0.1 Complex Convolution and Gradients

Consider a complex number $z = x + iy$, and the complex mapping

$$w = f(z) = u(x,y) + iv(x,y)$$

where u and v are called 'conjugate functions'. Let us examine the properties of $f(z)$ and its gradient.

The definition of gradient for complex numbers is:

$$\lim_{\Delta z \to 0} \frac{f(z+\Delta z) - f(z)}{\Delta z}$$

A necessary condition for $f(z,\bar{z})$ to be an analytic function is $\frac{\partial f}{\partial \bar{z}} = 0$. I.e. f must be purely a function of z, and not $\bar{z}$.

A geometric interpretation of complex gradient is shown in Figure 1. As Δz shrinks to 0, what does Δw converge to? E.g. consider the gradient of approach $m = \frac{dy}{dx} = \tan\theta$, then the derivative is

$$\gamma = \alpha + i\beta = D(x,y) + P(x,y)e^{-2i\theta}$$

where

$$\begin{aligned} D(x,y) &= \frac{1}{2}(u_x + v_y + i(v_x - u_y)) \\ P(x,y) &= \frac{1}{2}(u_x - v_y + i(v_x + u_y)) \end{aligned}$$

$P(x,y) = \frac{dw}{d\bar{z}}$ needs to be 0 for the function to be analytic. This is where we get the Cauchy-Riemann equations:

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}$$

and

$$\frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}$$

The function $f(z)$ is analytic (or regular or holomorphic) if the derivative $f'(z)$ exists at all points z in a region R. If R is the entire z-plane, then f is entire.

Grad Operator

Recall, the gradient is a multi-variable generalization of the derivative. The gradient is a vector valued function. In the case of complex numbers, it can be represented as a complex

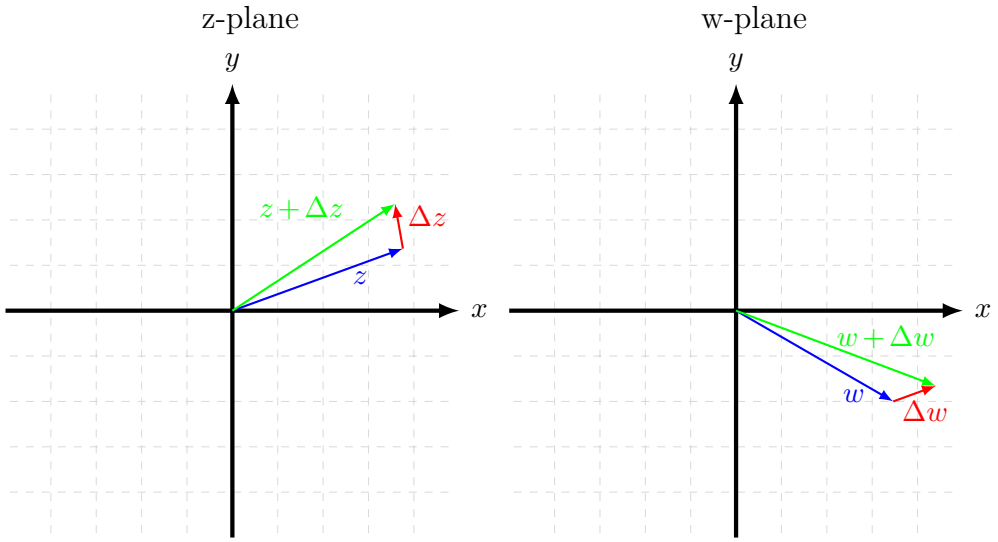


Figure 1: **Geometric interpretation of complex gradient.** The gradient is defined as $f'(z) = \lim_{\Delta z \to 0} \frac{\Delta w}{\Delta z}$. It must approach the same value independent of the direction Δz approaches zero. This turns out to be a very strong and somewhat restrictive property.

number too. E.g. consider $W(z) = F(x,y)$ (note that in general it may be simple to find F given G, but they are different functions).

I.e.

$$\nabla F = \frac{\partial F}{\partial x} + i\frac{\partial F}{\partial y}$$

Consider the case when F is purely real, then $F(x,y) = F(\frac{z+\bar{z}}{2}, \frac{z-\bar{z}}{2i}) = G(z,\bar{z})$ Then

$$\nabla F = \frac{\partial F}{\partial x} + i\frac{\partial F}{\partial y} = 2\frac{\partial G}{\partial \bar{z}}$$

If F is complex, let $F(x,y) = P(x,y) + iQ(x,y) = G(z,\bar{z})$, then

$$\nabla F = \left(\frac{\partial}{\partial x} + i\frac{\partial}{\partial y}\right)(P+iQ) = \left(\frac{\partial P}{\partial x} - \frac{\partial Q}{\partial x}\right) + i\left(\frac{\partial P}{\partial y} + \frac{\partial Q}{\partial x}\right) = 2\frac{\partial G}{\partial \bar{z}}$$

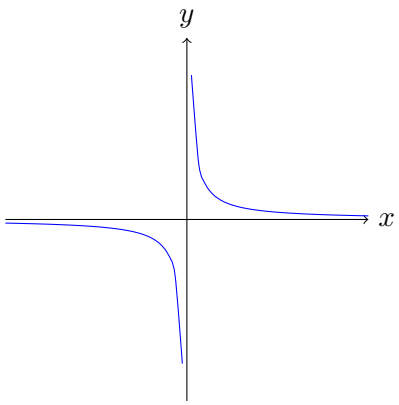
It is clear to see how the purely real case is a subset of this (set Q=0 and all its partials will be 0 too).

If G is an analytic function, then $\frac{\partial G}{\partial \bar{z}} = 0$ and so the gradient is 0, and the Cauchy-Riemann equations hold $\frac{\partial P}{\partial x} = \frac{\partial Q}{\partial y}$ and $\frac{\partial P}{\partial y} = -\frac{\partial Q}{\partial x}$

Hilbert Pairs of General Functions

How does this affect me? I don't think I'll be able to use analytic non-linearities, however I may be able to have analytic filters, like those of the DTCWT.

The Hilbert pair of the cosine is the sine function, but what about in general? If $x = \delta(t)$, its Hilbert pair $jy = \frac{-j}{\pi t}$. Like the dirac delta function, this also has a flat spectrum, and the figure for it is shown below.



That means if we wanted to get the Hilbert pair of a sampled signal, then we would have to add shifts and scales of y, which unfortunately has infinite support. We would also have to lowpass it, as we do for the sampled version (so their frequency spectrums are the same).

Usefulness of Complex Numbers

Nick made a good point in our recent meeting that when trying to use the complex plane, we must know/understand what it is we want to gain from the added representation. For the case of the DTCWT, he converted the non-analytic sinusoids of the wavelet transform into an analytic signal.

I.e. let us ignore the previous notation of $x = \Re(z), y = \Im(z)$ and redefine them to indicate the horizontal and vertical directions in an image.

For a real wavelet transform, all of the cosine terms are

$$\cos\omega_1 x = \frac{1}{2}\left(e^{j\omega_1 x} + e^{-j\omega_1 x}\right)$$

If we consider $z_x = e^{j\omega_1 x}$, then this is clearly a function of both z_x and its conjugate (as are all real valued functions). I.e. $\cos\omega_1 x = F(z_x, \bar{z_x})$. Nick replaced this with the analytic equivalent of this function by adding in the Hilbert pair term.

$$\cos\omega_1 x + j\sin\omega_1 x = e^{j\omega_1 x} = F(z_x)$$

From our above definitions of analytic functions, it is clear to see that this is now no longer a function of the conjugate term $\bar{z_x}$, so it is analytic. The benefit for Nick was that now he could separably multiply the x and the y sinusoids to get:

$$e^{j\omega_1 x} \times e^{j\omega_2 y} = F(z_x)F(z_y) = e^{j(\omega_1 x + \omega_2 y)} = F(z_x + z_y)$$

Also, I must be careful with regularizing complex weights. We want to set some of the weights to 0, and let the remaining ones evolve to whatever phase they please. To do this, either use the L-2 norm on the real and imaginary parts independently, or be careful about using the L-1 norm. This is because we really want to be penalising the magnitude of the complex weights, r and:

$$\|r\|_2^2 = \left\|\sqrt{x^2+y^2}\right\|_2^2 = \sum x^2+y^2 = \sum x^2 + \sum y^2 = \|x\|_2^2 + \|y\|_2^2$$

But this wouldn't necessarily be the case for the L-1 norm case.

Working with Complex weights in CNNs

As a first pass, I think I shouldn't concern myself too much with analytic functions and having the Cauchy-Riemann equations met. Instead, I will focus on implementing the CNN with a real and imaginary component to the filters, and have these stored as independent variables.

Unfortunately, most current neural network tools only work with real numbers, so we must write out the equations for the forward and backwards passes, and ensure ourselves that we can achieve the equivalent of a complex valued filter.

A review of what we have done so far

This was inspired by our recent work with creating 12 tap complex filters that worked across the 6 DTCWT orientations (and their complex conjugate). We found that having symmetric filters, with a 90° offset between taps created a nice corner like object. In fact, we could then shift these filters along to the next coefficients and the output would rotate by 30° (as we'd simply used the next set of wavelet coefficients).

Let us give a concrete example. Consider the 4 tap filter:

$$[1, j, j, 1]$$

Now consider the result of doing a forward transform on a $N \times N$ sized image. The highpass outputs (let us call them Yh) have size $2^{-k}N \times 2^{-k}N \times 6$ for each scale, k. When we put the above filter into the first 4 DTCWT coefficients at the same spatial position (x, y) at given scale, k i.e.

$$\begin{aligned} \mathrm{Yh}[\,\mathrm{k}\,][\,\mathrm{y}\,,\mathrm{x}\,,0\,] &= 1 \\ \mathrm{Yh}[\,\mathrm{k}\,][\,\mathrm{y}\,,\mathrm{x}\,,1\,] &= \mathrm{j} \\ \mathrm{Yh}[\,\mathrm{k}\,][\,\mathrm{y}\,,\mathrm{x}\,,2\,] &= \mathrm{j} \end{aligned}$$

$$\mathrm{Yh}[\,\mathrm{k}\,][\,\mathrm{y}\,,\mathrm{x}\,,3\,] \;=\; 1$$

and then do the inverse DTCWT we get the result shown in Figure 2. Doing this is equivalent to the deconvolution of Zeiler and Fergus. What we're really seeing in Figure 2 is the input shape that would give the highest response if we applied a DTCWTto it, and then applied the above filter across the orientations.

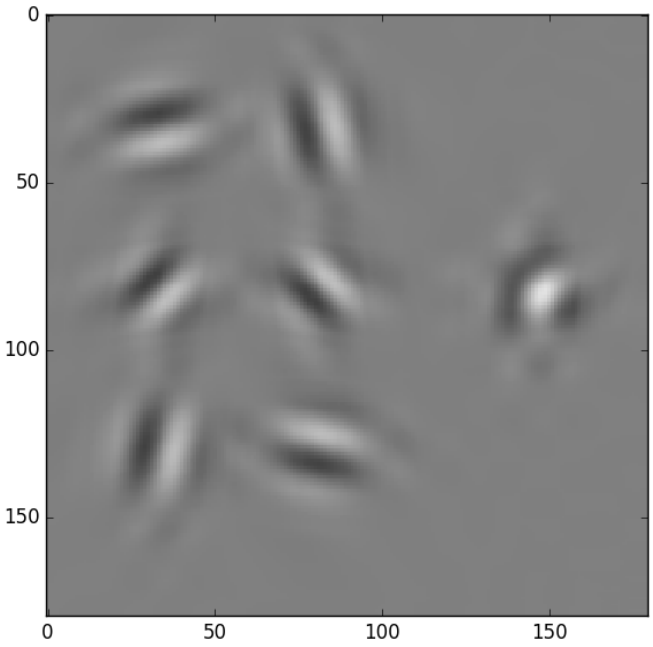


Figure 2: Left of image shows the imaginary part of 6 DTCWT impulse responses. Right of image shows the effect of combining with the filter $[1, j, j, 1]$

If we were now to expand our definition of the above filter to be:

$$\mathbf{f} = [1, j, j, 1, 0, 0, 0, 0, 0, 0, 0, 0]$$

and denote the circular shift of f by k poistions to be f_k, then the output from shifting f through all 12 positions is shown in Figure 3. In particular, it is the first shape rotated by 30° for each k.

Forward pass

Convolution

In the above example **f**has a spatial support of only 1×1. We still were able to get a somewhat complex shape by shifting the relative phases of the complex coefficients, but we

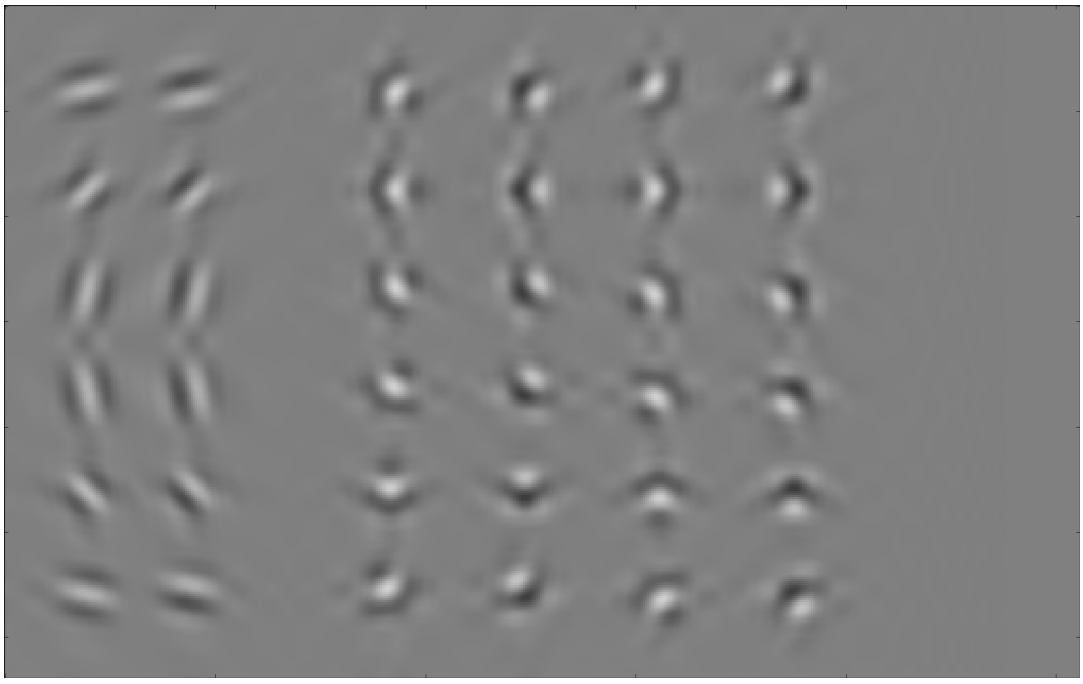


Figure 3: Left shows the 6 real and imaginary DTCWT impulse responses. Right of image shows the resulting 12 corners achieved from rotating $[1, j, j, 1]$ through the 12 of these. The even columns are the 'Hilbert pair' of the odd columns, obtained by shifting all the coefficients by 90° (i.e. $[j, -1, -1, j]$).

are inherently limited (as we can only rotate the coefficient by at most 2π). So in general, we want to be able to consider the family of filters $\mathbf{f} \in \mathbb{C}^{m_1 \times m_2 \times C}$. For ease, let us consider only square filters of spatial support m, so $\mathbf{f} \in \mathbb{C}^{m \times m \times C}$. Note that we have restricted the third dimension of our filter to be $C = 12$ in this case. This means that convolution is only in the spatial domain, rather than across channels. Ultimately we would like to be able to handle the more general case of allowing the filter to rotate through channels, but we will tackle the simpler problem first[1]

Let us represent the complex input with z, which is of shape $\mathrm{C}^{n_1 \times n_2 \times C}.We call w the result we get from convolvi$ $\mathbb{C}^{n_1+m-1, n_2+m-1, 1}$. With appropriate zero or reflective padding, we can make $w$have the same spatial shape as $\mathbf{z}$. Now, consider the full complex convolution to get w:

$$w[l_1, l_2] = \sum_{c=0}^{C-1} \sum_{k_1, k_2} f[k_1, k_2, c] z[l_1 - k_1, l_2 - k_2, c]$$

[1]Recall from Figure 3, the benefit of allowing a filter to rotate through the channel dimension was we could easily obtain 30° shifts of the sensitive shape.

Let us define

$$\begin{aligned} z &= z_R + jz_I \\ w &= w_R + jw_I \\ f &= f_R + jf_I \end{aligned}$$

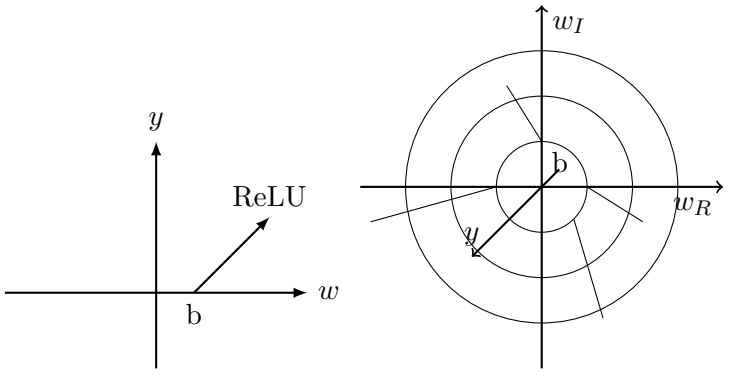
where all of these belong to the real space of the same dimension as their parent. Then

$$\begin{aligned} w[l_1,l_2] &= w_R + jw_I \\ &= \sum_{c=0}^{C-1} \sum_{k_1,k_2} f[k_1,k_2,c] z[l_1-k_1, l_2-k_2, c] \\ &= \sum_{c=0}^{C-1} \sum_{k_1,k_2} (f_R[k_1,k_2,c] + jf_I[k_1,k_2,c])(z_R[l_1-k_1,l_2-k_2,c] + jz_I[l_1-k_1,l_2-k_2,c]) \\ &= \sum_{c=0}^{C-1} \sum_{k_1,k_2} (z_R[l_1-k_1,l_2-k_2,c] f_R[k_1,k_2,c] - z_I[l_1-k_1,l_2-k_2,c] f_I[k_1,k_2,c]) \\ &+ j \sum_{c=0}^{C-1} \sum_{k_1,k_2} (z_R[l_1-k_1,l_2-k_2,c] f_I[k_1,k_2,c] + z_I[l_1-k_1,l_2-k_2,c] f_R[k_1,k_2,c]) \\ &= ((z_R * f_R) - (z_I * f_I))[l_1,l_2] + ((z_R * f_I) + (z_I * f_R))[l_1,l_2] \end{aligned}$$

Unsurprisingly, complex convolution is then the sum and difference of 4 real convolutions.

Non-Linearity

70105



As a first attempt at a non-linearity, we will try a modified L2-norm. In particular, for input w(coming from the convolutional layer), the output $y$is:

$$y = g(w,b) = \sqrt{\max(0, w_R^2 + w_I^2 - b^2)}$$

This is an attempt to modify the ReLU nonlinearity by rotating it around the complex plane. Really this is more like the complex generalization of the function $y = \max(0, |x| - b)$, as the ReLU rejects half of the real line, whereas this function only rejects small magnitude values. I justify this like so:

- If a patch of pixels strongly correlates with a filter[2], it will produce a large output.
- A large output will be caught by the ReLU and 'passed' through, with a small subtraction penalty, b.
- If a patch of pixels does not correlate with a filter, it produces a small output, which gets clipped to 0 by the ReLU.
- If a patch of pixels negatively correlates with a filter, say if we multiply a strongly correlated input with -1, then **this should be distinguishable from zero correlation**. A way to do this would be use a soft limiting function ($y = \min(0, x + b) + \max(0, x - b)$), or the rectified version of the soft limiter ($y = \max(0, |x| - b)$.
- The only downside to these non-linearities is that they theoretically 'fire' on much more of the input, which could mean a dramatic change in network behaviour, but really it shouldn't be so drastically different. We can verify this by measuring the probability distribution at the input to non-linearities in a real valued CNN. It is easy to show that they are roughly normally distributed.

Backwards Pass

Now we need to calculate what the complex gradients will be, and importantly, how to implement them using only real representations.

A typical loss function is:

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{\text{data}} + \mathcal{L}_{\text{reg.}} \\ &= \frac{1}{N} \sum_{n=0}^{N-1} \sum_{k} y_{nk} \log \pi_{nk} + \lambda \sum_{l} \theta_l^2 \end{aligned}$$

The derivative of the regularizer loss w.r.t. the parameters is trivial to find, so let us only focus on the data loss for now.

[2] really this should read 'strongly correlates with the mirror image of a filter', but the meaning is the same