
GEO FOR GOOD
SUMMIT 2019

Google

#GeoForGood19

Neural Segmentation

For Remote Sensing

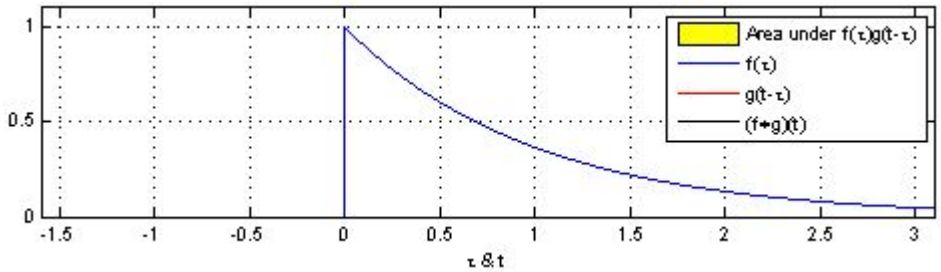
Chris Brown / 2019

Convolutional Neural Nets

Convolution

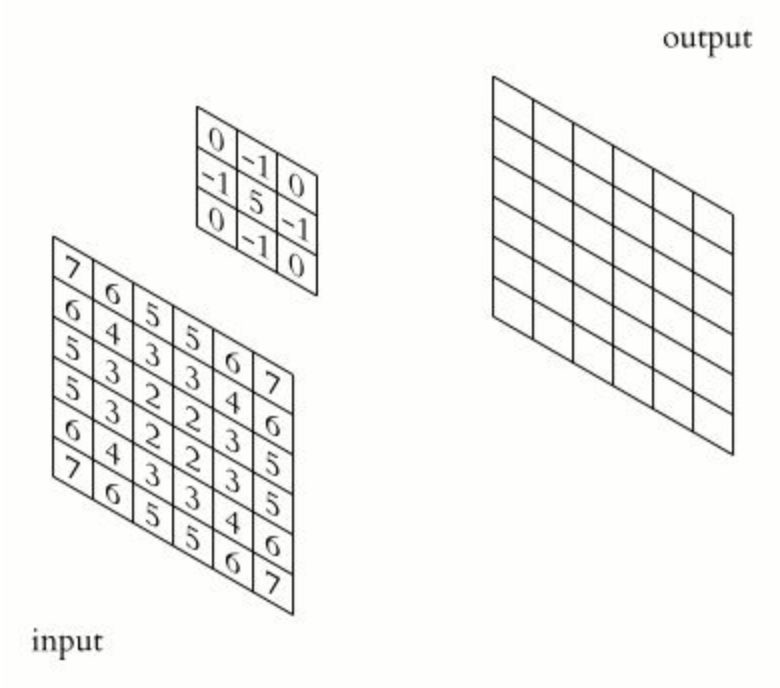
$$f(x) * g(x) = \int_{-\infty}^{\infty} f(\tau) \cdot g(x-\tau)\, d\tau$$

$$f(x,y) * g(x,y) = \int_{\tau_1=-\infty}^{\infty} \int_{\tau_2=-\infty}^{\infty} f(\tau_1,\tau_2) \cdot g(x-\tau_1, y-\tau_2)\, d\tau_1\, d\tau_2$$



https://en.wikipedia.org/wiki/Convolution#/media/File:Convolution_of_spiky_function_with_box2.gif

Discrete Convolution in 2D



https://commons.wikimedia.org/wiki/File:3D_Convolution_Animation.gif

Discrete 2D Convolution as a Feature Detector

$\mathbf{g}(x,y)=$

-1	1	1
-1	1	-1
1	-1	1

$\mathbf{f}(x,y)=$

1	0	0	1	0
0	1	1	1	1
0	1	0	1	1
1	0	1	1	0
1	0	1	1	0

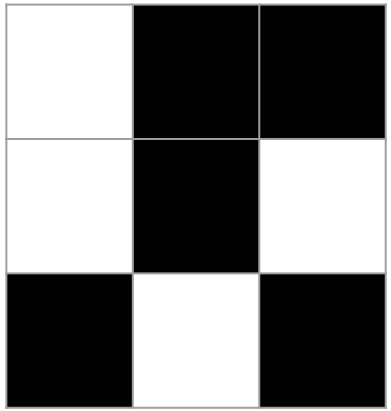
Discrete 2D Convolution as a Feature Detector

$\mathbf{g}(x,y)*\mathbf{f}(x,y)=$

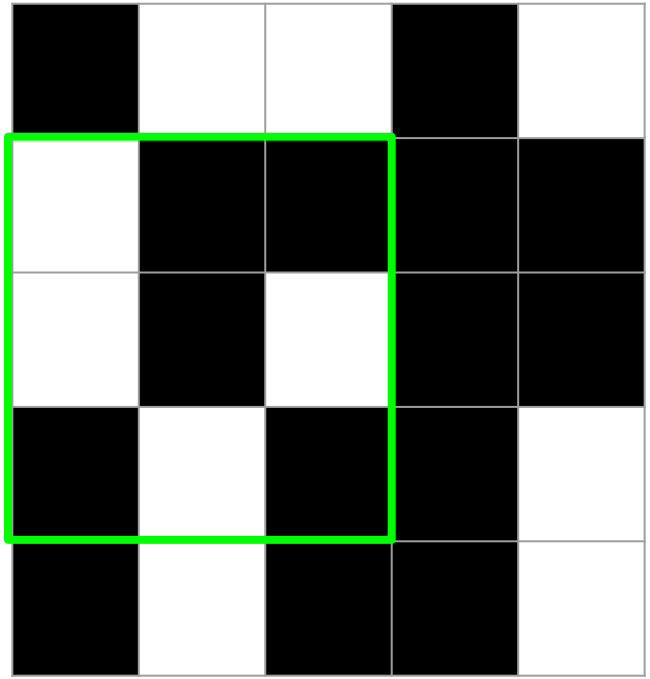
2	-1	0	2	-1
1	-2	2	0	1
-1	5	-1	1	1
1	1	0	2	0
2	-2	2	0	-2

Discrete 2D Convolution as a Feature Detector

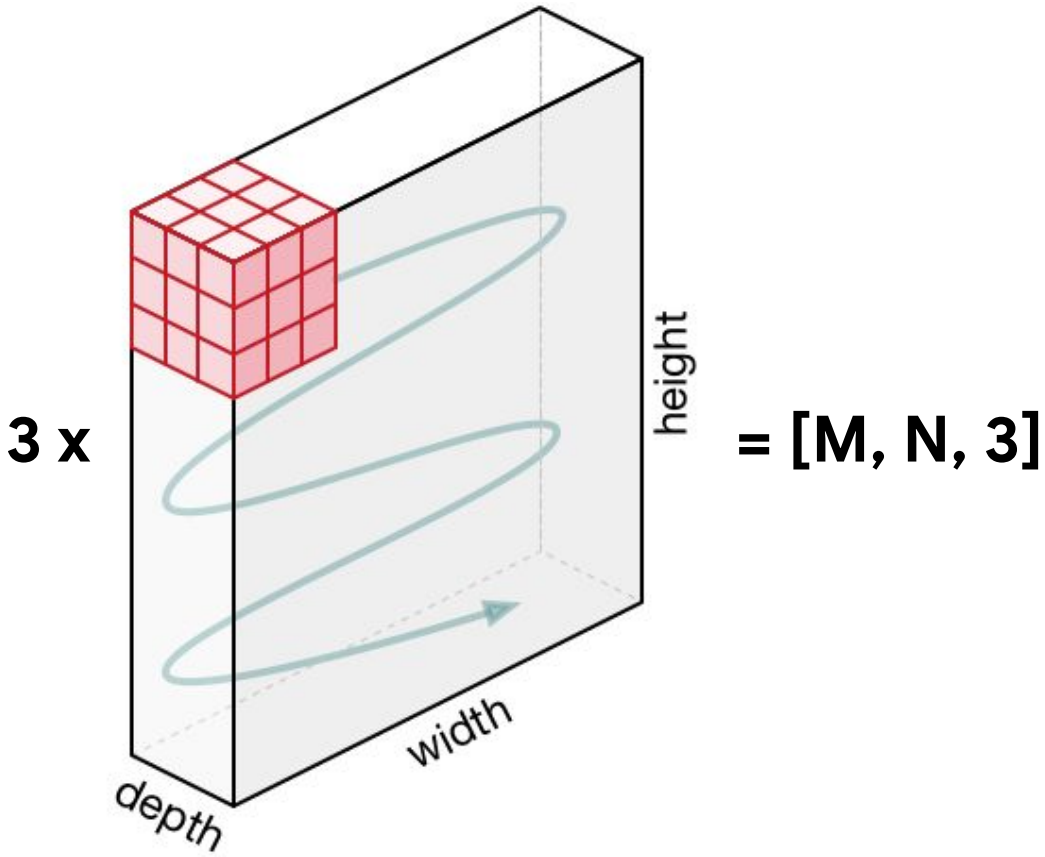
g(x,y)=



f(x,y)=



Discrete 3D Convolution



Hierarchical Feature Detection

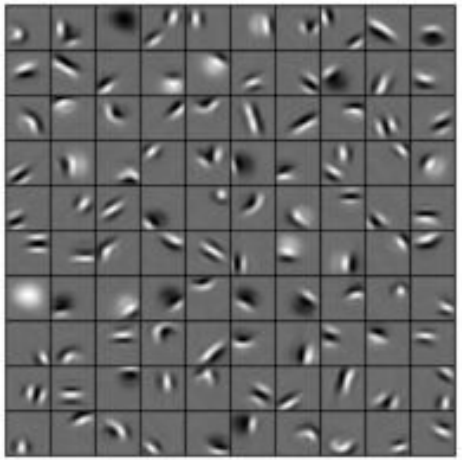
$g_1 * (g_0(x,y) * f(x,y)) =$

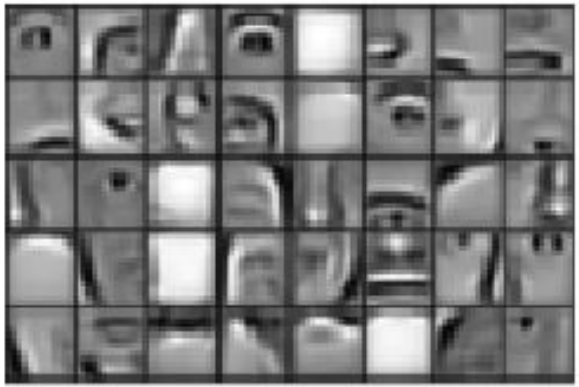
?	?	?
?	?	?
?	?	?

*

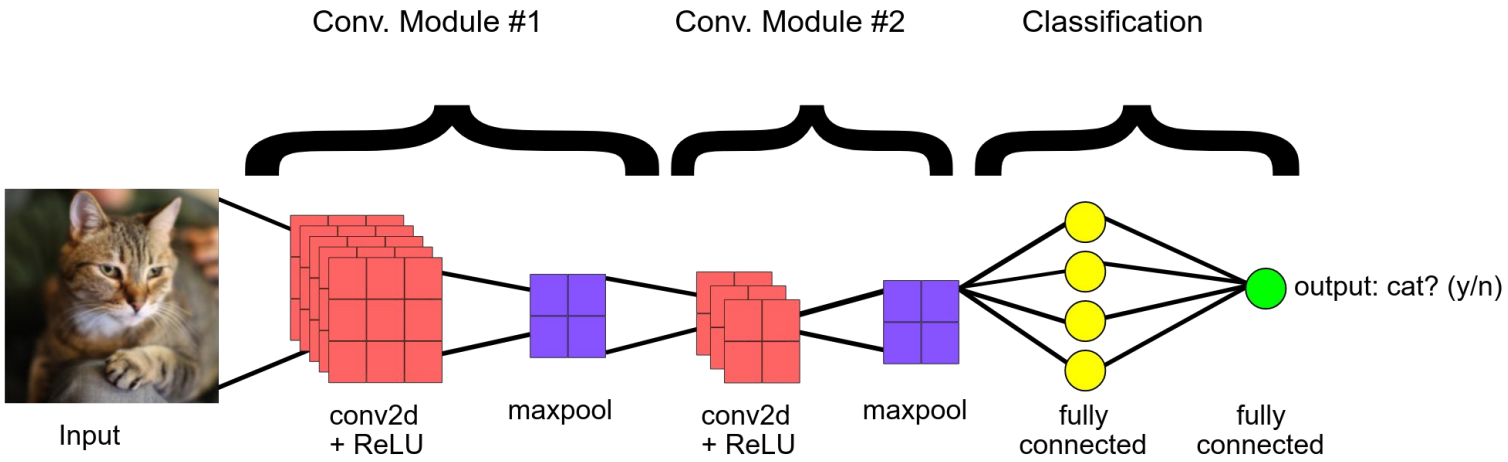
2	-1	0	2	-1
1	-2	2	0	1
-1	5	-1	1	1
1	1	0	2	0
2	-2	2	0	-2

Hierarchical Feature Detection



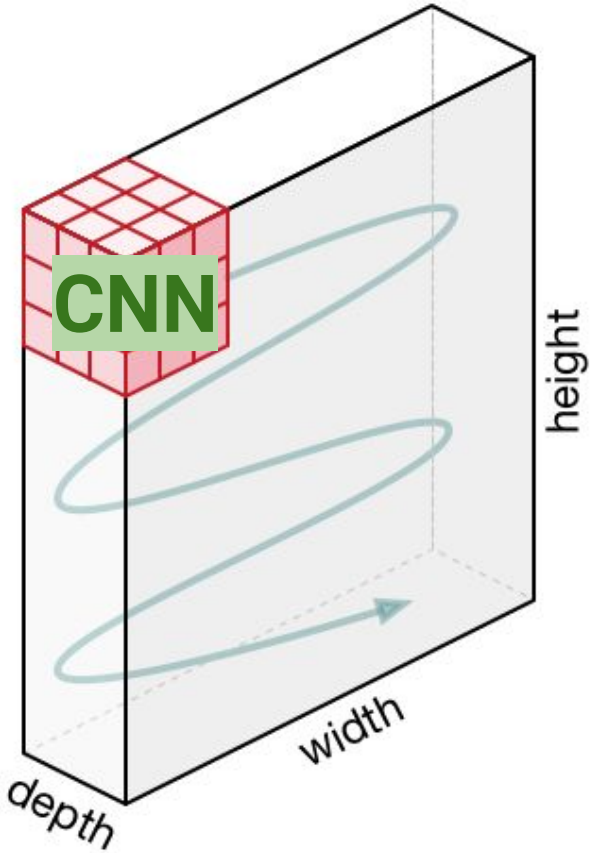


Convolutional Neural Nets



Fully Convolutional Neural Nets

First, a bad idea:



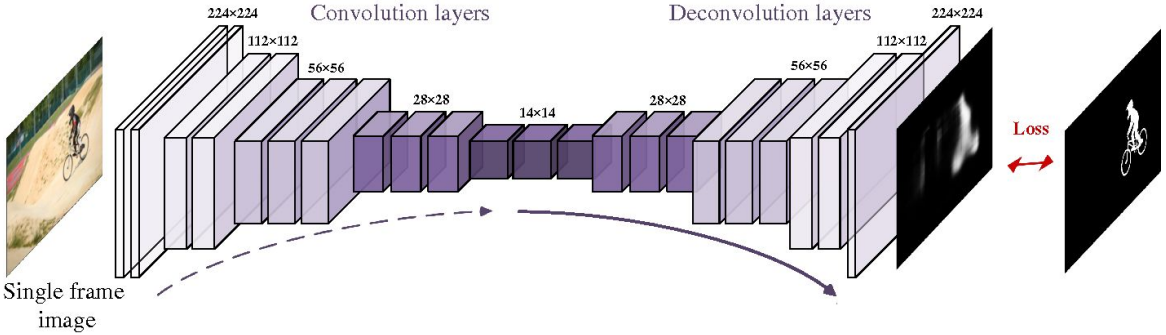
Remember this?

Why don't we do this with a convolution!

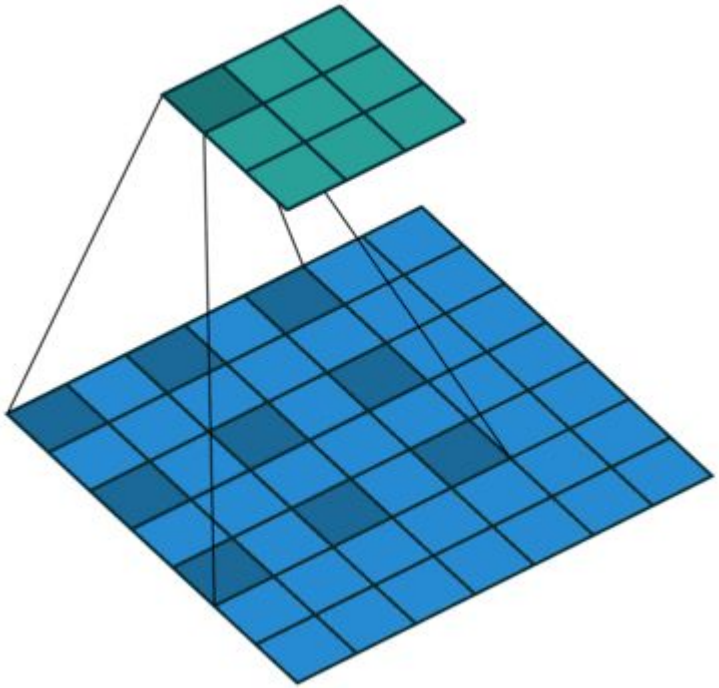
https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf

Fully Convolutional Neural Nets

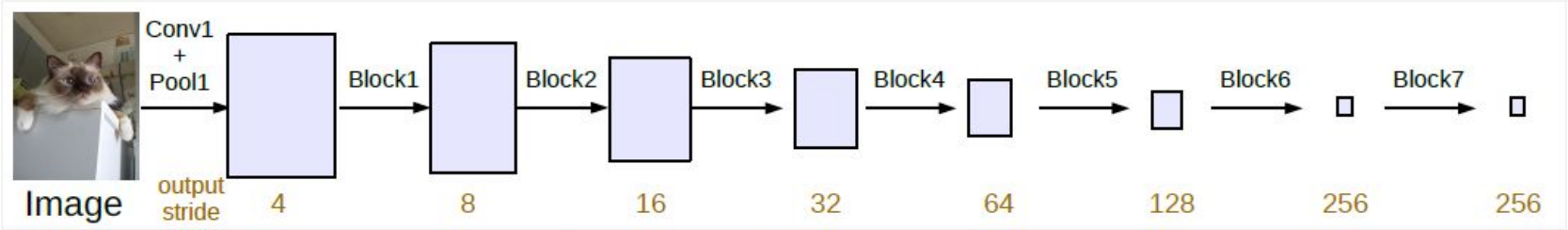
- Convolutional Neural Nets (CNNs) produce 1 prediction for NxM (fixed) inputs
- FCNNs can predict 1x1, NxM or virtually any output size for NxM inputs.
- For narrow receptive fields, they're a computationally efficient way to create things like per-pixel dense networks by way of NxM depthwise convolutions.



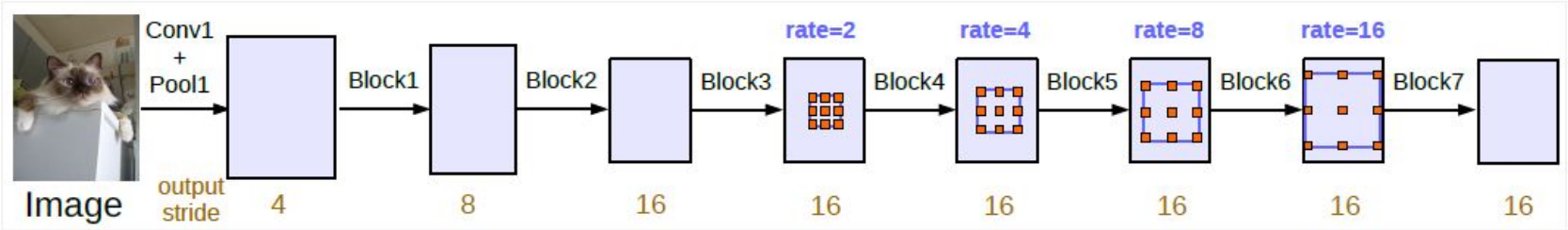
Atrous Convolution



Atrous Convolution



(a) Going deeper without atrous convolution.



(b) Going deeper with atrous convolution. Atrous convolution with $rate > 1$ is applied after block3 when $output\_stride = 16$.

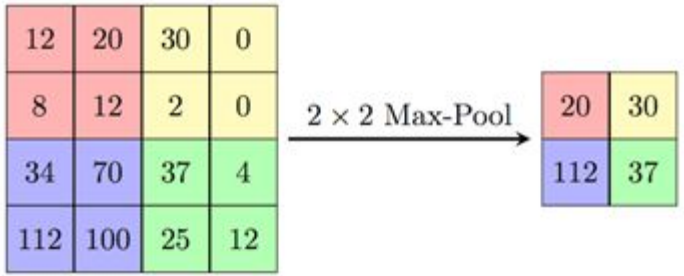
We can train a FCNN with tile-level labels!

(no mean collapse!)

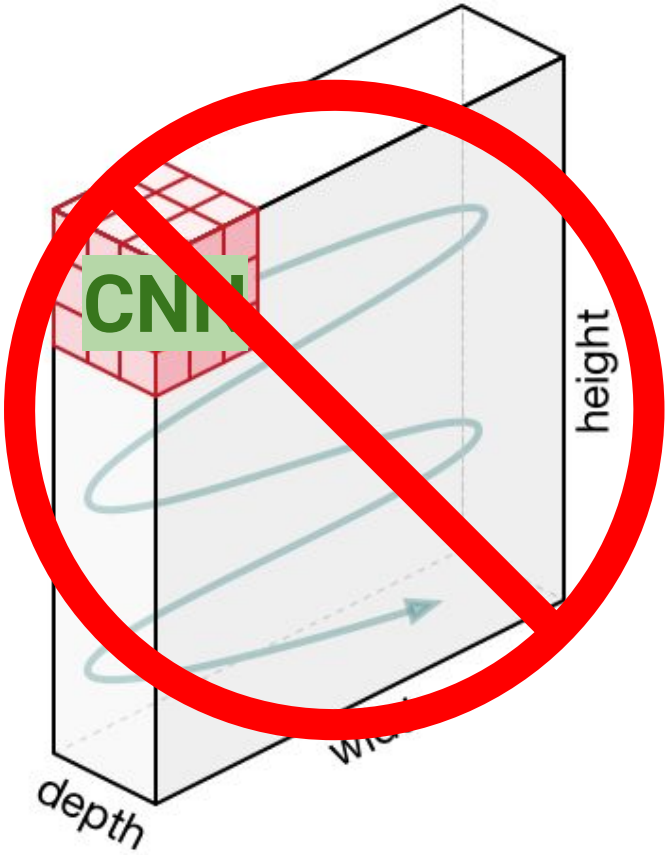
Atrous Convolution

Atrous convolution captures features at different scales without loss of information.

Technically we wouldn't even need encoder->decoder residual connections to reintroduce spatial information.



So back here...



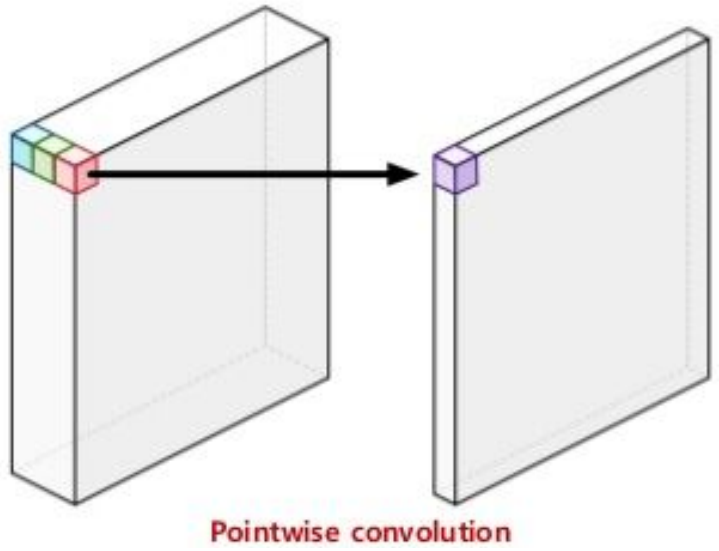
CNN's lose spatial information (no residuals, no atrous convolutions)

Computationally expensive! If you have point samples, transform to FCNN.

Practical Considerations

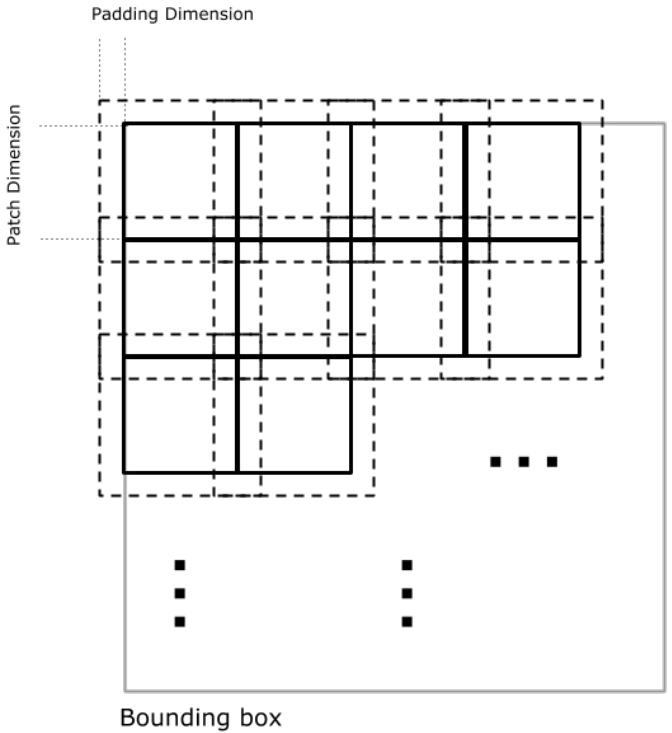
Per-Pixel Dense Nets Using 1x1 Convolutions

- Rather than splitting multi-spectral patches of pixels into vectors and applying them as features to a “dense net,” we can accomplish the same thing without the splitting.
- Has the advantage that convolutions on 2D patches of pixels is a *highly* optimized operation on GPUs and TPUs.
- Easy to translate a series of “dense layers” to a series of 1x1 depthwise convolutions.
- (this is technically a FCNN!)



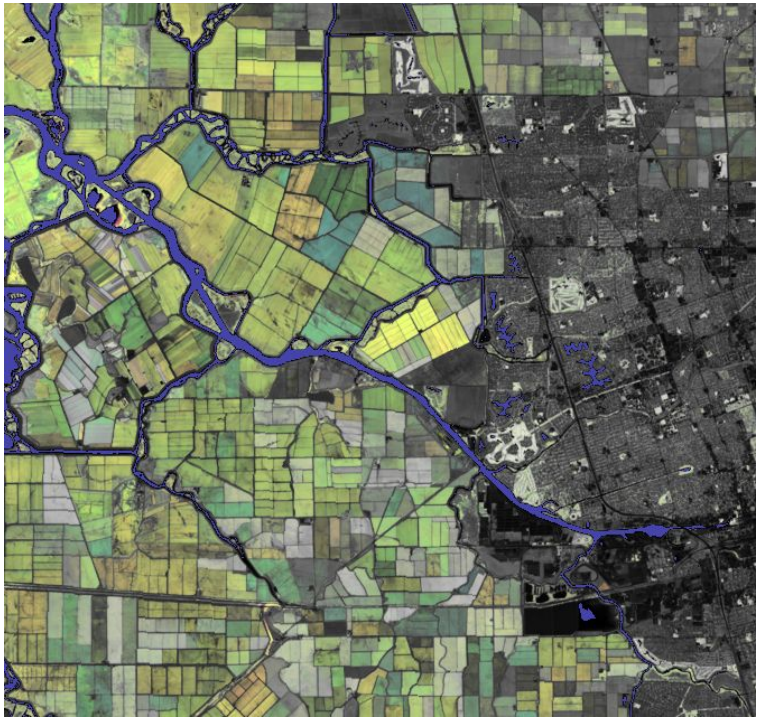
Overtiling

- Unlike FCNNs for camera imagery, we usually want to tile predictions to cover large projected regions (i.e. Landsat scene sized).
- We run into border artifacts because some pixels have a receptive field extending outside the tile, so when we produce a tiling we can “overtile” (known as kernelDimensions in EE and intended for a different purpose) and crop the predictions so that all pixels in the final predicted mosaic have a fully-specified rf..



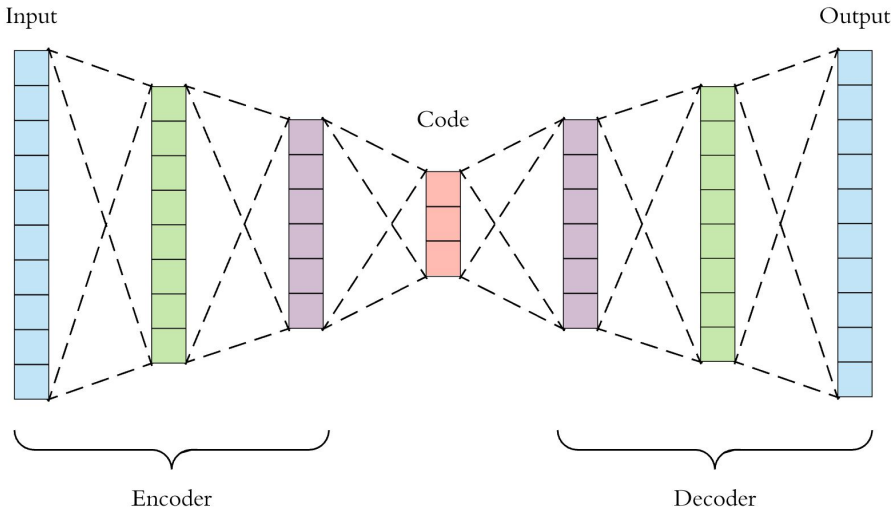
Receptive Field Size

- The lower the resolution, the less relevant an output's RF becomes to it's final predicted value.
- Many big FCNNs have huge RFs, but for tasks like LULC, we very rarely need 300x300 / 512x512 labels and can get state-of-the-art accuracy using 128x128.
- Intuition is that, i.e., we don't need to see the objects outside of a forest to know what it is; unlike a cat's tail that requires the presence of kitty fase :3.
- Objects are all at similar scales.



Dealing With Little Training Data

- In the cases where we have no model from which to transfer-learn, we're not completely out of luck.
- Auto-encoders provide a means to learn the color -> shape -> texture *CNN hierarchy in a *completely unsupervised way*. From this we can transfer learn
- We can push this even further to extract "stronger" features by making our AE "denoising" and "sparse".
- EE's catalog is basically an AE paradise.



What Am I Missing?

- Finding objects at high resolution!
- Cars, oil tanks, etc…
- There's plenty of information floating around on this, but much high-res satellite imagery isn't public, is often RGB, collected with a low temporal frequency, and is perfectly suited for more standard computer vision models.