Computer Vision – Lecture 13

Deep Learning IV

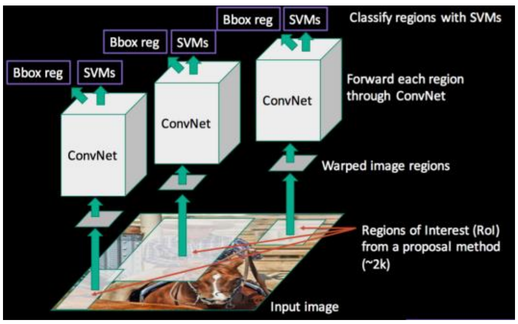
18.06.2019

Bastian Leibe
Visual Computing Institute
RWTH Aachen University
http://www.vision.rwth-aachen.de/

leibe@vision.rwth-aachen.de

Course Outline

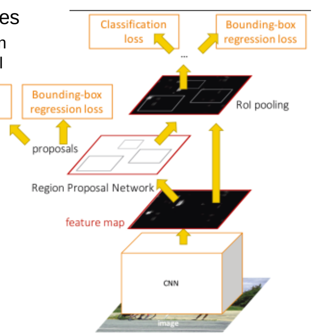
Recap: R-CNN for Object Detection



Slide credit: Ross Girshick

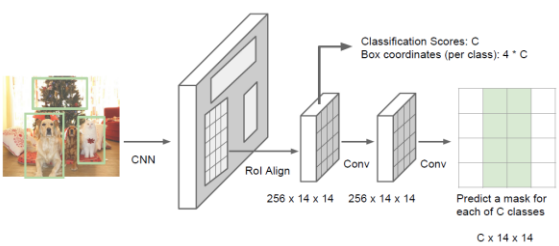
Recap: Faster R-CNN

- 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.



Slide credit: Ross Girshick

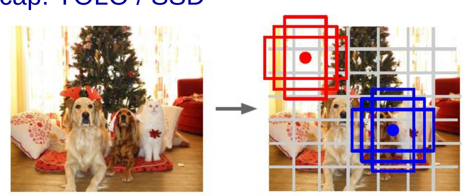
Recap: Mask R-CNN



K. He, G. Gkioxari, P. Dollar, R. Girshick, Mask R-CNN, arXiv 1703.06870.

Slide credit: FeiFei Li

Recap: YOLO / SSD



Input image
3 x H x W

Divide image into grid
7 x 7

- 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)

Slide credit: FeiFei Li

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

Image source: Lucas Beyer

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.

Augmented training data (from one original image)

Image source: Lucas Beyer

Glorot Initialization

[Glorot & Bengio, '10]

- Variance of neuron activations
  - Suppose we have an input X with n components and a linear neuron with random weights W that spits out a number Y.
  - We *want the variance of the input and output of a unit to be the same*, therefore $n\ \mathrm{Var}(W_i)$ should be 1. This means

$$\mathrm{Var}(W_i) = \frac{1}{n} = \frac{1}{n_{\text{in}}}$$

  - Or for the backpropagated gradient

$$\mathrm{Var}(W_i) = \frac{1}{n_{\text{out}}}$$

  - As a compromise, Glorot & Bengio propose to use

$$\mathrm{Var}(W) = \frac{2}{n_{\text{in}} + n_{\text{out}}}$$

$\Rightarrow$ Randomly sample the initial weights with this variance.

He Initialization

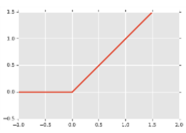
[He et al., '15]

- 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

$$\mathrm{Var}(W) = \frac{2}{n_{\text{in}}}$$

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 $(+\varepsilon)$ 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

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_{in} + n_{out}}}, \frac{\sqrt{6}}{\sqrt{n_{in} + n_{out}}}\right]$$

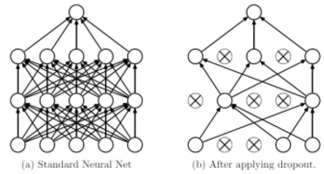
Batch Normalization

[Ioffe & Szegedy '14]

- 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

[Srivastava, Hinton '12]



- 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.

$\Rightarrow$ Greatly improved performance

Choosing the Right Learning Rate

- Behavior for different learning rates

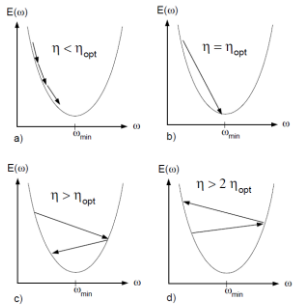


Image source: Yann LeCun et al., Efficient BackProp (1998)

Learning Rate vs. Training Error

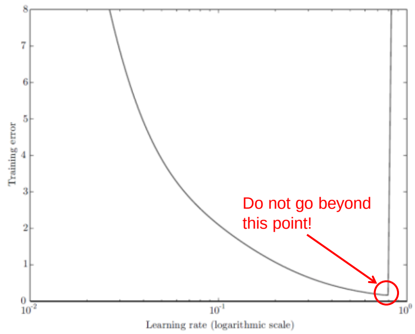
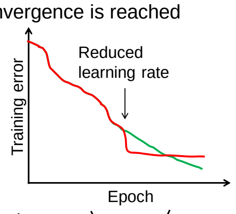


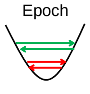
Image source: Goodfellow & Bengio book

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.

Slide adapted from Geoff Hinton

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,…

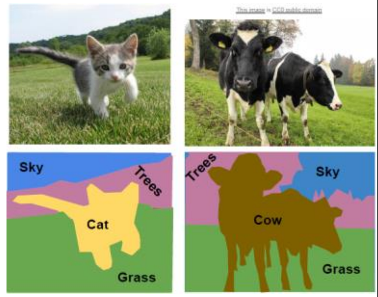
$\Rightarrow$ *Exercise 5 will guide you through those steps. Take advantage of it!*

Topics of This Lecture

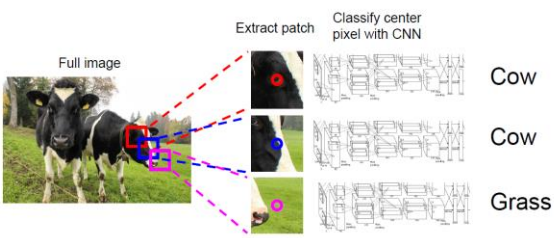
- Practical Advice on CNN training
  - Data Augmentation
  - Initialization
  - Batch Normalization
  - Dropout
  - Learning Rate Schedules
- CNNs for Segmentation
  - Fully Convolutional Networks (FCN)
  - Encoder-Decoder architecture
  - Transpose convolutions
  - Skip connections
- CNNs for Human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



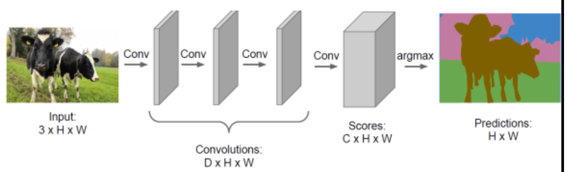
Segmentation Idea: Sliding Window



(e.g., AlexNet)

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

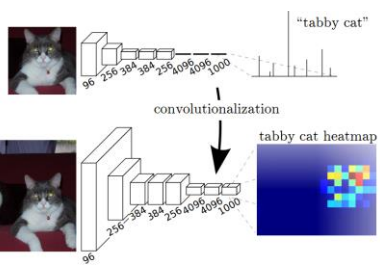
Segmentation Idea: Fully-Convolutional Nets



- 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 1×1 convolutions
    - Advantage: can process arbitrarily sized images

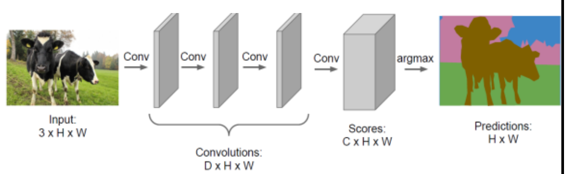
CNNs vs. FCNs

- CNN
- FCN



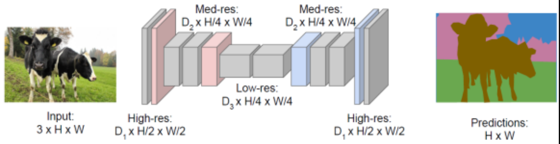
- Intuition
  - Think of FCNs as performing a sliding-window classification, producing a heatmap of output scores for each class
  - But: more efficient, since computations are reused between windows

Segmentation Idea: Fully-Convolutional Nets



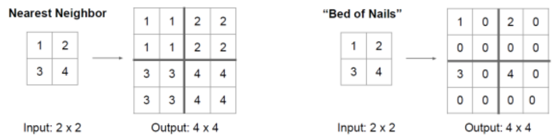
- Design a network as a sequence of convolutional layers
  - To make predictions for all pixels at once
- Problem
  - Convolutions at original image resolution will be very expensive!

Segmentation Idea: Fully-Convolutional Nets



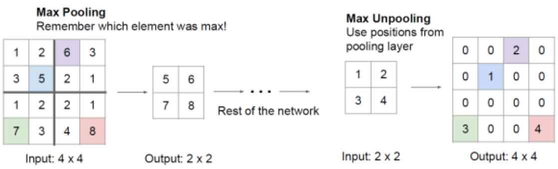
- Design a network as a sequence of convolutional layers
  - With downsampling and upsampling inside the network!
  - Downsampling
    - Pooling, strided convolution
  - Upsampling
    - ???

In-Network Upsampling: “Unpooling”

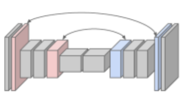


- Nearest-Neighbor
  - Simplest version
  - Problem: blocky output structure
- “Bed of Nails”
  - Preserve fine-grained structure of the output
  - Problem: fixed location for upsampled stimuli

In-Network Upsampling: “Max Unpooling”

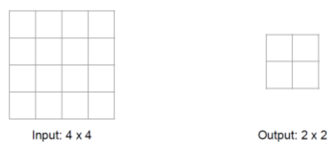


- Max Unpooling
  - Use corresponding pairs of downsampling and upsampling layers together
  - Remember which elements were max



Learnable Upsampling: Transpose Convolution

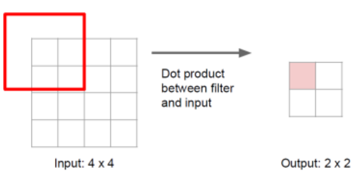
- Recall: Normal convolution, stride 2, pad 1



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

Learnable Upsampling: Transpose Convolution

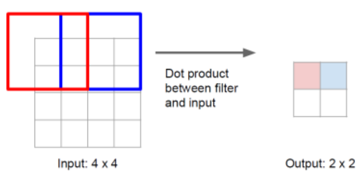
- Recall: Normal convolution, stride 2 pad 1



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

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 ration between movement in input and output

Learnable Upsampling: Transpose Convolution

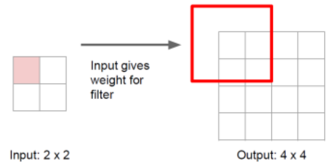
- Now: 3x3 transpose convolution, stride 2 pad 1

Computer Vision Summer'19

Slide credit: FeiFei Li

Learnable Upsampling: Transpose Convolution

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

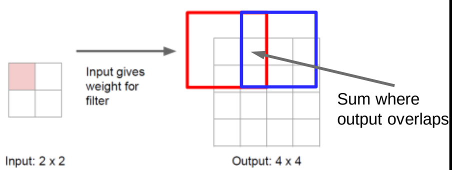


Computer Vision Summer'19

Slide credit: FeiFei Li

Learnable Upsampling: Transpose Convolution

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



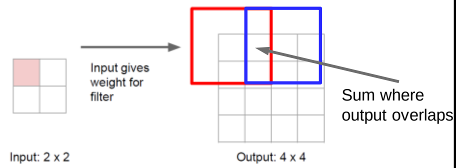
- Effect
  - Filter moves 2 pixels in the *output* for every one pixel in the *input*
  - Stride gives ration between movement in output and input

Computer Vision Summer'19

Slide credit: FeiFei Li

Learnable Upsampling: Transpose Convolution

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

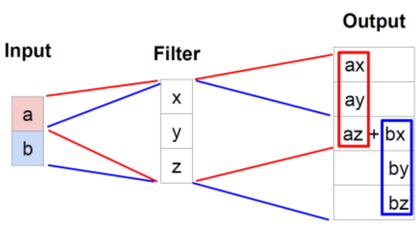


- Other names
  - Deconvolution (bad)
  - Upconvolution
  - Fractionally strided convolution
  - Backward strided convolution

Computer Vision Summer'19

Slide credit: FeiFei Li

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

Computer Vision Summer'19

Slide credit: FeiFei Li

Convolution as Matrix Multiplication (1D Example)

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

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & x & y & z & 0 & 0 \\ 0 & 0 & x & y & z & 0 \\ 0 & 0 & 0 & x & y & z \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

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

$$\vec{x} *^T \vec{a} = X^T\vec{a}$$

$$\begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay + bx \\ az + by + cx \\ bz + cy + dx \\ cz + dy \\ dz \end{bmatrix}$$

Computer Vision Summer'19

Slide credit: FeiFei Li

Convolution as Matrix Multiplication (1D Example)

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

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

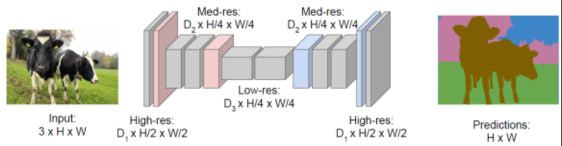
- Convolution transpose multiplies by the transpose of the same matrix
  - When stride > 1, convolution transpose is **no longer a normal convolution!**

$$\vec{x} *^T \vec{a} = X^T\vec{a}$$

$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

Slide credit: FeiFei Li

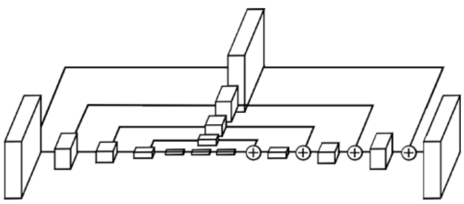
Segmentation Idea: Fully-Convolutional Nets



- 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

Slide credit: FeiFei Li

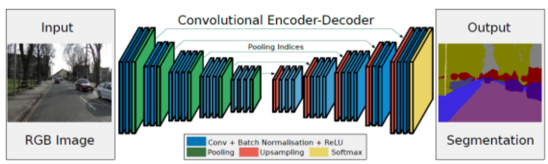
Extension: Skip Connections



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

Image source: Newell et al.

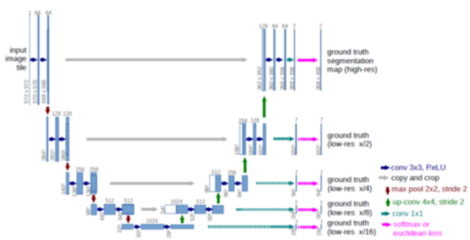
Example: SegNet



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

V. Badrinarayanan, A. Kendall, R. Cipolla, SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, arXiv 1511.00561, IEEE Trans. PAMI 2017.

Example: U-Net



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

O. Ronneberger, P. Fischer, T. Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, MICCAI 2015

Semantic Segmentation

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

[Pohlen, Hermans, Mathias, Leibe, CVPR 2017]

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

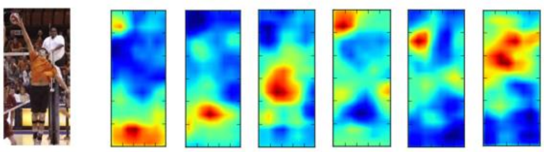
Image Keypoints Labels

- 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

Slide adapted from Georgia Gkioxari

Heat Map Predictions from FCN

Test Image Right Ankle Right Knee Right Hip Right Wrist Right Elbow Right Shoulder



Slide adapted from Georgia Gkioxari

Example Results: Human Pose Estimation

[Rafi, Gall, Leibe, BMVC 2016]

More Recently: Parts Affinity Fields

- https://www.youtube.com/watch?v=pW6nZXeWlGM

References

- ReLu
  - X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, AISTATS 2011.
- Initialization
  - X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, AISTATS 2010.
  - K. He, X.Y. Zhang, S.Q. Ren, J. Sun, Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, ArXiV 1502.01852v1, 2015.
  - A.M. Saxe, J.L. McClelland, S. Ganguli, Exact solutions to the nonlinear dynamics of learning in deep linear neural networks, ArXiV 1312.6120v3, 2014.

References and Further Reading

- Batch Normalization
  - S. Ioffe, C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, ArXiV 1502.03167, 2015.
- Dropout
  - N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, JMLR, Vol. 15:1929-1958, 2014.

References: Computer Vision Tasks

- Semantic Segmentation
  - J. Long, E. Shelhamer, T. Darrell, Fully Convolutional Networks for Semantic Segmentation, CVPR 2015.
  - O. Ronneberger, P. Fischer, T. Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, MICCAI 2015
  - V. Badrinarayanan, A. Kendall, R. Cipolla, SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, arXiv 1511.00561, IEEE Trans. PAMI 2017.
  - T-Y. Lin P. Dollar, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature Pyramid Networks for Object Detection, CVPR 2017.
  - L-C. Chen, G. Papandreou, F. Schroff, H. Adam, Rethinking Atrous Convolutions for Semantic Segmentation, arXiv 1706.05587 2017.

References: Computer Vision Tasks

- Human Body Pose Estimation
  - A. Toshev, C. Szegedy, DeepPose: Human Pose Estimation via Deep Neural Networks, CVPR 2014.
  - S.E. Wei, V. Ramakrishna, T. Kanade, Y. Sheikh, Convolutional Pose Machines, CVPR 2016.
  - A. Newell, K. Yang, J. Deng, Stacked Hourglass Networks for Human Pose Estimation, ECCV 2016.
  - Z. Cao, T. Simon, S.-E. Wei, Y. Sheikh, Realtime Multi-Person 2D Pose Estimation using Parts Affinity Fields, CVPR 2017.
  - B. Xiao, H. Wu, Y. Wei, Simple Baselines for Human Pose Estimation and Tracking, ECCV 2018. (Code)