as a Front End

Fujieda et. al. use a DWT in combination with a CNN to do texture classification and image annotation [5], [6]. In particular, they take a multiscale wavelet transform of the input image,

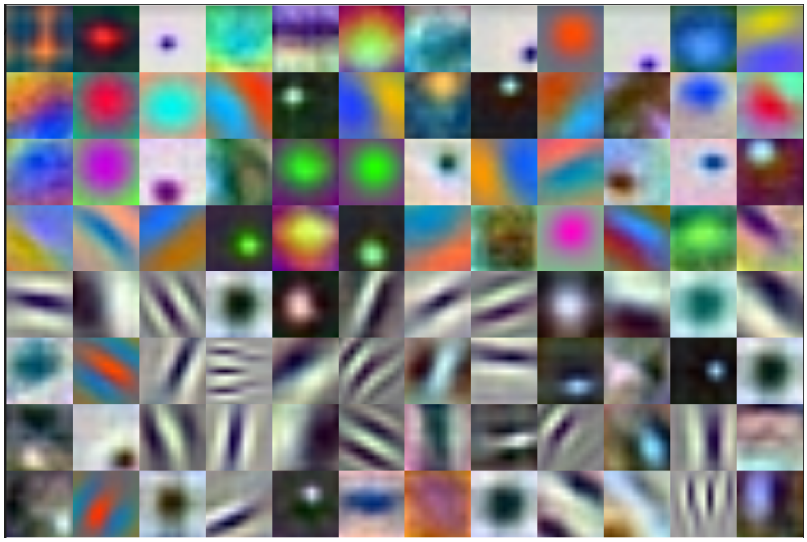


Figure 1.1: **First layer filters of the AlexNet architecture.** The first layer filters of the seminal AlexNet [1] are an inspiration for considering learning filters in the wavelet domain. Each of these $11 \times 11$ filters would only require a handful of non zero coefficients in the wavelet domain. Weights taken from torchvision [2].

combine the actviations at each scale independently with learned weights, and feed these back into the network where the activation resolution size matches the subband resolution. The architecture block diagram is shown in Figure 1.2, taken from the original paper. This work found that their dubbed 'Wavelet-CNN' could outperform competetive non wavelet based CNNs on both texture classification and image annotation.

Several works also use wavelets in deep neural networks for super-resolution [7] and for adding detail back into dense pixel-wise segmentation tasks [8]. These typically save wavelet coefficients and use them for the reconstruction phase, so are a little less applicable than the first work.

### 1.1.2 Parameterizing filters in Fourier Domain

In "Spectral Representations for Convolutional Neural Networks" [9], Rippel et. al. explore parameterization of filters in the DFT domain. Note that they do not necessarily do the convolution in the Frequency domain, they simply parameterize a filter $\mathbf{w} \in \mathbb{R}^{F \times C \times K \times K}$ as a set of fourier coefficients $\hat{\mathbf{w}} \in \mathbb{C}^{F \times C \times K \times \lceil K/2 \rceil}$ (the reduced spatial size is a result of enforcing that the inverse DFT of their filter to be real, so the parameterization is symmetric). On the forward pass of the neural network, they take the inverse DFT of $\hat{\mathbf{w}}$ to obtain $\mathbf{w}$ and then convolve this with the input $\mathbf{x}$ as a normal CNN would do.[1].

[1]The convolution may be done by taking both the image and filter back into the fourier space but this is typically decided by the framework, which selects the optimal convolution strategy for the filter and input size. Note that there is not necessarily a saving to be gained by enforcing it to do convolution by product of FFTs, as the FFT size needed for the filter will likely be larger than $K \times K$, which would require resampling the coefficients

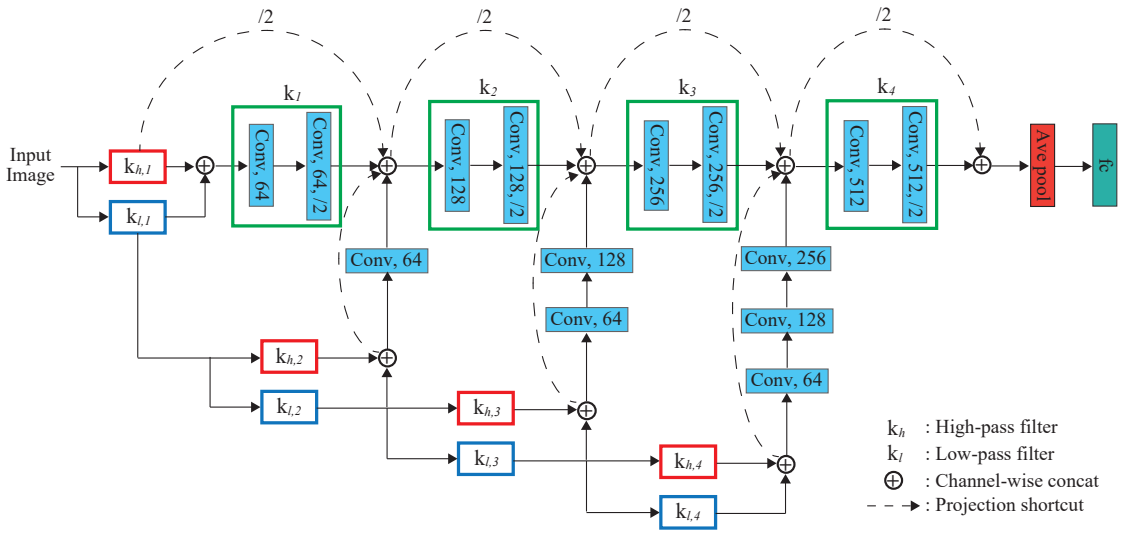



Figure 1.2: **Architecture using the DWT as a frontend to a CNN.** Figure 1 from [6]. Fujieda et. al. take a multiscale wavelet decomposition of the input before passing the input through a standard CNN. They learn convolutional layers independently on each subband and feed these back into the network at different depths, where the resolution of the subband and the network activations match.

## 1.2 Invertible Transforms and Optimization

Note that an important point should be laboured about reparameterizing filters in either the wavelet or Fourier domains. That is that any invertible linear transform of the parameter space will not change the updates if a linear optimization scheme (like standard gradient descent, or SGD with momentum) is used.

To see this, let us consider the work from [9] where filters are parameterized in the Fourier domain.

If we define the DFT as the orthonormal version, i.e. let:

$$U_{ab} = \frac{1}{\sqrt{N}} \exp\{\frac{-2j\pi ab}{N}\}$$

then call $X = \text{DFT}\{x\}$. In matrix form the 2-D DFT is then:

$$X = \text{DFT}\{x\} = UxU \tag{1.2.1}$$
$$x = \text{DFT}^{-1}\{X\} = U^*YU^* \tag{1.2.2}$$

When it comes to gradients, these become:

$$\frac{\partial L}{\partial X} = U\frac{\partial L}{\partial x}U = \mathrm{DFT}\left\{\frac{\partial L}{\partial x}\right\} \tag{1.2.3}$$

$$\frac{\partial L}{\partial x} = U^*\frac{\partial L}{\partial X}U^* = \mathrm{DFT}^{-1}\left\{\frac{\partial L}{\partial X}\right\} \tag{1.2.4}$$

Now consider a single filter parameterized in the DFT and spatial domains presented with the exact same data and with the same $\ell_2$ regularization $\epsilon$ and learning rate $\eta$. Let the spatial filter at time $t$ be $\mathbf{w}_t$, the Fourier-parameterized filter be $\hat{\mathbf{w}}_t$, and let

$$\hat{\mathbf{w}}_1 = \mathrm{DFT}\{\mathbf{w}_1\} \tag{1.2.5}$$

After presenting both systems with the same minibatch of samples $\mathcal{D}$ and calculating the gradient $\frac{\partial L}{\partial \mathbf{w}}$ we update both parameters:

$$\mathbf{w}_2 = \mathbf{w}_1 - \eta\left(\frac{\partial L}{\partial \mathbf{w}} + \epsilon\mathbf{w}_1\right) \tag{1.2.6}$$

$$= (1-\eta\epsilon)\mathbf{w}_1 - \eta\frac{\partial L}{\partial \mathbf{w}} \tag{1.2.7}$$

$$\hat{\mathbf{w}}_2 = \hat{\mathbf{w}}_1 - \eta\left(\frac{\partial L}{\partial \hat{\mathbf{w}}} + \epsilon\hat{\mathbf{w}}_1\right) \tag{1.2.8}$$

$$= (1-\eta\epsilon)\hat{\mathbf{w}}_1 - \eta\frac{\partial L}{\partial \hat{\mathbf{w}}} \tag{1.2.9}$$

$$\tag{1.2.10}$$

Where we have shortened the gradient of the loss evaluated at the current parameter values to $\delta_{\mathbf{w}}$ and $\delta_{\hat{\mathbf{w}}}$. We can then compare the effect the new parameters would have on the next minibatch by calculating $\mathrm{DFT}^{-1}\{\hat{\mathbf{w}}_2\}$. Using equations 1.2.3 and 1.2.5 we then get:

$$\mathrm{DFT}^{-1}\{\hat{\mathbf{w}}_2\} = \mathrm{DFT}^{-1}\left\{(1-\eta\epsilon)\hat{\mathbf{w}}_1 - \eta\,\frac{\partial L}{\partial \hat{\mathbf{w}}}\right\} \tag{1.2.11}$$

$$= (1-\eta\epsilon)\mathbf{w}_1 - \eta\ \mathrm{DFT}^{-1}\left\{\frac{\partial L}{\partial \hat{\mathbf{w}}}\right\} \tag{1.2.12}$$

$$= (1-\eta\epsilon)\mathbf{w}_1 - \eta\frac{\partial L}{\partial \mathbf{w}} \tag{1.2.13}$$

$$= \mathbf{w}_2 \tag{1.2.14}$$

### 1.2.1 Regularization

If we use $\ell_1$ then the above doesn't hold.

### 1.2.2 Optimization

If we us adam things r different.

This does not hold for the Adam [10] or Adagrad optimizers, which automatically rescale the learning rates for each parameter based on estimates of the parameter's variance. Rippel et. al. use this fact in their paper [9].

## 1.3 A Summary of Choices

As mentioned in the inspiration for this chapter, many filters that have complex support in the pixel domain would have simple support in the wavelet domain, but as the previous section showed, naively reparameterizing things in a different domain may not afford us any benefit in the optimization procedure.

There are two possible ways we can try to leverage the wavelet domain for learning:

1. We can reparameterize filters in the wavelet domain if we use nonlinear optimizers like ADAM, or $\ell_1$ regularization to impose sparsity.
2. We can take wavelet transforms of the inputs and learn filters on the activations. Doing the latter would require a little care though. In the wavelet domain we can then apply new nonlinearities like wavelet shrinkage. On the output of this, we have the choice of either staying in the wavelet domain or returning to the pixel domain with an inverse wavelet transform.

This chapter explores both possible methods and the merits and drawbacks of each.

## 1.4 Method

## 1.5 Aliasing in the DWT

Consider a single level critically sampled DWT in 1-D. The aliasing cancelling condition is:

$$G_0(z)H_0(-z) + G_1(z)H_1(-z) = 0$$

This is typically solved by using Quadrature Mirror Filters, i.e.

$$H_1(z) = H_0(-z) \tag{1.5.1}$$
$$G_0(z) = H_0(z) \tag{1.5.2}$$
$$G_1(z) = -H_1(z) = -H_0(-z) \tag{1.5.3}$$

## 1.6 Shift Invariance of the DT$\mathbb{C}$WT

Firstly, let us look at what would happen if we retained only 1 of the subbands. Note that we have to keep the same band from each tree. E.g. if we wanted to keep the $x_{001}$ coefficients, we must keep $x_{001a}$ and $x_{001b}$. For any pair of coefficients on the tree, this would look like:

e.g. if we kept $x_{001a}$ and $x_{001b}$ then $M = 8$ and $A(z) = H_{0a}(z)H_{00a}(z^2)H_{001a}(z^4)$ is the transfer fucntion from $x$ to $x_{001a}$. The transfer functions for $B(z)$, $C(z)$ and $D(z)$ are obtained similarly. It is well known that:

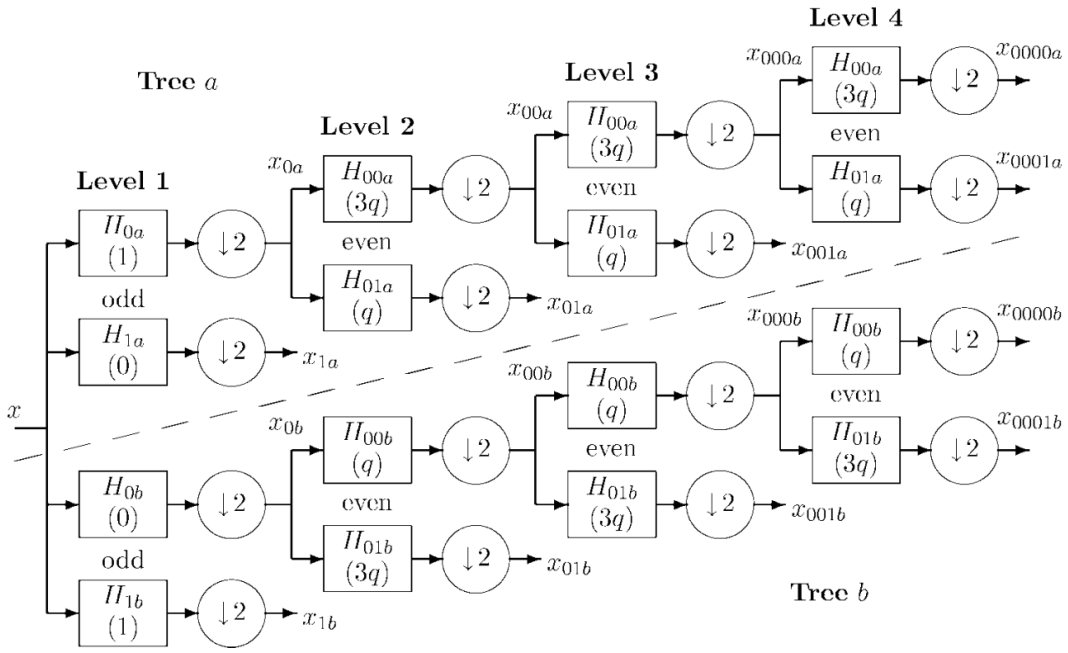



**FIG. 3.** The Q-shift version of the DT CWT, giving real and imaginary parts of complex coefficients from tree $a$ and tree $b$, respectively. Figures in brackets indicate the delay for each filter, where $q = \frac{1}{4}$ sample period.

Figure 1.3: **Full 1-D** DT$\mathbb{C}$WT**.**

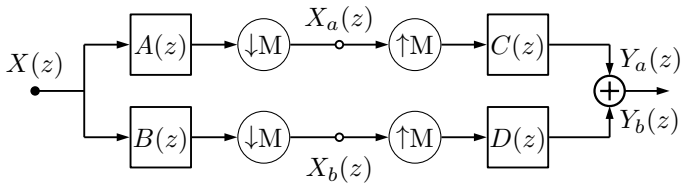
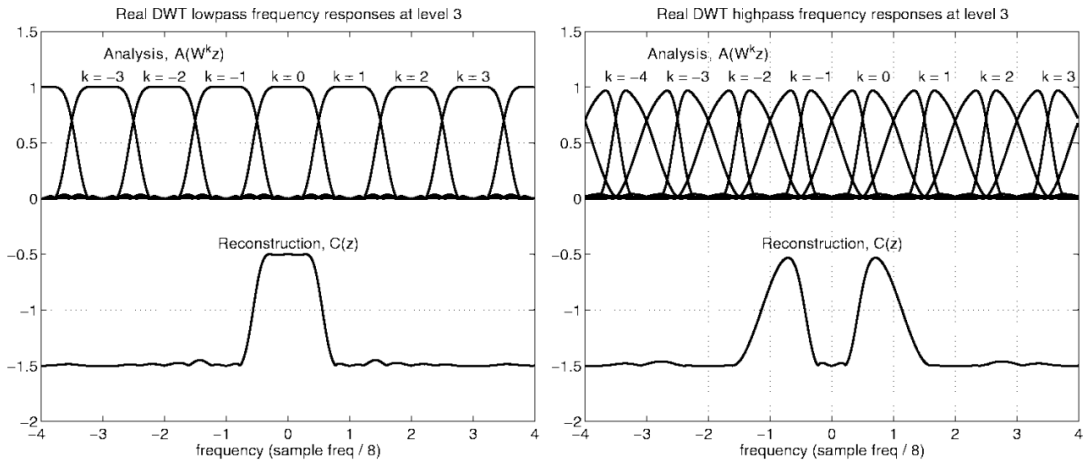



Figure 1.4: **Block Diagram of 1-D** DT$\mathbb{C}$WT**.** Note the top and bottom paths are through the wavelet or scaling functions from just level m ($M = 2^m$). Figure based on Figure 4 in [4].




(a) (b)

$$U(z)\downarrow M \quad \rightarrow \quad U(z^M) \tag{1.6.1}$$

$$U(z)\uparrow M \quad \rightarrow \quad \frac{1}{M}\sum_{k=0}^{M-1} U(W^k z^{1/M}) \tag{1.6.2}$$

Where $W = e^{j2\pi/M}$. So downsampling followed by upsampling becomes:

$$U(z)\downarrow M\uparrow M \rightarrow \frac{1}{M}\sum_{k=0}^{M-1} U(W^k z)$$

This means that

$$Y(z) = Y_a(z) + Y_b(z) = \frac{1}{M}\sum_{k=0}^{M-1} X(W^k z)[A(W^k z)C(z) + B(W^k z)D(z)]$$

The aliasing terms for which are everywhere where $k \neq 0$ (as $X(W^k z)$ is $X(z)$ shifted by $\frac{2k\pi}{M}$). I.e. to avoid aliasing in this reduced tree, we want to make sure that $A(W^k z)C(z) + B(W^k z)D(z) = 0$ for all $k \neq 0$.

The figure below (Fig 5 from [4]) shows what $A(W^k z)$ and $C(z)$ look like for both the lowpass case (left) and the highpass case (right). Note that these responses will be similar for $B(W^k z)$ and $D(z)$. For large values of k, there is almost no overlap (i.e. $A(W^k z)C(z) \approx B(W^k z)D(z) \approx 0$, but for small values of k (particularly $k = \pm 1$), the transition bands have significant overlap with the central response. It is here that we need to use the flexibility of having 2 trees to ensure that $A(W^k z)C(z)$ and $B(W^k z)D(z)$ cancel out.

To do this, let us consider the lowpass filters first. If we let:

$$B(z) = z^{\pm M/2}A(z) \quad (1.6.3)$$
$$D(z) = z^{\mp M/2}C(z) \quad (1.6.4)$$

Then

$$A(W^k z)C(z)+B(W^k z)D(z) = A(W^k z)C(z)+(W^k z)^{\pm M/2}A(W^k z)z^{\mp M/2}C(z) \quad (1.6.5)$$
$$= A(W^k z)C(z)+e^{\frac{jk2\pi}{M}\times(\pm\frac{M}{2})}z^{\pm M/2}z^{\mp M/2}A(W^k z)C(z) \quad (1.6.6)$$
$$= A(W^k z)C(z)+(-1)^k A(W^k z)C(z) \quad (1.6.7)$$

Which cancels when k is odd

Now consider the bandpass case. For shifts of $k=1$ the right half of the left peak overlaps with the left half of the right peak. For a shift of $k=2$, the left half of the left peak overlaps with the right half of the right peak. Similarly for $k=-1$ and $k=-2$. For $|k|>2$, there is no overlap. The fact that we have overlaps at both even and odd shifts of $k$ means that we can't use the same clever trick from before. However, what we do note is that the overlap is always caused by opposite peaks (i.e. the left with the right peak, and never the left with itself, or the right with itself). The solution then is to have $B$ and $D$ have upper and lower passbands of opposite polarity, whil $A$ and $C$ should have passbands of the same polarity.

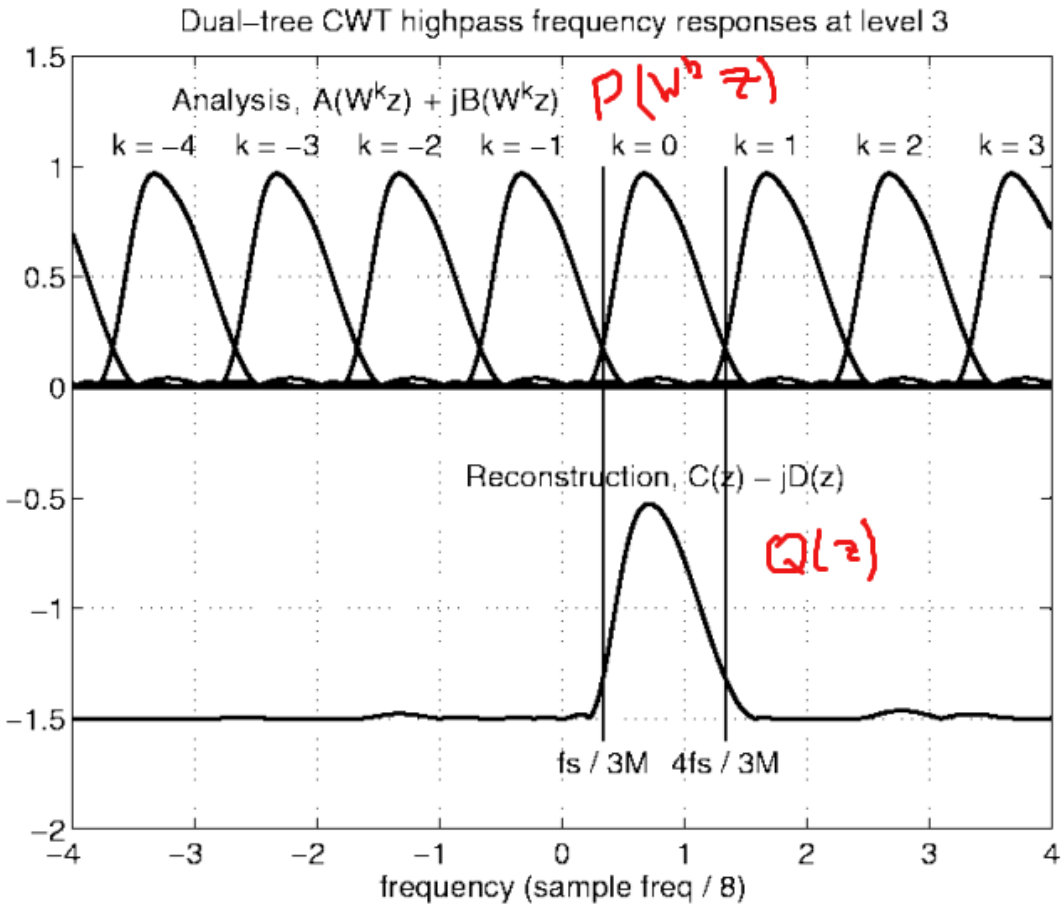
Consider two prototype complex filters $P(z)$ and $Q(z)$ each with single passbands going from $f_s/2M \to f_s/M$ (or $\frac{pi}{M} \to \frac{2*pi}{M}$) - they must be complex to have support in only one half of the frequency plane. Now say $P^*(z)=\sum_r p_r^* z^{-r}$ is the z-transform of the conjugate of $p_r$, which has support only in the negative half of the frequency plane. Then we can get the required filters by:

$$A(z) = 2\Re[P(z)] = P(z)+P^*(z) \quad (1.6.8)$$
$$B(z) = 2\Im[P(z)] = -j[P(z)-P^*(z)] \quad (1.6.9)$$
$$C(z) = 2\Re[Q(z)] = Q(z)+Q^*(z) \quad (1.6.10)$$
$$D(z) = -2\Im[Q(z)] = j[Q(z)-Q^*(z)] \quad (1.6.11)$$

Then:

$$A(W^k z)C(z)+B(W^k z)D(z) = [P(W^k z)+P^*(W^k z)][Q(z)+Q^*(z)]+$$
$$(-j*j)[P(W^k z)-P^*(W^k z)][Q(z)-Q^*(z)] \quad (1.6.12)$$
$$= P(W^k z)Q(z)[1+1]+P^*(W^k z)Q(z)[1-1]+$$
$$P(W^k z)Q^*(z)[1-1]+P^*(W^k z)Q^*(z)[1+1] \quad (1.6.13)$$
$$= 2P(W^k z)Q(z)+2P^*(W^k z)Q^*(z) \quad (1.6.14)$$

So now we only need to ensure that $P(W^k z)$ overlaps as little as possible with $Q(z)$. This is somewhat more manageable, the diagram below shows the problem.


Dual–tree CWT highpass frequency responses at level 3
Analysis, A(W^k z) + jB(W^k z)
P(W^k z)
k = −4
k = −3
k = −2
k = −1
k = 0
k = 1
k = 2
k = 3
Reconstruction, C(z) − jD(z)
Q(z)
fs / 3M
4fs / 3M
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−4
−3
−2
−1
0
1
2
3
4
frequency (sample freq / 8)

(d)

## 1.7 DT$\mathbb{C}$WT Convolution

To do full DT$\mathbb{C}$WT convolution, we need to do more than the above. We don't want to only keep some passbands, but we want to apply different gains to each one. First, let us check that analytically this makes sense and we still won't have aliasing when we do more general operations to the passbands. I.e. instead of considering the above reduced tree, consider the below tree:

$$Y(z) = Y_a(z) + Y_b(z) = \frac{1}{M}\sum_{k=0}^{M-1} X(W^k z)[A(W^k z)G(W^k z^M)C(z) + B(W^k z)H(W^k z^M)D(z)]$$

If we set $H = G$ then we get:

$$Y(z) = Y_a(z) + Y_b(z) = \frac{1}{M}\sum_{k=0}^{M-1} X(W^k z)G(W^k z^M)[A(W^k z)C(z) + B(W^k z)D(z)]$$

And assuming the above analysis holds, we see that we are safe with aliasing, as we know $[A(W^k z)C(z) + B(W^k z)D(z)] \approx 0$ for all $k \neq 0$. This means that our equivalent filter becomes

$$\begin{aligned} Y(z) &= \frac{1}{M}X(z)G(z^M)[A(z)C(z) + B(z)D(z)] && (1.7.1)\\ &= \frac{1}{M}X(z)G(z^M)[P(z)Q(z) + jP^*(z)Q^*(z)] && (1.7.2)\end{aligned}$$

Now we know can assume that our DT$\mathbb{C}$WT is well designed and extracts frequency bands at local areas, then our filter $G(z^M)$ allows us to modify these passbands (e.g. by simply scaling if $G(z) = C$, or by more complex functions.

The output from Figure 1.4 is:

$$Y(z) = Y_a(z) + Y_b(z) = \frac{1}{M}\sum_{k=0}^{M-1} X\left(W^k z\right)\left[A\left(W^k z\right)C(z) + B\left(W^k z\right)D(z)\right]$$

Where $W = e^{j2\pi/M}$. To achieve shift invariance we need $A\left(W^k z\right)C(z) + B\left(W^k z\right)D(z)$ to be very small or to cancel each other out for all $k \neq 0$.

The complex analysis filter (taking us into the wavelet domain) is

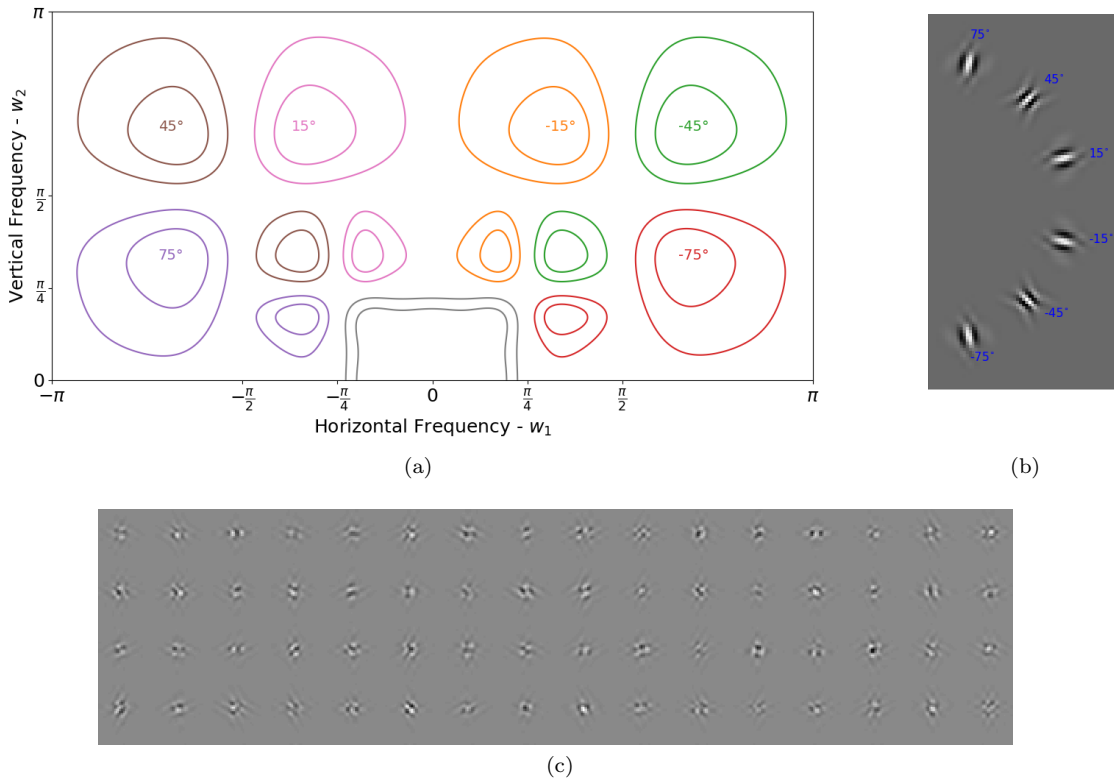
$$P(z) = \frac{1}{2}\left(A(z) + jB(z)\right)$$

and the complex synthesis filter (returning us to the pixel domain) is

$$Q(z) = \frac{1}{2}\left(C(z) - jD(z)\right)$$

where $A, B, C, D$ are real. If $G(z) = G_r(z) + jG_i(z) = 1$ then the end-to-end transfer function is (from section 4 of [4]):

$$\frac{Y(z)}{X(z)} = \frac{2}{M}\left(P(z)Q(z) + P^*(z)Q^*(z)\right) \quad (1.7.3)$$




(a) (b)

(c)

Figure 1.5: (a) Contour plots at -1dB and -3dB showing the support in the Fourier domain of the 6 subbands of the DT$\mathbb{C}$WT at scales 1 and 2 and the scale 2 lowpass. These are the product $P(z)Q(z)$ from Equation 1.7.4.(b) The pixel domain impulse responses for the second scale wavelets. (c) Example impulses of our layer when $g_1$, and $g_{lp}$ are 0 and $g_2 \in \mathbb{C}^{6\times1\times1}$, with each real and imaginary element drawn from $\mathcal{N}(0,1)$. I.e., only information in the 6 subbands with $\frac{\pi}{4} < |w_1|, |w_2| < \frac{\pi}{2}$ from (a) is passed through.

where $P, Q$ have support only in the top half of the Fourier plane and $P^*, Q^*$ are $P$ and $Q$ reflected in the horizontal frequency axis. Examples of $P(z)Q(z)$ for different subbands of a 2-D DT$\mathbb{C}$WT have spectra shown in **??**, $P^*(z)Q^*(z)$ make up the missing half of the frequency space.

In a standard convolutional layer, an input with $C$ channels, $H$ rows and $W$ columns is $X \in \mathbb{R}^{C\times H\times W}$, which is then convolved with $F$ filters of spatial size $K$ - $w \in \mathbb{R}^{F\times C\times K\times K}$, giving $Y \in \mathbb{R}^{F\times H\times W}$. In many systems like **he˙deep˙2015**, [1], the first layer is typically a selection of bandpass filters, selecting edges with different orientations and center frequencies. In the wavelet space this would be trivial - take a decomposition of each input channel and keep individual subbands (or equivalently, attenuate other bands), then take the inverse wavelet transform. Figure 1.5 shows the frequency space for the DT$\mathbb{C}$WT and makes it clearer as to how this could be done practically for a two scale transform. To attenuate all but say the 15° band at the first scale for the first input channel, we would need to have $13C$ gains for the 13 subbands and $C$ input channels,

$13C-1$ of which would be zero and the remaining coefficient one.

Instead of explicitly setting the gains, we can randomly initialize them and use backpropagation to learn what they should be. This gives us the power to learn more complex shapes rather than simple edges, as we can mix the regions of the frequency space per input channel in an arbitrary way.

### 1.7.1 Forward propagation

Figure 1.6 shows the block diagram using $Z$-transforms for a single band of our system (it is based on Figure 4 in [4]). To keep things simple for the rest of section 1.4 the figure shown is for a 1-D system; it is relatively straightforward to extend this to 2-D[3]. The complex analysis filter (taking us into the wavelet domain) is $P(z) = \frac{1}{2}(A(z) + jB(z))$ and the complex synthesis filter (returning us to the pixel domain) is $Q(z) = \frac{1}{2}(C(z) - jD(z))$ where $A, B, C, D$ are real. If $G(z) = G_r(z) + jG_i(z) = 1$ then the end-to-end transfer function is (from section 4 of [4]):

$$\frac{Y(z)}{X(z)} = \frac{2}{M}(P(z)Q(z) + P^*(z)Q^*(z)) \tag{1.7.4}$$

where $P, Q$ have support only in the top half of the Fourier plane and $P^*, Q^*$ are $P$ and $Q$ reflected in the horizontal frequency axis. Examples of $P(z)Q(z)$ for different subbands of a 2-D DT$\mathbb{C}$WT have spectra shown in 1.5a, $P^*(z)Q^*(z)$ make up the missing half of the frequency space.

Modifying this from the standard wavelet equations by adding the subband gains $G_r(z)$ and $G_i(z)$, the transfer function becomes:

$$\frac{Y(z)}{X(z)} = \frac{2}{M}\left[\, G_r(z^M)(P(z)Q(z) + P^*(z)Q^*(z)) + jG_i(z^M)(P(z)Q(z) - P^*(z)Q^*(z))\right] \tag{1.7.5}$$

### 1.7.2 Backpropagation

We start with the commonly known property that for a convolutional block, the gradient with respect to the input is the gradient with respect to the output convolved with the time reverse of the filter. More formally, if $Y(z) = H(z)X(z)$:

$$\Delta X(z) = H(z^{-1})\Delta Y(z) \tag{1.7.6}$$

where $H(z^{-1})$ is the $Z$-transform of the time/space reverse of $H(z)$, $\Delta Y(z) \triangleq \frac{\partial L}{\partial Y}(z)$ is the gradient of the loss with respect to the output, and $\Delta X(z) \triangleq \frac{\partial L}{\partial X}(z)$ is the gradient of the loss with respect to the input. If H were complex, the first term in Equation 1.7.6 would be $\bar{H}(1/\bar{z})$, but as each individual block in the DT$\mathbb{C}$WT is purely real, we can use the simpler form.

Assume we already have access to the quantity $\Delta Y(z)$ (this is the input to the backwards pass). **??** illustrates the backpropagation procedure. An interesting result is that the backwards pass of an inverse wavelet transform is equivalent to doing a forward wavelet transform.[2] Similarly, the backwards pass of the forward transform is equivalent to doing the inverse transform. The weight

[2] As shown in **??**, the analysis and synthesis filters have to be swapped and time reversed. For orthogonal wavelet transforms, the synthesis filters are already the time reverse of the analysis filters, so no change has to be done. The q-shift filters of the DT$\mathbb{C}$WT [11] have this property.

update gradients are then calculated by finding $\Delta W(z) = \text{DT}\mathbb{C}\text{WT}\{\Delta Y(z)\}$ and then convolving with the time reverse of the saved wavelet coefficients from the forward pass - $V(z)$.

$$\Delta G_r(z) = \Delta W_r(z)V_r(z^{-1}) + \Delta W_i(z)V_i(z^{-1}) \tag{1.7.7}$$
$$\Delta G_i(z) = -\Delta W_r(z)V_i(z^{-1}) + \Delta W_i(z)V_r(z^{-1}) \tag{1.7.8}$$

Unsurprisingly, the passthrough gradients have similar form to Equation 1.7.5:

$$\Delta X(z) = \frac{2\Delta Y(z)}{M}\left[G_r(z^{-M})\left(PQ + P^*Q^*\right) + jG_i(z^{-M})\left(PQ - P^*Q^*\right)\right] \tag{1.7.9}$$

where we have dropped the $z$ terms on $P(z), Q(z), P^*(z), Q^*(z)$ for brevity.

Note that we only need to evaluate equations 1.7.7,1.7.8,1.7.9 over the support of $G(z)$ i.e., if it is a single number we only need to calculate $\Delta G(z)|_{z=0}$.

### 1.7.3 Memory Cost

Again considering a two scale transform — instead of learning $w \in \mathbb{R}^{F\times C\times K\times K}$ we learn complex gains at the two scales, and a real gain for the real lowpass:

$$\begin{aligned} g_1 &\in \mathbb{C}^{F\times C\times 6\times 1\times 1} \\ g_2 &\in \mathbb{C}^{F\times C\times 6\times 1\times 1} \\ g_{lp} &\in \mathbb{R}^{F\times C\times 1\times 1} \end{aligned}$$

We have set the spatial dimension to be $1\times 1$ to show that this gain is identical to a $1\times 1$ convolution over the complex wavelet coefficients. If we wish, we can learn larger spatial sizes to have more complex attenuation/magnification of the subbands. We also can use more/fewer than 2 wavelet scales. At first glance, we have increased our parameterization by a factor of 25 (13 subbands, of which all but the lowpass are complex), but each one of these gains affects a large spatial size. For the first scale, the effective size is about $5\times 5$ pixels, for the second scale it is about $15\times 15$.

### 1.7.4 Computational Cost

A standard convolutional layer needs $K^2F$ multiplies per input pixel (of which there are $C\times H\times W$). In comparison, the wavelet gain method does a set number of operations per pixel for the forward and inverse transforms, and then applies gains on subsampled activations. For a 2 level DT$\mathbb{C}$WT the transform overhead is about 60 multiplies for both the forward and inverse transform. It is important to note that unlike the filtering operation, this does not scale with $F$. The learned gains in each subband do scale with the number of output channels, but can have smaller spatial size (as they have larger effective sizes) as well as having fewer pixels to operate on (because of the decimation). The end result is that as $F$ and $C$ grow, the overhead of the $C$ forward and $F$ inverse transforms is outweighed by cost of $FC$ mixing processes, which should in turn be significantly less than the cost of $FC$ $K\times K$ standard convolutions for equivalent spatial sizes.

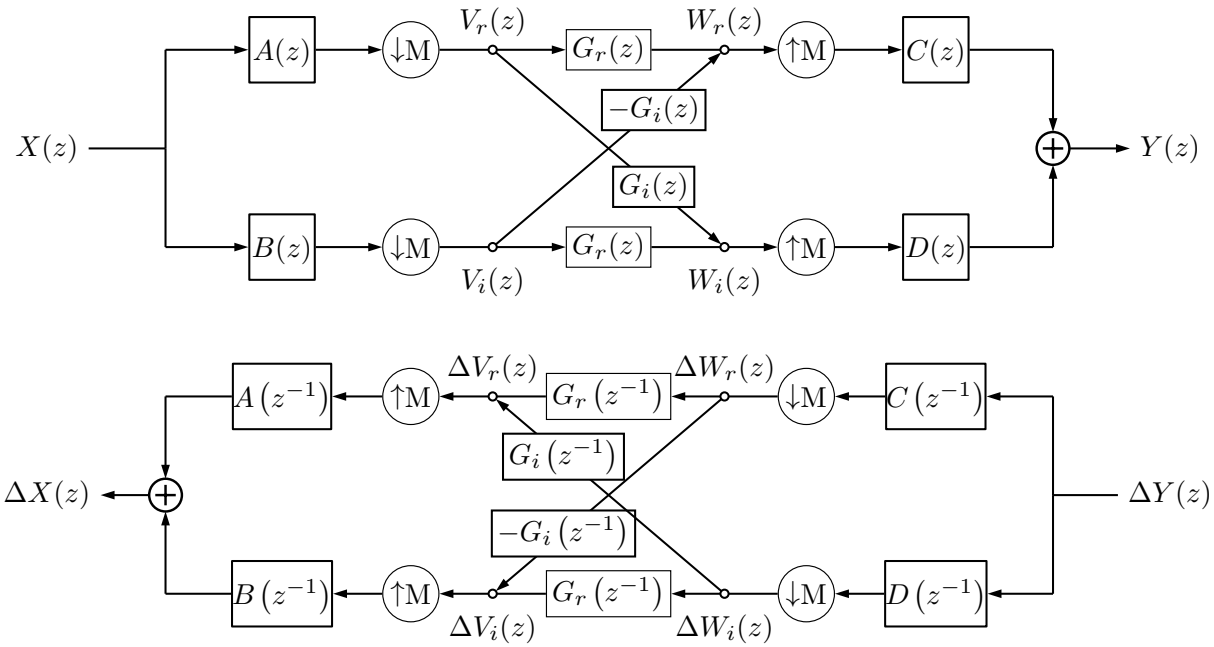



Figure 1.6: **Forward and backward block diagrams for** DT$\mathbb{C}$WT **gain layer.** Based on Figure 4 in [4]. Ignoring the $G$ gains, the top and bottom paths (through $A, C$ and $B, D$ respectively) make up the the real and imaginary parts for *one subband* of the dual tree system. Combined, $A+jB$ and $C-jD$ make the complex filters necessary to have support on one side of the Fourier domain (see Figure 1.5). Adding in the complex gain $G_r+jG_i$, we can now attenuate/shape the impulse response in each of the subbands. To allow for learning, we need backpropagation. The bottom diagram indicates how to pass gradients $\Delta Y(z)$ through the layer. Note that upsampling has become downsampling, and convolution has become convolution with the time reverse of the filter (represented by $z^{-1}$ terms).

### 1.7.5 Examples

1.5c show example impulse responses of our layer. These impulses were generated by randomly initializing both the real and imaginary parts of $g_2 \in \mathbb{C}^{6\times1\times1}$ from $\mathcal{N}(0,1)$ and $g_1, g_{lp}$ are set to 0. I.e. each shape has 12 random variables. It is good to see that there is still a large degree of variability between shapes. Our experiments have shown that the distribution of the normalized cross-correlation between 512 of such randomly generated shapes matches the distribution for random vectors with roughly 11.5 degrees of freedom.

## 1.8 Experiments and Preliminary Results

To examine the effectiveness of our convolutional layer, we do a simple experiment on CIFAR-10 and CIFAR-100. For simplicity, we compare the performance using a simple yet relatively effective convolutional architecture - LeNet [12]. LeNet has 2 convolutional layers of spatial size $5\times5$ followed by 2 fully connected layers and a softmax final layer. We swap both these convolutional layers out

Table 1.1: Comparison of LeNet with standard convolution to our proposed method which learns in the wavelet space (WaveLenet) on CIFAR-10 and CIFAR-100. Values reported are the average top-1 accuracy (%) rates for different train set sizes over 5 runs.

| | Train set size | 1000 | 2000 | 5000 | 10000 | 20000 | 50000 |
|---|---|---|---|---|---|---|---|
| CIFAR-10 | LeNet | 48.5 | 52.4 | 59.5 | 65.0 | 69.5 | 73.3 |
| | WaveLeNet | 47.3 | 52.1 | 58.7 | 63.8 | 68.0 | 72.4 |
| CIFAR-100 | LeNet | 11.1 | 15.8 | 23.1 | 29.5 | 34.4 | 41.1 |
| | WaveLeNet | 11.1 | 15.4 | 23.2 | 28.4 | 33.9 | 39.6 |

for two of our proposed wavelet gain layers (keeping the ReLU between them). As CIFAR has very small spatial size, we only take a single scale DT$\mathbb{C}$WT. Therefore each gain layer has 6 complex gains for the 6 subbands, and a $3 \times 3$ real gain for the lowpass (a total of $21C$ parameters vs $25C$ for the original system). We train both networks for 200 epochs with Adam [10] optimizer with a constant learning rate of $10^{-3}$ and a weight decay of $10^{-5}$. The code is available at [13]. Table 1.1 shows the mean of the validation set accuracies for 5 runs. The different columns represent undersampled training set sizes (with 50000 being the full training set). When undersampling, we keep the samples per class constant. We see our system perform only very slightly worse than the standard convolutional layer.

## 1.9 Conclusion and Future Work

In this work we have presented the novel idea of learning filters by taking activations into the wavelet domain, learning mixing coefficients and then returning to the pixel space. This work is done as a preliminary step; we ultimately hope that learning in both the wavelet and pixel space will have many advantages, but as yet it has not been explored. We have considered the possible challenges this proposes and described how a multirate system can learn through backpropagation.

Our experiments so far have been promising. We have shown that our layer can learn in an end-to-end system, achieving very near similar accuracies on CIFAR-10 and CIFAR-100 to the same system with convolutional layers instead. This is a good start and shows the plausibility of such an idea, but we need to search for how to improve these layers if they are to be useful. It will be interesting to see how well we can learn on datasets with larger images - our proposed method naturally learns large kernels, so should scale well with the image size.

In our experiments so far, we only briefly go into the wavelet domain before coming back to the pixel domain to do ReLU nonlinearities, however we plan to explore using nonlinearities in the wavelet domain, such as soft-shrinkage to denoise/sparsify the coefficients [14]. We feel there are strong links between ReLU non-linearities and denoising/sparsity ideas, and that there may well be useful performance gains from mixing real pixel-domain non-linearities with complex wavelet-domain shrinkage functions. Thus we present these ideas here as a starting point for a novel and exciting avenue of deep network research.

# References


[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", in *NIPS*, Curran Associates, Inc., 2012, pp. 1097–1105.

[2] S. Marcel and Y. Rodriguez, "Torchvision the Machine-vision Package of Torch", in *Proceedings of the 18th ACM International Conference on Multimedia*, ser. MM '10, New York, NY, USA: ACM, 2010, pp. 1485–1488.

[3] I. W. Selesnick, R. G. Baraniuk, and N. G. Kingsbury, "The dual-tree complex wavelet transform", *Signal Processing Magazine, IEEE*, vol. 22, no. 6, pp. 123–151, 2005.

[4] N. Kingsbury, "Complex wavelets for shift invariant analysis and filtering of signals", *Applied and Computational Harmonic Analysis*, vol. 10, no. 3, pp. 234–253, May 2001.

[5] S. Fujieda, K. Takayama, and T. Hachisuka, "Wavelet Convolutional Neural Networks for Texture Classification", *arXiv:1707.07394 [cs]*, Jul. 2017. arXiv: 1707.07394 [cs].

[6] ——, "Wavelet Convolutional Neural Networks", *arXiv:1805.08620 [cs]*, May 2018. arXiv: 1805.08620 [cs].

[7] T. Guo, H. S. Mousavi, T. H. Vu, and V. Monga, "Deep Wavelet Prediction for Image Super-Resolution", in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Honolulu, HI, USA: IEEE, Jul. 2017, pp. 1100–1109.

[8] L. Ma, J. Stückler, T. Wu, and D. Cremers, "Detailed Dense Inference with Convolutional Neural Networks via Discrete Wavelet Transform", *arXiv:1808.01834 [cs]*, Aug. 2018. arXiv: 1808.01834 [cs].

[9] O. Rippel, J. Snoek, and R. P. Adams, "Spectral Representations for Convolutional Neural Networks", in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., Curran Associates, Inc., 2015, pp. 2440–2448.

[10] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization", *arXiv:1412.6980 [cs]*, Dec. 2014. arXiv: 1412.6980 [cs].

[11] N. Kingsbury, "Design of Q-shift complex wavelets for image processing using frequency domain energy minimization", in *2003 International Conference on Image Processing, 2003. ICIP 2003. Proceedings*, vol. 1, Sep. 2003, I-1013-16 vol.1.

[12] 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, Nov. 1998.

[13] F. Cotter, *DTCWT Gainlayer*, Nov. 2018.

[14] D. L. Donoho and J. M. Johnstone, "Ideal spatial adaptation by wavelet shrinkage", en, *Biometrika*, vol. 81, no. 3, pp. 425–455, Sep. 1994.