Computer Vision

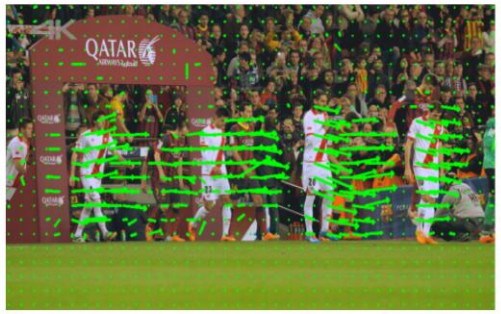
Lecture 8: Object Detection

Oliver Bimber

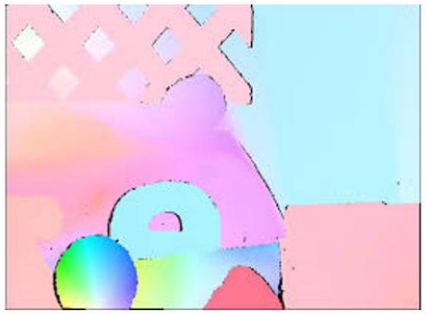
JOHANNES KEPLER
UNIVERSITY LINZ

Last Week: Optical Flow

What is Optical Flow?



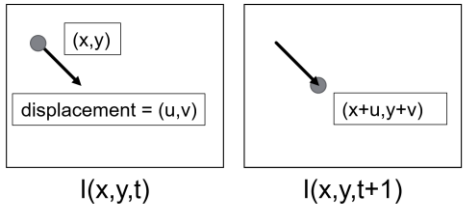
Sparse Optical Flow (Flow Vector per Region)



Dense Optical Flow (Flow Vector per Pixel)

Motion Vector of Pixel in Time Series (two consecutive Video Frames at Times t and t+1)

The Optical Flow Equation



Brightness Constancy:

$$I(x+u, y+v, t+1) = I(x, y, t)$$

$$0 \approx I(x+u, y+v, t+1) - I(x, y, t)$$

Taylor Expansion:

$$\approx I(x, y, t+1) + I_x u + I_y v - I(x, y, t)$$

$$= I(x, y, t+1) - I(x, y, t) + I_x u + I_y v$$

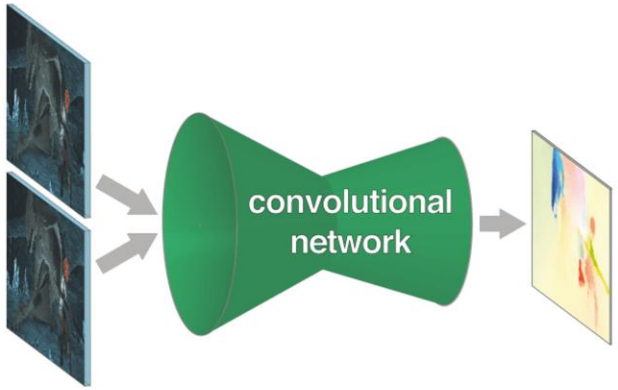
Optical Flow Equation:

$$0 = I_t + I_x u + I_y v$$

$$= I_t + \nabla I \cdot [u, v]$$

I_t, I_x, I_y are partial derivatives of image intensity (gradients) in *t,x,y*

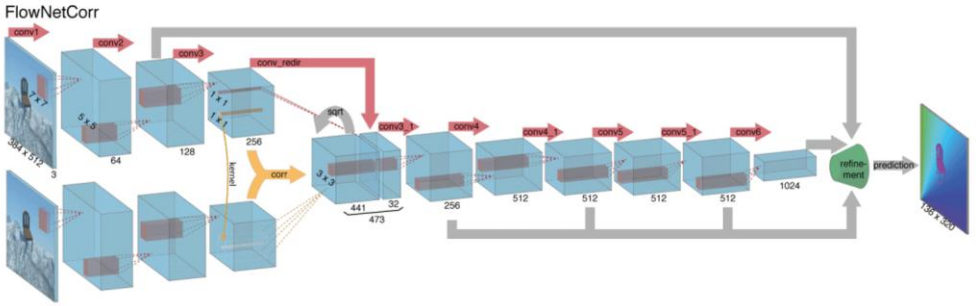
Optical Flow and Machine Learning



Encoder+Decoder Architectures (e.g. U-Nets)

Example: FlowNetCorr (Correlation)

https://lmb.informatik.uni-freiburg.de/Publications/2015/DFIB15/flownet.pdf



Input: 2 Tensors of individual RGB Images (Feature Maps are computed later → Correlation Layer)

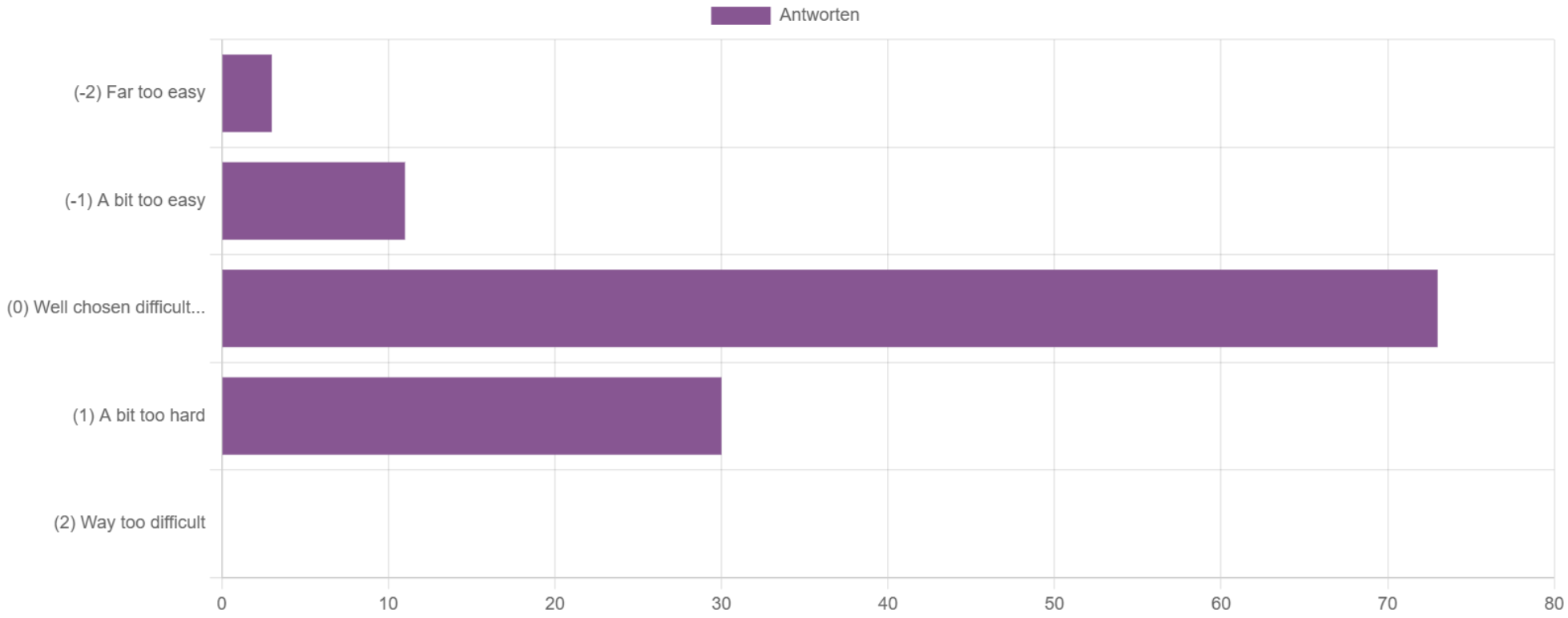
JOHANNES KEPLER UNIVERSITY LINZ

Course Overview

CW	Topic	Date	Place	Lab
41	Introduction and Course Overview	07.10.2025	Zoom	Lab 1
42	Capturing Digital Images	14.10.2025	Zoom	Lab 2
43	Digital Image Processing	21.10.2025	Zoom	Assignment 1
44	Machine Learning	28.10.2025	Zoom	
45	Feature Extraction	04.11.2025	Zoom	Open Lab 1
46	Segmentation	11.11.2025	Zoom	Assignment 2
47	Optical Flow	18.11.2025	Zoom	Open Lab 2
48	Object Detection	25.11.2025	Zoom	Assignment 3
49	Multi-View Geometry	02.12.2025	Zoom	Open Lab 3
50	3D Vision	09.12.2025	Zoom	Assignment 4
3	Trends in Computer Vision	13.01.2026	Zoom	
4	Q&A	20.01.2026	Zoom	Open Lab 4
5	Exam	27.01.2026	HS1 (Linz), S1/S3 (Vienna), S5 (Bregenz)	
9	Retry Exam	24.02.2026	tba	

Assignment 1: Image Processing Basics

How do you rate the Difficulty of the Assignment?



Average Time spent: 7.38 h

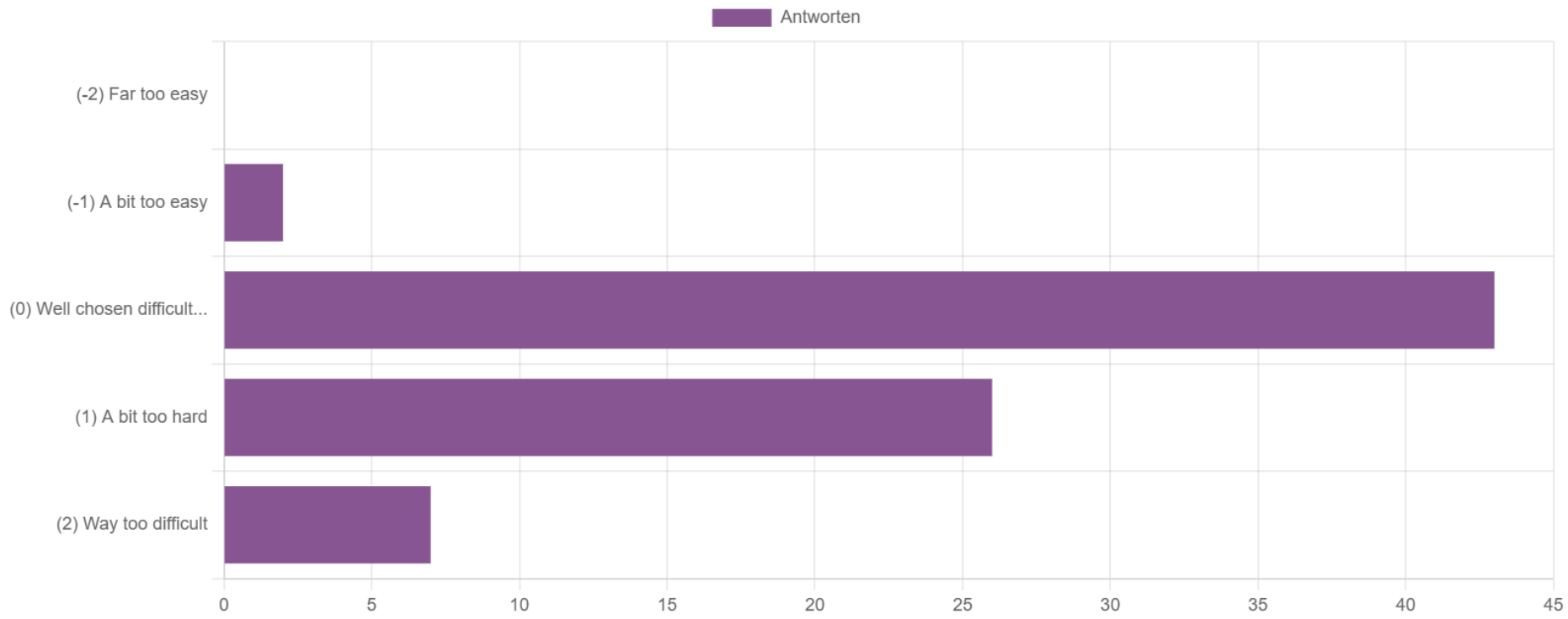
JOHANNES KEPLER UNIVERSITY LINZ

297 total submitted Assignments

117 gave Feedback – thank you!

Assignment 2: Feature Extraction and Segmentation

How do you rate the Difficulty of the Assignment?



Average Time spent: 10.45 h

JOHANNES KEPLER UNIVERSITY LINZ

285 total submitted Assignments

78 gave Feedback – thank you!

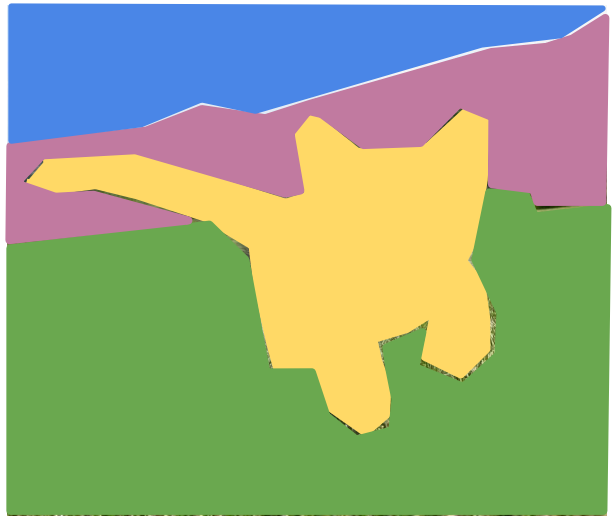
Classification vs. Segmentation

Classification

CAT

No spatial extent

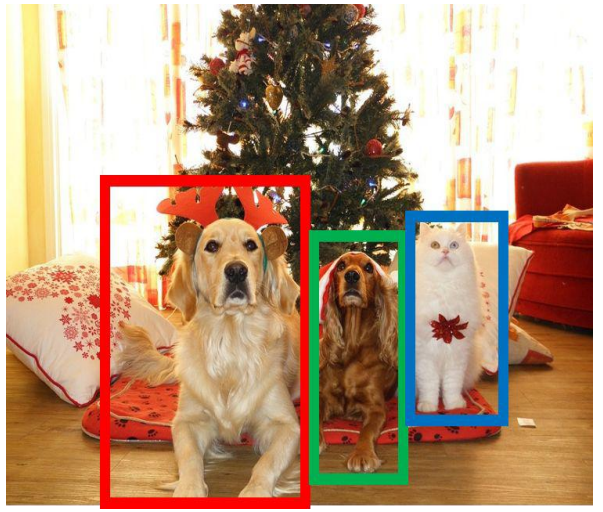
Semantic Segmentation



GRASS, CAT, TREE, SKY

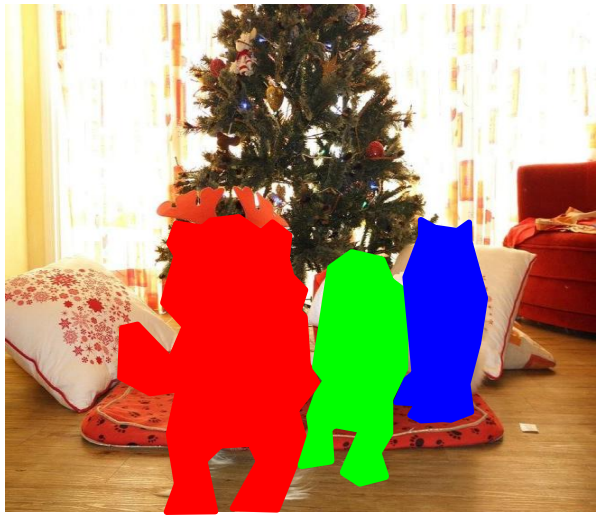
No objects, just pixels

Object Detection



DOG, DOG, CAT

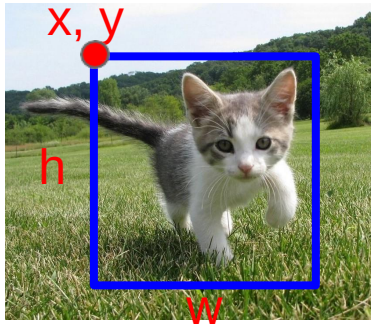
Instance Segmentation



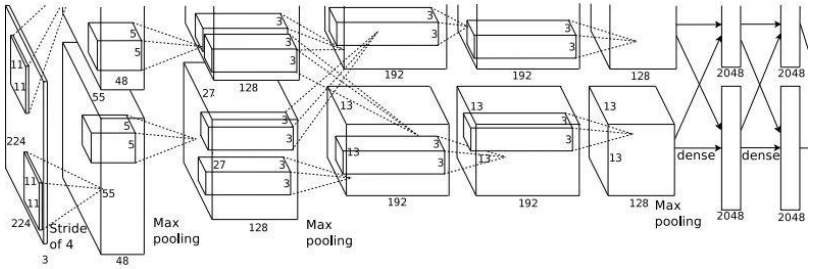
DOG, DOG, CAT

Multiple Object

Object Detection: Single Object



This image is CC0 public domain



Vector: 4096

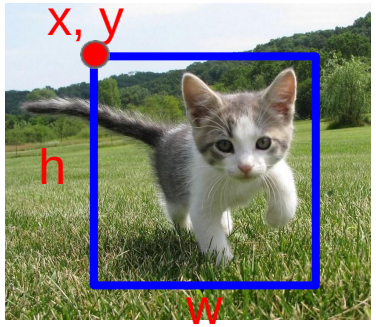
Fully Connected: 4096 to 1000

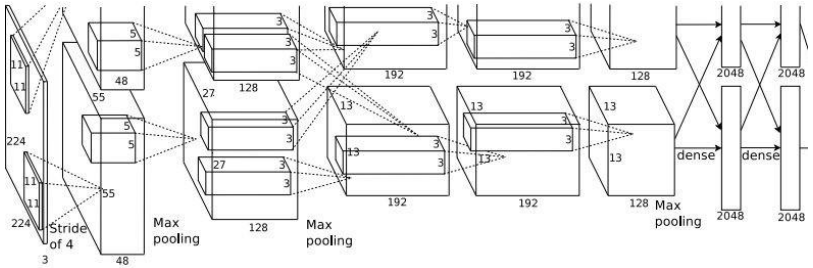
Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Fully Connected: 4096 to 4

Box Coordinates
(x, y, w, h)

Object Detection: Single Object





This image is CC0 public domain

Correct label:
Cat

Fully Connected:
4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Vector:
4096

Fully Connected:
4096 to 4

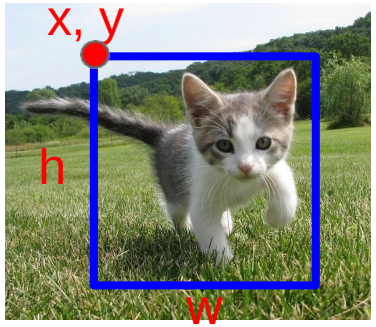
Box Coordinates
(x, y, w, h)

L2 Loss

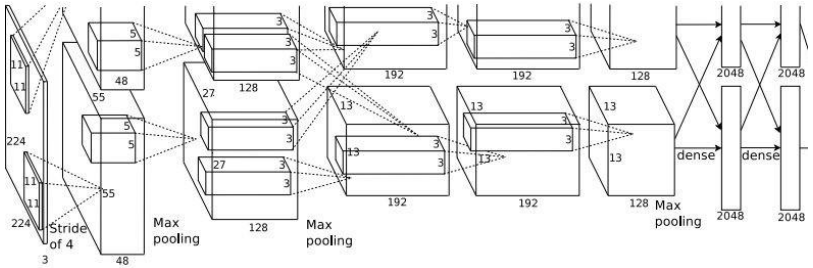
Correct box:
(x', y', w', h')

Treat localization as a Regression Problem!

Object Detection: Single Object



This image is CC0 public domain



Vector: 4096

Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Correct label: Cat

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

L2 Loss

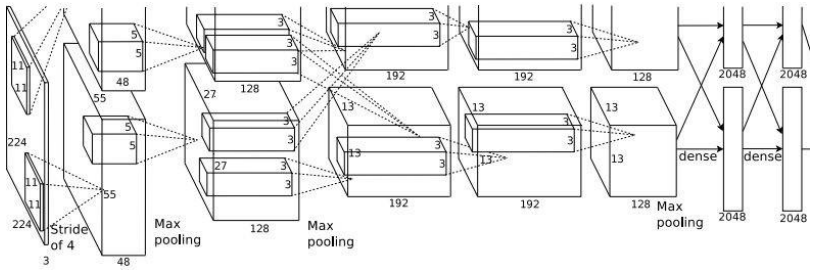
Correct box: (x', y', w', h')

Multitask Loss

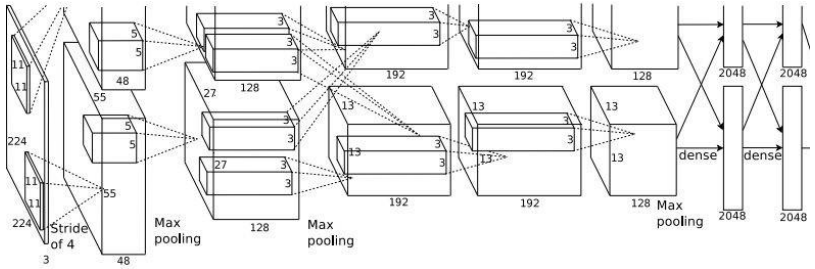
+ → **Loss**

Treat localization as a Regression Problem!

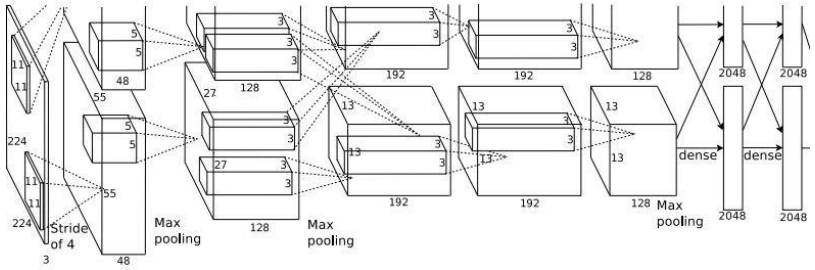
Object Detection: Multiple Objects



CAT: (x, y, w, h)



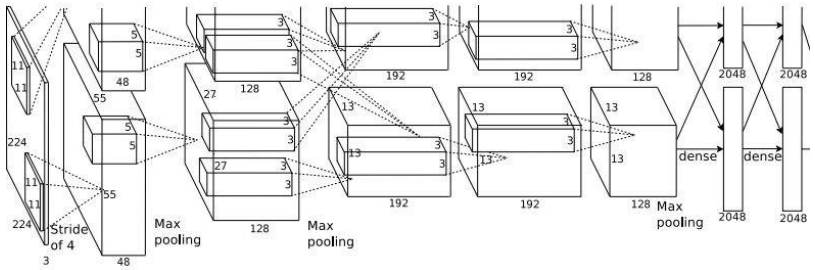
DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)



DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

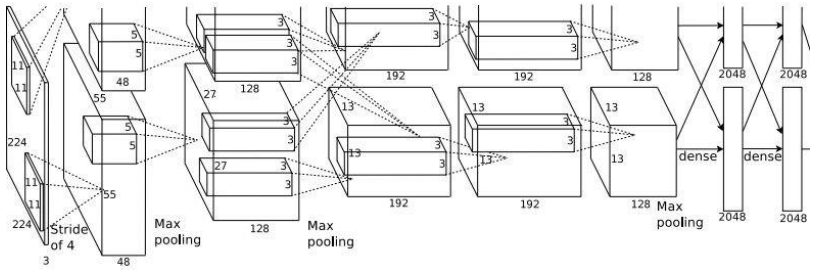
Object Detection: Multiple Objects

Each Image needs a different Number of Outputs!



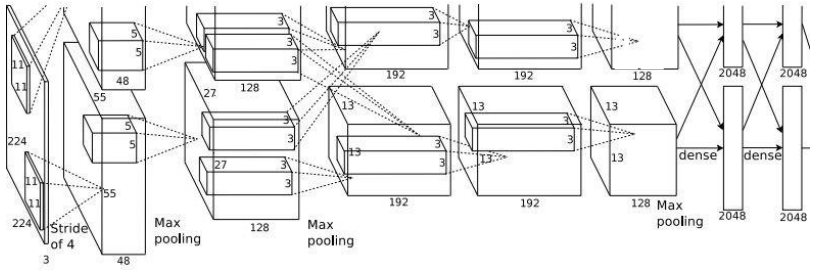
CAT: (x, y, w, h)

4 numbers



DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

12 numbers

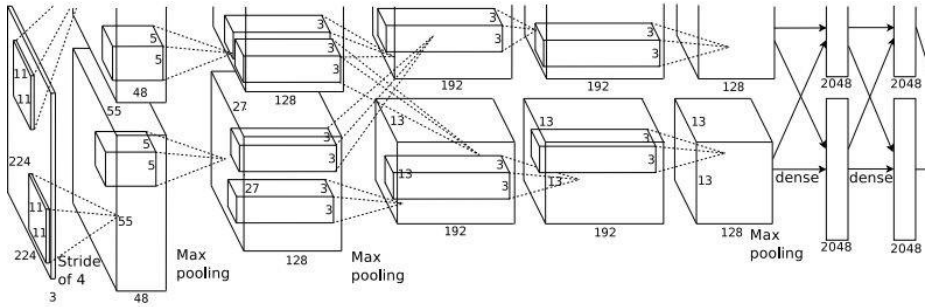


DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
….

Many numbers!

Object Detection: Multiple Objects

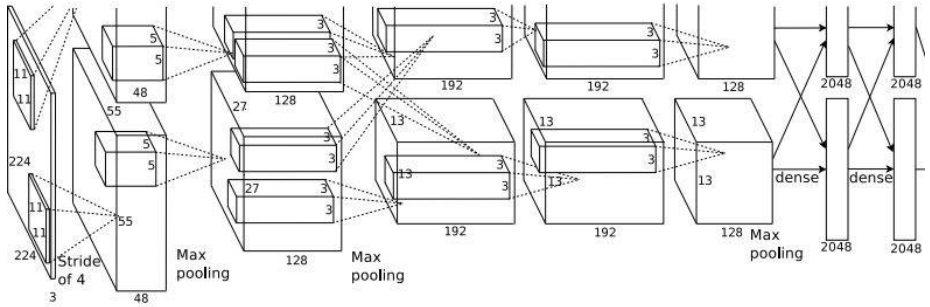
Apply a CNN to many different Crops of the Image, CNN classifies each Crop as Object or Background



Dog? NO
Cat? NO
Background? YES

Object Detection: Multiple Objects

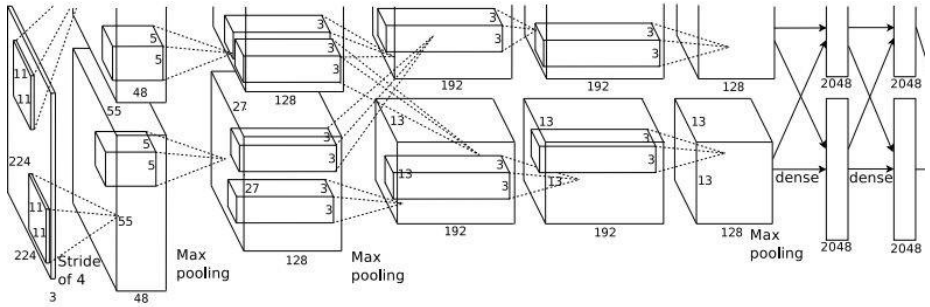
Apply a CNN to many different Crops of the Image, CNN classifies each Crop as Object or Background



Dog? YES
Cat? NO
Background? NO

Object Detection: Multiple Objects

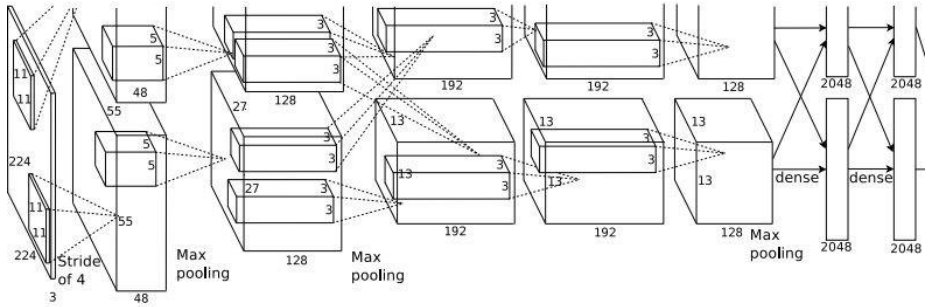
Apply a CNN to many different Crops of the Image, CNN classifies each Crop as Object or Background



Dog? YES
Cat? NO
Background? NO

Object Detection: Multiple Objects

Apply a CNN to many different Crops of the Image, CNN classifies each Crop as Object or Background

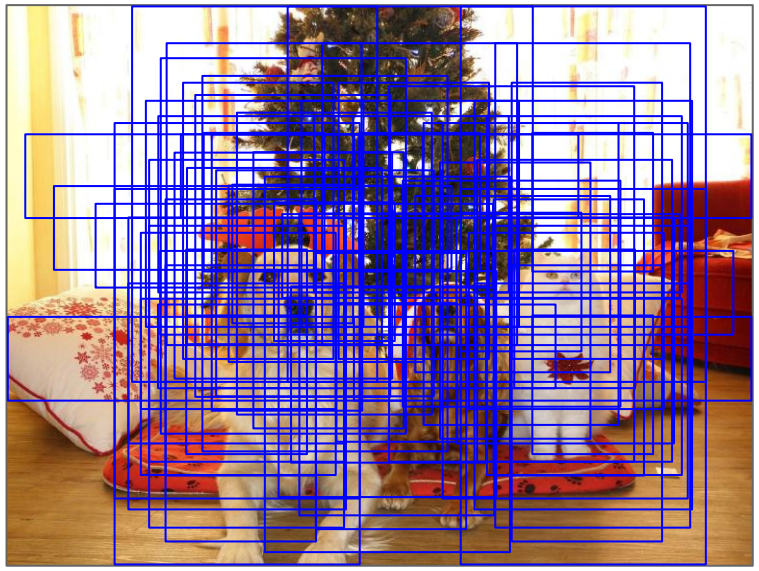


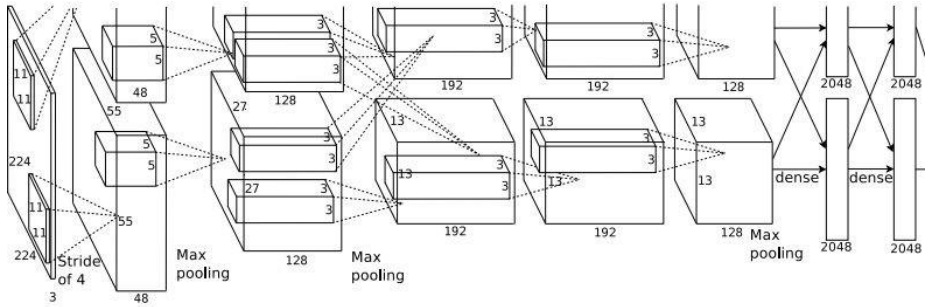
Dog? NO
Cat? YES
Background? NO

Q: What's the Problem with this Approach?

Object Detection: Multiple Objects

Apply a CNN to many different Crops of the Image, CNN classifies each Crop as Object or Background



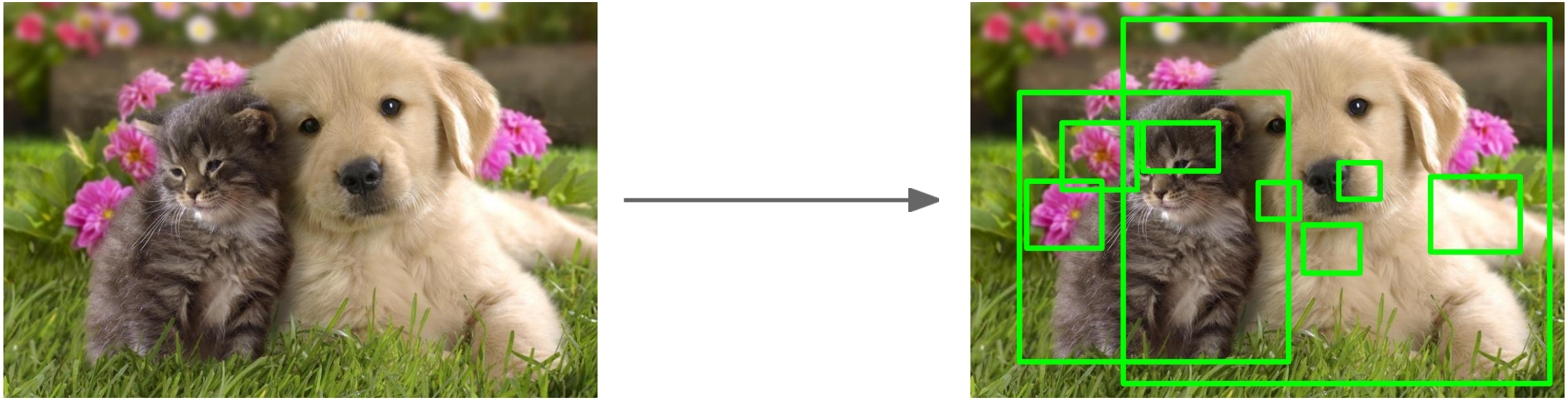


Dog? NO
Cat? YES
Background? NO

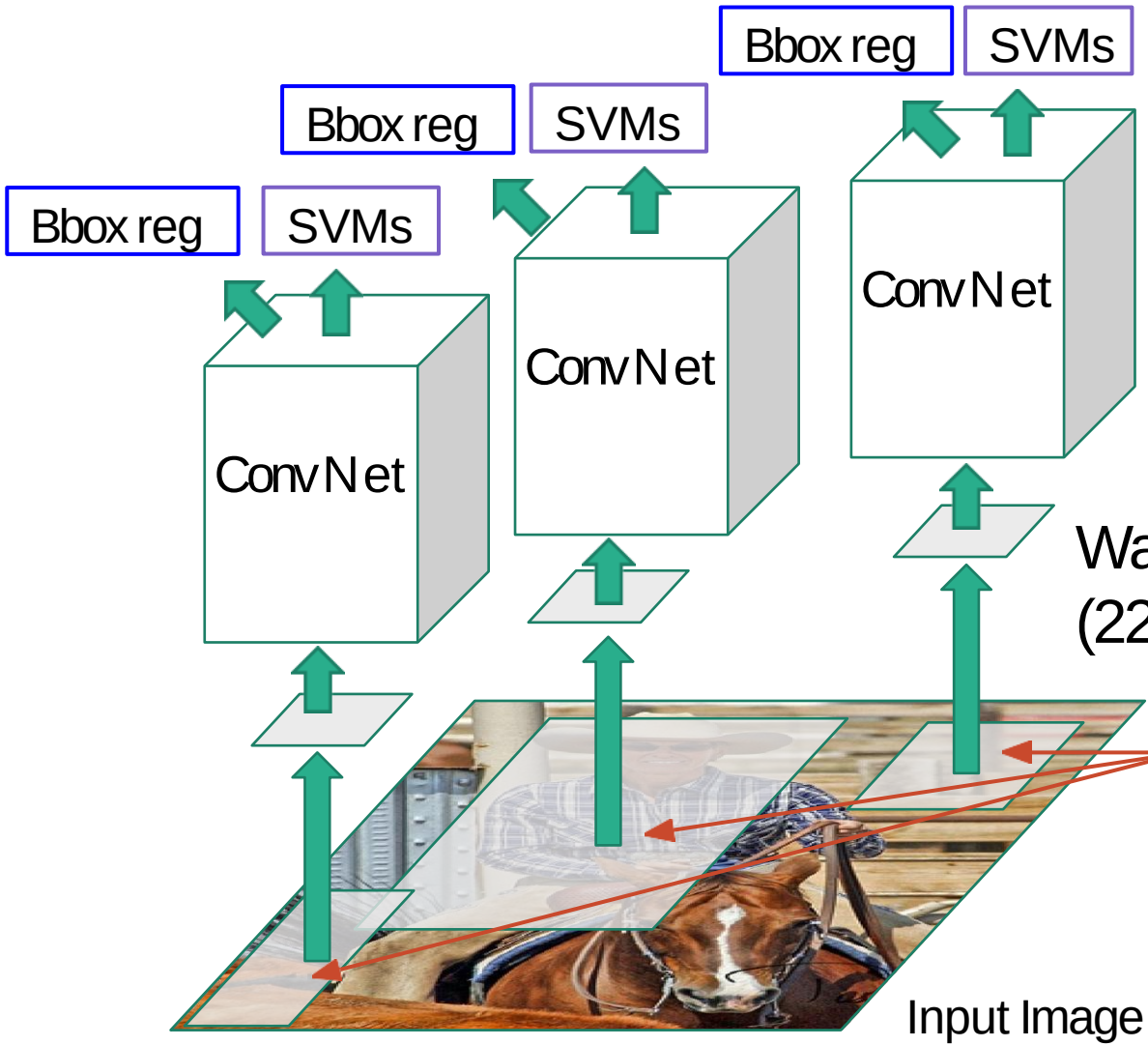
Problem: Need to apply CNN to huge Number of Locations, Scales, and Aspect Ratios, very computationally expensive!

Region Proposal: Selective Search

- Find “blobby” Image Regions that are likely to contain Objects
- Relatively fast to run; e.g. Selective Search gives 2000 Region Proposals in a few Seconds on CPU



Regional-Based (R)-CNN



Classify Regions with SVMs

Forward each Region through ConvNet

Problem: Very slow! Need to do ~2k independent forward Passes for each Image!

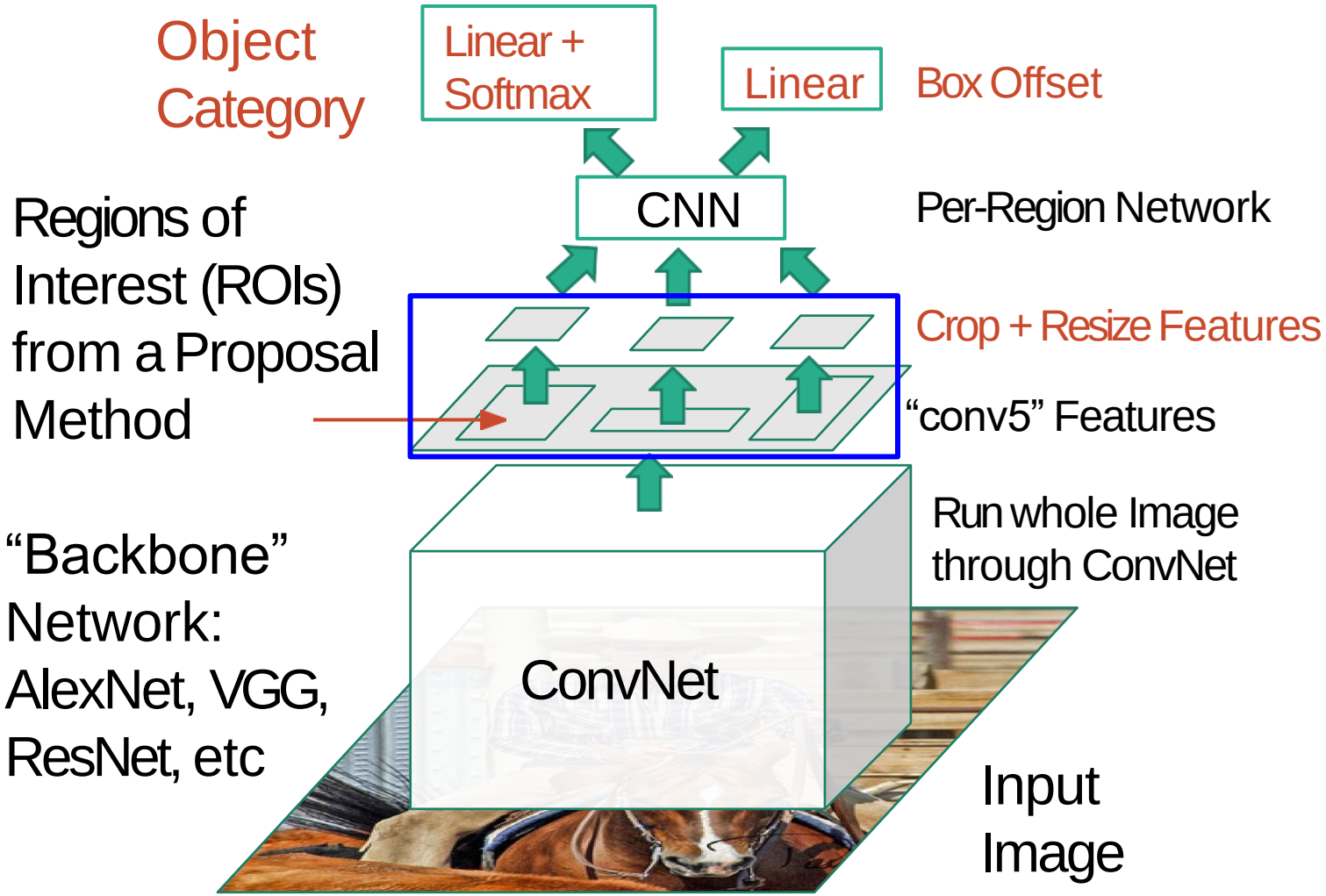
Warped Image Regions (224x224 pixels)

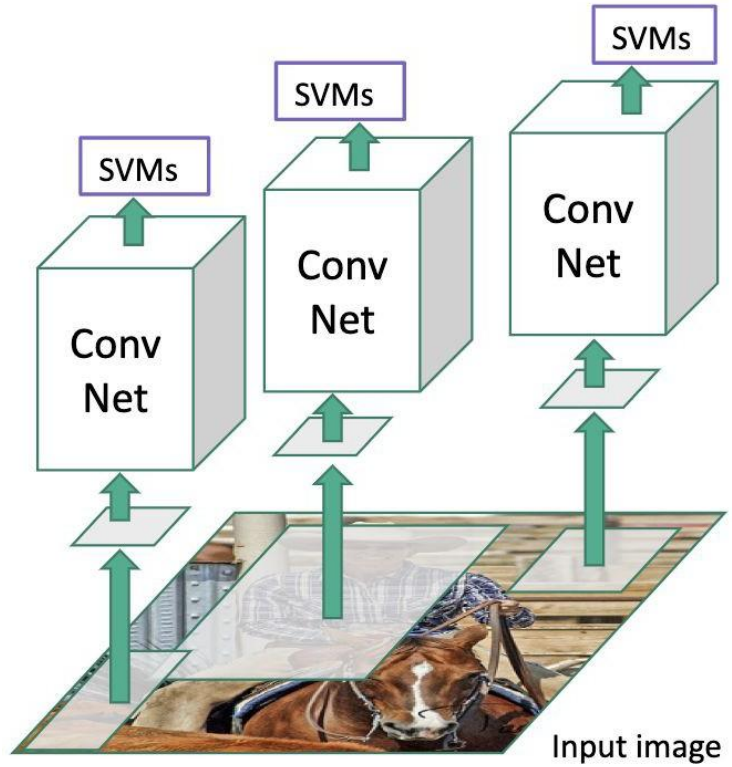
Regions of Interest (RoI) from a Proposal Method (~2k)

Input Image

Predict "Corrections" to the RoI: 4 Numbers: (dx, dy, dw, dh)

Fast R-CNN


Object Category
Linear + Softmax
Linear
Box Offset
CNN
Per-Region Network
Regions of Interest (ROIs) from a Proposal Method
Crop + Resize Features
"conv5" Features
Run whole Image through ConvNet
"Backbone" Network: AlexNet, VGG, ResNet, etc
ConvNet
Input Image

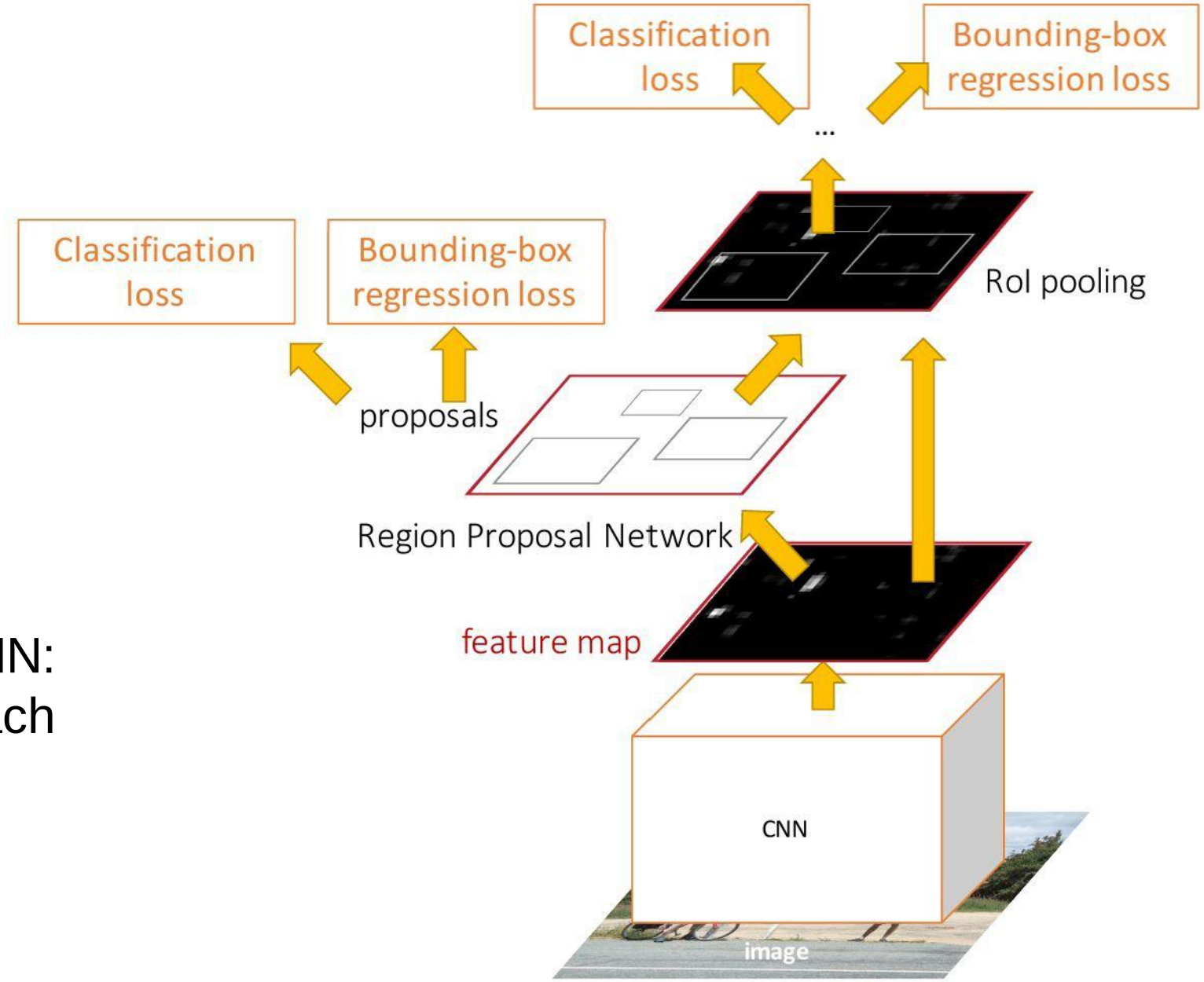

"Slow" R-CNN
SVMs
SVMs
SVMs
Conv Net
Conv Net
Conv Net
Input image

Faster R-CNN

Make CNN do Proposals!

Insert **Region Proposal Network (RPN)** to predict Proposals from Features

Otherwise same as Fast R-CNN: Crop Features for each Proposal, Classify each one



Region Proposal Network

Input Image
(e.g. 3 x 640 x 480)

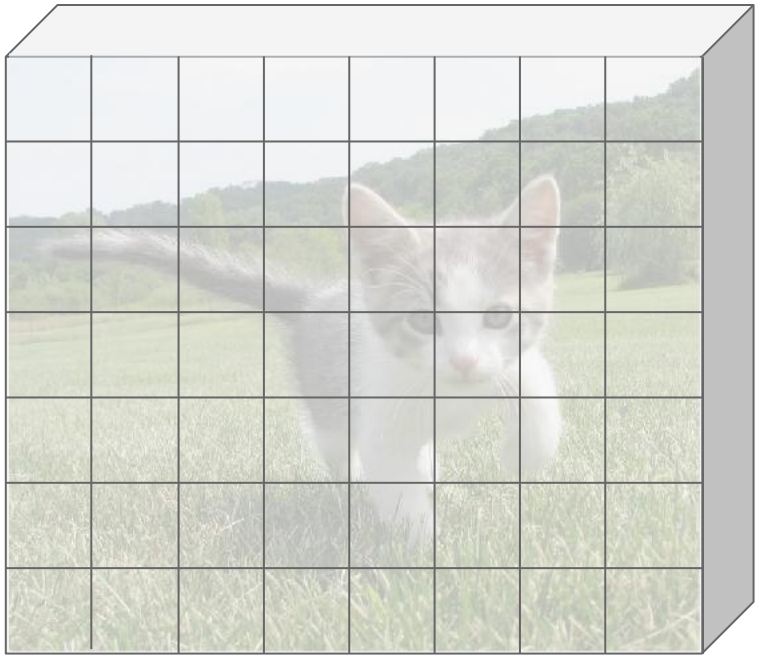


Image features
(e.g. 512 x 20 x 15)

Region Proposal Network

Imagine an **Anchor Box** of fixed Size at each Point in the Feature Map

Input Image
(e.g. 3 x 640 x 480)

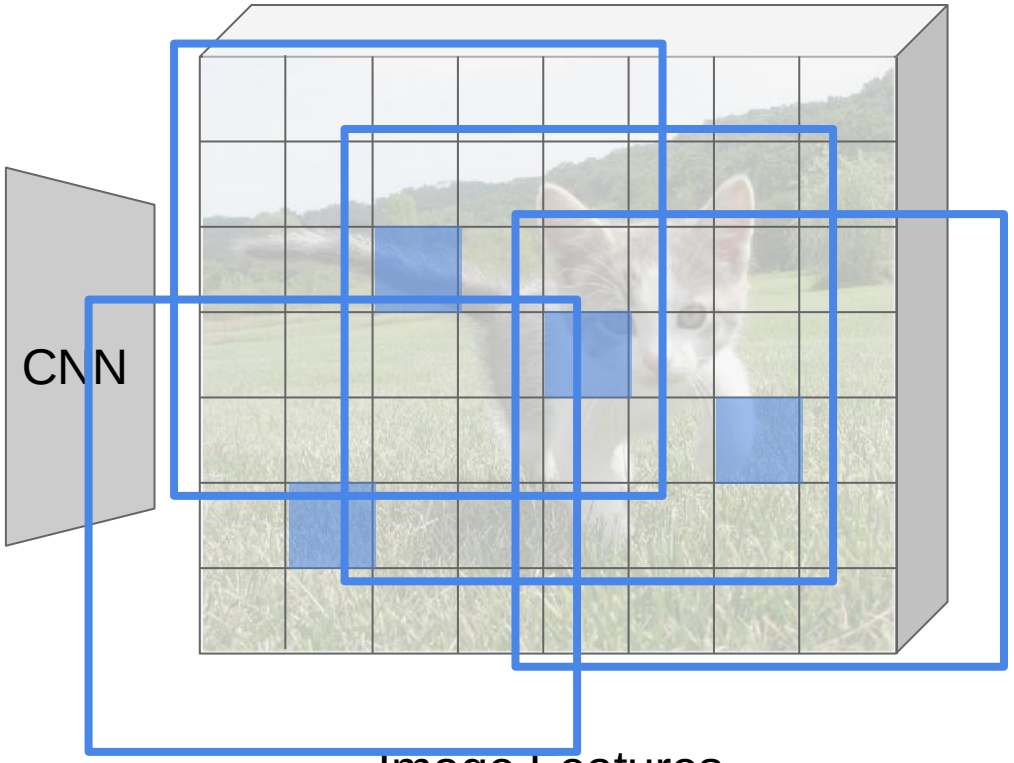


Image Features
(e.g. 512 x 20 x 15)

Region Proposal Network

Imagine an **Anchor Box** of fixed Size at each Point in the Feature Map

Input Image
(e.g. 3 x 640 x 480)

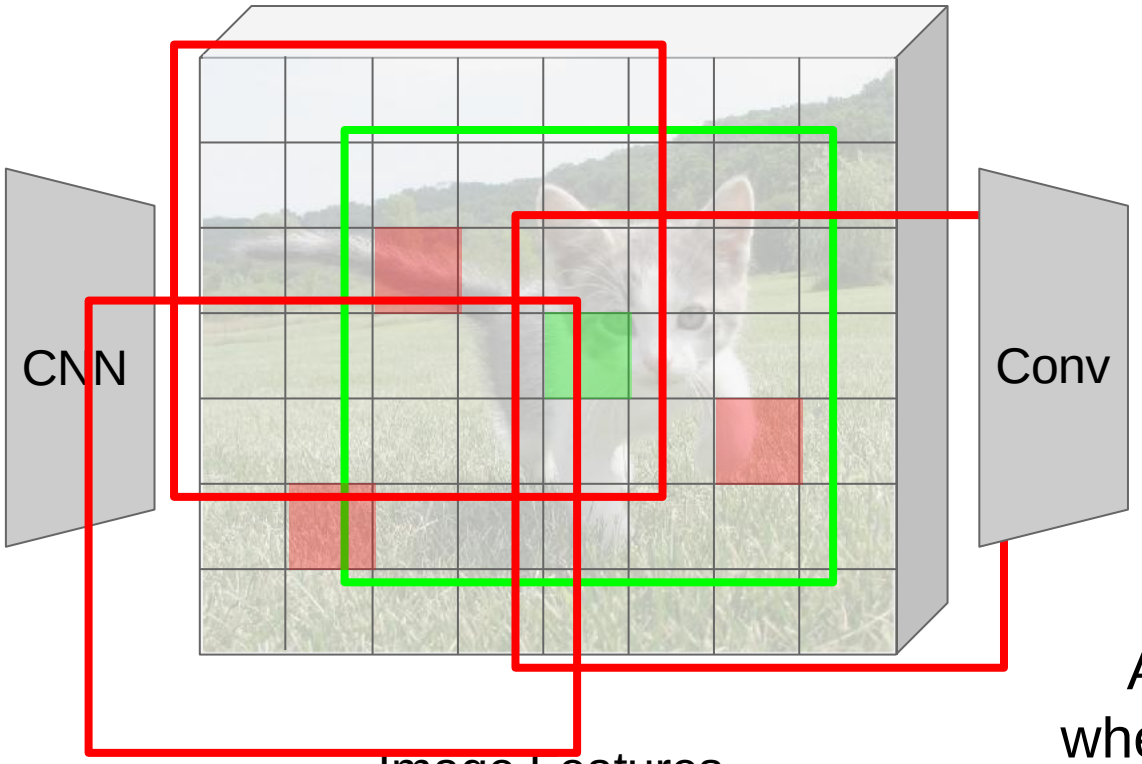


Image Features
(e.g. 512 x 20 x 15)

Anchor is an Object?
1 x 20 x 15

At each Point, predict whether the corresponding Anchor contains an Object (binary Classification)

Region Proposal Network

Imagine an **Anchor Box** of fixed Size at each Point in the Feature Map

Input Image
(e.g. 3 x 640 x 480)

CNN

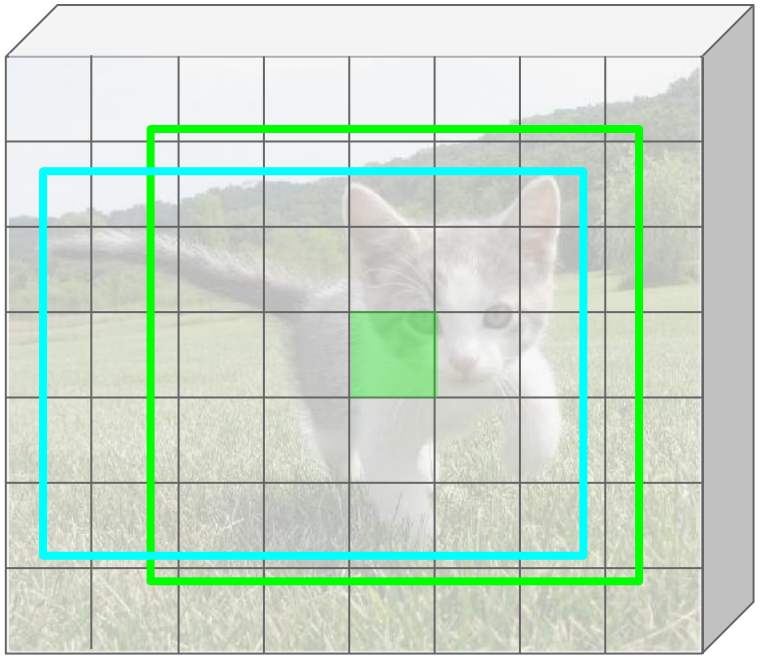


Image Features
(e.g. 512 x 20 x 15)

Conv

Anchor is an object?
1 x 20 x 15

Box corrections
4 x 20 x 15

For positive Boxes, also predict a Corrections from the Anchor to the Ground-Truth Box (regress 4 Numbers per Pixel)

Region Proposal Network

In practice use **K** different Anchor Boxes of different Size / Scale at each Point

Input Image
(e.g. 3 x 640 x 480)

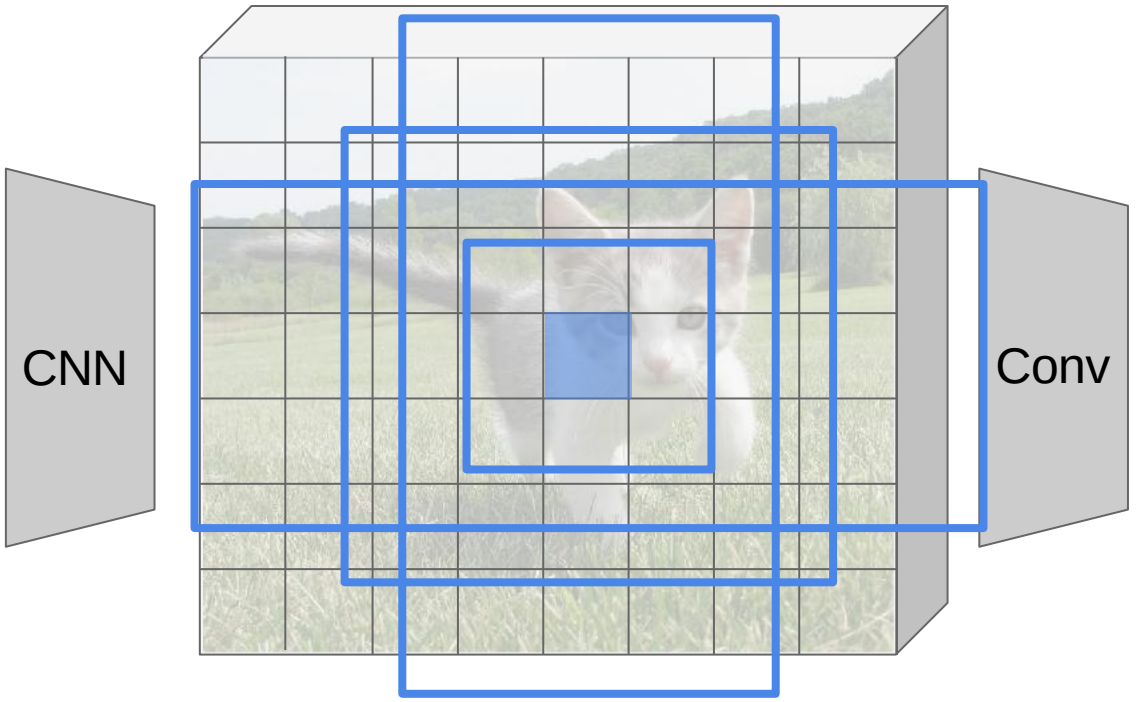


Image Features
(e.g. 512 x 20 x 15)

Anchor is an Object?
K x 20 x 15

Box Transforms
4K x 20 x 15

Region Proposal Network

In practice use **K** different Anchor Boxes of different Size / Scale at each Point

Input Image
(e.g. 3 x 640 x 480)

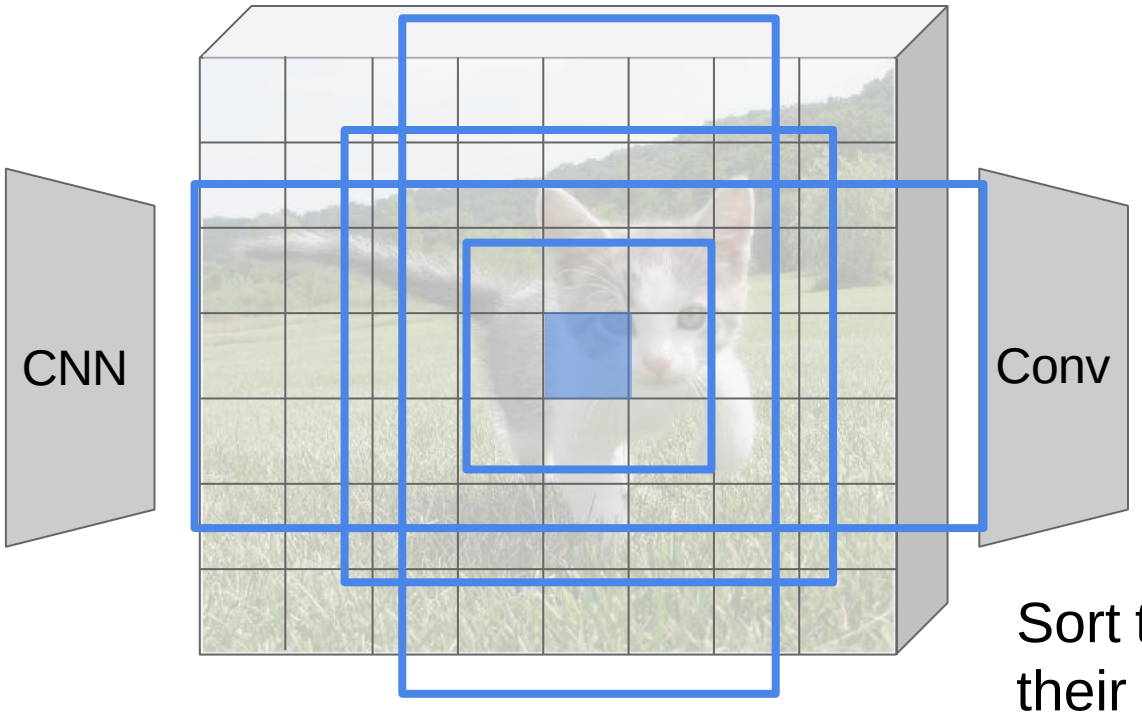


Image Features
(e.g. 512 x 20 x 15)

Anchor is an Object?
K x 20 x 15

Box Transforms
4K x 20 x 15

Sort the K*20*15 Boxes by their “Objectness” Score, take top ~300 as our Proposals

R-CNN, Fast R-CNN, Faster R-CNN

Rich feature hierarchies for accurate object detection and semantic segmentation

Tech report (v5)

Ross Girshick Jeff Donahue Trevor Darrell Jitendra Malik
UC Berkeley
{rbg,jdonahue,trevor,malik}@eecs.berkeley.edu

arXiv:1311.2524v5 [cs.CV] 22 Oct 2014

Abstract

Object detection performance, as measured on the canonical PASCAL VOC dataset, has plateaued in the last few years. The best-performing methods are complex ensemble systems that typically combine multiple low-level image features with high-level context. In this paper, we propose a simple and scalable detection algorithm that improves mean average precision (mAP) by more than 30% relative to the previous best result on VOC 2012—achieving a mAP of 53.3%. Our approach combines two key insights: (1) one can apply high-capacity convolutional neural networks (CNNs) to bottom-up region proposals in order to localize and segment objects and (2) when labeled training data is scarce, supervised pre-training for an auxiliary task, followed by domain-specific fine-tuning, yields a significant performance boost. Since we combine region proposals with CNNs, we call our method R-CNN: *Regions with CNN features. We also compare R-CNN to OverFeat, a recently proposed sliding-window detector based on a similar CNN architecture. We find that R-CNN outperforms OverFeat by a large margin on the 200-class ILSVRC2013 detection dataset. Source code for the complete system is available at http://www.cs.berkeley.edu/~rbg/rcnn.*

R-CNN: *Regions with CNN features*

Figure 1: Object detection system overview. Our system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs. R-CNN achieves a mean average precision (mAP) of **53.7% on PASCAL VOC 2010**. For comparison, [39] reports 35.1% mAP using the same region proposals, but with a spatial pyramid and bag-of-visual-words approach. The popular deformable part models perform at 33.4%. On the 200-class **ILSVRC2013 detection dataset, R-CNN's mAP is 31.4%**, a large improvement over OverFeat [34], which had the previous best result at 24.3%.

1. Introduction

Features matter. The last decade of progress on various visual recognition tasks has been based considerably on the use of SIFT [29] and HOG [7]. But if we look at performance on the canonical visual recognition task, PASCAL VOC object detection [15], it is generally acknowledged that progress has been slow during 2010-2012, with small gains obtained by building ensemble systems and employing minor variants of successful methods.

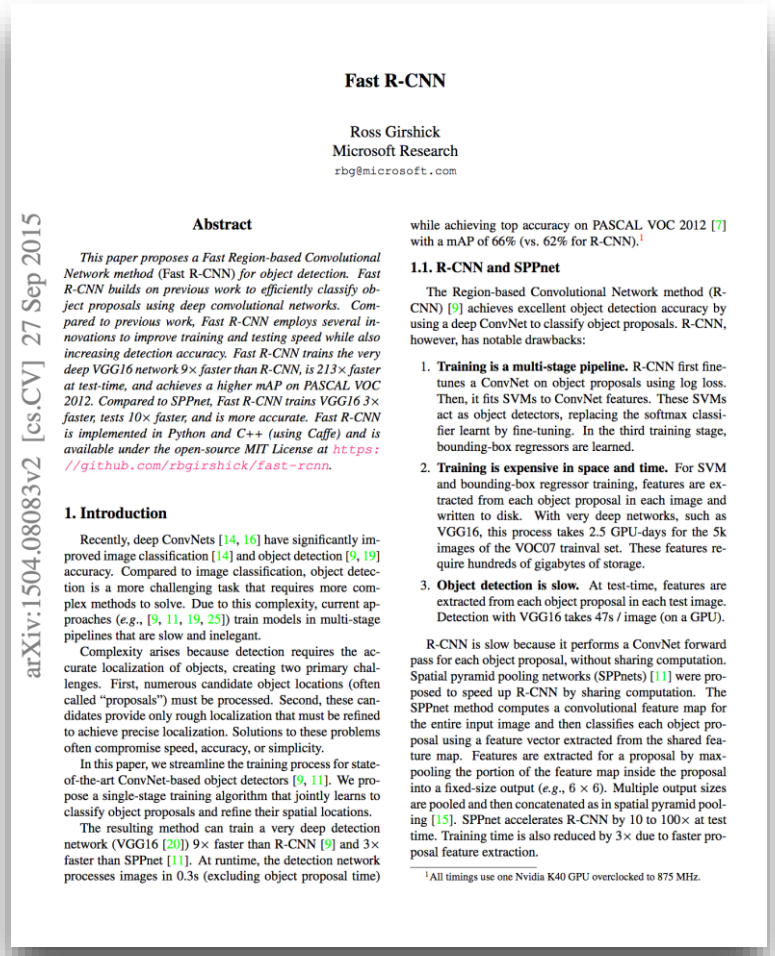
SIFT and HOG are blockwise orientation histograms, a representation we could associate roughly with complex cells in V1, the first cortical area in the primate visual pathway. But we also know that recognition occurs several stages downstream, which suggests that there might be hierarchical, multi-stage processes for computing features that are even more informative for visual recognition.

Fukushima's "neocognitron" [19], a biologically-inspired hierarchical and shift-invariant model for pattern recognition, was an early attempt at just such a process. The neocognitron, however, lacked a supervised training algorithm. Building on Rumelhart et al. [33], LeCun et al. [26] showed that stochastic gradient descent via back-propagation was effective for training convolutional neural networks (CNNs), a class of models that extend the neocognitron.

CNNs saw heavy use in the 1990s (e.g., [27]), but then fell out of fashion with the rise of support vector machines. In 2012, Krizhevsky et al. [25] rekindled interest in CNNs by showing substantially higher image classification accuracy on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [9, 10]. Their success resulted from training a large CNN on 1.2 million labeled images, together with a few twists on LeCun's CNN (e.g., $\max(x, 0)$ rectifying non-linearities and "dropout" regularization).

The significance of the ImageNet result was vigorously

https://arxiv.org/pdf/1311.2524.pdf



Fast R-CNN

Ross Girshick
Microsoft Research
rbg@microsoft.com

arXiv:1504.08083v2 [cs.CV] 27 Sep 2015

Abstract

This paper proposes a Fast Region-based Convolutional Network method (Fast R-CNN) *for object detection. Fast R-CNN builds on previous work to efficiently classify object proposals using deep convolutional networks. Compared to previous work, Fast R-CNN employs several innovations to improve training and testing speed while also increasing detection accuracy. Fast R-CNN trains the very deep VGG16 network 9× faster than R-CNN, is 213× faster at test-time, and achieves a higher mAP on PASCAL VOC 2012. Compared to SPPnet, Fast R-CNN trains VGG16 3× faster, tests 10× faster, and is more accurate. Fast R-CNN is implemented in Python and C++ (using Caffe) and is available under the open-source MIT License at https://github.com/rbgirshick/fast-rcnn.*

1. Introduction

Recently, deep ConvNets [14, 16] have significantly improved image classification [14] and object detection [9, 19] accuracy. Compared to image classification, object detection is a more challenging task that requires more complex methods to solve. Due to this complexity, current approaches (*e.g.*, [9, 11, 19, 25]) train models in multi-stage pipelines that are slow and inelegant.

Complexity arises because detection requires the accurate localization of objects, creating two primary challenges. First, numerous candidate object locations (often called "proposals") must be processed. Second, these candidates provide only rough localization that must be refined to achieve precise localization. Solutions to these problems often compromise speed, accuracy, or simplicity.

In this paper, we streamline the training process for state-of-the-art ConvNet-based object detectors [9, 11]. We propose a single-stage training algorithm that jointly learns to classify object proposals and refine their spatial locations.

The resulting method can train a very deep detection network (VGG16 [20]) 9× faster than R-CNN [9] and 3× faster than SPPnet [11]. At runtime, the detection network processes images in 0.3s (excluding object proposal time) while achieving top accuracy on PASCAL VOC 2012 [7] with a mAP of 66% (vs. 62% for R-CNN).[1]

1.1. R-CNN and SPPnet

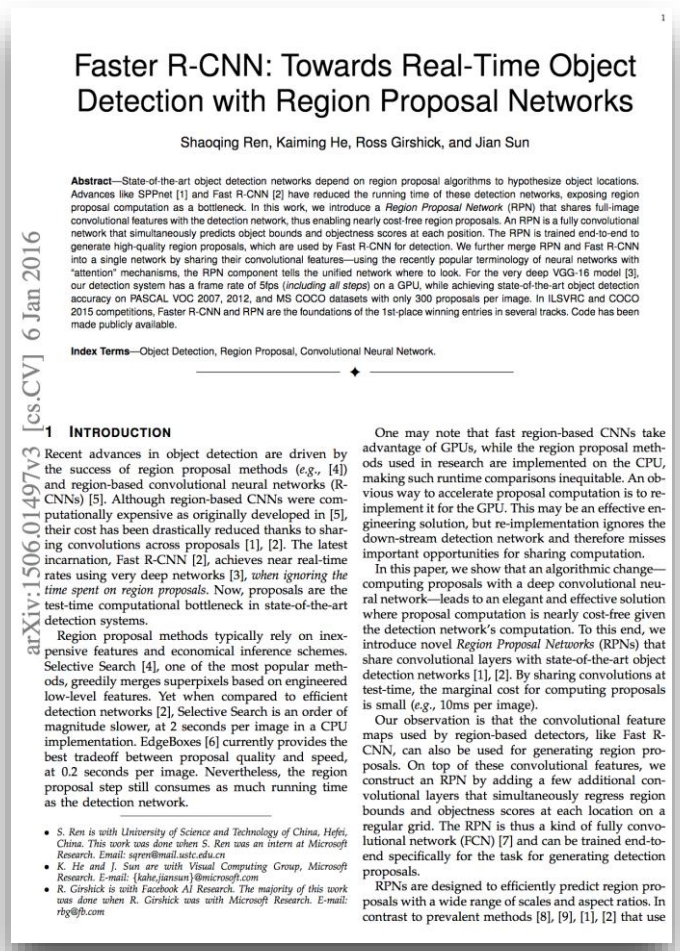
The Region-based Convolutional Network method (R-CNN) [9] achieves excellent object detection accuracy by using a deep ConvNet to classify object proposals. R-CNN, however, has notable drawbacks:

1. **Training is a multi-stage pipeline.** R-CNN first fine-tunes a ConvNet on object proposals using log loss. Then, it fits SVMs to ConvNet features. These SVMs act as object detectors, replacing the softmax classifier learnt by fine-tuning. In the third training stage, bounding-box regressors are learned.
2. **Training is expensive in space and time.** For SVM and bounding-box regressor training, features are extracted from each object proposal in each image and written to disk. With very deep networks, such as VGG16, this process takes 2.5 GPU-days for the 5k images of the VOC07 trainval set. These features require hundreds of gigabytes of storage.
3. **Object detection is slow.** At test-time, features are extracted from each object proposal in each test image. Detection with VGG16 takes 47s / image (on a GPU).

R-CNN is slow because it performs a ConvNet forward pass for each object proposal, without sharing computation. Spatial pyramid pooling networks (SPPnets) [11] were proposed to speed up R-CNN by sharing computation. The SPPnet method computes a convolutional feature map for the entire input image and then classifies each object proposal using a feature vector extracted from the shared feature map. Features are extracted for a proposal by max-pooling the portion of the feature map inside the proposal into a fixed-size output (*e.g.*, 6×6). Multiple output sizes are pooled and then concatenated as in spatial pyramid pooling [15]. SPPnet accelerates R-CNN by 10 to 100× at test time. Training time is also reduced by 3× due to faster proposal feature extraction.

[1] All timings use one Nvidia K40 GPU overclocked to 875 MHz.

https://arxiv.org/pdf/1504.08083.pdf



Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun

arXiv:1506.01497v3 [cs.CV] 6 Jan 2016

Abstract—State-of-the-art object detection networks depend on region proposal algorithms to hypothesize object locations. Advances like SPPnet [1] and Fast R-CNN [2] have reduced the running time of these detection networks, exposing region proposal computation as a bottleneck. In this work, we introduce a *Region Proposal Network* (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. An RPN is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position. The RPN is trained end-to-end to generate high-quality region proposals, which are used by Fast R-CNN for detection. We further merge RPN and Fast R-CNN into a single network by sharing their convolutional features—using the recently popular terminology of neural networks with "attention" mechanisms, the RPN component tells the unified network where to look. For the very deep VGG-16 model [3], our detection system has a frame rate of 5fps (*including all steps*) on a GPU, while achieving state-of-the-art object detection accuracy on PASCAL VOC 2007, 2012, and MS COCO datasets with only 300 proposals per image. In ILSVRC and COCO 2015 competitions, Faster R-CNN and RPN are the foundations of the 1st-place winning entries in several tracks. Code has been made publicly available.

Index Terms—Object Detection, Region Proposal, Convolutional Neural Network.

1 Introduction

Recent advances in object detection are driven by the success of region proposal methods (*e.g.*, [4]) and region-based convolutional neural networks (R-CNNs) [5]. Although region-based CNNs were computationally expensive as originally developed in [5], their cost has been drastically reduced thanks to sharing convolutions across proposals [1], [2]. The latest incarnation, Fast R-CNN [2], achieves near real-time rates using very deep networks [3], *when ignoring the time spent on region proposals*. Now, proposals are the test-time computational bottleneck in state-of-the-art detection systems.

Region proposal methods typically rely on inexpensive features and economical inference schemes. Selective Search [4], one of the most popular methods, greedily merges superpixels based on engineered low-level features. Yet when compared to efficient detection networks [2], Selective Search is an order of magnitude slower, at 2 seconds per image in a CPU implementation. EdgeBoxes [6] currently provides the best tradeoff between proposal quality and speed, at 0.2 seconds per image. Nevertheless, the region proposal step still consumes as much running time as the detection network.

- *S. Ren is with University of Science and Technology of China, Hefei, China. This work was done when S. Ren was an intern at Microsoft Research. Email: sqren@mail.ustc.edu.cn*
- *K. He and J. Sun are with Visual Computing Group, Microsoft Research. E-mail: {kahe,jiansun}@microsoft.com*
- *R. Girshick is with Facebook AI Research. The majority of this work was done when R. Girshick was with Microsoft Research. E-mail: rbg@fb.com*

One may note that fast region-based CNNs take advantage of GPUs, while the region proposal methods used in research are implemented on the CPU, making such runtime comparisons inequitable. An obvious way to accelerate proposal computation is to re-implement it for the GPU. This may be an effective engineering solution, but re-implementation ignores the down-stream detection network and therefore misses important opportunities for sharing computation.

In this paper, we show that an algorithmic change—computing proposals with a deep convolutional neural network—leads to an elegant and effective solution where proposal computation is nearly cost-free given the detection network's computation. To this end, we introduce novel *Region Proposal Networks* (RPNs) that share convolutional layers with state-of-the-art object detection networks [1], [2]. By sharing convolutions at test-time, the marginal cost for computing proposals is small (*e.g.*, 10ms per image).

Our observation is that the convolutional feature maps used by region-based detectors, like Fast R-CNN, can also be used for generating region proposals. On top of these convolutional features, we construct an RPN by adding a few additional convolutional layers that simultaneously regress region bounds and objectness scores at each location on a regular grid. The RPN is thus a kind of fully convolutional network (FCN) [7] and can be trained end-to-end specifically for the task for generating detection proposals.

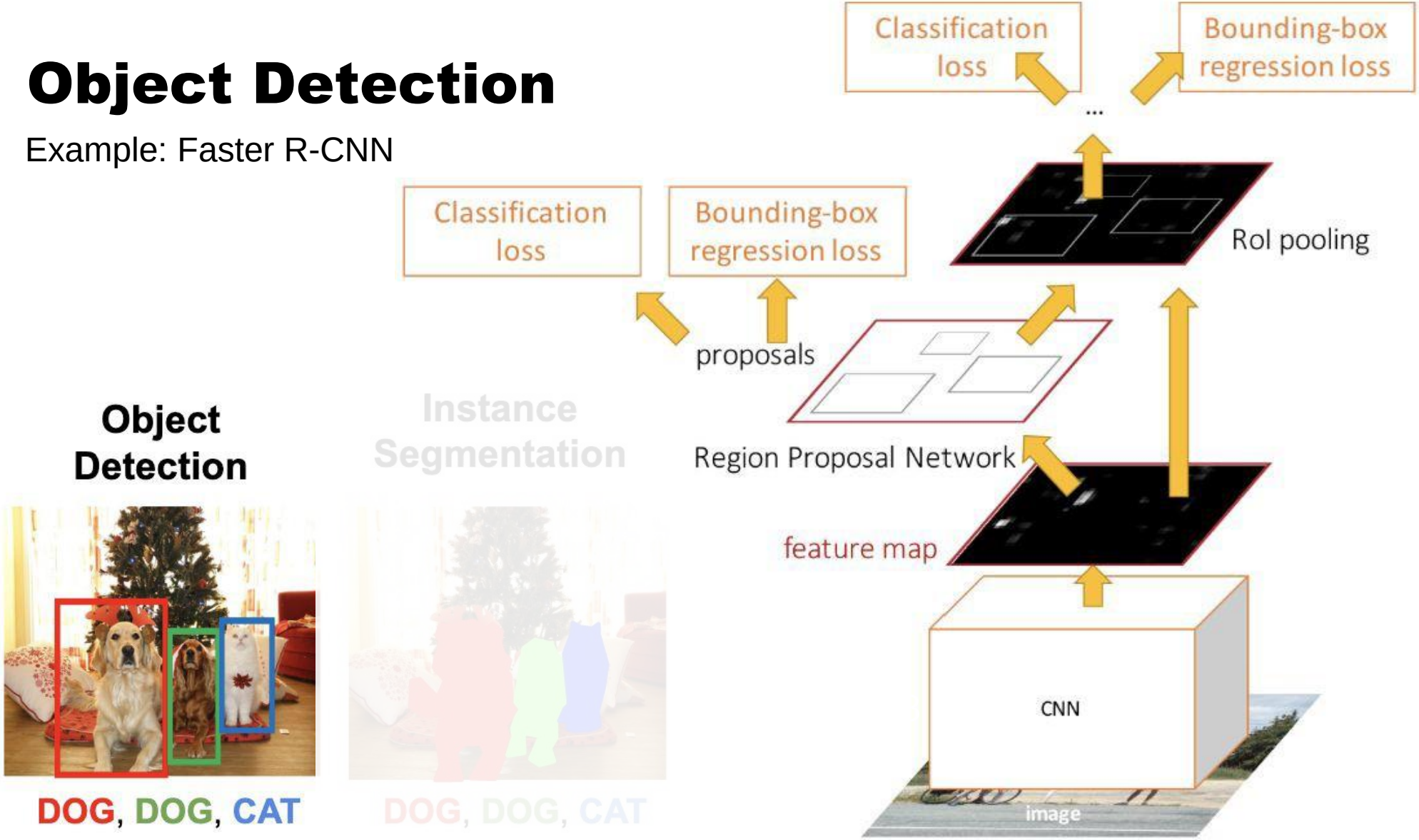
RPNs are designed to efficiently predict region proposals with a wide range of scales and aspect ratios. In contrast to prevalent methods [8], [9], [1], [2] that use

https://arxiv.org/pdf/1506.01497.pdf

JOHANNES KEPLER UNIVERSITY LINZ

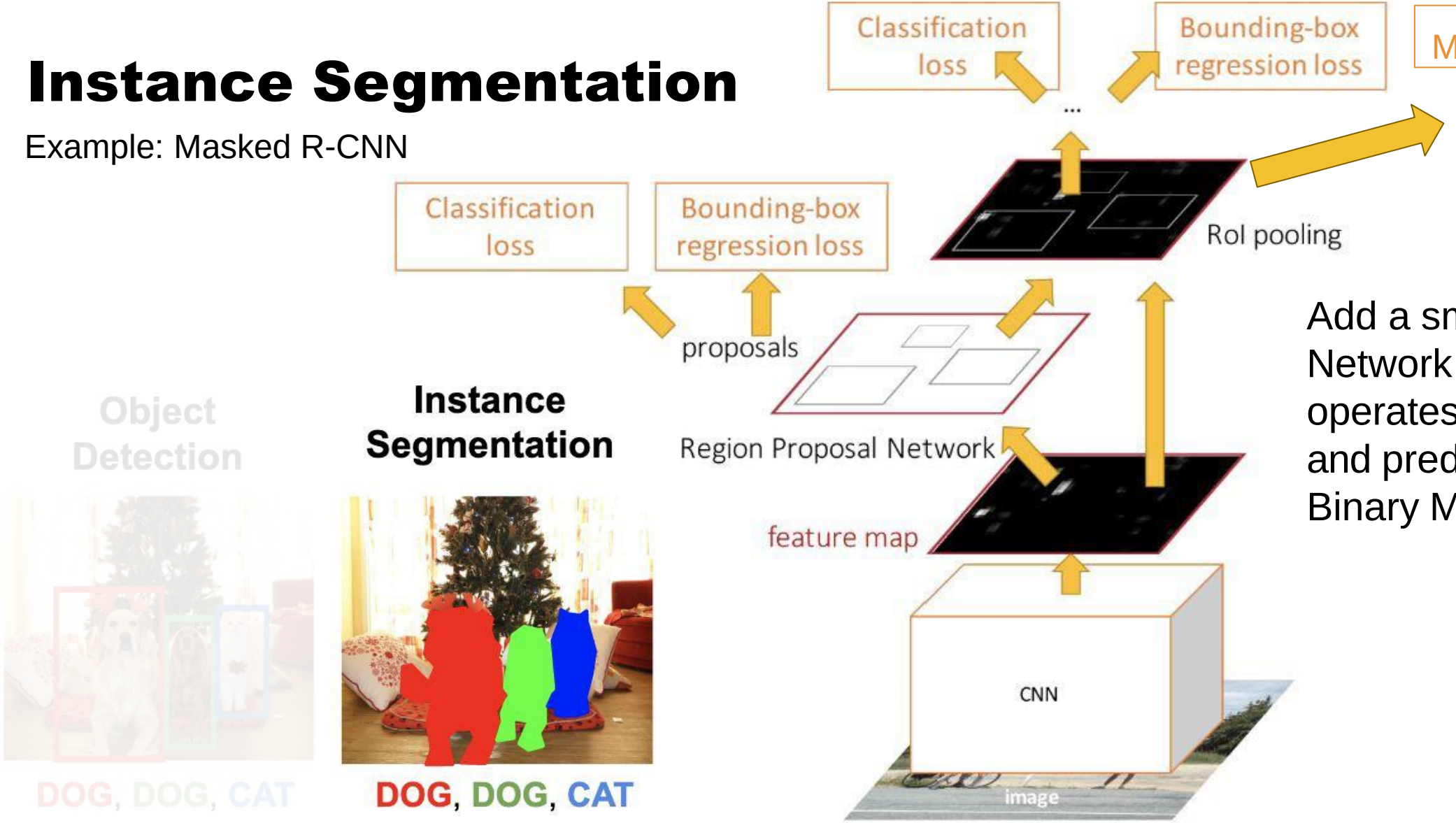
Object Detection

Example: Faster R-CNN



Instance Segmentation

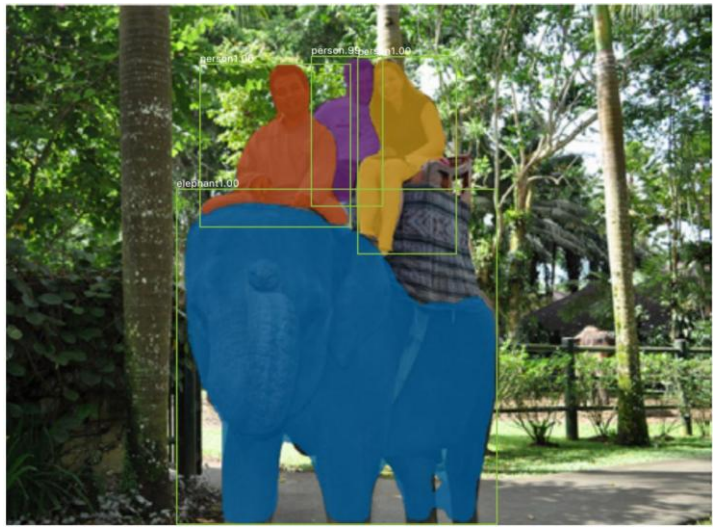
Example: Masked R-CNN

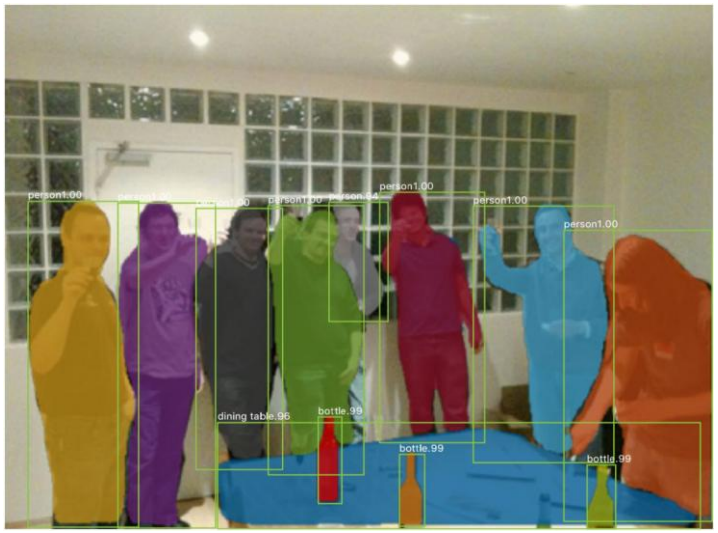


Add a small Mask Network that operates on each RoI and predicts a 28x28 Binary Mask

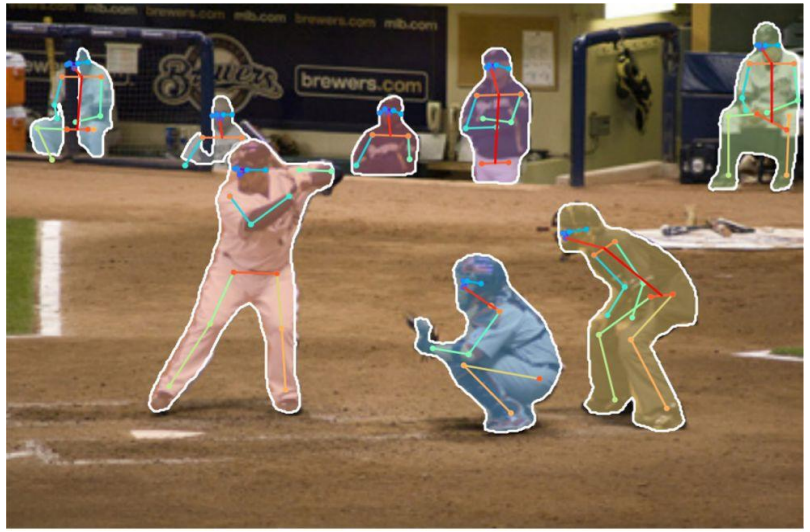
He et al, “Mask R-CNN”, ICCV 2017

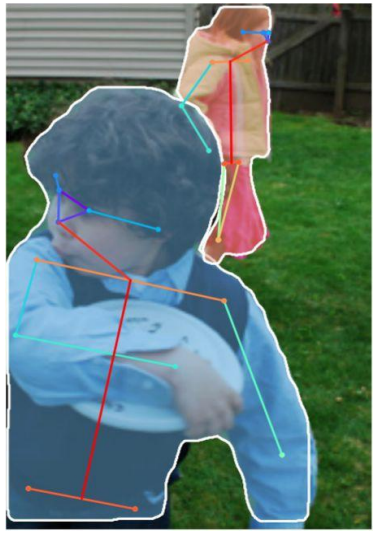
Instance Segmentation





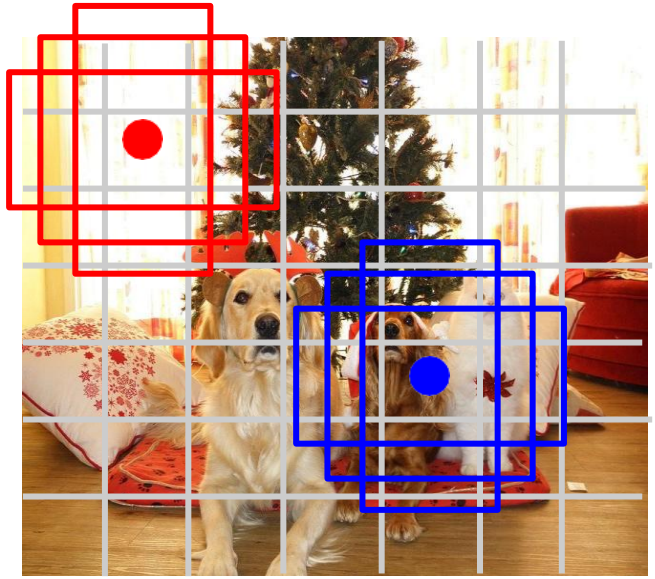
Can also do Poses





Single-Shot Object Detectors: YOLO/SSD/RetinaNet

Input Image
3 x H x W



Divide Image into
Grid 7 x 7

Image a set of **Base Boxes** centered at each Grid Cell Here B = 3

Within each Grid Cell:

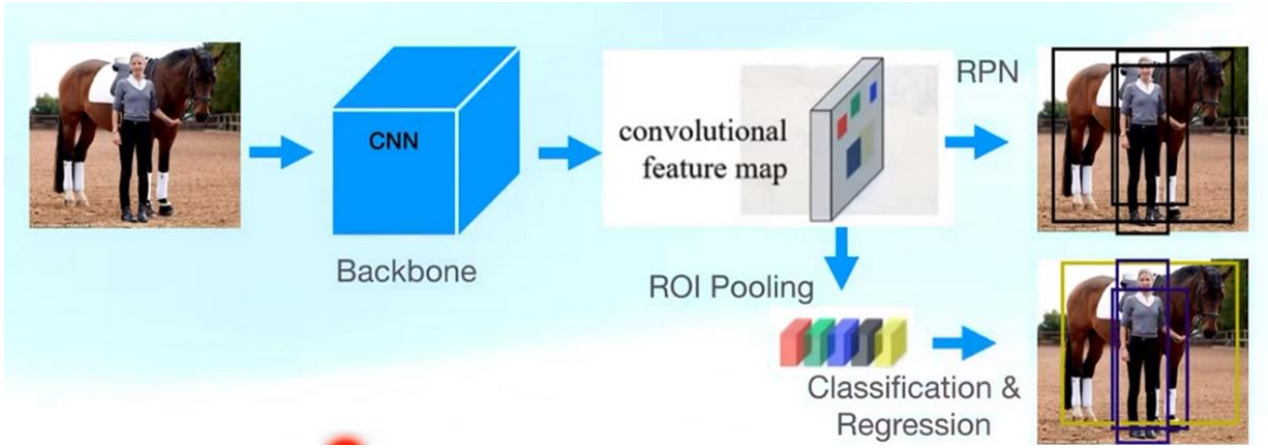
- Regress from each of the B Base Boxes to a final Box with 5 Numbers: (dx, dy, dh, dw, confidence)
- Predict Scores for each of C Classes (including Background as a Class)
- Looks a lot like RPN, but category-specific!
- Only a single Stage needed (Classification and prediction of BB in the same step)

Output:
7 x 7 x (5 * B + C)

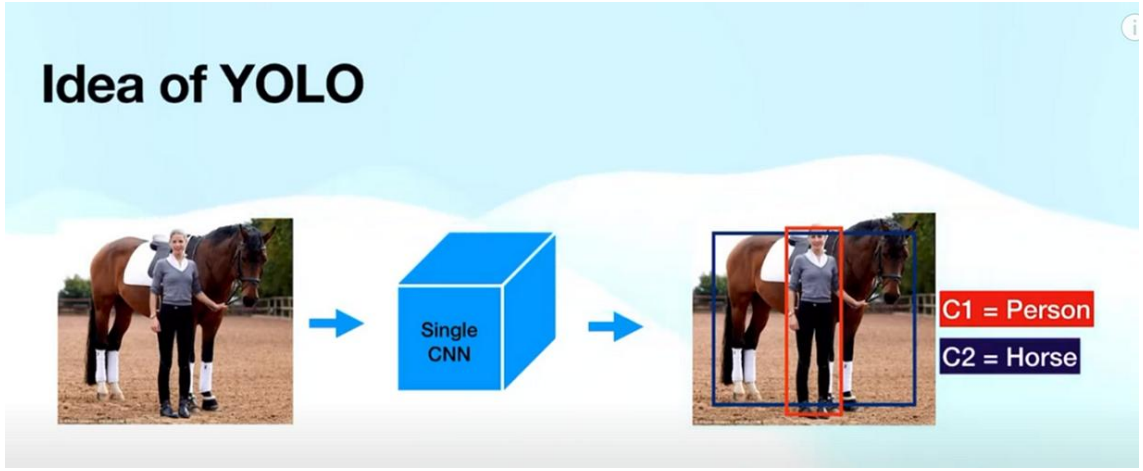
Redmon et al, “You Only Look Once:
Unified, Real-Time Object Detection”, CVPR 2016
Liu et al, “SSD: Single-Shot MultiBox Detector”, ECCV 2016
Lin et al, “Focal Loss for Dense Object Detection”, ICCV 2017

YOLO: You Only Look Once

One CNN to predict Bounding Boxes and Classes



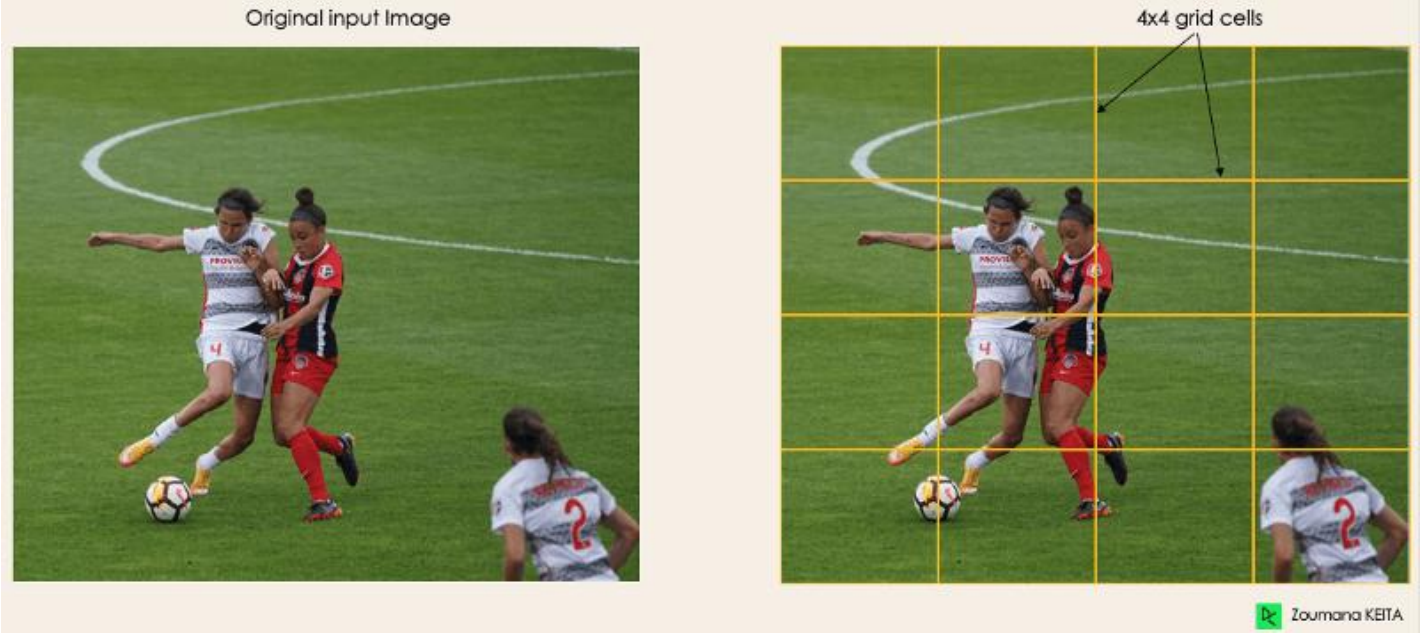
R-CNN



YOLO

YOLO: You Only Look Once

Step 1 (Residual Blocks):
Split image into Grid of SxS (in original paper: 7x7) equal Cells that go through the CNN



The Key Question is, which Cell is responsible for predicting an Object? → The Cell where the center of the object falls

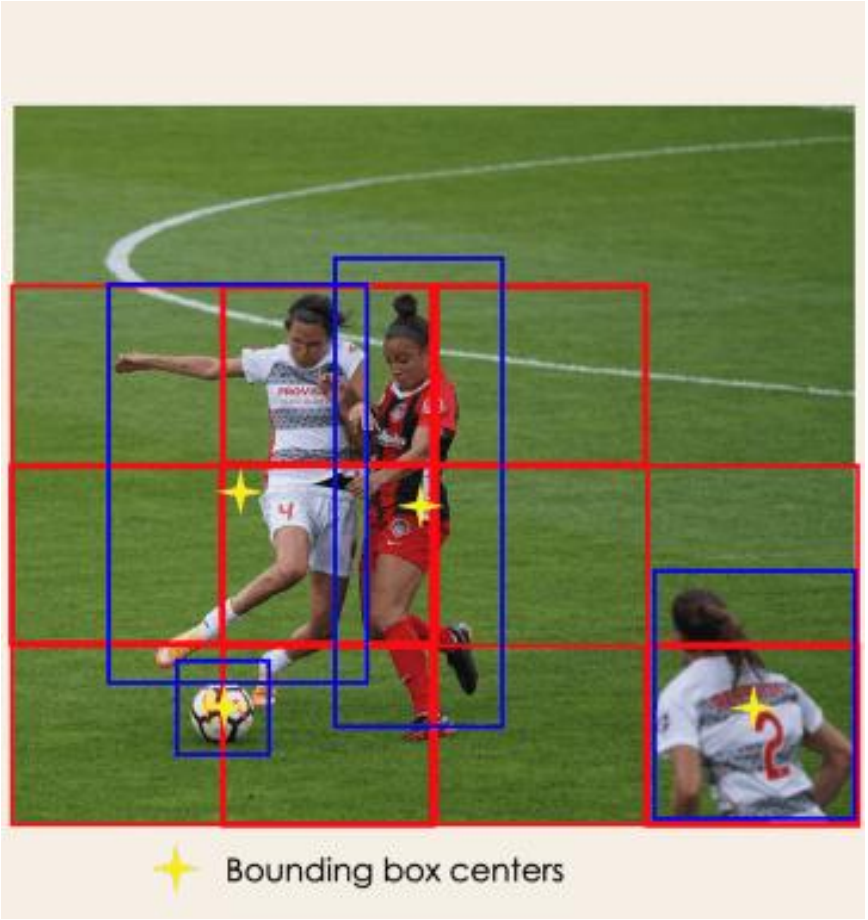
YOLO: You Only Look Once

Step 2 (Bounding Box Regression):
For each Cell predict (regression) Bounding Box Parameters:
Y = [pc, bx, by, bh, bw, c1, c2, …]

pc: Probability of containing an Object (Confidence Score)

bx,by,bh,bw: bounding box coordinates wrt. cell center

c1,c2,…: contained classes

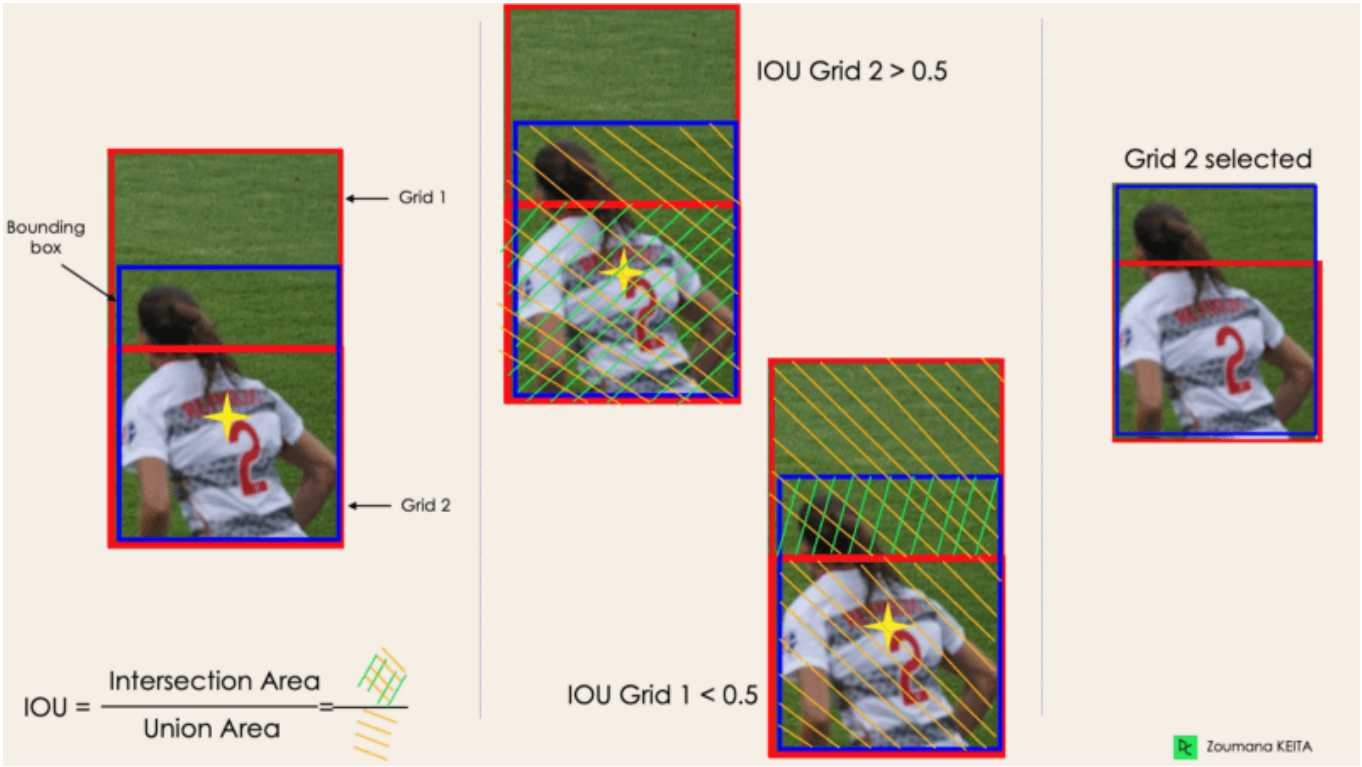


YOLO: You Only Look Once

Step 3 (Intersection over Union):
discard cells with too small IoU (ignore predictions)

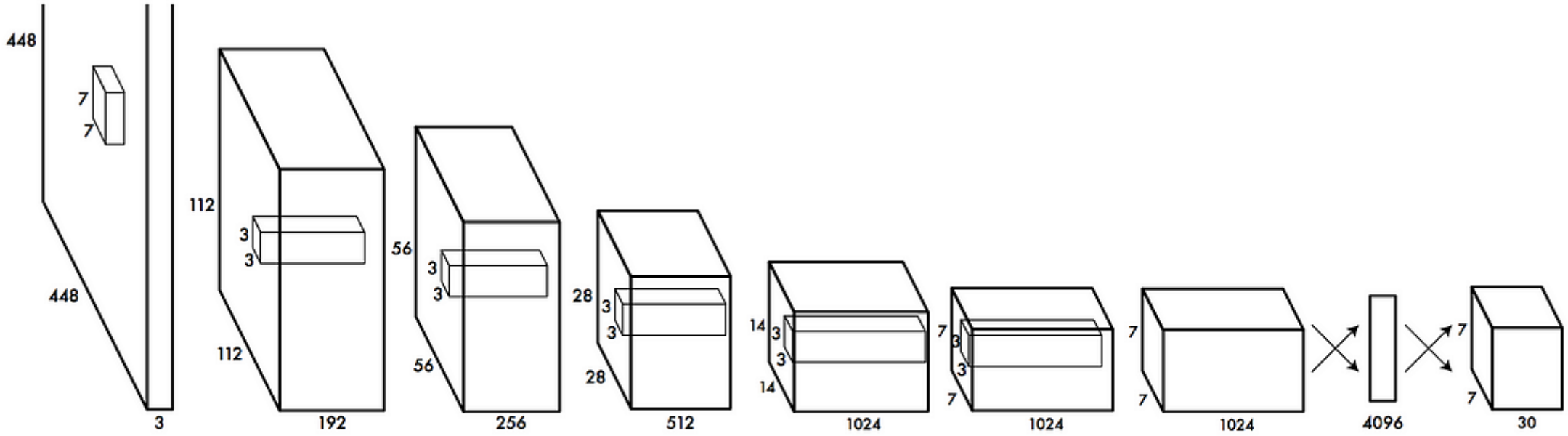
Step 4: (Non-Maximum Suppression):
reduce too many noisy BBs per Object

- Select BB with highest Confidence Score and suppress all other BBs with high IoU wrt. Selection
- Repeat until no BB can be selected anymore



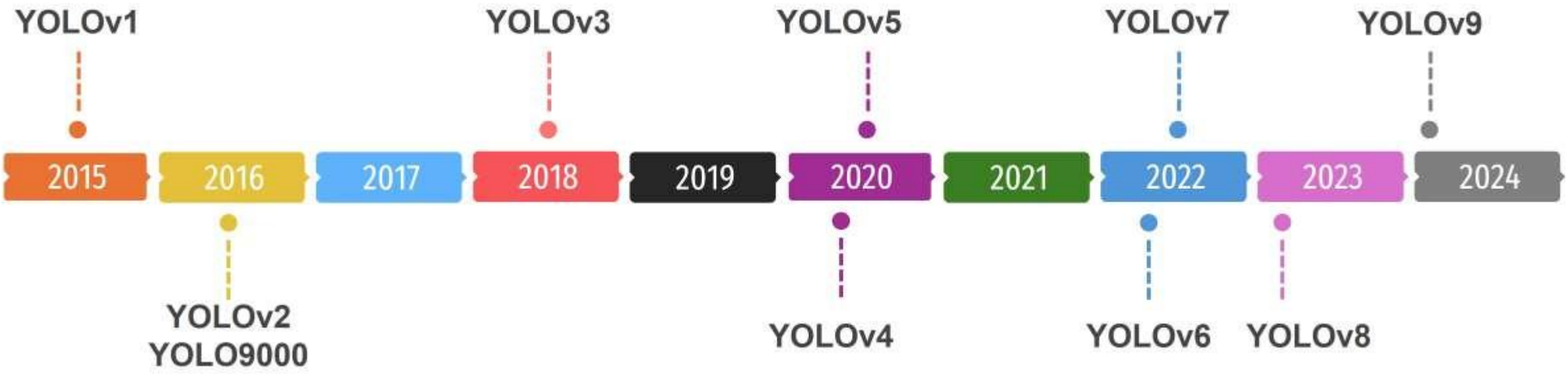
YOLO: You Only Look Once

YOLOv1: Architecture



https://arxiv.org/abs/1506.02640

YOLO: You Only Look Once



Evaluating Classification Performance

We decide (e.g., based on a confidence score threshold) whether a classification is considered correct or not, and this leads to the following cases:

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positives (TP)	False Negative (FN)
Actual Negative	False Positives (FP)	True Negatives (TN)

These cases can now be formalized into different metrics:

Accuracy = (TP+TN)/(TP+TN+FP+FN): *the proportion of all classifications that were correct, whether positive or negative*

Precision = TP/(TP+FP): *proportion of all the model's positive classifications that are actually positive*

Recall = TP/(TP+FN): *the proportion of all actual positives that were classified correctly as positives*

False Positive Rate = FP/(FP+TN): *the proportion of all actual negatives that were classified incorrectly as positives*

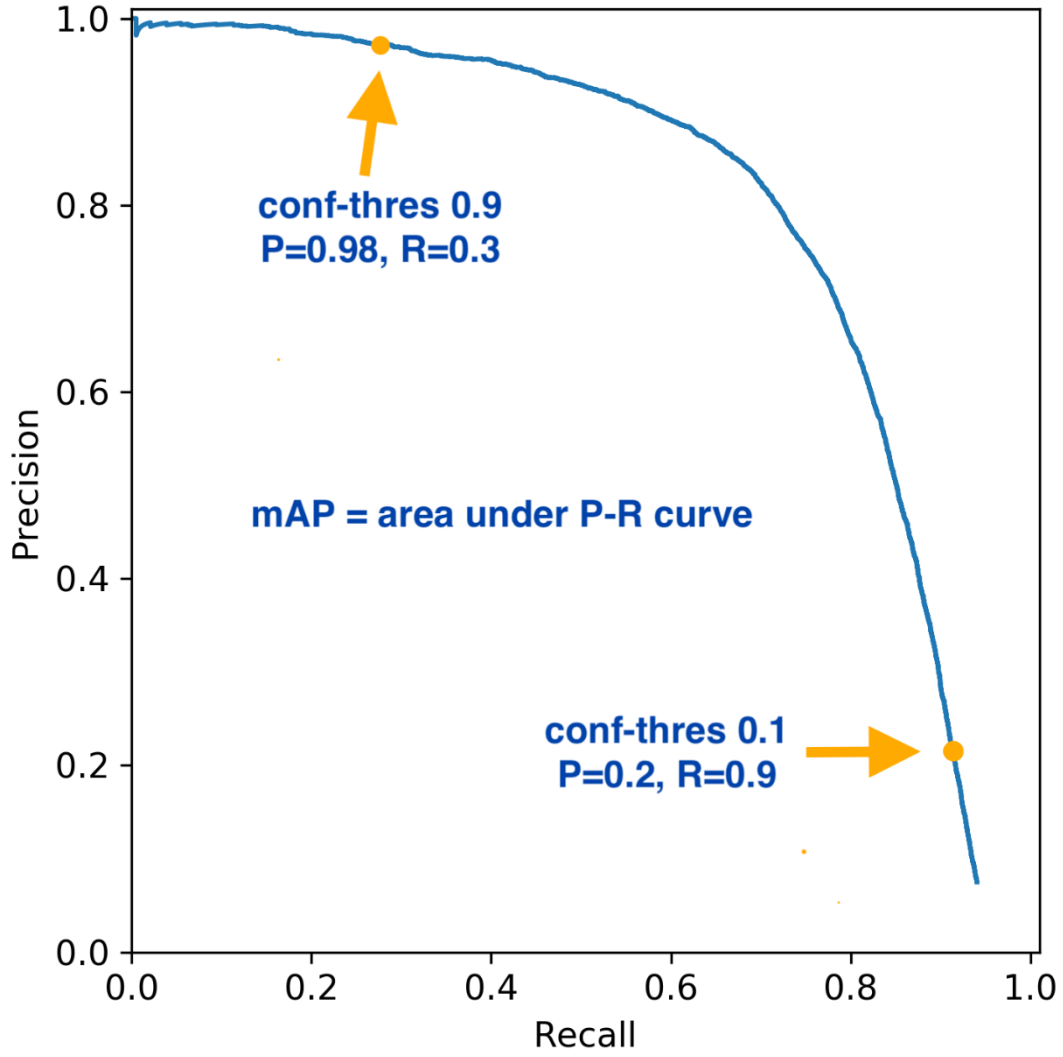
Precision over Recall (P-R) Curve

Let's now consider many Confidence Score Thresholds, each leading to individual Precision and Recall Values

A well trained Classifier has a large Area under the P-R curve (for all Classes)!

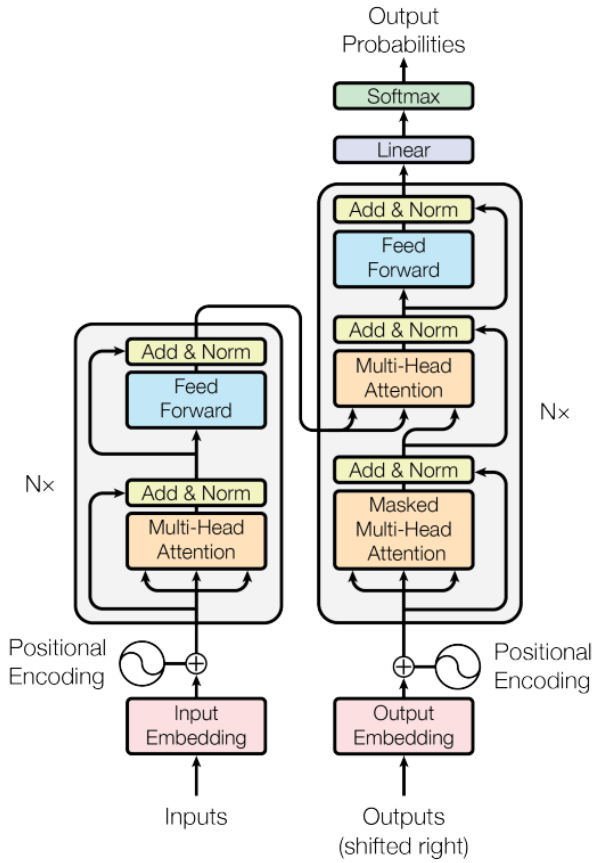
That area is called **Average Precision, AP** (0..1)

AP can be determined for each Class, and the mean over them is called **Mean Average Precision, mAP** (0..1)



A Note on (Vision-) Transformers

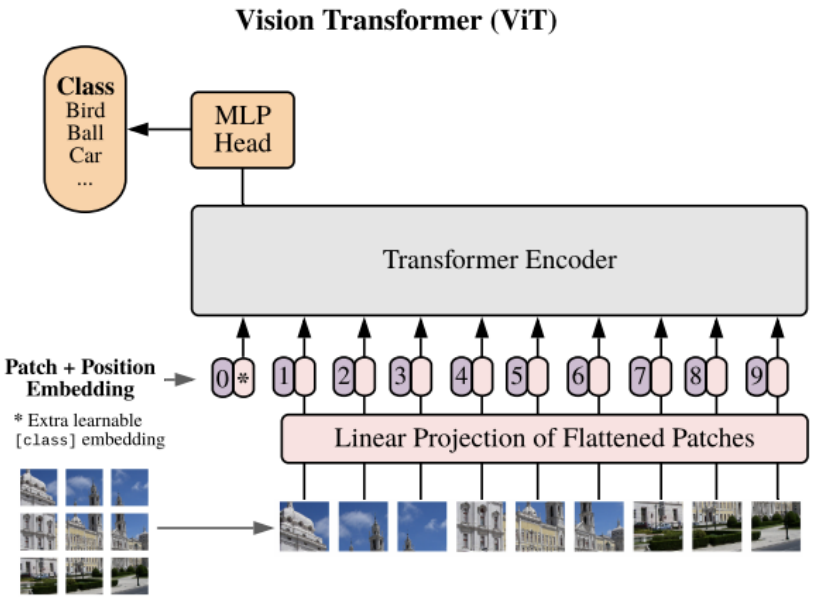
- **Transformers** used for sequences (e.g., in NLP)
  - in contrast to deep CNNs or RNNs can better handle long-distance dependencies (vanishing / exploding gradient problem)
  - relies on attention (e.g., determining context = semantic relations for words at particular positions in a sentence)



Transformer: https://arxiv.org/abs/1706.03762

A Note on (Vision-) Transformers

- **Vision Transformers (ViTs)** used for computer vison (e.g. classification, semantic labeling, etc.)
  - sequence of words = sequence of image patches
  - only the encoder part is used
  - flattened patches = reshape to 1D vector
  - linear projection = flattened patches * weight vector + bias (weight vector + bias has lower dimension and is learned)
  - position encoded added to ensure transformer knows about order of patches (= tokens)
  - for classification, add class token
  - attention determined/learned between all tokens in transformer encoder network (result are feature vectors)
  - example classification: output of transformer encoder (e.g., only class feature vector) = input of MPL / softmax for classification
  - **key idea:** not only features are relevant (like in CNNs) but their relationship (context) to each other



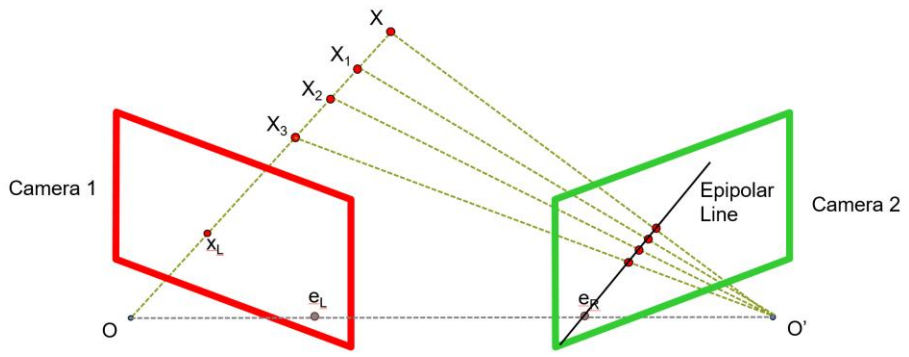
Vision Transformer (for classification):
https://github.com/gupta-abhay/pytorch-vit

Course Overview

CW	Topic	Date	Place	Lab
41	Introduction and Course Overview	07.10.2025	Zoom	Lab 1
42	Capturing Digital Images	14.10.2025	Zoom	Lab 2
43	Digital Image Processing	21.10.2025	Zoom	Assignment 1
44	Machine Learning	28.10.2025	Zoom	
45	Feature Extraction	04.11.2025	Zoom	Open Lab 1
46	Segmentation	11.11.2025	Zoom	Assignment 2
47	Optical Flow	18.11.2025	Zoom	Open Lab 2
48	Object Detection	25.11.2025	Zoom	Assignment 3
49	Multi-View Geometry	02.12.2025	Zoom	Open Lab 3
50	3D Vision	10.12.2025	Zoom	Assignment 4
3	Trends in Computer Vision	13.01.2026	Zoom	
4	Q&A	20.01.2026	Zoom	Open Lab 4
5	Exam	27.01.2026	HS1 (Linz), S1/S3 (Vienna), S5 (Bregenz)	
9	Retry Exam	24.02.2026	tba	

Next Week: Multi-View Geometry

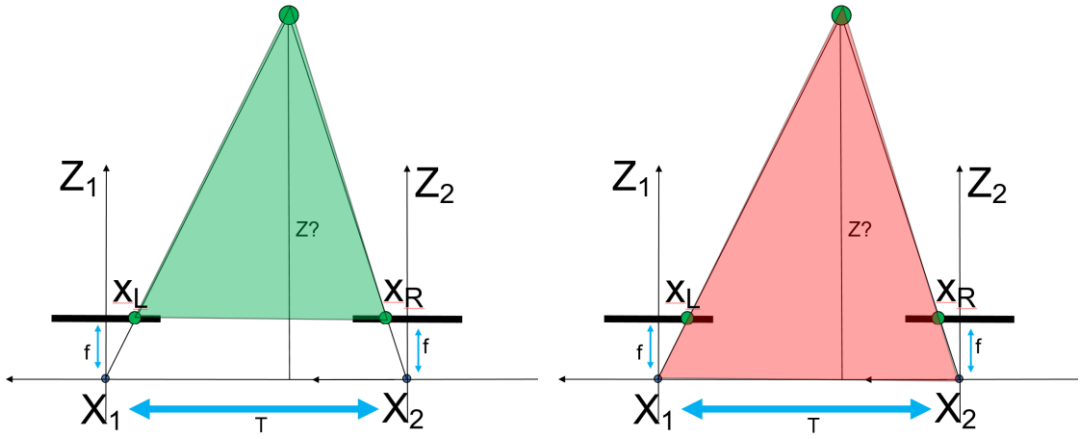
Epipolar Constraints



It's a 1D Search Problem!
But how do we get the Epipolar Line for a given Point?

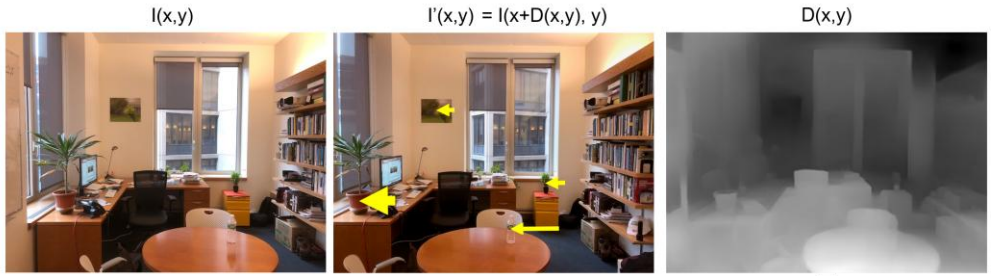
JOHANNES KEPLER UNIVERSITY LINZ

Stereoscopic Depth-Reconstruction



JOHANNES KEPLER UNIVERSITY LINZ

Disparity Maps



$$Z(x,y) \propto \frac{1}{D(x,y)}$$

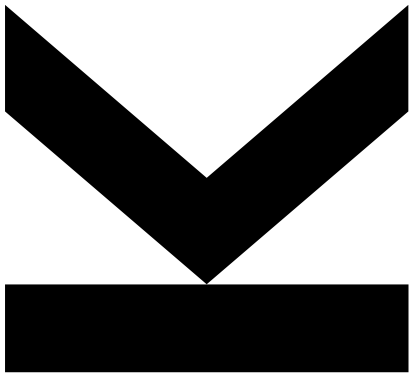
JOHANNES KEPLER UNIVERSITY LINZ

Structure-from-Motion

JOHANNES KEPLER UNIVERSITY LINZ

JOHANNES KEPLER UNIVERSITY LINZ

Thank You



JOHANNES KEPLER
UNIVERSITY LINZ