# U-Nets

D.A. Forsyth

University of Illinois at Urbana Champaign

# Last blocks

- Build an encoder and a decoder to:
  - Accept noisy image, produce clean version
- By:
  - Constructing loss
  - Applying SGD to get minimal loss on training data
- Using various tricks to get good behavior

# Change what the decoder does…

- Regression – predict “image like thing” from image
- An “image like” thing
  - may have the same resolution as the image
  - continuous (not categorical)
  - lots of examples
  - can be predicted from an image (but how do we know?)
- Examples:
  - depth
  - normal
  - defogged image
  - superresolution
  - lots of others..

# This recipe isn’t universal …

- Generally, want cases where output
  - is image like
  - is determined by input
- For example, very hard to apply to colorization
  - there are many right outputs for the same input
- Mostly, quite hard to tell what will work

# New recipe, depth case

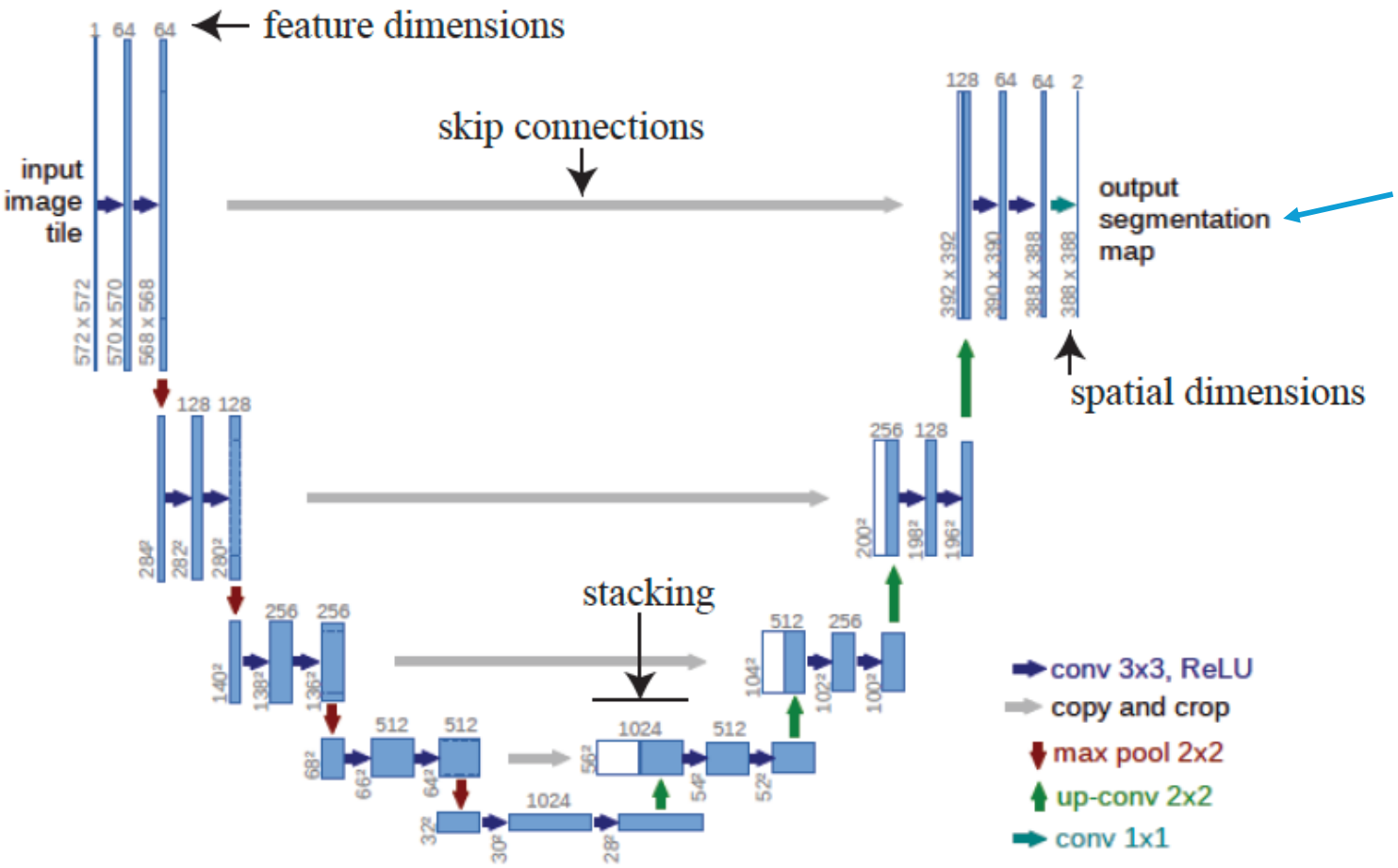
- Procedure:
  - find many training pairs (image, depth)
  - adjust filters so that
    - Decode(Encode(image)) is close to depth
  - on average, over pairs
  - hope that this generalizes to new images

- Result:
  - Single image depth predictor

# The U-Net

- Originally
  - a particular architecture
- Increasingly
  - an encoder followed by a decoder

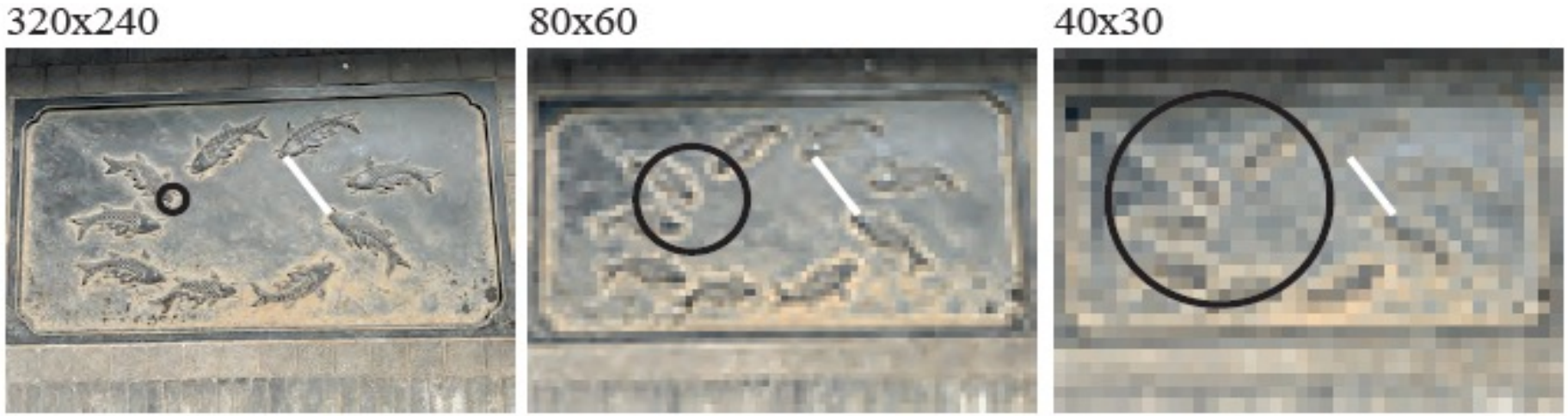
# The original U-net




This doesn't have to be a segmentation

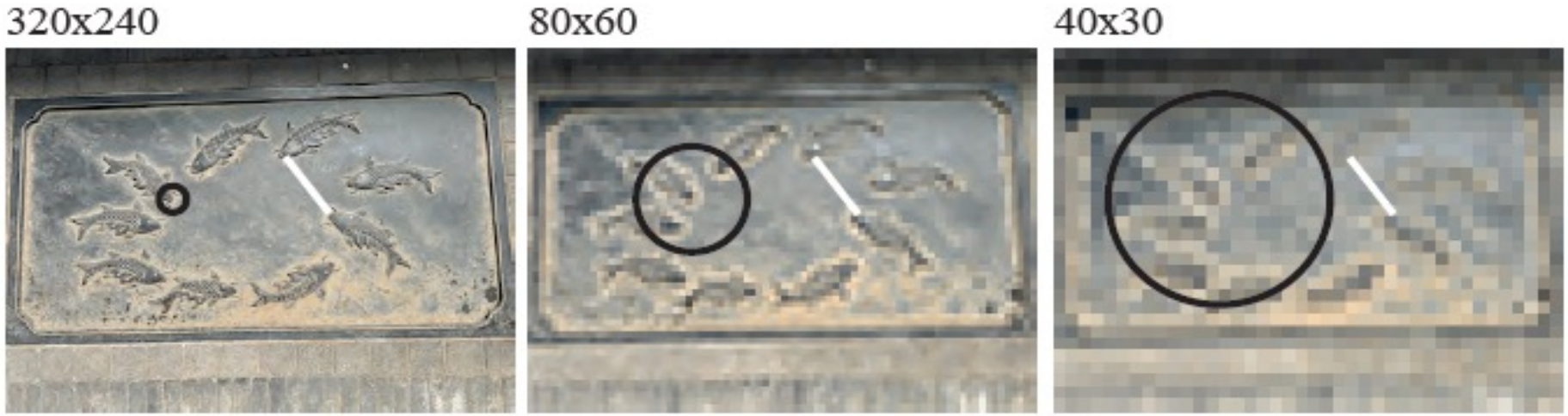
# This is fully convolutional

- Advantage:
  - you can train on one size, test on another
- Issue:
  - very odd spatial behavior is implicit



# This is fully convolutional

- Not every image regression network is FC
  - in some cases, scale matters a lot – fixed size images



# Things to think about...

**20.1.** Section 20.1.1 says: "The minimum size depends on the network (you need to ensure that the smallest feature block is big enough that the convolution will succeed)." How do you determine the minimum size from the network?