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

Now change what the decoder does…

An ”image like” thing

- may have the same resolution as the image
- continuous
- lots of examples
- can be predicted from an image (but how do we know?)

Examples:

- depth
- normal
- defogged image
- superresolution
- lots of others..

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

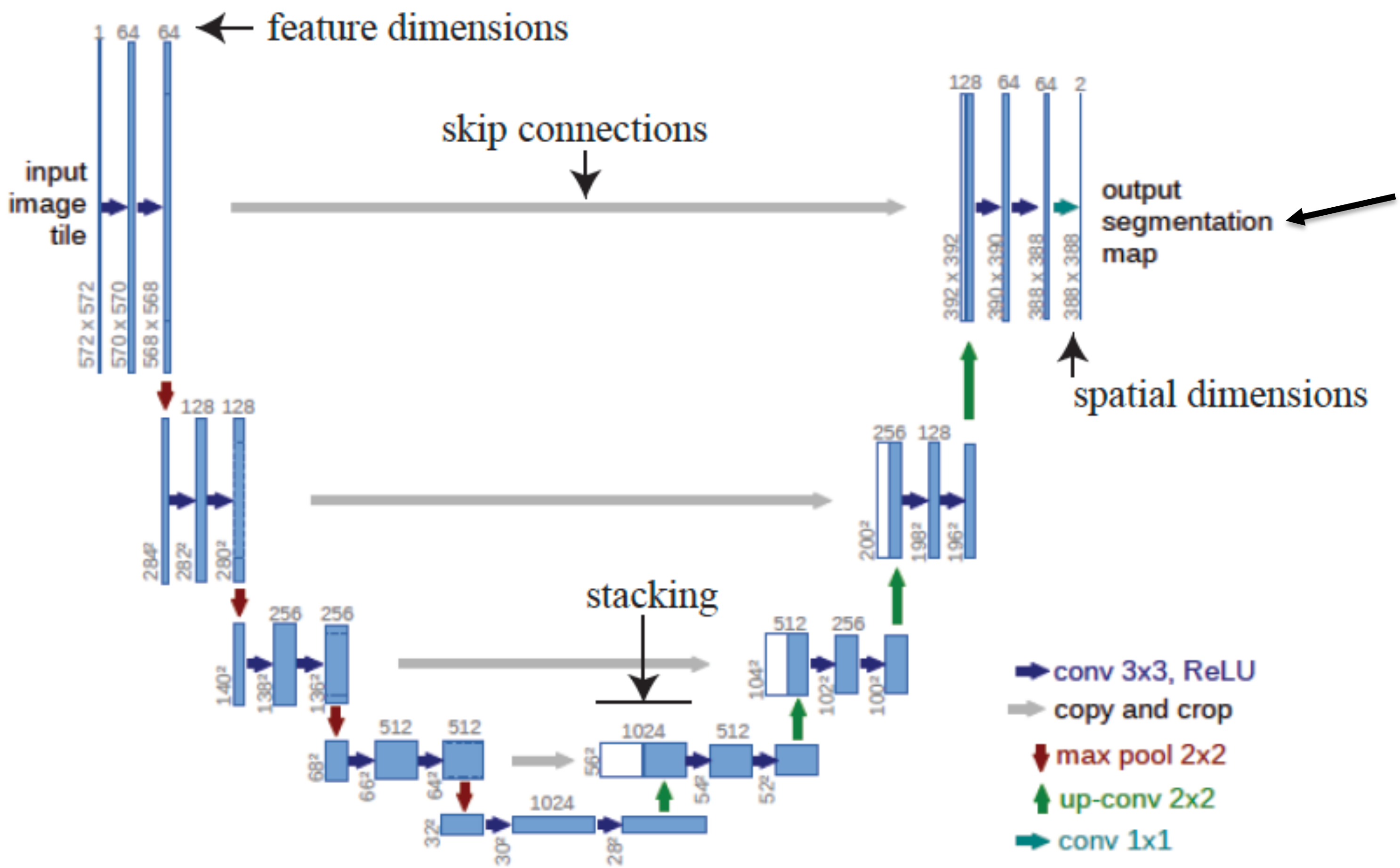
Originally

a particular architecture

Increasingly

an encoder followed by a decoder

The original U-net



This doesn’t have to be a segmentation

Depth from single image

Data example from NYUV2

What should a depth predictor predict?

Absolute depth:

Might be very difficult to get right

compare:

zoomed picture of a doll-house room

picture of real room

Relative depth:

depth up to per-image scale

more likely to get this right, but harder to use

Scale and error

Nearby objects are more interesting than distant ones

you collide with them first

Want smaller error in small depths than in large depths

squared error tends to yield small errors in large predictions

Relative error = error/(true value)

Want smaller relative error in small depths, too

What should a depth predictor predict?

Disparity = (1/depth)

tends to be dictated by the error in large depths. The *inverse depth*, sometimes known as *disparity* (Section 15.10) has a convenient property. Assume the true depth is d, so true disparity is $\delta = 1/d$. If the predictor predicts disparity with some error ϵ, then the corresponding depth is

$$\frac{1}{\delta + \epsilon} \approx d - \epsilon d^2$$

which means that a fixed error in disparity is a bigger error in large depth and a smaller error in small depth.

Depth predictors mostly predict

$$a_d \delta + b_d$$

where a_d and b_d are constants, determined *per image*. At training time, these

Losses

Depth prediction losses tend to be a mixture of L_1 and L_2 norms on the prediction

Depth predictors work really well

Evaluating depth predictions

Absrel

Desirable properties of a depth predictor are: (a) accuracy and (b) good zero-shot behavior. One key metric for accuracy of depth predictors is $AbsRel$. Compute the depth prediction from whatever the predictor produces for an image. You might need to use the ground truth depth map for that image to do this. For example, if it predicts $a_d\delta + b_d$, use the depth map to extract a_d and b_d and then recover predicted depth $\hat{z}_{ij}$ for the i, j'th pixel. Write z^*_{ij} for the ground truth depth at each pixel. Now evaluate the mean over all pixels and all images of

$$\frac{|\,\hat{z}_{ij} - z^*_{ij}\,|}{z^*_{ij}}.$$

Delta

Another metric, usually called δ, measures the fraction of pixels such that

$$\max\left(\frac{\hat{z}_{ij}}{z^*_{ij}}, \frac{z^*_{ij}}{\hat{z}_{ij}}\right) > 1.25.$$

Rough SOTA

As of writing, depending on the dataset and on the model, for depth predictors you could expect an AbsRel of between 0.06 and 0.25, a δ of between 1.9% and 25%, and a frame rate of between 5 and 90 FPS. For normal predictors, you could

Generally, faster is less accurate
(good sign that payoff makes sense)