Data Science & Artificial Intelligence

Summer Term 2022

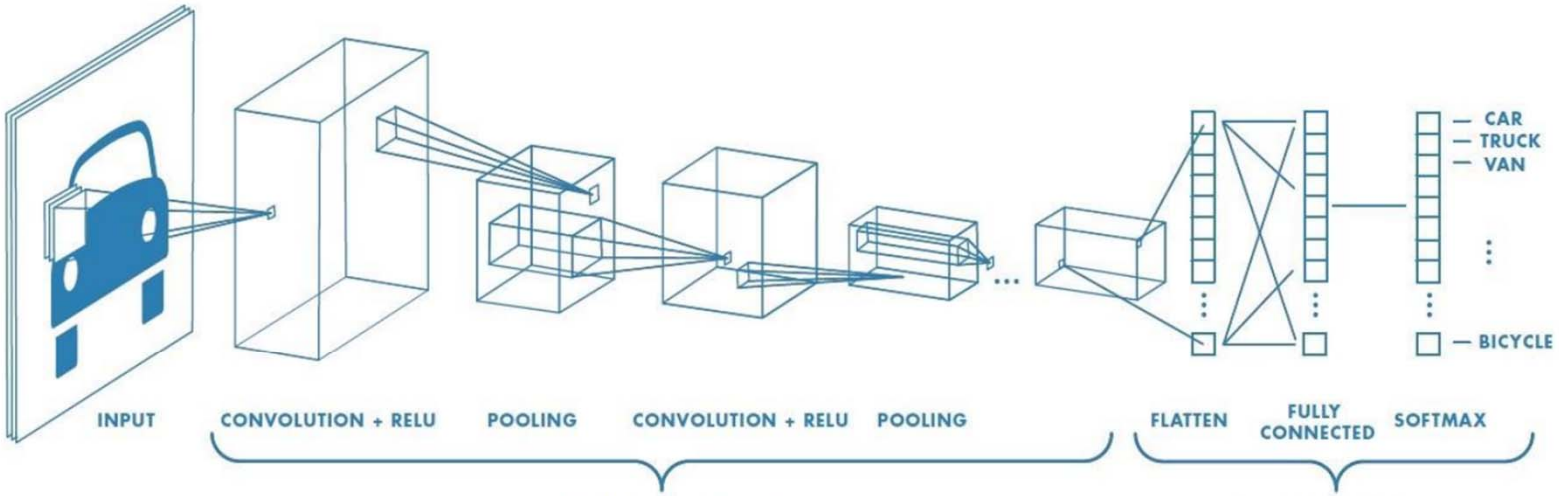
IFM Institut für Fahrerassistenz und vernetzte Mobilität

L10 CNNs Part II

J. Haselberger, B. Stuhr, D. Schneider

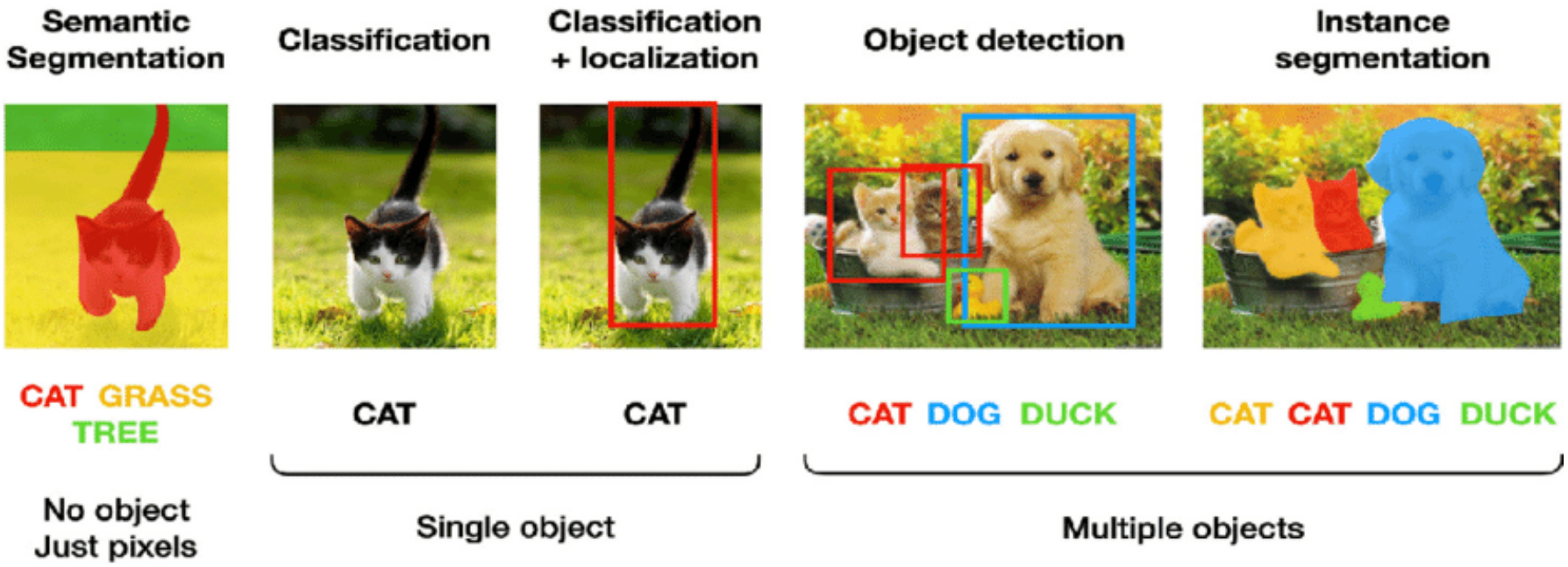
L 10.1 Recap

- What we have seen in the lecture so far:



L 10.1 Recap

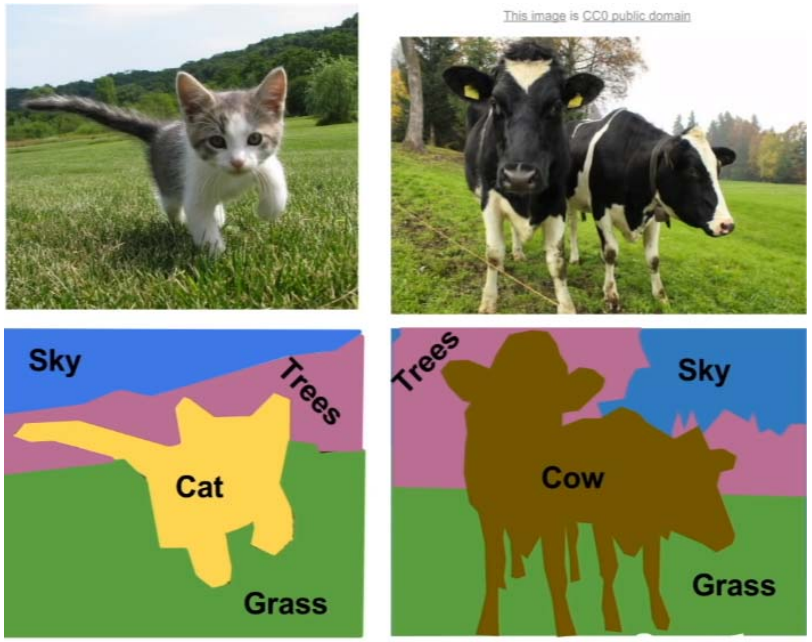
- But what about the other Computer Vision tasks?



L 10.2 Semantic Segmentation

Semantic Segmentation:

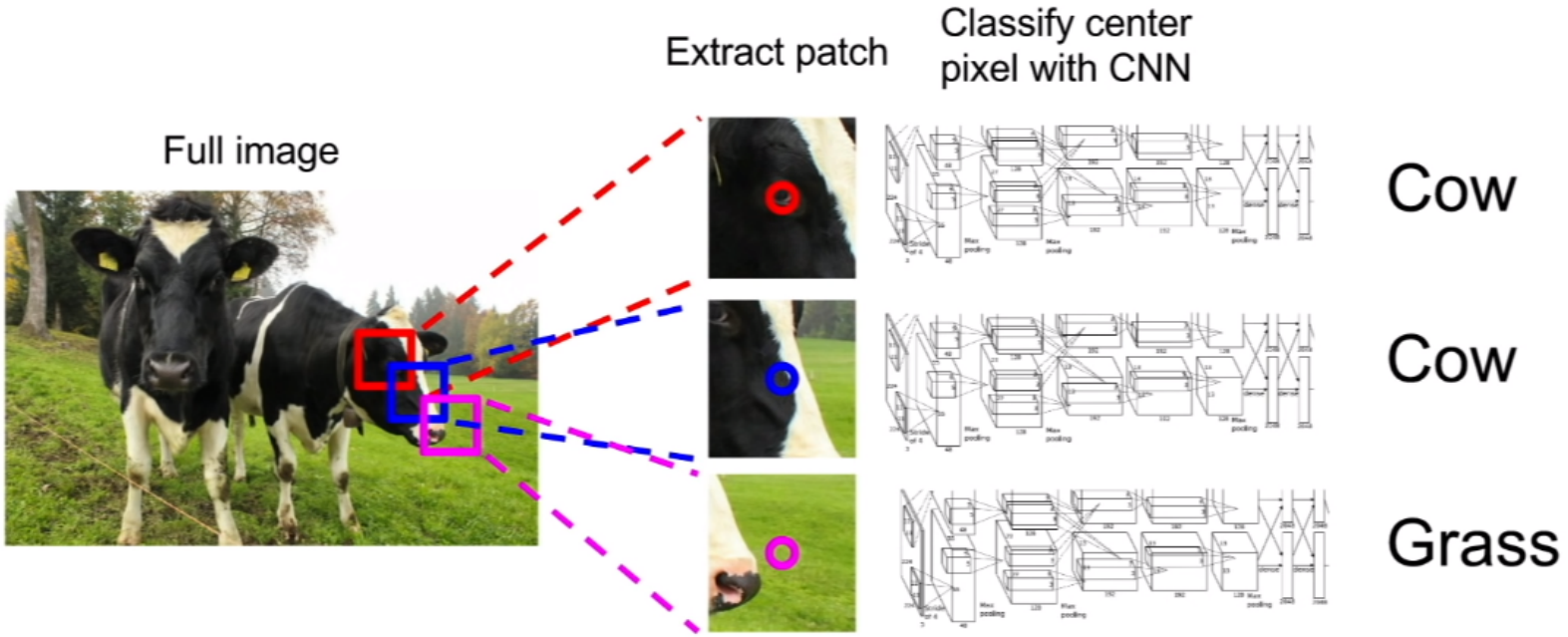
- Assign a category label to each pixel of the image
- no objects, just pixels are considered ➔ no instance differentiation



L 10.2 Semantic Segmentation

Intuitive Idea I:

- Sliding Window approach

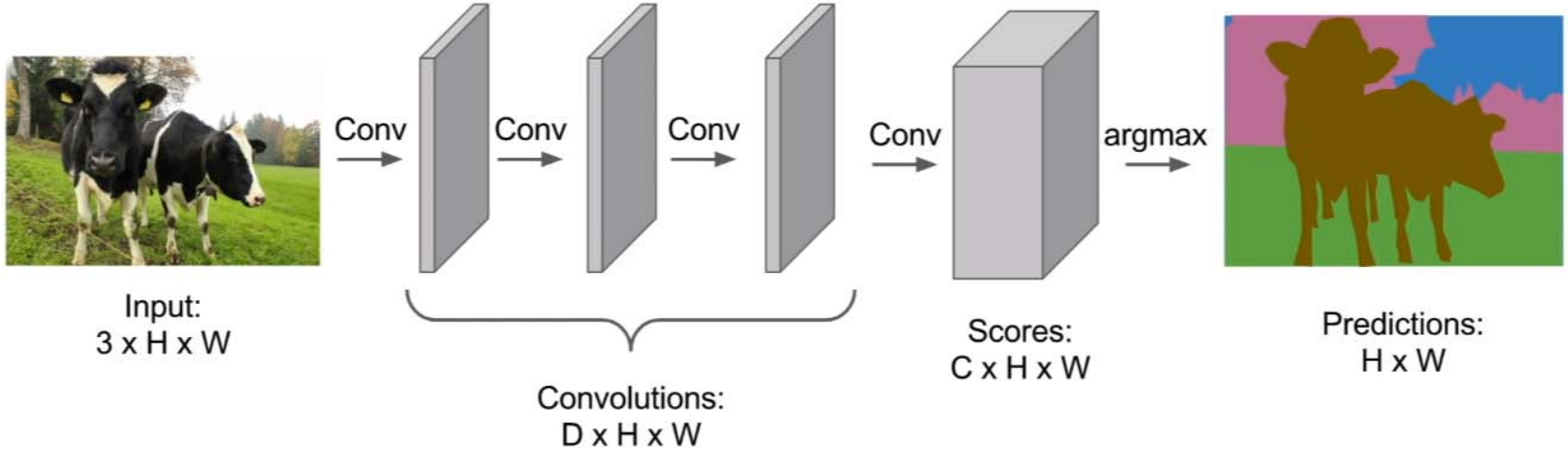


Computation time not manageable
CNN is run individually for each patch. Several thousand iterations!

L 10.2 Semantic Segmentation

Intuitive Idea II:

- Fully Convolutional
- network of convolution layers to make all pixel predictions at once

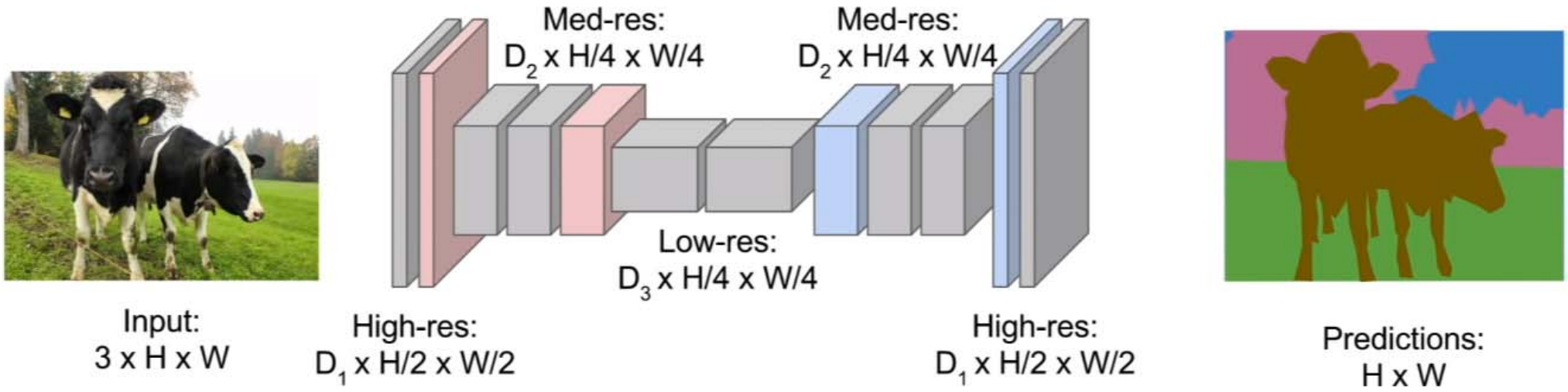


High memory requirements: Convolution on high res images is expensive!
Without reducing the latent space, the memory requirement is nearly infeasible!

L 10.2 Semantic Segmentation

The Solution:

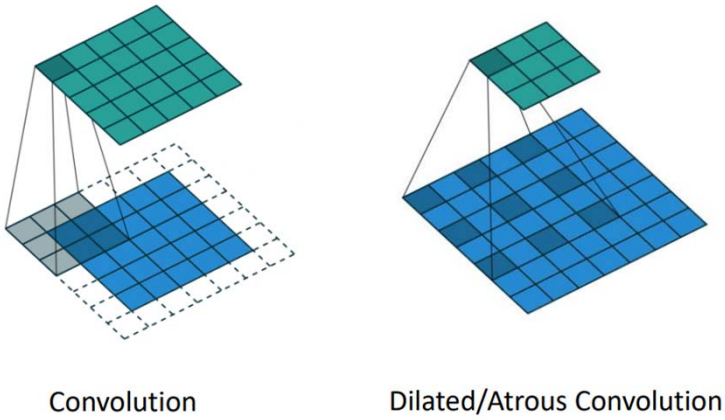
- Network architecture which down- and upsamples the data inside the network [Long2015, Noh2015]



L 10.2 Semantic Segmentation

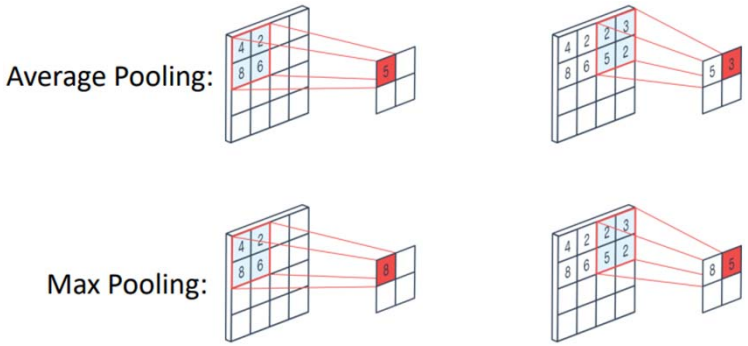
Recap: how does downsampling work?

- Convolution and Pooling layers



Convolution | Dilated/Atrous Convolution

Convolution

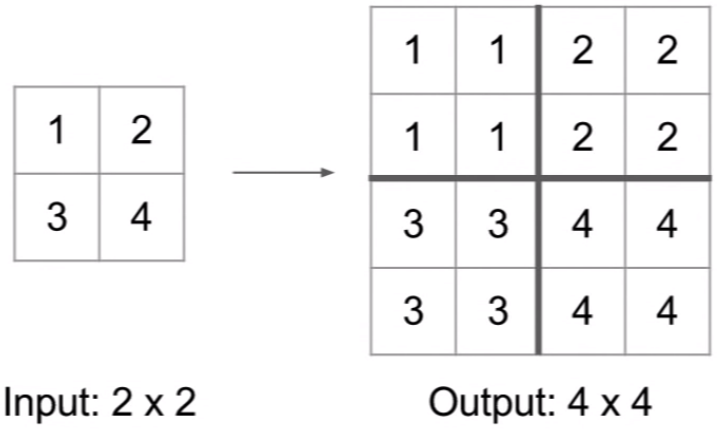


Pooling

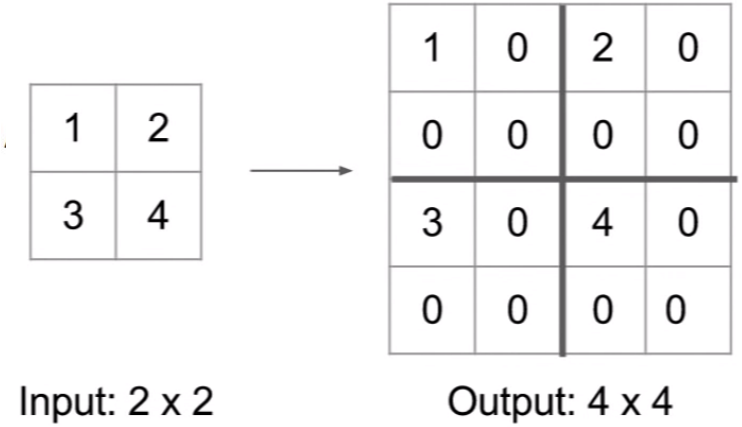
L 10.2 Semantic Segmentation

In-Network upsampling

- Unpooling



Nearest Neighbor

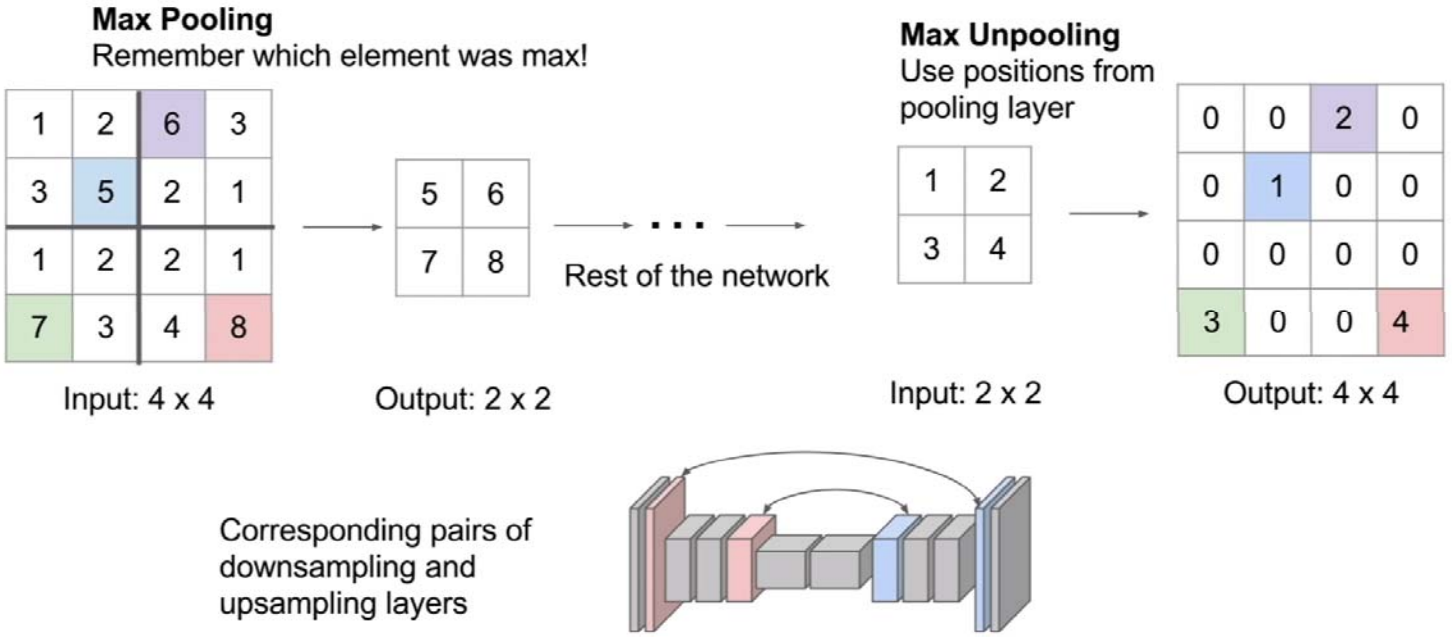


„Bed of Nails“

L 10.2 Semantic Segmentation

In-Network upsampling

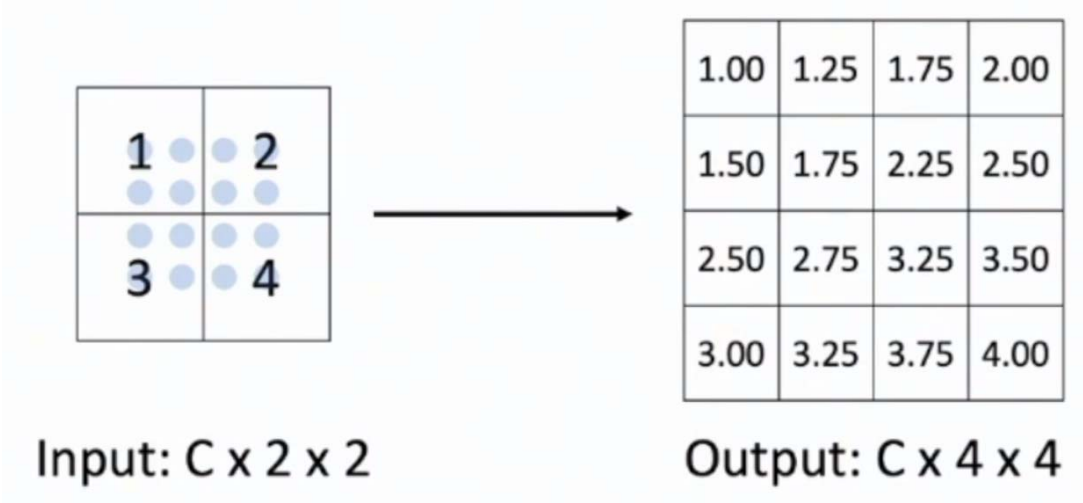
- Max Unpooling



L 10.2 Semantic Segmentation

In-Network upsampling

- Bilinear Interpolation

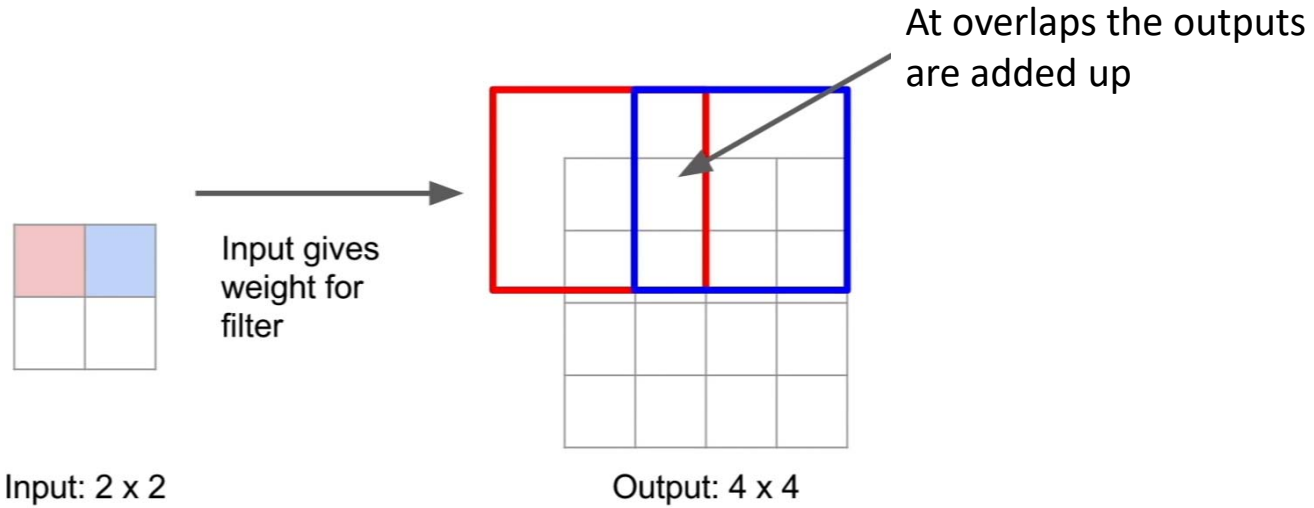


L 10.2 Semantic Segmentation

In-Network upsampling: Transpose Convolution

- In contrast to the fixed unpooling operations, the filter values of the Transpose convolution can be learned

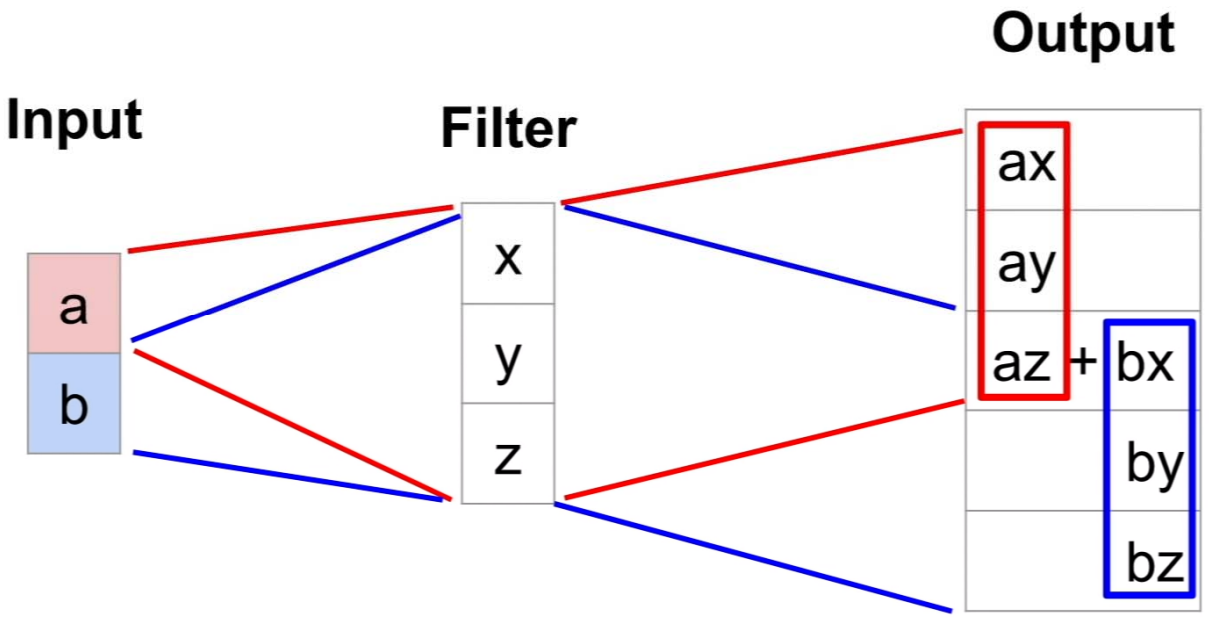
- Example: 3x3 filter kernel, stride=2, padding=1 ➔ 9 learnable kernel values



L 10.2 Semantic Segmentation

In-Network upsampling: Transpose Convolution

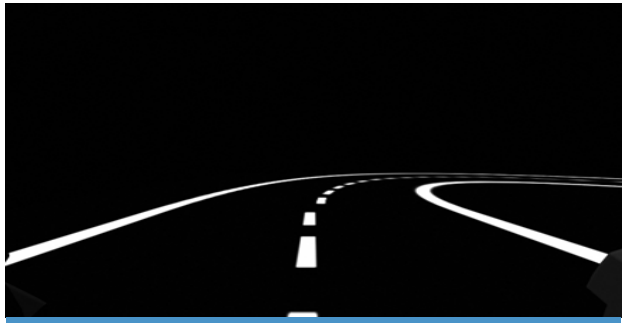
- 1D Example
- The output contains copies of the filter weighted by the input and summed in case of overlap



L10.2 Recap: Loss Functions - Segmentation

Overview Segmentation metrics

- The loss function for segmentation models also depends on the nature of the task



binary

- **only one class** which pixels are marked with 1
- remaining pixels are background and are marked with 0

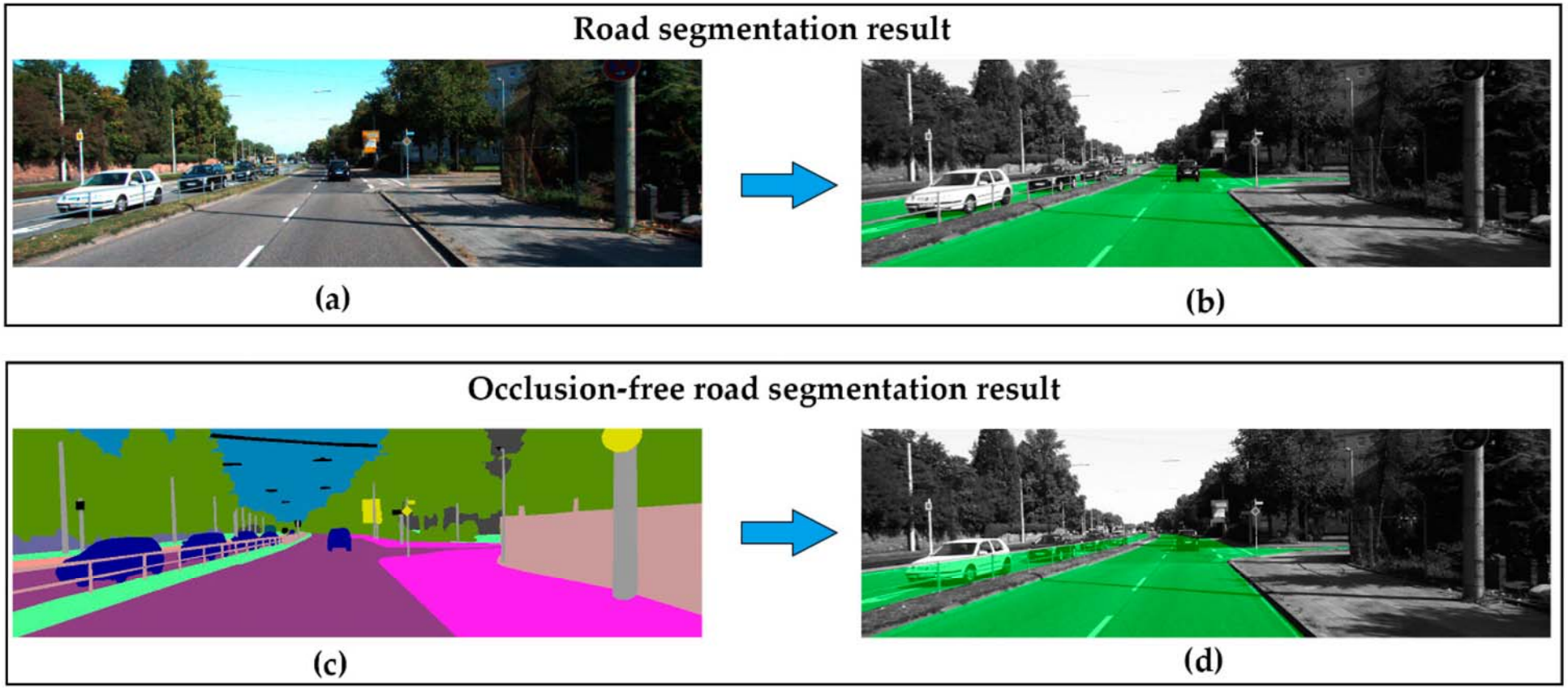
multiclass

- C = **1.. N classes** that have unique identification values
- **classes are mutually exclusive**
- all pixels are labeled

multilabel

- C = **1..N classes** which pixels are **labeled as 1**
- **classes are not mutually exclusive**
- **one channel per class**
- pixels in each channel that do **not belong to the class marked as 0**

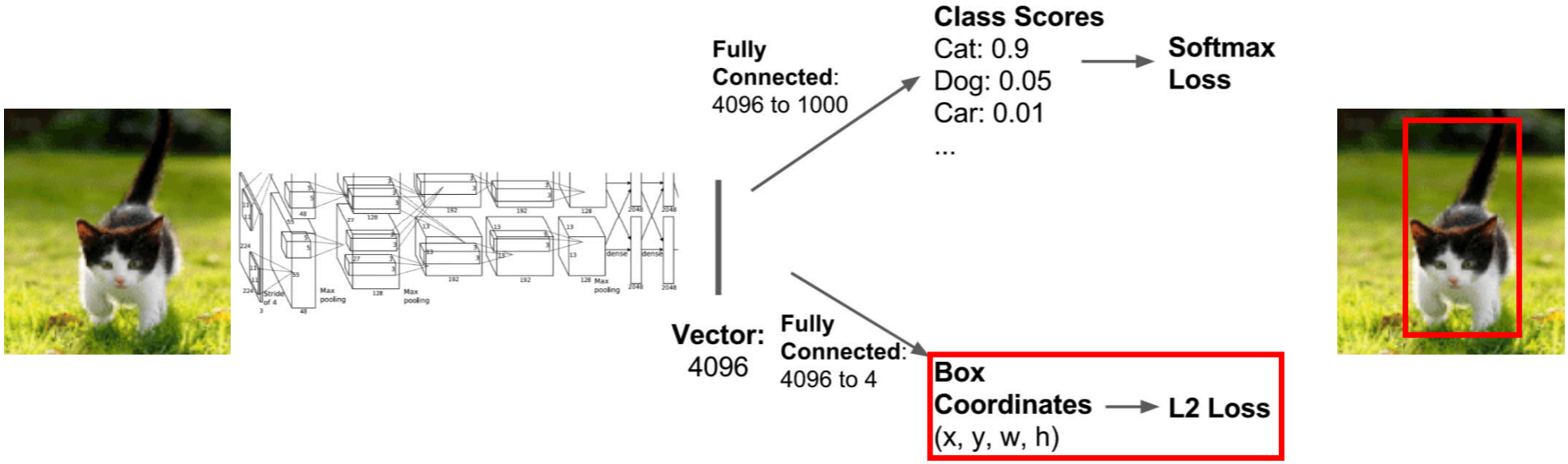
L10.2 Segmentation



L10.2 Segmentation

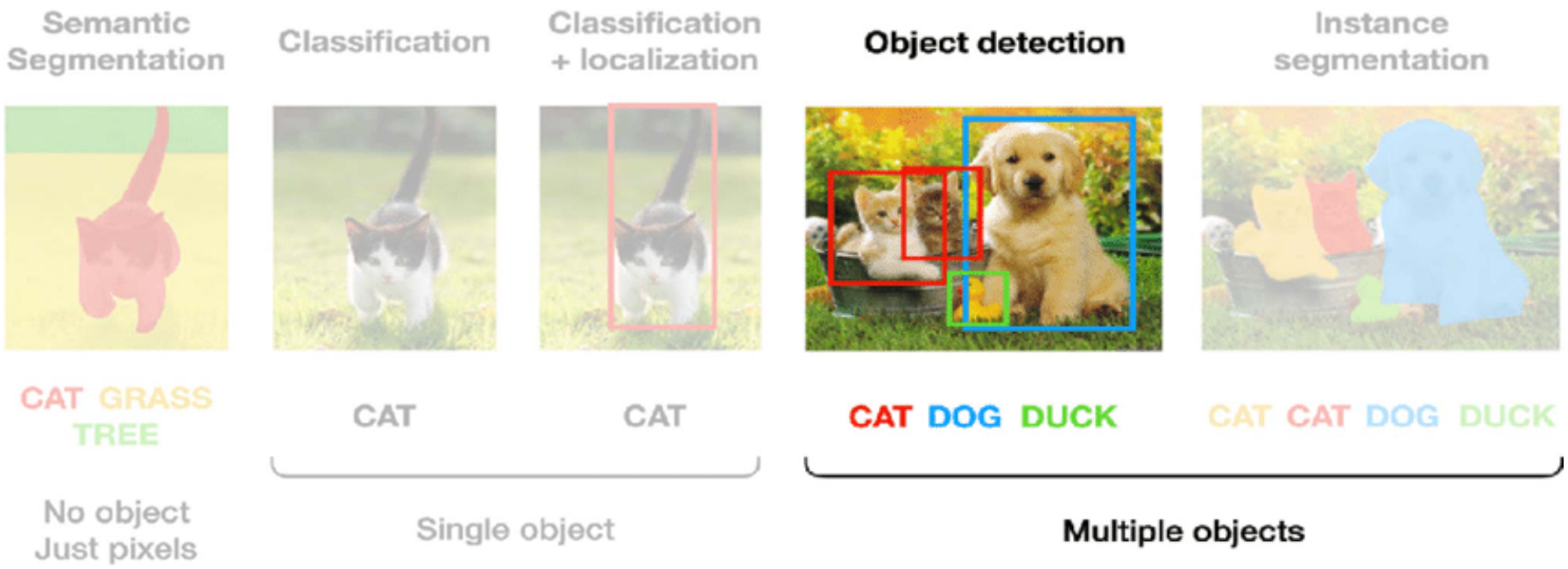
L 10.3 Classification + Localization

We treat localization as regression problem



L 10.4 Object Detection

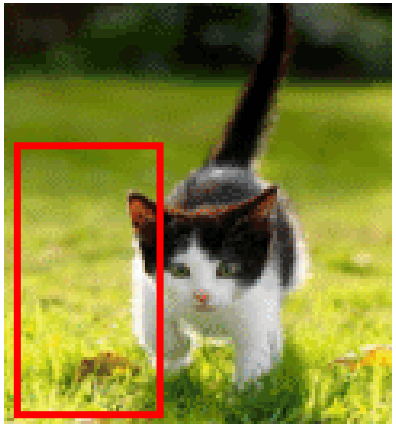
But what about multiple objects?

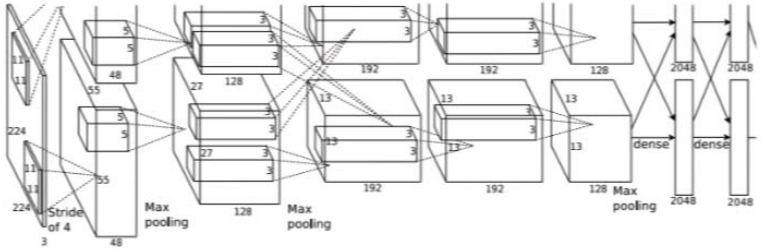


L 10.4 Object Detection

Intuitive Idea I:

- Sliding Window: apply a CNN to many different crops of the image
- CNN classifies each crop as object or background





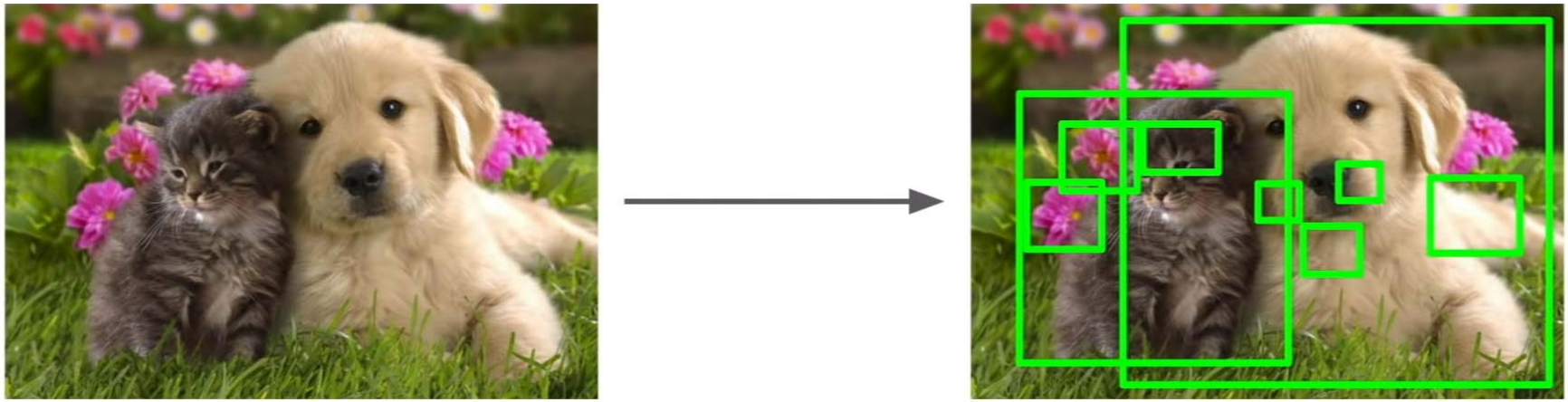
Dog? NO
Cat? NO
Background? YES

Computation time not manageable
For example: a 800x600 px image could have around 58M crops! Too many iterations per image

L 10.4 Object Detection

Solution Approach: Region Proposals

- Find „blobby“ image regions that are likely to contain objects
- Selective Search:
  - heuristic approach
  - fast to run: 1000 region proposals in few seconds

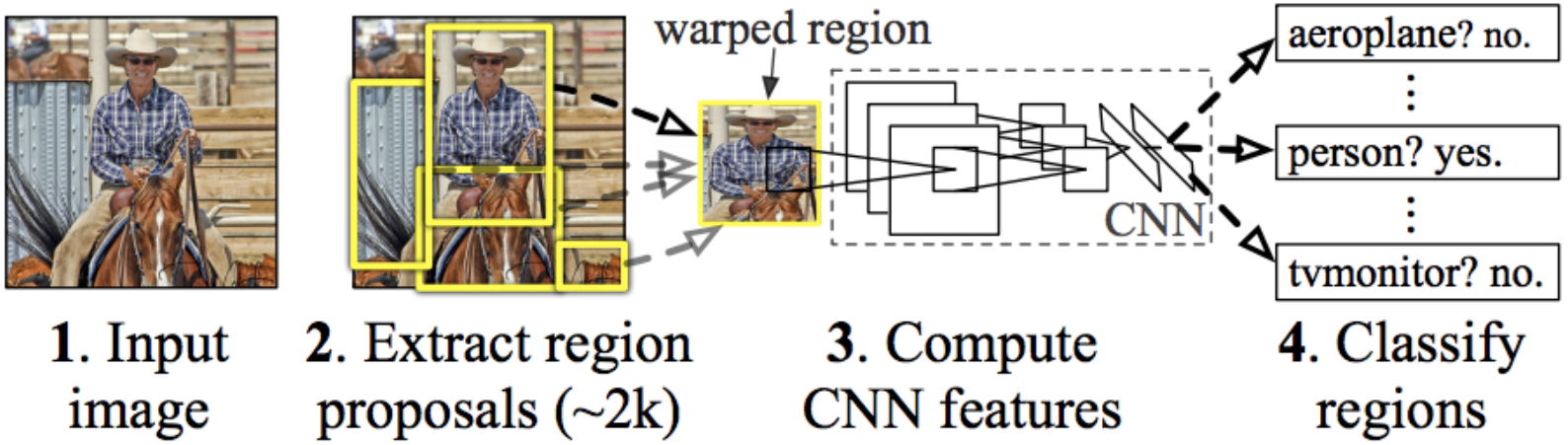


Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012
Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013
Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014
Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014

L 10.4 Object Detection

R-CNN: Regions with CNN features [Girshick2014]

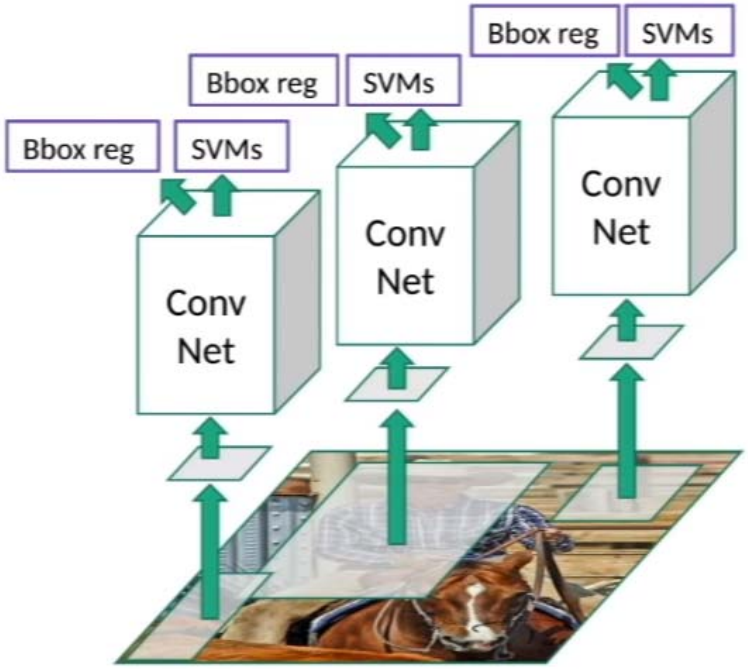
- Extraction of Regions of Interest
- warped images are feed through CNN
- latent space is used by SVM to classify the regions



L 10.4 Object Detection

R-CNN: Problems

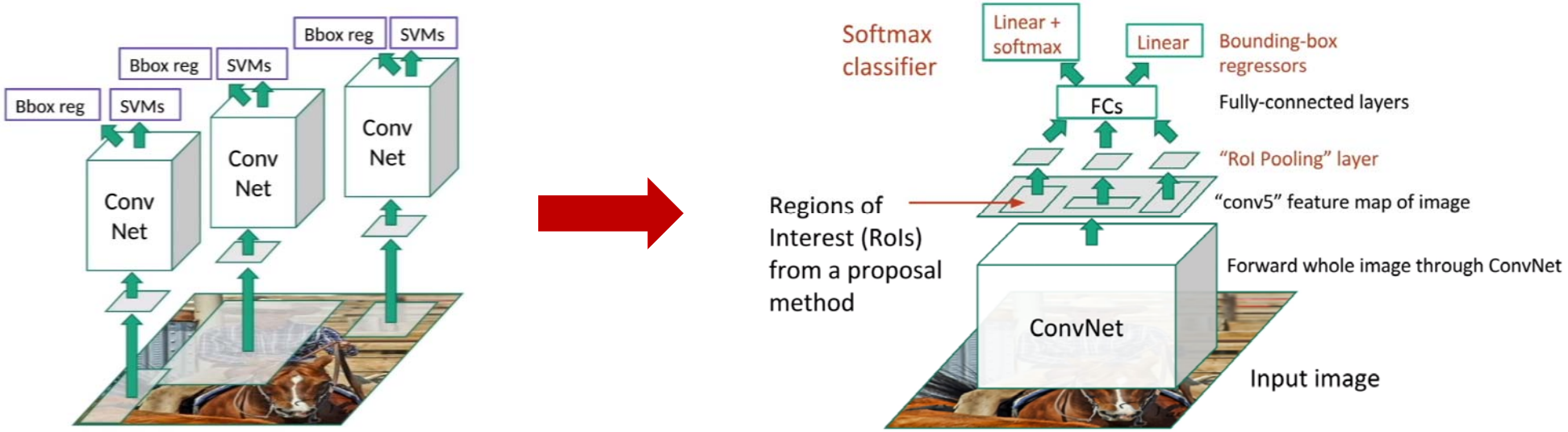
- Multiple training objectives
  - network finetuning with softmax classifier
  - training of the post-hoc SVM
  - training of the post-hoc bounding-box regressions
- Training is slow (84h) and takes a lot of disk space
- Inference is slow (49s per image)



L 10.4 Object Detection

Evolution I: Fast R-CNN [Girshick2015]

- before extracting the region proposals, the input image is feed through a CNN
- RoIs are extracted from the feature map
- Classification and Bounding Box Regression is performed by Fully-connected layers



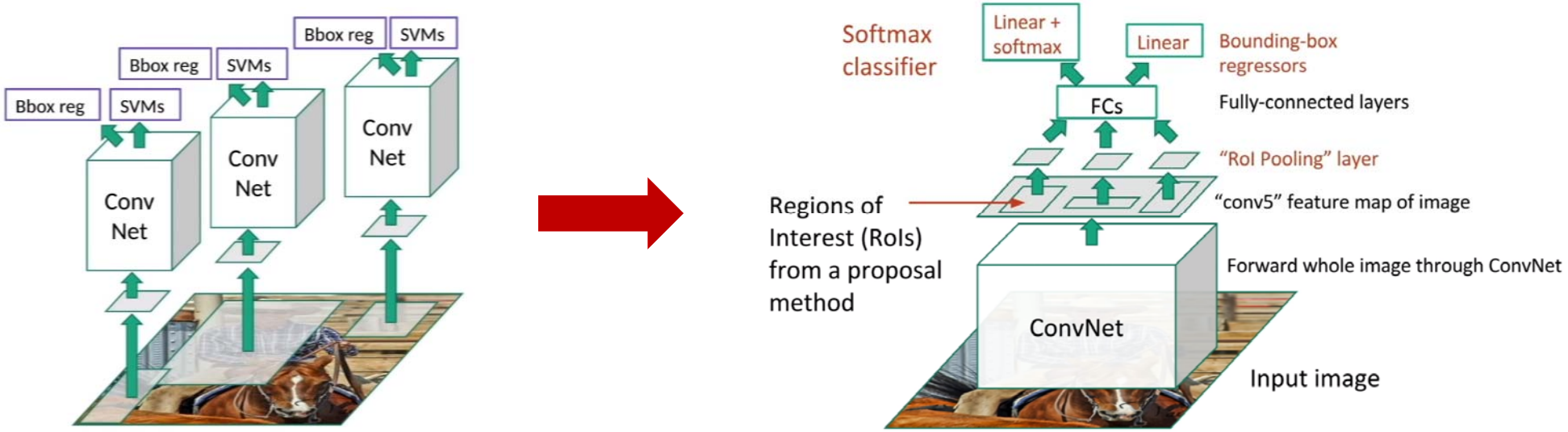
R-CNN

Fast R-CNN

L 10.4 Object Detection

Evolution I: Fast R-CNN [Girshick2015]

- before extracting the region proposals, the input image is feed through a CNN
- RoIs are extracted from the feature map
- Classification and Bounding Box Regression is performed by Fully-connected layers



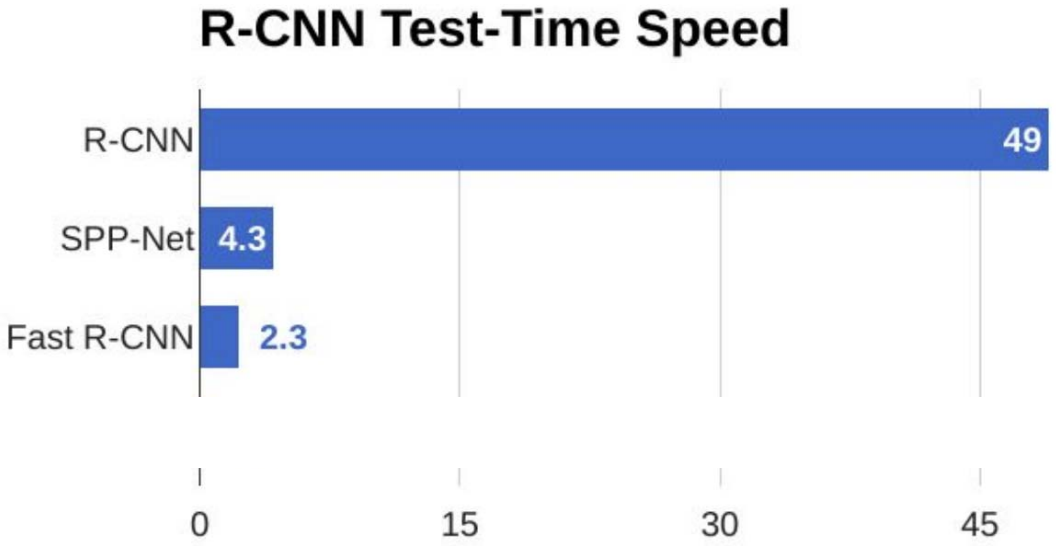
R-CNN

Fast R-CNN

L 10.4 Object Detection

Evolution I: Fast R-CNN [Girshick2015]

- but this is still not enough
- Experiments have shown that the region proposal approach is now the bottleneck of performance

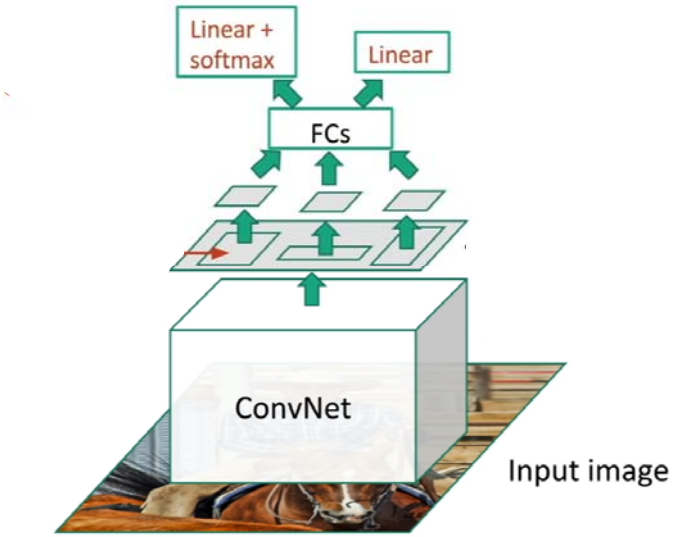


Apply CNN to reduce dimensionality, but whats next?

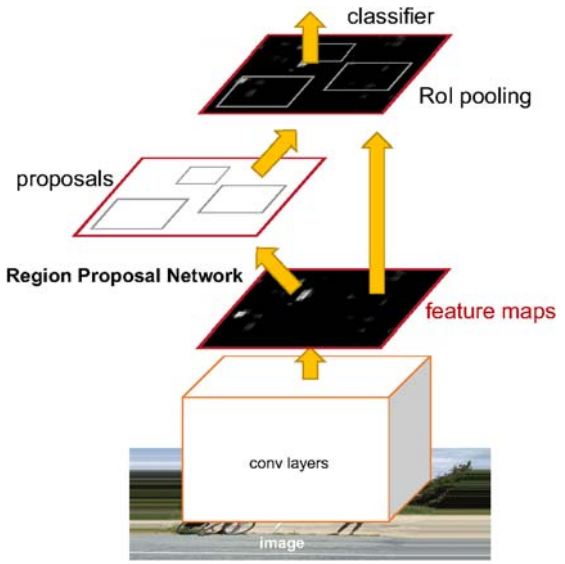
L 10.4 Object Detection

Evolution II: Faster R-CNN [Ren2015]

- Let the CNN do proposals ➔ Region Proposal Network (RPN) ➔ Learnable region proposals
- Now we have four objectives: RPN classification, RPN box coordinates regression, final classification, final box coordinates



Fast R-CNN

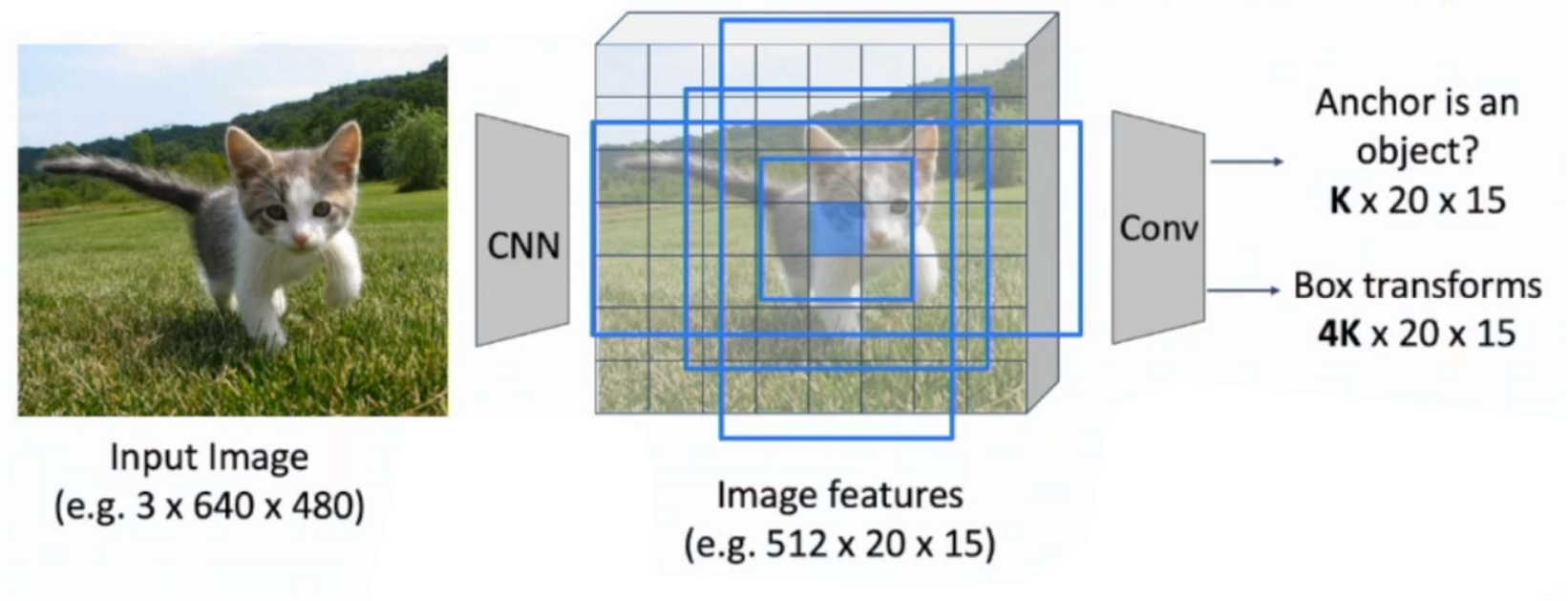


Faster R-CNN

L 10.4 Object Detection

Region Proposal Network (RPN)

- Backbone model is used to get features aligned to the input image
- To solve the problem that the Anchor Box may have the wrong size and shape, K different anchor boxes are used at each point



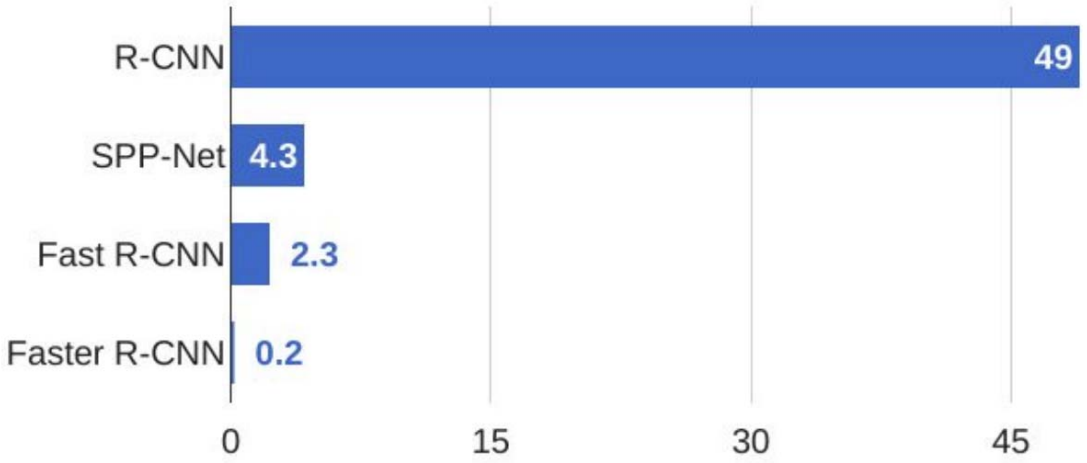
L 10.4 Object Detection

Two-stage object detection

- Faster R-CNN is a two-stage object detector

- **First Stage (run once per image):**
  - Backbone model
  - Region Proposal Network

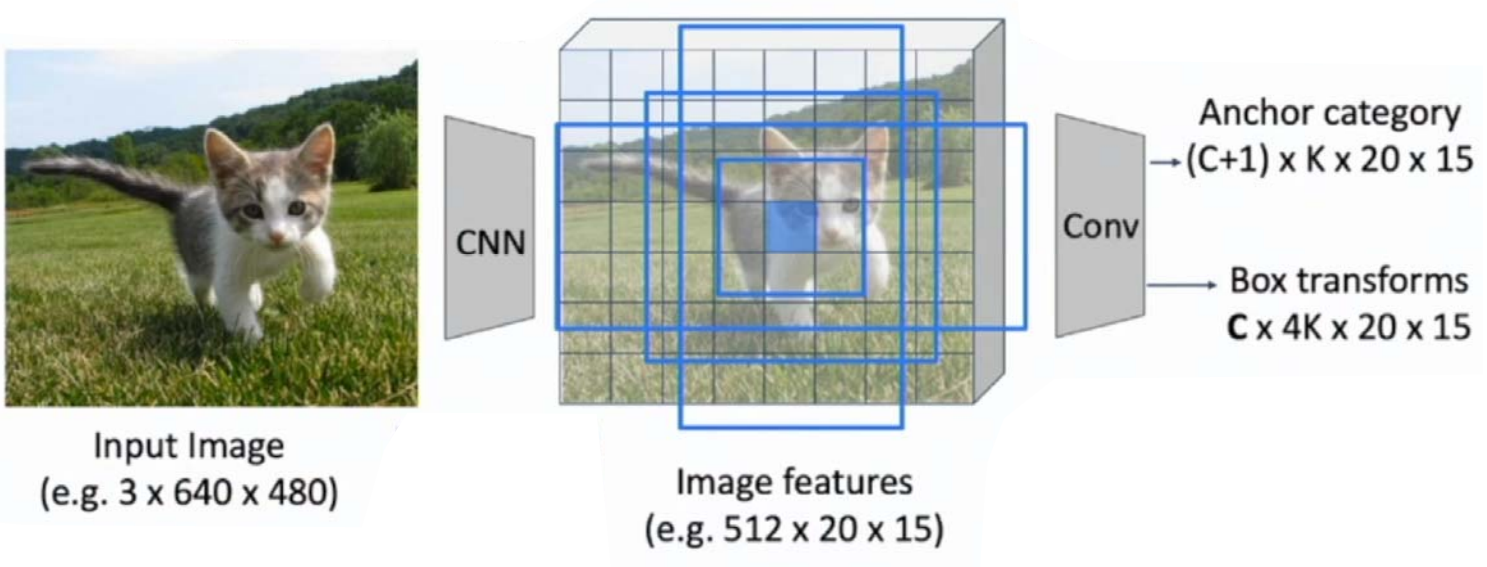
- **Second Stage (run once per region):**
  - Crop features RoI pool / align
  - Predict object class
  - Predict Bounding Box offset



L 10.5 Single Stage Object Detection

Detection without proposals

- From input image to output tensor with only one big network
- You Only Look Once (YOLO) [Redmon2016] and Single-Shot MultiBox Detector (SSD) [Liu2016]

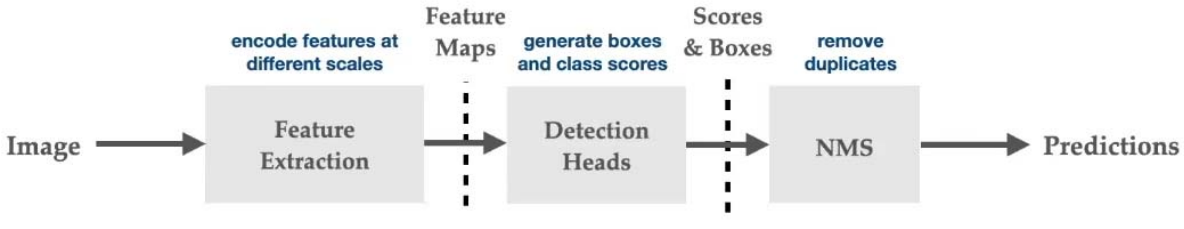


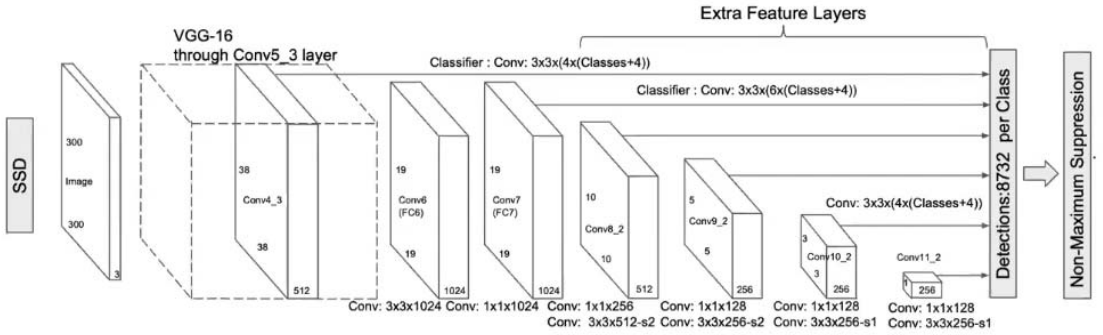
L 10.5 Single Stage Object Detection

Single-Shot MultiBox Detector (SSD) [Liu2016]

Key ideas:

- multiple layers ➔ handle different scales
- different filters predict boxes of different shapes and sizes

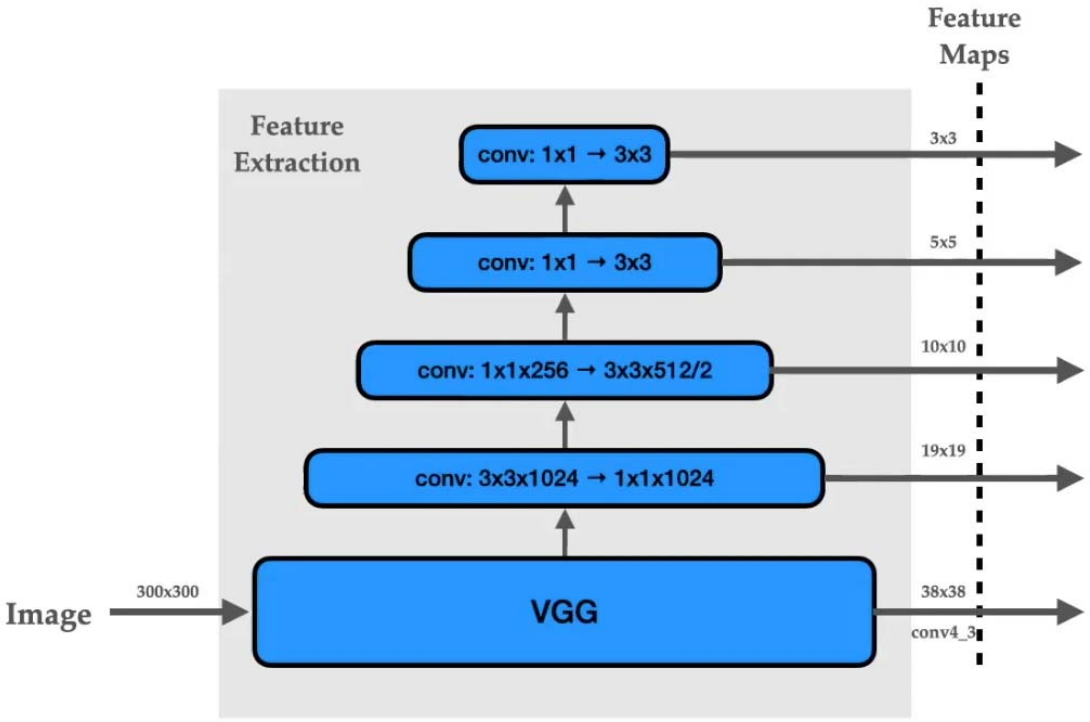




L 10.5 Single Stage Object Detection

Single-Shot MultiBox Detector (SSD) [Liu2016]

Feature Extraction: stacked convolution layers to handle different scales of the objects

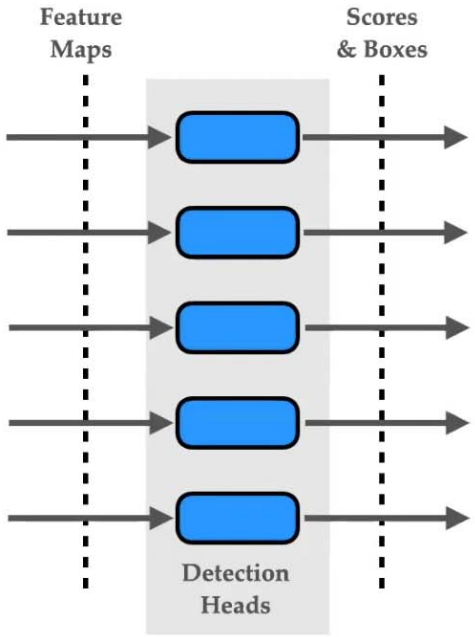


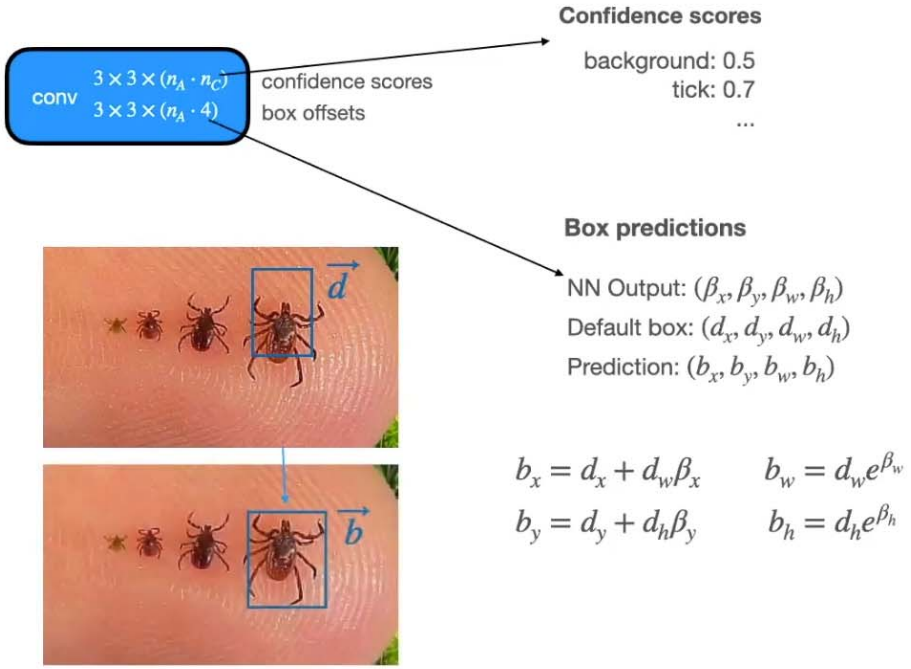
Flexible architecture
Backbone can be changed to newer models

L 10.5 Single Stage Object Detection

Single-Shot MultiBox Detector (SSD) [Liu2016]

Multibox Detector: different filters predict boxes of different shapes and sizes





L 10.5 Side Topic: Evaluationg Boxes

Intersection over Union (IoU)

- also called „Jaccard similarity“ or „Jaccard index“

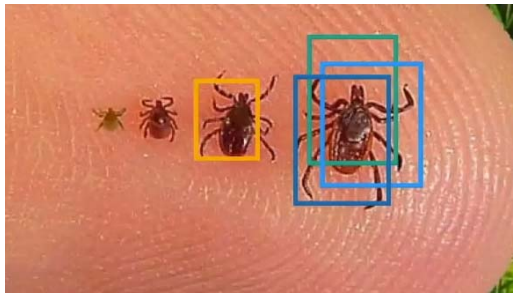
$$IoU = \frac{\text{area of intersection}}{\text{area of union}}$$

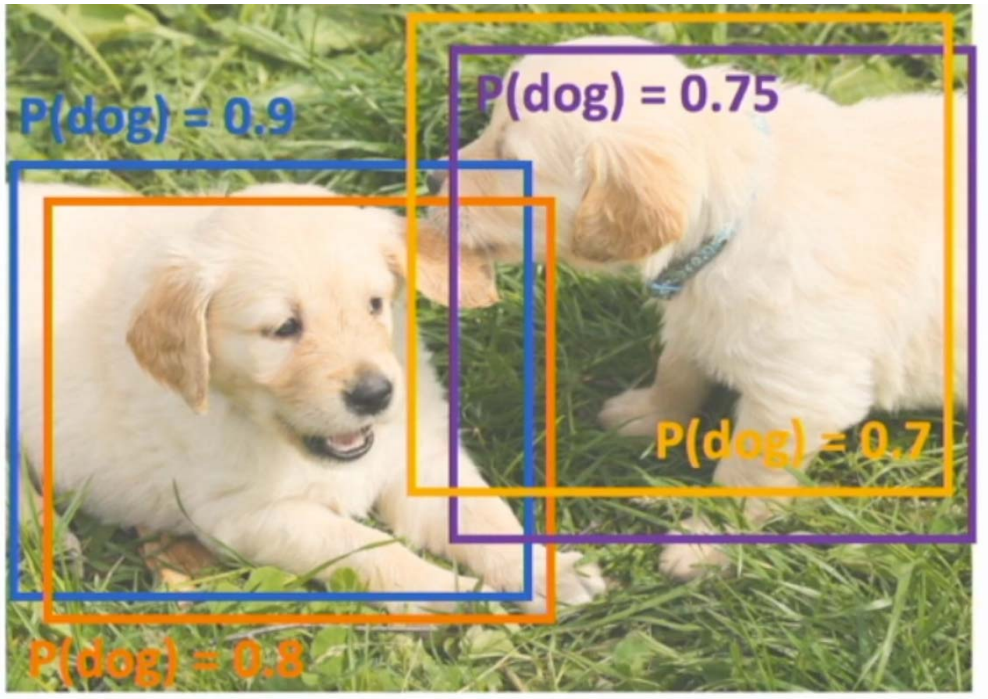
L 10.5 Side Topic: Non Maximum Suppression

Non Maximum Suppression (NMS)

- Problem: Object detectors often output many overlapping detections

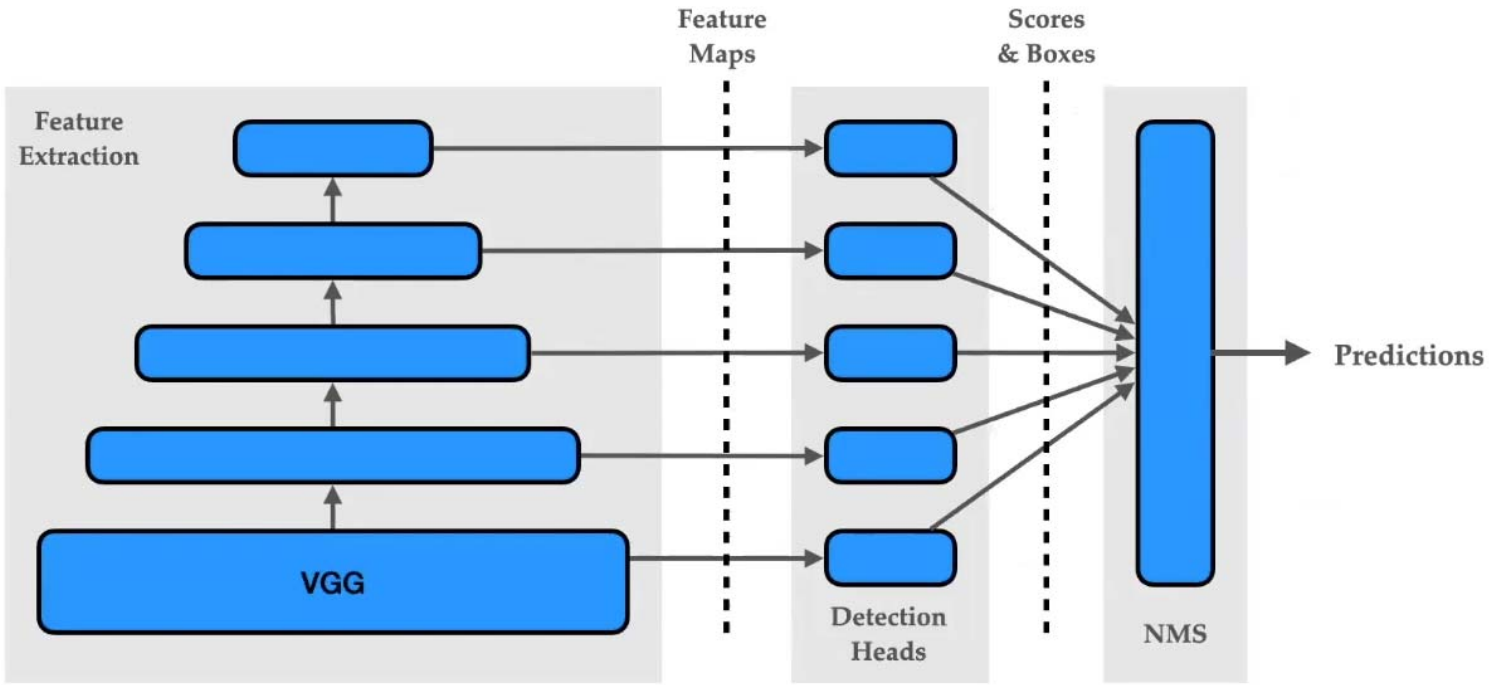
1. Select next highest-scoring box
2. Eliminate lower-scoring boxes (IoU > threshold)





L 10.5 Single Stage Object Detection

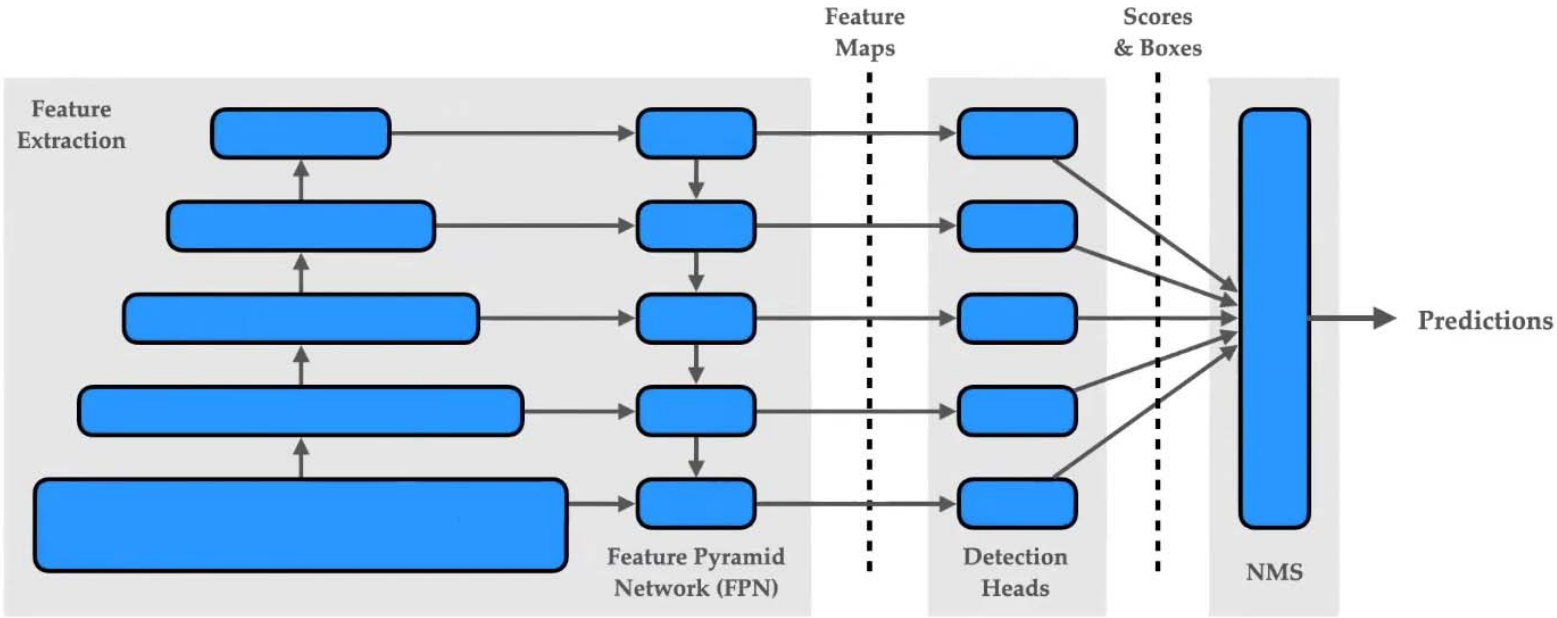
Single-Shot MultiBox Detector (SSD) [Liu2016]



L 10.5 Single Stage Object Detection

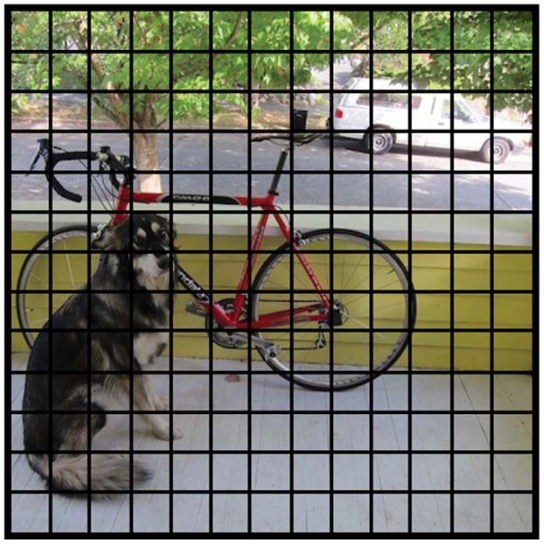
Single-Shot MultiBox Detector (SSD) [Liu2016]

- Possible Modification: Feature Pyramid Network (FPN)

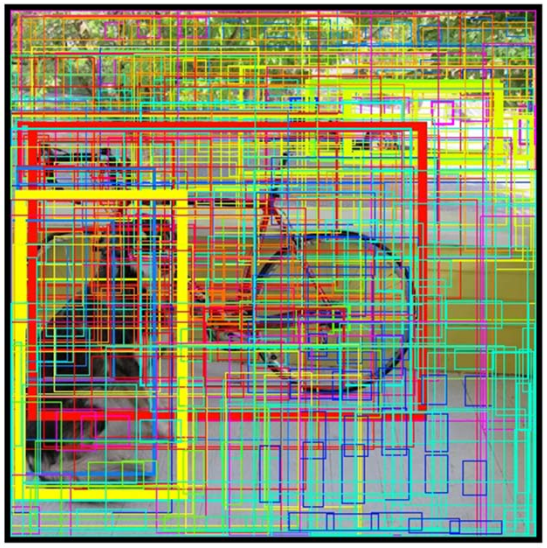


L 10.5 Single Stage Object Detection

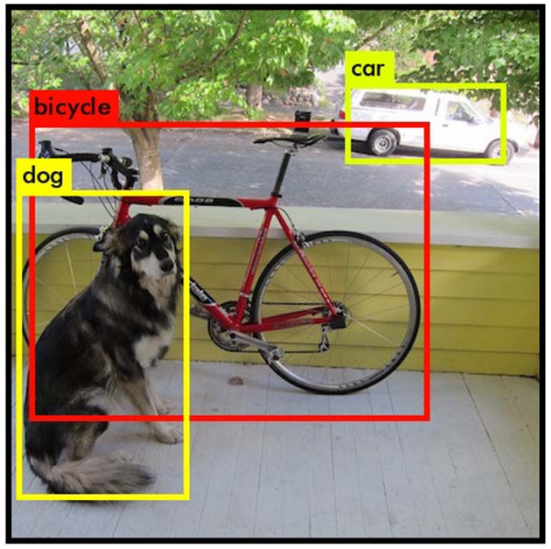
You Only Look Once (YOLO) [Redmon2016]



divide the image into a grid of 13 by 13 cells



Predict bounding boxes with class probabilities

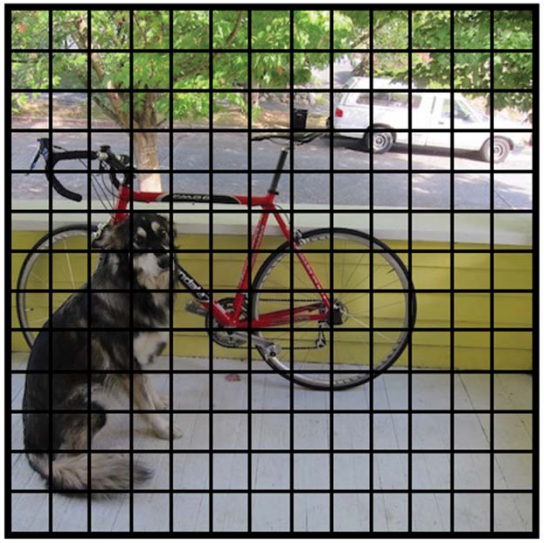


filter the final predictions

L 10.5 Single Stage Object Detection

You Only Look Once (YOLO) [Redmon2016]

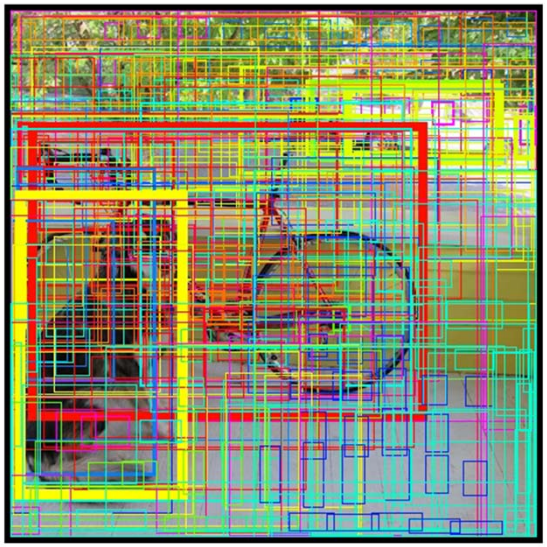
- YOLO divides up the image into a grid of 13 by 13 cells
- Each of these cellsis responsible for predicting 5 bounding boxes
- also outputs a confidence value indicating how certain it is that the predicted bounding box actually encloses an object
  - this score says nothing about what kind of object is in the box, but only whether the shape of the box is good



L 10.5 Single Stage Object Detection

You Only Look Once (YOLO) [Redmon2016]

- For each bounding box, the cell also predicts a class
- This works just like a classifier: it gives a probability distribution over all the possible classes
- The confidence value for the bounding rectangle and the class prediction are combined into a final value that tells us the probability that this bounding rectangle contains a particular object type



- Since there are 13×13 = 169 grid cells and each cell predicts 5 bounding boxes, we get a total of 845 bounding boxes
- Keep only the boxes whose final score is at least 30%

L 10.5 Single Stage Object Detection

You Only Look Once (YOLO) [Redmon2016]

- Network (V1) architecture

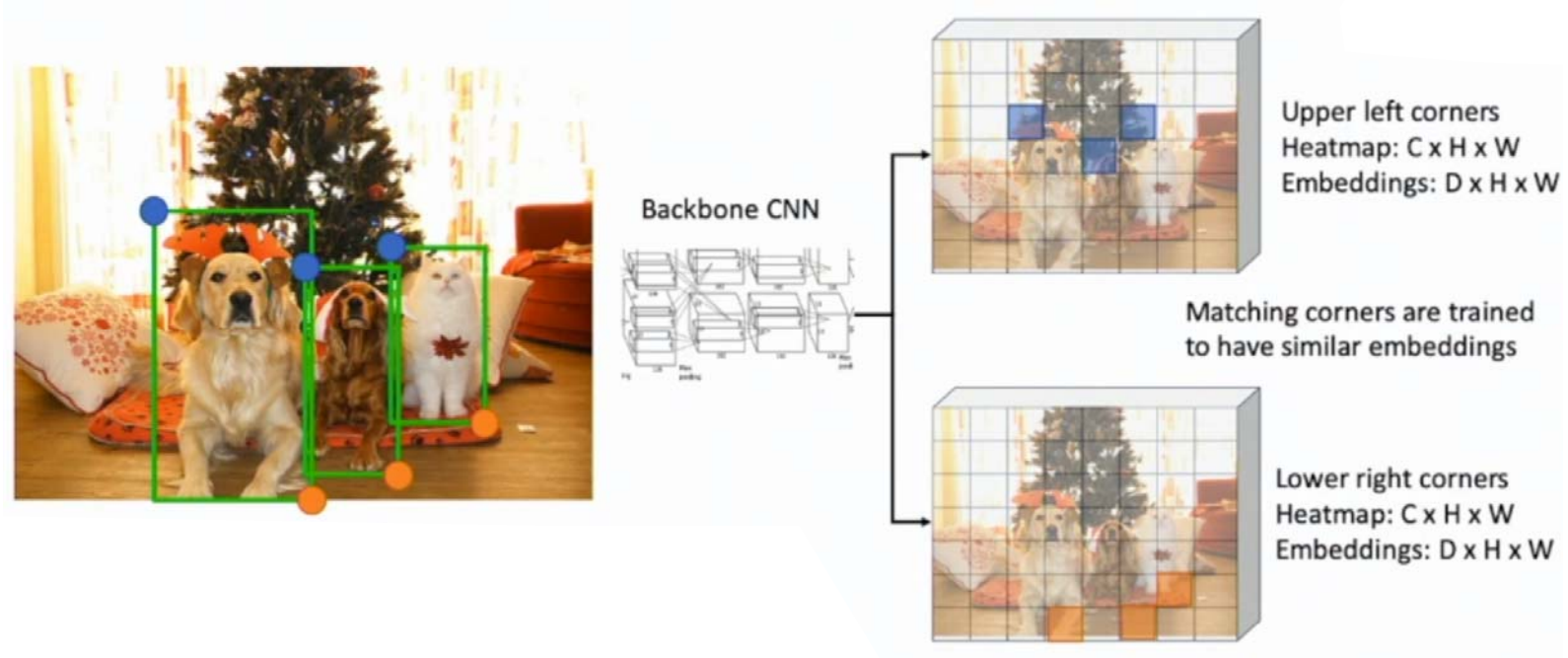
- last convolutional layer has a 1×1 kernel and exists to reduce the data to the shape 13×13×125
  - 13×13 corresponds to the grid size
- 125 channels for every grid cell, each grid cell predicts 5 bounding boxes and a bounding box is described by 25 data elements:
  - x, y, width, height for the bounding box's rectangle
  - the confidence score
  - the probability distribution over the classes

Layer	kernel	stride	output shape
Input			(416, 416, 3)
Convolution	3×3	1	(416, 416, 16)
MaxPooling	2×2	2	(208, 208, 16)
Convolution	3×3	1	(208, 208, 32)
MaxPooling	2×2	2	(104, 104, 32)
Convolution	3×3	1	(104, 104, 64)
MaxPooling	2×2	2	(52, 52, 64)
Convolution	3×3	1	(52, 52, 128)
MaxPooling	2×2	2	(26, 26, 128)
Convolution	3×3	1	(26, 26, 256)
MaxPooling	2×2	2	(13, 13, 256)
Convolution	3×3	1	(13, 13, 512)
MaxPooling	2×2	1	(13, 13, 512)
Convolution	3×3	1	(13, 13, 1024)
Convolution	3×3	1	(13, 13, 1024)
Convolution	1×1	1	(13, 13, 125)

L 10.5 Single Stage Object Detection

Alternative method

CornerNet [Law2018]: Represent Bounding Boxes by pairs of corners



L 10.X Instance Segmentation

Instance Segmentation

- Detect all objects in the image and identify the pixels that belong to each object

Approach:

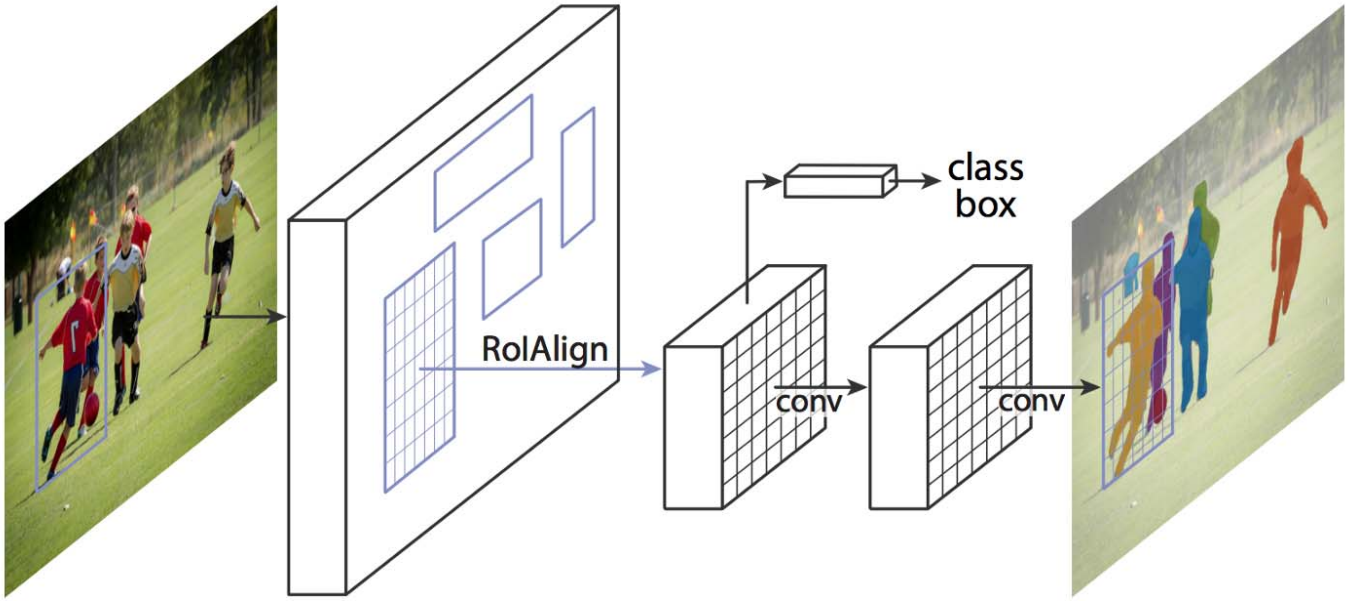
- Perform Object Detection and predict a segmentation mask for each object

Panoptic Segmentation [Kirillov2019, Kirillov2019a]**:**

- for „thing“ categories also separate into instances

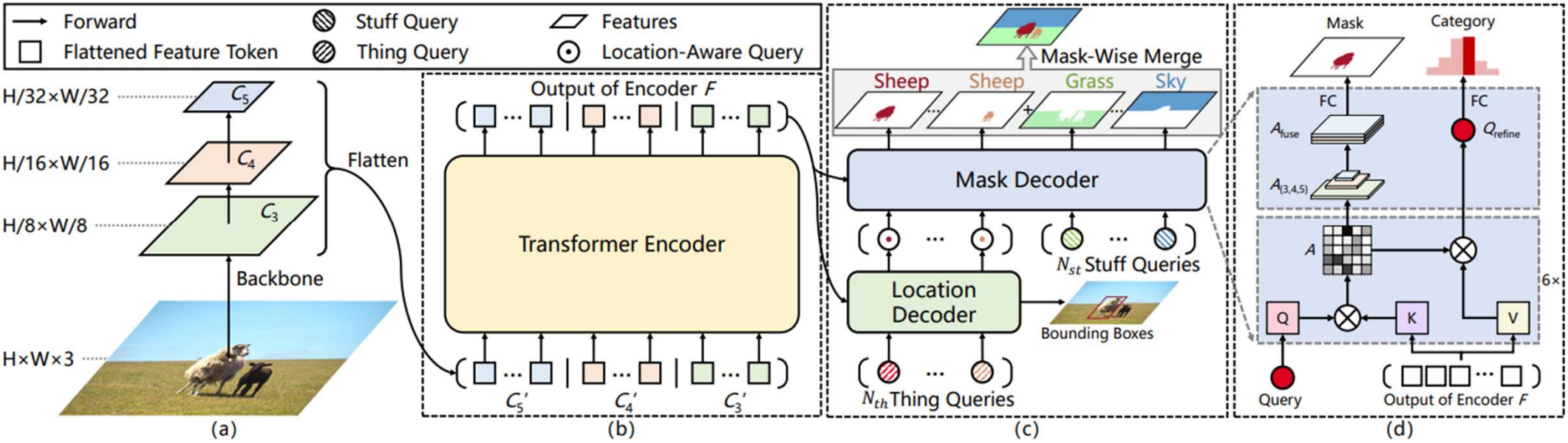
L 10.X Instance Segmentation

Mask R-CNN [He2017]



L 10.X Instance Segmentation

Beyond Mask R-CNN

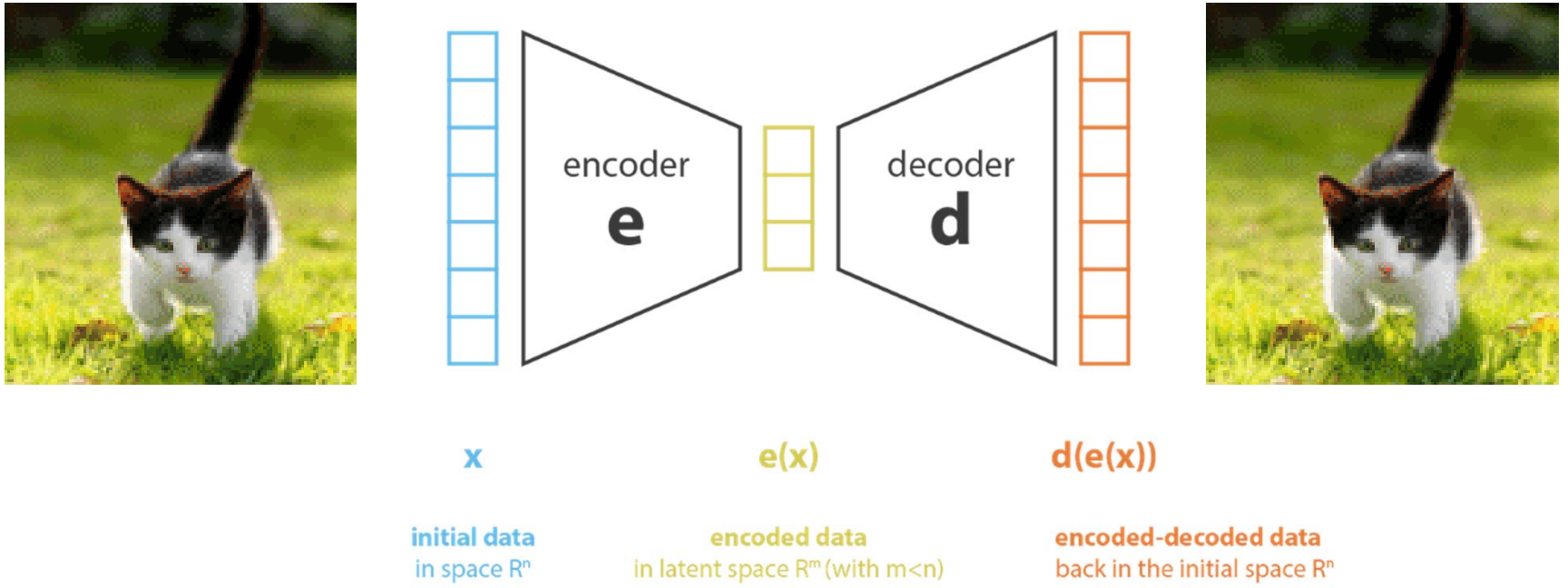


Panoptic SegFormer: Delving Deeper into Panoptic Segmentation with Transformers (NVIDIA) [Li2022]:
customized pipelines for things and stuff. More flexible for various segmentation tasks.

L 10.X Autoencoder

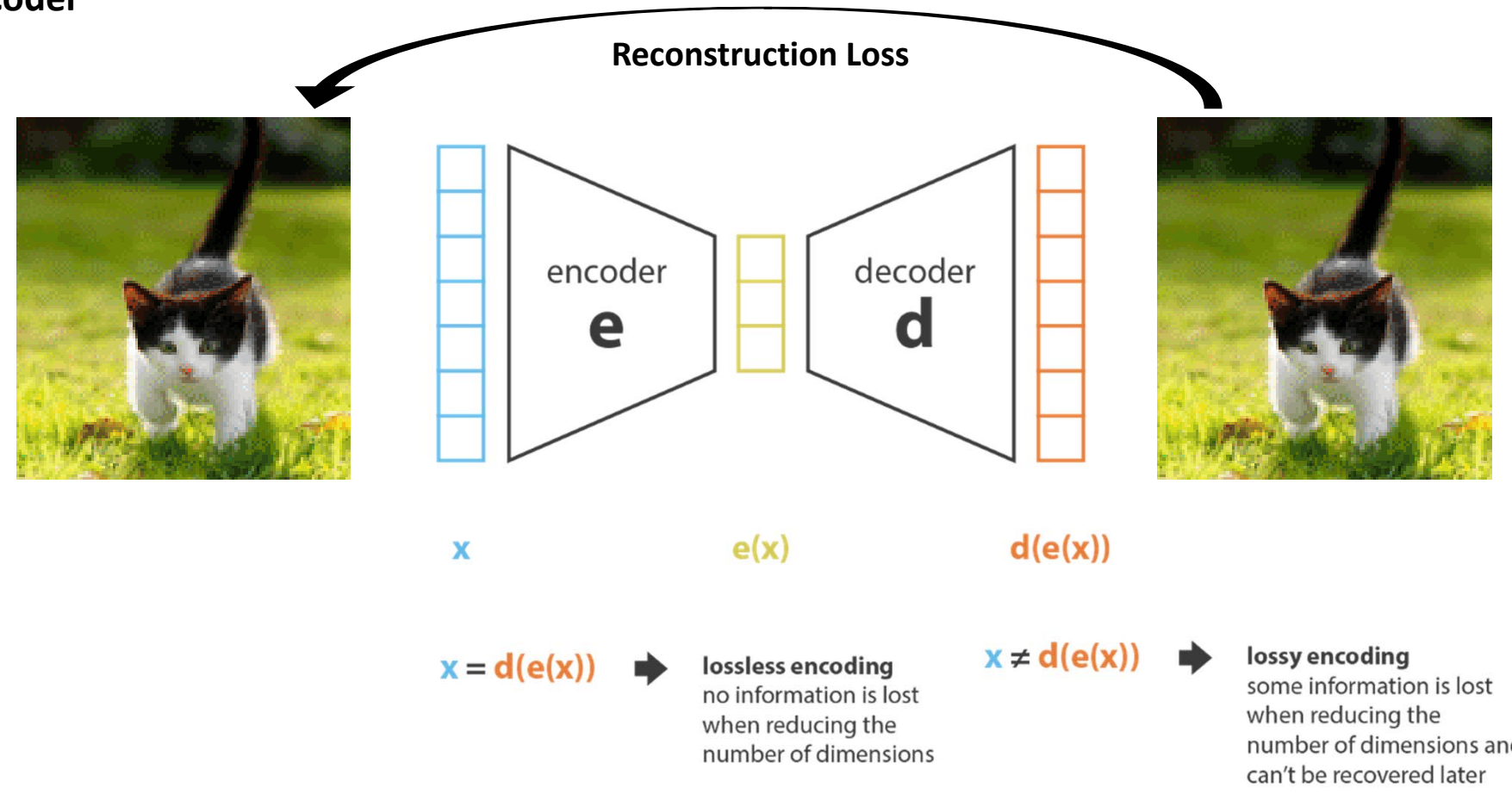
Autoencoder

- Learned compression of very high dimensional data (image or any vector) to a smaller representation
- encoded data in latent space often called bottleneck



L 10.X Autoencoder

Autoencoder



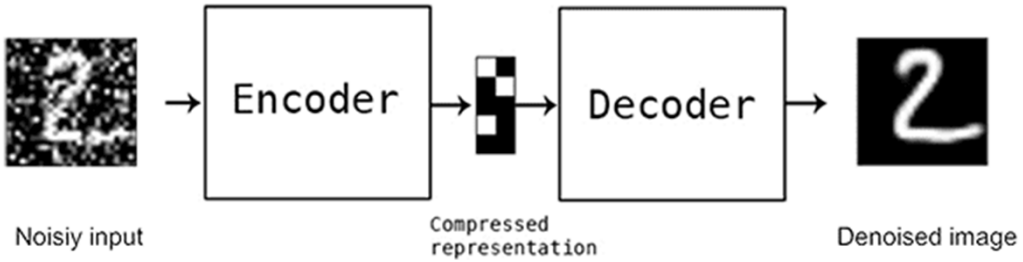
L 10.X Autoencoder

Entropy-based
hardcoded compression

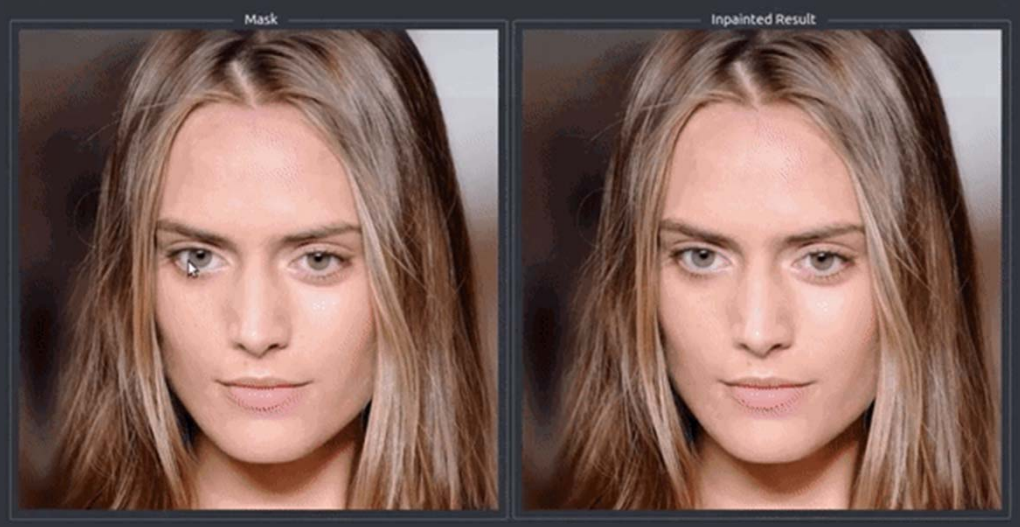
VS

data specific
learned compression

- Besides pure usage for compression:
  - denoising
  - neural inpainting



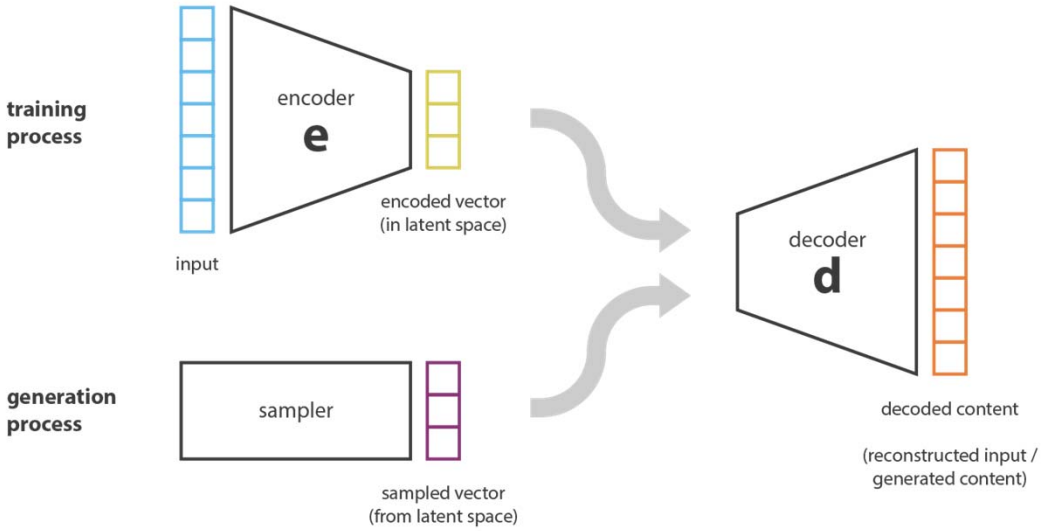
L 10.X Autoencoder



L 10.X Autoencoder

What about generating new data

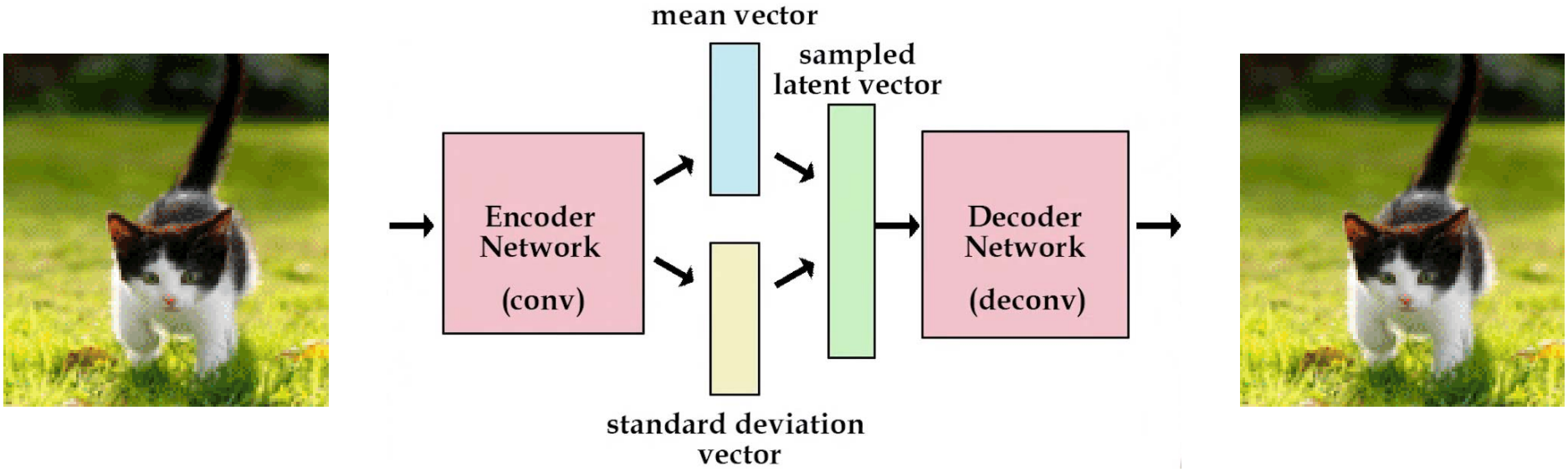
- autoencoder is solely trained to **encode and decode** with as few loss as possible, no matter how the latent space is organized
- the latent space can be extremely irregular
  - close points in latent space can give very different decoded data
  - some point of the latent space can give meaningless content once decoded
- not a generative process that simply samples a point from latent space and sends it through the decoder to get new data



L 10.X Variational Autoencoder

Intuition

- instead of mapping the input to a fixed vector (latent space), we want to map it to a distribution



$$L(\Theta, \phi; x, z) = \underbrace{\mathbb{E}_{q\phi(z|x)}[log\ p_\Theta(x|z)]}_{\text{Reconstruction Loss}} - \underbrace{D_{KL}(q_\Theta(z|x)||p(z))}_{\text{Distance to } N(0,1)}$$

L 10.X Variational Autoencoder

Intuition

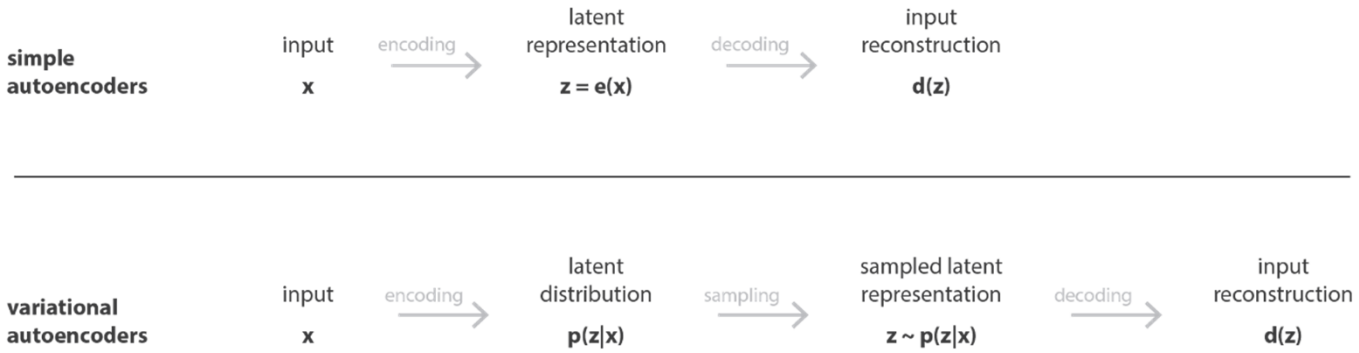
$$L(\Theta, \phi; x, z) = \underbrace{\mathbb{E}_{q\phi(z|x)}[log\ p_\Theta(x|z)]}_{\text{Reconstruction Loss}} - \underbrace{D_{KL}(q_\Theta(z|x)||p(z))}_{\text{Distance to } N(0,1)}$$

- enforcing distributions to be close to a standard normal distribution:
- covariance matrices to be close to the identity ➔ preventing punctual distributions
- mean close to 0 ➔ preventing encoded distributions to be too far apart from each others

L 10.X Variational Autoencoder

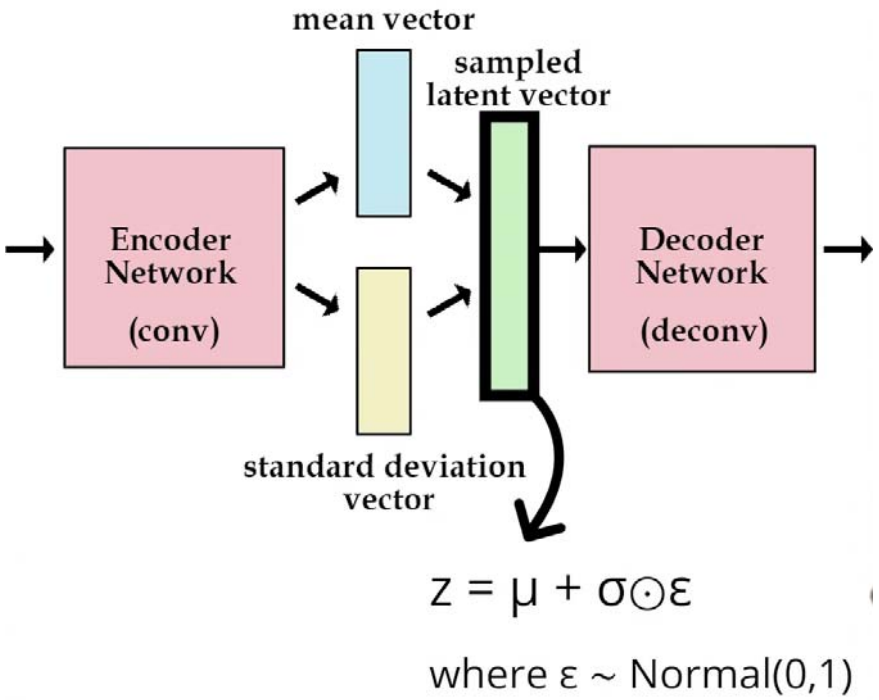
A variational autoencoder can be defined as an autoencoder whose training is regulated to avoid overfitting and to ensure that the latent space has properties good enough to allow a generative process.

- Instead of encoding an input as a single point, it is encoded as a distribution over latent space
- the distributions returned by the encoder are enforced to be close to a standard normal distribution
- ensure both a local and global regularisation of the latent space
  - local ➔ variance control
  - global ➔ mean control



L 10.X Variational Autoencoder

- sampling operation poses a problem for backpropagation
- gradient is not defined for sampling



Recap Backpropagation / Gradient Descent:

Gradient Descent:

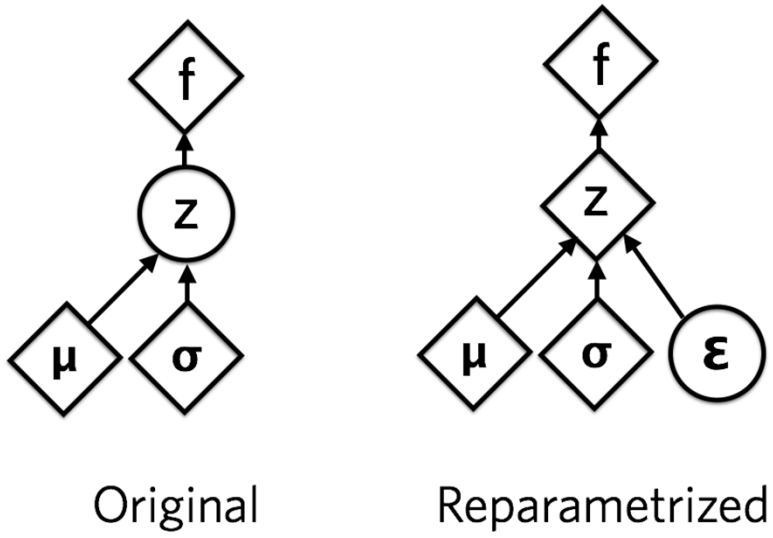
$$\Theta \rightarrow \Theta - \eta \nabla_{\Theta} L(Y, f(\Theta, X))$$

L 10.X Variational Autoencoder

Reparameterization Trick

- Instead from sampling from μ and σ directly the stochastic part is introduced separately
- ϵ is always standard gaussian ($N(0,1)$)

$$z = \mu + \sigma \cdot \epsilon$$



L 10.X Disentangled Variational Autoencoder

Disentanglement ensures:

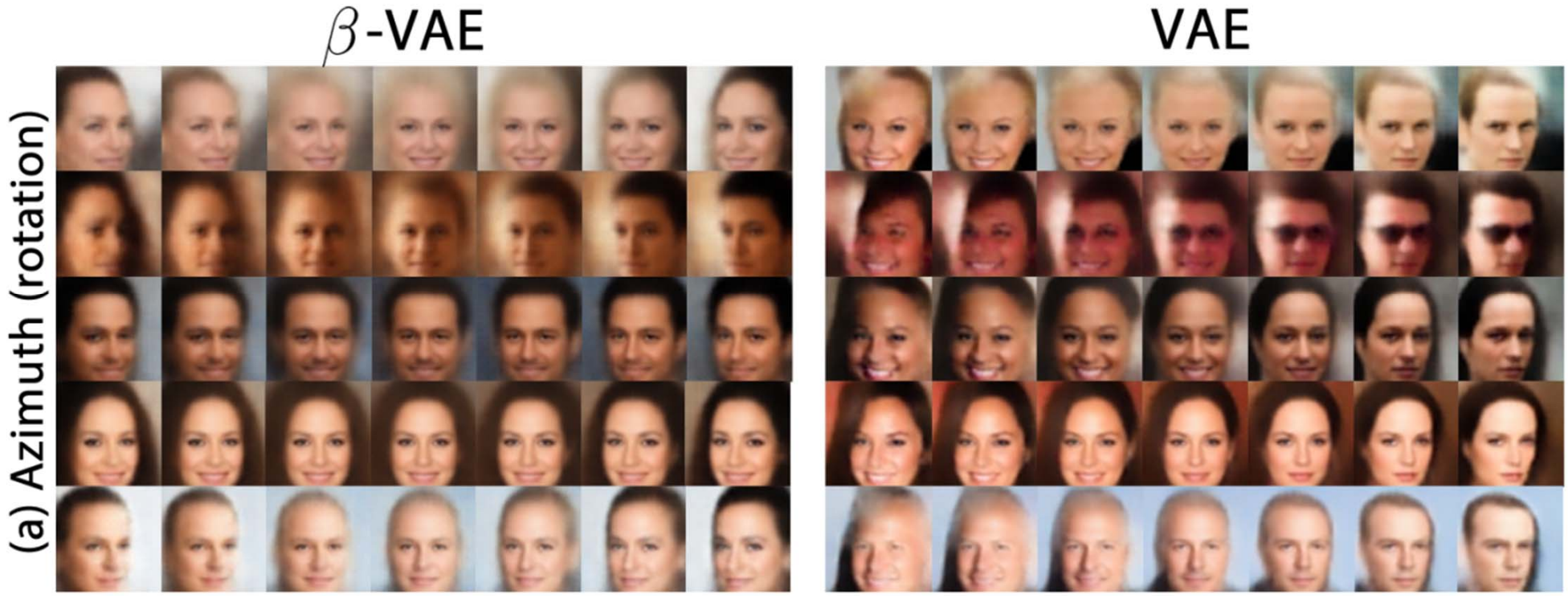
- neurons in the latent distributions are uncorrelated
- neurons try to learn different aspects of the input data

$$L(\Theta, \phi; x, z) = \underbrace{\mathbb{E}_{q\phi(z|x)}[log\, p_\Theta(x|z)]}_{\text{Reconstruction Loss}} - \beta \underbrace{D_{KL}(q_\Theta(z|x)||p(z))}_{\text{Distance to } N(0,1)}$$

- β weights how much the KL divergence is present in the Loss function
- increasing β forces the Autoencoder to map the information onto only a few latent variables

L 10.X Disentangled Variational Autoencoder

Quantitative sampling results from VAE VS. Disentangled VAE



www.hs-kempten.de/ifm