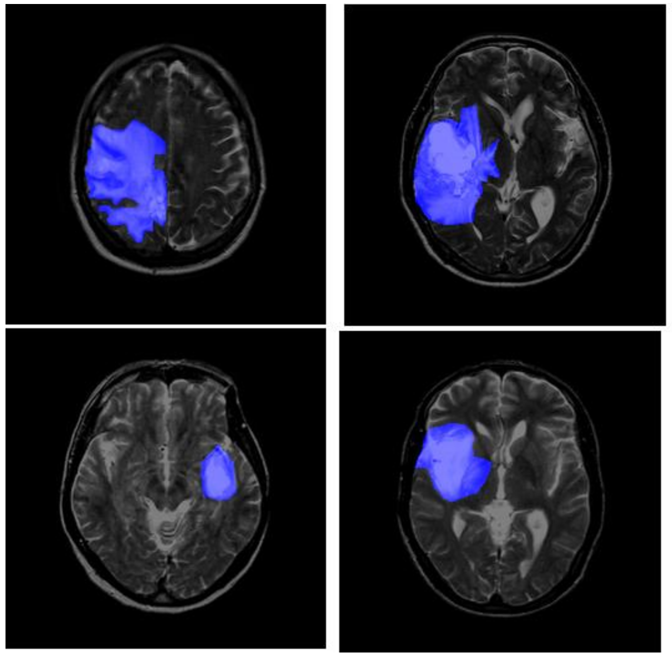
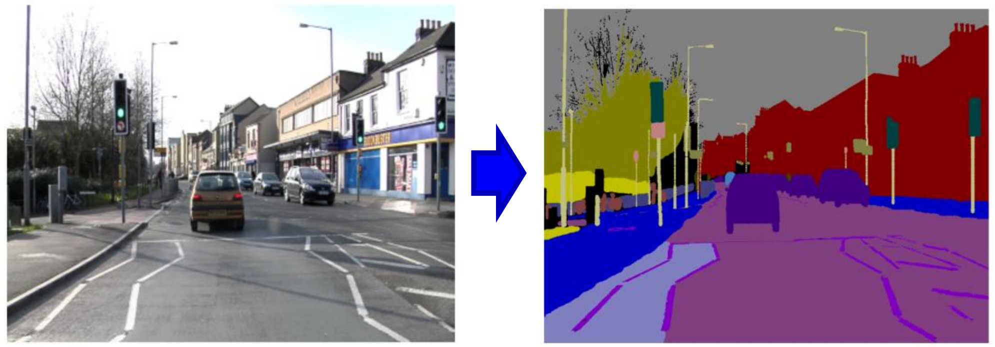
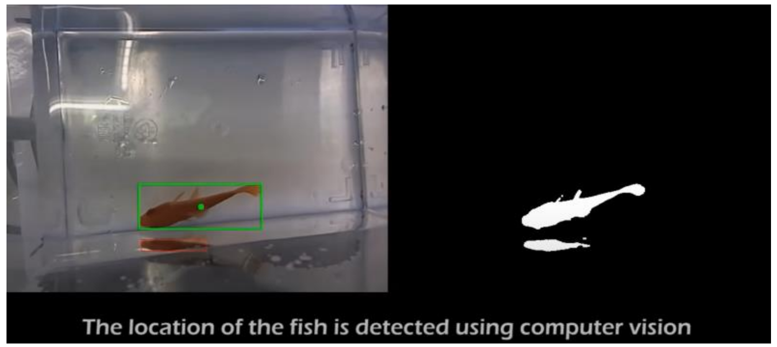
Next Topic: Fully-Convolutional Networks

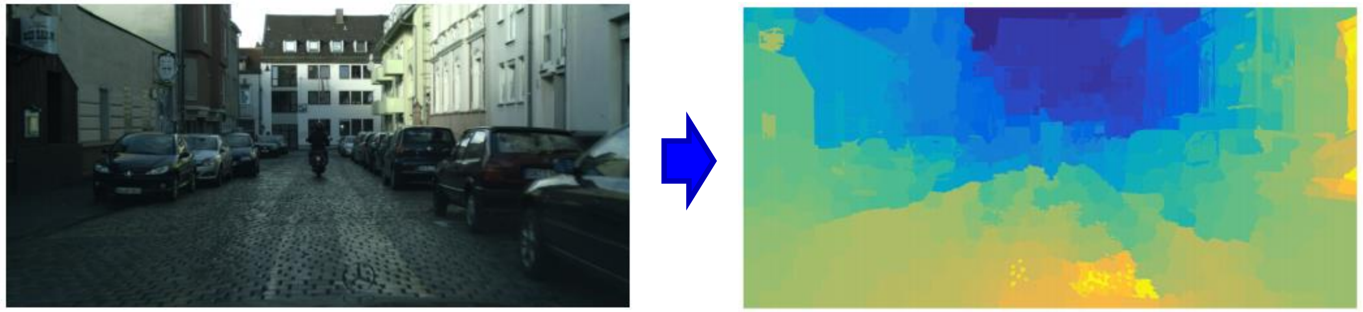
Motivation: Pixel Classification

- Suppose we want to make a prediction at each pixel in an image:
  - Segmenting tumours, segmenting street scenes, depth estimation.







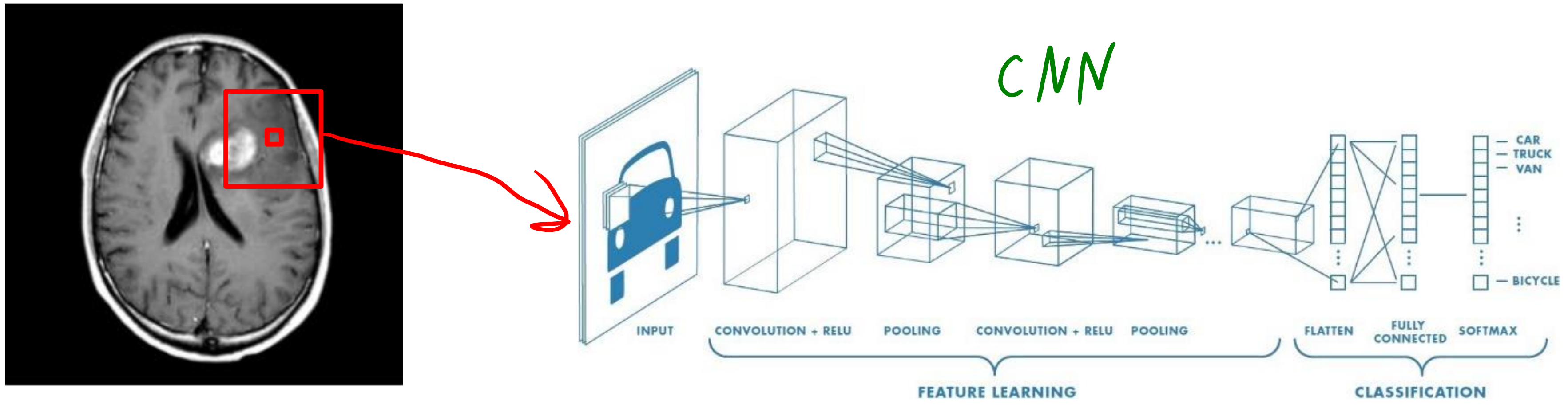


- How can we use CNNs for this problem?

https://www.youtube.com/watch?v=YbNmL6hSNKw

Naïve Approach: Sliding Window Classifier

- Train a CNN that predicts pixel label given its neighbourhood.

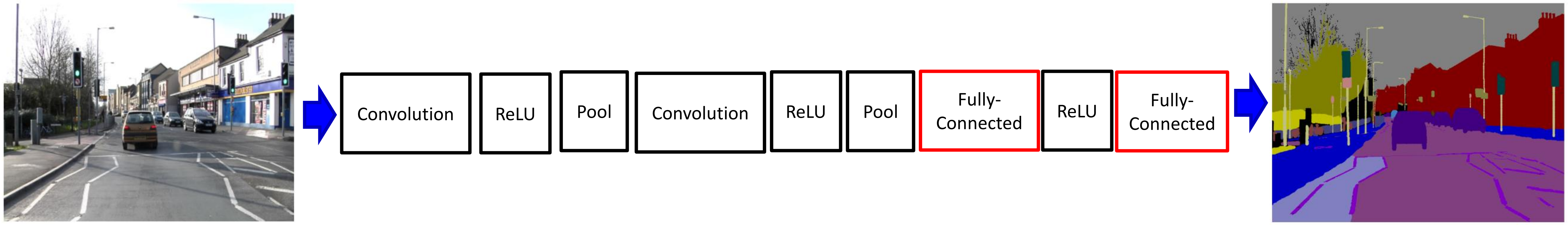


- To label all pixels, apply the CNN to each pixel.
  - Advantages:
    - Turns pixel labeling into image classification.
    - Can be applied to images of different sizes.
  - Disadvantage: this is slow.
    - (Cost of applying CNN) * (number of pixels in the image).

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Encoding-Decoding for Pixel Classification

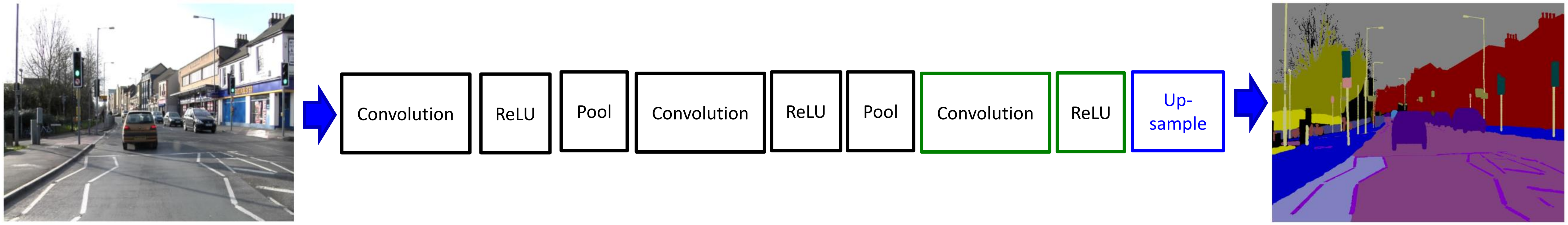
- Similar to multi-label, could use CNN to generate an image encoding.
  - With output layer making a prediction at each pixel.



- Much-faster classification.
  - Small number of “global” convolutions, instead of repeated “local” convolutions.
- But, the encoding has mixed up all the space information.
  - Due to fully-connected layers.
  - Fully-connected layer needs to learn “how to put the image together”.
    - And all input images must be the same size.

Fully-Convolutional Networks

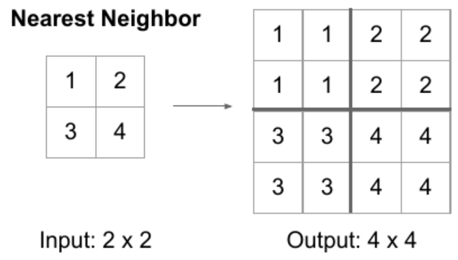
- Fully-convolutional networks (FCNs):
  - CNNs with no fully-connected layers (only convolutional and pooling).



- Maintains fast classification of the encoding-decoding approach.
- Same parameters used across space at all layers.
  - This allows using the network on inputs of different sizes.
- Needs upsampling layer(s) to get back to image size.
- FCNs quickly achieved state of the art results on many tasks.

Traditional Upsampling Methods

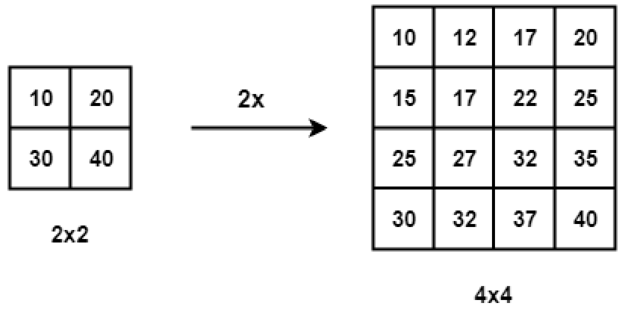
- In upsampling, we want to go from a small image to a bigger image.
  - This requires interpolation: guessing "what is inbetween the pixels".
- Classic upsampling operator is nearest neighbours interpolation:
  - But this creates blocky/pixelated images.



1	2
3	4

1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

- Another classic method is bilinear interpolation:
  - Weighted combination of corners:
  - More smooth methods include "bicubic" and "splines".



10	20
30	40

10	12	17	20
15	17	22	25
25	27	32	35
30	32	37	40

- In FCNs, we learn the upsampling/interpolation operator.

https://towardsdatascience.com/transposed-convolution-demystified-84ca81b4baba

Upsampling with Transposed Convolution

- FCN Upsampling layer is implemented with a transposed convolution.
  - Sometimes called "deconvolution" in ML or "fractionally-strided convolution".
    - But not related to deconvolution in signal processing.

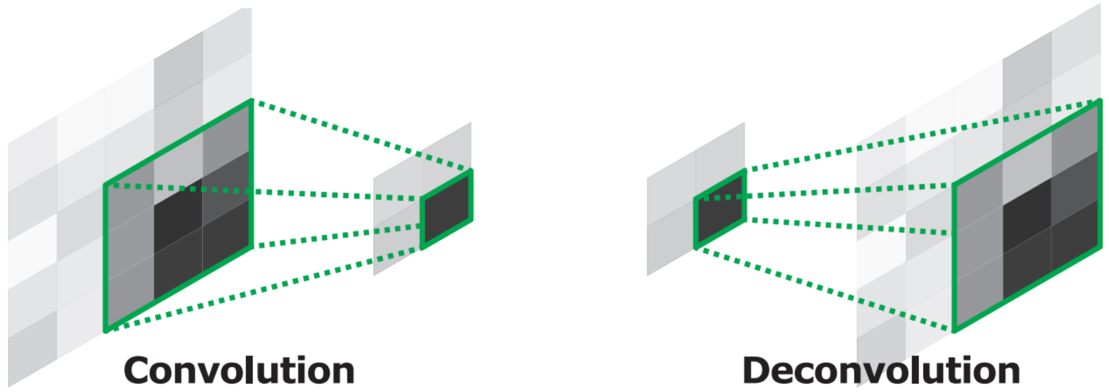
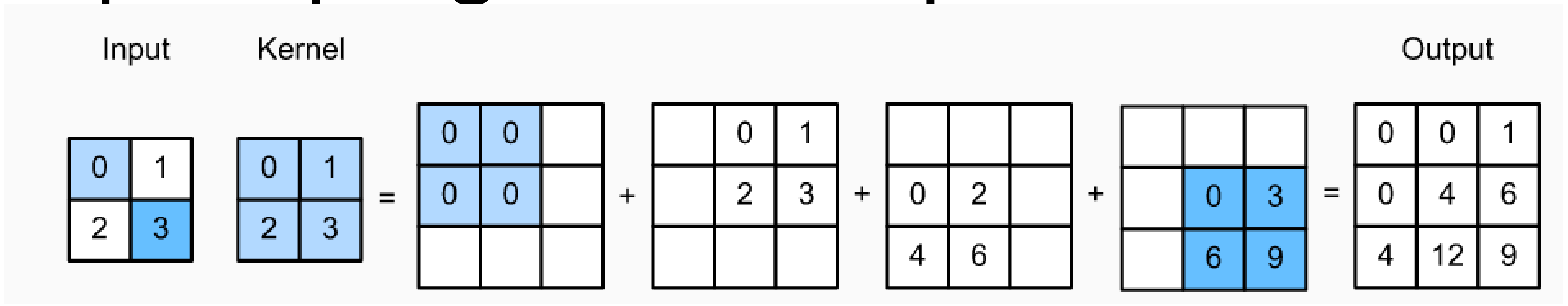


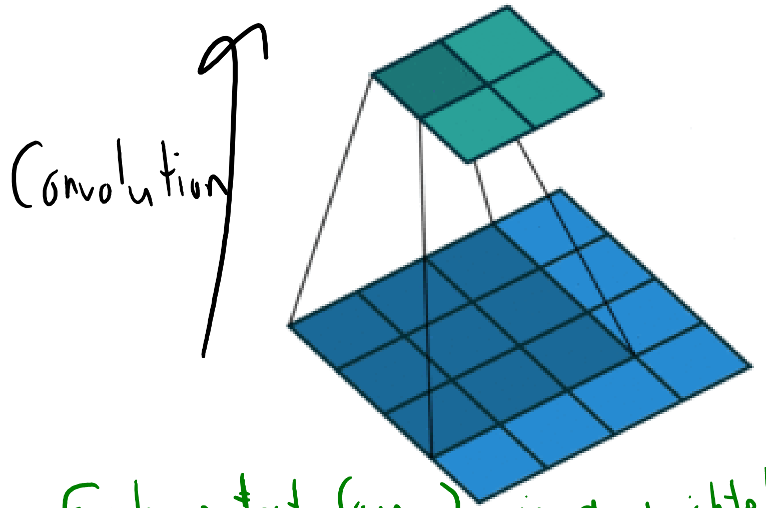
Figure 3. Illustration of deconvolution and unpooling operations.

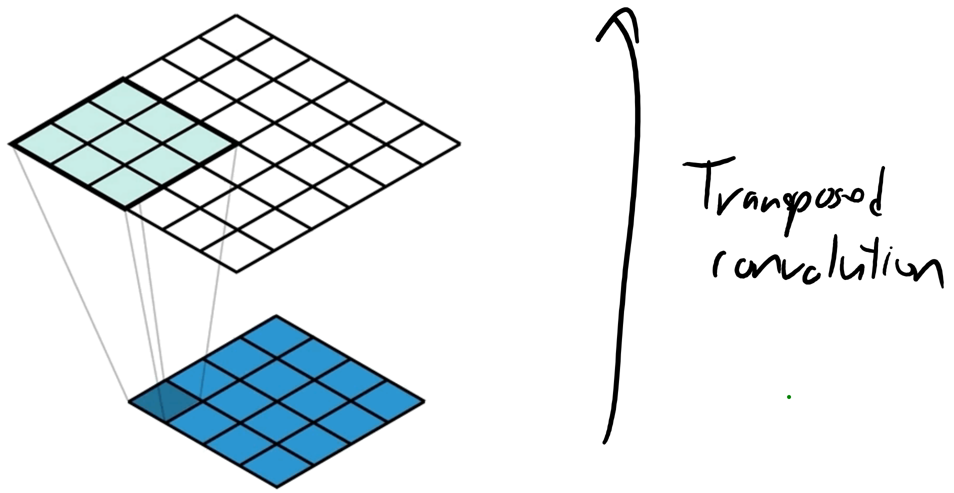
- Convolution generates 1 pixel by taking weighted combination of several pixels.
  - And we learn the weights.
- Transposed convolution generates several pixels by weighting 1 pixel.
  - And we learn the weights.
  - This generates overlapping regions, which get added together to make final image.

https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Noh_Learning_Deconvolution_Network_ICCV_2015_paper.pdf

Upsampling with Transposed Convolution







Each output (green) is a weighted combination of one 3×3 region

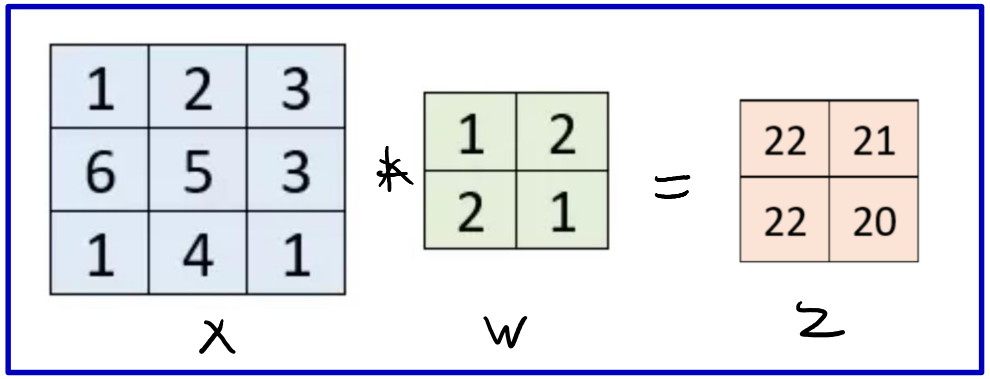
Each input (blue) gives a 3×3 region. These regions overlap, so we take a weighted combination at each pixel to give final result.

- Animations here and here.

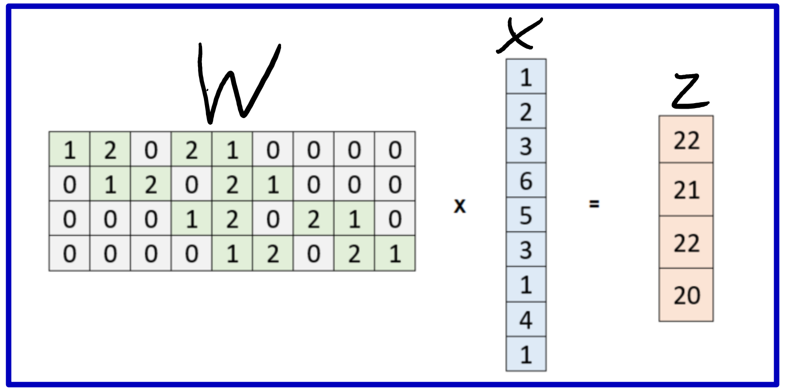
https://towardsdatascience.com/transposed-convolution-demystified-84ca81b4baba

Why is it called "transposed" convolution?

- We can write the convolution operator as a matrix multiplication, z=Wx.

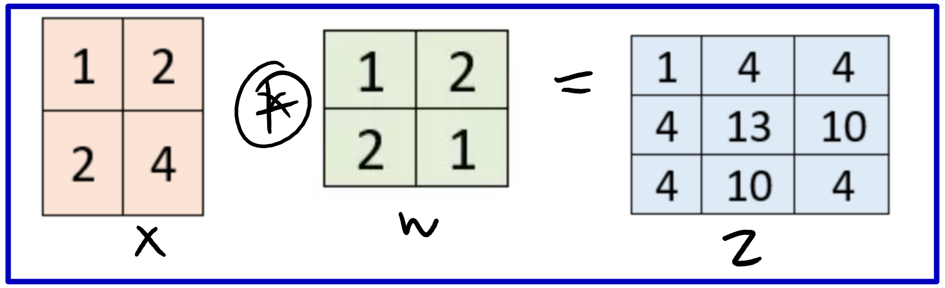


$$\underbrace{\begin{bmatrix}1&2&3\\6&5&3\\1&4&1\end{bmatrix}}_{x} * \underbrace{\begin{bmatrix}1&2\\2&1\end{bmatrix}}_{w} = \underbrace{\begin{bmatrix}22&21\\22&20\end{bmatrix}}_{z}$$

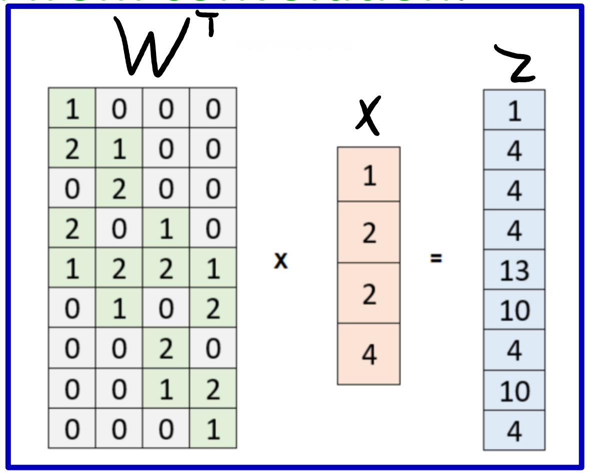


$$\overset{W}{\begin{bmatrix}1&2&0&2&1&0&0&0&0\\0&1&2&0&2&1&0&0&0\\0&0&0&1&2&0&2&1&0\\0&0&0&0&1&2&0&2&1\end{bmatrix}} \times \overset{x}{\begin{bmatrix}1\\2\\3\\6\\5\\3\\1\\4\\1\end{bmatrix}} = \overset{z}{\begin{bmatrix}22\\21\\22\\20\end{bmatrix}}$$

- In transposed convolution, non-zero pattern of 'W' is transposed from convolution.
  - You can implement transposed convolution as a convolution.



$$\underbrace{\begin{bmatrix}1&2\\2&4\end{bmatrix}}_{x} \circledast \underbrace{\begin{bmatrix}1&2\\2&1\end{bmatrix}}_{w} = \underbrace{\begin{bmatrix}1&4&4\\4&13&10\\4&10&4\end{bmatrix}}_{z}$$



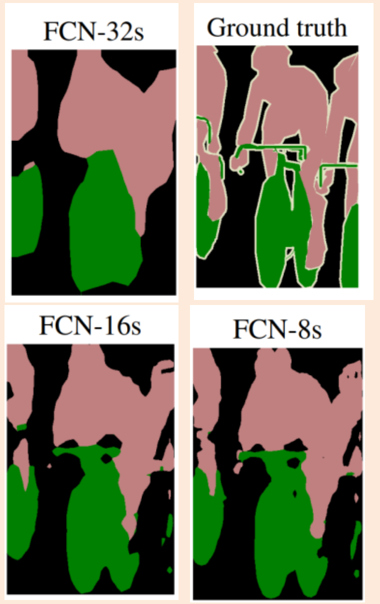
$$\overset{W^T}{\begin{bmatrix}1&0&0&0\\2&1&0&0\\0&2&0&0\\2&0&1&0\\1&2&2&1\\0&1&0&2\\0&0&2&0\\0&0&1&2\\0&0&0&1\end{bmatrix}} \times \overset{x}{\begin{bmatrix}1\\2\\2\\4\end{bmatrix}} = \overset{z}{\begin{bmatrix}1\\4\\4\\4\\13\\10\\4\\10\\4\end{bmatrix}}$$

- In this example the filter is the same, but does not need to be:
  - Transposed convolution is not the "reverse" of convolution (it only "reverses" the size).

https://www.machinecurve.com/index.php/2019/09/29/understanding-transposed-convolutions/

Increasing Resolution: FCN Skip Connections

- Convolutions and pooling lose a lot of information.
- Original FCN paper considered adding skip connections to help upsampling:



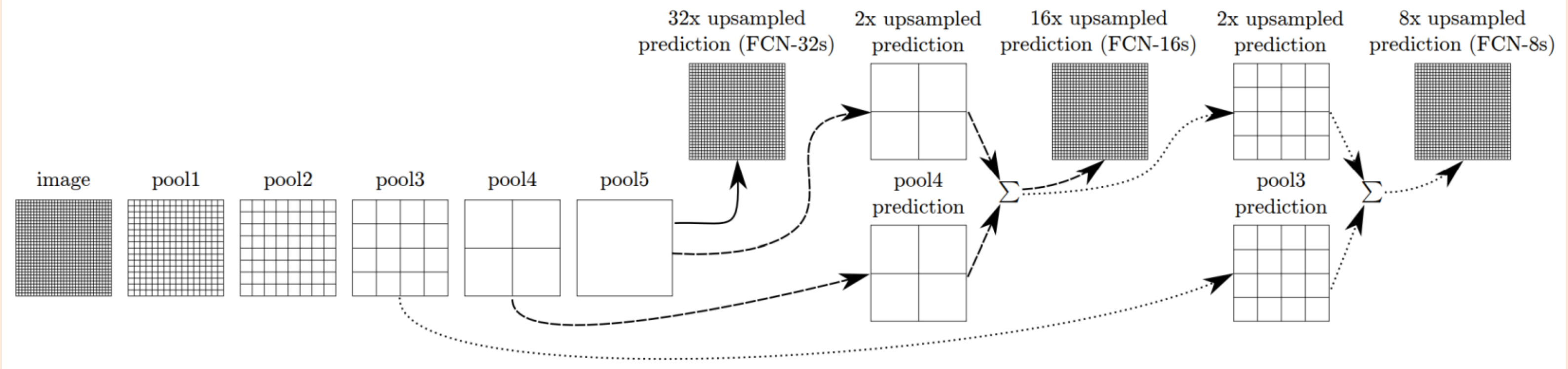


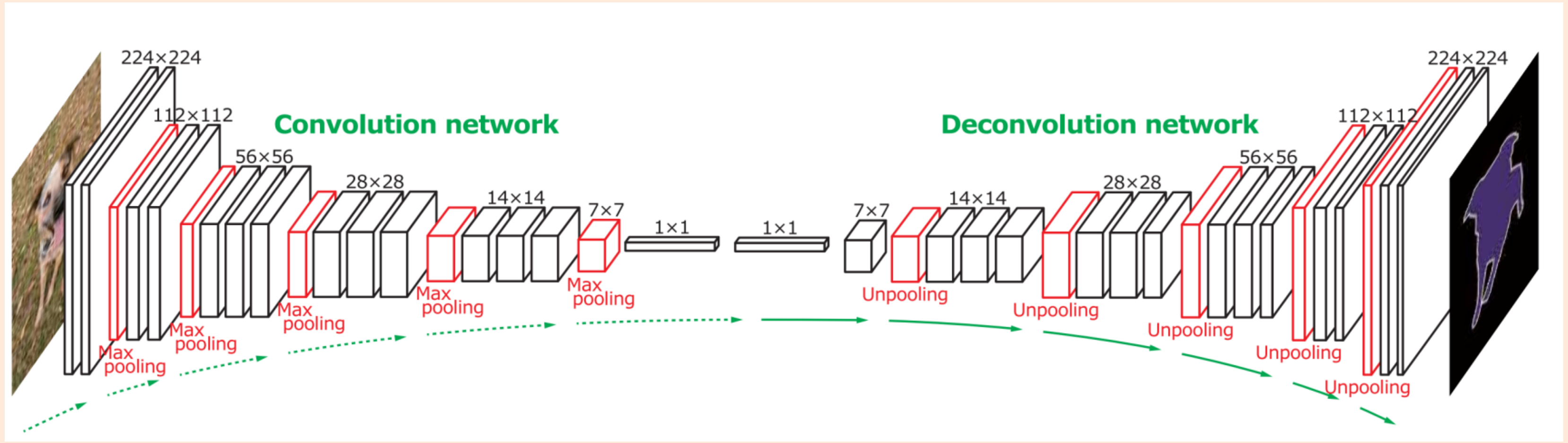
Figure 3. Our DAG nets learn to combine coarse, high layer information with fine, low layer information. Layers are shown as grids that reveal relative spatial coarseness. Only pooling and prediction layers are shown; intermediate convolution layers (including our converted fully connected layers) are omitted. Solid line (FCN-32s): Our single-stream net, described in Section 4.1, upsamples stride 32 predictions back to pixels in a single step. Dashed line (FCN-16s): Combining predictions from both the final layer and the `pool4` layer, at stride 16, lets our net predict finer details, while retaining high-level semantic information. Dotted line (FCN-8s): Additional predictions from `pool3`, at stride 8, provide further precision.

- Allows using high-resolution information from earlier layers.
- They first trained the low-resolution FCN-32, then FCN-16, then FCN-8.
  - "First learn to encode at a low resolution", then slowly increase resolution.
  - Parameters of transposed convolutions initialized to simulate "bilinear interpolation".

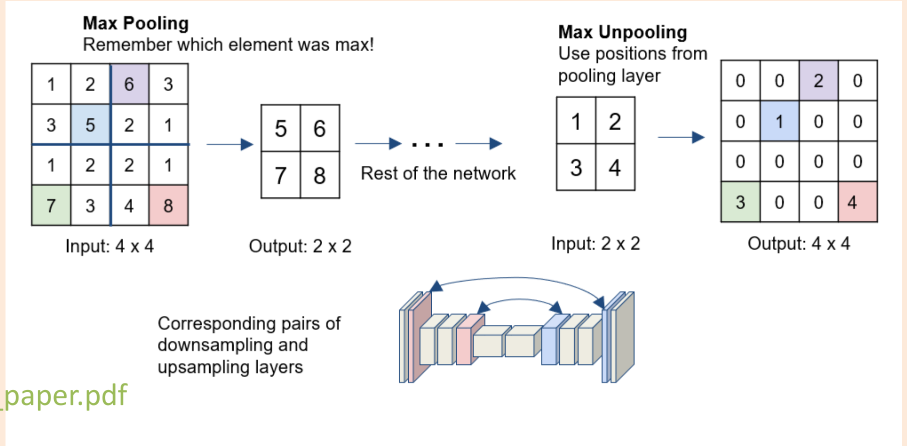
https://arxiv.org/pdf/1411.4038.pdf

Increasing Resolution: Deconvolution Networks

- Alternate resolution-increasing method is deconvolution networks:



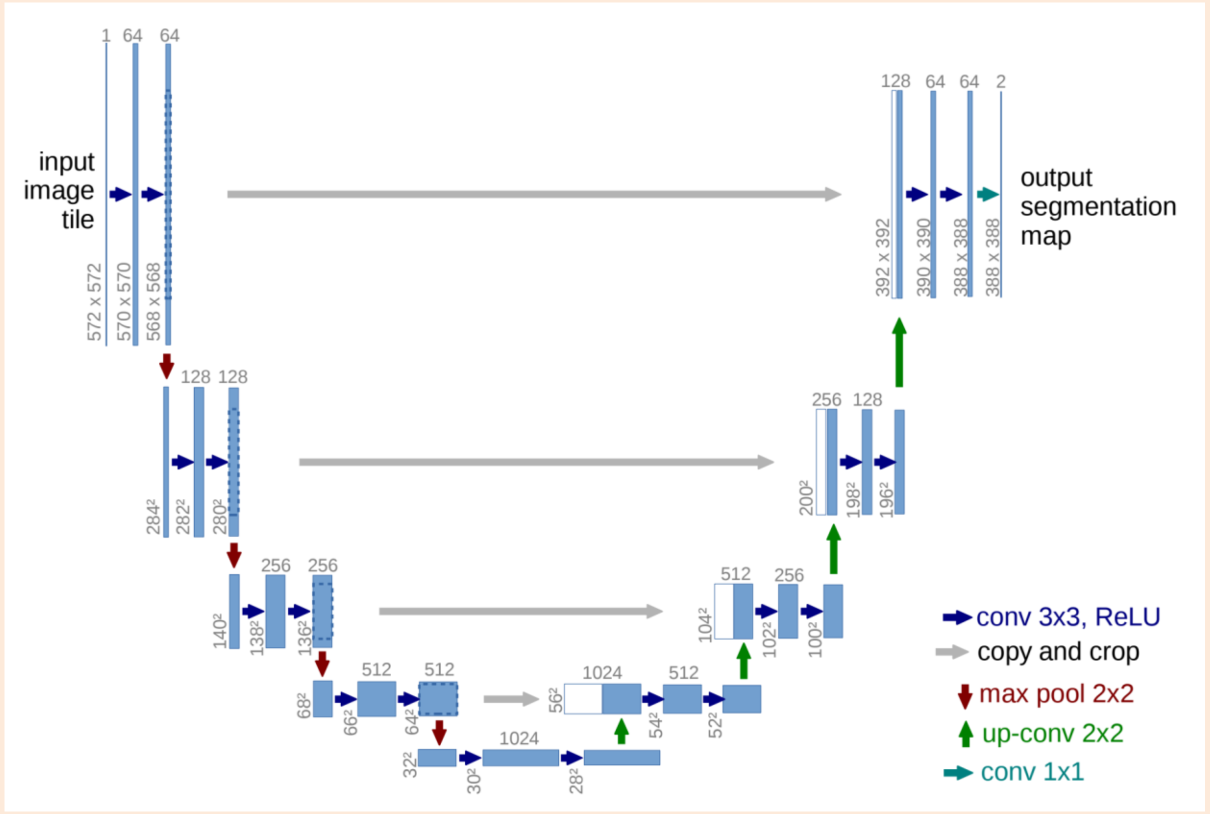
- Includes transposed convolution layers and unpooling layers.
  - Store the max pooling argmax values.
  - Restores "where" activation happened.
    - Still loses the "non-argmax" information.


Max Pooling
Remember which element was max!
Input: 4 x 4
Output: 2 x 2
Rest of the network
Max Unpooling
Use positions from pooling layer
Input: 2 x 2
Output: 4 x 4
Corresponding pairs of downsampling and upsampling layers

https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Noh_Learning_Deconvolution_Network_ICCV_2015_paper.pdf
https://towardsdatascience.com/transposed-convolution-demystified-84ca81b4baba

Increasing Resolution: U-Nets

- Another popular variant is u-nets:



- Decoding has connections to same-resolution encoding step.

https://arxiv.org/pdf/1505.04597.pdf

High Resolution Predictions with FCNs

- “AutoPortrait”: automatic high-resolution photo re-touching.

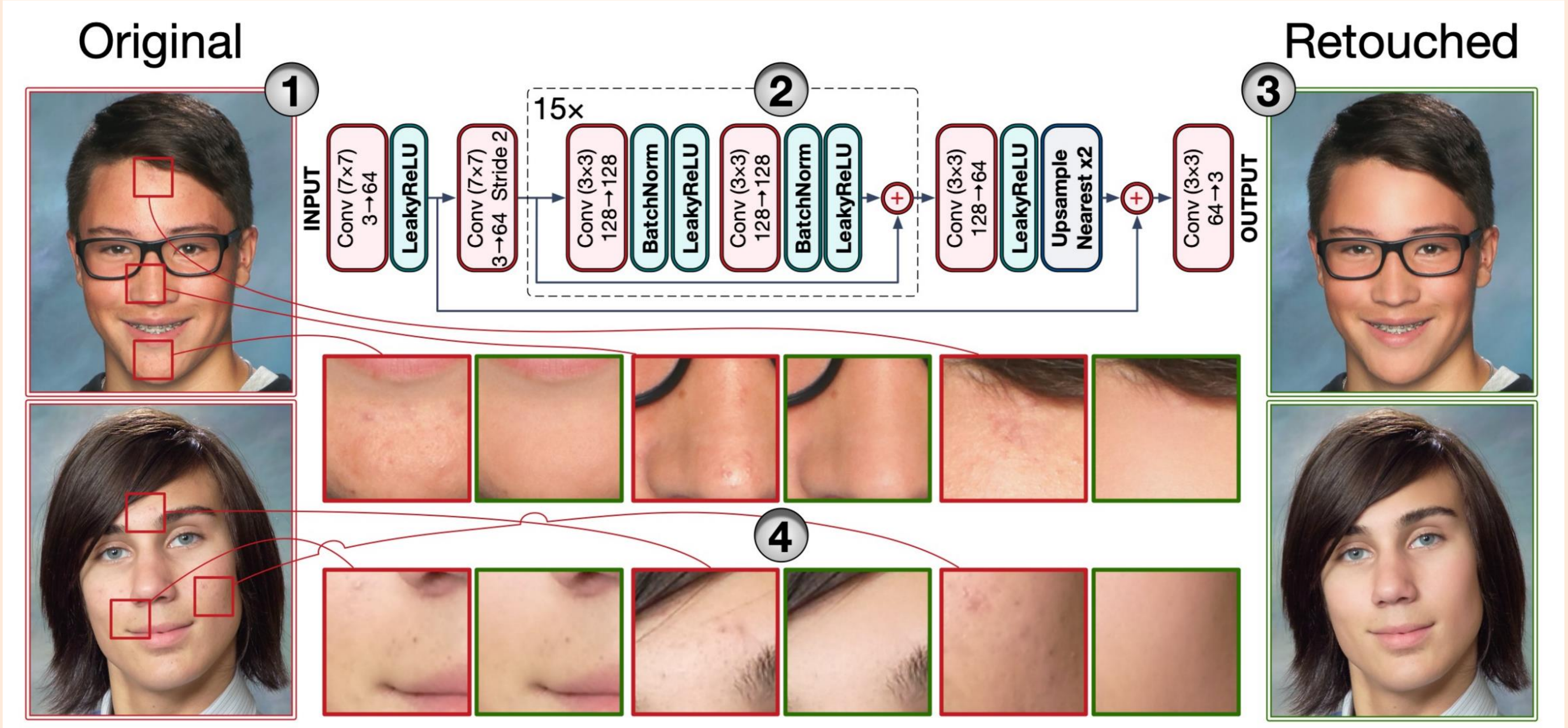
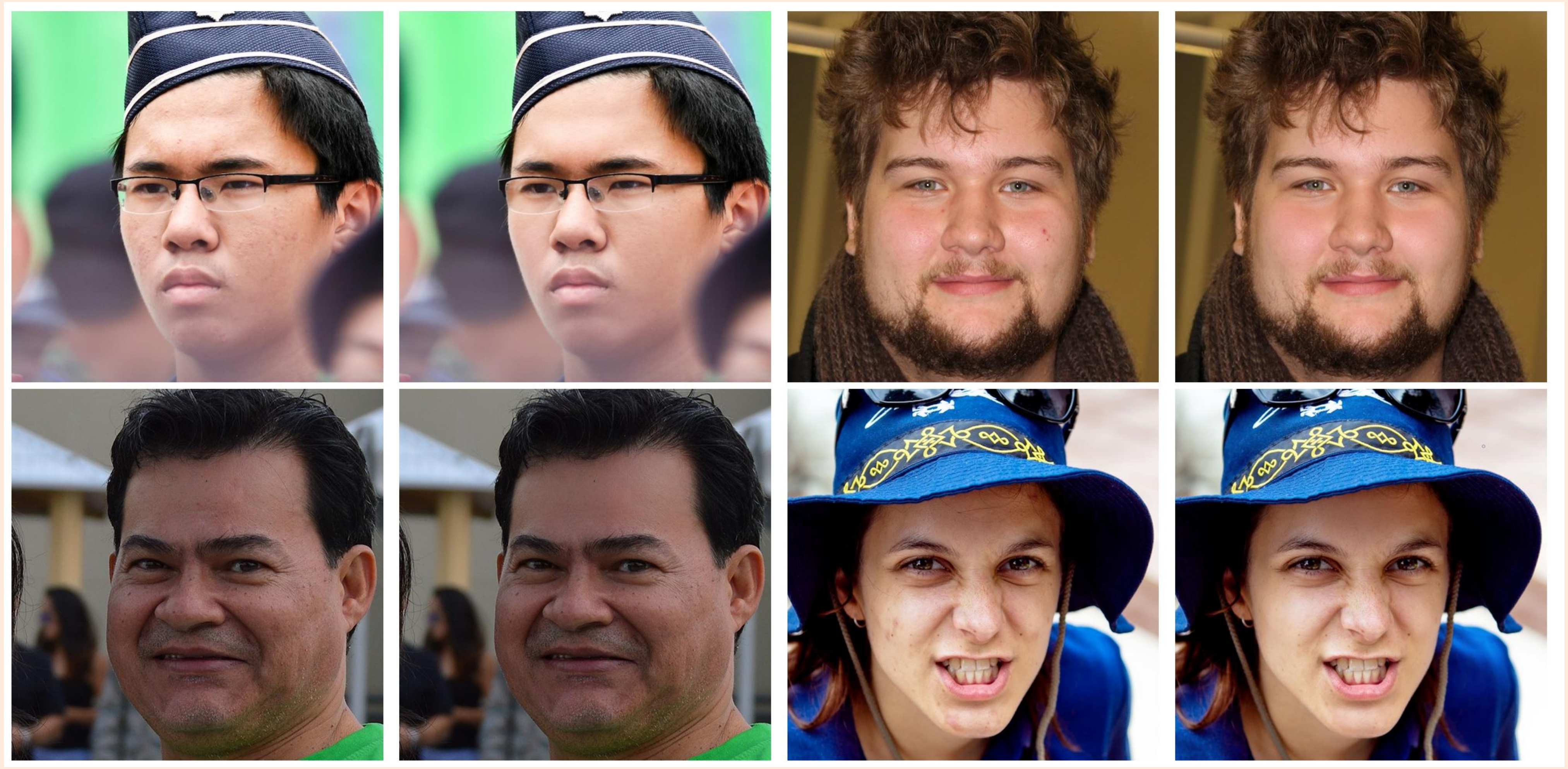


Figure 1: Overview of AutoRetouch: (1) input portraits cropped to the head, (2) the images pass through the fully convolutional architecture of AutoRetouch, (3) the output portraits are retouched while preserving fine textures. (4) Sample before-after patches scaled up. We empirically show that our final models have desirable generalization properties.

High Resolution Predictions with FCNs

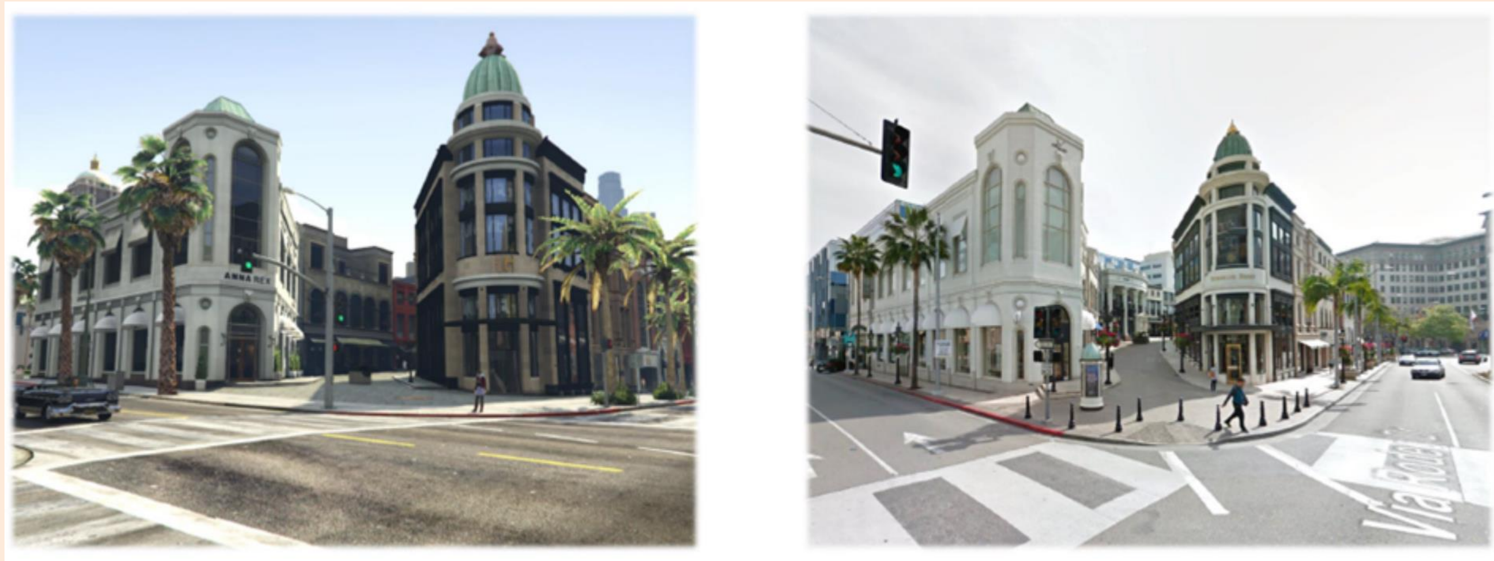
- “AutoPortrait”: automatic high-resolution photo re-touching.



https://openaccess.thecvf.com/content/WACV2021/papers/Shafaei_AutoRetouch_Automatic_Professional_Face_Retouching_WACV_2021_paper.pdf

Source of Labels

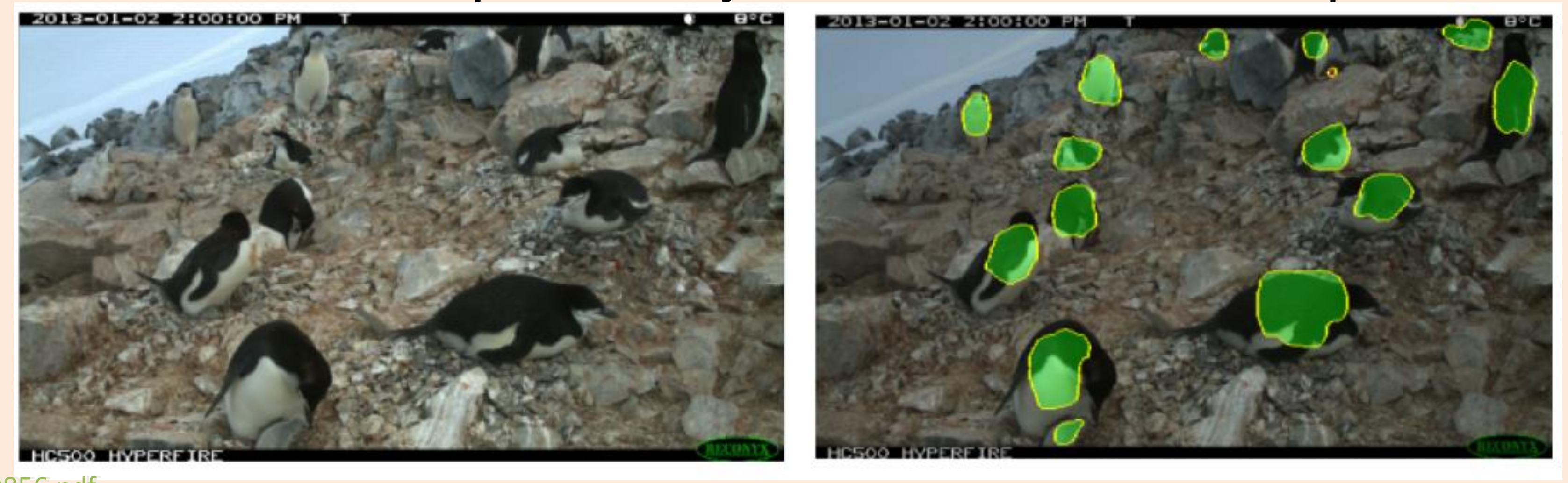
- Labeling every pixel takes a long time.
- Where we get the labels from?
  - Domain expert (medical images).
  - Grad students or paid labelers (ImageNet).
  - Simulated environments.
    - High number of lower-quality examples.
    - Often a net gain with fine-tuning on real images.
    - Can get data at night, in fog, or dangerous situations.


Video game Google street view

https://arxiv.org/pdf/1608.01745v1.pdf

Source of Labels

- Recent works recognize you do not need to label every pixel.
  - You can evaluate loss/gradient on a subset of labeled pixels.
  - Could have labeler click on a few pixels inside objects, and a few outside.
    - Many variations are possible, that let you label a lot of images in a short time.
- Penguin counting based on "single pixel" labels in training data:
  - And some tricks to separate objects and remove false positives:



Summary

- Autoencoders:
  - Neural network where the output is the input.
    - Non-linear generalization of PCA.
  - Encode data into a bottleneck layer, then decode predict original input.
  - Can be used for visualization, compression, outlier detection, pre-training.
- Denoising autoencoders train to uncorrupt/enhance images.
  - Useful for removing noise, adding colour, super-resolution, and so on.
- Encoding-Decoding approach to multi-label classification:
  - Have all classes shared the same hidden layer(s).
  - Reduces number of parameters.
  - Models dependencies between classes, while keeping inference easy.
- Pre-training:
  - Use parameters from model trained a on large diverse dataset, to initialize SGD for new dataset.
- Fully-convolutional networks (FCNs):
  - CNNs where every layer maintains spatial information.
  - Useful for handling images of different sizes.
  - Requires upsampling to be used for pixel classification.
  - Transposed convolutions learn upsampling operators.

- Next time: are we learning harmful biases?