Neural Networks and Deep Learning Lecture 7

Wei WANG

cs5242@comp.nus.edu.sg

National University of Singapore

School of Computing

Administrative

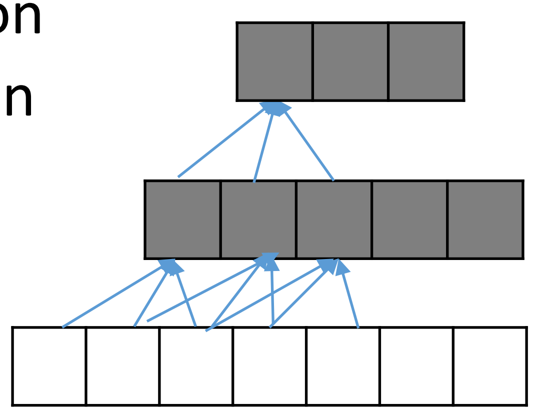
- Quiz on Week 9
  - 1 hour: 19:20-20:20
  - Closed book with one page cheat sheet
  - Scope: all materials from week 1 to week 9 (inclusive)
  - Question types: MCQ, True/False, Calculation, Explanation.

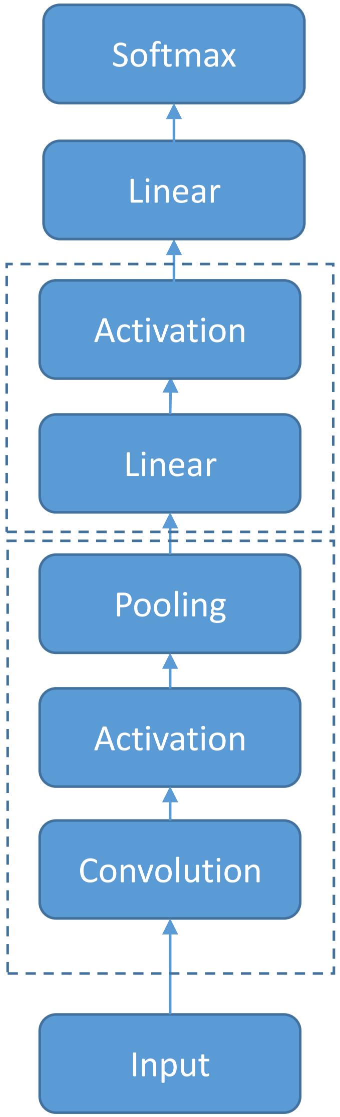
Outline

- CNN applications
  - Image classification
  - Object detection
  - Image segmentation
- Hands-on tutorial
  - Transfer learning
  - Real-time object detection
- Goals
  - know simple models for the three applications
  - Apply transfer learning for your own tasks

Recap

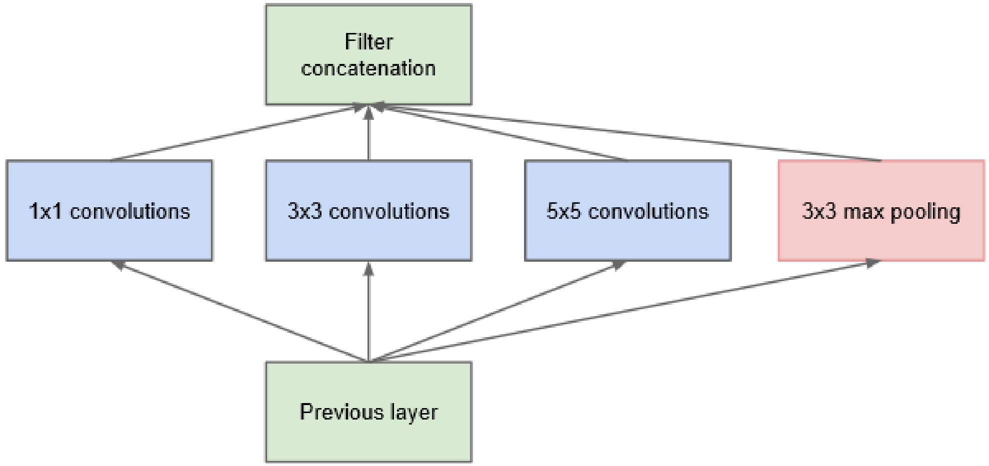
- AlexNet
  - Deep architecture over large dataset
  - Dropout
- VGG
  - The same filter size: 3x3 convolution
  - 2 3x3 convolution ~ 5x5 convolution



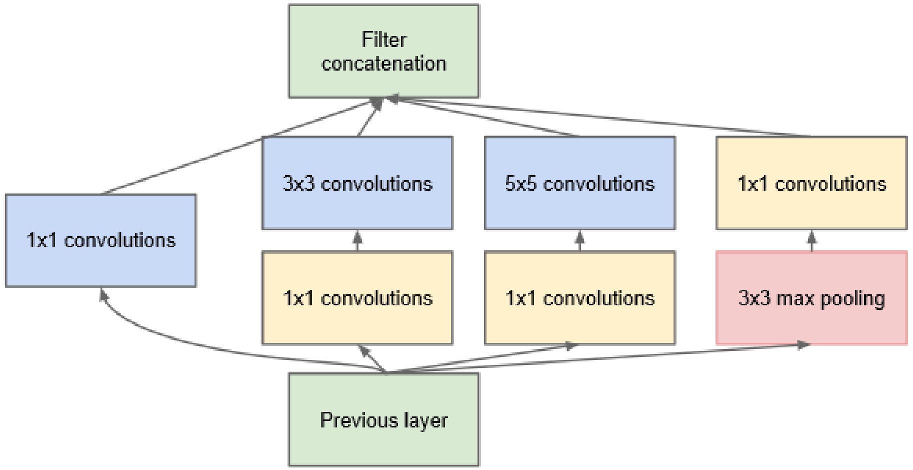


Recap

- InceptionNet
  - Mix 1x1, 3x3, 5x5 convolutions and max pooling
  - Add 1x1 convolution before others to reduce channels → save cost



(a) Inception module, naïve version



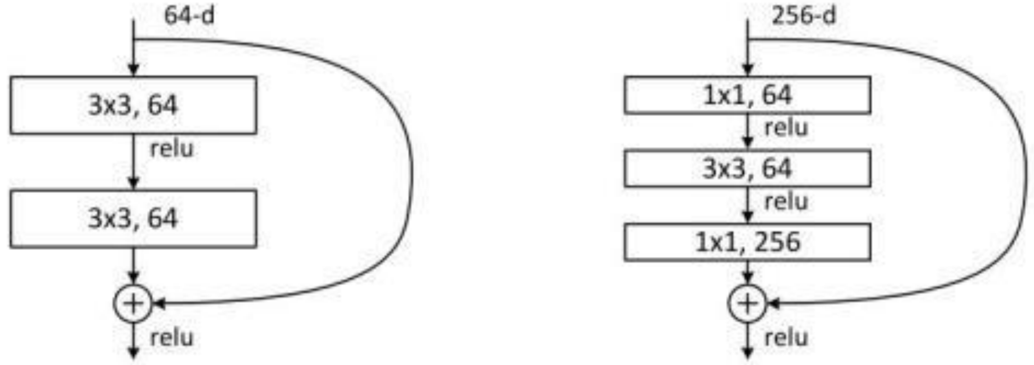
(b) Inception module with dimension reductions

Source: C. Szegedy et al., Going deeper with convolutions, CVPR 2015

Recap

- ResNet

Identity / skip-connection

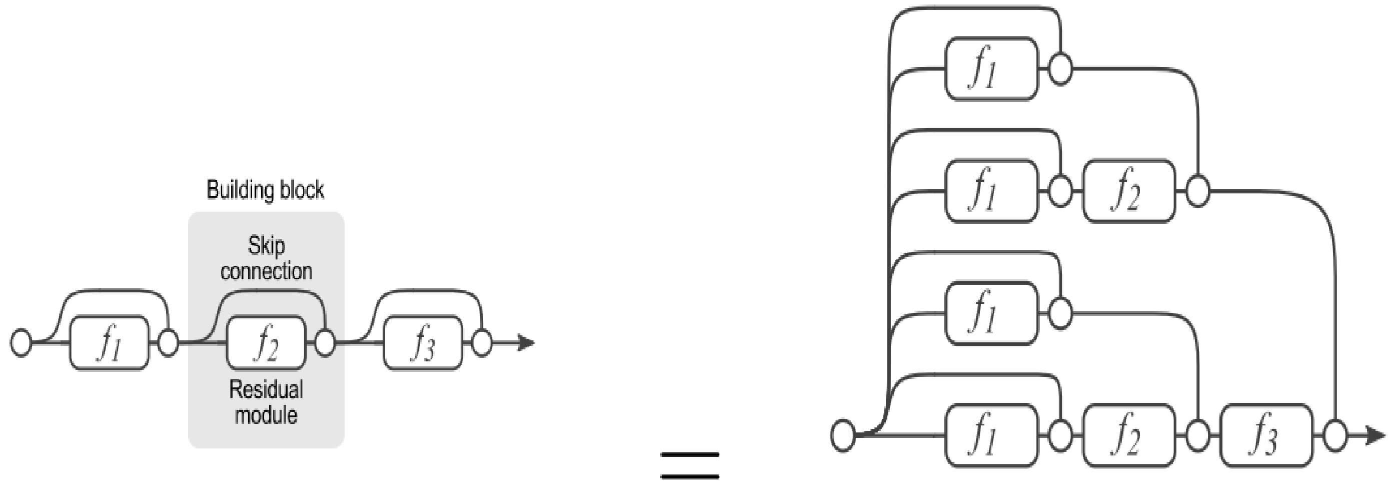


Source: Kaiming He, Xiangyu Zhang, Shaoqing Ren , and Jian Sun, Deep Residual Learning for Image Recognition, CVPR 2016

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}.$$

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}.$$

One convolution layer has stride=2 → feature map area halved → replace the identity connection with a linear transformation to make the feature maps of the same shape as those from the convolution layers



(a) Conventional 3-block residual network (b) Unraveled view of (a)

Source: A. Veit, M. Wilber and S. Belongie. Residual Networks Behave Like Ensembles of Relatively Shallow Networks. arXiv:1605.06431v2,2016

Applications

Image classification

- Predict the class/label of the image
- Training label
  - ground truth label (index)
- Test output
  - A probability distribution vector, one probability per label; sum up to 1

Source from [13]

Training label: bicycle
Prediction output:
bicycle 0.6; people 0.3; mountain 0.05;

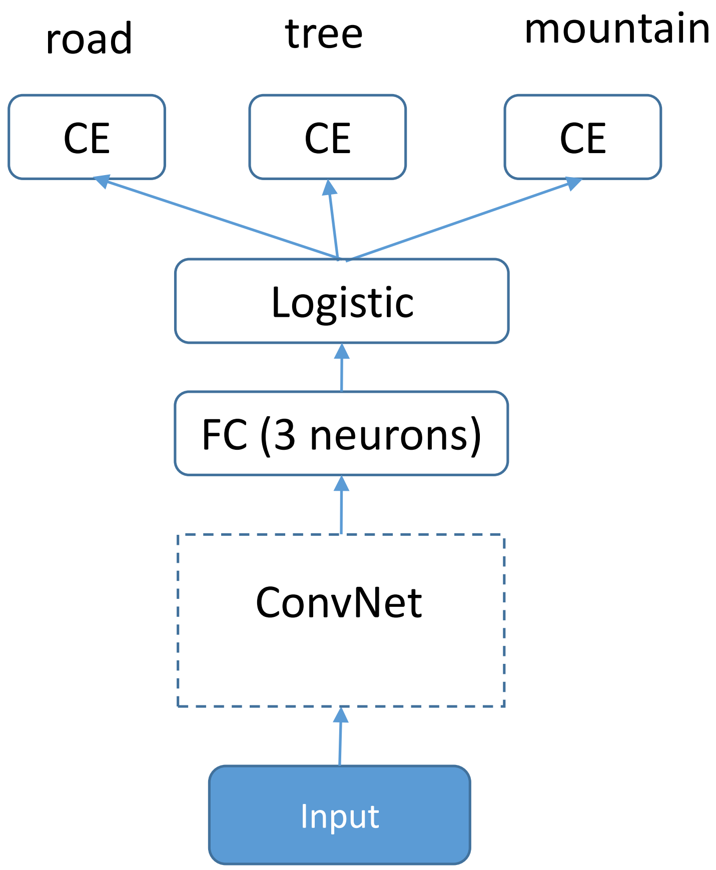
Image classification

- Approaches
  - AlexNet, VGG, InceptionNet, ResNet, DenseNet, etc
  - With a Softmax layer as the final output layer
  - With cross-entropy as the loss function
- Dataset
  - ImageNet
- Evaluation
  - Top-1: accuracy = #(top1 prediction is truth label) / # test samples
  - Top-5: accuracy = #(one of top5 prediction is truth label) / # test samples

Applications

- Logo classification
- Traffic sign classification
  - notebook
- Ecommerce product classification
- Medical image classification
- Food image classification
- ImageNet classification
- Dogs vs Cats
  - notebook

Image annotation



- Binary classification for each label
  - Training label: (0,1,1,0,1)
  - Prediction prob: (0.2, 0.6, 0.8, 0.1, 0.4)
  - Prediction label
    - One threshold per label, e.g. (0.5, 0.7, 0.6, 0.4, 0.6)
    - Label vector: (0, 0, 1, 0, 0)
- Approaches
  - Same architecture as image classification
  - Logistic function as the output layer
  - Cross-entropy (CE) loss for each label
  - Other output and loss layers [1]
- Evaluation [1]
  - Precision = average over all test samples {|Prediction ∩ Truth| / #Prediction}
  - Recall = average over all test samples {|Prediction ∩ Truth| / #Truth}

Image annotation

- Application
  - Example: [satellite image annotation]
  - [Notebook]

Source from: https://www.kaggle.com/c/planet-understanding-the-amazon-from-space

Object detection

- Detect the location of all object instances of all classes
- Training label
  - a list of <class, bounding box of each object instance>
- Prediction output
  - a list of <class, probability, bounding box of each object instance>

(x, y, w, h)

Training label:
bicycle (10, 100, 110, 110) (200, 200, 180, 80) …
People (200, 80, 71, 71) (300, 50, 20, 80) …

Prediction output:
bicycle 0.9 (9, 93, 100, 111), 0.8 (200, 200, 180, 80), …
people 0.8 (200, 80, 71, 71), …
….

Object detection

- Applications
  - Face detection
    - Point-and-shoot camera
  - Surveillance
    - Count cars, peoples, animals
  - Indexing
    - Get objects from images for search
- Evaluation
  - Matched prediction = detected bounding box has enough overlap with truth and its label is correct
  - Precision = #matched prediction / #total predictions
  - Recall = #matched prediction / #truth instance (bounding box)

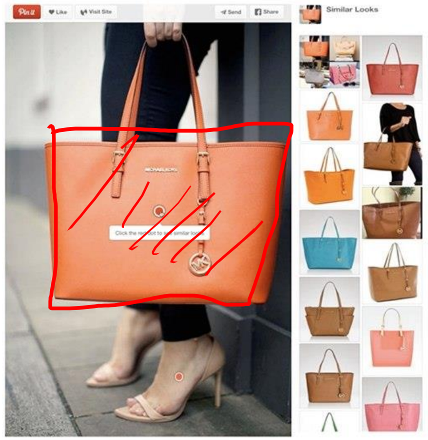
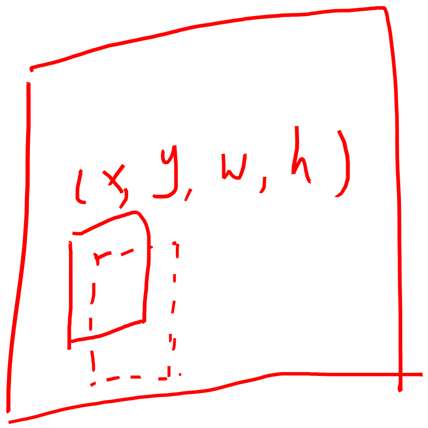


Figure 1: Similar Looks: We apply object detection to localize products such as bags and shoes. In this prototype, users click on automatically tagged objects to view similar-looking products.

Source from [10]

Object detection



- Solution
  - Find some candidate regions with objects
  - Extract CNN feature from this region
  - Refine the region boundary (bounding box) using a regressor
    - Generate 4 values (x, y, h, w)
  - Predict the class label using Softmax

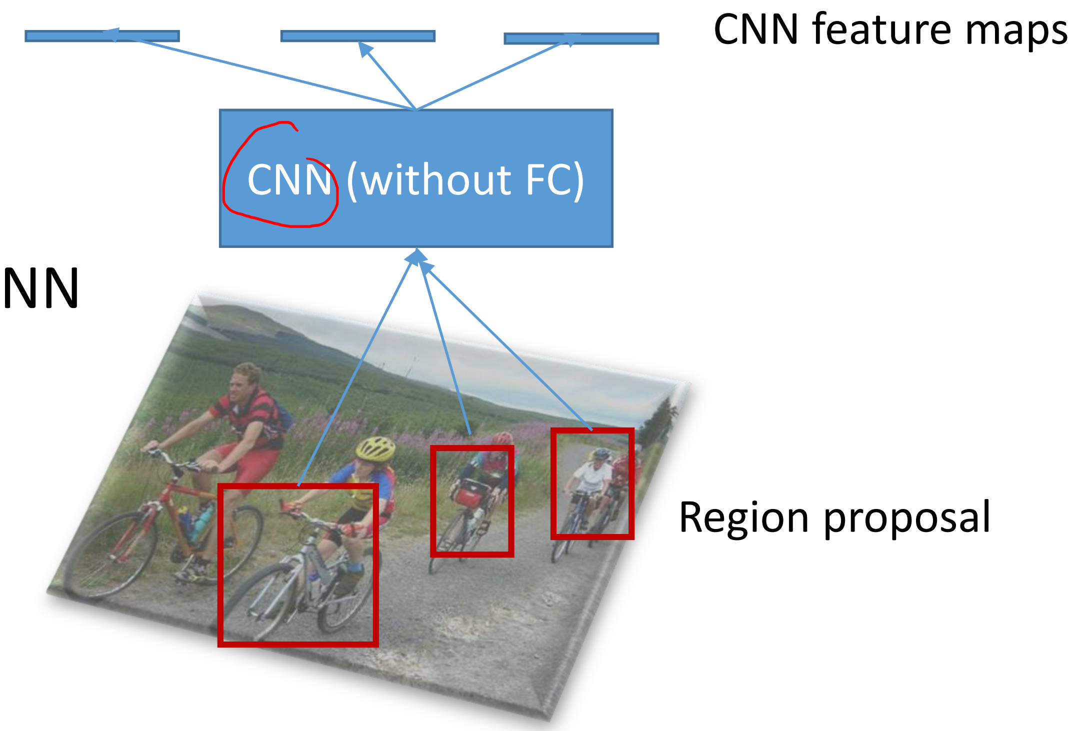
R-CNN [7]

- Generate region proposal
  - Candidate object regions selective search
  - Using existing methods

R-CNN

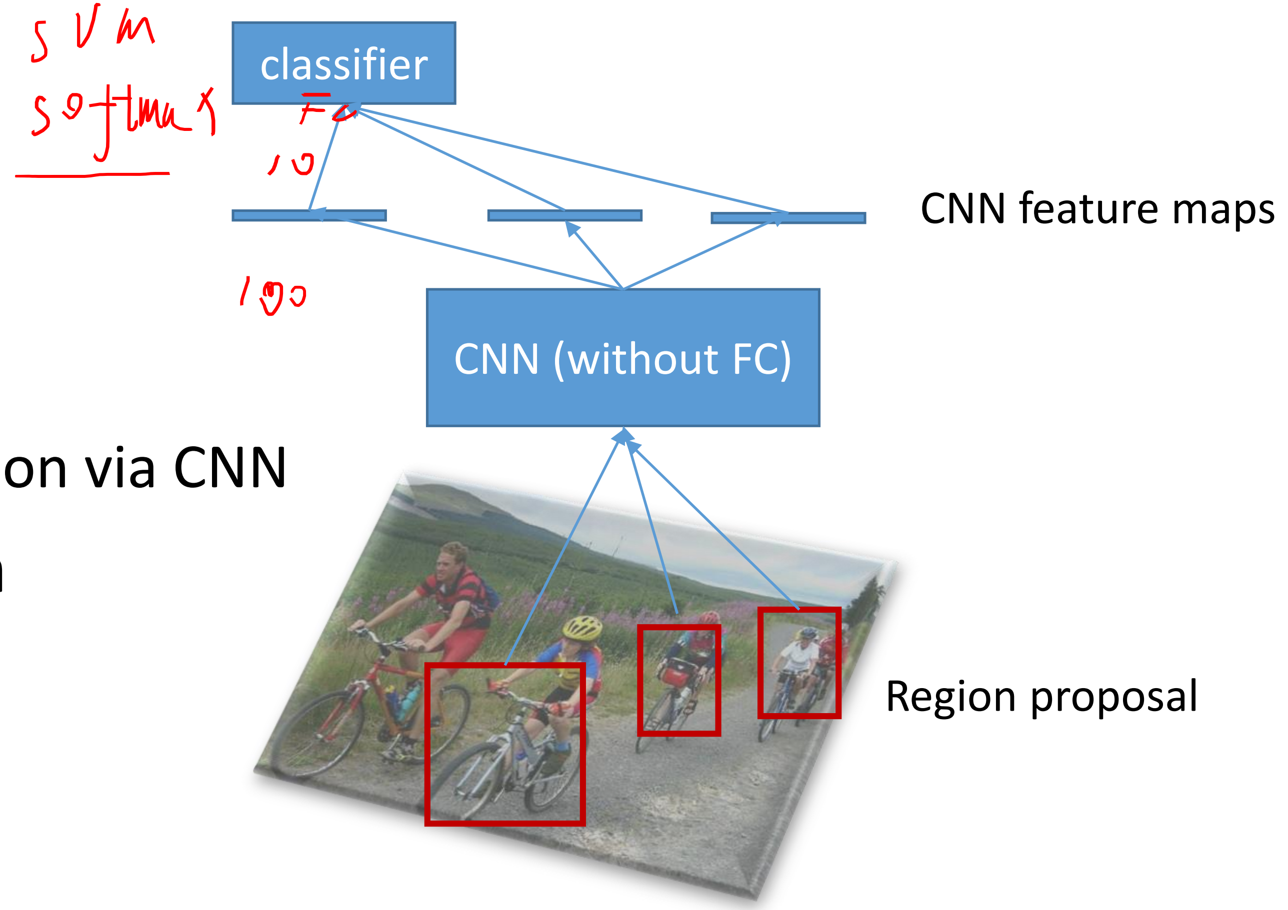
5 conv layer.
2 FC
softmax.

- Generate region proposal
  - Candidate object regions
  - Using existing methods
- Extract CNN feature
  - For each region
  - Forward-propagate each region via CNN
  - Using popular CNN architecture
    - VGG/ResNet/etc



R-CNN

- Generate region proposal
  - Candidate object regions
  - Using existing methods
- Extract CNN feature
  - For each region
  - Forward-propagate each region via CNN
- Predict label for each region
  - Like image classification
  - Linear layer + softmax



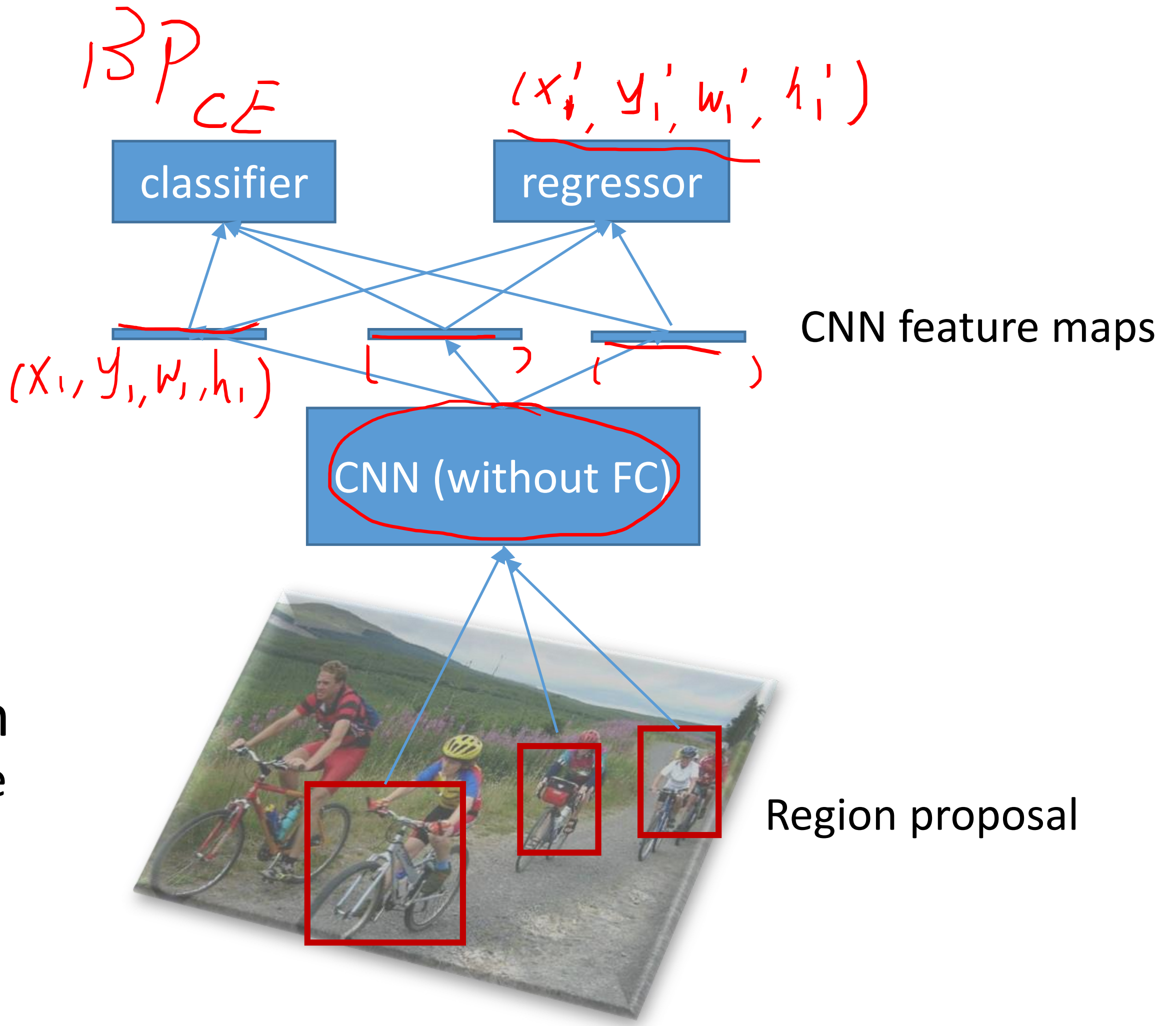
$$L_{bbox} = \|x_1' - x_1\|^2 + \|y_1' - y_1\|^2 + \cdots$$

R-CNN

- Generate region proposal
  - Candidate object regions
  - Using existing methods

2000

- Extract CNN feature
  - For each region
  - Forward-propagate each region via CNN
- Predict label for each region
  - Like image classification
  - Linear layer + softmax
- Regress bounding box for each region
  - Linear regression (L2 loss) for each value
  - There are 4 values
    - Coordinates for the left top corner: x, y
    - Height and width: h, w



R-CNN

- Slow
  - Too many region proposals~2000
  - Each has to go through the CNN
- Training is ad-hoc
  - Fine tune the CNN for the target dataset for image classification
  - Train label classifier for regions
  - Train SVM regressor for bounding box

Image segmentation

- Label each pixel with a class
- Training label
  - A class (index) per pixel
- Prediction output
  - For each pixel, a probability vector (one per class)

Training label:

(0, 0) bicycle …

(200, 80) people (200, 81) people

Prediction output:

(0, 0) background 0.9; mountain: 0.1 tree : 0

…

(200, 80) people 0.8; bicycle 0.1; tree: 0.1

Image segmentation

- Applications
  - Medical image analysis
  - Self-driving car
- Evaluation [1]
  - Matched pixels = the predicted class of a pixel is the truth class
  - Mean IoU = average over all classes{#matched pixels/(truth pixels U predicted pixels)}

Image segmentation

- Solution
  - Encoder to extract a semantic-rich representation
    - For label prediction
    - Subsampling by (pooling or convolution with stride > 1)
  - Decoder to incorporate location information
    - To generate a final feature map as the same size as the input
    - Upsampling
  - Loss
    - Softmax loss for each pixel

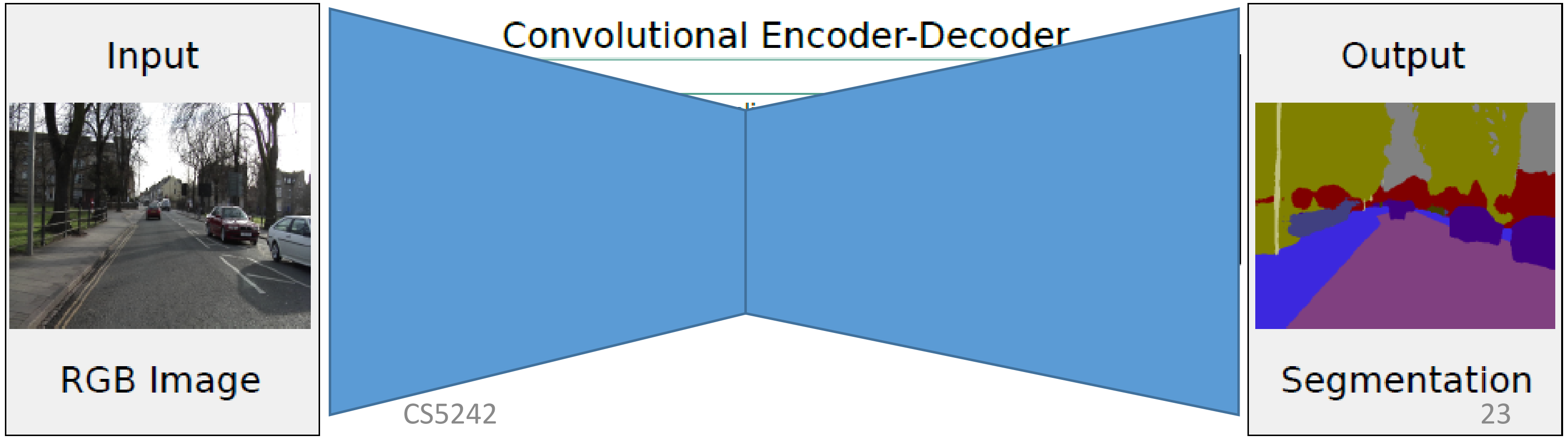


Image segmentation

- Upsampling
  - Nearest neighbour

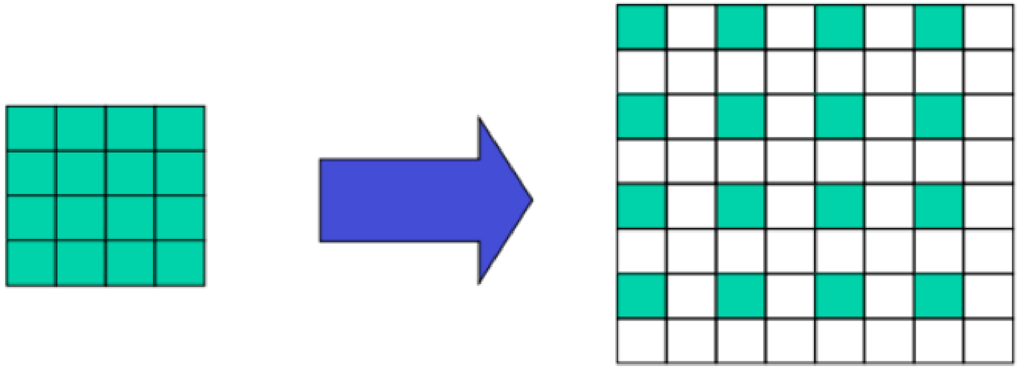
3	2
0	1

3	3	2	2
3	3	2	2
0	0	1	1
0	0	1	1

Image segmentation

1. Copy the values to the big matrix
2. Fill in the empty cells with 0
3. Do convolution with manually set kernel values (e.g. all 1)
4. Rescale the output to match the norm of the input (like Dropout)

- Bilinear upsampling



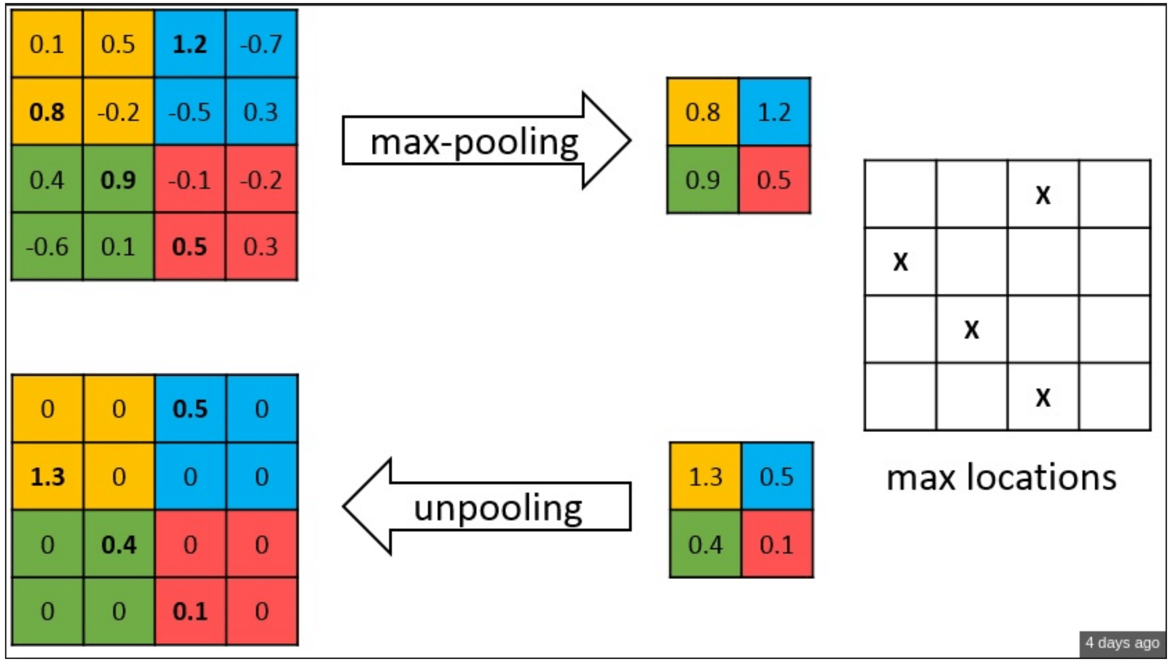
Original image:  x 10

Nearest-neighbor interpolation   Bilinear interpolation

- The empty pixels are initially set to 0
- Convolve with a (Gaussian, or another) filter
- If the filter sums to 1, multiply the result by 4
  - ¾ of the new image was initially 0

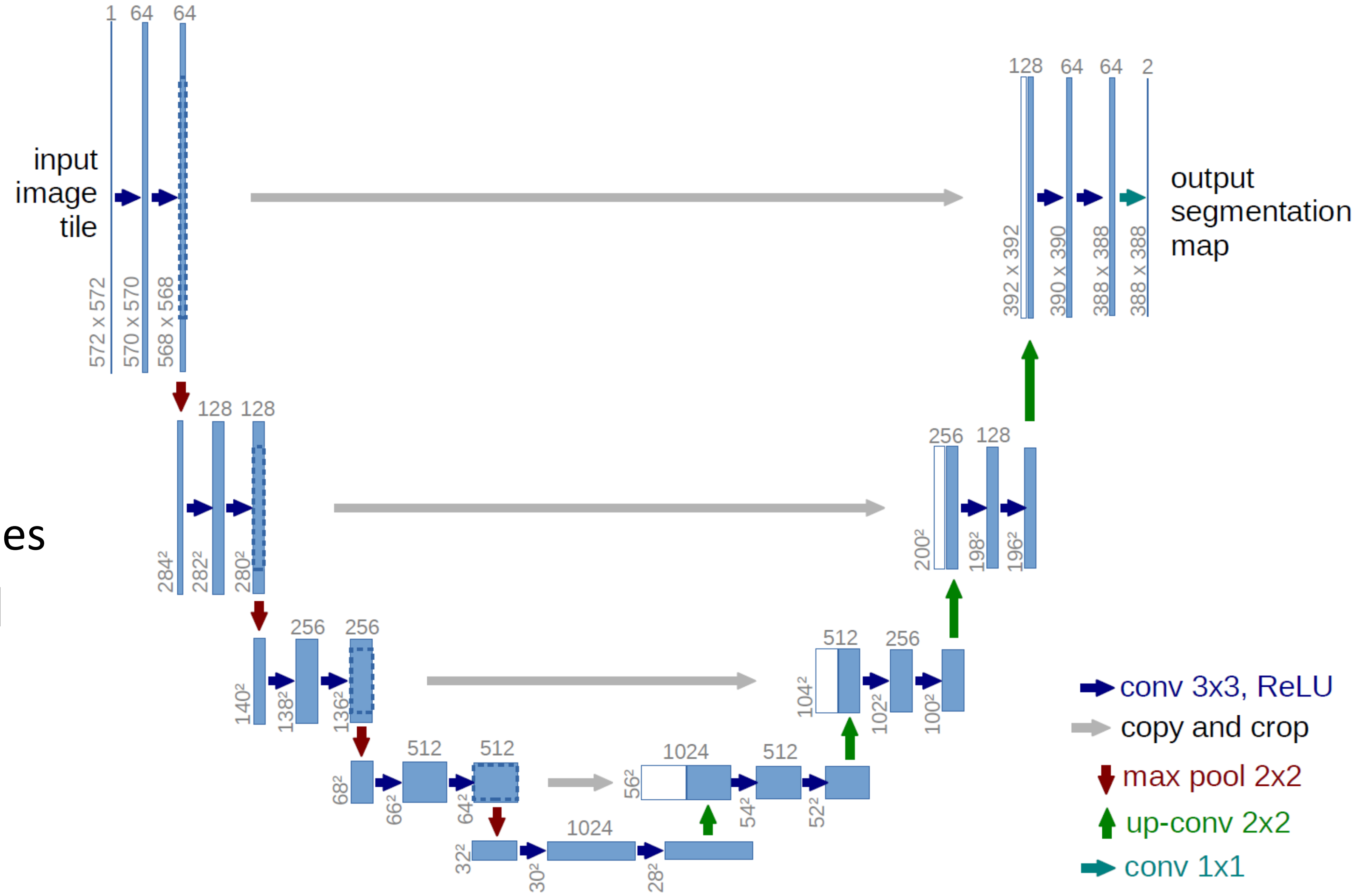
Image segmentation

- Max unpooling



U-Net[5]

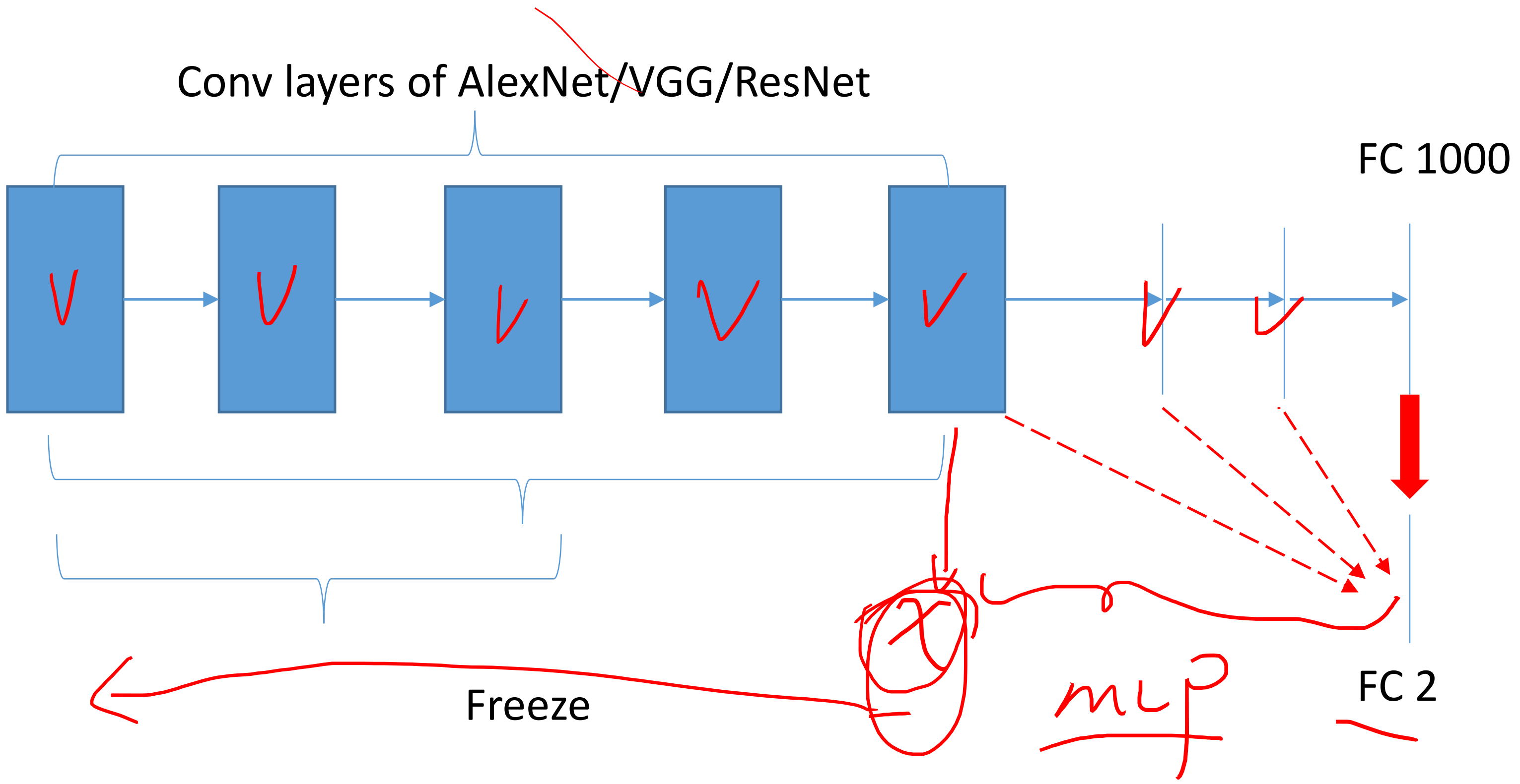
- Prototxt visualization
- Input and output
  - Different size
  - Due to valid padding
- 3x3 convolution
- Final convolution
  - # filters/channels = # classes
- Softmax over the channel dimension
- Examples
  - 1, 2



Hands-on Tutorial

Transfer ConvNets trained on ImageNet

Notebook is on IVLE



Tips for transfer learning

- Use ConvNets as feature extractor
  - Train another simple model (e.g. SVM) to do classification
- Fine-tune some layers of the ConvNets for your own tasks

	very similar dataset	**very different dataset**
very little data	Use Linear Classifier on top layer	You're in trouble… Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

From cs231n

Real-time object detection:YOLO [3]

- [Slides]
- [Notebook]

Reference

- [1]Yunchao Gong, Yangqing Jia, Thomas Leung, Alexander Toshev, Sergey Ioffe. Deep convolutional ranking for multilabel image annotation. https://arxiv.org/pdf/1312.4894.pdf
- [2]Ross Girshick. Fast R-CNN. https://arxiv.org/abs/1504.08083
- [3]Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. https://arxiv.org/abs/1506.02640
- [4]SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla. 2016
- [5]U-Net: Convolutional Networks for Biomedical Image Segmentation. Olaf Ronneberger, Philipp Fischer, Thomas Brox. 2015
- [7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014