Chapter 1

Visualizing and Improving Scattering Networks

Deep Convolutional Neural Networks have become the *de facto* solution for image understanding tasks since the rapid development in their growth in recent years. Since AlexNet**??** in 2012, there have been many new improvements in their design, taking them 'deeper', such as the VGG network in 2014 [1], the Inception Network [2], Residual Networks in 2016 [3] and DenseNets in [4]. Each iteration has abstracted the inner workings of the model more and more.

Despite their success, they are often criticized for being 'black box' methods. Indeed, once you train a CNN, you can view the first layer of filters quite easily (see **??**) as they exist in RGB space. Beyond that things get trickier, as the filters have a third, 'depth' dimension typically much larger than its two spatial dimensions, and representing these dimensions with different colours becomes tricky and uninformative.

This has started to become a problem, and while we are happy to trust modern CNNs for isolated tasks, we are less likely to be comfortable with them driving cars through crowded cities, or making executive decisions that affect people directly. A commonly used contrived example, it is not hard to imagine a deep network that could be used to assess whether giving a bank loan to an applicant is a safe investment. Trusting a black box solution is deeply unsatisfactory in this situation. Not only from the customer's perspective, who, if declined, has the right to know why goodman_european_2016, but also from the bank's — before lending large sums of money, most banks would like to know why the network has given the all clear. 'It has worked well before' is a poor rule to live by.

A recent paper titled 'Why Should I Trust You?' [5] explored this concept in depth. They rated how highly humans trusted machine learning models. Unsurprisingly, a model with an interpretable methodology was trusted more than those which did not have one, even if it had a lower prediction accuracy on the test set. To build trust and to aid training, we need to probe these networks and visualize how and why they are making decisions.

Some good work has been done in this area. In particular, Zeiler and Fergus [6] design a DeConvNet to visualize what a filter in a given layer is mostly highly activated by. [7] learn to invert representations by updating a noisy input until its latent feature vector matches a desired target. [8] develop saliency maps by projecting gradients back to the input space and measuring where they have largest magnitude.

In recent years, ScatterNets have been shown to perform very well as image classifiers and have received a fair share of attention themselves. They have been one of the biggest developments and successes in applying wavelets in deep learning systems, and are particularly inspiring due to their well defined properties. They are typically used as unsupervised feature extractors [9]–[12] and can outperform CNNs for classification tasks with reduced training set sizes, e.g. in CIFAR-10 and CIFAR-100 (Table 6 from [13] and Table 4 from [11]). They are also near state-of-the-art for Texture Discrimination tasks (Tables 1–3 from [14]). Despite this, there still exists a considerable gap between them and CNNs on challenges like CIFAR-10 with the full training set (83% vs. 93%). Even considering the benefits of ScatterNets, this gap must be addressed.

While ScatterNets have good theoretical foundation and properties [15], it is difficult to understand the second order scattering. In particular, how useful are the second order coefficients for training? How similar are the scattered features to a modern state of the art convolutional network? To answer these questions, this chapter interrogates ScatterNet frontends. Taking inspiration from the interrogative work applied to CNNs, we build a DeScatterNet to visualize what the second order features are. We also heuristically probe a trained Hybrid Network and quantify the importance of the individual features.

We first revise the operations that form a ScatterNet in section 1.2. We then introduce our DeScatterNet (section 1.3), and show how we can use it to examine the layers of ScatterNets (using a similar technique to the CNN visualization in [6]). We use this analysis tool to highlight what patterns a ScatterNet is sensitive to (section 1.4), showing that they are very different from what their CNN counterparts are sensitive to, and possibly less useful for discriminative tasks.

We then measure the ScatterNet channel saliency by peforming an occlusion test on a trained hybrid network ScatterNet-CNN, iteratively switching off individual Scattering channels and measuring the effect this has on the validation accuracy in section 1.5. The results from the occlusion tests strengthen the idea that some of the ScatterNet patterns may not be well suited for deep learning systems.

We use these observations to propose an architectural change to ScatterNets, which have not changed much since their inception in [15], and show that it is possible to get visually more appealing shapes by filtering across the orientations of the ScatterNet. We present this in section 1.6.

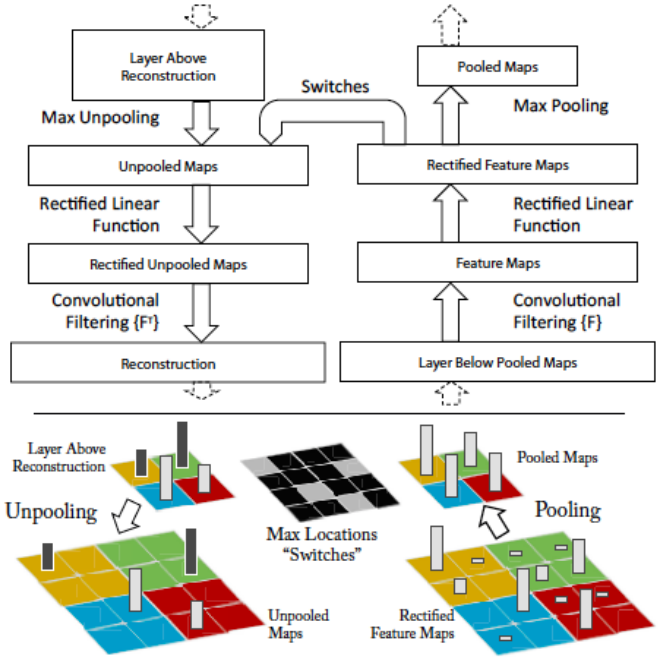


Figure 1.1: **Deconvolution Network Block Diagram.** Note the switches that are saved before the pooled features, and the filters used for deconvolution are the transpose of the filters used for a forward pass. Taken from [6].

1.1 Related Work

Zeiler, Taylor, and Fergus first attempted to use 'deconvolution' to improve their learning [16], then later for purely visualization purposes [6]. Their method involves monitoring network nodes, seeing what input image causes the largest activity and mapping activations at different layers of the network back to the pixel space using meta-information from these images.

Figure 1.1 shows the block diagram for how deconvolution is done. Inverting a convolutional layer is done by taking the 2D transpose of each slice of the filter. Inverting a ReLU is done by simply applying a ReLU again (ensuring only positive values can flow back through the network). Inverting a max pooling step is a little trickier, as max pooling is quite a lossy operation. Zeiler, Taylor, and Fergus get around this by saving extra information on the forward pass of the model — switches that store the location of the input that caused the maximum value. This way, on the backwards pass, it is trivial to store activations to the right position in the larger feature map. Note that the positions that did not contribute to the max pooling operation remain as zero on the backwards pass. This is shown in the bottom half of Figure 1.1.

Mahendran and Vedaldi take a slightly different route on deconvolution networks [7]. They do not store this extra information but instead define a cost function to maximize. This

results in visualization images that look very surreal, and can be quite different from the input.

[17] design an *all convolutional* model, where they replace the max pooling layers commonly seen in CNNs with a convolutional block with sride 2, i.e. a convolution followed by decimation. They did this by first adding an extra convolutional layer before the max pooling layers, then taking away the max pooling and adding decimation after convolution, noting that removing the max pooling had little effect.

The benefit of this is that they can now reconstruct images as Zeiler and Fergus did, but without having to save switches from the max pooling operation. Additionally, they modify the handling of the ReLU in the backwards pass to combine the regular backpropagation action and the ReLU action from deconv. They call this 'guided backprop'.

Another interrogation tool commonly used is occlusion or perturbation. [6] occlude regions in the input image and measure the impact this has on classification score. [18] use gradients to find the minimal mask to apply to the input image that causes misclassification.

1.2 The Scattering Transform

While we have introduced the scattering trasnform before, we clarify the format we use for this chapter's analysis.

We use the DT$\mathbb{C}$WT based scatternet introduced in the previous chapter **??** as a front end with $J = M = 2$. Consider a single channel input signal $x(\mathbf{u})$, $\mathbf{u} \in \mathbb{R}^2$.

The zeroth order **scatter** coefficient is the lowpass output of a J level FB:

$$S_0 x(\mathbf{u}) \triangleq x(\mathbf{u}) * \phi_J(\mathbf{u}) \tag{1.2.1}$$

This is invariant to translations of up to 2^J pixels[1]. In exchange for gaining invariance, the S_0 coefficients have lost information (contained in the rest of the frequency space). The remaining energy of x is contained within the first order **wavelet** coefficients:

$$W_1 x(\mathbf{u}, j_1, \theta_1) \triangleq x * \psi_{j_1,\theta_1} \tag{1.2.2}$$

for $j_1 \in \{1,2\}, \theta_1 \in \{15°, 45° \ldots, 165°\}$ (we may sometimes index θ with $1 \leq k \leq K$). We will want to retain this information in these coefficients to build a useful classifier.

Let us call the set of available scales and orientations Λ_1 and use λ_1 to index it. For both Morlet and DT$\mathbb{C}$WT implementations, ψ is complex-valued, i.e., $\psi = \psi^r + j\psi^i$ with ψ_r and ψ_i forming a Hilbert Pair, resulting in an analytic ψ. This analyticity provides a source of

[1] From here on, we drop the $\mathbf{u}$ notation when indexing x, for clarity.

invariance — small input shifts in x result in a phase rotation (but little magnitude change) of the complex wavelet coefficients[2].

Taking the magnitude of W_1 gives us the first order propagated signals:

$$U_1 x(\lambda_1, \mathbf{u}) \triangleq |x * \psi_{\lambda_1}| = \sqrt{(x * \psi^r_{\lambda_1})^2 + (x * \psi^i_{\lambda_1})^2} \tag{1.2.3}$$

The first order scattering coefficient makes U_1 invariant up to our scale J by averaging it:

$$S_1 x(\lambda_1, \mathbf{u}) \triangleq |x * \psi_{\lambda_1}| * \phi_J \tag{1.2.4}$$

This has $KJ = 6 \times 2 = 12$ output channels for each input channel. Later in this chapter we will want to distinguish between the first and second scale coefficients of the S_1 terms, which we will do by moving the j index to a superscript. I.e., S_1^1 and S_1^2 refer to the set of 6 S_1 terms at the first and second scales.

Higher order scattering coefficients recover the information lost by averaging U_1, and are defined as:

$$W_m = U_{m-1} * \psi_{\lambda_m} \tag{1.2.5}$$

$$U_m = |W_m| \tag{1.2.6}$$

$$S_m = U_m * \phi_J \tag{1.2.7}$$

Previous work shows that for natural images we get diminishing returns after $m = 2$. The second order scattering coefficients are defined only on paths of increasing energy[9] as:

$$S_2 x(\lambda_1, \lambda_2, \mathbf{u}) \triangleq ||x * \psi_{\lambda_1}| * \psi_{\lambda_2}| * \phi_J \tag{1.2.8}$$

As we only go on the paths of increasing energy $j_1 = 1$, $j_2 = 2$. This then has $6 \times 6 = 36$ output channels per input channel.

Our output is then a stack of these 3 outputs:

$$Sx = \{S_0 x, S_1 x, S_2 x\} \tag{1.2.9}$$

with $1 + 12 + 36 = 49$ channels per input channel.

1.2.1 Scattering Colour Images

A wavelet transform like the DT$\mathbb{C}$WT accepts single channel input, while we often work on RGB images. This leaves us with a choice. We can either:

[2] In comparison to a system with purely real filters such as a CNN, which would have rapidly varying coefficients for small input shifts [19].

1. Apply the wavelet transform (and the subsequent scattering operations) on each channel independently. This would triple the output size to $3C$.

2. Define a frequency threshold below which we keep colour information, and above which, we combine the three channels.

The second option uses the well known fact that the human eye is far less sensitive to higher spatial frequencies in colour channels than in luminance channels. This also fits in with the first layer filters seen in the well known Convolutional Neural Network, AlexNet. Roughly one half of the filters were low frequency colour 'blobs', while the other half were higher frequency, greyscale, oriented wavelets.

For this reason, we choose the second option for the architecture described in this chapter. We keep the 3 colour channels in our S_0 coefficients, but work only on greyscale for high orders (the S_0 coefficients are the lowpass bands of a J-scale wavelet transform, so we have effectively chosen a colour cut-off frequency of $2^{-J}\frac{f_s}{2}$).

We combine the three channels by modifying our magnitude operation from (**??**) to now be:

$$r_s = \sqrt{x_r^2 + y_r^2 + x_g^2 + y_g^2 + x_b^2 + y_b^2 + b^2} - b \tag{1.2.10}$$

Where x_r, x_g, x_b are the real parts of the wavelet response for the red, green and blue channels, and y is the corresponding imaginary part. This only affects the S_1 coefficients, and the S_2 coefficients then are calculated as normal.

An alternative to (1.2.10) is to simply combine the colours before scattering into a luminance channel. However we choose to use (1.2.10) instead as this has the ability to detect colour edges with constant luminance.

The resulting scattering output now has $3 + 12 + 36 = 51$ channels at $1/16$ the spatial input size.

1.3 The Inverse Scatter Network

We now introduce our inverse scattering network. This allows us to back-project scattering coefficients to the image space; it is inspired by the DeconvNet used by Zeiler and Fergus in [6] to look into the deeper layers of CNNs. Like the DeConvNet, the inverse Scattering Network is similar to backpropagating a single strong activation (rather than usual gradient terms).

We emphasize that instead of thinking about perfectly reconstructing x from $S \in \mathbb{R}^{C \times H' \times W'}$, we want to see what signal/pattern in the input image caused a large activation in each channel. This gives us a good idea of what each output channel is sensitive to, or what it extracts from the input. Note that we do not use any of the log normalization layers described in [10], [11].

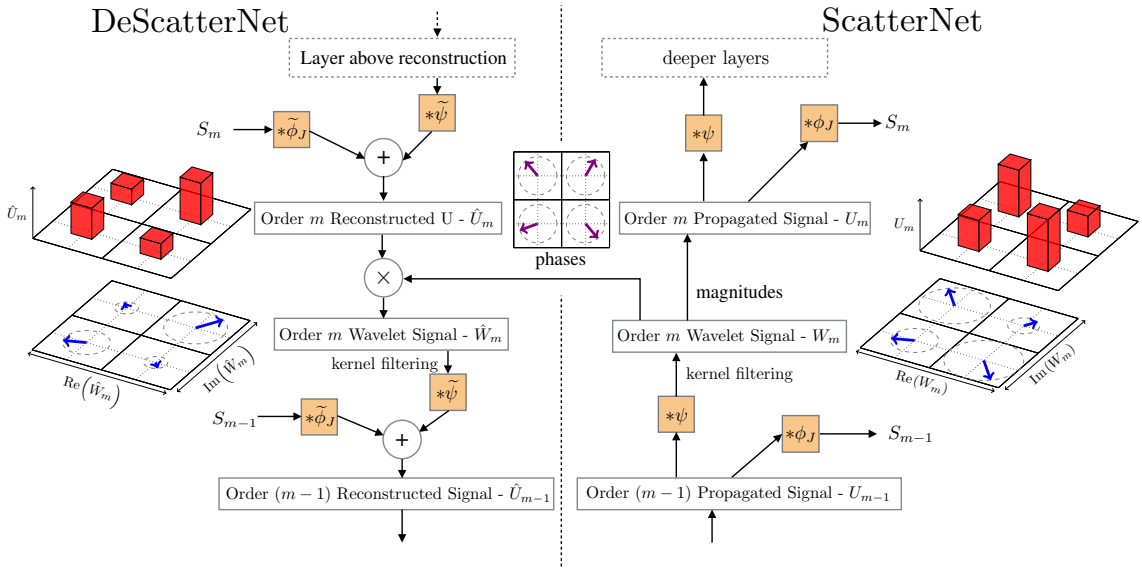


Figure 1.2: **The Descattering Network.** Comprised of a DeScattering layer (left) attached to a Scattering layer (right). We are using the same convention as [6] Figure 1 - i.e. the input signal starts in the bottom right hand corner, passes forwards through the ScatterNet (up the right half), and then is reconstructed in the DeScatterNet (downwards on the left half). The DeScattering layer will reconstruct an approximate version of the previous order's propagated signal. The 2×2 grids shown around the image are either Argand diagrams representing the magnitude and phase of small regions of *complex* (De)ScatterNet coefficients, or bar charts showing the magnitude of the *real* (De)ScatterNet coefficients (after applying the modulus non-linearity). For reconstruction, we need to save the discarded phase information and reintroduce it by multiplying it with the reconstructed magnitudes.

1.3.1 Inverting the Low-Pass Filtering

Going from the U coefficients to the S coefficients involved convolving by a low pass filter, ϕ_J followed by decimation to make the output $(H \times 2^{-J}) \times (W \times 2^{-J})$. ϕ_J is a purely real filter, and we can 'invert' this operation by interpolating S to the same spatial size as U and convolving with the mirror image of ϕ_J, $\widetilde{\phi}_J$ (this is equivalent to the transpose convolution described in [6]).

$$\hat{S}_m = S_m * \widetilde{\phi}_J \tag{1.3.1}$$

This will not recover U as it was on the forward pass, but will recover all the information in U that caused a strong response in S.

1.3.2 Inverting the Magnitude Operation

In the same vein as [6], we face a difficult task in inverting the non-linearity in our system. We lend inspiration from the switches introduced in the DeconvNet; the switches in a DeconvNet save the location of maximal activations so that on the backwards pass activation layers could

be unpooled trivially. We do an equivalent operation by saving the phase of the complex activations. On the backwards pass we reinsert the phase to give our recovered W.

$$\hat{W}_m = \hat{U}_m e^{j\theta_m} \tag{1.3.2}$$

1.3.3 Inverting the Wavelet Decomposition

Using the DT$\mathbb{C}$WT makes inverting the wavelet transform simple, as we can simply feed the coefficients through the synthesis filter banks to regenerate the signal. For complex ψ, this is convolving with the conjugate transpose $\widetilde{\psi}$:

$$\begin{aligned} \hat{U}_{m-1} &= \hat{S}_{m-1} + \hat{W}_m & (1.3.3)\\ &= S_{m-1} * \widetilde{\phi}_J + \sum_{j,\theta} W_m(\boldsymbol{u}, j, \theta) * \widetilde{\psi}_{j,\theta} & (1.3.4) \end{aligned}$$

1.4 Visualization with Inverse Scattering

To examine our ScatterNet, we scatter all of the images from ImageNet's validation set and record the top 9 images which most highly activate each of the C channels in the ScatterNet. This is the *identification* phase (in which no inverse scattering is performed).

Then, in the *reconstruction* phase, we load in the $9 \times C$ images, and scatter them one by one. We take the resulting 52 channel output vector and mask all but a single value in the channel we are currently examining.

This 1-sparse tensor is then presented to the inverse scattering network from Figure 1.2 and projected back to the image space. Some results of this are shown in Figure 1.3. This figure shows reconstructed features from the layers of a ScatterNet. For a given output channel, we show the top 9 activations projected independently to pixel space. For the first and second order coefficients, we also show the patch of pixels in the input image which cause this large output. We display activations from various scales (increasing from first row to last row), and random orientations in these scales.

The order 1 scattering (labelled with 'Order 1' in Figure 1.3) coefficients look quite similar to the first layer filters from the well known AlexNet CNN [20]. This is not too surprising, as the first order scattering coefficients are simply a wavelet transform followed by average pooling. They are responding to images with strong edges aligned with the wavelet orientation.

The second order coefficients (labelled with 'Order 2' in Figure 1.3) appear very similar to the order 1 coefficients at first glance. They too are sensitive to edge-like features, and some of them (e.g. third row, third column and fourth row, second column) are mostly just

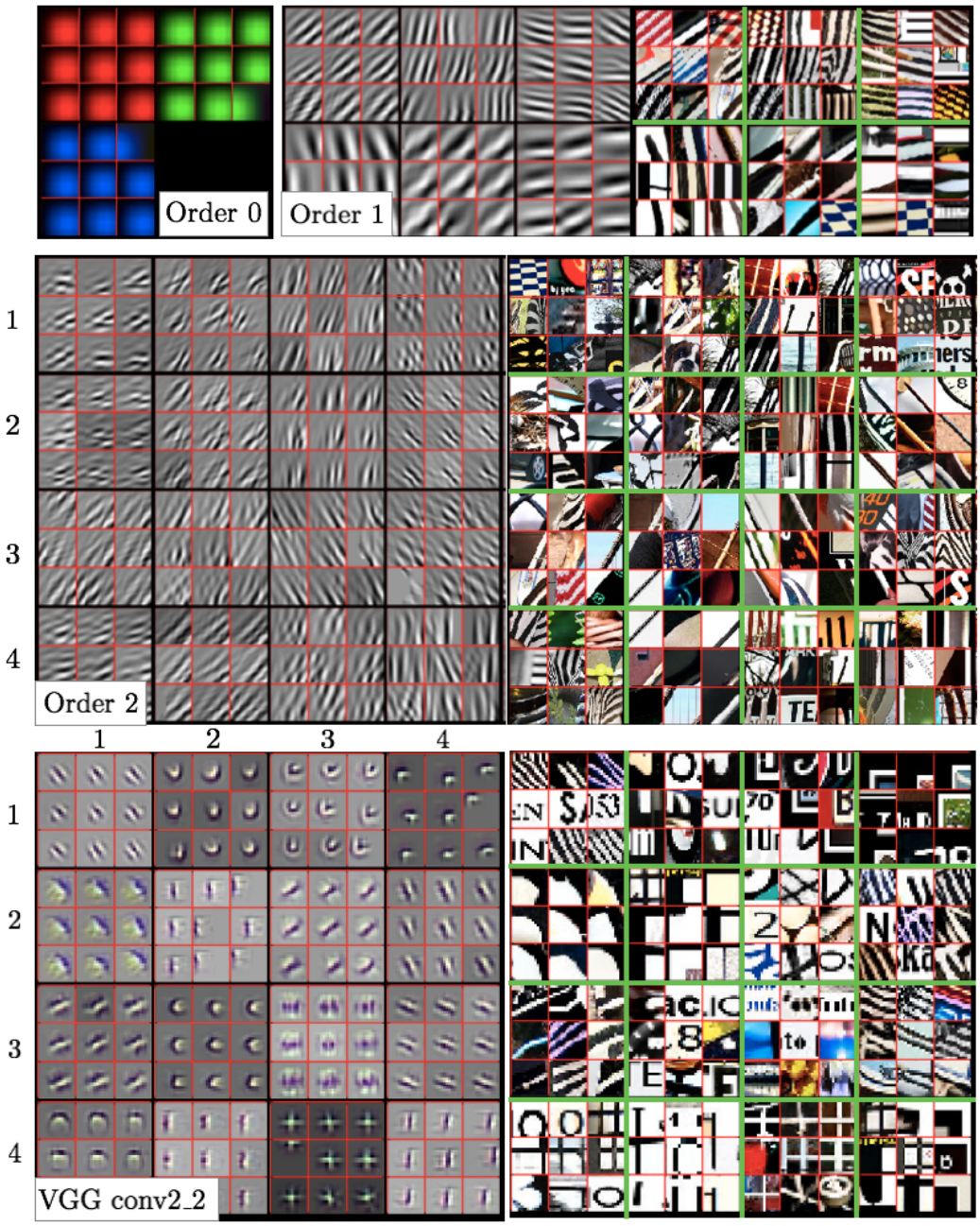


Figure 1.3: Visualization of a random subset of features from S_0 (all 3), S_1 (6 from the 12) and S_2 (16 from the 36) scattering outputs. We record the top 9 activations for the chosen features and project them back to the pixel space. We show them alongside the input image patches which caused the large activations. We also include reconstructions from layer conv2_2 of VGG Net [1](a popular CNN, often used for feature extraction) for reference — here we display 16 of the 128 channels. The VGG reconstructions were made with a CNN DeconvNet based on [6]. Image best viewed digitally.

that. These are features that have the same oriented wavelet applied at both the first and second order. Others, such as the nine in the top left square (first row, first column), and top right square (first row, fourth column) are more sensitive to checker-board like patterns. Indeed, these are activations where the orientation of the wavelet for the first and second order scattering were far from each other (15° and 105° for the first row, first column and 105° and 45° for the first row, fourth column).

For comparison, we include reconstructions from the second layer of the well-known VGG CNN (labelled with 'VGG conv2_2', in Figure 1.3). These were made with a DeconvNet, following the same method as [6]. Note that while some of the features are edge-like, we also see higher order shapes like corners, crosses and curves.

These reconstructions show that the features extracted from ScatterNets vary significantly from those learned in CNNs after the first order. In many respects, the features extracted from a CNN like VGGNet look preferable for use as part of a classification system.

1.5 Channel Saliency

To get another heuristic on the importance of the ScatterNet channels, let us examine the effect on inference scores observed when zeroing out Scattering channels. [6], [21] have done similar studies but over patches of the input image, with the former using a patch of grey values as the occlusion mask, and the latter using a set of random pixels.

We must be careful to occlude with a sensible mask, the S_0x, S_1x and S_2x all have very different probability densities. Assuming $x \sim \mathcal{N}(0, \sigma^2 I)$ (already a fairly weak assumption), the pdf of S_0x will also be a zero mean gaussian. However, the distributions of S_1x and S_2x are more complex - the real and imaginary parts of the DT$\mathbb{C}$WT are sparse but are strongly correlated. Choosing a sensible random mask is therefore difficult so we instead use a constant mask. Analysis of the datasets show that zero is very close to the maximum likelihood value for each channel so we occlude channels by simply setting them to zero at every spatial location.

1.5.1 Experiment Setup

We take a network similar to the one from **??** but using the colour operation described in subsection 1.2.1 so the scattering output has 51 output channels. We further set $C = 50$ so the conv1 layer has 100 channels. We train this network on the same 3 datasets - CIFAR-10, CIFAR-100 and Tiny ImageNet, and report the drop in classification scores on the validation set after removing one channel at a time.

We additionally display the weight matrix for the first learned layer of the network trained on Tiny ImageNet. This gives us a second perspective on the channel importance by looking at the relative weights of the ScatterNet channels passed on to the successive 100 channels.

The weight matrix is convolutional filter of size $w \in \mathbb{R}^{100\times51\times3\times3}$. We define:

$$A^{rms}_{c,f} = \sqrt{\frac{\sum_{i,j} w[f,c,i,j]^2}{\sum_f \sum_{i,j} w[f,c,i,j]^2}} \tag{1.5.1}$$

This gives us a matrix A^{rms} (for root mean squared) which has columns of unit energy representing the different output channels after conv1. The row values then show how much each scattering channel contributes to each output channel. This is shown in Figure 1.5.

1.5.2 Discussion

First we look at Tiny ImageNet in Figure 1.4. Note that when any of the S_0 channels are removed, the validation accuracy drops sharply for all 3 datasets. A similar result happens when any of the S_1 channels are zeroed out.

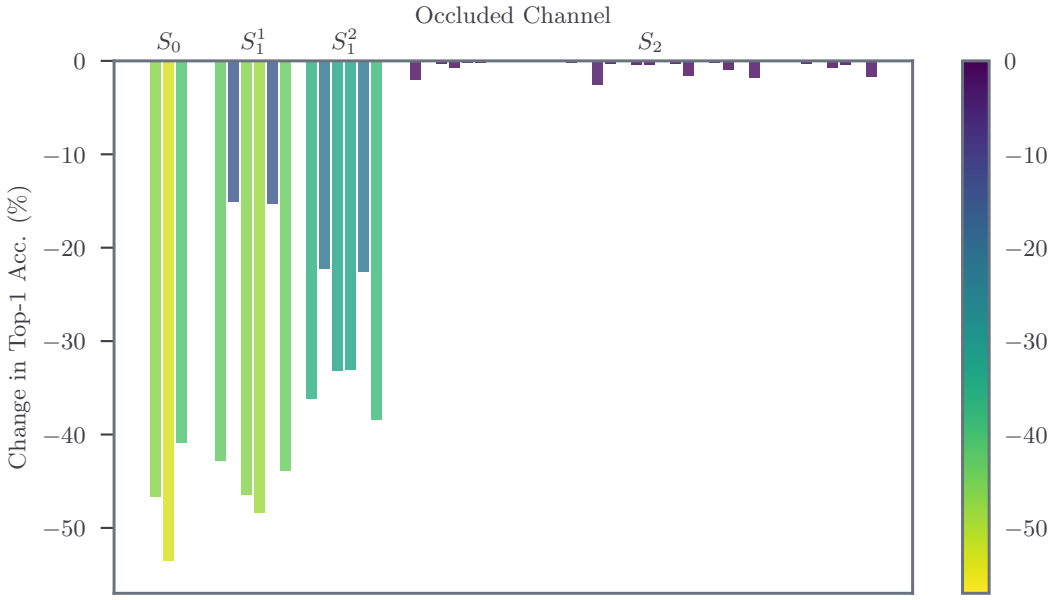
For both the first and second scales of the first order coefficients, S_1^1 and S_1^2, there are two channels that seem less important - the second and fifth channels, corresponding to the 45° and 135° wavelets. Often the high-high portion of the first scale coefficients are considered mostly noise, but this does not explain why the 45° and 135° channels for the second scale coefficients are also less important. A possible interesting conclusion to be drawn from this is that the dataset does not have as many diagonal edges in it as horizontal and vertical edges.

To test this, we retrain the network but this time rotate the input images randomly 30° clockwise or anti-clockwise in both training and validation. We then rerun the occlusion experiment for all channels and plot the resulting changes in 1.4b. Interestingly, for this network, the 45° and 135° wavelets for S_1^2 are now the most important of the 6, which validates our assumption. The corresponding wavelets for S_1^1 have become more important, but it is likely that they remain less salient because of the effects of the higher bandwidth for the diagonal wavelets.

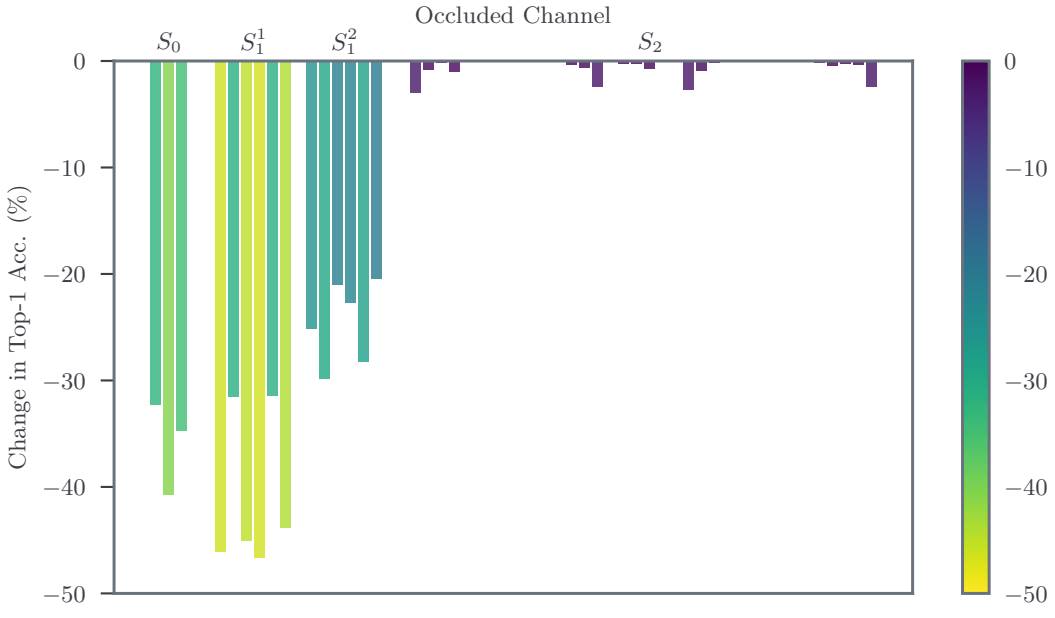
Comparatively, the S_2 channels have little effect on the classification score when individually masked. The four largest drops in accuracy for S_2 happen when $\theta_1 = \theta_2 \in \{15°, 75°, 105°, 165°\}$. When $\theta_1 \neq \theta_2$ we saw the ripple like patterns in Figure 1.3, and we see here that the network has mostly learned to not depend on them.

Figure 1.5 tells a similar tale for the Scattering channels. Here we see directly how much and how little each of the channels is used by the first layer of the network, with the low intensity values in S_2 indicating that the next layer's outputs are less dependent on these coefficients.

We include the same analysis for the two CIFAR datasets in Figure 1.6 for completion, although the insight gained here is the same - the S_2 coefficients are the least important. One notable difference to Figure 1.4 is in the S_1^2 coefficients, which have reduced importance in CIFAR, but with the smaller input spatial size, this comes as no surprise.



(a) Tiny ImageNet



(b) Rotated Tiny ImageNet

Figure 1.4: **Tiny ImageNet changes in accuracy from channel occlusion.** Numbers reported are the drop in final classification accuracy when a channel is set to zero. The bars are coloured relative to their magnitude to aid seeing the differences for the S_1 coefficients. (a) When any of the lowpass channels S_0 are removed, the classification accuracy drops sharply, note that the middle channel, corresponding to green, is unsurprisingly the most important of the three colours. The first scale, first order scattering coefficients S_1^1 are slightly more important than the second scale coefficients. The 36 S_2 coefficients have very little individual effect on the validation score when removed. (b) The same network trained with input samples rotated by $\pm 30°$. In (a) the second and fifth orientations for both S_1^1 and S_1^2, corresponding to the 45° and 135° wavelets, are comparatively less important than other orientations at the same scale. This suggests that perhaps the dataset does not have much diagonal information. When rotated this trend changes and the diagonal wavelets at both scales become more important.

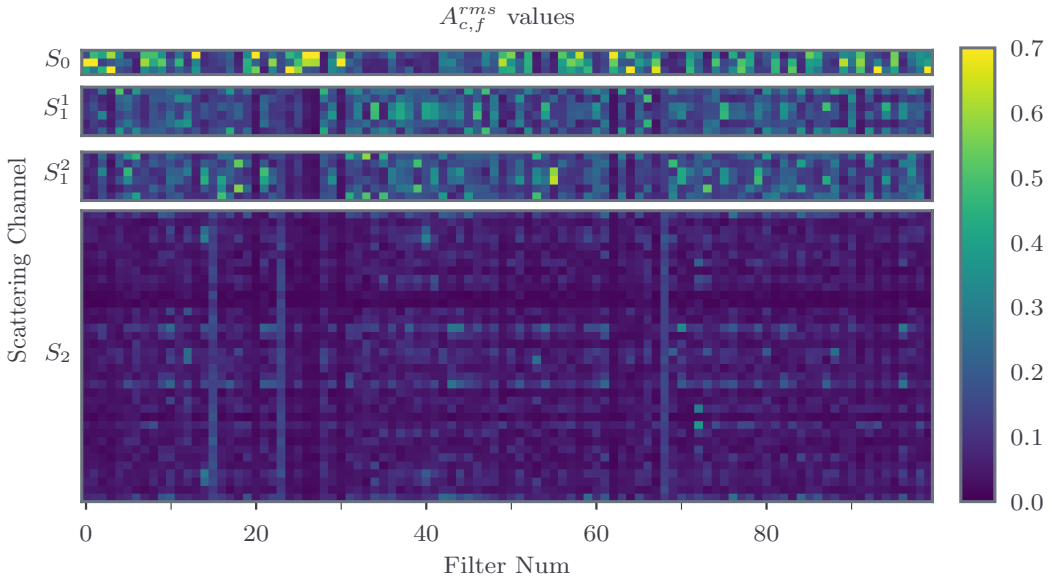


Figure 1.5: **Channel weights for first learned layer.** A visualiztion of the matrix A^{rms} from (1.5.1). The columns of the matrix all have unit norm and represent how much relative energy comes from each scattering output channel. Most of the filters are heavily dependent on S_0, many are dependent on S_1 and only a few take information from S_2.

1.6 Corners, Crosses and Curves

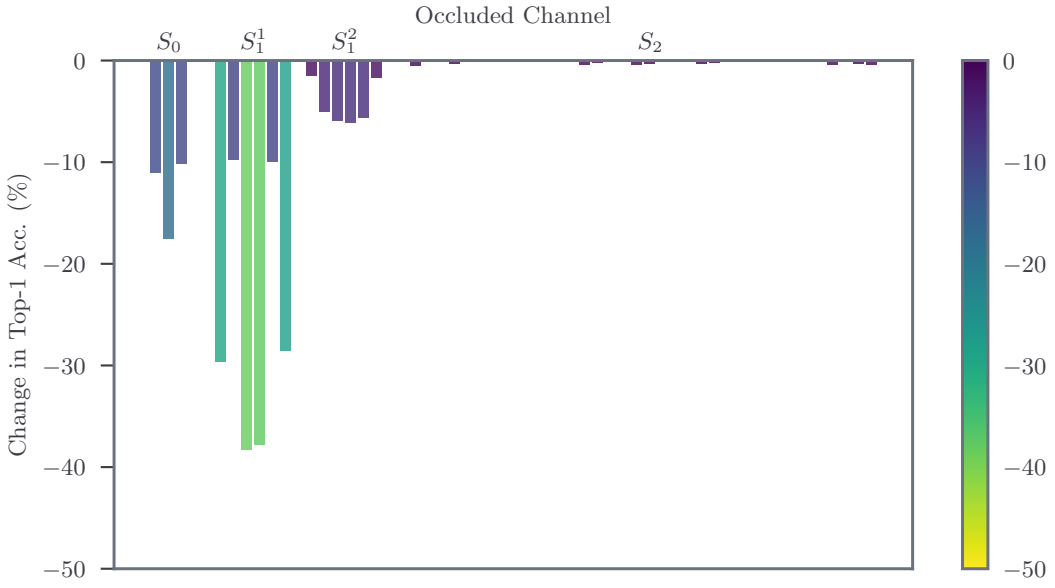
As a final part of this chapter, we would like to highlight some of the filters possible by making small modifications to the ScatterNet design. The visualizations shown here are mostly inspirational, as we did not see any marked improvement in using them as a fixed front end for the ScatterNet system. However they are the basis for the next chapter of work in adding learning in between Scattering layers.

[14] and [10] introduced the idea of a 'Roto-Translation' ScatterNet. Invariance to rotation could be made by applying averaging (and bandpass) filters across the K orientations from the wavelet transform *before* applying the complex modulus. Momentarily ignoring the form of the filters they apply, referring to them as $h \in \mathbb{C}^K$, we can think of this stage as stacking the K outputs of a complex wavelet transform on top of each other, and convolving these filters h over all spatial locations of the wavelet coefficients $W_m x$ (this is equivalent to how filters in a CNN are fully connected in depth):

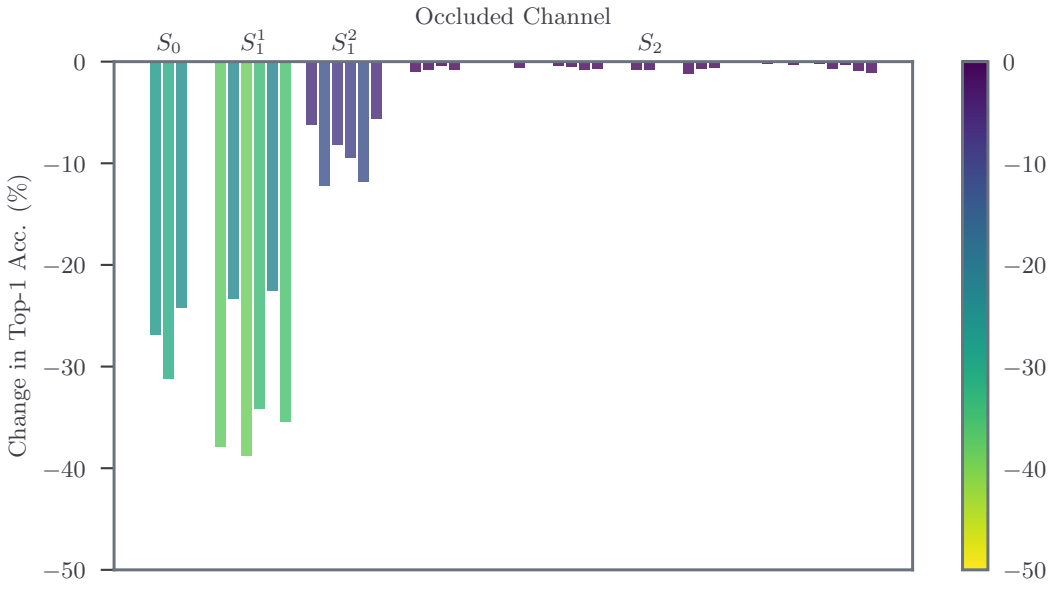
$$V_m x(\lambda, \boldsymbol{u}) = W_m x * h = \sum_{\theta} W_m x(\lambda, \boldsymbol{u}) h(\theta) \tag{1.6.1}$$

We then take the modulus of these complex outputs to make a second propagated signal:

$$U'_m x \triangleq |V_m x| = |W_m x * h| = |U_{m-1} x * \psi_{\lambda_m} * h| \tag{1.6.2}$$



(a) CIFAR-10



(b) CIFAR-100

Figure 1.6: **CIFAR changes in accuracy from channel occlusion.** Numbers reported are the drop in final classification accuracy when a channel is set to zero. The bars are coloured relative to their magnitude to aid seeing the differences for the S_1 coefficients. Unlike Figure 1.4 the S_1^2 coefficients are less important. CIFAR has a smaller image size than Tiny ImageNet so this is not surprising.

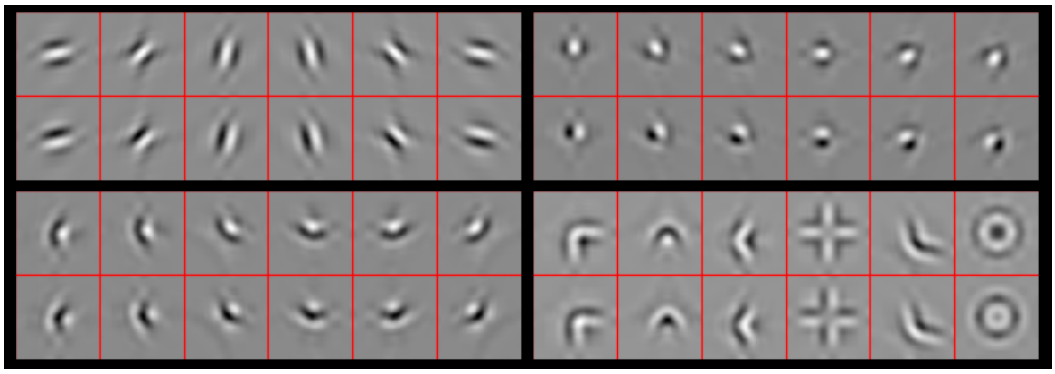


Figure 1.7: **Shapes possible by filtering across the wavelet orientations with complex coefficients.** All shapes are shown in pairs: the top image is reconstructed from a purely real output, and the bottom image from a purely imaginary output. These 'real' and 'imaginary' shapes are nearly orthogonal in the pixel space (normalized dot product < 0.01 for all but the doughnut shape in the bottom right, which has 0.15) but produce the same U', something that would not be possible without the complex filters of a ScatterNet. Top left - reconstructions from U_1 (i.e. no cross-orientation filtering). Top right- reconstructions from U_1' using a $12 \times 1 \times 1$ Morlet Wavelet, similar to what was done in the 'Roto-Translation' ScatterNet described in [10], [14]. Bottom left - reconstructions from U_1' made with a more general $12 \times 1 \times 1$ filter, described in Equation 1.6.3. Bottom right - some reconstructions possible by filtering a general $12 \times 3 \times 3$ filter.

We present a variation on this idea, by filtering with a more general $h \in \mathbb{C}^{12 \times H \times W}$. We use 12 channels rather than 6, as we use the $K = 6$ orientations and their complex conjugates; each wavelet is a 30° rotation of the previous, so with 12 rotations, we can cover the full 360°.

Figure 1.7 shows some reconstructions from these V coefficients. Each of the four quadrants show reconstructions from a different class of ScatterNet layer. All shapes are shown in real and imaginary Hilbert-like pairs; the top images in each quadrant are reconstructed from a purely real V, while the bottom inputs are reconstructed from a purely imaginary V. This shows one level of invariance of these filters, as after taking the complex magnitude, both the top and the bottom shape will activate the filter with the same strength. In comparison, for the purely real filters of a CNN, the top shape would cause a large output, and the bottom shape would cause near 0 activity (they are nearly orthogonal to each other).

In the top left, we display the 6 wavelet filters for reference (these were reconstructed from U_1, not V_1). In the top right of the figure we see some of the shapes made by using the h's from the Roto-Translation ScatterNet [10], [14]. The bottom left is where we present some of our novel kernels. These are simple corner-like shapes made by filtering with $h \in \mathbb{C}^{12 \times 1 \times 1}$

$$h = [1, j, j, 1, 0, 0, 0, 0, 0, 0, 0, 0] \tag{1.6.3}$$

The six orientations are made by rolling the coefficients in h along one sample (i.e. $[0,1,j,j,1,0,\ldots]$, $[0,0,1,j,j,1,0,\ldots]$, $[0,0,0,1,j,j,1,0,\ldots]$ $\ldots$). Coefficients roll back around (like circular convolution) when they reach the end.

Finally, in the bottom right we see shapes made by $h \in \mathbb{C}^{12\times3\times3}$. Note that with the exception of the ring-like shape which has 12 non-zero coefficients, all of these shapes were reconstructed with h's that have 4 to 8 non-zero coefficients of a possible 64. These shapes are now beginning to more closely resemble the more complex shapes seen in the middle stages of CNNs.

1.6.1 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 DT$\mathbb{C}$WT 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.

$$\begin{aligned}
\mathrm{Yh}[\,\mathrm{k}\,][\,\mathrm{y}\,,\mathrm{x}\,,0\,] &= 1\\
\mathrm{Yh}[\,\mathrm{k}\,][\,\mathrm{y}\,,\mathrm{x}\,,1\,] &= \mathrm{j}\\
\mathrm{Yh}[\,\mathrm{k}\,][\,\mathrm{y}\,,\mathrm{x}\,,2\,] &= \mathrm{j}\\
\mathrm{Yh}[\,\mathrm{k}\,][\,\mathrm{y}\,,\mathrm{x}\,,3\,] &= 1
\end{aligned}$$

and then do the inverse DTCWT we get the result shown in Figure 1.8. Doing this is equivalent to the deconvolution of Zeiler and Fergus. What we're really seeing in Figure 1.8 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.

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

$$\mathbf{f} = [1,j,j,1,0,0,0,0,0,0,0,0] \tag{1.6.4}$$

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 1.9. In particular, it is the first shape rotated by 30° for each k.

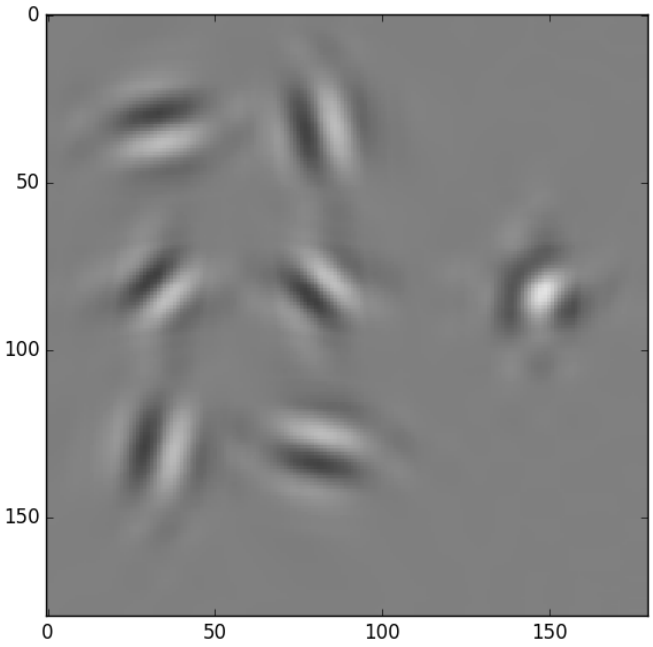


Figure 1.8: Left of image shows the imaginary part of 6 DTCWT impulse responses. Right of image shows the effect of combining with the filter $[1, j, j, 1]$

1.7 Discussion

This chapter presents a way to investigate what the higher orders of a ScatterNet are responding to - the DeScatterNet described in section 1.3. Using this, we have shown that the second 'layer' of a ScatterNet responds strongly to patterns that are very dissimilar to those that highly activate the second layer of a CNN. As well as being dissimilar to CNNs, visual inspection of the ScatterNet's patterns reveal that they may be less useful for discriminative tasks, and we believe this may be causing the current gaps in state-of-the-art performance between the two.

Additionally, we performed occlusion tests to heuristically measure the importance of the individual scattering channels. The results of this test reaffirmed the suspicions raised from the visualizations. In particular, many of the second order Scattering coefficients may not be very useful in a CNN. Those that were visible were typically when the second order wavelet had the same orientation as the first.

Finally, we demonstrated the possible shapes attainable when we filtering across orientations with complex mixing coefficients. We believe that this mixing is a key step in the development of ScatterNets and wavelets in deep learning systems.

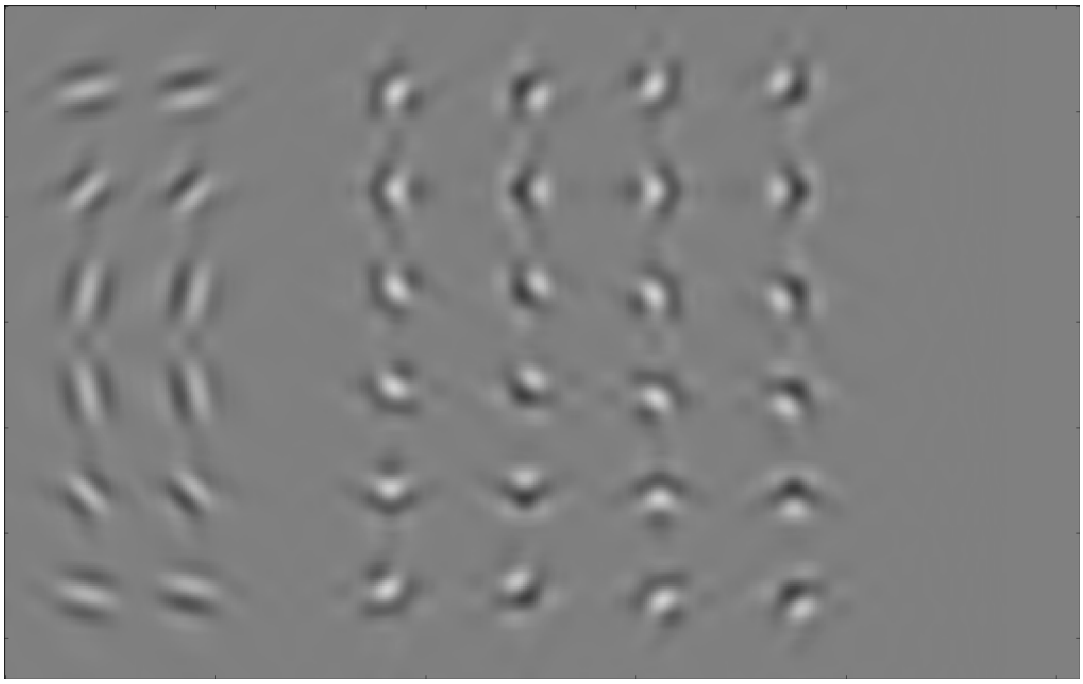


Figure 1.9: Left shows the 6 real and imaginary DTCWT impulse responses. Right of image shows the resulting 12 corners achieved from rotating $[1, j, j, 1]$ through the 12 of these. The even columns are the 'Hilbert pair' of the odd columns, obtained by shifting all the coefficients by $90°$ (i.e. $[j, -1, -1, j]$).

References

[1] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", *arXiv:1409.1556 [cs]*, Sep. 2014. arXiv: 1409.1556 [cs].

[2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper With Convolutions", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition", in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778. arXiv: 1512.03385.

[4] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely Connected Convolutional Networks", in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 2261–2269. arXiv: 1608.06993.

[5] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why Should I Trust You?": Explaining the Predictions of Any Classifier", *arXiv:1602.04938 [cs, stat]*, Feb. 2016. arXiv: 1602.04938 [cs, stat].

[6] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks", en, in *Computer Vision – ECCV 2014*, Sep. 2014, pp. 818–833.

[7] A. Mahendran and A. Vedaldi, "Understanding Deep Image Representations by Inverting Them", *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[8] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[9] J. Bruna and S. Mallat, "Invariant Scattering Convolution Networks", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1872–1886, Aug. 2013.

[10] E. Oyallon and S. Mallat, "Deep Roto-Translation Scattering for Object Classification", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2865–2873.

[11] A. Singh and N. Kingsbury, "Dual-Tree Wavelet Scattering Network with Parametric Log Transformation for Object Classification", in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2017, pp. 2622–2626. arXiv: 1702.03267.

[12] A. Singh and N. Kingsbury, "Multi-Resolution Dual-Tree Wavelet Scattering Network for Signal Classification", en, in *11th International Conference on Mathematics in Signal Processing*, Birmingham, UK, Dec. 2016.

[13] E. Oyallon, E. Belilovsky, and S. Zagoruyko, "Scaling the Scattering Transform: Deep Hybrid Networks", in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 5619–5628. arXiv: 1703.08961.

[14] L. Sifre and S. Mallat, "Rotation, Scaling and Deformation Invariant Scattering for Texture Discrimination", in *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2013, pp. 1233–1240.

[15] S. Mallat, "Group Invariant Scattering", en, *Communications on Pure and Applied Mathematics*, vol. 65, no. 10, pp. 1331–1398, Oct. 2012.

[16] M. Zeiler, G. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning", in *2011 IEEE International Conference on Computer Vision (ICCV)*, Nov. 2011, pp. 2018–2025.

[17] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for Simplicity: The All Convolutional Net", *arXiv:1412.6806 [cs]*, Dec. 2014. arXiv: 1412.6806 [cs].

[18] R. Fong and A. Vedaldi, "Interpretable Explanations of Black Boxes by Meaningful Perturbation", *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 3449–3457, Oct. 2017. arXiv: 1704.03296.

[19] 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.

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

[21] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Object Detectors Emerge in Deep Scene CNNs", *arXiv:1412.6856 [cs]*, Dec. 2014. arXiv: 1412.6856 [cs].