Encoding and Decoding

D.A. Forsyth

University of Illinois at Urbana Champaign

Image representations - Encoding

- Idea:
  - Filter banks + ReLU yield scores for many different patterns

- Idea:
  - You can compose this, so patterns of patterns of …
  - Result: a code that describes the image

- Idea:
  - if filters are well chosen, you could use the representation to:
    - denoise images
    - find edges
    - find interest points
    - classify images…

BUT what filters/patterns should we use?

Many different pattern detectors

- Yield an overcomplete representation
  - redundant information
  - one local image patch is scored against many different patterns

- downside:
  - representation is larger
- upside:
  - you can recover image despite some errors in representation

Reconstruction

- Find the image that produces the representation closest to the one observed

- Now imagine input image is noisy:
  - The reconstruction might not be
    - (with some care and some luck)

- Idea:
  - Build device that can accept noisy image, produce clean
  - Denoising autoencoder

Helps choose filters/patterns – the ones that denoise

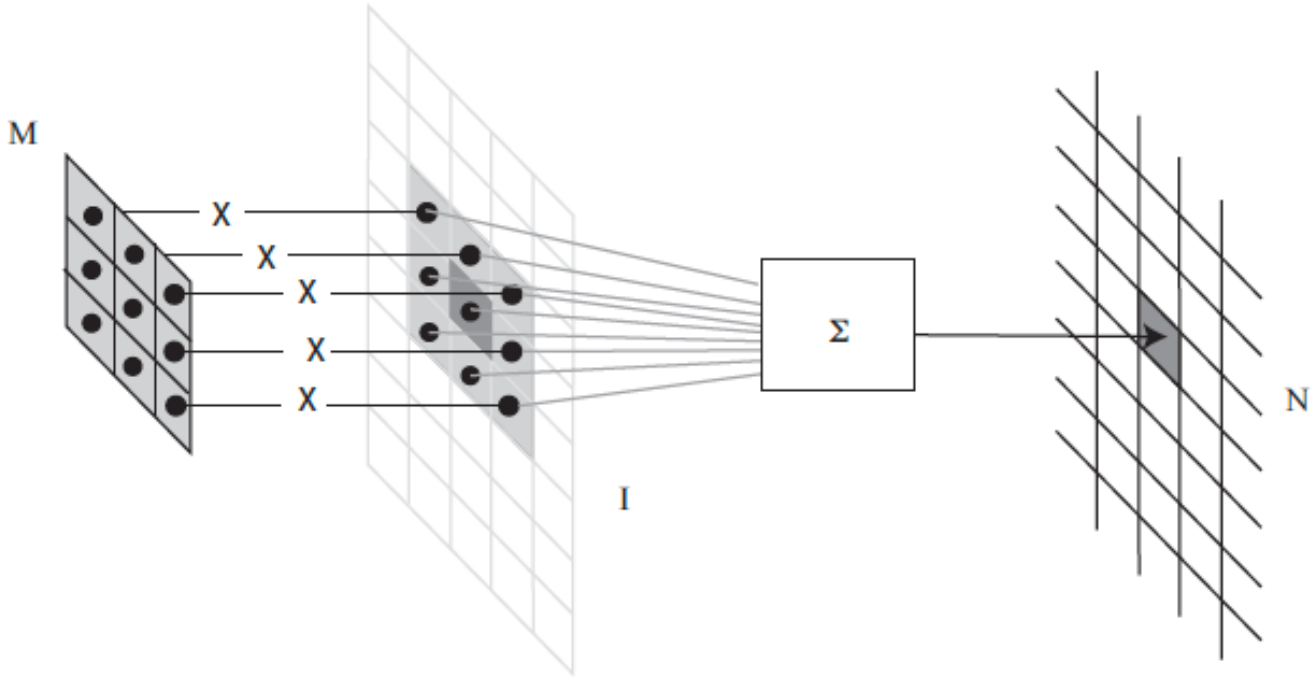
Images from codes - Decoding

- Idea:
  - Code likely (roughly) invertible
    - Reconstruct filter responses from ReLU outputs
    - Reconstruct image from responses (conv. Theorem)
- Decode by:
  - Place down instances of detected patterns
  - This is filtering
  - Use a ReLU to prevent negative values accumulating

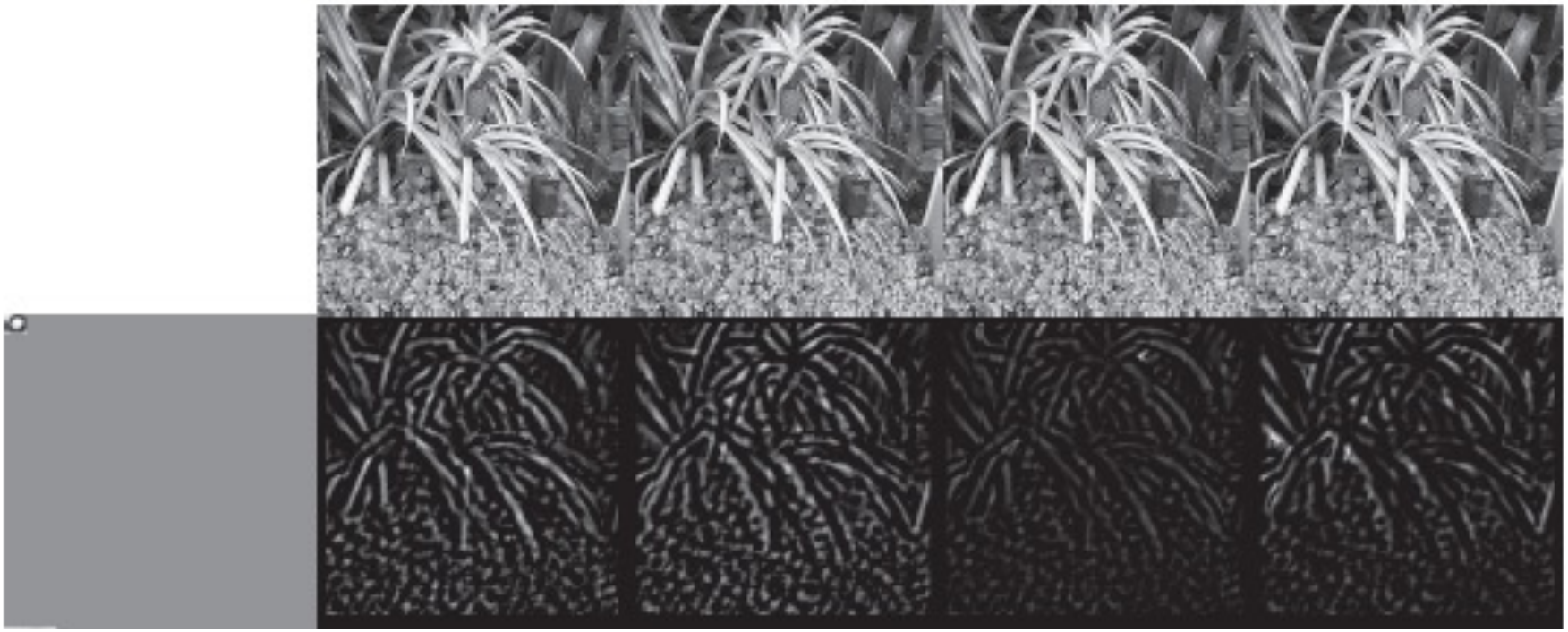
Learned Image Representations

- Idea:
  - It might be possible to produce an image representation from a lot of filters
  - AND reconstruct the image from the representation using a lot more filters

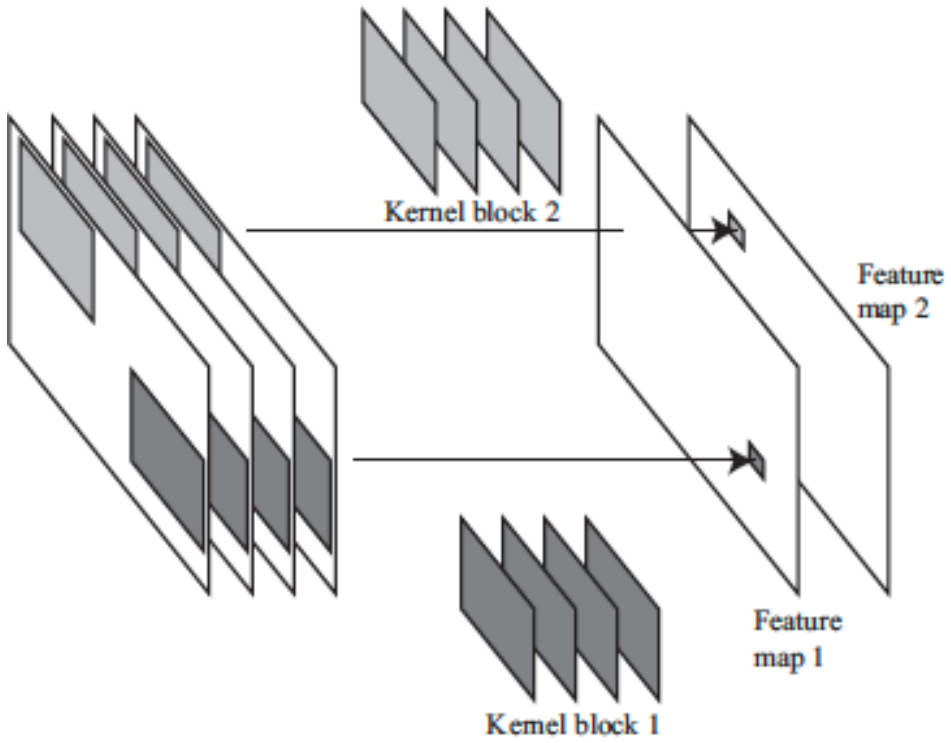
Recall convolution

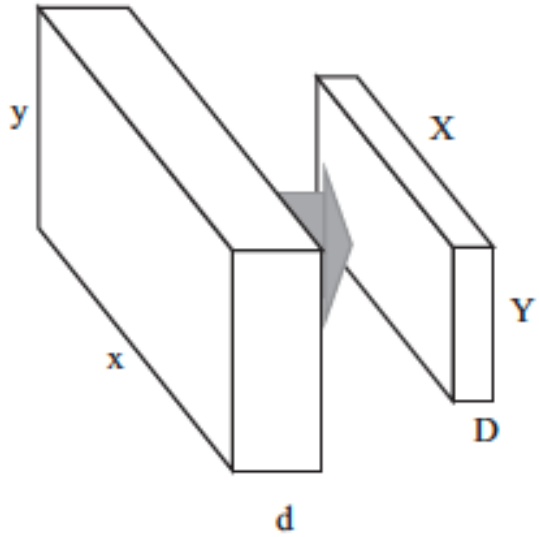


Recall Convolution + ReLU = pattern detector

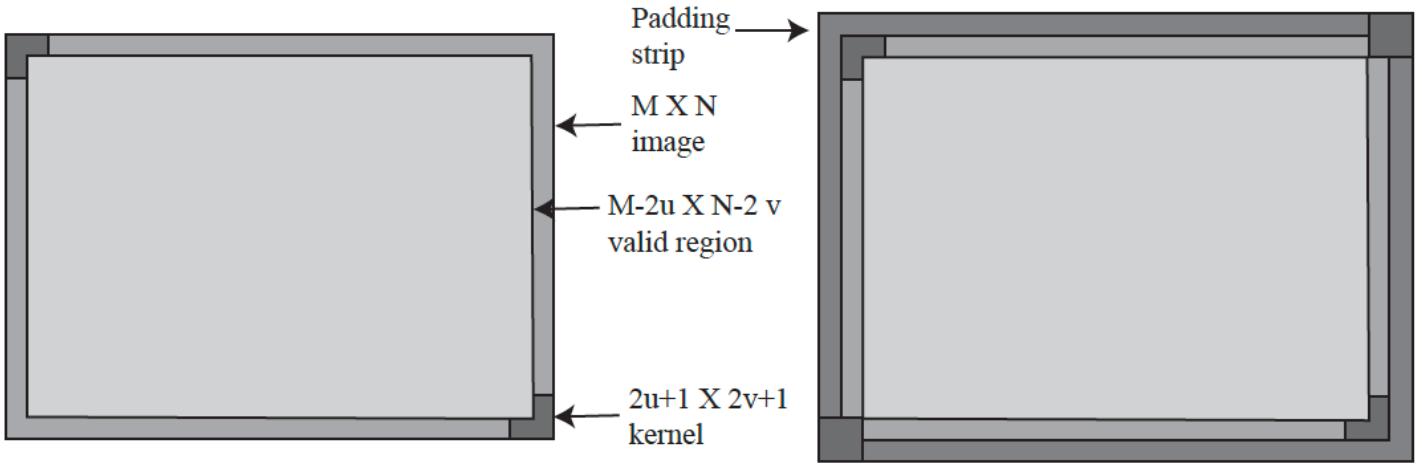


Recall Multi-channel Convolution





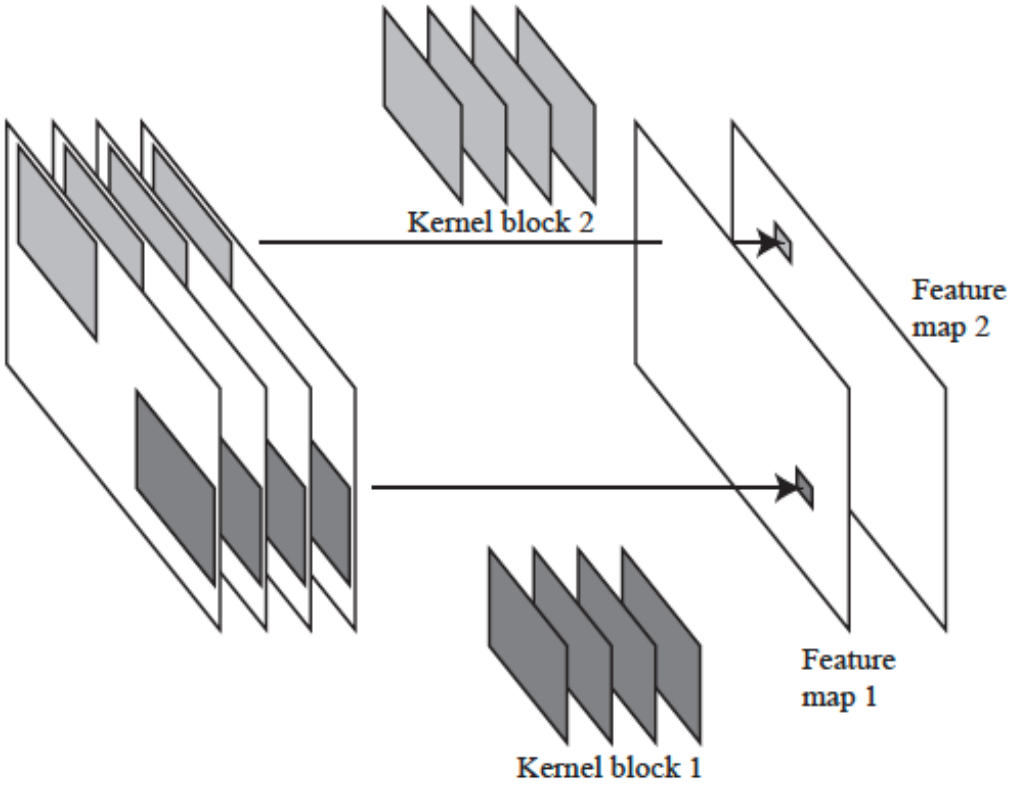
Recall padding

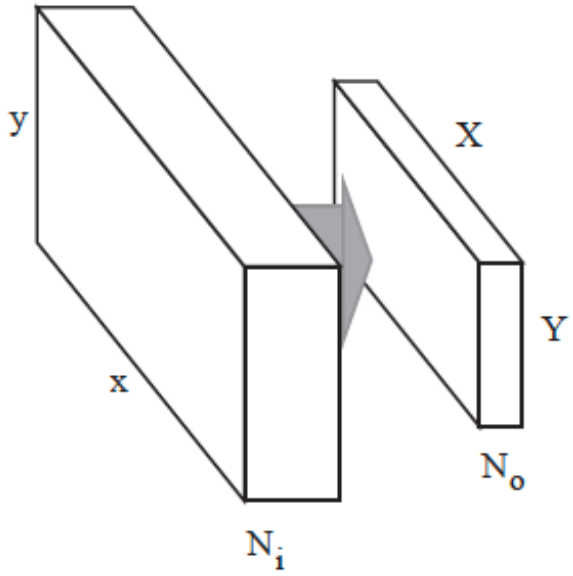


Stride

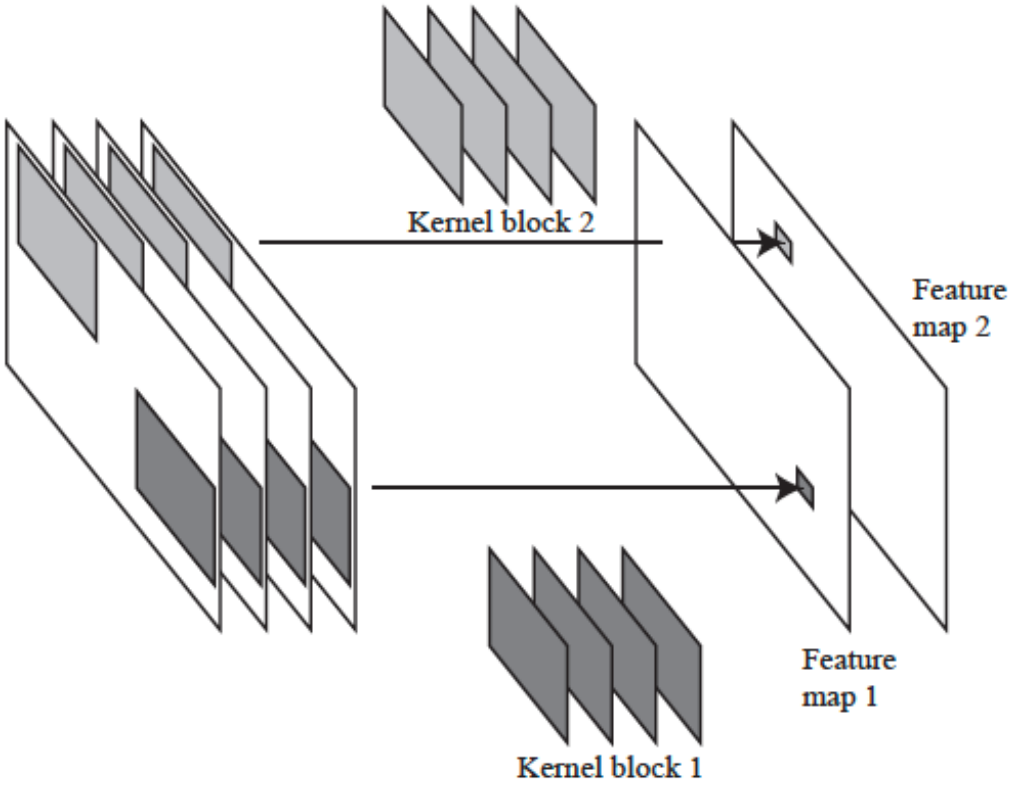
- How far across/down to go to the next pixel?
- Stride 1: what we're used to
- Stride 2: place the kernel on every second pixel

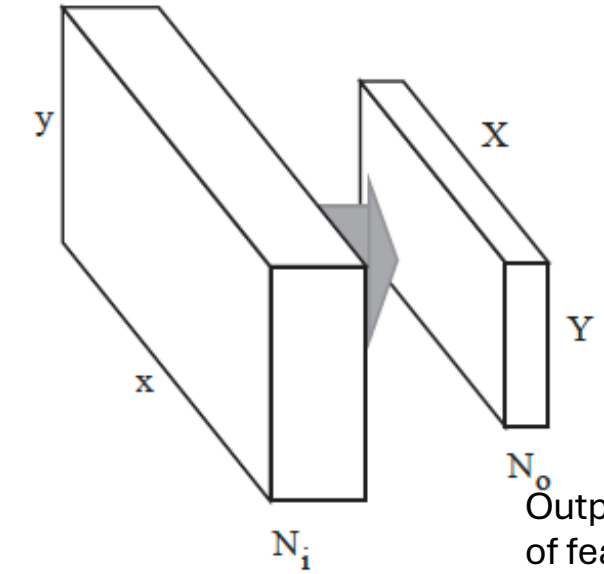
Convolutional Layers





Convolutional Layers

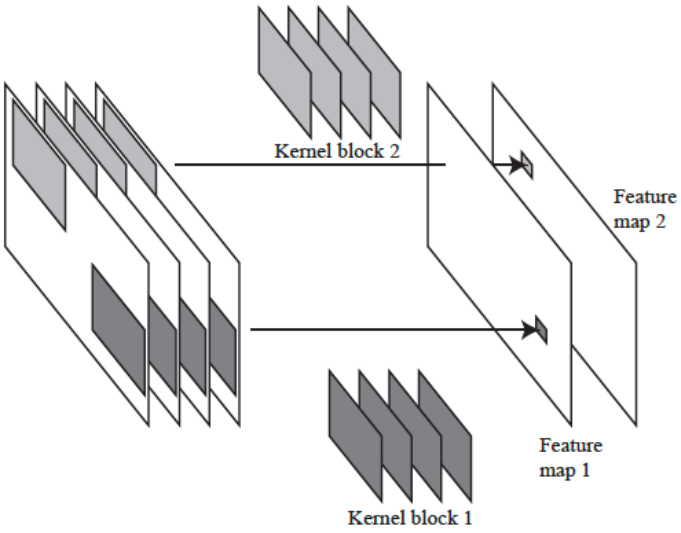


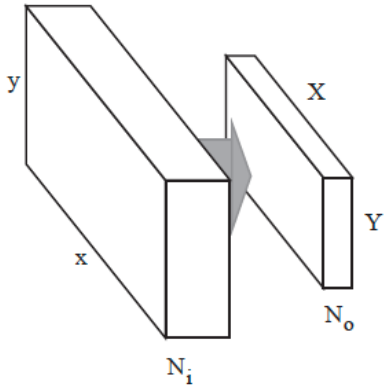


Input number of features

Output number of features

Convolutional Layers





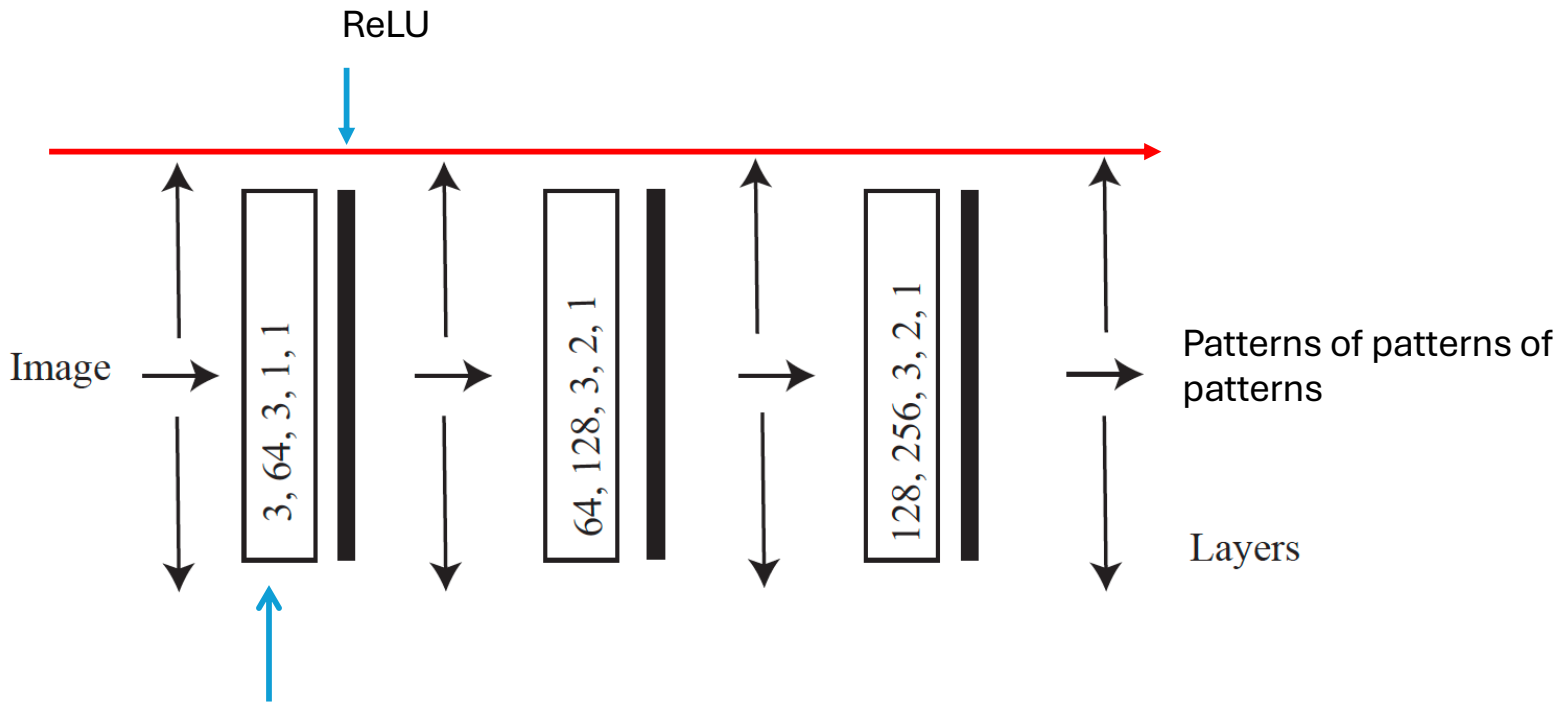
Output size will be determined by:

- input size,
- kernel size,
- padding,
- stride,

ReLU operates on data block

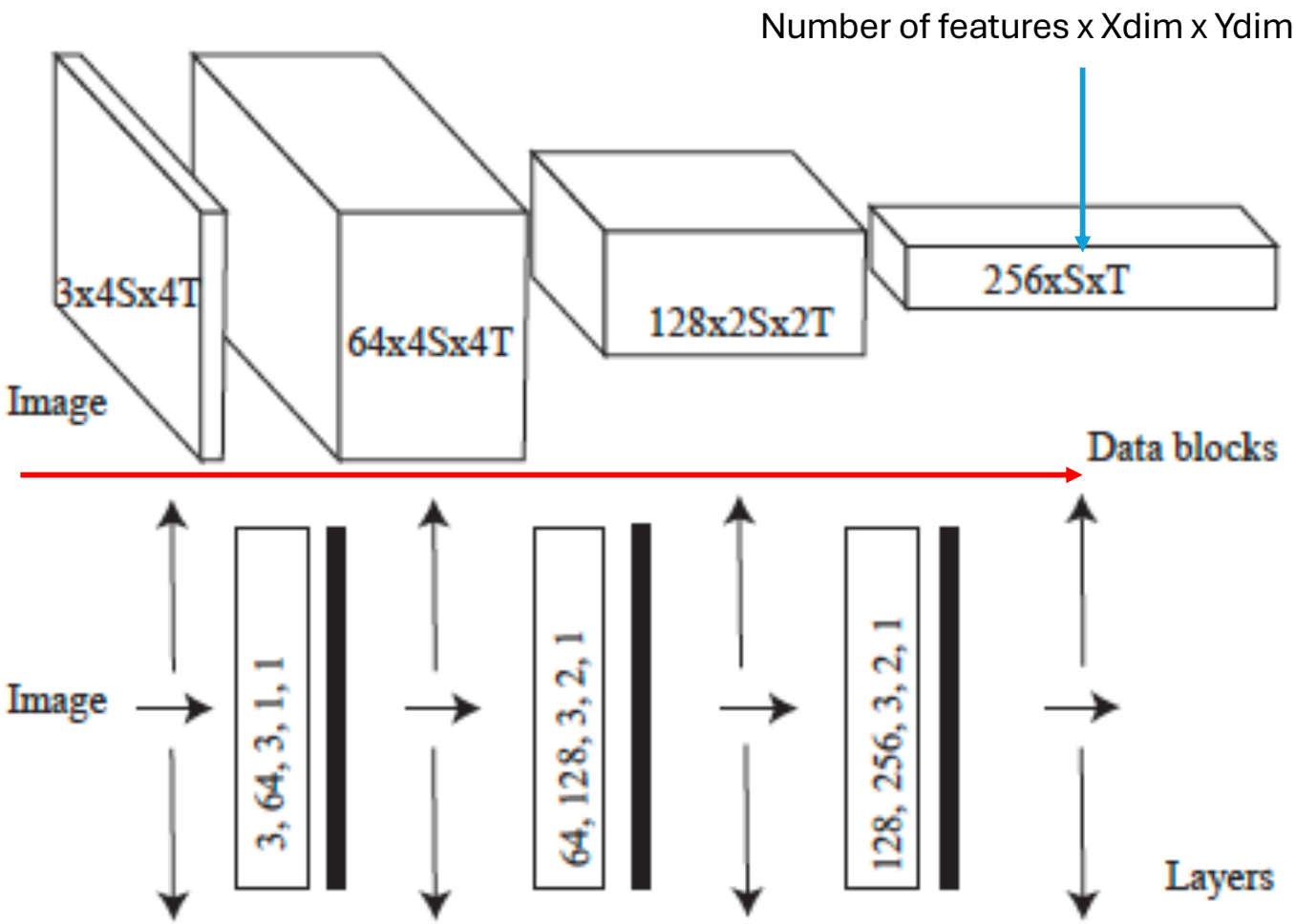
- Trivially – just ReLU at each location

A very simple encoder



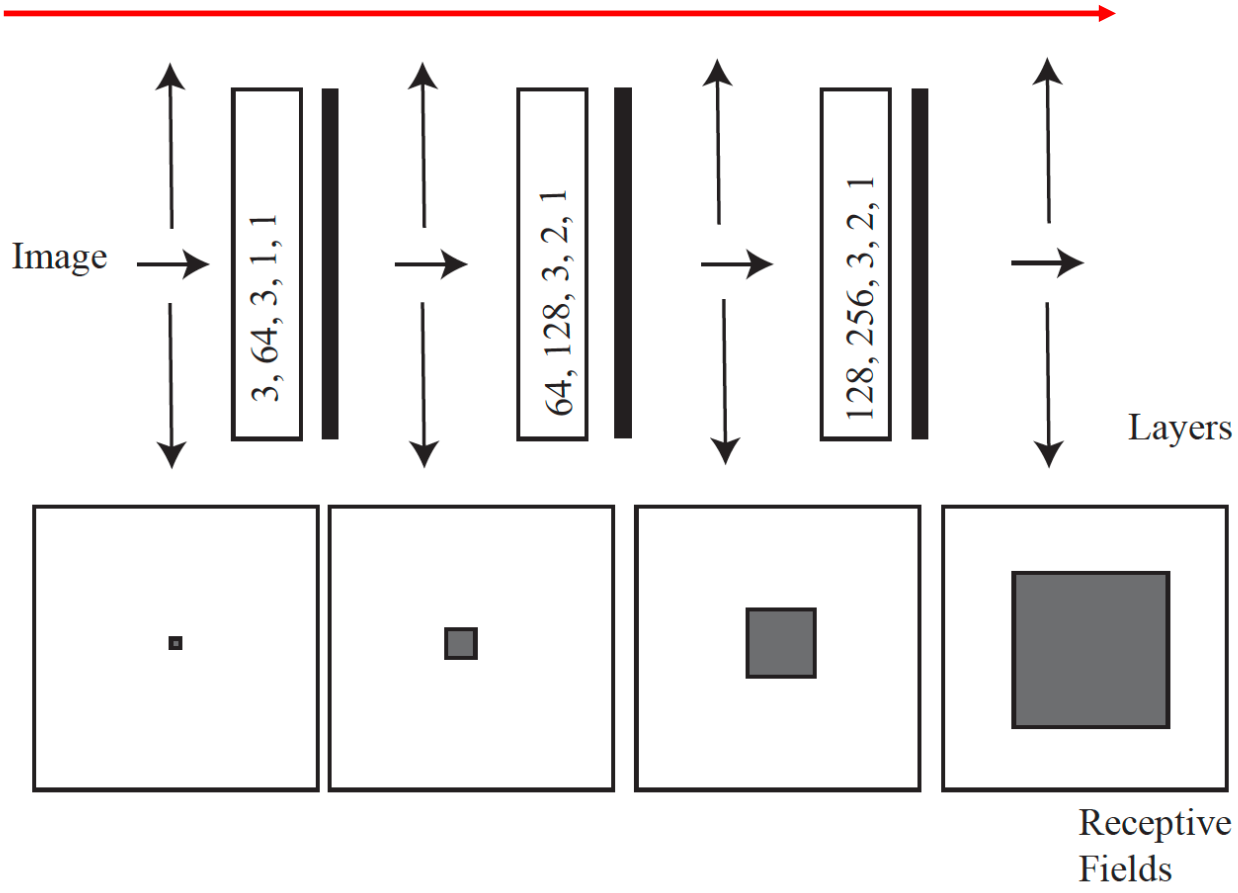
Input number of features, output number of features, kernel size, padding, stride

As data blocks



Receptive fields

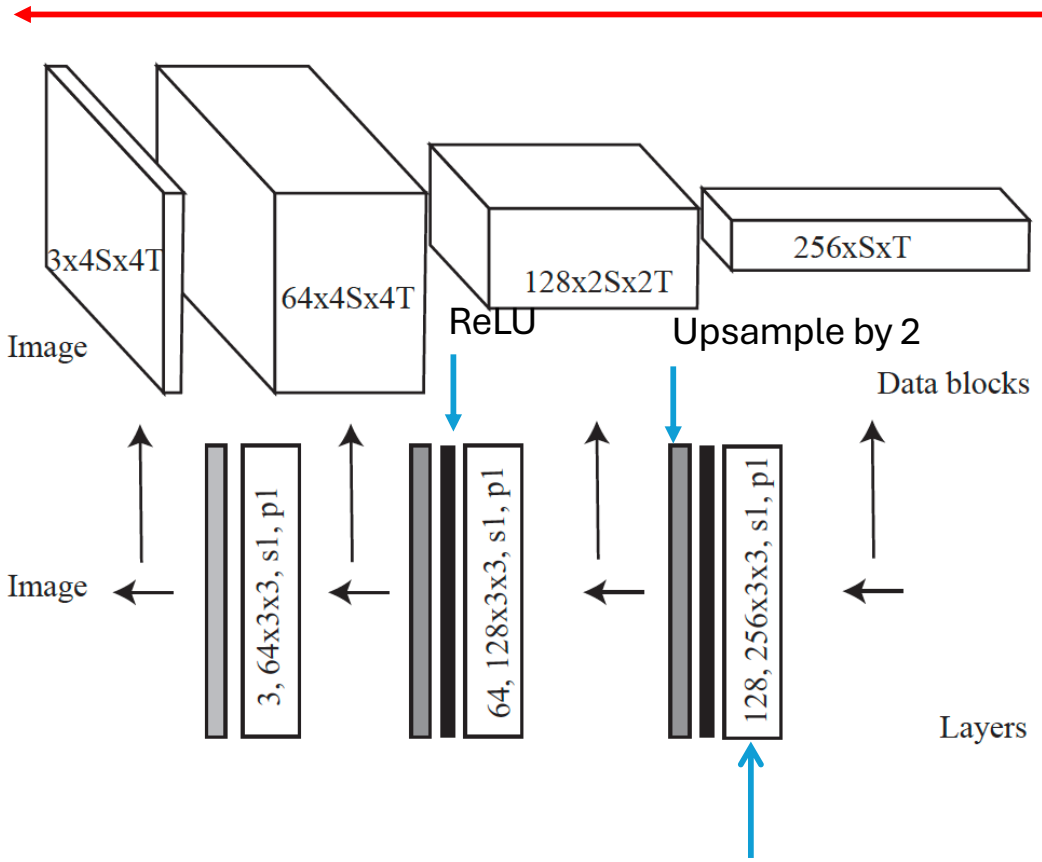
- Support for a value in the feature map



Decoding

- Want:
  - map rep’n (patterns of patterns of patterns…) to image
- Have:
  - If rep’n is filter outputs, convolution is enough
  - Rep’n is spatially smaller than image
- Idea:
  - Filter+ReLU+upsample on occasion might do it

A decoder



Patterns of patterns of patterns

Input number of features, output number of features, kernel size, padding, stride

Big idea

- With the right choice of filters
  - a decoder could reconstruct an image from an encoders rep'n
- The rep'n is overcomplete
  - "sees" the image at many scales
  - so the pair should be able to denoise
- But what is the right choice of filters?

Choose filters/patterns that denoise

Things to think about...

17.1. You are given a filtered image. How would you recover the original? what might go wrong?

17.2. A multichannel convolution with stride 1, kernel size $2d + 1$, padding d and N_o filters accepts an $N_i \times X \times Y$ block. How big is the block that comes out?

17.3. A multichannel convolution with stride 2, kernel size $2d+1$, padding d and N_o filters accepts an $N_i \times 2X \times 2Y$ block. How big is the block that comes out?

17.4. A multichannel convolution with stride 1, kernel size $2d+1$, padding 0 and N_o filters accepts an $N_i \times X \times Y$ block. How big is the block that comes out?

17.5. A convolutional layer with stride 1 and kernel size $2d + 1$ is followed by a second convolutional layer with stride 1 and kernel size $2d+1$. How big is the receptive field for a feature in the second layer?