Recap: R-CNN for Object Detection

Recap: Faster R-CNN

One network, four losses
- Remove dependence on external region proposal algorithm.
- Instead, infer region proposals from same CNN.
- Feature sharing
- Joint training
⇒ Object detection in a single pass becomes possible.

Recap: Mask R-CNN


Recap: YOLO / SSD

- Idea: Directly go from image to detection scores
- Within each grid cell
  - Start from a set of anchor boxes
  - Regress from each of the B anchor boxes to a final box
  - Predict scores for each of C classes (including background)
Topics of This Lecture

• Practical Advice on CNN training
  - Data Augmentation
  - Initialization
  - Batch Normalization
  - Dropout
  - Learning Rate Schedules
• CNNs for Segmentation
  - Fully Convolutional Networks (FCN)
  - Encoder-Decoder architecture
  - Transpose convolutions
  - Skip connections
• CNNs for Human Body Pose Estimation

Data Augmentation

• Idea
  - Augment original data with synthetic variations to reduce overfitting
• Example augmentations for images
  - Cropping
  - Zooming
  - Flipping
  - Color PCA

Data Augmentation

• Effect
  - Much larger training set
  - Robustness against expected variations
• During testing
  - When cropping was used during training, need to again apply crops to get same image size.
  - Beneficial to also apply flipping during test.
  - Applying several ColorPCA variations can bring another ~1% improvement, but at a significantly increased runtime.

Glorot Initialization

[Glrorot & Bengio, ’10]

• Variance of neuron activations
  - Suppose we have an input $X$ with $n$ components and a linear neuron with random weights $W$ that splits out a number $Y$.
  - We want the variance of the input and output of a unit to be the same, therefore $n \text{Var}(W_i)$ should be 1. This means
  $$\text{Var}(W_i) = \frac{1}{n} \frac{1}{n_{in}}$$
  - Or for the backpropagated gradient
    $$\text{Var}(W_i) = \frac{1}{n_{out}}$$
  - As a compromise, Glorot & Bengio propose to use
    $$\text{Var}(W) = \frac{2}{n_{in} + n_{out}}$$
  - Randomly sample the initial weights with this variance.

He Initialization

[He et al., ’15]

• Extension of Glorot Initialization to ReLU units
  - Use Rectified Linear Units (ReLU)
    $$g(a) = \max \{0, a\}$$
  - Effect: gradient is propagated with a constant factor
    $$\frac{\partial g(a)}{\partial a} = \begin{cases} 1, & a > 0 \\ 0, & \text{else} \end{cases}$$
  - Same basic idea: Output should have the input variance
    - However, the Glorot derivation was based on tanh units, linearity assumption around zero does not hold for ReLU.
    - He et al. made the derivations, proposed to use instead
      $$\text{Var}(W) = \frac{2}{n_{in}}$$

Practical Advice

• Initializing the weights
  - Draw them randomly from a zero-mean distribution.
  - Common choices in practice: Gaussian or uniform.
  - Common trick: add a small positive bias ($+\epsilon$) to avoid units with ReLU nonlinearities getting stuck-at-zero.
• When sampling weights from a uniform distribution $[a,b]$
  - Keep in mind that the standard deviation is computed as
    $$\sigma^2 = \frac{1}{12} (b - a)^2$$
  - Glorot initialization with uniform distribution
    $$W \sim U \left( -\frac{\sqrt{6}}{\sqrt{n_{in} + n_{out}}}, \frac{\sqrt{6}}{\sqrt{n_{in} + n_{out}}} \right)$$
Batch Normalization

Motivation
• Optimization works best if all inputs of a layer are normalized.

Idea
• Introduce intermediate layer that centers the activations of the previous layer per minibatch.
  • i.e., perform transformations on all activations and undo those transformations when backpropagating gradients
• Complication: centering + normalization also needs to be done at test time, but minibatches are no longer available at that point.
  • Learn the normalization parameters to compensate for the expected bias of the previous layer (usually a simple moving average)

• Effect
  • Much improved convergence (but parameter values are important!)
  • Widely used in practice

Dropout

Idea
• Randomly switch off units during training.
• Change network architecture for each data point, effectively training many different variants of the network.
• When applying the trained network, multiply activations with the probability that the unit was set to zero.

⇒ Greatly improved performance

Choosing the Right Learning Rate

Behavior for different learning rates

Reducing the Learning Rate

Final improvement step after convergence is reached
• Reduce learning rate by a factor of 10.
• Continue training for a few epochs.
• Do this 1-3 times, then stop training.

• Effect
  • Turning down the learning rate will reduce the random fluctuations in the error due to different gradients on different minibatches.
  • Be careful: Do not turn down the learning rate too soon!
  • Further progress will be much slower/impossible after that.

Summary
• Deep multi-layer networks are very powerful.
• But training them is hard!
  • Complex, non-convex learning problem
  • Local optimization with stochastic gradient descent
• Main issue: getting good gradient updates for the lower layers of the network
  • Many seemingly small details matter!
  • Weight initialization, normalization, data augmentation, choice of nonlinearities, choice of learning rate, choice of optimizer,…

⇒ Exercise 5 will guide you through those steps. Take advantage of it!
Topics of This Lecture

- Practical Advice on CNN training
  - Data Augmentation
  - Initialization
  - Batch Normalization
  - Dropout
  - Learning Rate Schedules
- CNNs for Segmentation
  - Fully Convolutional Networks (FCN)
  - Encoder-Decoder architecture
  - Transpose convolutions
  - Skip connections
- CNNs for Human Body Pose Estimation

Semantic Segmentation

- Semantic Segmentation
  - Label each pixel in the image with a category label
  - Don’t differentiate instances, only care about pixels
- Instance segmentation
  - Also give an instance label per pixel

Segmentation Idea: Sliding Window

- Problem
  - Very inefficient
  - No reuse of features between shared patches

Segmentation Idea: Fully-Convolutional Nets

- Design a network as a sequence of convolutional layers
  - To make predictions for all pixels at once
- Fully Convolutional Networks (FCNs)
  - All operations formulated as convolutions
  - Fully-connected layers become 1 x 1 convolutions
  - Advantage: can process arbitrarily sized images

CNNs vs. FCNs

- CNN
- FCN
- Intuition
  - Think of FCNs as performing a sliding-window classification, producing a heatmap of output scores for each class
  - But: more efficient, since computations are reused between windows

Segmentation Idea: Fully-Convolutional Nets

- Design a network as a sequence of convolutional layers
  - To make predictions for all pixels at once
- Problem
  - Convolutions at original image resolution will be very expensive!
Segmentation Idea: Fully-Convolutional Nets

- Design a network as a sequence of convolutional layers
  - With downsampling and upsampling inside the network!
  - Downsampling:
    - Pooling, strided convolution
  - Upsampling:
    - ???

In-Network Upsampling: "Unpooling"

- Nearest-Neighbor
  - Simplest version
  - Problem: blocky output structure
- "Bed of Nails"
  - Preserve fine-grained structure of the output
  - Problem: fixed location for upsampled stimuli

Learnable Upsampling: Transpose Convolution

- Recall: Normal convolution, stride 2, pad 1
- Effect
  - Filter moves 2 pixels in the input for every one pixel in the output
  - Stride gives ratio between movement in input and output
Learnable Upsampling: Transpose Convolution

- Now: 3x3 transpose convolution, stride 2 pad 1

<table>
<thead>
<tr>
<th>Input: 2x2</th>
<th>Output: 4x4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Effect
  - Filter moves 2 pixels in the output for every one pixel in the input
  - Stride gives ration between movement in output and input

- Other names
  - Deconvolution (bad)
  - Upconvolution
  - Fractionally strided convolution
  - Backward strided convolution

Learnable Upsampling: 1D Example

- Observations
  - Output contains copies of the filter weighted by the input, summing overlaps in the output
  - Need to crop one pixel from output to make output exactly 2x input

Convolution as Matrix Multiplication (1D Example)

- Express convolution in terms of matrix multiplication
  - Example:
    - 1D conv
    - Kernel size = 3
    - Stride 1, padding = 1

- Convolution transpose multiplies by the transpose of the same matrix
  - When stride = 1, convolution transpose is just a regular convolution (with different padding rules)
Convolution as Matrix Multiplication (1D Example)

- Express convolution in terms of matrix multiplication
  - Example:
    - 1D conv
    - Kernel size = 3
    - Stride 2, padding = 1

- Convolution transpose multiplies by the transpose of the same matrix
  - When stride > 1, convolution transpose is no longer a normal convolution!

Segmentation Idea: Fully-Convolutional Nets

- Design a network as a sequence of convolutional layers
  - With downsampling and upsampling inside the network!
    - Downsampling
      - Pooling, strided convolution
    - Upsampling
      - Unpooling or strided transpose convolution

Extension: Skip Connections

- Encoder-Decoder Architecture with skip connections
  - Problem: downsampling loses high-resolution information
  - Use skip connections to preserve this higher-resolution information

Example: SegNet

- SegNet
  - Encoder-Decoder architecture with skip connections
  - Encoder based on VGG-16
  - Decoder using Max Unpooling
  - Output with K-class Softmax classification

Example: U-Net

- U-Net
  - Similar idea, popular in biomedical image processing
  - Encoder-Decoder architecture with skip connections

Semantic Segmentation

- Recent results
  - Based on an extension of ResNets for high-resolution segmentation

Perceptual and Sensory Augmented Computing
Computer Vision Summer '19
Topics of This Lecture

- Practical Advice on CNN training
  - Data Augmentation
  - Initialization
  - Batch Normalization
  - Dropout
  - Learning Rate Schedules

- CNNs for Segmentation
  - Fully Convolutional Networks (FCN)
  - Encoder-Decoder architecture
  - Transpose convolutions
  - Skip connections

- CNNs for Human Body Pose Estimation

FCNs for Human Pose Estimation

- Input data
  - Image
  - Keypoints
  - Labels

- Task setup
  - Annotate images with keypoints for skeleton joints
  - Define a target disk around each keypoint with radius \( r \)
  - Set the ground-truth label to 1 within each such disk
  - Infer heatmaps for the joints as in semantic segmentation

Heat Map Predictions from FCN

Example Results: Human Pose Estimation

More Recently: Parts Affinity Fields

- [https://www.youtube.com/watch?v=pW6nZXeWIGM](https://www.youtube.com/watch?v=pW6nZXeWIGM)

References

- ReLu

- Initialization
References and Further Reading

• Batch Normalization

• Dropout

References: Computer Vision Tasks

• Semantic Segmentation

• Human Body Pose Estimation
  ▶ S.E. Wei, V. Ramakrishna, T. Kanade, Y. Sheikh, Convolutional Pose Machines, CVPR 2016.
  ▶ B. Xiao, H. Wu, Y. Wei, Simple Baselines for Human Pose Estimation and Tracking, ECCV 2018. (Code)