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 just before 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 could be parameterized with only a few nonzero wavelet coefficients, or alternatively, we could take a decomposition of each input channel and keep individual subbands (or equivalently, attenuate other bands), then take the inverse wavelet transform.

Our experiments show that ... **FINISH ME**

1.1 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 ℓ_1 regularization to impose sparsity. This is presented in section 1.3.

2. We can take wavelet transforms of the inputs and learn filters on the wavelet coefficients. We can also apply nonlinearities to the wavelet coefficients such as 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 is presented in section 1.4

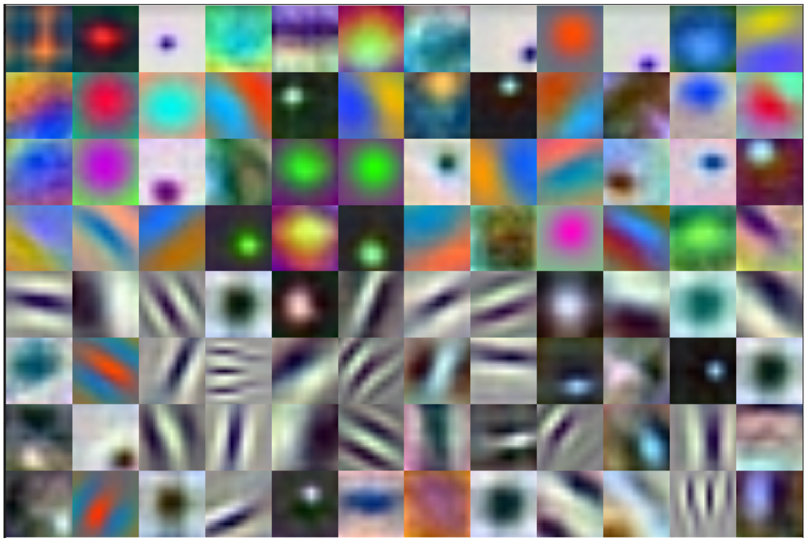


Figure 1.1: **First layer filters of the AlexNet architecture.** The first layer filters of the seminal AlexNet [1] are an inspiration for considering learning filters in the wavelet domain. Each of these 11×11 filters would only require a handful of non zero coefficients in the wavelet domain. The weights shown here were taken from a pretrained network from torchvision [2].

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

1.2 Related Work

1.2.1 Wavelets as a Front End

Fujieda et. al. use a DWT in combination with a CNN to do texture classification and image annotation [3], [4]. In particular, they take a multiscale wavelet transform of the input image, 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 [5] and for adding detail back into dense pixel-wise segmentation tasks [6]. These typically save wavelet coefficients and use them for the reconstruction phase, so are a little less applicable than the first work.

1.2.2 Parameterizing filters in Fourier Domain

In "Spectral Representations for Convolutional Neural Networks" [7], 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

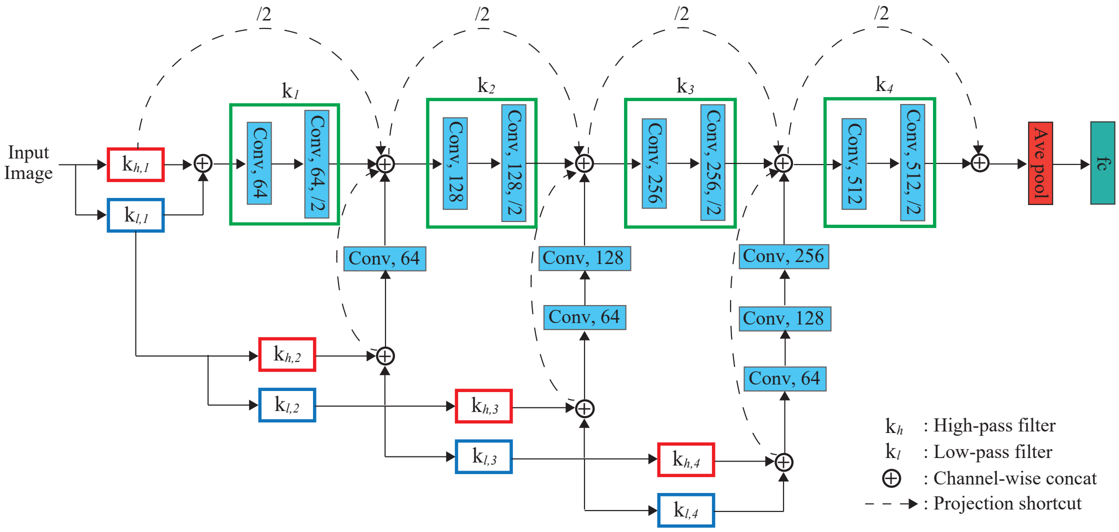


Figure 1.2: **Architecture using the DWT as a frontend to a CNN.** Figure 1 from [4]. Fujieda et. al. take a multiscale wavelet decomposition of the input before passing the input through a standard CNN. They learn convolutional layers independently on each subband and feed these back into the network at different depths, where the resolution of the subband and the network activations match.

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].

1.3 Parameterizing Filters in the Wavelet Domain

This is a simple extension of the work done by Rippel et. al. Their work also added another layer called ‘Spectral Pooling’ which effectively is a lowpass filter. As with the Fourier parameterization, a Wavelet parameterization introduces some challenges. Most notably that any filters parameterized with a decimated wavelet transform will naturally want to have a spatial support of a power of 2 whereas most convolutional filters in CNNs have an odd spatial support, typically of size 3 (and maybe more for earlier layers). This is done for a very good reason too, in that we do not want our filters to shift the activations.

[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

1.3.1 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 [7] 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 $X = \mathrm{DFT}\{x\}$. In matrix form the 2-D DFT is then:

$$\begin{aligned} X &= \mathrm{DFT}\{x\} = UxU && (1.3.1) \\ x &= \mathrm{DFT}^{-1}\{X\} = U^*YU^* && (1.3.2) \end{aligned}$$

When it comes to gradients, these become:

$$\begin{aligned} \frac{\partial L}{\partial X} &= U\frac{\partial L}{\partial x}U = \mathrm{DFT}\left\{\frac{\partial L}{\partial x}\right\} && (1.3.3) \\ \frac{\partial L}{\partial x} &= U^*\frac{\partial L}{\partial X}U^* = \mathrm{DFT}^{-1}\left\{\frac{\partial L}{\partial X}\right\} && (1.3.4) \end{aligned}$$

Now consider a single filter parameterized in the DFT and spatial domains presented with the exact same data and with the same ℓ_2 regularization ϵ and learning rate η. Let the spatial filter at time t be $\mathbf{w}_t$, the Fourier-parameterized filter be $\hat{\mathbf{w}}_t$, and let

$$\hat{\mathbf{w}}_1 = \mathrm{DFT}\{\mathbf{w}_1\} \qquad (1.3.5)$$

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:

$$\begin{aligned} \mathbf{w}_2 &= \mathbf{w}_1 - \eta\left(\frac{\partial L}{\partial \mathbf{w}} + \epsilon \mathbf{w}_1\right) && (1.3.6) \\ &= (1-\eta\epsilon)\mathbf{w}_1 - \eta\frac{\partial L}{\partial \mathbf{w}} && (1.3.7) \\ \hat{\mathbf{w}}_2 &= \hat{\mathbf{w}}_1 - \eta\left(\frac{\partial L}{\partial \hat{\mathbf{w}}} + \epsilon \hat{\mathbf{w}}_1\right) && (1.3.8) \\ &= (1-\eta\epsilon)\hat{\mathbf{w}}_1 - \eta\frac{\partial L}{\partial \hat{\mathbf{w}}} && (1.3.9) \\ & && (1.3.10) \end{aligned}$$

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 1.3.3 and 1.3.5 we then get:

$$
\begin{aligned}
\mathrm{DFT}^{-1}\left\{\hat{\mathbf{w}}_2\right\} &= \mathrm{DFT}^{-1}\left\{(1-\eta\epsilon)\hat{\mathbf{w}}_1 - \eta\,\frac{\partial L}{\partial \hat{\mathbf{w}}}\right\} && (1.3.11)\\
&= (1-\eta\epsilon)\mathbf{w}_1 - \eta\ \mathrm{DFT}^{-1}\left\{\frac{\partial L}{\partial \hat{\mathbf{w}}}\right\} && (1.3.12)\\
&= (1-\eta\epsilon)\mathbf{w}_1 - \eta\frac{\partial L}{\partial \mathbf{w}} && (1.3.13)\\
&= \mathbf{w}_2 && (1.3.14)
\end{aligned}
$$

1.3.2 Regularization

If we use ℓ_1 then the above doesn't hold.

1.3.3 Optimization

If we us adam things r different.

This does not hold for the Adam [8] 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 [7].

1.4 Taking Wavelet Transforms of Inputs

In contrast to the previous section where we only parameterized filters in the wavelet domain and transformed the filters back to the pixel domain to do convolution, this section explores learning wholly in the wavelet domain. I.e., we want to take a wavelet decomposition of the input and learn gains to apply to these coefficients, and optionally return to the pixel domain.

As neural network training involves presenting thousands of training samples on memory limited GPUs, we want our layer to be fast and as memory efficient as possible. 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. Another option is to use the DT$\mathbb{C}$WT [9]. This comes with a memory overhead which we discuss more in **??**, but it enables us to have have better directional selectivity and allows for the possibility of returning to the pixel domain with minimal aliasing [10].

In the next section we describe in more detail how the proposed layer works, agnostic of the wavelet transform used, before describing the differences between using the DWT and the DT$\mathbb{C}$WT.

1.4.1 Background

As we now want to consider the DWT and the DT$\mathbb{C}$WT which are both implemented as filter bank systems, we deviate slightly from the notation in the previous chapter (which was inspired by sampling a continuous wavelet transform).

Firstly, instead of talking about the continuous spatial variable $\mathbf{u}$, we now consider the discrete spatial variable $\mathbf{n} = [n_1, n_2]$. We switch to square brackets to make this clearer. With the new

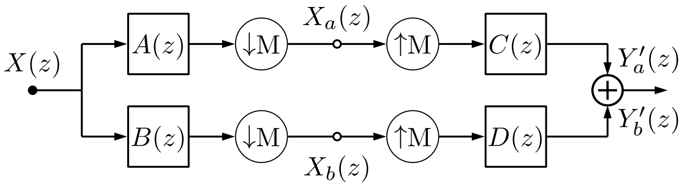


Figure 1.3: **Block Diagram of 1-D** DTℂWT**.** Note the top and bottom paths are through the wavelet or scaling functions from just level m ($M = 2^m$). Figure based on Figure 4 in [10].

discrete notation, the output of a CNN at layer l is:

$$x^{(l)}[c, \mathbf{n}], \quad c \in \{0, \dots C_l - 1\}, \mathbf{n} \in \mathbb{Z}^2 \tag{1.4.1}$$

where c indexes the channel dimension. We also make use of the 2-D Z-transform to simplify our analysis:

$$X(\mathbf{z}) = \sum_{n_1}\sum_{n_2} x[n_1, n_2] z_1^{-n_1} z_2^{-n_2} = \sum_{\mathbf{n}} x[c, \mathbf{n}]\mathbf{z}^{-\mathbf{n}} \tag{1.4.2}$$

As we are working with three dimensional arrays (two spatial and one channel) but are only doing convolution in two, we introduce a slightly modified 2-D Z-transform which includes the channel index:

$$X(c, \mathbf{z}) = \sum_{n_1}\sum_{n_2} x[c, n_1, n_2] z_1^{-n_1} z_2^{-n_2} = \sum_{\mathbf{n}} x[c, \mathbf{n}]\mathbf{z}^{-\mathbf{n}} \tag{1.4.3}$$

Recall that a typical convolutional layer in a standard CNN gets the next layer's output in a two-step process:

$$y^{(l+1)}[f, \mathbf{n}] = \sum_{c=0}^{C_l - 1} x^{(l)}[c, \mathbf{n}] * h_f^{(l)}[c, \mathbf{n}] \tag{1.4.4}$$

$$x^{(l+1)}[f, \mathbf{u}] = \sigma\left(y^{(l+1)}[f, \mathbf{u}]\right) \tag{1.4.5}$$

With the new Z-transform notation introduced in (1.4.3), we can rewrite (1.4.4) as:

$$Y^{(l+1)}(f, \mathbf{z}) = \sum_{c=0}^{C_l - 1} Z^{(l)}(c, \mathbf{z}) H_f^{(l)}(c, \mathbf{z}) \tag{1.4.6}$$

Note that we cannot rewrite (1.4.5) with Z-transforms as it is a nonlinear operation.

Also recall that with multirate systems, upsampling by M takes $X(z)$ to $X(z^M)$ and downsampling by M takes $X(z)$ to $\frac{1}{M}\sum_{k=0}^{M-1} X(W_M^k z^{1/k})$ where $W_M^k = e^{\frac{j2\pi k}{M}}$. We will drop the M subscript below unless it is unclear of the sample rate change, simply using W^k.

1.5 DTℂWT Subband Gains

Let us consider one subband of the DTℂWT. This includes the coefficients from both tree A and tree B. For simplicity in this analysis we will consider the 1-D DTℂWT without the channel parameter c. If we only keep coefficients from a given subband and set all the others to zero, then we have a reduced tree as shown in Figure 1.3. The end to end transfer function is:

$$\frac{Y(z)}{X(z)} = \frac{1}{M}\sum_{k=0}^{M-1}\left[A(W^k z)C(z) + B(W^k z)D(z)\right] \tag{1.5.1}$$

where the aliasing terms are formed from the addition of the rotated z transforms, i.e. when $k \neq 0$.

Theorem 1.1. *Suppose we have complex filters $P(z)$ and $Q(z)$ with support only in the positive half of the frequency space. If $A(z) = 2\mathrm{Re}(P(z))$, $B(z) = 2\mathrm{Im}(P(z))$, $C(z) = 2\mathrm{Re}(Q(z))$ and $D(z) = -2\mathrm{Im}(Q(z))$, then the aliasing terms in (1.5.1) are nearly zero and the system is nearly shift invariant.*

Proof. See section 4 of [10] for the full proof of this, and section 7 for the bounds on what 'nearly' shift invariant means. In short, from the definition of A, B, C and D it follows that:

$$\begin{aligned} A(z) &= P(z) + P^*(z) \\ B(z) &= -j(P(z) - P^*(z)) \\ C(z) &= Q(z) + Q^*(z) \\ D(z) &= j(Q(z) - Q^*(z)) \end{aligned}$$

where $H^*(z) = \sum_n h^*[n]z^{-n}$ is the Z-transform of the complex conjugate of the complex filter h. This reflects the purely positive frequency support of $P(z)$ to a purely negative one. Substituting these into (1.5.1) gives:

$$A(W^k z)C(z) + B(W^k z)D(z) = 2P(W^k z)Q(z) + 2P^*(W^k z)Q^*(z) \tag{1.5.2}$$

Using (1.5.2), Kingsbury shows that it is easier to design single side band filters so $P(W^k z)$ does not overlap with $Q(z)$ and $P^*(W^k z)$ does not overlap with $Q^*(z)$ for $k \neq 0$. □

Using Theorem 1.1 (1.5.1) reduces to:

$$\frac{Y(z)}{X(z)} = \frac{1}{M}\left[A(z)C(z) + B(z)D(z)\right] \tag{1.5.3}$$

Let us extend this idea to allow for any linear gain applied to the passbands (not just zeros and ones). Ultimately, we may want to allow for nonlinear operations applied to the wavelet coefficients, but we initially restrict ourselves to linear gains so that we can build from a sensible base. In particular, if we want to have gains applied to the wavelet coefficients, it would be nice to maintain the shift invariant properties of the DTℂWT.

Figure 1.4 shows a block diagram of the extension of the above to general gains. This is a two port network with four individual transfer functions. Let the transfer fucntion from X_i to Y_j be G_{ij} for $i, j \in \{a, b\}$. Then Y_a and Y_b are:

$$\begin{aligned} Y_a(z) &= X_a(z)G_{aa}(z) + X_b(z)G_{ba}(z) && (1.5.4) \\ &= \frac{1}{M}\sum_k X(W^k z^{1/k})\left[A(W^k z^{1/k})G_{aa}(z) + B(W^k z^{1/k})G_{ba}(z)\right] && (1.5.5) \\ Y_b(z) &= X_a(z)G_{ab}(z) + X_b(z)G_{bb}(z) && (1.5.6) \\ &= \frac{1}{M}\sum_k X(W^k z^{1/k})\left[A(W^k z^{1/k})G_{ab}(z) + B(W^k z^{1/k})G_{bb}(z)\right] && (1.5.7) \end{aligned}$$

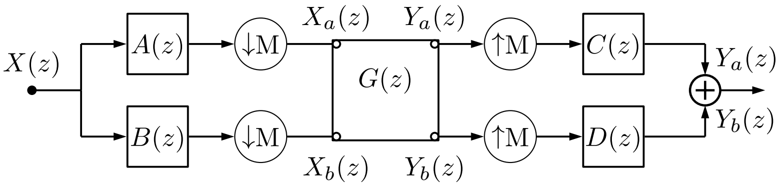


Figure 1.4: **Block Diagram of 1-D** DTℂWT**.** Note the top and bottom paths are through the wavelet or scaling functions from just level m ($M = 2^m$). Figure based on Figure 4 in [10].

Further, Y_a' and Y_b' are:

$$\begin{aligned} Y_a'(z) &= C(z)Y_a(z^M) && (1.5.8) \\ Y_b'(z) &= D(z)Y_b(z^M) && (1.5.9) \end{aligned}$$

Then the end to end transfer function is:

$$Y(z) = Y_a'(z) + Y_b'(z) = \frac{1}{M}\sum_{k=0}^{M-1} X(W^k z)\Big[A(W^k z)C(z)G_{aa}(z^k) + B(W^k z)D(z)G_{bb}(z) + \\ B(W^k z)C(z)G_{ba}(z^k) + A(W^k z)D(z)G_{ba}(z)\Big] \quad (1.5.10)$$

Theorem 1.2. *If we let $G_{aa}(z^k) = G_{bb}(z^k) = G_r(z^k)$ and $G_{ab}(z^k) = -G_{ba}(z^k) = G_i(z^k)$ then the end to end transfer function is shift invariant.*

Proof. Using the above substitutions, the terms in the square brackets of (1.5.10) become:

$$G_r(z^k)\Big[A(W^k z)C(z) + B(W^k z)D(z)\Big] + G_i(z^k)\Big[A(W^k z)D(z) - B(W^k z)C(z)\Big] \quad (1.5.11)$$

Theorem 1.1 already showed that the G_r terms are shift invariant and reduce to $A(z)C(z) + B(z)D(z)$. To prove the same for the G_i terms, we follow the same procedure. Using our definitions of A, B, C, D from Theorem 1.1 we note that:

$$\begin{aligned} A(W^k z)D(z) - B(W^k z)C(z) &= j\Big[P(W^k z) + P^*(W^k z)\Big][Q(z) - Q^*(z)] + && (1.5.12) \\ & \quad j\Big[P(W^k z) - P^*(W^k z)\Big][Q(z) + Q^*(z)] && (1.5.13) \\ &= 2j\Big[P(W^k z)Q(z) - P^*(W^k z)Q^*(z)\Big] && (1.5.14) \end{aligned}$$

We note that the difference between the G_r and G_i terms is just in the sign of the negative frequency parts, $AD - BC$ is the Hilbert pair of $AC + BD$. To prove shift invariance for the G_r terms in Theorem 1.1, we ensured that $P(W^k z)Q(z) \approx 0$ and $P^*(W^k z)Q^*(z) \approx 0$ for $k \neq 0$. We can use this again here to prove the shift invariance of the G_i terms in (1.5.11). This completes our proof. □

Using Theorem 1.2, the end to end transfer function with the gains is now

$$\frac{Y(z)}{X(z)} = \frac{2}{M}X(z)\Big[G_r(z^M)\,(A(z)C(z) + B(z)D(z)) + G_i(z^M)\,(A(z)D(z) - B(z)C(z))\Big] \quad (1.5.15)$$

Now we know can assume that our DTℂWT 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 **??** 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 [10]):

$$\frac{Y(z)}{X(z)} = \frac{2}{M}\left(P(z)Q(z) + P^*(z)Q^*(z)\right) \tag{1.5.16}$$

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 DTℂWT have spectra shown in **??**, $P^*(z)Q^*(z)$ make up the missing half of the frequency space.

1.5.1 Forward propagation

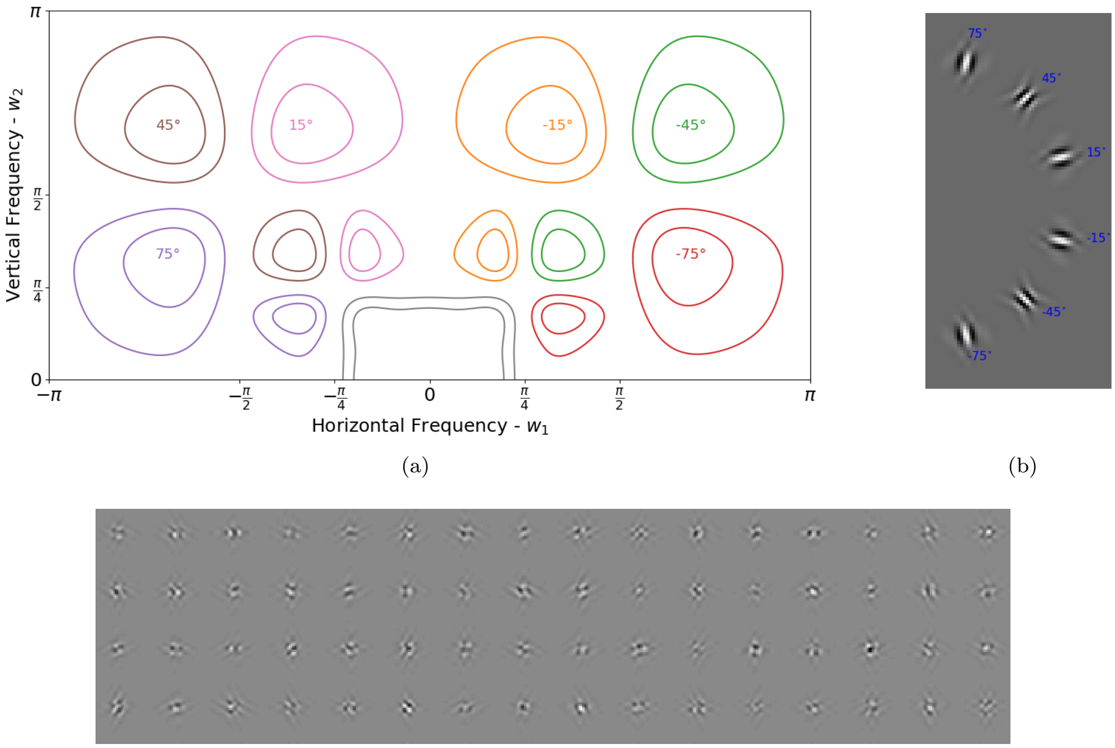
Figure 1.6 shows the block diagram using Z-transforms for a single band of our system (it is based on Figure 4 in [10]). To keep things simple for the rest of **??** the figure shown is for a 1-D system; it is relatively straightforward to extend this to 2-D[9]. The complex analysis filter (taking us 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 [10]):

$$\frac{Y(z)}{X(z)} = \frac{2}{M}\left(P(z)Q(z) + P^*(z)Q^*(z)\right) \tag{1.5.17}$$

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 DTℂWT 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:

$$\frac{Y(z)}{X(z)} = \frac{2}{M}\left[\, G_r(z^M)\left(P(z)Q(z) + P^*(z)Q^*(z)\right) + jG_i(z^M)\left(P(z)Q(z) - P^*(z)Q^*(z)\right)\right] \tag{1.5.18}$$



(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 DTℂWT at scales 1 and 2 and the scale 2 lowpass. These are the product $P(z)Q(z)$ from Equation 1.5.17.(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.

1.5.2 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 $Y(z) = H(z)X(z)$:

$$\Delta X(z) = H(z^{-1})\Delta Y(z) \tag{1.5.19}$$

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 1.5.19 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 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 - $V(z)$.

$$\Delta G_r(z) = \Delta W_r(z)V_r(z^{-1}) + \Delta W_i(z)V_i(z^{-1}) \tag{1.5.20}$$

$$\Delta G_i(z) = -\Delta W_r(z)V_i(z^{-1}) + \Delta W_i(z)V_r(z^{-1}) \tag{1.5.21}$$

Unsurprisingly, the passthrough gradients have similar form to Equation 1.5.18:

$$\Delta X(z) = \frac{2\Delta Y(z)}{M}\Big[G_r(z^{-M})\left(PQ + P^*Q^*\right) + jG_i(z^{-M})\left(PQ - P^*Q^*\right)\Big] \tag{1.5.22}$$

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 1.5.20,1.5.21,1.5.22 over the support of $G(z)$ i.e., if it is a single number we only need to calculate $\Delta G(z)|_{z=0}$.

1.5.3 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 1×1 to show that this gain is identical to a 1×1 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] 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.

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 5×5 pixels, for the second scale it is about 15×15.

1.5.4 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.

1.5.5 Examples

1.5c show example impulse responses of our layer. These impulses were generated by randomly initializing both the real and imaginary parts of $g_2 \in \mathbb{C}^{6\times 1\times 1}$ 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.

1.6 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 5×5 followed by 2 fully connected layers and a softmax final layer. We swap both these convolutional layers out 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 3×3 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 [8] 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.

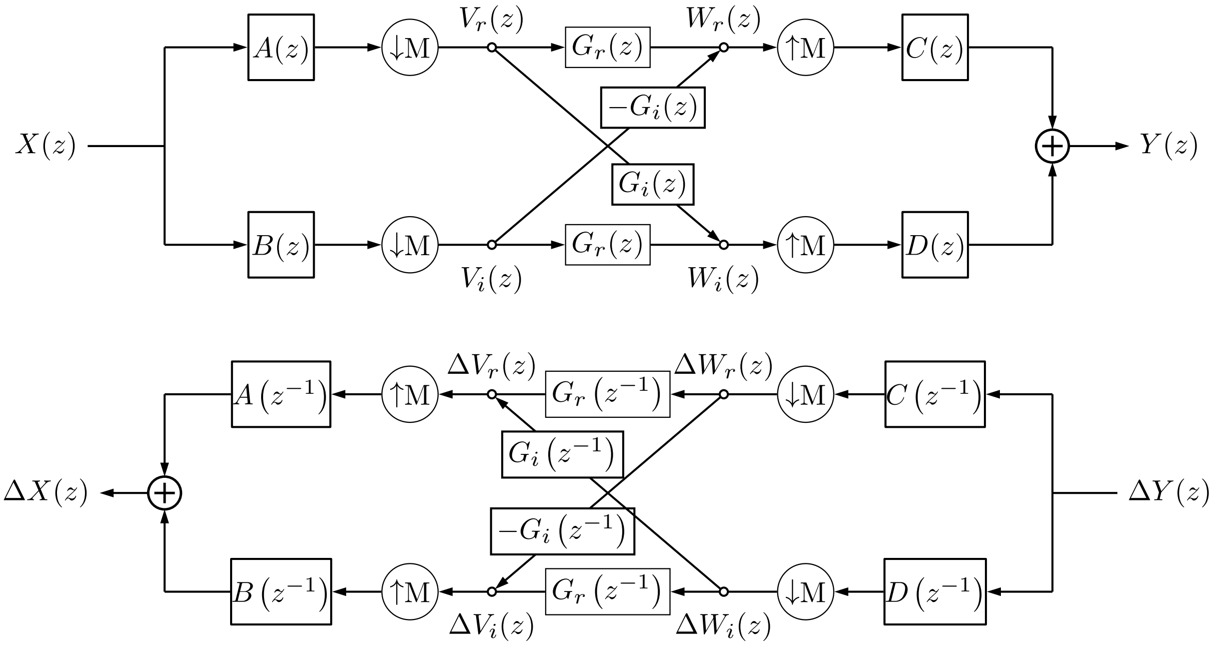


Figure 1.6: **Forward and backward block diagrams for** DTℂWT **gain layer.** Based on Figure 4 in [10]. Ignoring the G gains, the top and bottom paths (through A, C and B, D respectively) make up the the real and imaginary parts for *one subband* of the dual tree system. Combined, $A+jB$ and $C-jD$ 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 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

Algorithm 1 DTℂWT gain layer forward and backward passes

```
procedure GAINFWD($x, w_l,$)
    $yl,\ yh \leftarrow$ DTℂWT($x^l$, nlevels = 1)        ▷ yh has 6 orientations and is complex
    $U \leftarrow$ COMPLEXMAG($yh$)
    $yl \leftarrow$ AVGPOOL2x2($yl$)        ▷ Downsample and recentre lowpass to match U size
    $Z \leftarrow$ CONCATENATE($yl,\ U$)        ▷ concatenated along the channel dimension
    $Y \leftarrow AZ$        ▷ Mix
    save $Z$        ▷ For the backwards pass
    return $Y$
end procedure

procedure GAINBWD($\frac{\partial L}{\partial Y},\ A$)
    load $Z$
    $\frac{\partial L}{\partial A} \leftarrow \frac{\partial L}{\partial Y} Z^T$        ▷ The weight gradient
    $\Delta Z \leftarrow A^T \frac{\partial L}{\partial Y}$
    $\Delta yl,\ \Delta U \leftarrow$ UNSTACK($\Delta Z$)
    $\Delta yl \leftarrow$ AVGPOOL2x2BWD($\Delta yl$)
    $\Delta yh \leftarrow$ COMPLEXMAGBWD($\Delta U$)
    $\frac{\partial L}{\partial x} \leftarrow$ DTℂWTBWD($\Delta yl,\ \Delta yh$)        ▷ The propagated gradient
    return $\frac{\partial L}{\partial x},\ \frac{\partial L}{\partial A}$
end procedure
```

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.

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

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

[4] ——, “Wavelet Convolutional Neural Networks”, *arXiv:1805.08620 [cs]*, May 2018. arXiv: 1805.08620 [cs].

[5] 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.

[6] 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].

[7] 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.

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

[9] 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.

[10] 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.

[11] ——, “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.