CPSC 540: Machine Learning

Fully-Convolutional Networks

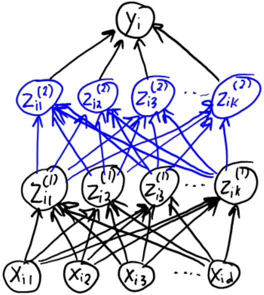
Mark Schmidt

University of British Columbia

Winter 2018

Last Time: Deep Neural Networks

- We reviewed deep neural networks, where we have multiple hidden layers of learned features:



- Mathematically, with 3 hidden layers the classic model uses

$$\hat{y}^i = v^T h(W^3 h(W^2 h(W^1 x^i)).$$

- Can be viewed as a DAG model where inference is easy.
    - Due to deterministic connections leading into hidden variables.

Training Deep Neural Networks

- If we're training a 3-layer network with squared error, our objective is

$$f(v, W^1, W^2, W^3) = \frac{1}{2}\sum_{i=1}^{n}(\underbrace{v^T h(W^3 h(W^2 h(W^1 x^i)))}_{\hat{y}^i} - y^i)^2.$$

- Usual training procedure is stochastic gradient.
  - But we're discovering sets of tricks to make things easier to tune.

- Highly non-convex and notoriously difficult to tune.

- Recent empirical/theoretical work indicates non-convexity may not be an issue:
  - All local minima may be good for "large enough" networks.

Training Deep Neural Networks

- Some common data/optimization tricks we discussed in 340:
  - Data transformations.
    - For images, translate/rotate/scale/crop each x^i to make more data.
  - Data standardization: centering and whitening.
  - Adding bias variables.
  - Parameter initialization: "small but different", standardizing within layers.
  - Step-size selection: "babysitting", Bottou trick, Adam.
  - Momentum: heavy-ball and Nesterov-style modifications.
  - Batch normalization: adaptive standardizing within layers.
  - ReLU: replacing sigmoid with $\max\{0, w_c^T x^i\}$.
    - Avoids gradients extremely-close to zero.

Training Deep Neural Networks

- Common forms of regularization:
  - Standard L2-regularization or L1-regularization "weight decay".
    - Sometimes with different λ for each layer.
  - Early stopping of the optimization based on validation accuracy.
  - Dropout randomly zeroes z values to discourage dependence.
  - Hyper-parameter optimization to choose various tuning parameters.
  - Special architectures like convolutional neural networks:
    - Yields W^m that are very sparse and have many tied parameters.
- Recent tricks based on changing graph structure (adding edges to DAG):
  - Residual networks: include inputs from previous layers.
    - Doesn't need to "memorize input in the output".
  - Dense networks: connect to inputs from many previous layers.

Backpropagation as Message-Passing

- Computing the gradient in neural networks is called backpropagation.
  - Derived from the chain rule and memoization of repeated quantities.
- We're going to view backpropagation as a message-passing algorithm.
- Key advantages of this view:
  - It's easy to handle different graph structures.
  - It's easy to handle different non-linear transformations.
  - It's easy to handle multiple outputs (as in structured prediction).
  - It's easy to add non-deterministic parts and combine with other graphical models.

Backpropagation Forward Pass

- Consider computing the output of a neural network for an example i,

$$
\begin{aligned}
y^i &= v^T h(W^3 h(W^2 h(W^1 x^i))) \\
&= \sum_{c=1}^{k} v_c h\left(\sum_{c'=1}^{k} W^3_{c'c} h\left(\sum_{c''=1}^{k} W^2_{c''c'} h\left(\sum_{j=1}^{d} W^1_{c''j} x^i_j\right)\right)\right).
\end{aligned}
$$

  where we've assume that all hidden layers have k values.
  - In the second line, the h functions are single-input single-output.

- The nested sum structure is similar to our message-passing structures.

- However, it's easier because it's deterministic: no random variables to sum over.
  - The messages will be scalars rather than functions.

Backpropagation Forward Pass

- Forward propagation through neural network as message passing:

$$
\begin{aligned}
y^i &= \sum_{c=1}^{k} v_c h\left(\sum_{c'=1}^{k} W^3_{c'c} h\left(\sum_{c''=1}^{k} W^2_{c''c'} h\left(\sum_{j=1}^{d} W^1_{c''j} x^i_j\right)\right)\right) \\
&= \sum_{c=1}^{k} v_c h\left(\sum_{c'=1}^{k} W^3_{c'c} h\left(\sum_{c''=1}^{k} W^2_{c''c'} h(M_{c''})\right)\right) \\
&= \sum_{c=1}^{k} v_c h\left(\sum_{c'=1}^{k} W^3_{c'c} h(M_{c'})\right) \\
&= \sum_{c=1}^{k} v_c h(M_c) \\
&= M_y,
\end{aligned}
$$

where intermediate messages are the z values.

Backpropagation Backward Pass

- The backpropagation backward pass computes the partial derivatives.
  - For a loss f, the partial derivatives in the last layer have the form

$$\frac{\partial f}{\partial v_c} = z_c^{i3} f'(v^T h(W^3 h(W^2 h(W^1 x^i)))),$$

where

$$z_{c'}^{i3} = h\left(\sum_{c'=1}^{k} W_{c'c}^3 h\left(\sum_{c''=1}^{k} W_{c''c'}^2 h\left(\sum_{j=1}^{d} W_{c''j}^1 x_j^i\right)\right)\right).$$

  - Written in terms of messages it simplifies to

$$\frac{\partial f}{\partial v_c} = h(M_c) f'(M_y).$$

Backpropagation Backward Pass

- In terms of forward messages, the partial derivatives have the forms:

$$\frac{\partial f}{\partial v_c} = h(M_c)f'(M_y),$$

$$\frac{\partial f}{\partial W^3_{c'c}} = h(M_{c'})h'(M_c)w_c f'(M_y),$$

$$\frac{\partial f}{\partial W^2_{c''c'}} = h(M_{c''})h'(M_{c'})\sum_{c=1}^{k} W^3_{c'c}h'(M_c)w_c f'(M_y),$$

$$\frac{\partial f}{\partial W^1_{jc''}} = h(M_j)h'(M_{c''})\sum_{c'=1}^{k} W^2_{c''c'}h'(M_{c'})\sum_{c=1}^{k} W^3_{c'c}h'(M_c)w_c f'(M_y),$$

which are ugly but notice all the repeated calculations.

Backpropagation Backward Pass

- It's again simpler using appropriate messages

$$\frac{\partial f}{\partial v_c} = h(M_c) f'(M_y),$$

$$\frac{\partial f}{\partial W^3_{c'c}} = h(M_{c'}) h'(M_c) w_c V_y,$$

$$\frac{\partial f}{\partial W^2_{c''c'}} = h(M_{c''}) h'(M_{c'}) \sum_{c=1}^{k} W^3_{c'c} V_c,$$

$$\frac{\partial f}{\partial W^1_{jc''}} = h(M_j) h'(M_{c''}) \sum_{c'=1}^{k} W^2_{c''c'} V_{c'},$$

where $M_j = x_j$.

Backpropagation as Message-Passing

- The general forward message for child c with parents p and weights W is

$$M_c = \sum_p W_{cp} h(M_p),$$

  which computes weighted combination of non-linearly transformed parents.
  - In the first layer we don't apply h to x.
- The general backward message from child c to *all* its parents is

$$V_c = h'(M_c) \sum_{c'} W_{cc'} V_{c'},$$

  which weights the "grandchildren's gradients".
  - In the last layer we use f instead of h.
- The gradient of W_{cp} is $h(M_p)V_c$, which works for general graphs.

Neural Networks + CRFs = Conditional Neural Fields

- Last time we saw conditional random fields like

$$p(y \mid x) \propto \exp\left(\sum_{c=1}^{k} y_c v^T x_c + \sum_{(c,c')\in E} y_c y_{c'} w\right),$$

which can use logistic regression at each location c and Ising dependence on y_c.

- Instead of logistic regression, you could put a neural network in there:

$$p(y \mid x) \propto \exp\left(\sum_{c=1}^{k} y_c v^T h(W^3 h(W^2(W^1 x_c))) + \sum_{(c,c')\in E} y_c y_{c'} w\right).$$

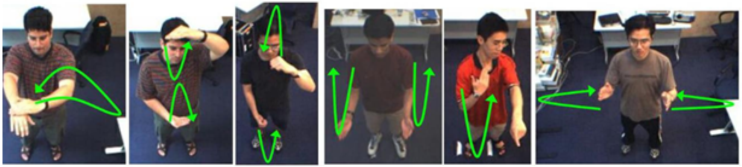
- Sometimes called a conditional neural field (CNF), and backprop generalizes:
  1. Forward pass through neural network to get y_c predictions.
  2. Belief propagation to get marginals of y_c (or Gibbs samplign if high treewidth).
  3. Backwards pass through neural network to get all gradients.

Beyond Combining CRFs and Neural Nets

- Conditional random fields combine UGMs with supervised learning.
- Conditioanl neural fields add deep learning to the mix.
  - Many variations exist and are possible.
- But we said that UGMs are more powerful when combined with other tricks:
  - Mixture models, latent factors, approximate inference.

Motivation: Gesture Recognition

- Want to recognize gestures from video:

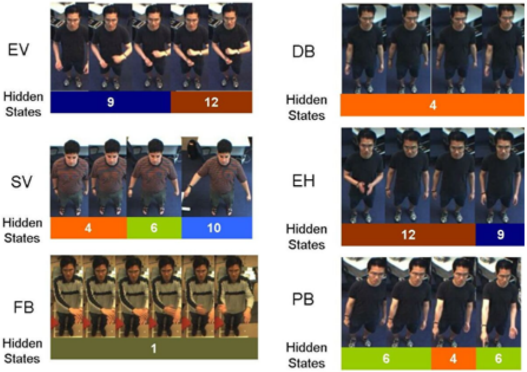


http://groups.csail.mit.edu/vision/vip/papers/wang06cvpr.pdf

- A gesture is composed of a sequence of parts:
  - And some parts appear in different gestures.

Motivation: Gesture Recognition

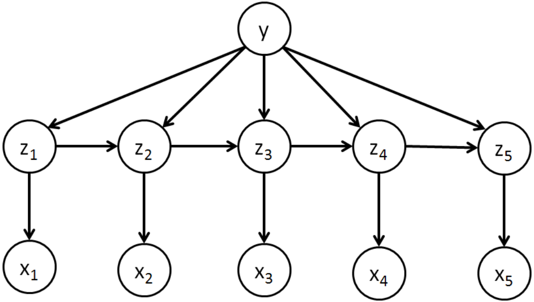
- We have a label for the whole sequence ("gesture") but no part labels.
  - We don't even know the set of possible parts.



http://groups.csail.mit.edu/vision/vip/papers/wang06cvpr.pdf

Generative Classifier based on an HMM

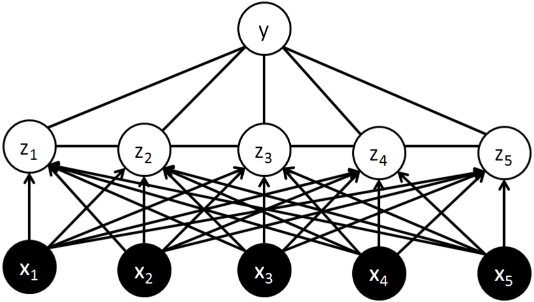
- We could address this scenario using a generative HMM model.



- Observed variable x_j is the image at time j (in this case x_j is a video frame).
- The gesture y is defined by sequence of parts z_j.
  - And we're learning what the parts should be.
- But modelling $p(x_j \mid z_j)$ is hard (probability of video frame given the hidden part).

Hidden Conditional Random Field

- A discriminative alternative is a hidden conditional random field.

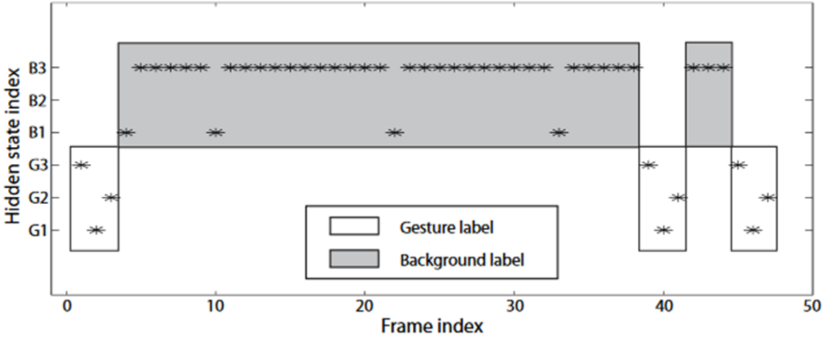


- The label y is based on a "hidden" CRF on the z_j values.
    - Again learns the parts as well as their temporal dependence.

- Treats the x_j as fixed so we don't need to model the video.

Motivation: Gesture Recognition

- What if we want to label video with multiple potential gestures?
  - We're given a labeled video sequence.

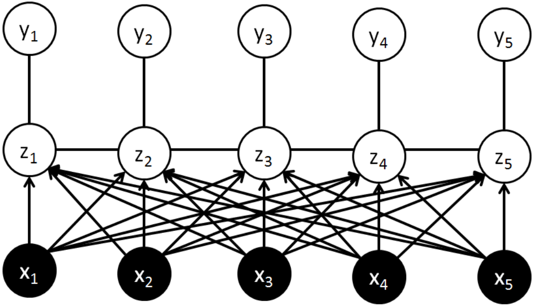


http://www.lsi.upc.edu/~aquattoni/AllMyPapers/cvpr_07_L.pdf

- Our videos are labeled with "gesture" and "background" frames,
  - But we again don't know the parts (G1, G2, G3, B1, B2, B3) that define the labels.

Latent-Dynamic Conditional Random Field

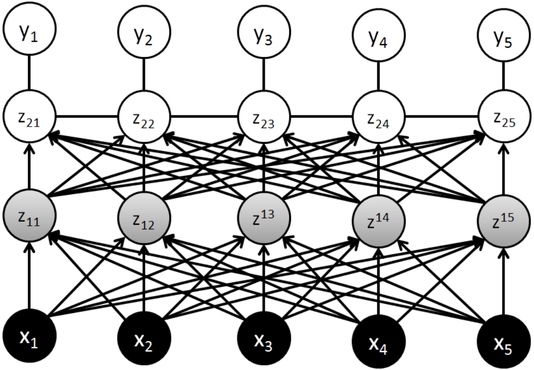
- Here we could use a latent-dynamic conditional random field



- The z_j still capture "latent dynamics", but we have a label y_j for each time.
- Notice in the above case that the conditional UGM is a tree.

Latent-Dynamic Conditional Neural Field

- Latent dynamic conditional neural fields also learn features with a neural network..

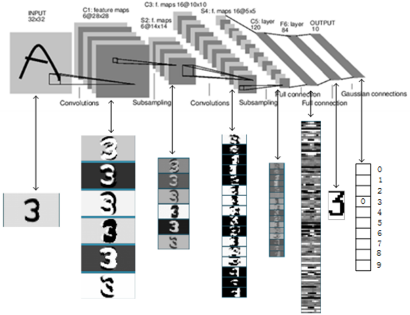


- Combines deep learning, mixture models, and graphical models.
  - Achieved among state of the art in several applications.

Outline

Convolutional Neural Networks

- In 340 we discussed convolutional neural networks (CNNs):

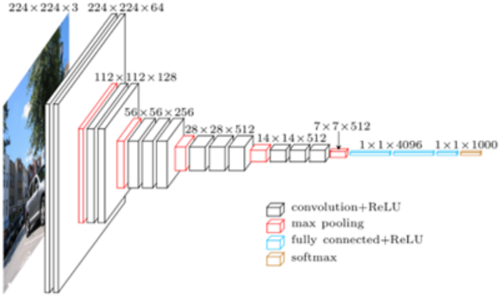


http://blog.csdn.net/strint/article/details/44163869

- Convolutional layers where W acts like a convolution (sparse with tied parameters).
- Pooling layers that usually take maximum among a small spatial neighbourhood.
- Fully-connected layers that use an unrestricted W.

Motivation: Beyond Classification

- Convolutional structure simplifies the learning task:
  - Parameter tieing means we have more data to estimate each parameter.
  - Sparsity drastically reduces number of parameters.

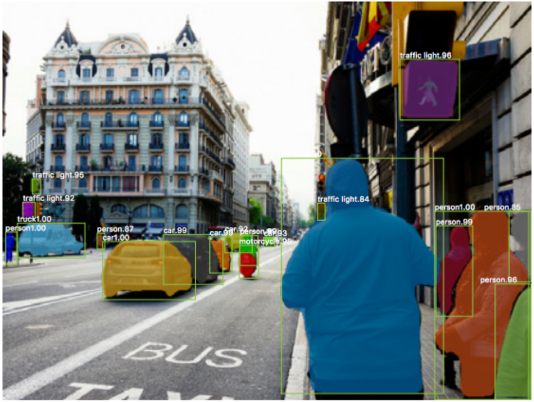


https://www.cs.toronto.edu/~frossard/post/vgg16

- We discussed CNNs for image classification: "is this an image of a cat?".
  - But many vision tasks are not image classification tasks.

Object Localization

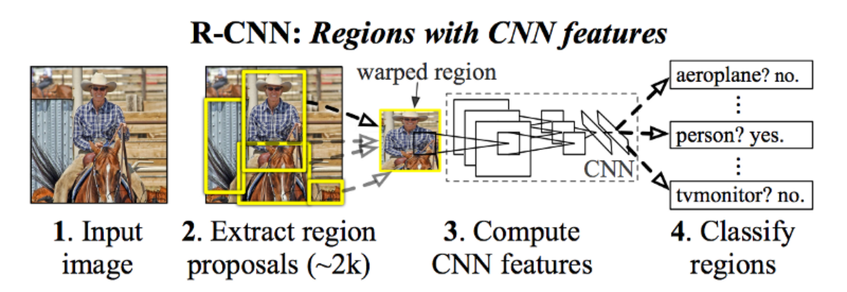
- Object localization is task of finding locations of objects:
  - Need to find *where* in the image the object is.
  - May need to recognize *more than one* object.



https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4

Region Convolutional Neural Networks: "Pipeline" Approach

- Early approach (region CNN):
  1. Propose a bunch of potential boxes.
  2. Compute features of box using a CNN.
  3. Classify each box based on an SVM.
  4. Refine each box using linear regression.

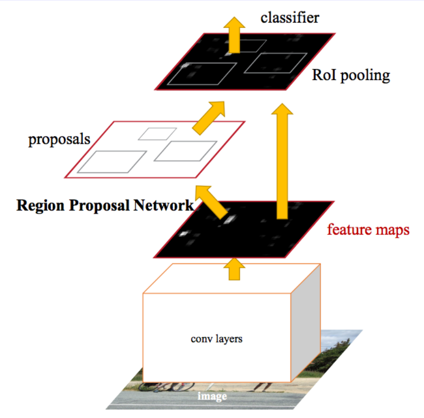


https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4

- Improved on state of the art, but not very elegant with its 4 steps.

Region Convolutional Neural Networks: "End to End" Approach

- Modern approaches try to do the whole task with one neural network.
  - The network extracts features, proposes boxes, and classifies boxes.



https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4

- This is called an end-to-end model.

End-to-End Computer Vision Models

- Key ideas behind end-to-end systems:
  1. Write each step as a differentiable operator.
  2. Train all steps using backpropagation and stochastic gradient.

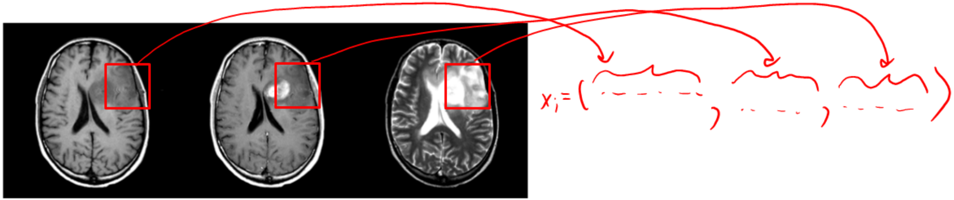
- There now exist end-to-end models for all the standard vision tasks.
  - Depth estimation, pose estimation, optical flow, tracking, 3D geometry, and so on.
  - A bit hard to track the progress at the moment.
  - A survey of ≈ 200 papers from 2016:
    - `http://www.themtank.org/a-year-in-computer-vision`

  .

- Let's focus on the task of pixel labeling...

Straightforward CNN Extensions to Pixels Labeling

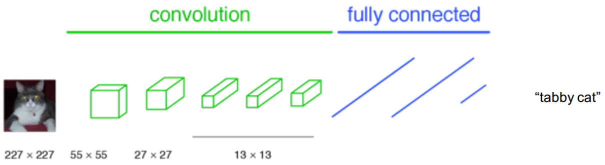
- Approach 1: apply an existing CNN to classify pixel given neighbourhood.
  - Misses long range dependencies in the image.
  - It's slow: for 200 by 200 image, need to do forward propagation 40000 times.



- Approach 2: add per-pixel labels to final layer of an existing CNN.
  - Fully-connected layers lose spatial information.
  - Relies on having fixed-size images.

Fully-Convolutional Neural Networks

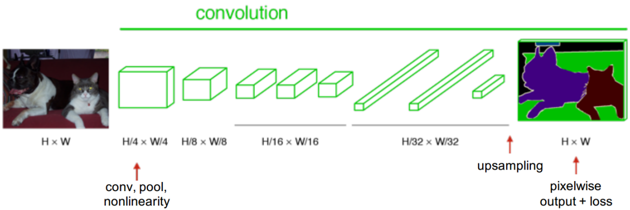
- Classic CNN architecture:



https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/image_segmentation.html

Fully-Convolutional Neural Networks

- Fully-convolutional neural networks (FCNs): CNNs with no fully-connected layers.
  - All layers maintain spatial information.



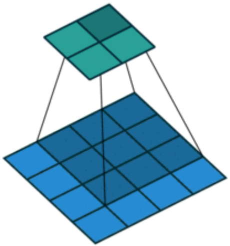
https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/image_segmentation.html

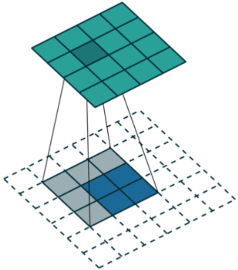
- Final layer upsamples to original image size.
  - With a learned "transposed convolution".

- Parameter tieing within convolutions allows images of different sizes.

Transposed Convolution Layer

- The upsampling layer is also called a transposed convolution or "deconvolution".
  - Implemented as another convolution.

Convolution: 

Transposed: 

https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/image_segmentation.html

- Reasons for the names:
  - "Tranposed" because sparsity pattern is transpose of a downsampling convolution.
  - "Deconvolution" is not related to the "deconvolution" in signal processing.

Fully-Convolutional Neural Networks

- FCNs quickly achieved state of the art results on many tasks.

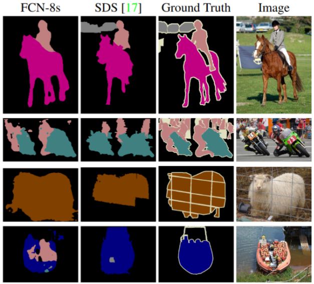


Figure 6. Fully convolutional segmentation nets produce state-of-the-art performance on PASCAL. The left column shows the output of our highest performing net, FCN-8s. The second shows the segmentations produced by the previous state-of-the-art system

https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf

- FCN end-to-end solution is very elegant compared to previous "pipeplines":
  - No super-pixels, object proposals, merging results from multiple classifiers, and so on.

Variations on FCNs

- The transposed convolution at the last layer can lose a lot of resolution.
- One option is adding “skip” connections from earlier higher-resolution layers.

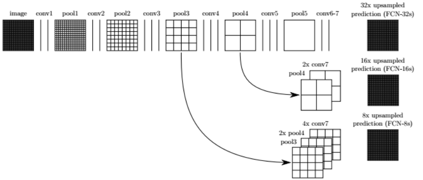
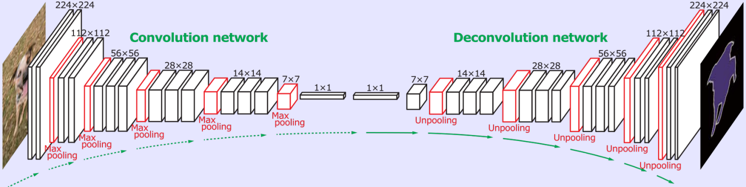


Figure 3. Our DAG nets learn to combine coarse, high layer information with fine, low layer information. Pooling and prediction layers are shown as grids that reveal relative spatial coarseness, while intermediate layers are shown as vertical lines. First row (FCN-32s): Our single-stream net, described in Section 4.1, upsamples stride 32 predictions back to pixels in a single step. Second row (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. Third row (FCN-8s): Additional predictions from pool3, at stride 8, provide further precision.

https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf

- Another framework addressing this is deconvolutional networks:



Combining FCNs and CRFs

- Another way to address this is combining FCNs and CRFs.

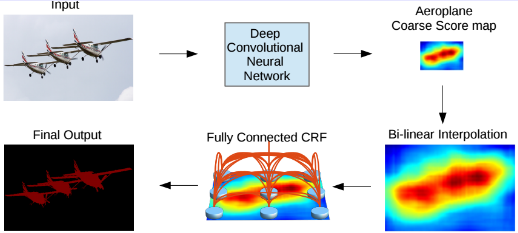


Fig. 1: Model Illustration. A Deep Convolutional Neural Network such as VGG-16 or ResNet-101 is employed in a fully convolutional fashion, using atrous convolution to reduce the degree of signal downsampling (from 32x down 8x). A bilinear interpolation stage enlarges the feature maps to the original image resolution. A fully connected CRF is then applied to refine the segmentation result and better capture the object boundaries.

https://arxiv.org/pdf/1606.00915.pdf

- DeepLab uses a fully-connected pairiwse CRF on output layer.
  - Though most recent version removed CRF.

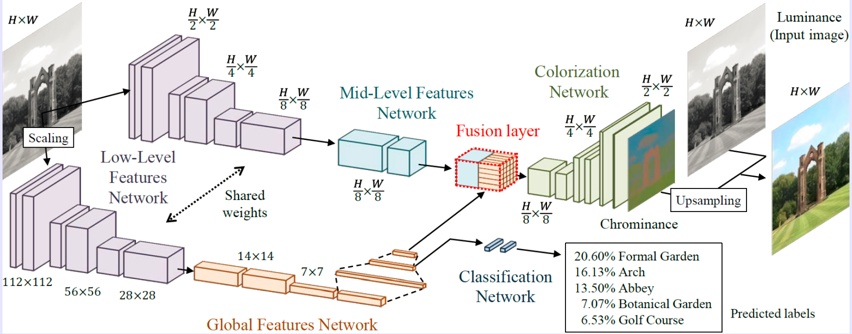
R-CNNs for Pixel Labeling

- An alternative approach: learn to apply binary mask to R-CNN results:

https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4

Image Colourization

- An end-to-end image colorization network:



http://hi.cs.waseda.ac.jp/~iizuka/projects/colorization/en

- Trained to reproduce colour of existing images after removing colour.

Image Colourization

- Image colorization results:

Colorado National Park, 1941 Textile Mill, June 1937 Berry Field, June 1909 Hamilton, 1936

http://hi.cs.waseda.ac.jp/~iizuka/projects/colorization/en

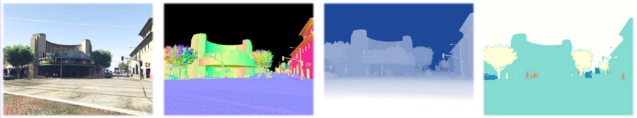
- Gallery:
  http://hi.cs.waseda.ac.jp/~iizuka/projects/colorization/extra.html
- Video: https://www.youtube.com/watch?v=ys5nMO4Q0iY

Where does data come from?

- Unfortunately, getting densely-labeled data is often hard.

- For pixel labeling and depth estimation, we explored getting data from GTA V:

Video game   Google street view



- Easy to collect data at night, in fog, or in dangerous situations.

Where does data come from?

- Recent works use that you don't need full labeling.
  - Unobserved children in DAG don't induce dependencies.
    - Although you would do better if you have an accurate dense labeling.

- Test object segmentation based on "single pixel" labels from training data:

- Show video...

Summary

- Backpropagation can be viewed as a message passing algorithm.
- Conditional neural fields combine CRFs with deep learning.
  - You can learn the features and the label dependency at the same time.
- End to end models: use a neural network to do all steps.
  - Computer vision can now actually work!
- Fully-convolutional networks:
  - Elegant way to apply convolutional networks for dense labeling problems.
- Next time: generating poetry, music, and dance moves.