CPSC 440: Machine Learning

Fully-Convolutional Networks

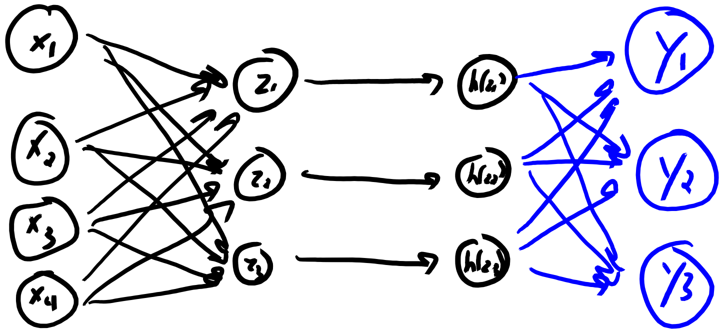
Winter 2022

Last Time: Multi-Label Classification

- Consider multi-label classification:
  - "Which of the 'k' objects are in this image?"

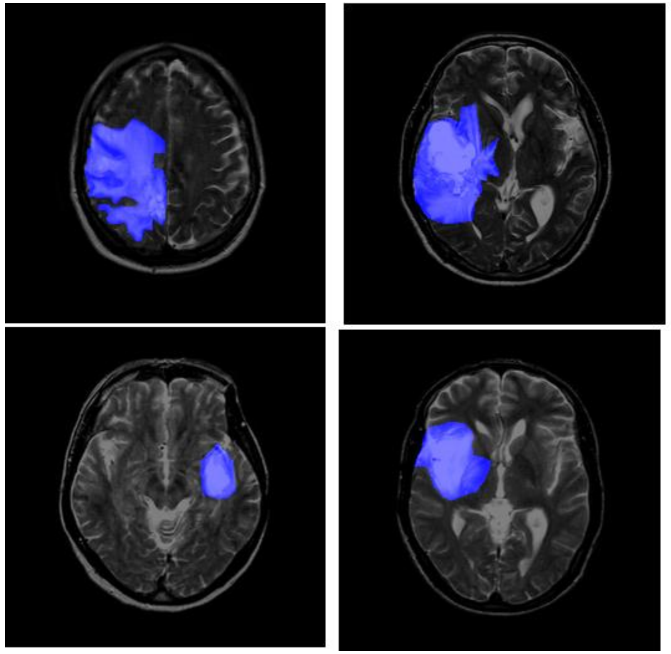
$$X = \underbrace{\left[\begin{array}{ccc} \quad & \quad & \quad \\ & & \\ & & \\ & & \end{array}\right]}_{d} \Big\} n \qquad Y = \begin{array}{c} \begin{array}{ccccc} \text{cat} & \text{dog} & \text{person} & \text{chair} & \text{mouse} \end{array} \\ \underbrace{\left[\begin{array}{ccccc} 1 & -1 & 1 & 1 & 1 \\ -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 \\ -1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & 1 \end{array}\right]}_{k} \end{array} \Big\} n$$

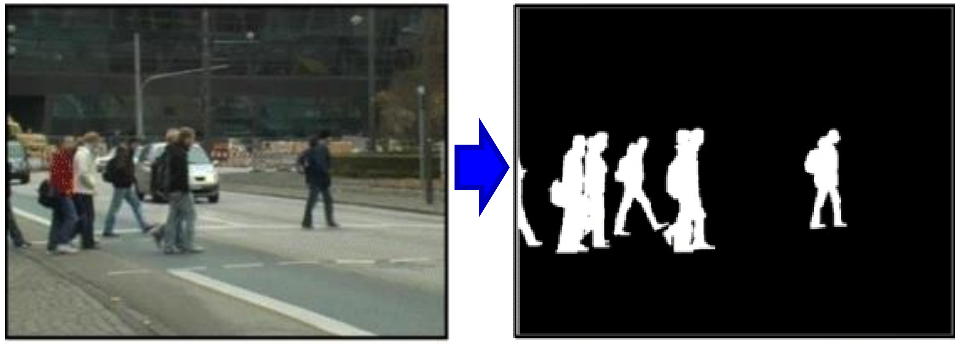
- We considered an encoding-decoding approach:
  - Fewer parameters than independent classifiers.
  - Captures dependencies through shared hidden layer.

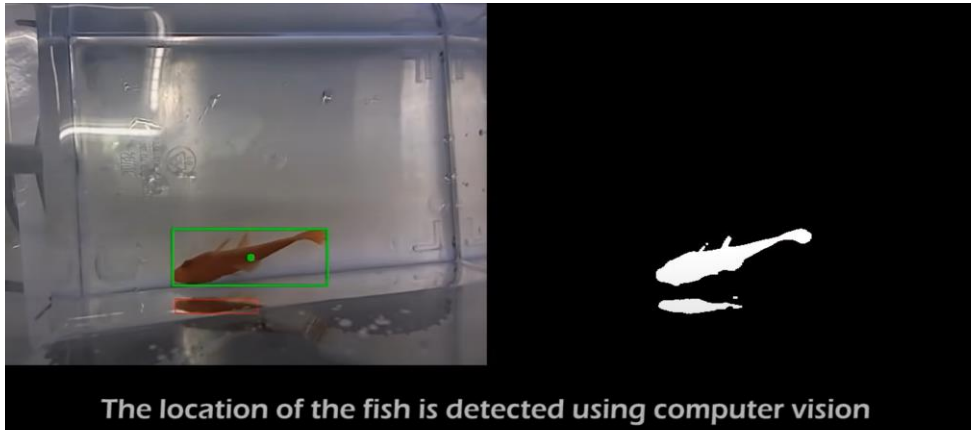


Motivation: Pixel Classification

- Suppose we want to assign a binary label to each pixel in an image:
  - Tumour vs. non-tumour, pedestrian vs. non-pedestrian, and so on.





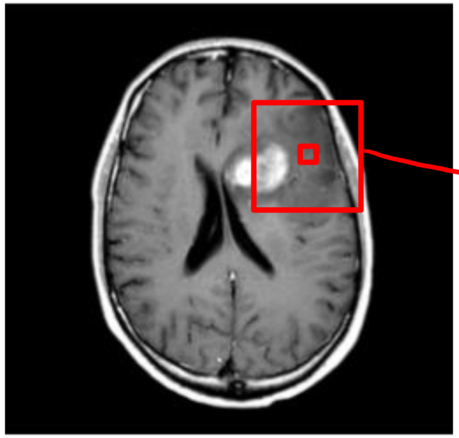


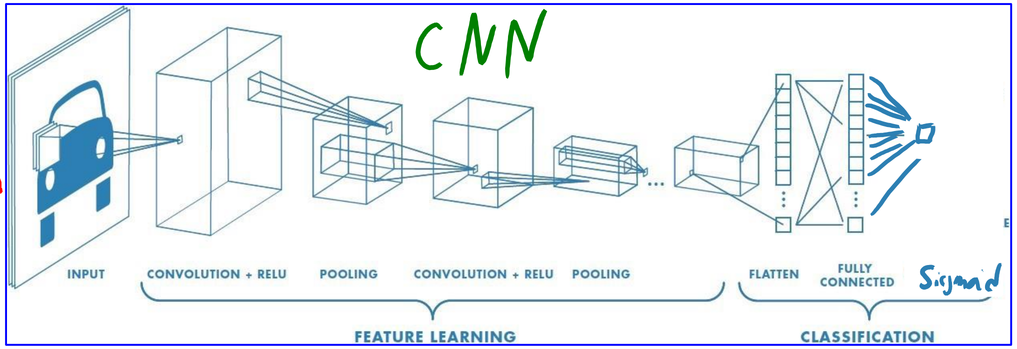
- How can we use CNNs for this problem?

https://ipsjcva.springeropen.com/articles/10.1186/s41074-019-0062-2
https://www.youtube.com/watch?v=YbNmL6hSNKw

Naïve Approach: Sliding Window Classifier

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

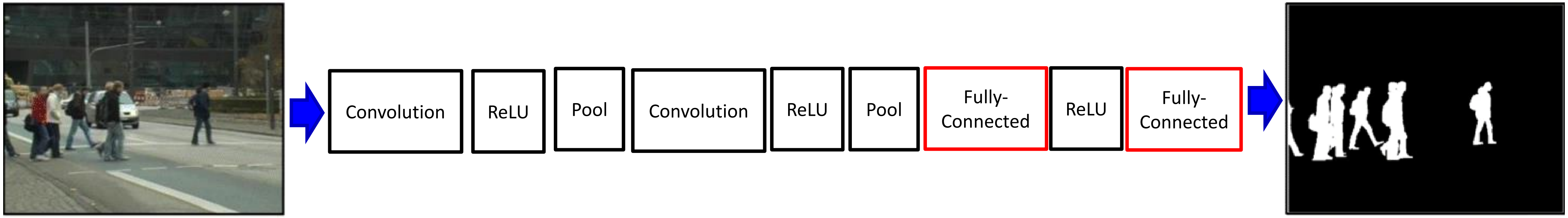




- To label all pixels, apply the CNN to each pixel.
  - Advantages:
    - Turns pixel labeling into image classification.
    - Can be applied to images of different sizes.
  - Disadvantage: this is slow.
    - (Cost of applying CNN) * (number of pixels in the image).

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.

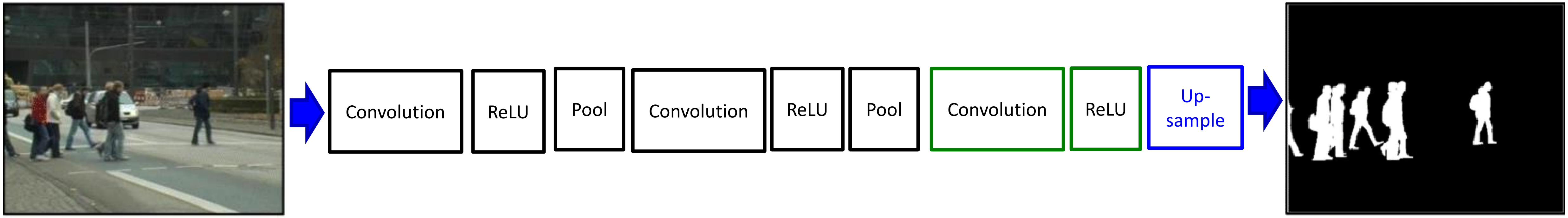


- Much-faster classification.
  - Small number of “global” convolutions, instead of repeated “local” convolutions.
- But, the encoding has mixed up all the space information.
  - Due to fully-connected layers.
  - Fully-connected layer needs to learn “how to put the image together”.
    - And images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

https://ipsjcva.springeropen.com/articles/10.1186/s41074-019-0062-2

Fully-Convolutional Networks

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

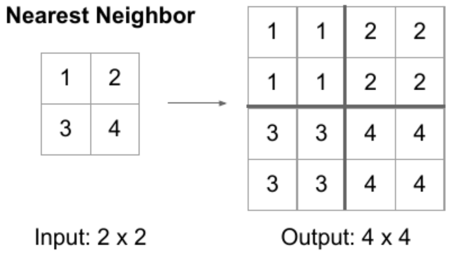


- Maintains fast classification of the encoding-decoding approach.
- Same parameters used across space at all layers.
  - This allows using the network on inputs of different sizes.
  - Needs upsampling layer(s) to get back to image size.
- FCNs quickly achieved state of the art results on many tasks.

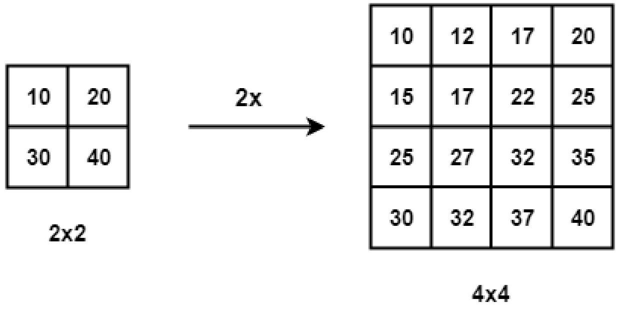
https://ipsjcva.springeropen.com/articles/10.1186/s41074-019-0062-2

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.



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



- 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.

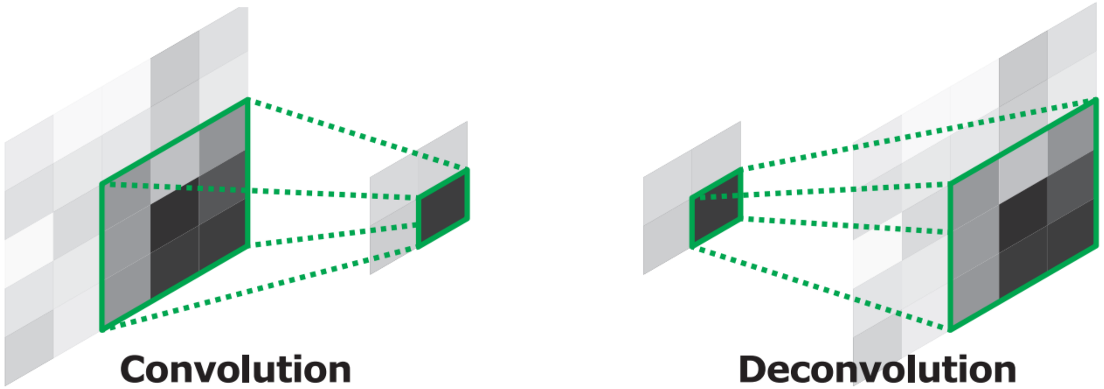
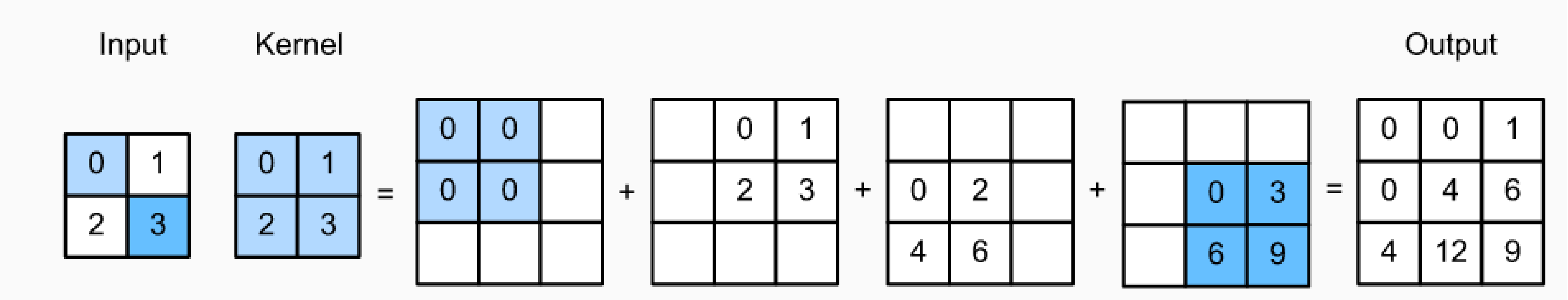


Figure 3. Illustration of deconvolution and unpooling operations.

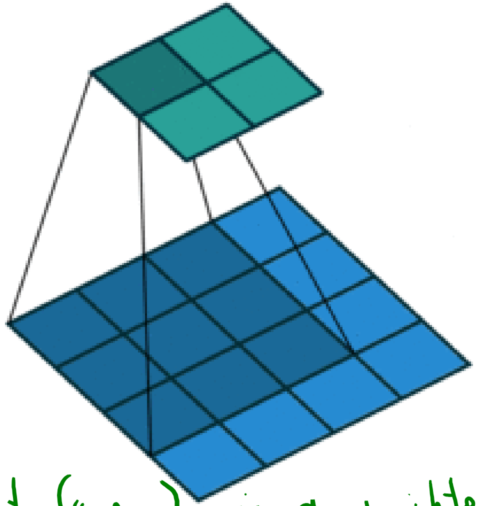
- Convolution generates 1 pixel by taking weighted combination of several pixels.
  - And we learn the weights.
- Transposed convolution several pixels by weighting 1 pixel.
  - And we learn the weights.
  - This generates overlapping regions, which get added together to make final image.

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

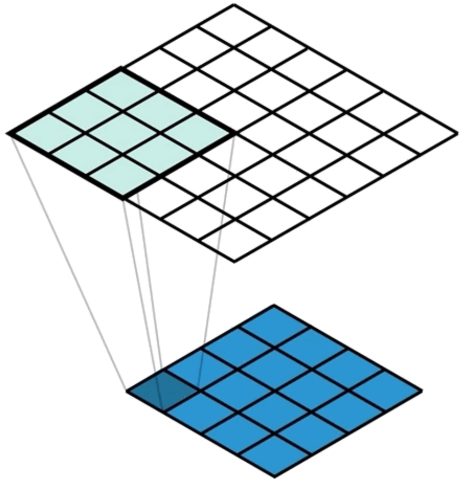
Upsampling with Transposed Convolution



Convolution



Each output (green) is a weighted combination of one 3x3 region



Transposed convolution

Each input (blue) gives a 3x3 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.

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

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

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

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

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

- 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:

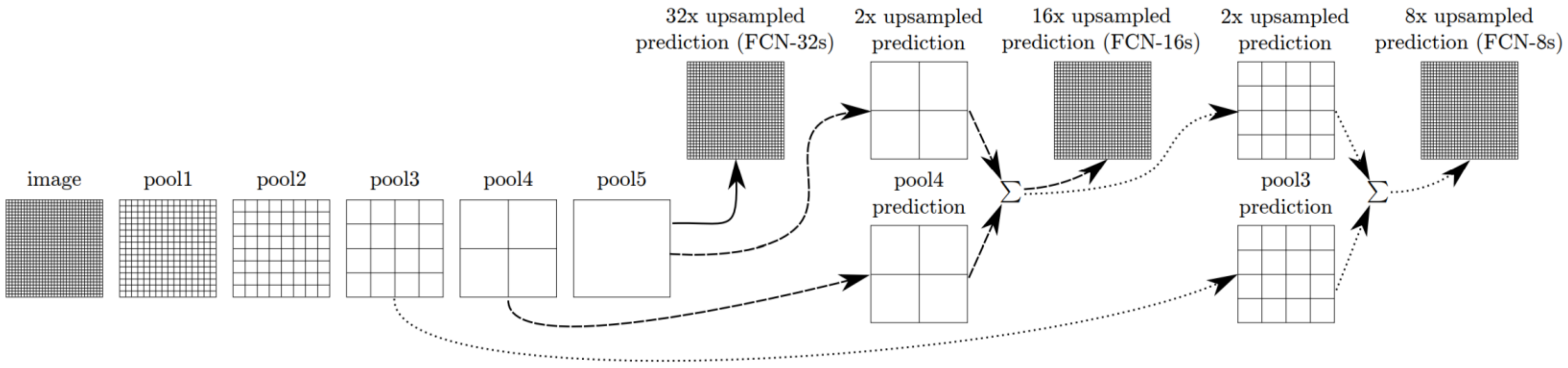
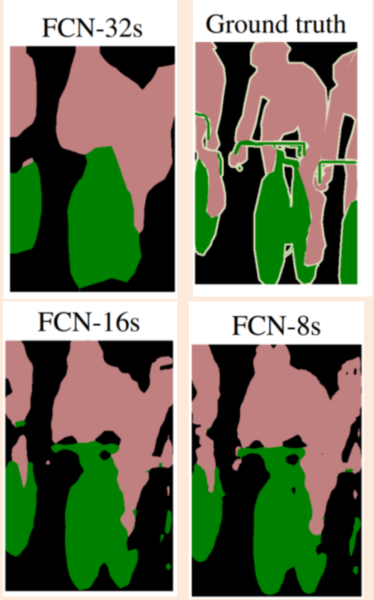


Figure 3. Our DAG nets learn to combine coarse, high layer information with fine, low layer information. Layers are shown as grids that reveal relative spatial coarseness. Only pooling and prediction layers are shown; intermediate convolution layers (including our converted fully connected layers) are 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.

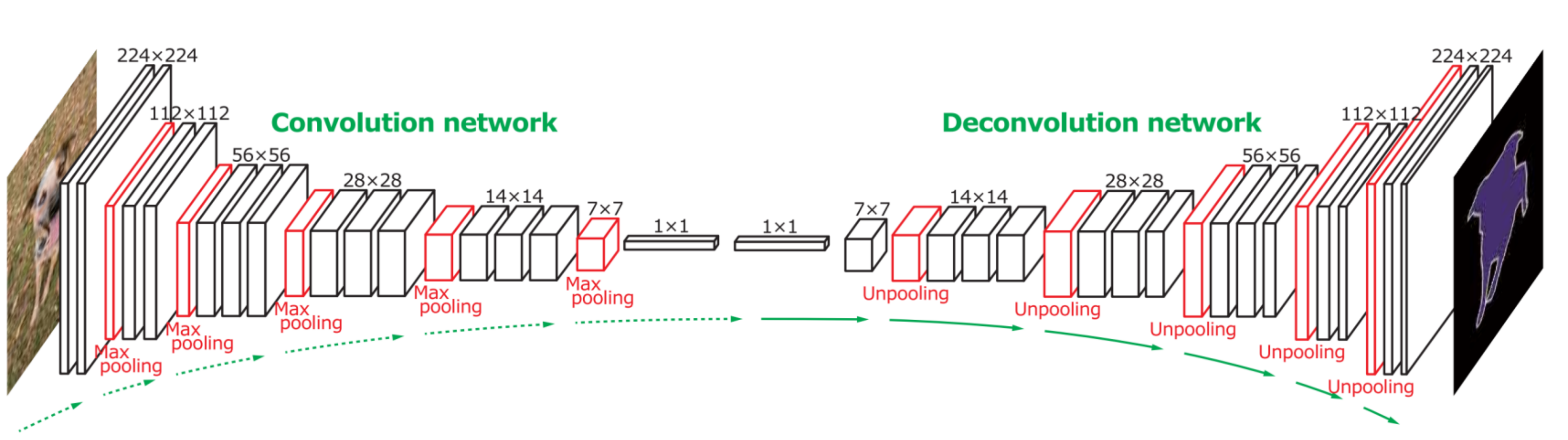


- Allows using high-resolution information from earlier layers.
- They first trained the low-resolution FCN-32, then FCN-16, then FCN-8.
  - “First learn to encode at a low resolution”, then slowly increase resolution.
  - Parameters of transposed convolutions initialized to simulate “bilinear interpolation”.

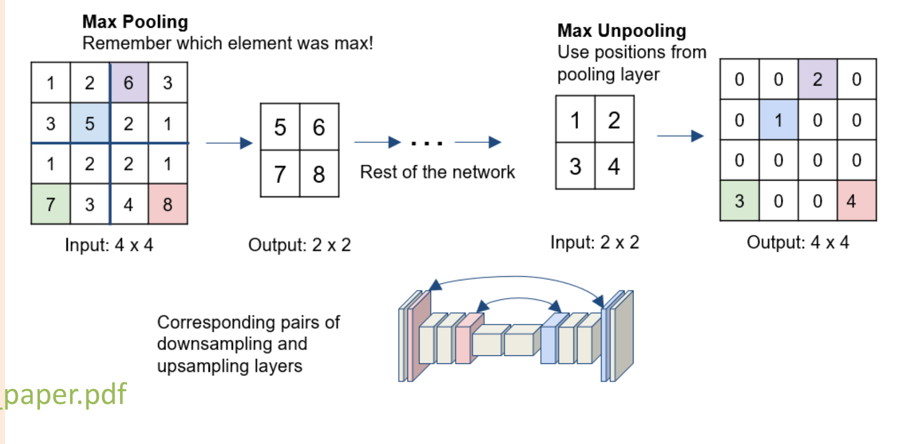
https://arxiv.org/pdf/1411.4038.pdf

Increasing Resolution: Deconvolution Networks

- Alternate resolution-increasing method is deconvolution networks:



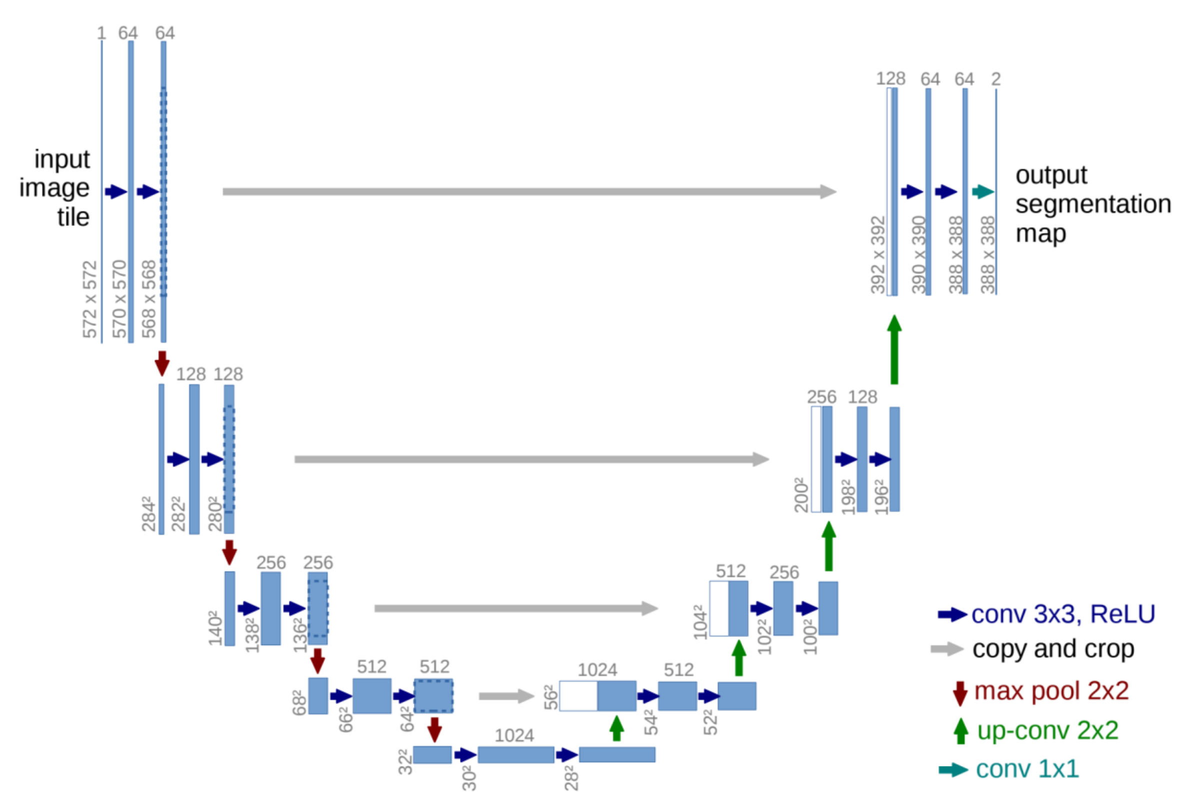
- Includes transposed convolution layers and unpooling layers.
  - Store the max pooling argmax values.
  - Restores "where" activation happened.
    - Still loses the "non-argmax" information.



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

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.

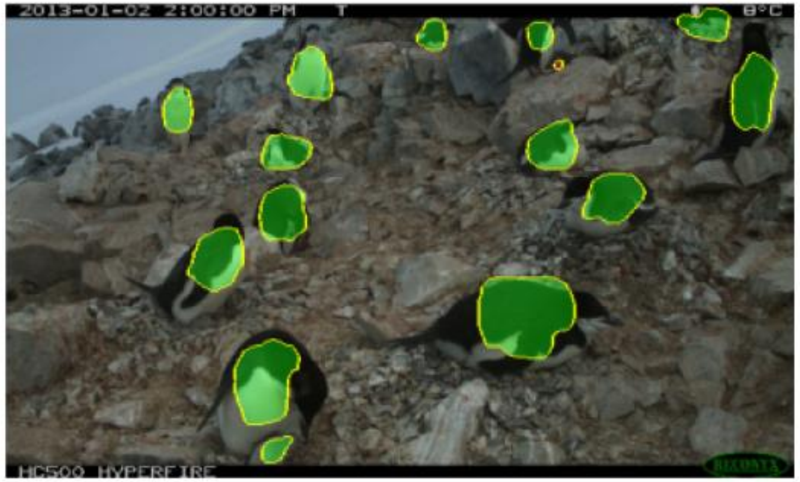
Video game

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 "single pixel" labels in training data:
  - And some tricks to separate objects and remove false positives:



https://arxiv.org/pdf/1807.09856.pdf

End of Part 1 ("Binary Variables"): Key Concepts

- We discussed binary density estimation.
  - Model the proportion of times a binary event happens.
- We discussed the Bernoulli parameterization.
- We discussed various inference tasks, given the parameter:
  - Compute probabilities, find decoding, generate samples.
- We discuss different learning strategies, given data:
  - Maximum likelihood estimation (MLE), maximum a posteriori (MAP).
  - Beta distribution as a prior gives a beta distribution as posterior ("$\propto$").
- We discussed modeling binary variables conditioned on features:
  - Tabular parameterization is flexible but has too many parameters.
  - Logistic regression is limited but has a linear number of parameters.

End of Part 1 ("Binary Variables"): Key Concepts

- We discussed multivariate binary density estimation.
  - Refined inference tasks when we have more than one random variable:
    - Joint probability, marginal probability, and conditional probability.
  - Product of Bernoullis assumes variables are independent.
    - Fast inference/learning but a strong assumption.
- We discussed generative classifiers:
  - Build a model of the joint probability of features and labels.
    - Compared to usual discriminative classifiers that model labels given features.
  - Naïve Bayes assumes features are independent given label.
- We discussed neural networks:
  - Model that learns the features and classifier simultaneously.
  - Alternate between linear and non-linear transformations (universal approximator).
  - Training is a non-convex problem, but SGD often works better than expected:
    - For large-enough networks we often find global, and SGD seems to have implicit regularization.

End of Part 1 (“Binary Variables”): Key Concepts

- We discussed deep learning with multiple hidden layers.
  - Biological motivations and efficient representation of some functions.
  - Vanishing gradient problem and modern solutions:
    - ReLU, skip connections, ResNets.
- We discussed automatic differentiation to generate gradient code.
  - Code that generates gradient code for you (using chain rule).
- We discussed convolutional neural networks (CNNs):
  - Include convolution layers that measure image features.
  - Include max pooling layers that highlight top features across space.
  - Reduces number of parameters and gives some spatial invariance.

End of Part 1 ("Binary Variables"): Key Concepts

- We discussed autoencoders:
  - Networks where the output is the input.
  - Encodes input into a bottleneck layer, then decodes back to input.
  - Non-linear dimensionality reduction.
  - Denoising autoencoders learn to enhance images.
- We discussed multi-label classification:
  - Where each training examples can have 0-k correct labels.
  - We discussed an encoding approach where the classes shares hidden layers.
    - Reduces parameters and captures dependencies between labels.
  - We discussed pre-training to learn new tasks with fewer labeled examples.
- We discussed pixel labeling:
  - Fully-convolutional networks maintain spatial information at all layers.
    - Requires upsampling to original image size.
    - Can label images of different sizes.

Next Topic: Categorical Variables

Motivating problem: Political Polling

- Want to know support for political parties among a voter group.
  - What percentage will vote the Liberal party? Conservative party? NDP?
    - What to know support for each party, since may want to attract voters?

- Where I live the last election results were:
  - 34.4% LIB
  - 33.5% NDP
  - 26.8% CPC
  - 2.9% GRN
  - 2.4% PPC

- We want to predict these quantities based on a sample (“poll”).

General Problem: Categorical Density Estimation

- This is a special case density estimation with a categorical variable:
  - Input: 'n' IID samples of categorical values x^1, x^2, x^3,…, x^n from a population.
  - Output: model of probability that x=1, x=2, x=3,…,x=k.
- Categorical density estimation as a picture:

X =

Party?
LIB
CPC
NDP
LIB
GRN

p(x = LIB) = 0.34, p(x=NDP) = 0.34,
p(x = CPC) = 0.27, p(x=GRN) = 0.03,
p(x = PPC) = 0.02.

- We are going to revisit many of our previous concepts in this case.
  - Again building up to more-complicated cases.
    - And introducing some concepts that we skipped in Part 1.

Other Applications of Categorical Density Estimation

- Other applications where categorical density estimation is useful:
  - What portion of my clients use cash, credit, or debit?
  - Prevalence of different blood types.
  - Probability of having different types of cancers.
  - Probability of seeing different words (natural language processing).

- For categorical variables, we do not assume there is an ordering.
  - Cateogry 4 is not “closer” to category 3 than it is to category 1.

Ordinal Variables

- Categorical variables with an ordering are called ordinal:
  - Dice (1, 2, 3, 4, 5, 6).
    - Though I may use dice to illustrate categorical ideas.
  - Survey results (“strongly disagree”, “disagree”, “neutral”,…).
  - Ratings (1 star, 2 star,…).
  - Tumour grading (Grade I, Grade II, Grade III, Grad IV).

- We will not cover ordinal variables, but several methods exist.
  - Such as “ordinal logistic regression”.
    - A loss function that reflects that “2 stars” is closer to “3 stars” than “4 stars”.
      - But the distances between adjacent “stars” may not be the same.
    - That loss function can be used in place of “softmax” in neural nets with ordinal labels.

Summary

- Pixel classification:
  - Assigning a label to every pixel in an image.
- 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.
  - Transpose convolutions learn upsampling operators.
- Categorical density estimation:
  - Modeling probabilities of several unordered categories.

- Next time: the second-most popular type of algorithm in ML.