Digitale Bildverarbeitung und Mustererkennung

Course overview

Deep Learning Foundations

Classification & Object Detection and Transfer Learning

Segmentation Networks

Deep Reinforcement Learning

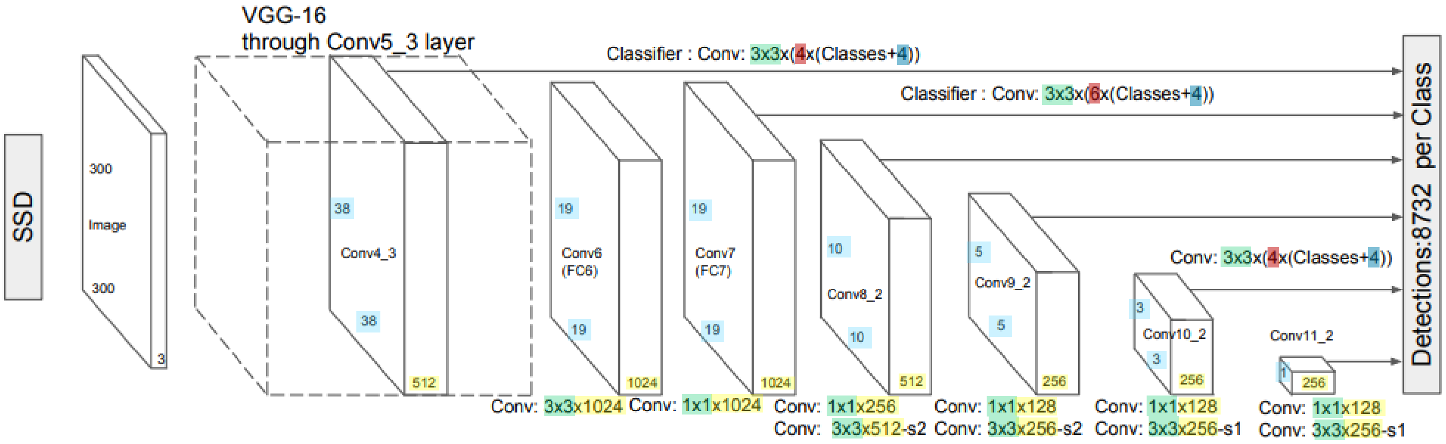
Generative Adversarial Networks

Recurrent Neural Networks

Neural Network Object Detection – Single Shot Detector with VGG-16 Backbone

Different Bounding Box Default Shapes, e.g. ▯ □ ▭

Offsets of coordinates for current bounding box



Kernel Size

Amount of Kernels in Layer = Depth of subsequent Feature Map

Width & Height of Feature Map

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). SSD: Single shot multibox detector. In *European conference on computer vision* (pp. 21-37). Springer, Cham. (*https://arxiv.org/pdf/1512.02325.pdf*)

Neural Network Object Detection - Basic structure

SSD: Single Shot MultiBox Detector

SSD: Single Shot MultiBox Detector

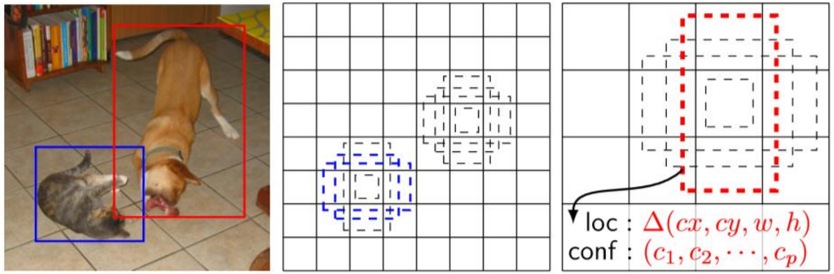
Wei Liu[1], Dragomir Anguelov[2], Dumitru Erhan[3], Christian Szegedy[3], Scott Reed[4], Cheng-Yang Fu[1], Alexander C. Berg[1]

[1]UNC Chapel Hill [2]Zoox Inc. [3]Google Inc. [4]University of Michigan, Ann-Arbor
wliu@cs.unc.edu, [2]drago@zoox.com, [3]{dumitru,szegedy}@google.com, [4]reedscot@umich.edu, [1]{cyfu,aberg}@cs.unc.edu

Discretization of the input image into a SxS grid of different sizes.

Fully Convolutional Network predicts class scores and box offsets for given default bounding boxes per size.

Final detections are chosen with non-maximum suppression.



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

https://arxiv.org/abs/1512.02325

Neural Network Object Detection - Basic structure

RFCN: Region-based Fully Convolutional Network

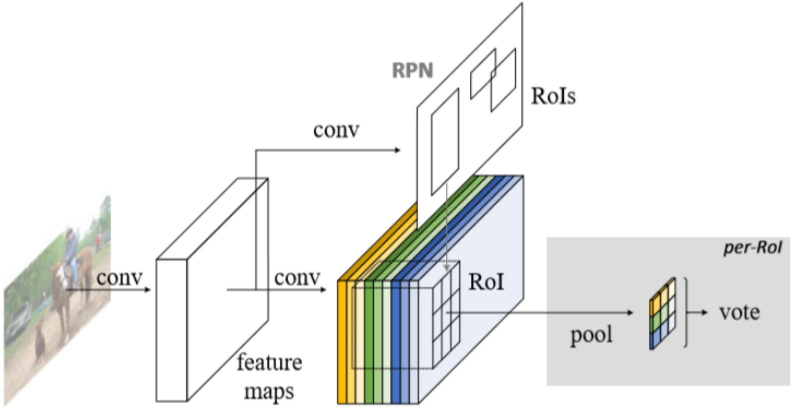
Two stage object detection.

First stage is Fully Convolutional Network, such as ResNet-101 for region proposals

Second stage does classification on the max-pooled proposed regions

R-FCN: Object Detection via Region-based Fully Convolutional Networks

Jifeng Dai Microsoft Research
Yi Li* Tsinghua University
Kaiming He Microsoft Research
Jian Sun Microsoft Research



https://arxiv.org/abs/1605.06409

TensorFlow Model Zoo

Collection of detection models pre-trained on different object detection datasets.

Model name	Speed (ms)	COCO mAP	Outputs
SSD MobileNet v2 320x320	19	20.2	Boxes
SSD MobileNet V1 FPN 640x640	48	29.1	Boxes
SSD MobileNet V2 FPNLite 320x320	22	22.2	Boxes
SSD MobileNet V2 FPNLite 640x640	39	28.2	Boxes
SSD ResNet50 V1 FPN 640x640 (RetinaNet50)	46	34.3	Boxes
SSD ResNet50 V1 FPN 1024x1024 (RetinaNet50)	87	38.3	Boxes
SSD ResNet101 V1 FPN 640x640 (RetinaNet101)	57	35.6	Boxes
SSD ResNet101 V1 FPN 1024x1024 (RetinaNet101)	104	39.5	Boxes
SSD ResNet152 V1 FPN 640x640 (RetinaNet152)	80	35.4	Boxes
SSD ResNet152 V1 FPN 1024x1024 (RetinaNet152)	111	39.6	Boxes
Faster R-CNN ResNet50 V1 640x640	53	29.3	Boxes
Faster R-CNN ResNet50 V1 1024x1024	65	31.0	Boxes
Faster R-CNN ResNet50 V1 800x1333	65	31.6	Boxes
Faster R-CNN ResNet101 V1 640x640	55	31.8	Boxes
Faster R-CNN ResNet101 V1 1024x1024	72	37.1	Boxes
Faster R-CNN ResNet101 V1 800x1333	77	36.6	Boxes
Faster R-CNN ResNet152 V1 640x640	64	32.4	Boxes
Faster R-CNN ResNet152 V1 1024x1024	85	37.6	Boxes
Faster R-CNN ResNet152 V1 800x1333	101	37.4	Boxes
Faster R-CNN Inception ResNet V2 640x640	206	37.7	Boxes
Faster R-CNN Inception ResNet V2 1024x1024	236	38.7	Boxes
Mask R-CNN Inception ResNet V2 1024x1024	301	39.0/34.6	Boxes/Masks

https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md

References

[2] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[3] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *14th european conference on computer vision*, pages 21–37, 2016.

[4] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: object detection via region-based fully convolutional networks. *CoRR*, abs/1605.06409, 2016.

[5] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[6] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, Jun 2010.

[7] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[8] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 233–240, New York, NY, USA, 2006. ACM.

[9] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, Jun 2010.

This lecture in one slide

Object Detection with neural networks

Transfer Learning with neural networks

Fine Tuning

Freezing Weights

Transfer Learning with Neural Networks

The intuition behind transfer learning is that if a model trained on a large and general enough dataset, this model will effectively serve as a **generic model of the visual world**.

Transfer Learning with Neural Networks

The intuition behind transfer learning is that if a model trained on a large and general enough dataset, this model will effectively serve as a generic model of the visual world.

You can then take advantage of these learned features without having to start from scratch training a large model on a large dataset.

Transfer Learning with Neural Networks - Fine Tuning

A **Pre-trained model** is a model that is trained on the source domain for a source task.

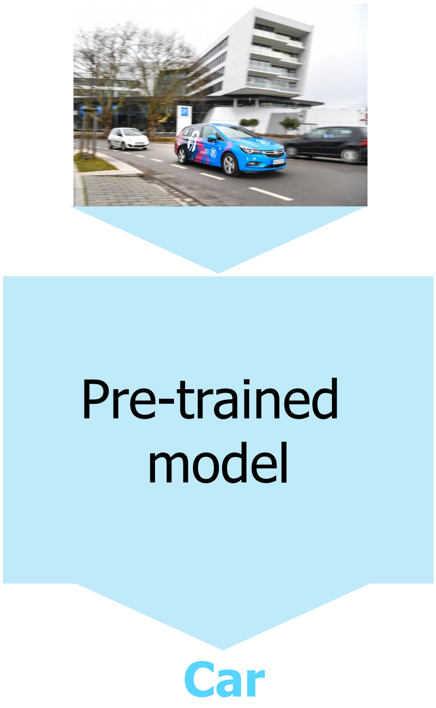
Pre-trained model

Car

Transfer Learning with Neural Networks - Fine Tuning

A **Pre-trained model** is a model that is trained on source domain for a source task.

Think **Object Detection Model Zoo**

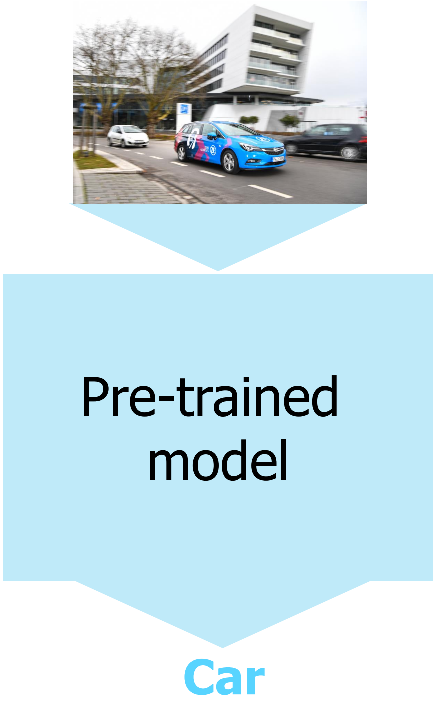


https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md

Transfer Learning with Neural Networks - Fine Tuning

A **Pre-trained model** is a model that is trained on the source domain for a source task.

The source domain is usually a **large dataset**.

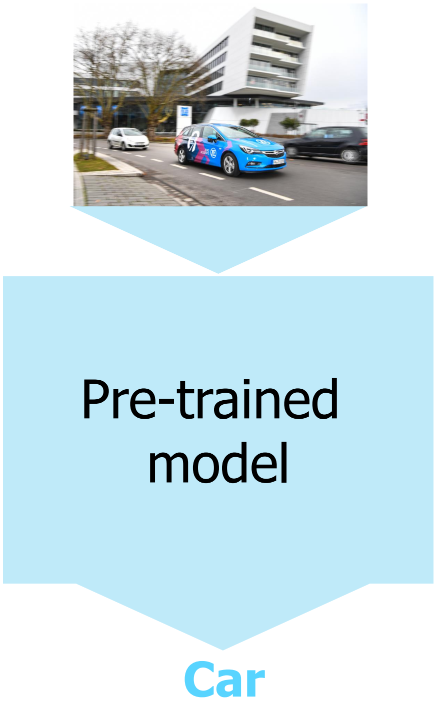


Transfer Learning with Neural Networks - Fine Tuning

A **Pre-trained model** is a model that is trained on the source domain for a source task.

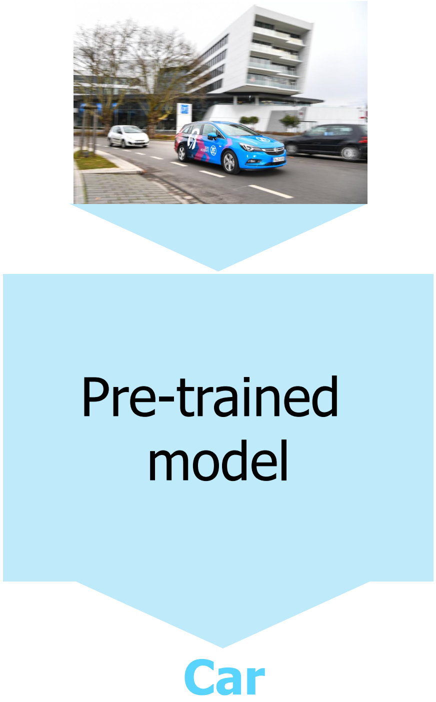
The source domain is usually a **large dataset**.

Think of **COCO**, **ImageNet**, **PASCAL VOC**



Transfer Learning with Neural Networks - Fine Tuning

You either use a **pre-trained model as it is**.



Transfer Learning with Neural Networks - Fine Tuning

You either use a pre-trained model as it is.

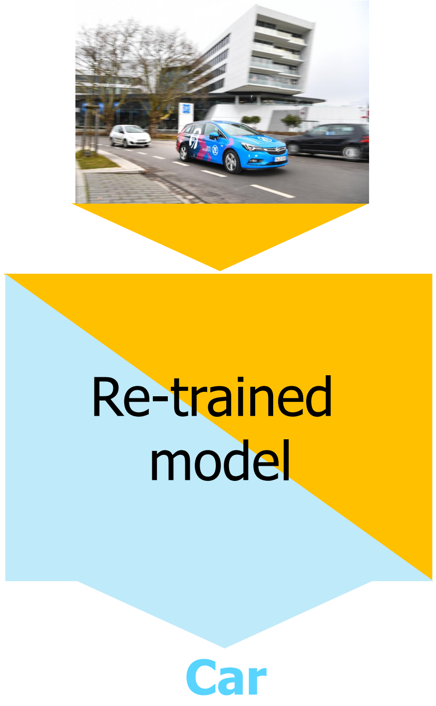
- No additional training
- Usually **performance average or low**

Pre-trained model

Car

Transfer Learning with Neural Networks - Fine Tuning

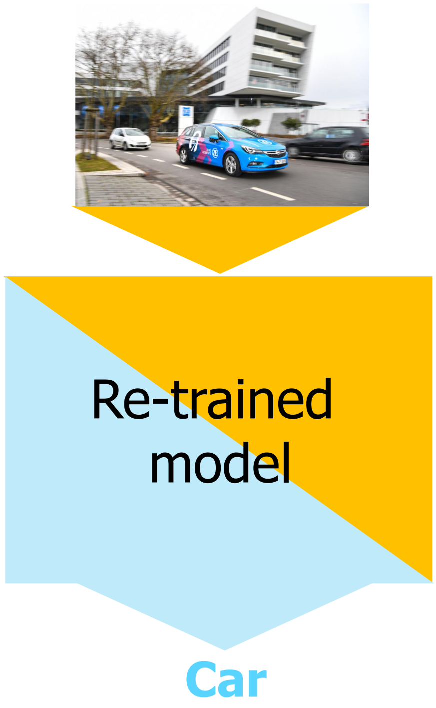
You can **fine-tune** a **pre-trained model** by retraining on additional data.



Transfer Learning with Neural Networks - Fine Tuning

You can fine-tune a pre-trained model by retraining on additional data.

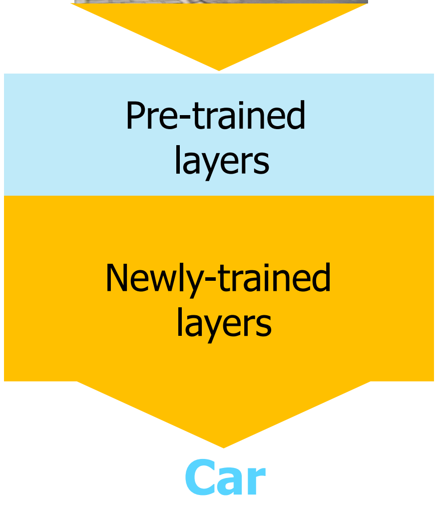
- Additional work to set up retraining pipeline
- Usually performance drops in the source domain
- **Performance improvement** in the target domain



Transfer Learning with Neural Networks - Feature Extraction

You can **repurpose** the **pre-trained layers** as feature extraction layers for low level features.

And adding layers that learn the required high level features on the new data.

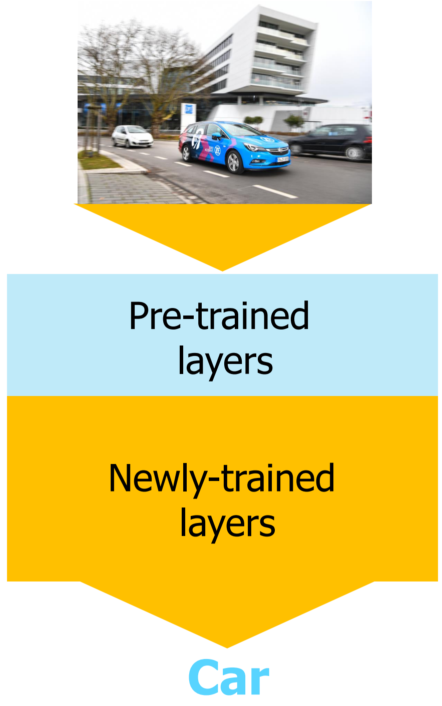


Transfer Learning with Neural Networks - Feature Extraction

You can repurpose the pre-trained layers as feature extraction layers for low level features.

And adding layers that learn the required high level features on the new data.

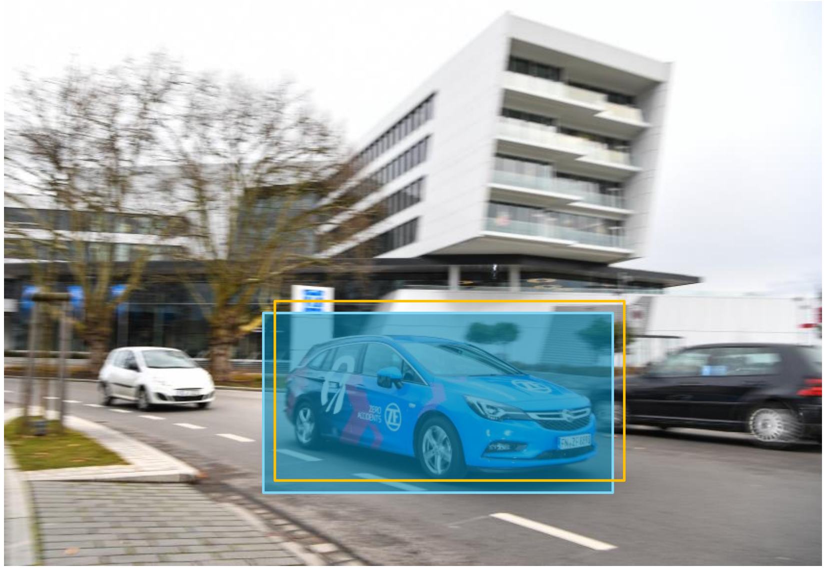
- Additional work to assemble layer structure
- **Regularization** effects
- **Performance improvement** in the target domain
- **Output** layer becomes **adjustable** (such as for adding a class)



Neural Network Object Detection - mAP

So how do we evaluate correctness of a detection?

Define when a detection is correct and when it is not.
A prediction will usually not overlap perfectly with the ground truth.



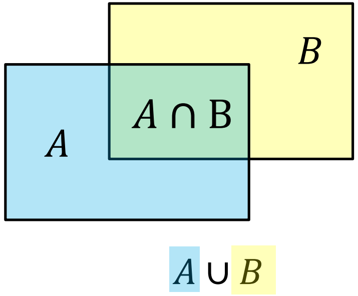
Neural Network Object Detection - mAP

So how do we evaluate correctness of a detection?

Define when a detection is correct, a true positive and when it is not.
A prediction will usually not overlap perfectly with the ground truth.

A correct detection is based on
Intersection over Union (IoU) with the
ground truth

$$IoU\ (A, B) = \frac{A \cap \mathrm{B}}{A \cup B}$$



$$IoU \geq \mathrm{p}, \qquad \mathrm{p} \in (0,1)$$

Neural Network Object Detection - mAP

Precision

$$P = \frac{TP}{TP + FP}$$

$TP \ = True\ Positives$

$FP = False\ Positives$

$P = Precision$

Correct detections over the number of predicted detections.

Recall

$$R = \frac{TP}{TP + FN}$$

$FN = False\ Negatives$

$R = Recall$

Correct detections over the number of groundtruth detection.

F1 Score

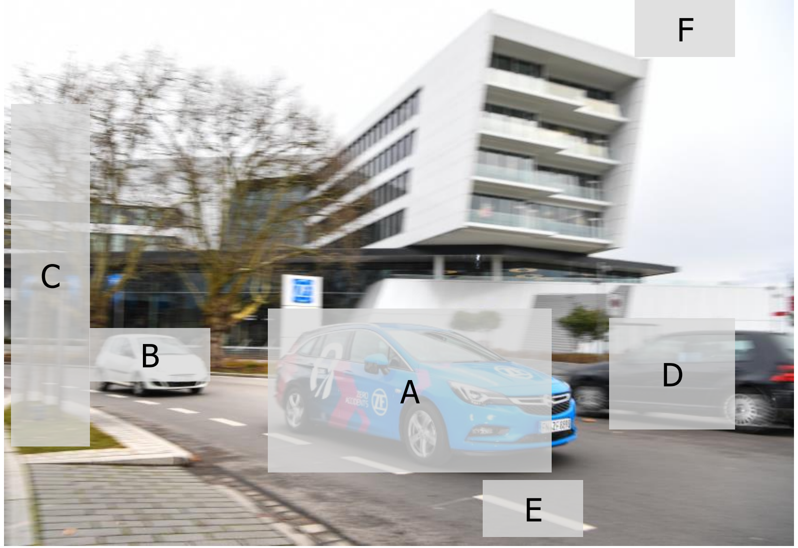
$$F1 \ = 2 \cdot \frac{P \cdot R}{P + R}$$

Harmonic mean of precision and recall.

Neural Network Object Detection - mAP

Precision Recall Curve
By moving the probability threshold of the outputs of the model we can fill in a precision recall curve.
When deploying a model you would need to set exactly <u>one</u> probability threshold.

Box	Prob.	Correct	TP	FP	FN	Prec.	Rec.
A	0.95	True	1	0	2	1	0.33
B	0.85	True	2	0	1	1	0.66
C	0.7	False	2	1	1	0.66	0.66
D	0.5	True	3	1	0	0.75	1
E	0.3	False	3	2	0	0.6	1
F	0.2	False	3	3	0	0.5	1



Neural Network Object Detection - mAP

Precision Recall Curve
By moving the probability threshold of the outputs of the model we can fill in a precision recall curve.
When deploying a model you would need to set exactly <u>one</u> probability threshold.

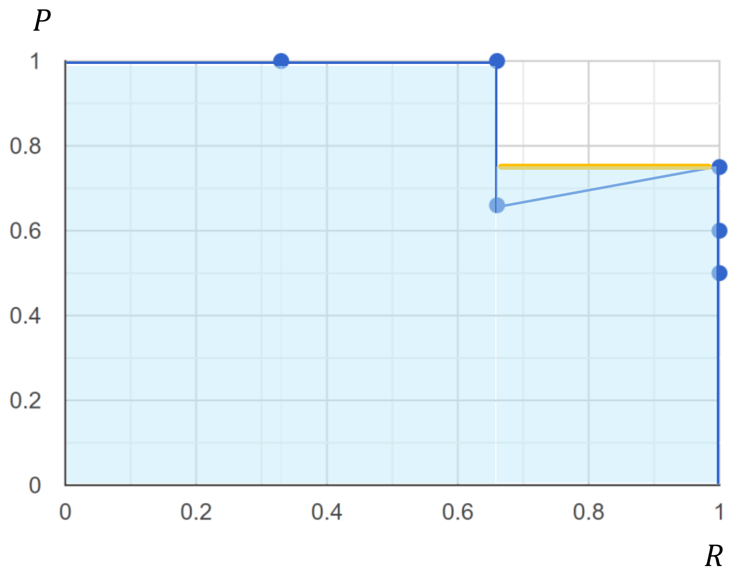
Average Precision (AP)
AP is the area under the curve
of the Precision Recall Curve.

$$AP = \sum_{n}^{N} (R_n - R_{n-1}) P_n$$

$R_n = Recall\ at\ threshold\ step\ n$

$P_n = Precision\ at\ threshold\ step\ n$

Prec.	Rec.
1	0.33
1	0.66
0.66	0.66
0.75	1
0.6	1
0.5	1



$$AP = 0.33 + 0.33 + 0 + 0.26 + 0 + 0 = 0.92$$

Neural Network Object Detection - mAP

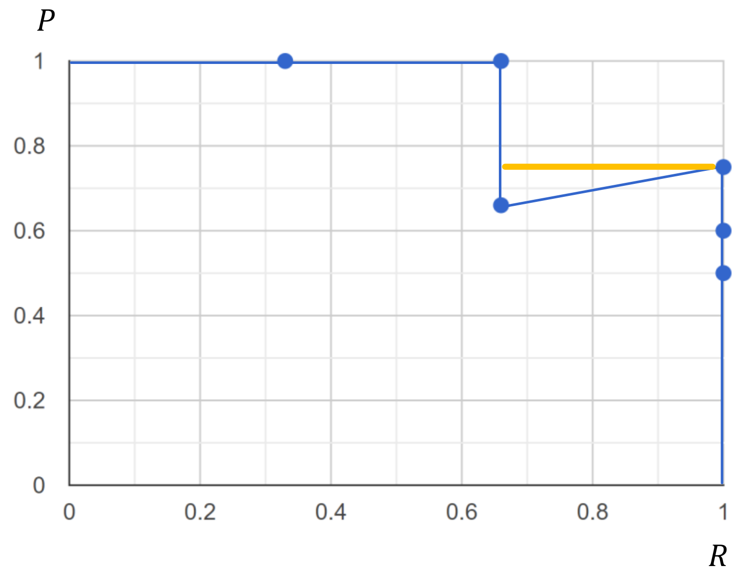
Precision Recall Curve
By moving the probability threshold of the outputs of the model we can fill in precision recall curve.
When deploying a model you would need to set exactly one threshold.

Average Precision (AP)
AP is the area under the curve of the Precision Recall Curve.

$$AP = \sum_{n}^{N} (R_n - R_{n-1}) P_n$$

$R_n = Recall\ at\ threshold\ step\ n$

$P_n = Precision\ at\ threshold\ step\ n$



mean Average Precision (mAP)
Average Precision over different classes.

$$mAP = \frac{1}{N} \sum_{c}^{N} AP_c$$

$c = current\ class$

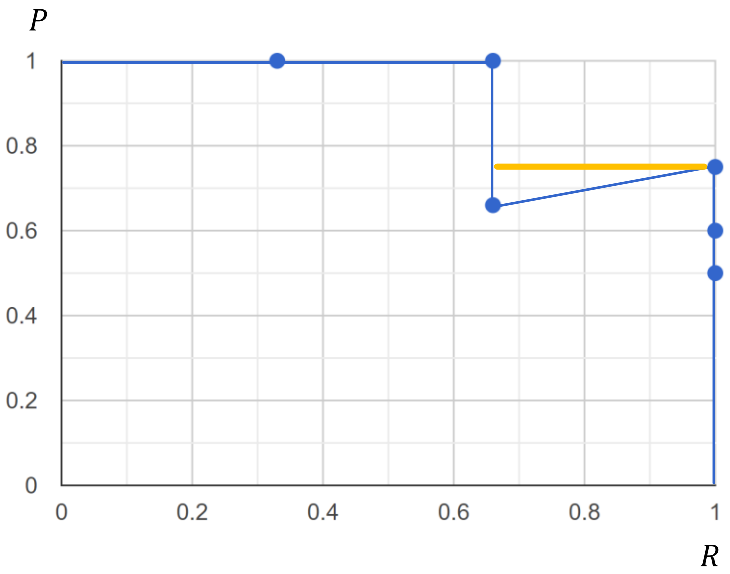
$N = amount\ of\ classes$

Neural Network Object Detection – COCO mAP

mAP@[0.5:0.05:0.95]

COCO mAP.

Average of Mean Average Precision over a set of IoU levels (0.5, 0.55, 0.6,…, 0.95).



Course overview

Deep Learning Foundations

Classification & Object Detection and Transfer Learning

Segmentation Networks

Deep Reinforcement Learning

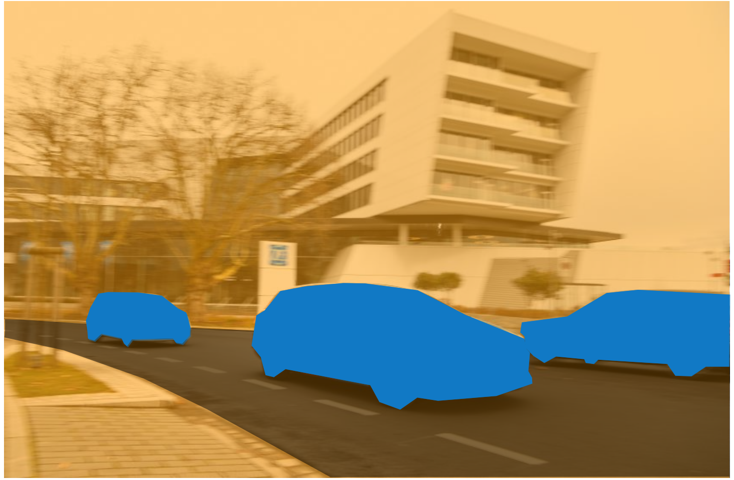
Generative Adversarial Networks

Recurrent Neural Networks

Introduction – Segmentation a problem statement

Segmentation is classification on pixel-level, which results in super-pixels or segments or groups of pixels based on some criteria.

Semantic Segmentation



Instance Segmentation

Autonomous Driving

- Scene understanding
- Understanding of shapes
- Free space detection

Geo Analytics

Medical Imaging

https://www.cityscapes-dataset.com/

Autonomous Driving

Geo Analytics

- Building structures
- Road network analysis
- Wildfire detection
- Water supply tracking
- Real time crisis management
- Weather prediction

https://github.com/mapbox/robosat

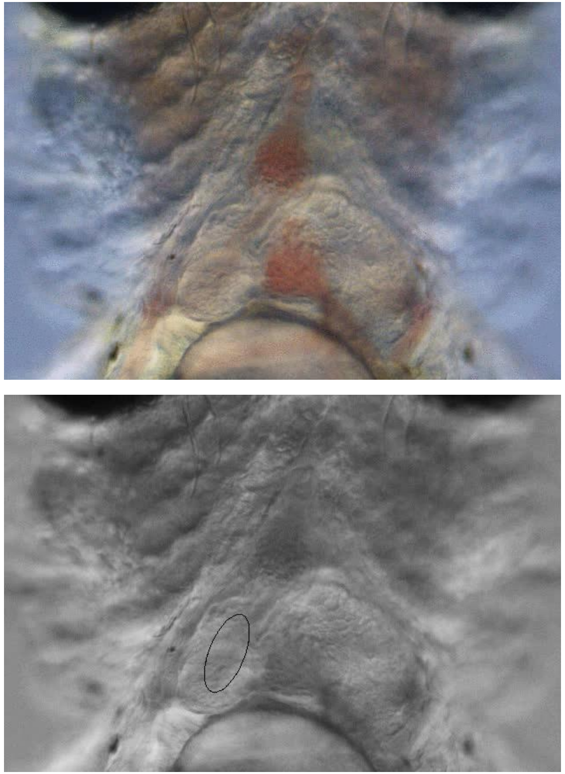
Medical Imaging

Introduction – Applications for segmentation

Autonomous Driving
Geo Analytics

Medical Imaging

- Tissue localization and analysis
- Volume approximations
- Surgery planning
- Temporal tumor or tissue development
- Tooling for drug testing



https://osf.io/snb6p/

Introduction – Conventional segmentation approaches

In order to design neural networks it is a good thing to really understand the task at hand.

Thresholding
Edge detection
Clustering

Introduction – Conventional segmentation approaches

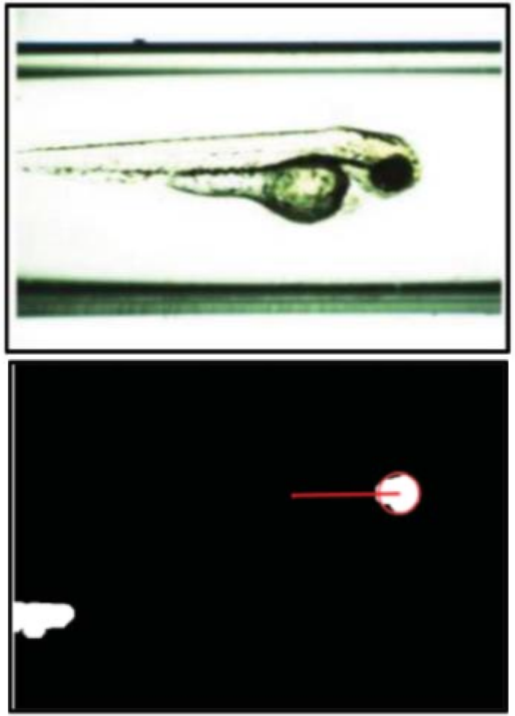
Thresholding

The simplest method of image segmentation is thresholding.

This method is based on a threshold value to turn a gray-scale image into a binary image (mask).

Usually this is just one step of many.

Edge detection
Clustering
Region growing



Thresholding on Zebrafish for eye segmentation

Lena Test image

Lena, the ‘hello world!’ of image processing. 330x330

Cover photo of 1972 Playboy magazine of the Swedish model Lena Söderberg.

Since then Lena was a guest at several IEEE conferences. The image also sparked discussions on gender-equality in the male-dominated field of engineering.

It is a good test image because of its detail, flat regions, shading, and texture.

https://en.wikipedia.org/wiki/Lenna

Introduction – Conventional segmentation approaches

Thresholding

Edge detection

Segment boundaries and edges are closely related.

Since there is often a large gradient at the segment boundaries.

Canny Edge Detection (mutli-step approach)

$$L_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} L \quad \text{and} \quad L_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} L.$$

Sobel Operators for Edge Detection

Clustering

Introduction – Conventional segmentation approaches

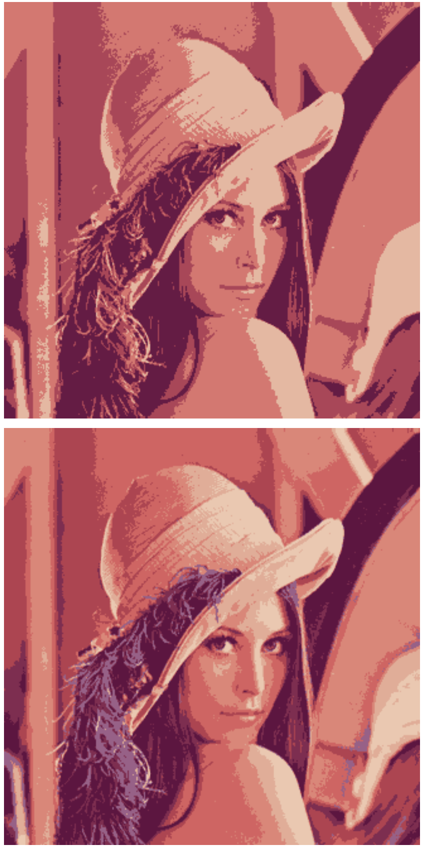
Thresholding
Edge detection

Clustering (Color quantization)

K-means with 3 features (R,G,B) and K centroids.

The centroids are iteratively adjusted until convergence.

After the clustering, the centroid values are applied to the pixels in their cluster.



Clustering for K=4 (top) and K=8 (bottom)

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[2] Andrej Karpathy. Cs231n: Convolutional neural networks for visual recognition. http://cs231n.github.io/neural-networks-3/, 2018. Zugriff: 20.01.2018.

[3] Schutera, Mark, Steffen Just, Jakob Gierten, Ralf Mikut, Markus Reischl, and Christian Pylatiuk. 2019. "Machine Learning Methods for Automated Quantification of Ventricular Dimensions." OSF. March 28. osf.io/snb6p.

[4] Mark Schutera, Thomas Dickmeis, Marina Mione, Ravindra Peravali, Daniel Marcato, Markus Reischl, Ralf Mikut, and Christian Pylatiuk. Automated phenotype pattern recognition of zebrafish for high-throughput screening. *Bioengineered*, 7(4):261–265, 2016.

[5] Canny, J., *A Computational Approach To Edge Detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986.

This lecture in one slide

Introduction to segmentation

Segmentation with neural networks

Basic structure

Overview state-of-the-art

Datasets and benchmarking

Deep dive U-Net

Neural Network Segmentation - Basic structure

Why classical segmentation approaches

- Interpretability
- Only a few samples needed
- No labeling needed
- No training needed
- Usually better runtime during inference

Why not?

Neural Network Segmentation - Basic structure

Why classical segmentation approaches

Why segmentation by neural networks?

- Do generalize better
- Feature engineering has a limited capacity to capture semantics
- Feature engineering is expensive and time consuming

Neural Network Segmentation - Basic structure

Feature Representation by Convolution

Idea is to classify each pixel of an input image by representation learning

Downsampling
Upsampling
Parameter sharing

Feature Representation by Convolution

0	0	0	0	0	0	0
0	1	2	0	0	0	0
0	2	2	0	0	0	0
0	2	2	2	1	0	0
0	1	0	0	2	0	0
0	1	1	0	1	0	0
0	0	0	0	0	0	0

Zero-Padding is adding zero-valued pixel to the image border (gray area).

Downsampling
Upsampling
Parameter sharing

Feature Representation by Convolution

0	0	0	0	0	0	0
0	1	2	0	0	0	0
0	2	2	0	0	0	0
0	2	2	2	1	0	0
0	1	0	0	2	0	0
0	1	1	0	1	0	0
0	0	0	0	0	0	0

Zero-padded image

0	0	-1
-1	0	0
-1	-1	-1

Filter

0

Bias

-4	-4	0
-3	-4	-3
0	-3	-1

Output

Downsampling

Upsampling

Parameter sharing

Feature Representation by Convolution

0	0	0	0	0	0	0
0	1	2	0	0	0	0
0	2	2	0	0	0	0
0	2	2	2	1	0	0
0	1	0	0	2	0	0
0	1	1	0	1	0	0
0	0	0	0	0	0	0

Filter

0	0	-1
-1	0	0
-1	-1	-1

Bias

0

Output

-4	-4	0
-3	-4	-3
0	-3	-1

Amount of filters or convolution depth: 1
Filter step size or Stride: 2
Zero-padded image

Downsampling
Upsampling
Parameter sharing

Feature Representation by Convolution

0	0	0	0	0	0	0
0	1	2	0	0	0	0
0	2	2	0	0	0	0
0	2	2	2	1	0	0
0	1	0	0	2	0	0
0	1	1	0	1	0	0
0	0	0	0	0	0	0

0	0	-1
-1	0	0
-1	-1	-1

Filter

0

Bias

-4	-4	0
-3	-4	-3
0	-3	-1

Output

Downsampling
Upsampling
Parameter sharing

Review edge detector:
Similar idea, now the parameters of the filters are learned. We want a lot of filters!

$$L_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} L \quad \text{and} \quad L_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} L.$$

Feature Representation by Convolution

0	0	0	0	0	0	0
0	1	2	0	0	0	0
0	2	2	0	0	0	0
0	2	2	2	1	0	0
0	1	0	0	2	0	0
0	1	1	0	1	0	0
0	0	0	0	0	0	0

Filter

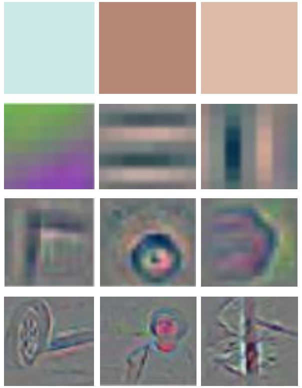
0	0	-1
-1	0	0
-1	-1	-1

Bias

0

Output

-4	-4	0
-3	-4	-3
0	-3	-1



Review edge detector:
Similar idea, now the parameters of the filters are learned. And we want to go deep!

Downsampling
Upsampling
Parameter sharing

Convolutions

Downsampling

Convolutions at original image resolution are computational expensive:
Filter dimensions x *image dimensions* x *number of filters* x *number of input channels.*

Motivating a convolutional
encoder-decoder
structure and Downsampling.

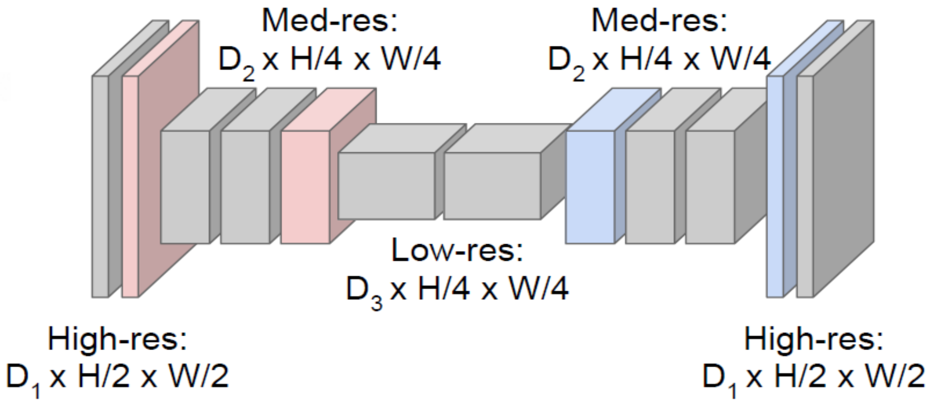
Upsampling
Parameter sharing

Convolutions

Downsampling

Convolutions at original image resolution are computational expensive:
Filter dimensions x *image dimensions* x *number of filters* x *number of input channels.*

Motivating a convolutional encoder-decoder structure and Downsampling.



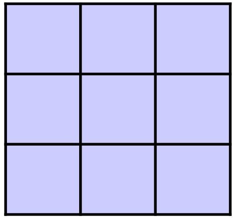
http://cs231n.github.io/

Upsampling
Parameter sharing

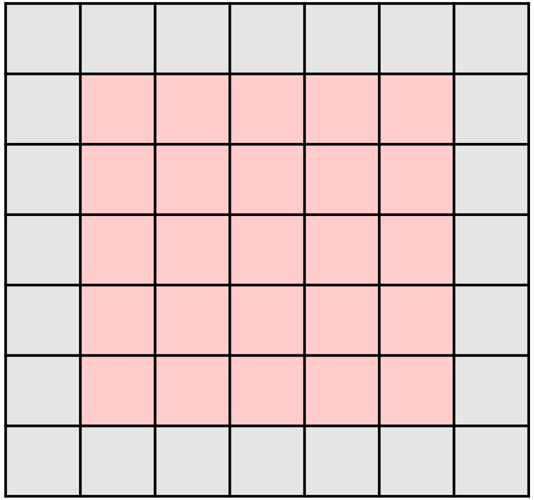
Convolutions

Downsampling

Strided convolutions



Filter 3x3x1



Zero-padded image

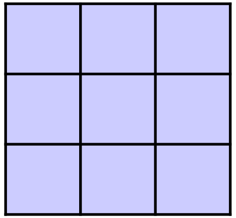
Upsampling

Parameter sharing

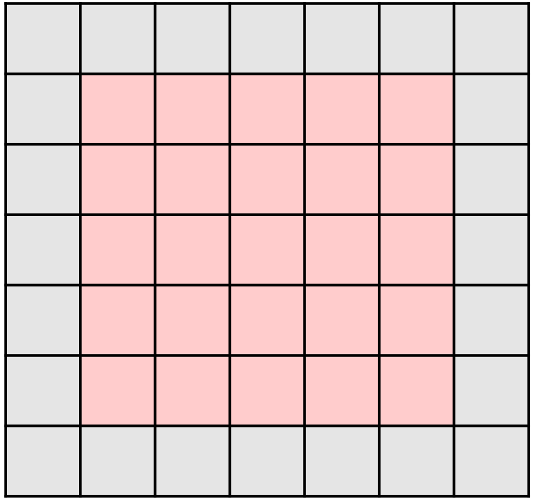
Convolutions

Downsampling

Strided convolutions

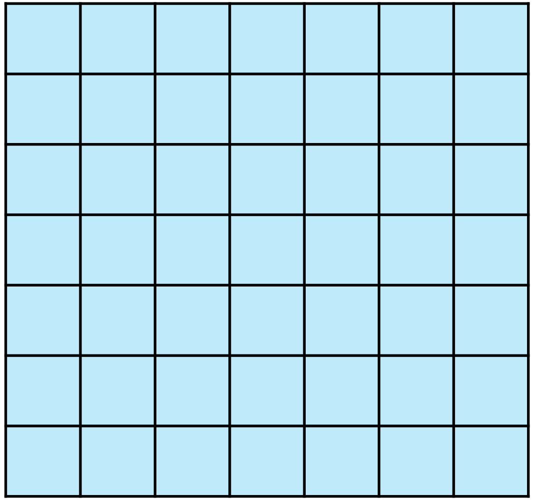


Filter 3x3x1



Zero-padded image

Stride 1



Upsampling

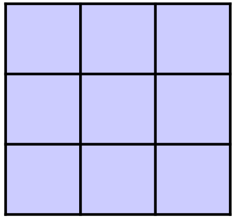
Parameter sharing

Neural Network Segmentation - Basic structure

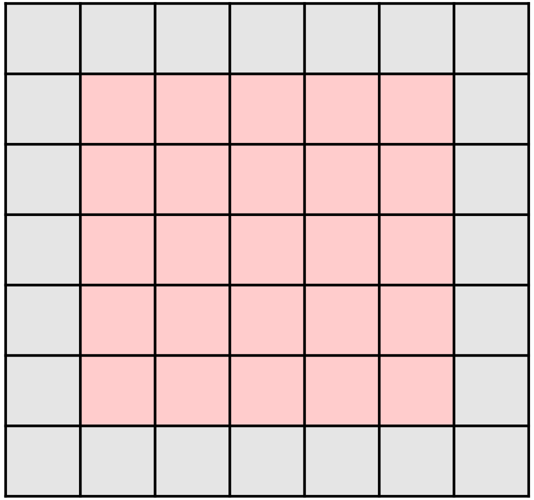
Convolutions

Downsampling

Strided convolutions

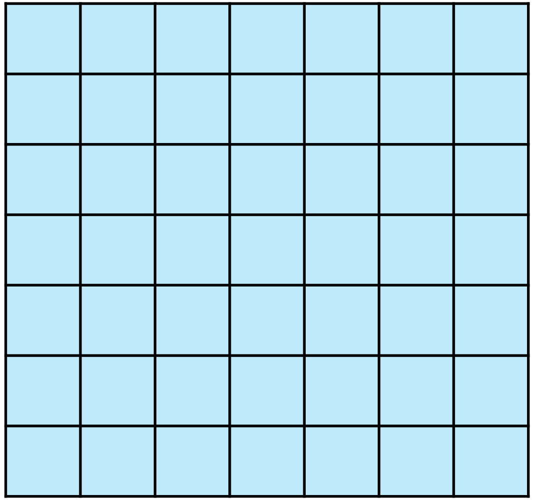


Filter 3x3x1

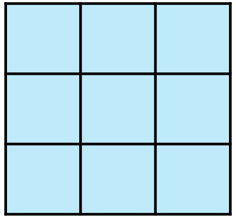


Zero-padded image

Stride 1



Stride 2



Upsampling

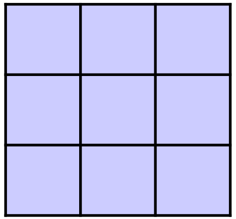
Parameter sharing

Neural Network Segmentation - Basic structure

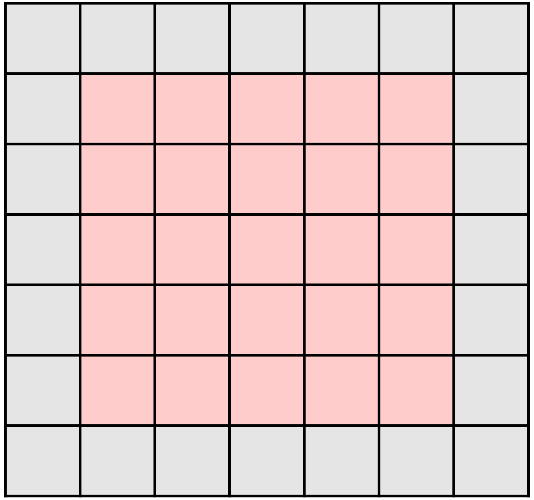
Convolutions

Downsampling

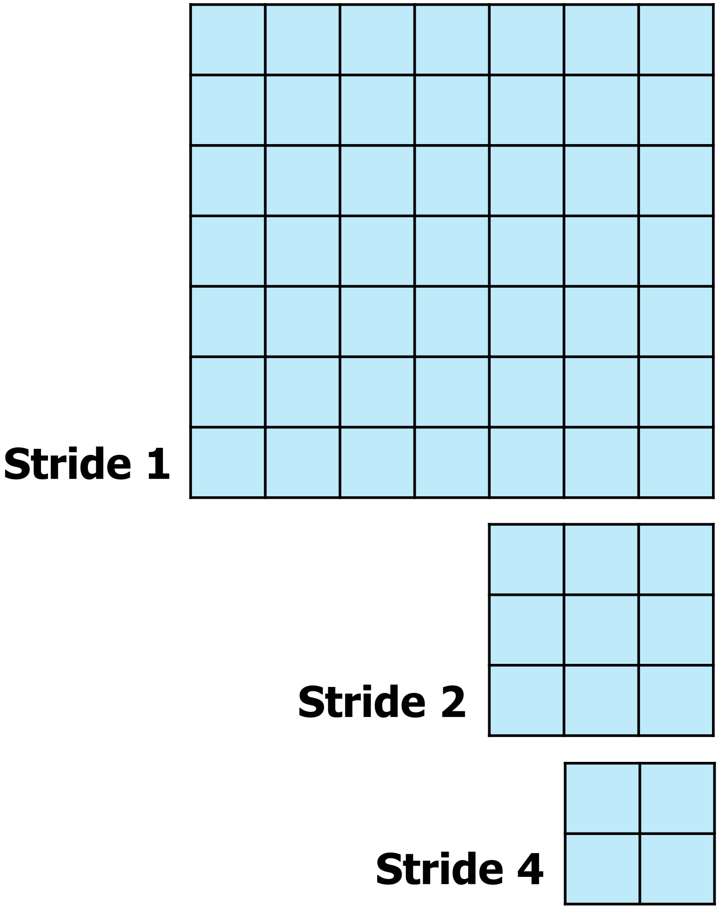
Strided convolutions



Filter 3x3x1



Zero-padded image



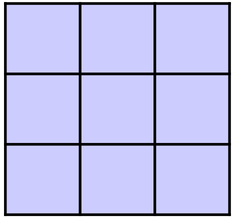
Upsampling

Parameter sharing

Convolutions

Downsampling

Max Pooling



Max Pooling
3x3 Stride 3

5	4	6	3	1	7
3	4	1	6	4	4
6	2	5	6	6	4
2	6	9	8	6	3
4	8	5	6	8	2
3	1	7	8	6	3

6	7
9	8

Upsampling
Parameter sharing

Neural Network Segmentation - Basic structure

Convolutions

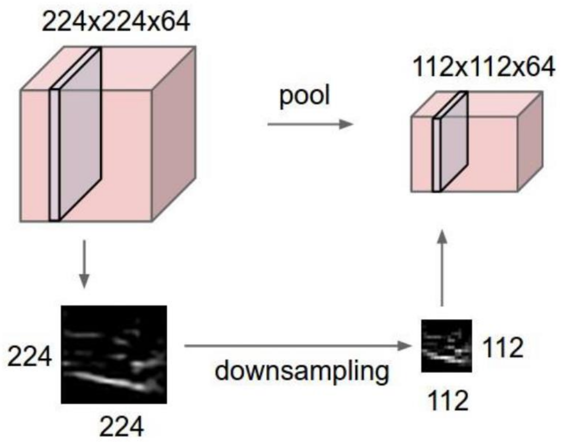
Downsampling

Max Pooling

Intuition is to decrease the resolution while keeping the strongest features of each channel.

Upsampling

Parameter sharing



https://selfdrivingcars.mit.edu/

Neural Network Segmentation - Basic structure

Convolutions

Downsampling

Upsampling

Classification needs to happen in original image resolution

Motivating Upsampling inside the network structure.

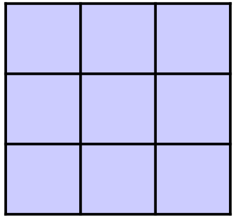
Parameter sharing

Convolutions

Downsampling

Upsampling

Nearest neighbor



3x3 Stride 3

6	7
9	8

→

6	6	6	7	7	7
6	6	6	7	7	7
6	6	6	7	7	7
9	9	9	8	8	8
9	9	9	8	8	8
9	9	9	8	8	8

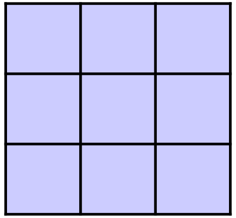
Parameter sharing

Convolutions

Downsampling

Upsampling

Bed of Nails



3x3 Stride 3

$$\begin{bmatrix} 6 & 7 \\ 9 & 8 \end{bmatrix} \rightarrow \begin{bmatrix} 6 & 0 & 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 9 & 0 & 0 & 8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Parameter sharing

Convolutions

Downsampling

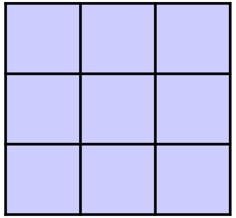
Upsampling

Transpose convolution

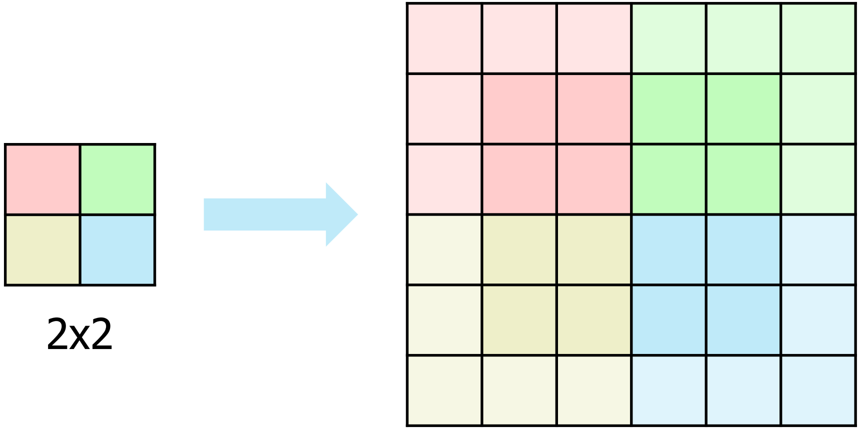
Learnable Upsampling, also known as: Upconvolution, or Deconvolution

Stride: 3

Padding: 1



3x3



2x2

4x4

Parameter sharing

Convolutions
Downsampling
Upsampling

Skip connections

Trade-off between classification and localization

- High level features from later in the network, enable high classification performance, since they are more discriminative and contain more useful semantic information.
- On the other hand, those deep features have low resolution and, thus pose a problem for localization performance.
- Problem of vanishing gradients was solved with skip connections.

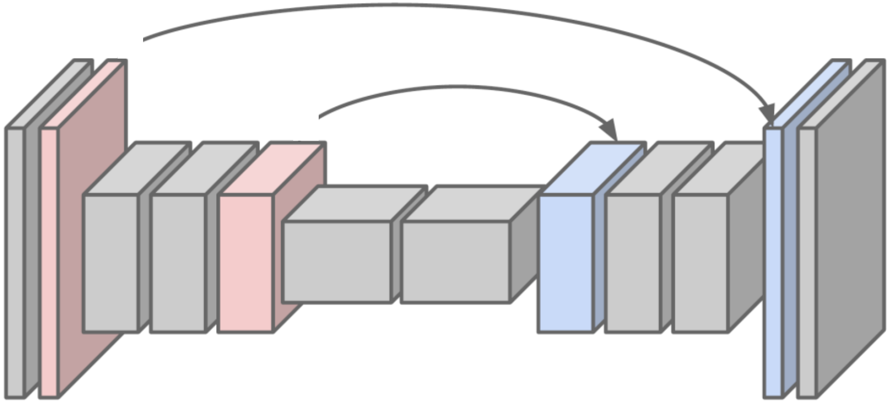
Neural Network Segmentation - Basic structure

Convolutions
Downsampling
Upsampling

Skip connections

Combining low-level features, which have high localization accuracy

With the high-level features, which have are descriptive but low-resolution.



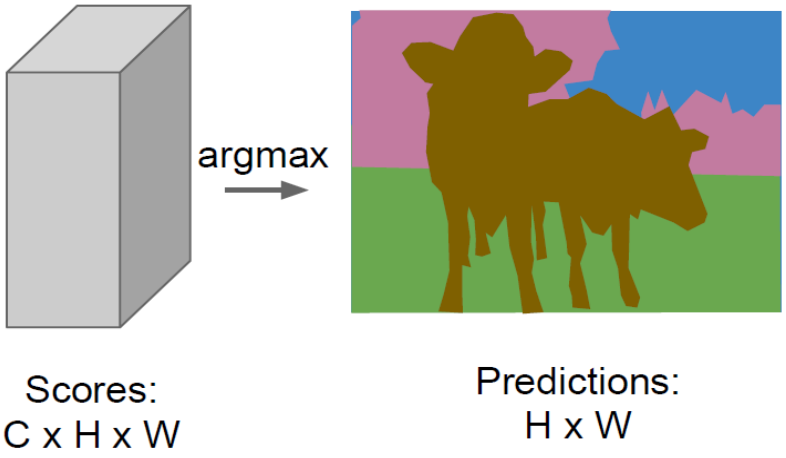
http://cs231n.github.io/

Last layer

Last Layer results in a tensor with H x W image resolution and a depth of
C: Number of classes to segment.

The last layer should encode the values into a range of values of (0;1).
Usually either with softmax or sigmoid function.

Cross-entropy
Dice-coefficient



http://cs231n.github.io/

Sigmoid

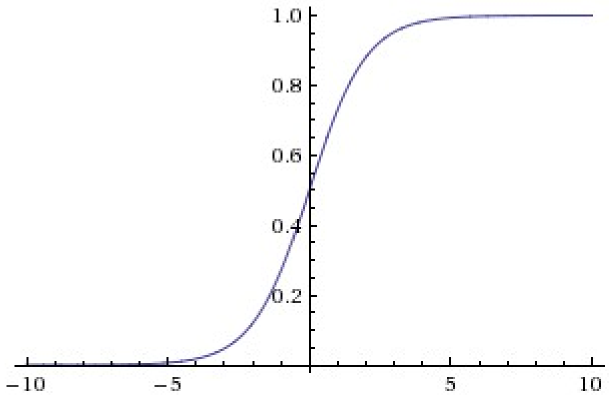
$$\hat{y}_i = 1 - \frac{1}{1 + e^{-x_i}}$$

Binary classification only.

The probability for different classes does not need to sum to one.

Simply take the highest class probability.

class	model out (x_i)	prob out ($\hat{y}_i$)
cow	2.0	0.88
grass	1.0	0.73
background	-3.0	0.47



http://cs231n.github.io/neural-networks-1/

Softmax

Normalized exponential function

$$\hat{y}_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

Often used for multi-class segmentation.

Probability sum will be 1.

class	**model out (x_i)**	**prob out ($\hat{y}_i$)**
cow	2.0	0.72
grass	1.0	0.27
background	0.1	0.01

Neural Network Segmentation - Optimization

Last layer

Binary Cross-entropy

Dice-coefficient

$$L = -\frac{1}{n}\sum_{i=1}^{n}(\tilde{y}_i \cdot \log(\hat{y}_i) + (1 - \tilde{y}_i) \cdot \log(1 - \hat{y}_i))$$

$$\tilde{y}_i = expected\ pixel\ class\ (boolean)$$

$$\hat{y}_i = predicted\ pixel\ class, \hat{y}_i \in (0,1)$$

Last layer

Binary Cross-entropy

Dice-coefficient

$$L = -\frac{1}{n}\sum_{i=1}^{n}(\tilde{y}_i \cdot \log(\hat{y}_i) + (1 - \tilde{y}_i) \cdot \log(1 - \hat{y}_i))$$

$\tilde{y}_i = expected\ pixel\ class\ (boolean)$

$\hat{y}_i = predicted\ pixel\ class, \hat{y}_i \in (0,1)$

$\tilde{y}_i \cdot \log(\hat{y}_i) = error\ from\ positive\ class$

$(1 - \tilde{y}_i) \cdot \log(1 - \hat{y}_i) = error\ from\ negative\ class$

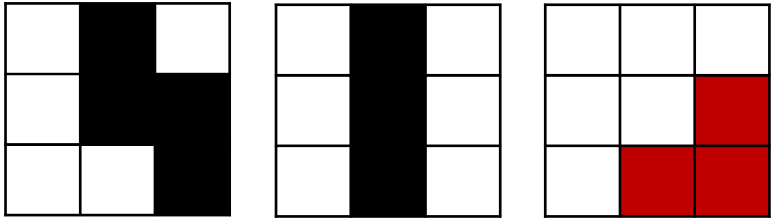
Neural Network Segmentation - Optimization

Last layer

Binary Cross-entropy

$$L = -\frac{1}{n}\sum_{i=1}^{n}(\tilde{y}_i \cdot \log(\hat{y}_i) + (1 - \tilde{y}_i) \cdot \log(1 - \hat{y}_i))$$

Dice-coefficient



prediction

ground truth

error 0.33

Last layer

Binary Cross-entropy

Dice-coefficient

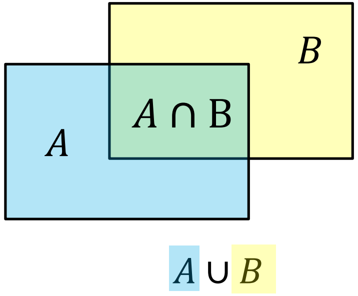
$$\mathrm{L} = 1 - \frac{2\sum_i^n \tilde{y}_i \cdot \hat{y}_i}{\sum_i^n \tilde{y}_i + \sum_i^n \hat{y}_i}$$

$\tilde{y}_i = expected\ pixel\ class\ (boolean)$

$\hat{y}_i = predicted\ pixel\ class$

Similar to the IoU (Intersection over union)

More robust with respect to imbalanced classes.



Neural Network Segmentation - Optimization

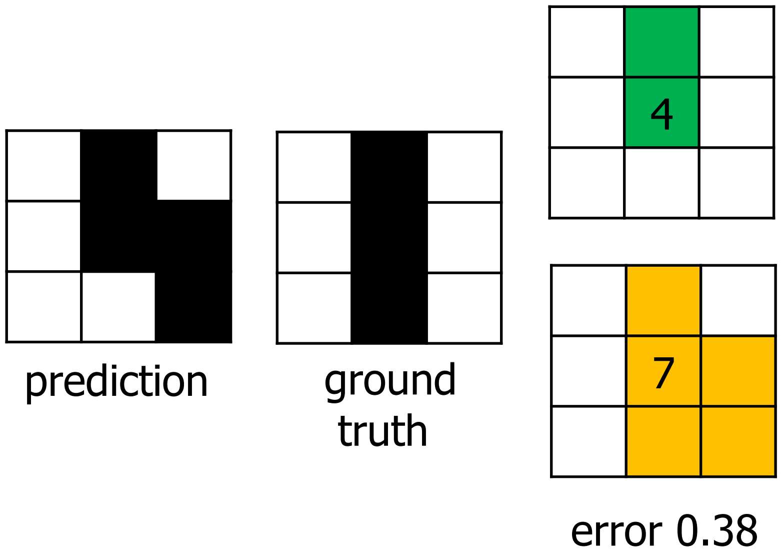
Last layer
Binary Cross-entropy

Dice-coefficient

Similar to the IoU (Intersection over union)

More robust with respect to imbalanced classes.

$$L = 1 - \frac{2\sum_i^n \tilde{y}_i \cdot \hat{y}_i}{\sum_i^n \tilde{y}_i^2 + \sum_i^n \hat{y}_i^2}$$

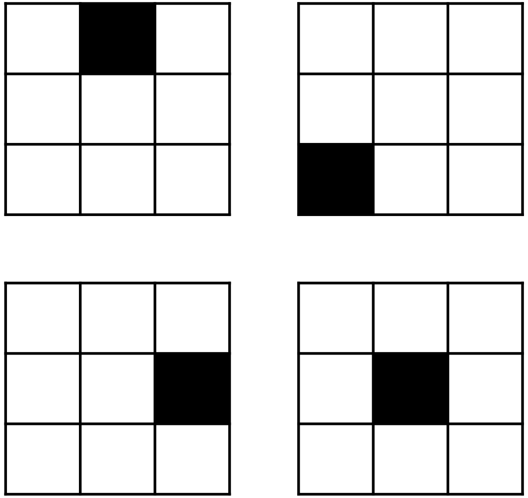


Thought experiment (at home)

Assumption:

The maximum number of class 1 pixels in a single sample is 1.

This simulates an extreme class imbalance ratio of 1 to 8.



ground truth

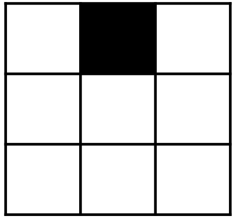
Neural Network Segmentation - Imbalanced classes

Thought experiment (at home)

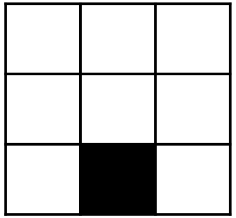
Assumption:

The maximum number of class 1 pixels in a single sample is 1

What is the expected error if the model tries to predict a single pixel of class 1 and fails?



prediction



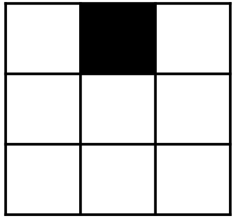
ground truth

Neural Network Segmentation - Imbalanced classes

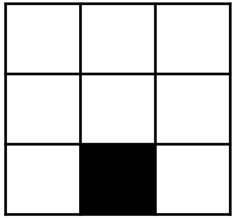
Thought experiment (at home)

Assumption:
The maximum number of class 1 pixels in a single sample is 1

What is the expected error if the model tries to predict a single pixel of class 1 and fails?



prediction



ground truth

BCE
0.22

DL
0.66

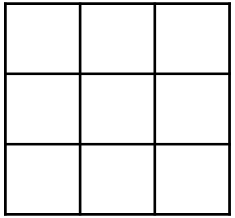
Neural Network Segmentation - Imbalanced classes

Thought experiment (at home)

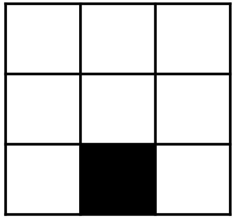
Assumption:
The maximum number of class 1 pixels in a single sample is 1

What is the expected error if the model tries to predict a single class 1?

What is the maximal expected error if the model predicts class 2 only?



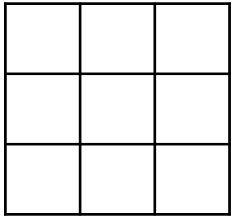
prediction

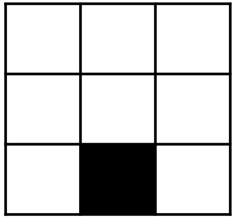


ground truth

Neural Network Segmentation - Imbalanced classes

Thought experiment (at home)


prediction


ground truth

Assumption:
The maximum number of class 1 pixels in a single sample is 1

What is the expected error if the model tries to predict a single class 1?

What is the maximal expected error if the model predicts class 2 only?

BCE	**DL**
0.11	**0.5**

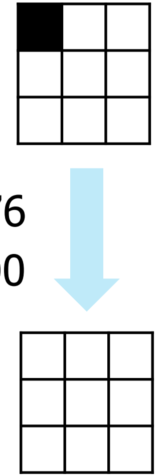
Thought experiment (at home)

How high is the pressure to get locked in a local minimum if predictions are initially random?

Thought experiment (at home)

How high is the pressure to get locked in a local minimum if predictions are initially random?

Binary cross-entropy



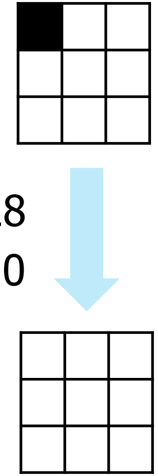
$$0.22 * 8 + 0.00 * 1 = 1.76$$
$$0.11 * 9 = 1.00$$

43 %

Thought experiment (at home)

How high is the pressure to get locked in a local minimum if predictions are initially random?

Dice-loss



$$0.66 * 8 + 0.00 * 1 = 5.28$$
$$0.50 * 9 = 4.50$$

15 %

How to deal with imbalanced classes

Choose dice-loss over cross-entropy.

How to deal with imbalanced classes

Choose dice-loss over cross-entropy.

Balance your cross-entropy according to the class imbalance.

In our case β *=7/8*

$$\mathrm{L} = -\frac{1}{n}\sum_{i=1}^{n}(\beta \cdot \tilde{y}_i \cdot \log(\hat{y}_i) + (1-\beta)(1-\tilde{y}_i) \cdot \log(1-\hat{y}_i))$$

PASCAL Visual Object Classes

Pixel-wise segmentation of objects from a number of visual object classes in realistic scenes (i.e. not pre-segmented objects).

Annotations

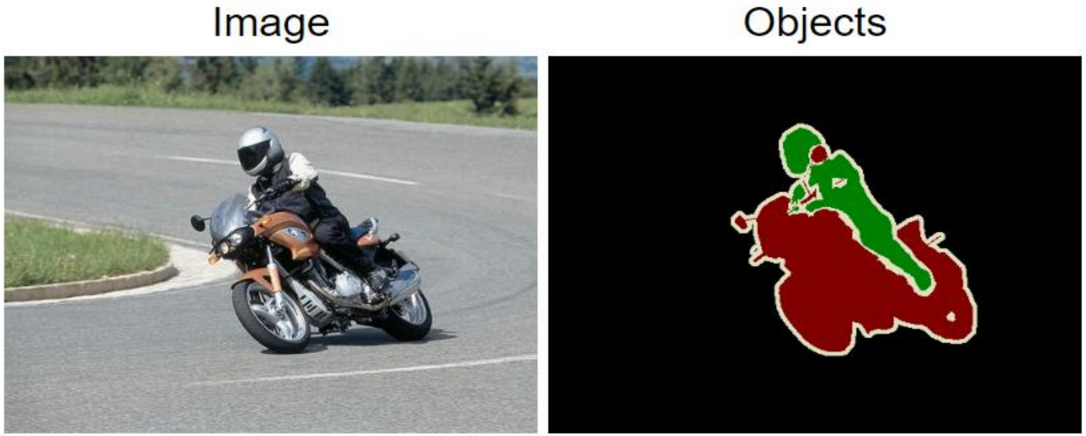
Person, animals, vehicles, indoor.

Number of samples

6929 Pixel-wise
instance level annotations.

Metric

Mean Intersection over Union.



http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html

Common Objects in Context

COCO-Stuff augments 164K images with pixel-level stuff annotations for semantic segmentation.

Annotations

91 stuff classes (wall, grass, etc.) and 80 thing classes (person, elephant, etc.), as well as captions.

Number of samples

164000 dense pixel-level annotations and instance level annotations for things.

Metric

Mean Intersection over Union.

https://github.com/nightrome/cocostuff

Cityscapes

The Cityscapes Dataset focuses on semantic understanding of urban street scenes.

Annotations

City scene semantic and instance-wise pixel annotations (road, person, pole, etc.).

Number of samples

30 classes in 5000 fine and 20000 coarse annotated images.

Metric

Mean Intersection over Union
and Instance Intersection over Union.

https://www.cityscapes-dataset.com/

Architectures over time

Fully Convolutional Network	2015
ParseNet	2015
Convolutional and Deconvolutional Networks	2015
U-Net	2015
Feature Pyramid Network	2016
Mask R-CNN	2017
DeepLab	2017

Fully Convolutional Network

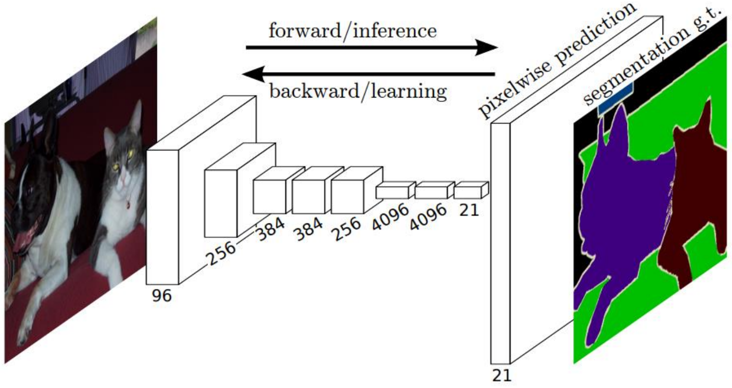
First end-to-end trained Fully Convolutional Network fc
image segmentation.

Transfer Learning approach, modifying well known architectures (such as VGG16).

Ending with an upsampling layer with one channel per class.

Fully Convolutional Networks for Semantic Segmentation

Jonathan Long* Evan Shelhamer* Trevor Darrell
UC Berkeley
{jonlong,shelhamer,trevor}@cs.berkeley.edu



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

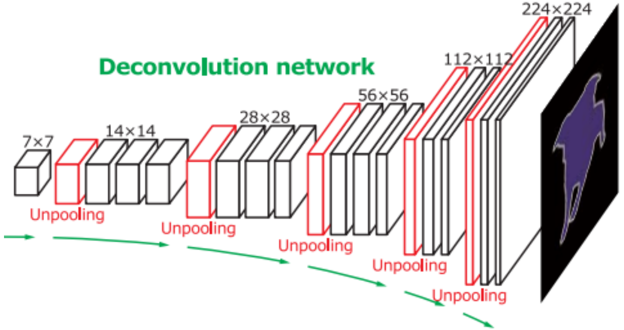
Neural Network Segmentation - Overview state-of

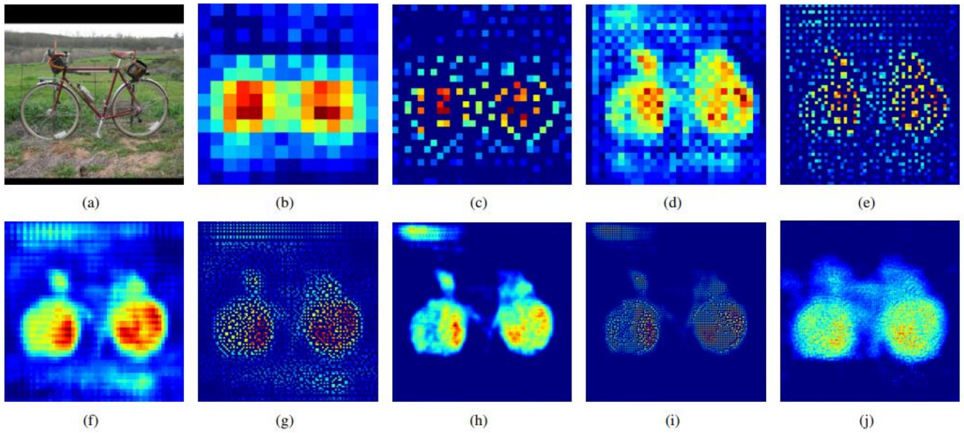
Convolutional and Deconvolutional Networks

Introducing a encoder-decoder architecture.

From the convolutional encoding, the deconvolution branch generates a dense pixel-wise class probability map, by successive:

Unpooling, deconvolutions, and rectifications.

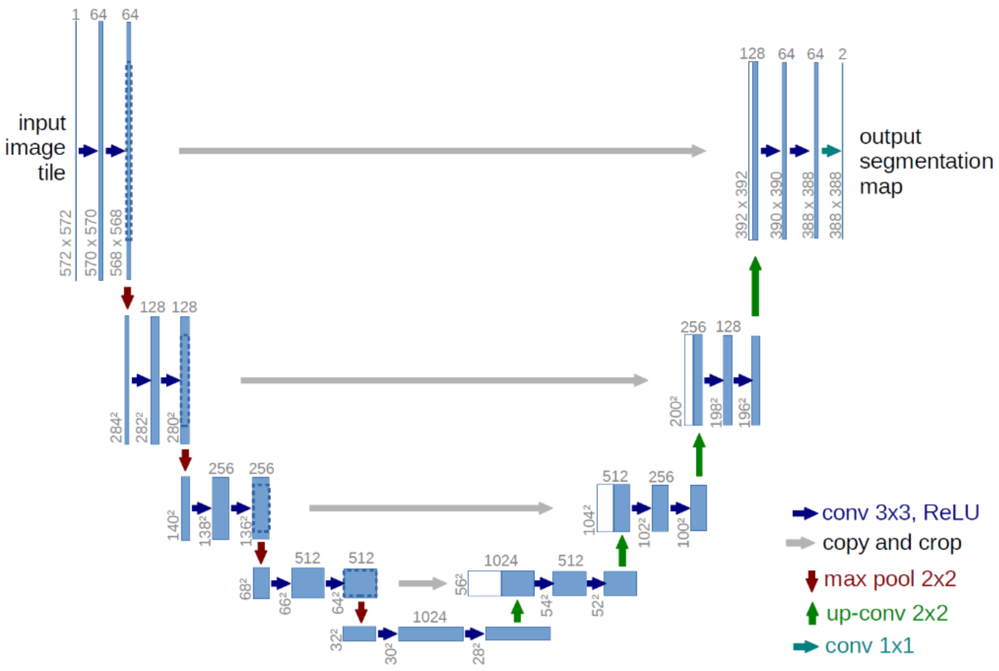




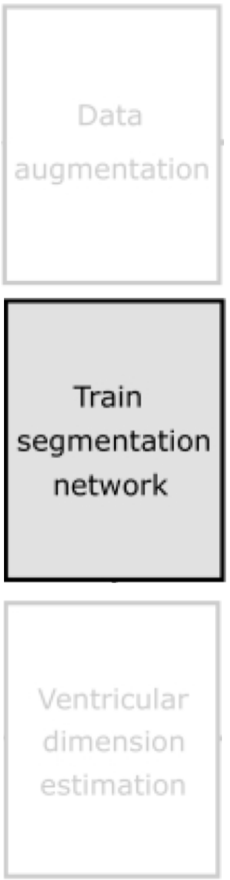
https://arxiv.org/pdf/1505.04366.pdf

Deep Dive U-Net

The U-Net is a symmetric, deep convolutional neural network.

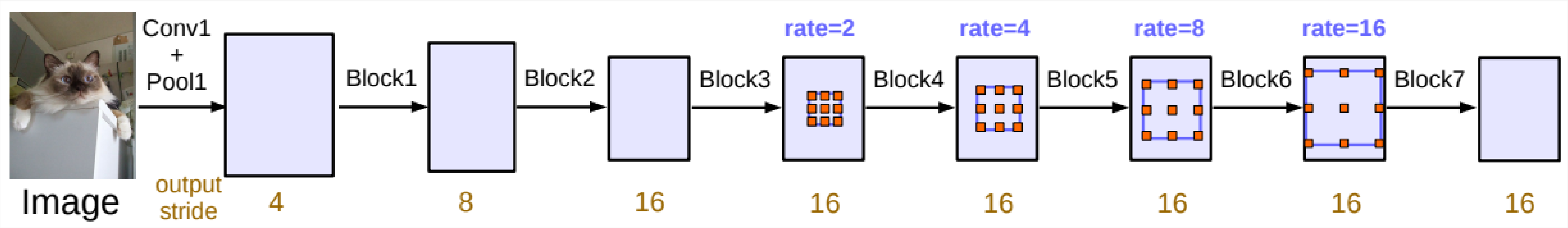


https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/



DeepLabv3

Combining Atrous Convolutions (dilated convolutions) with a pyramidal architecture.



https://arxiv.org/pdf/1706.05587.pdf

Atrous Convolution

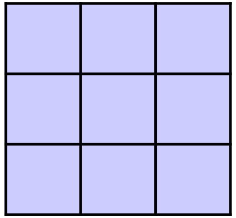
Introducing an additional parameter, called the dilation rate.

Defining a spacing between the values in a filter map.

Atrous Convolution

Introducing an additional parameter, called the dilation rate.

Defining a spacing between the values in a filter map.

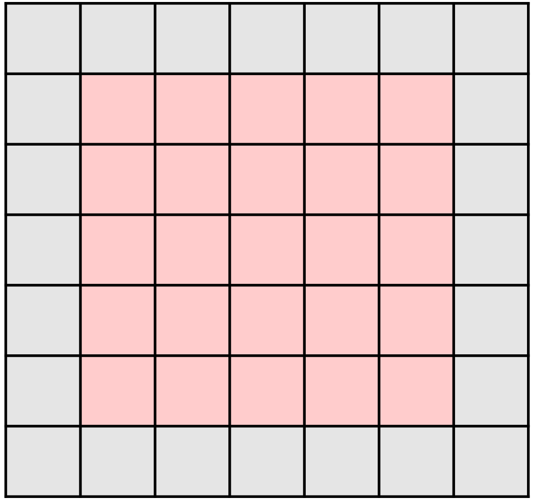


Filter 3x3

Dilation rate: 2

Stride: 1

Padding: 1

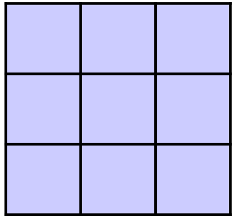


Atrous Convolution

Introducing an additional parameter, called the dilation rate.

Defining a spacing between the values in a filter map.

This enhances the field of view while keeping the computational cost low.



Filter 3x3

Dilation rate: 2

Stride: 1

Padding: 1