THE UNIVERSITY OF QUEENSLAND
AUSTRALIA
CREATE CHANGE

Spectral Autoencoders

Frequency-Based autoencoders

Jevi Waugh, Research Assistant & Casual Academic
Research Supervisor: Dr. Shekhar “Shakes” Chandra

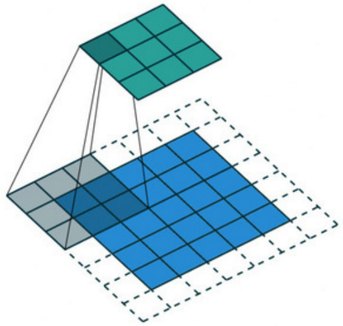
Acknowledgement of Country

- The University of Queensland (UQ) acknowledges the Traditional Owners and their custodianship of the lands on which we meet.
- We pay our respects to their Ancestors and their descendants, who continue cultural and spiritual connections to Country.
- We recognise their valuable contributions to Australian and global society.

Image: Digital reproduction of *A guidance through time* by Casey Coolwell and Kyra Mancktelow

Why Rethink Vision Models?

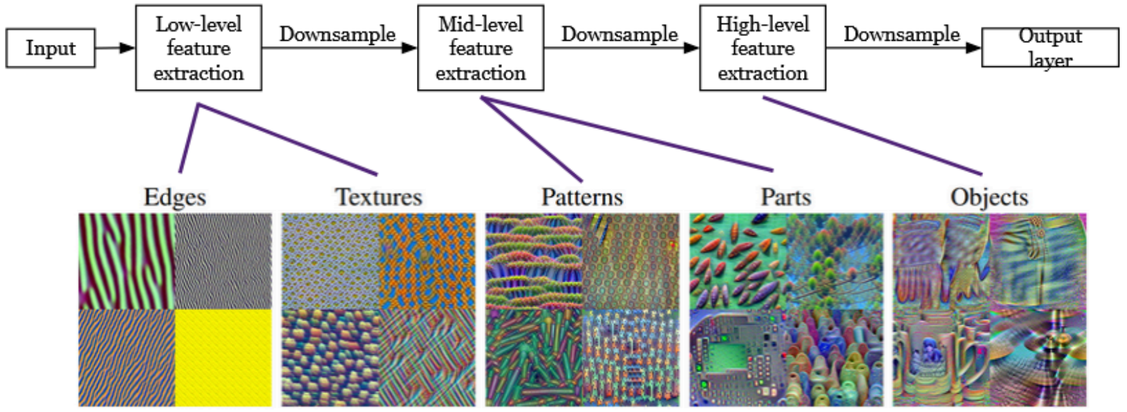
Today's AI sees the world like this.

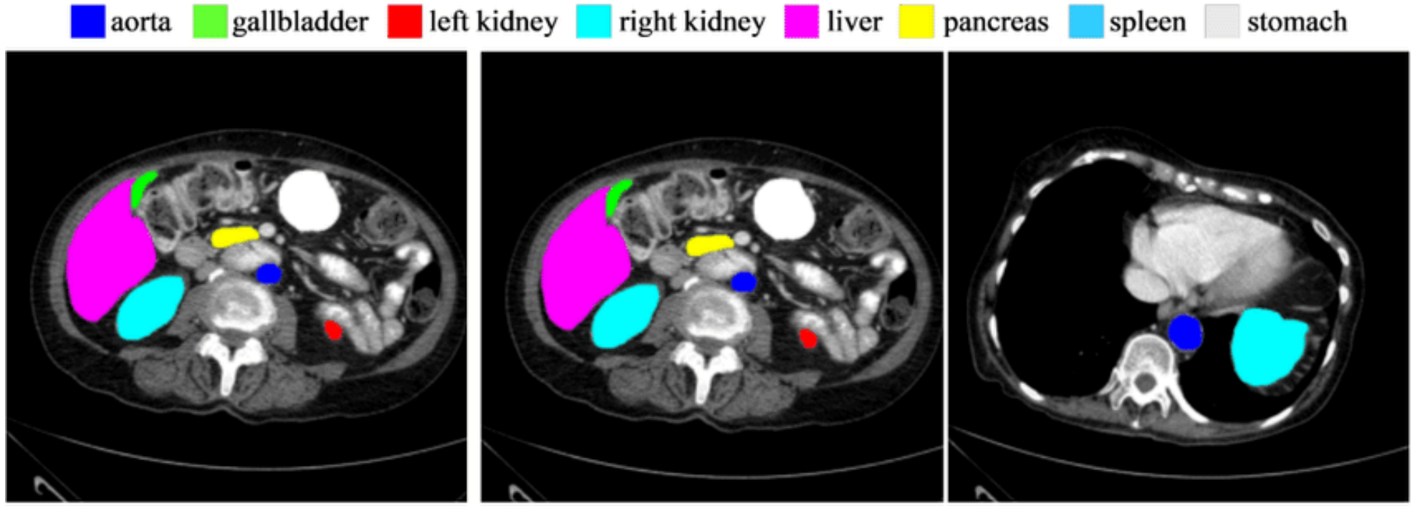


Powerful. But expensive. And hard to understand.

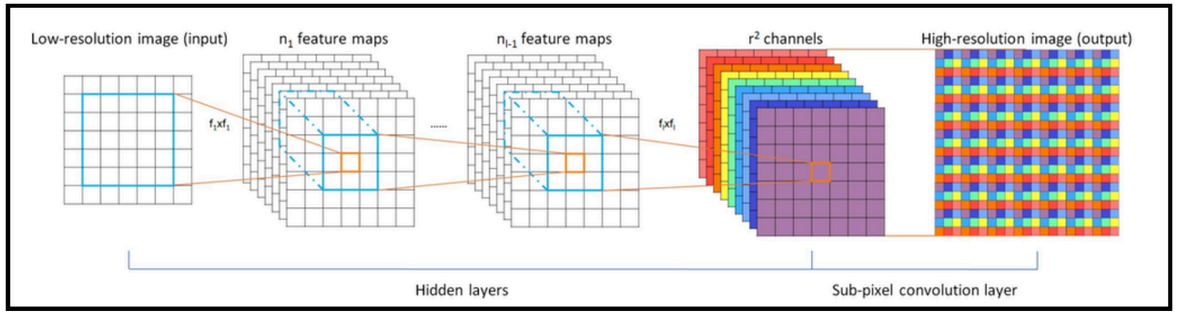
Segmentation is hard!

Is image-space the only way?





Pixel shuffle



Pixel shuffle takes r² channels at (h, w) and interleaves them into one channel at (rh, rw). It is normally a spatial reshape. In k-space it becomes a closed-form identity. no reshape, no inverse transform:

The mathematical definitions are as follows:

$$y[c_{out}, rh + d_y, rw + d_x] = x[c_{in}, h, w]$$

$$y = \text{PixelShuffle}(\text{X}) \in \mathbb{R}^{(C \cdot r^2) \times h \times w}$$

$$d_x = C_{in} \ mod \ r$$
$$d_y = C_{in} // r$$
$$\text{where } d_x \wedge d_y \in \{0, \cdots, r-1\}$$

For a local channel we have:

$$c_{\text{in}} = d_y r + d_x.$$

To represent the global space and all channels:

$$c_{\text{in}} = \underbrace{c_{\text{out}}}_{\text{current output channel}} \cdot \underbrace{r^2}_{\text{size of matrix}} + \underbrace{d_y r + d_x}$$

Why do we do this?

- Transpose convolutions insert zeros between pixels (zero-expansion), which can cause uneven overlapping of kernels and create a grid-like "checkerboard" pattern in the output.
- Pixel shuffle operates via a clean channel-to-space rearrangement, leading to a more uniform spatial grid. It's also much more efficient than Conv.T

The Fourier Shuffle

What if we could do the pixel shuffle in the frequency space.

We need an expression that matches what pixel shuffle does but in fourier space.

$$X[K_y, K_x] = \sum_{m=0}^{H-1} \sum_{n=0}^{W-1} x[m,n] \cdot e^{-2\pi i(\frac{K_y m}{H} + \frac{K_x n}{W})}$$

$$DFT(PixelShufle(x))$$

$$Y[c_{\text{out}}, k_y, k_x] = \sum_{d_y=0}^{r-1} \sum_{d_x=0}^{r-1} X^{e}[c_{\text{out}} r^2 + d_y r + d_x, k_y \ Mod \ h, \ k_x \ Mod \ w] \exp\left(-2\pi i \left(\frac{k_y d_y}{H} + \frac{k_x d_x}{W}\right)\right)$$

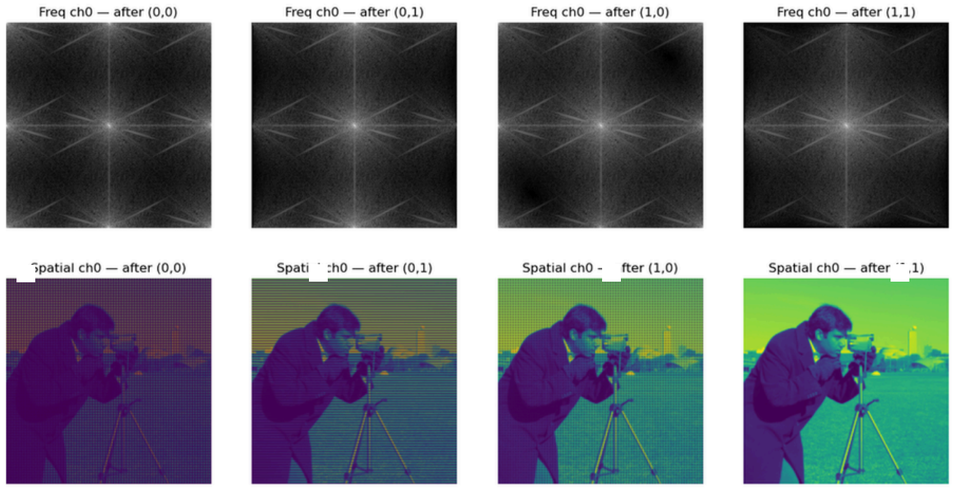
The *mod h, mod w* term tiles the small spectrum *r×r* times across the large one. Upsampling in space = periodic repetition in frequency. No new information is invented.
it is the same spectrum, read r^2 times.

The exponential shifts each sub-channel to its own sub-pixel offset *(dy, dx)*. A linear phase in k-space is a translation in image space. This is what keeps the r^2 copies from landing on top of each other.

A visual Example of the fourier Shuffle

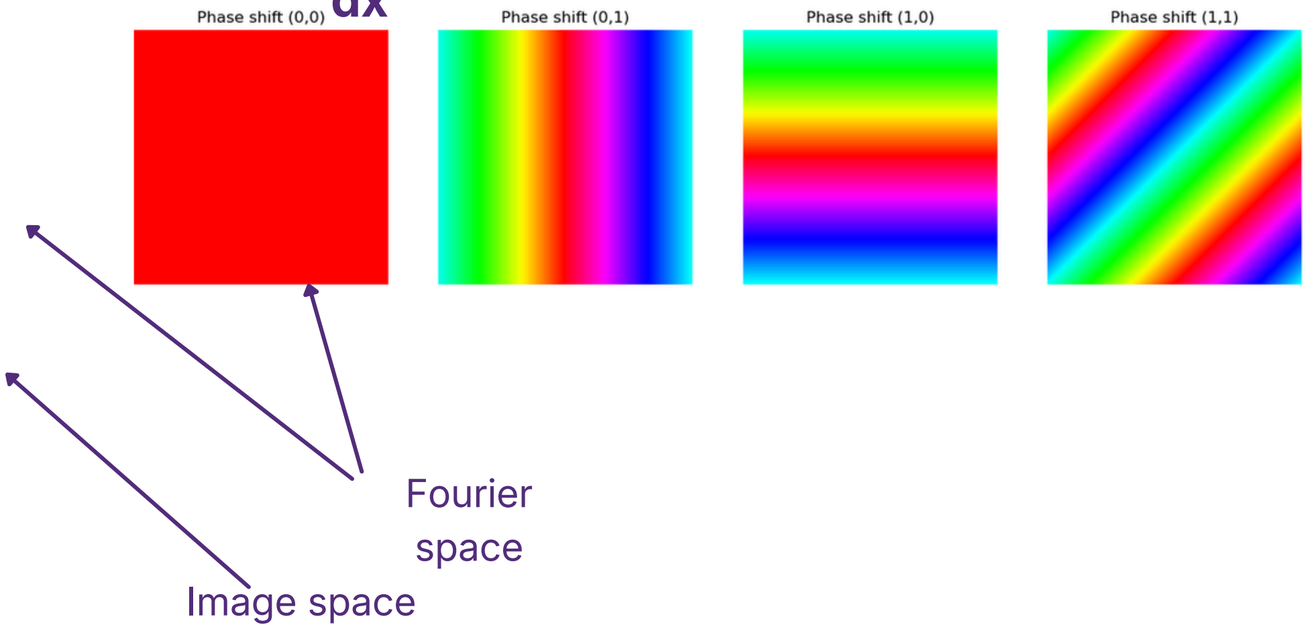
$$Y[c_{\text{out}}, k_y, k_x] = \sum_{d_y=0}^{r-1} \sum_{d_x=0}^{r-1} X^{e}\left[c_{\text{out}} r^2 + d_y r + d_x, k_y \ Mod\ h,\ k_x \ Mod\ w\right] \exp\left(-2\pi i \left(\frac{k_y d_y}{H} + \frac{k_x d_x}{W}\right)\right)$$

After Accumulating for each Y



Please note that we accumulate then transform back to image space for the second row

This is for each dy, dx



The unshuffle problem, and the Kaleidoscope

Kaleidoscope and the fourier version

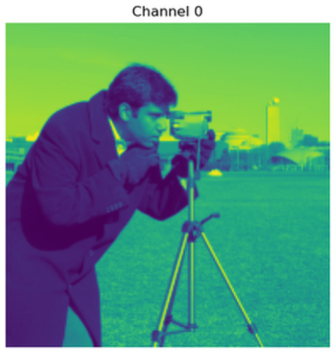
Downsampling is the unshuffle: split one channel at (H, W) back into r² channels at (H/r, W/r). Those r² channels are the polyphase components, the pixels on each sub-grid, x[i::r, j::r].

So in practice we complete the fold outside k-space
ifft2 → fold / partition channels → fft2 which equates to Fourier Unshuffle

This is 1D Kaleidoscope in the frequency domain

$$\hat{\mathcal{K}}^{\varrho}[m] = \underbrace{\sum_{r=0}^{v-1} e^{-j2\pi mr/v}}_{constant} \underbrace{\sum_{q=0}^{N/v} x[qv+r]\cdot e^{-j2\pi m\sigma q/N}}_{P(q,v,r;\sigma)}$$

The 2D version is simply done across the opposite axis of the input.

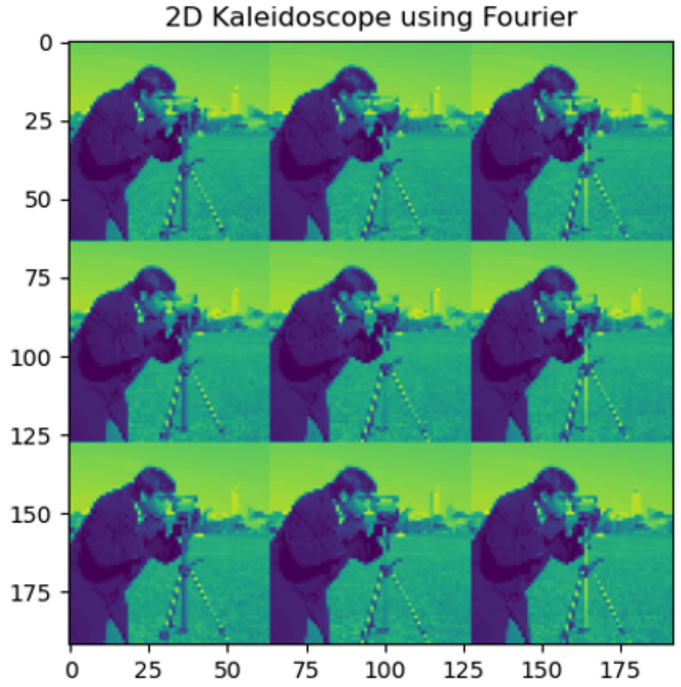


This is 1D Kaleidoscope in the image domain

$$K_{\nu,\sigma}\{x[n]\}[k] = \sum_{n=0}^{N-1} x[n]\cdot\delta\big(k-\kappa[n;\nu,\sigma]\big), \quad k\in\mathbb{Z}_N$$

where $N,\nu\in\mathbb{Z}^{+},\ \sigma\in\mathbb{Z},\ n\in\mathbb{Z}_N$

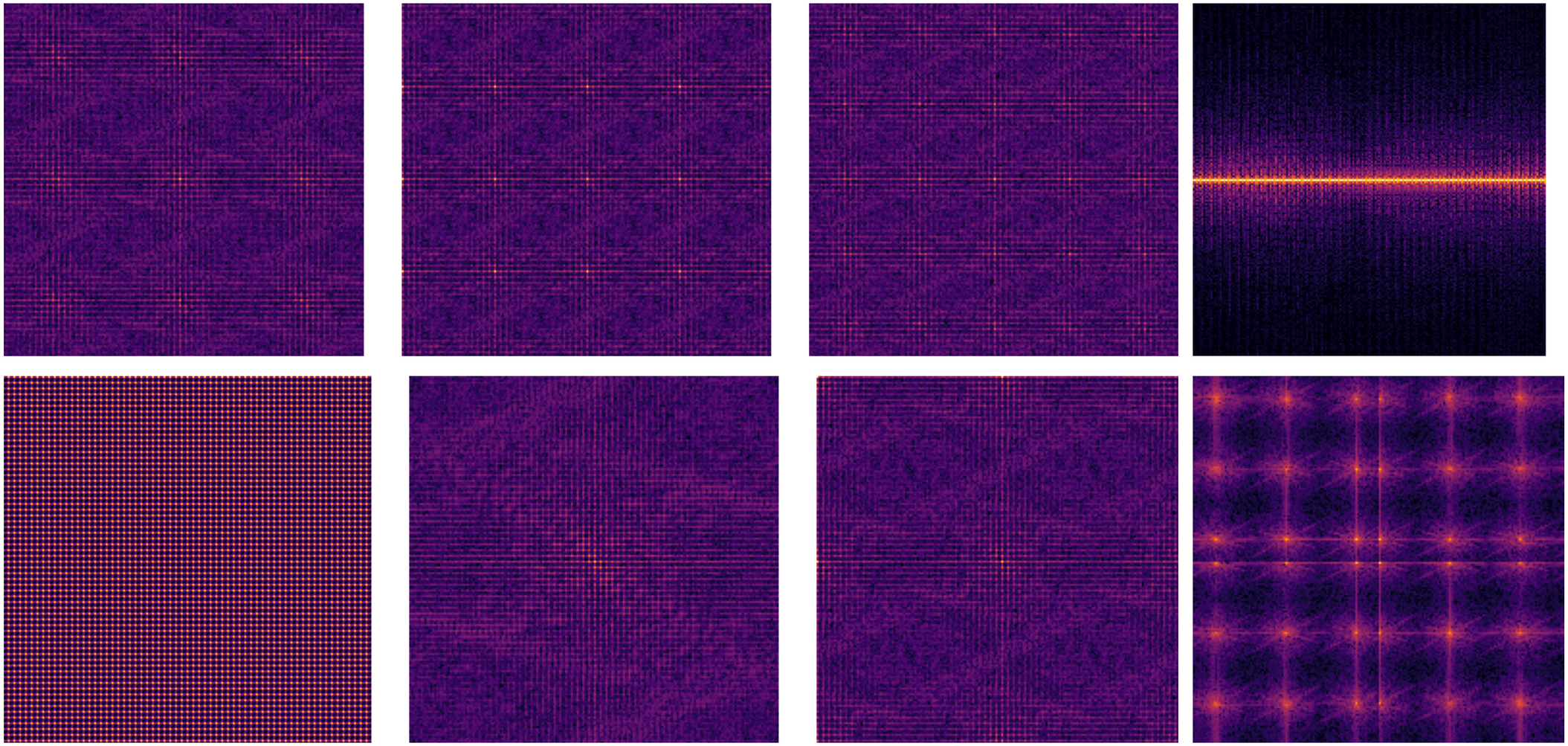
Traditional Kaleidoscope



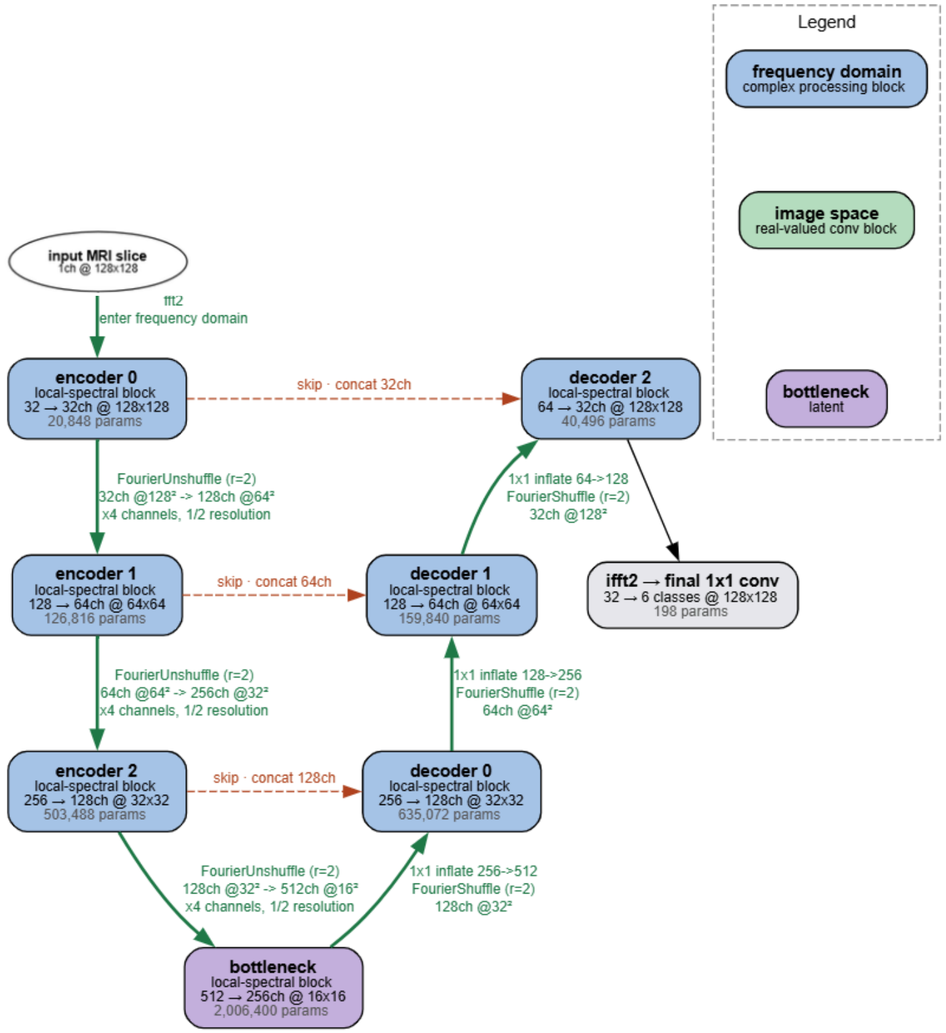
1D Mapping

$$\kappa(n) = \frac{N}{\nu}(n \bmod \nu) + \sigma\lfloor n/\nu \rfloor$$

Side effects: Emerging Fractals



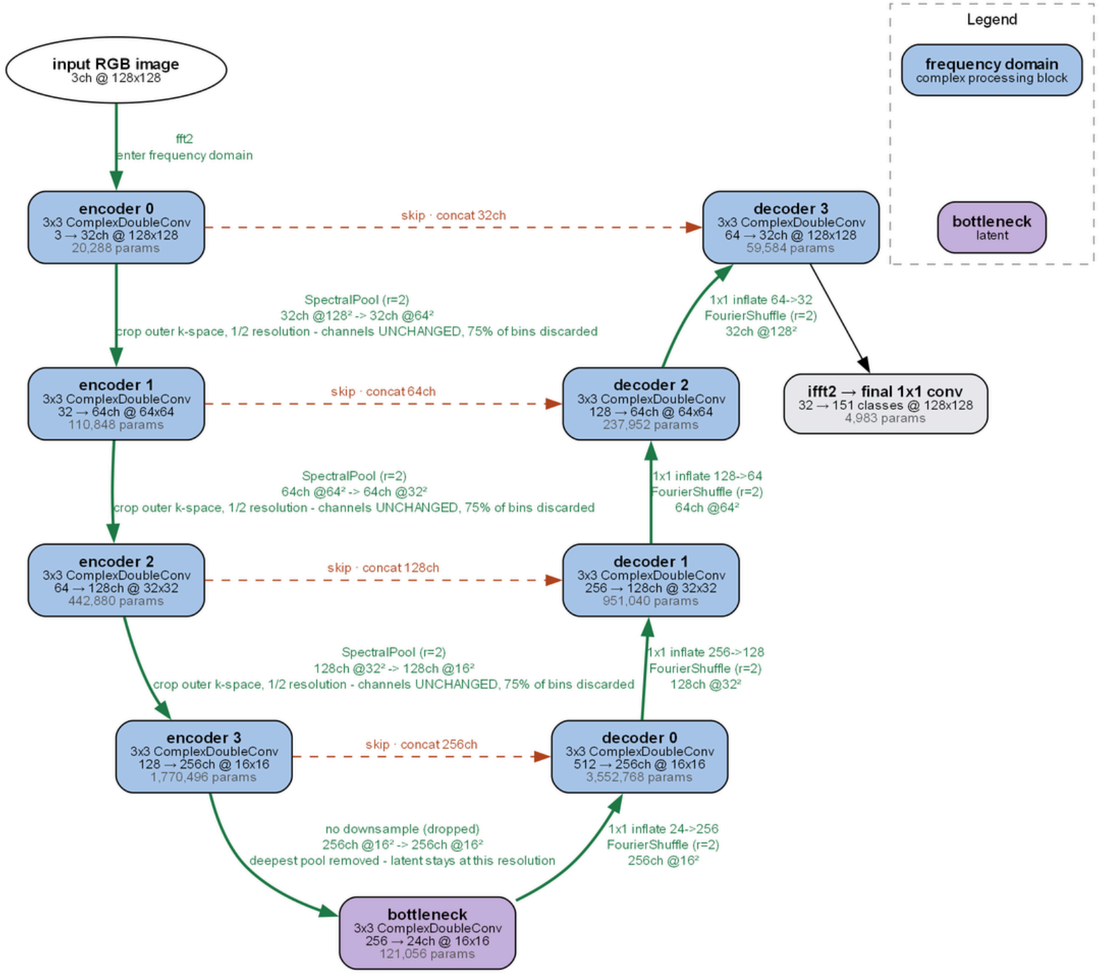
FUNET: FourierUnet



One fft2 in, one ifft2 out. Everything between is complex-valued and never leaves k-space

- 1. Pad, then transform. Input is zero-padded so H/W divide cleanly by r ** len(features) (so every encoder level lands on an integer size), then x = torch.fft.fft2(x). From here to the end, every tensor is complex64.
- 2. Encoder. Per level: conv → save skip → downsample.
  a. conv is a ComplexDoubleConv two ComplexConv2d + ComplexReLU, where a complex conv is the four-real-conv expansion (a+bi)*(c+di).
  b. The skip is saved before the downsample, so it carries full-frequency content.
  c. Downsample is FourierUnshuffle
- 3. Bottleneck
- 4. Decoder. Per level: 1×1 ComplexConv inflate → FourierShuffle → cat skip → conv. The inflate restores the channel count the shuffle needs to fold back into space.
- 5. Invert. ifft2(x).real, crop the padding, then a single real Conv2d to the class logits. That final conv is the only layer that ever sees pixels.

FUNET: FourierUnet with Spectral pooling



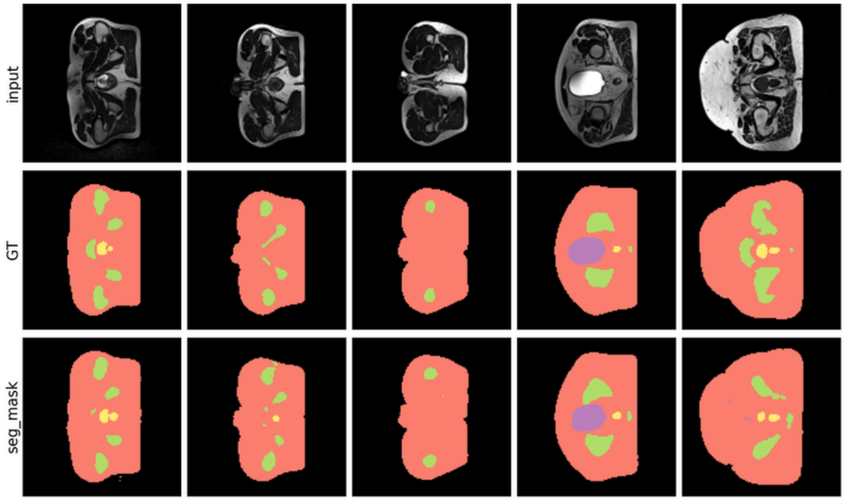
What is Spectral Pooling

Spectral pooling downsamples by cropping the spectrum instead of the image: fftshift to bring DC to the centre, slice out the central low-frequency block, ifftshift back to the layout the next layer expects.

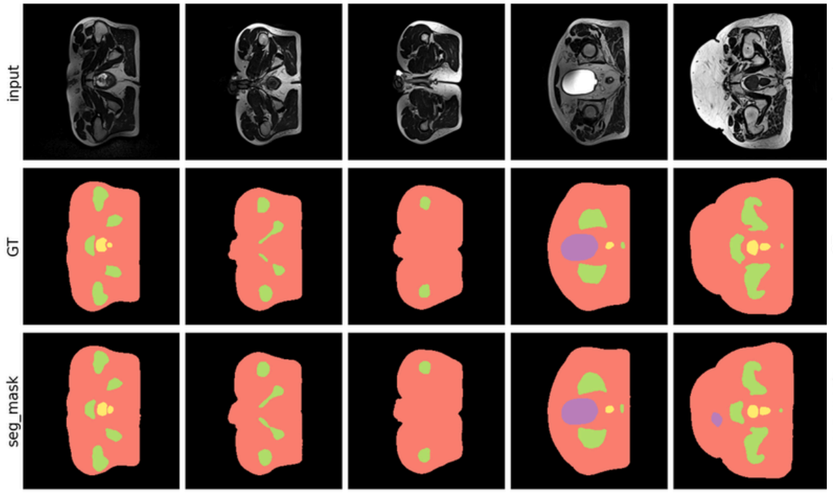
The following are experiments done on Ade20K dataset

Model	Skips	fg Dice	Train Dice	Params
U-Net (base 48)	**all four**	**0.1633**	**0.547**	**7.392M**
U-Net (base 48)	none	0.1395	0.49	7.392M
FourierUNet + spectral pool	all four	0.1105	0.308	7.267M
FourierUNet + spectral pool	none	0.0789	0.247	7.267M

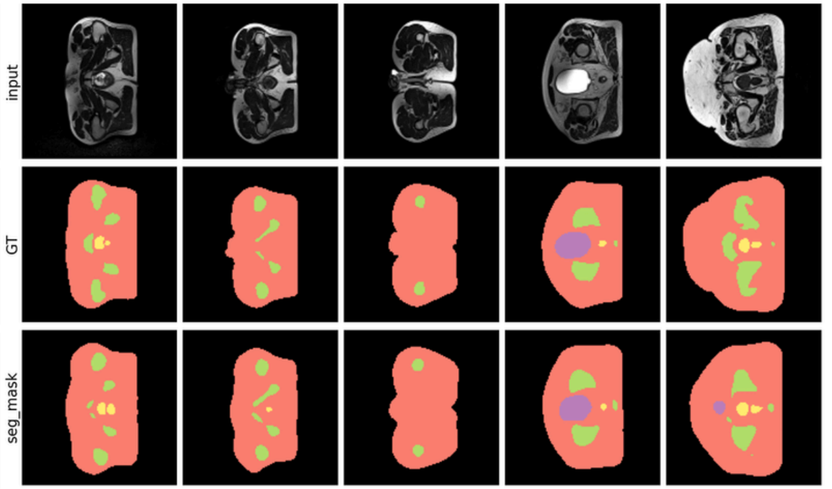
FUNET for Medical Segmentation



FourierUnet with Spectral pooling



UNET



FourierUnet

Model	fg Dice	Params
U-Net	0.827	1.95M
FourierUnet + Spectral pooling	0.717	3.60M
FourierUnet	0.712	6.35M

What do you notice?
Any differences?