6 - Object Detection

Marco Willi

Introduction

Object detection is a core task of computer vision. In object detection, the goal is to localize and classify all objects (from a set of known object classes) in an image. Figure 1 shows an example from the paper by Redmon et al. (2016b). Each object is localized with a bounding box and assigned an object class. A bounding box is defined by four parameters: x, y, height, and width.

Figure 2 illustrates the differences between image classification, classification with localization, and object detection.

We will now look step-by-step at how to go from image classification to object detection. First, we will look at landmark detection. In this step, we want to localize specific points in an image or object. This could be the nose, eyes, etc., of a person. Figure 3 shows an example of landmark detection: On the one hand, we want to determine which object class is in the image, and on the other hand, we want to determine the position of the nose. If there are 3 classes, as shown in the example, the network has 3 outputs (logits) $C1, C2, C3$, which are converted into a probability distribution via the softmax transformation.

The question now is: How can I additionally localize the nose?

Figure 4 shows that a simple extension of the network output by 2 scalars is sufficient. This can be used to model the x and y coordinates of the nose. The coordinates could be defined relative to the entire image in the range $x, y \in [0, 1]$.

In the next step, we can go from landmark detection to classification with localization. Figure 5 shows the problem. Now, we want to classify an image and simultaneously localize the object. In addition to x, y coordinates, further outputs need to be defined.

Figure 6 illustrates that with two additional outputs, a bounding box can be defined, which specifies height, width, and a corner point (or the center).

Figure 7 shows how to modify a CNN to localize and classify a single object. One could add two outputs (*heads*) to the CNN backbone: a classification head that models the probability of the object class via softmax transformation and 4 parameters for the bounding box coordinates,

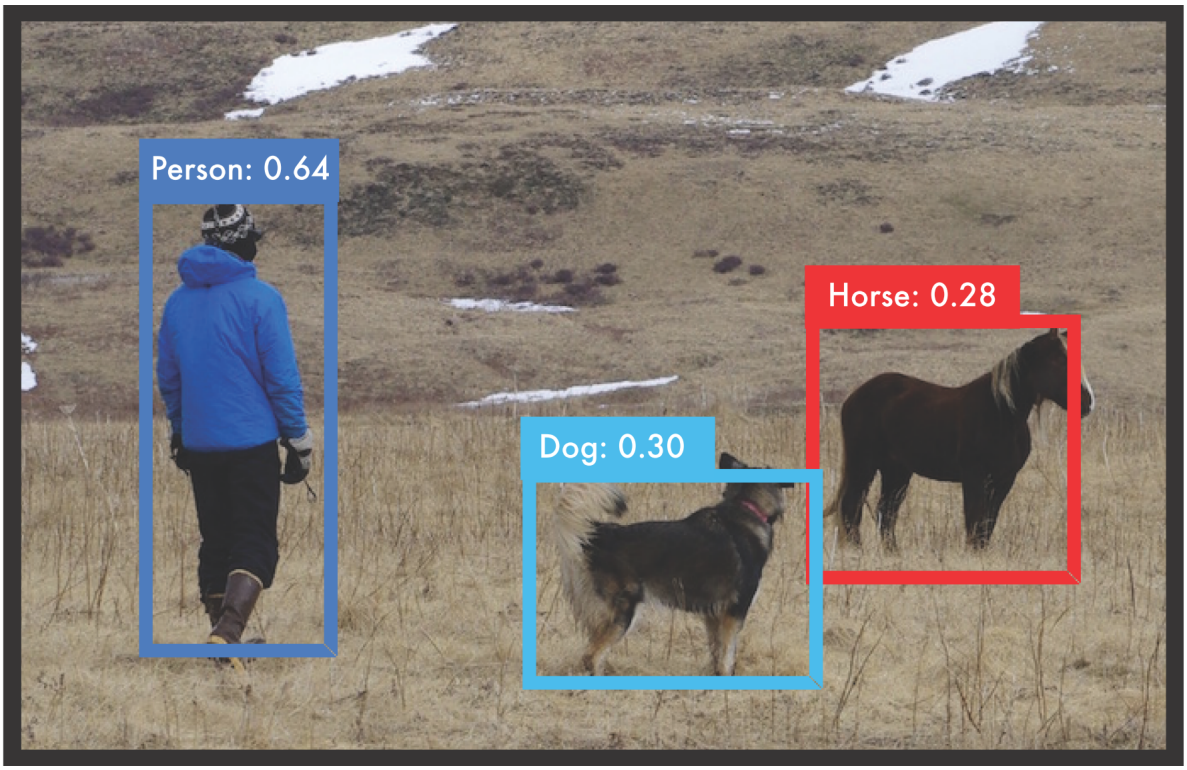

Figure 1: Object detection example (from Redmon et al. (2016b)). Bounding boxes localize the objects, with the most probable class and confidence for each object.

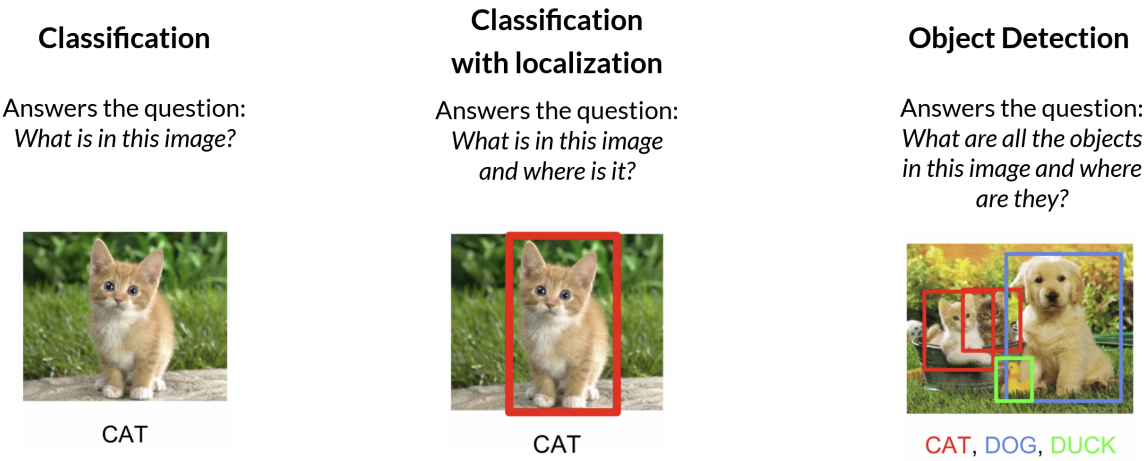

Figure 2: Classification and detection (from Austin et al. (2022)).

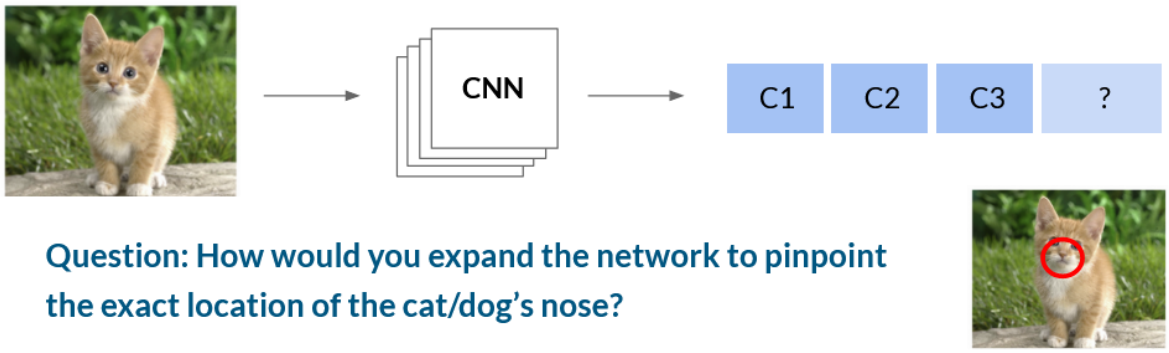

Figure 3: Landmark detection (from Austin et al. (2022)).

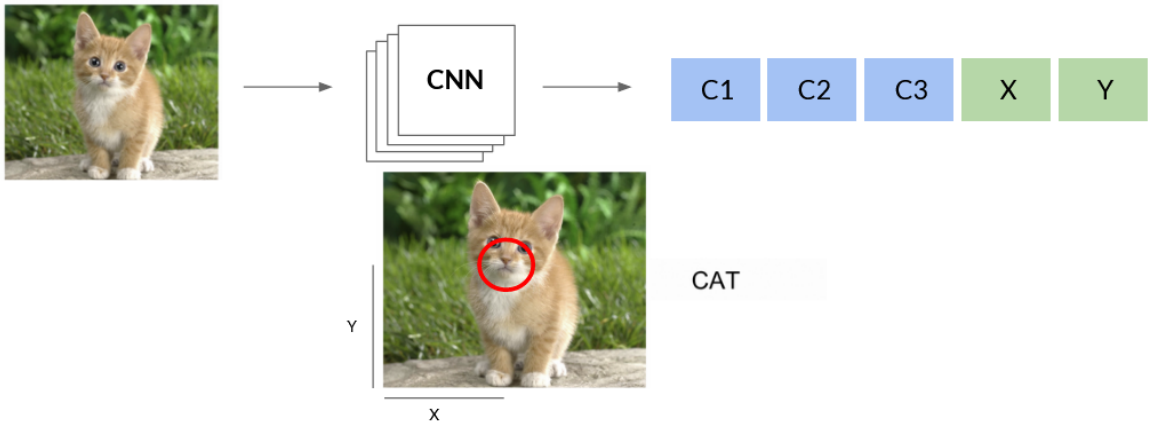

Figure 4: Landmark detection (from Austin et al. (2022)).

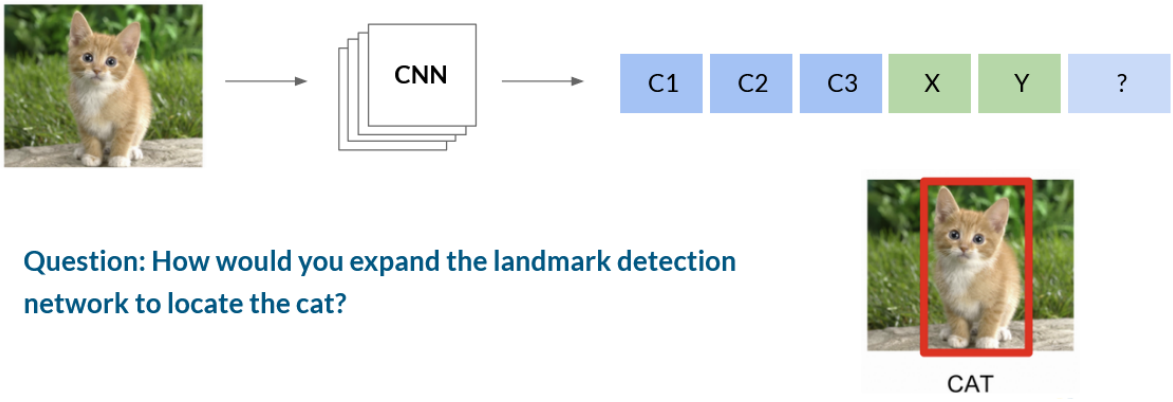

Figure 5: Classification and localization (from Austin et al. (2022)).

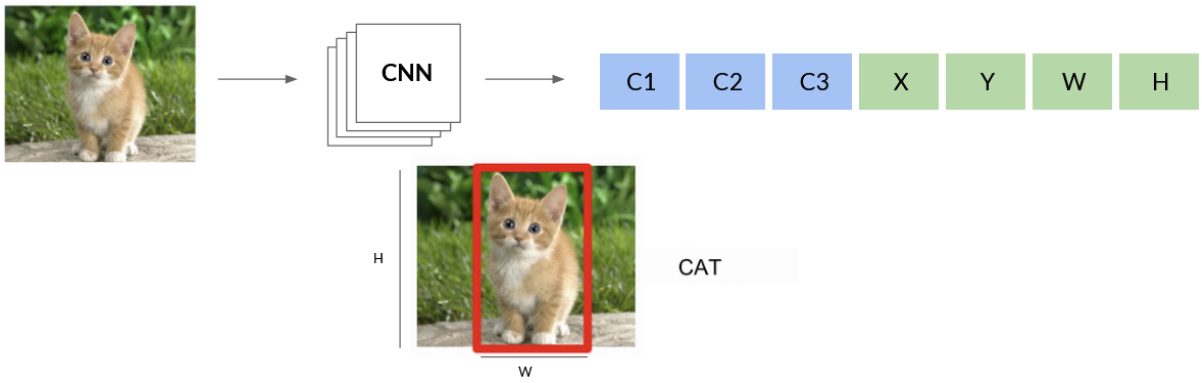

Figure 6: Classification and localization (from Austin et al. (2022)).

which could be optimized with a regression loss, such as Euclidean distance. Thus, two tasks (localization and classification) would be solved simultaneously (*multitask loss*).

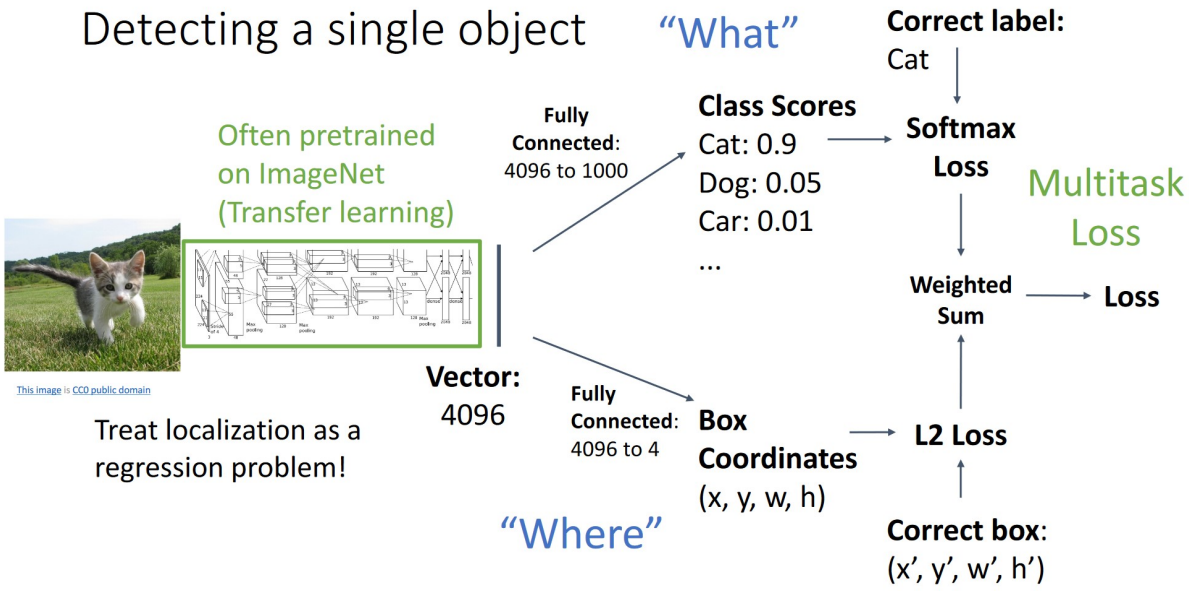

Figure 7: Source: Johnson (2019).

The Challenge

> **Question:**
> What happens if you want to detect more than one object?

Figure 8 illustrates the problem of the variable number of objects. Depending on the image, more or fewer objects need to be detected. This affects the number of outputs the model must have. This variability is one of the biggest challenges in object detection.

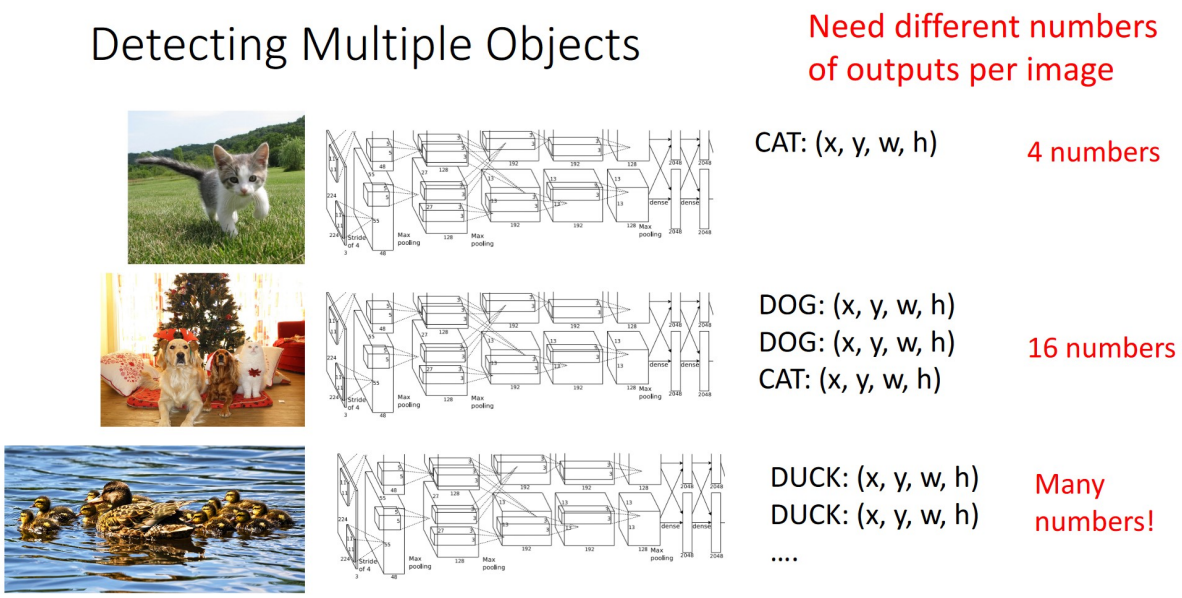

Figure 8: Source: Johnson (2019).

The main challenges in object detection are:

Variable number of outputs: Depending on how many objects are present in an image, the model must be able to output a variable number of detections. This is inherently challenging since model architectures have fixed-size tensors and cannot be easily implemented variably.

Different output types: We need to solve a regression problem (where is the object - bounding box) and a classification problem (what kind of object is it - probability).

Image size: Unlike image classification, object detection requires significantly larger input resolutions, as smaller objects also need to be recognized. This increases the hardware requirements, and such models are more complex to train.

One approach to this problem is the sliding window method. Here, one would classify all possible bounding boxes and additionally add the class background (no object). Figure 9

illustrates the approach.

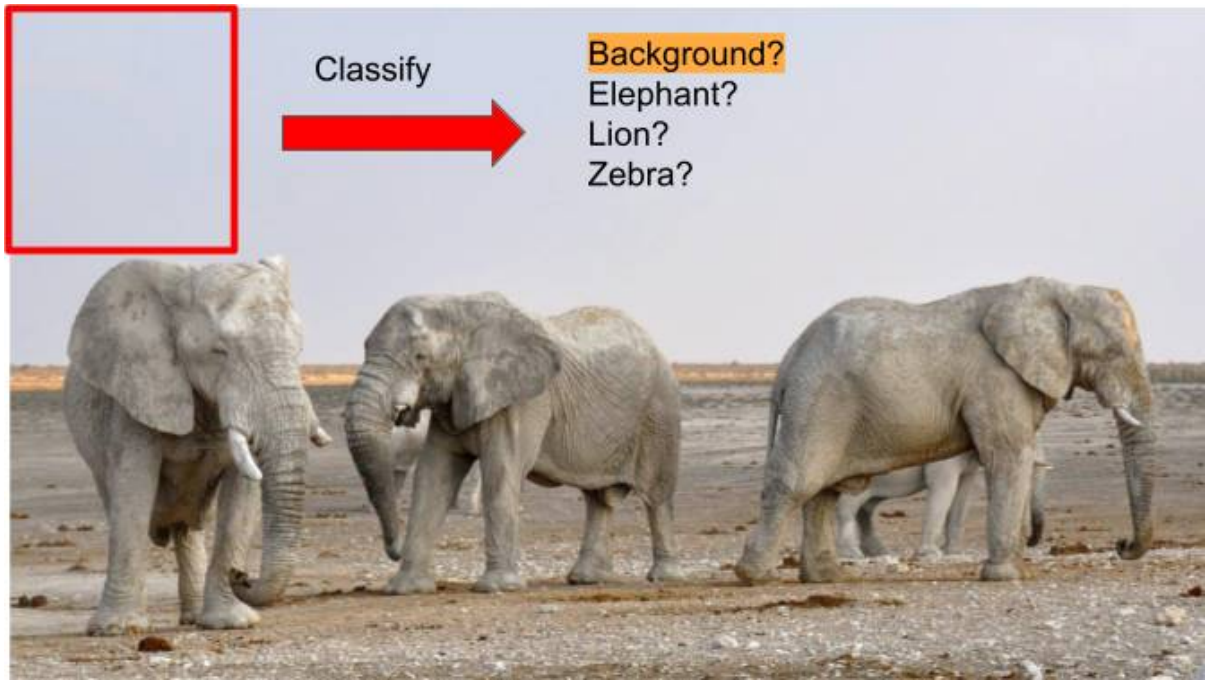

Figure 9: Sliding window approach example.

The problem is that there are too many possible bounding boxes that need to be evaluated. For an image of dimension $H \times W$ and a fixed bounding box of dimension h, w there would be:

- Possible x positions: $W - w + 1$
- Possible y positions: $H - h + 1$
- Possible positions: $(W - w + 1)(H - h + 1)$

> **Object Detection is Hard**
>
> Object detection is a difficult problem and requires many design choices and engineering work.

Object detection has a long history and, like image classification, made a significant leap when deep learning with convolutional neural networks demonstrated efficient image modeling. The publication by Zou et al. (2023) describes this evolution up to the most modern methods and approaches. We will focus on a selection of methods from the two most important approaches in object detection: two-stage detectors and single-stage detectors (see Figure 10).

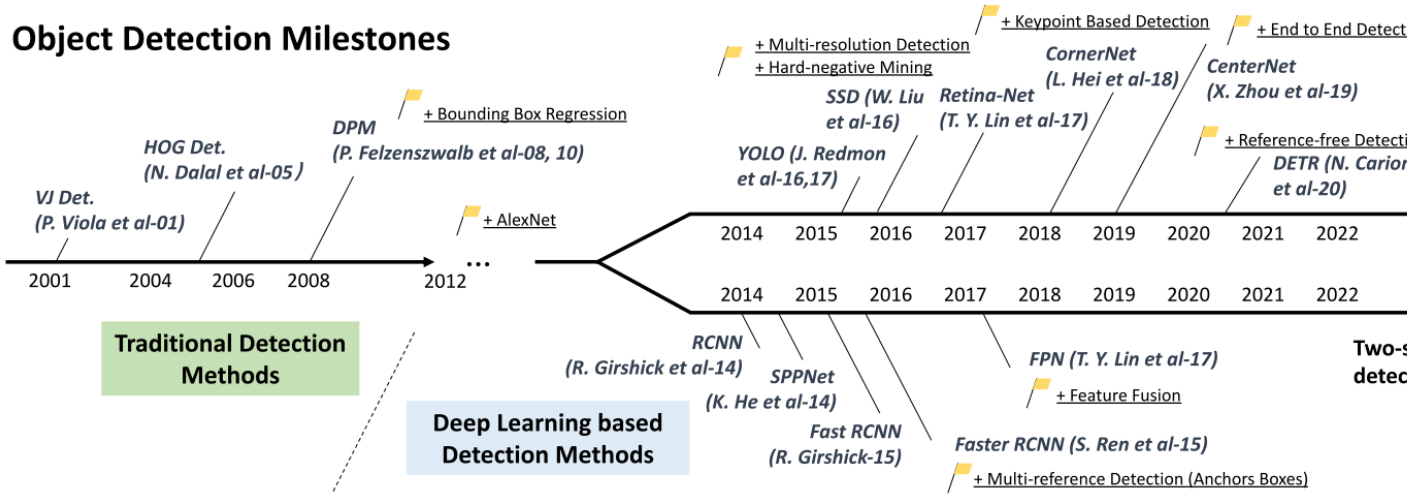

Figure 10: Object detection history (from Zou et al. (2023)).

Two-Stage Detectors

Two-stage detectors conceptually consist of two phases/models: 1) Finding regions of interest (ROIs), i.e., locations with possible objects, and 2) Classifying and refining the found ROIs.

R-CNN: Region-Based CNN

R-CNN (Regions with CNN Features) was published in 2014 Girshick et al. (2014). A generic region proposal method is applied to a given image. The idea behind region proposals is to find good candidates for bounding boxes (regions) that possibly contain an object. This would significantly reduce the effort to classify regions (compared to the sliding window).

A well-known algorithm for identifying possible objects is selective search (Uijlings et al. (2013)). This identifies object candidates based on regions with similar color, texture, or shape. Selective search can be run on the CPU and finds many, e.g., 2000 regions for an image in a few seconds. Figure 11 shows an example of applying this algorithm to an image.

Then, each of these regions is aligned in dimensionality (warping) so that all ROIs have the same spatial dimensions. This is necessary so that they can be processed with the same CNN (batch-wise). These warped ROIs are then individually classified with a CNN. Figure 12 illustrates the process.

Additionally, the region proposals are improved by learning a bounding box regression. Figure 13 shows the entire process.

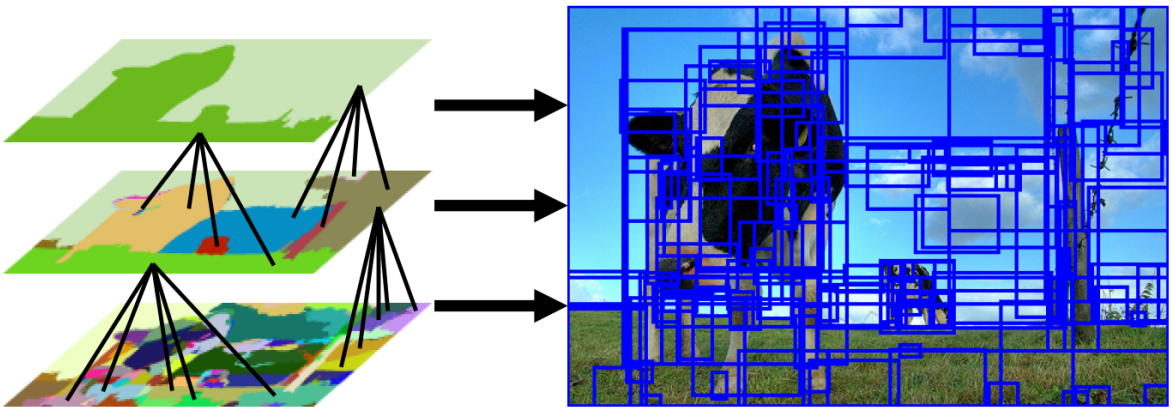

Figure 11: Left: Results of selective search (at different scales), right: Object hypotheses. Source: Uijlings et al. (2013).

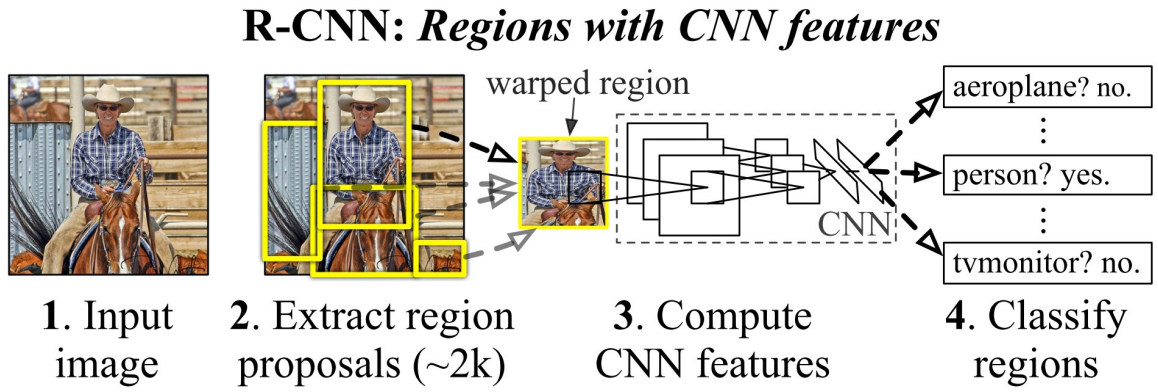

Figure 12: Source: Girshick et al. (2014).

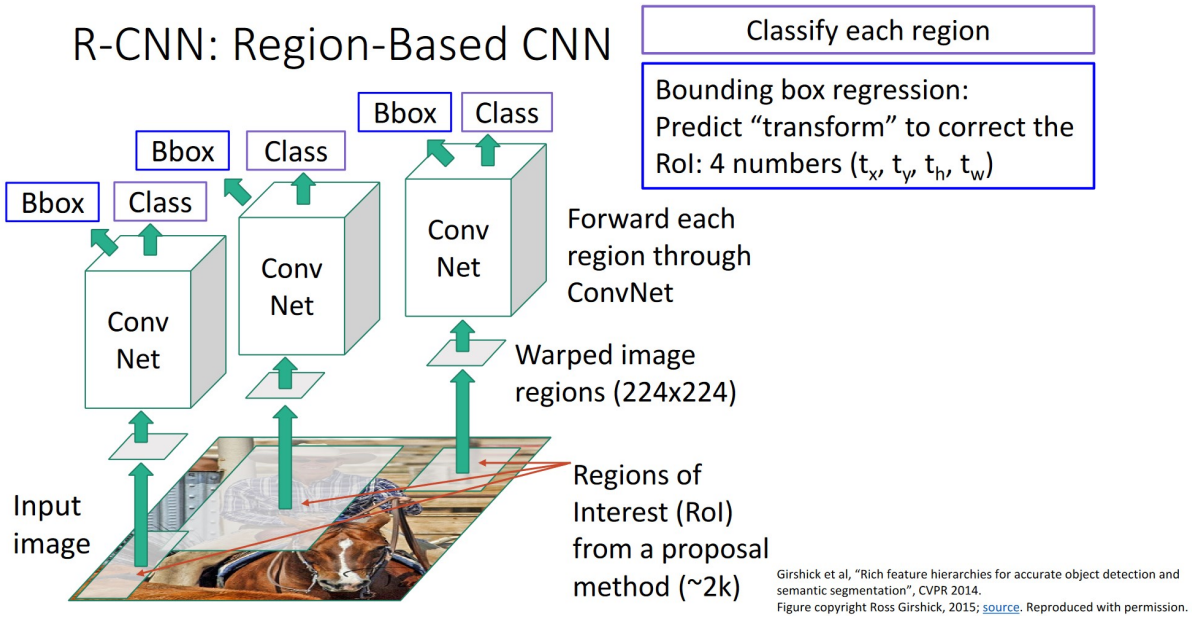

Figure 13: Source: Johnson (2019).

Note

Bounding box regression models a transformation of the ROIs. The transformation is parameterized by four numbers (t_x, t_y, t_h, t_w), just like the ROI proposals (p_x, p_y, p_h, p_w). The predicted bounding box is then (b_x, b_y, b_h, b_w). The position and extent are modeled as follows:

$$b_x = p_x + p_w t_x b_y = p_y + p_h t_y b_w = p_w \cdot e^{t_w} b_h = p_h \cdot e^{t_h} \tag{1}$$

The position of the box is modeled scale-invariant (relative to width/height). Width/height are modeled in log-space, so only valid values are possible (negative would not be possible).

The individual transformations t_* are modeled with a ridge regression, which uses the ROI features x as input, where i indexes individual proposals.

$$J(w) = \sum_i (t_*^i - w_*^T x_i)^2 + \lambda \|w_*\|^2 \tag{2}$$

Figure 14 illustrates bounding-box regression with an example.

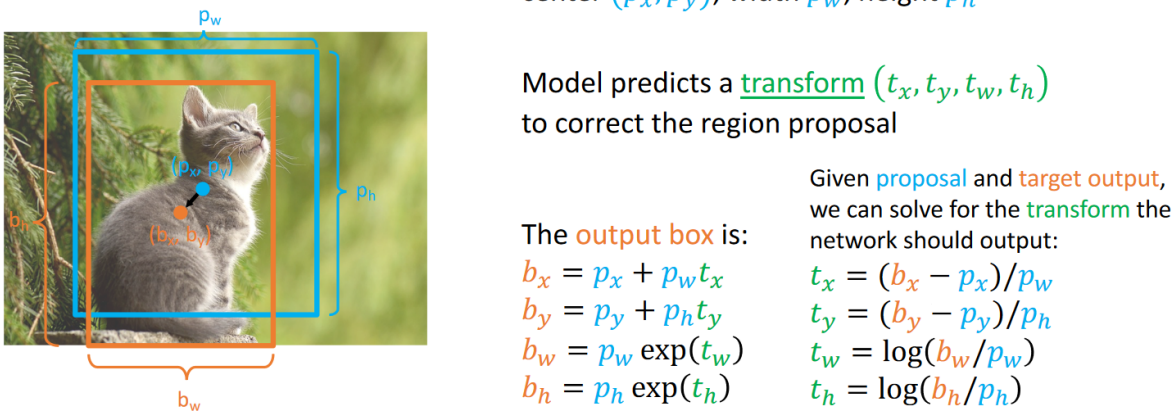

Figure 14: Source: Johnson (2022).

R-CNN optimizes cross-entropy for classification and least squares for bounding box coordinates.

Fast R-CNN

R-CNN is very slow because a forward pass through the CNN is required for each region proposal. The follow-up paper to R-CNN Girshick (2015) changed the method somewhat.

Instead of processing each region proposal separately, the entire image is processed once with a CNN (feature extraction) to obtain activation maps that are somewhat reduced in spatial dimension (see Figure 15).

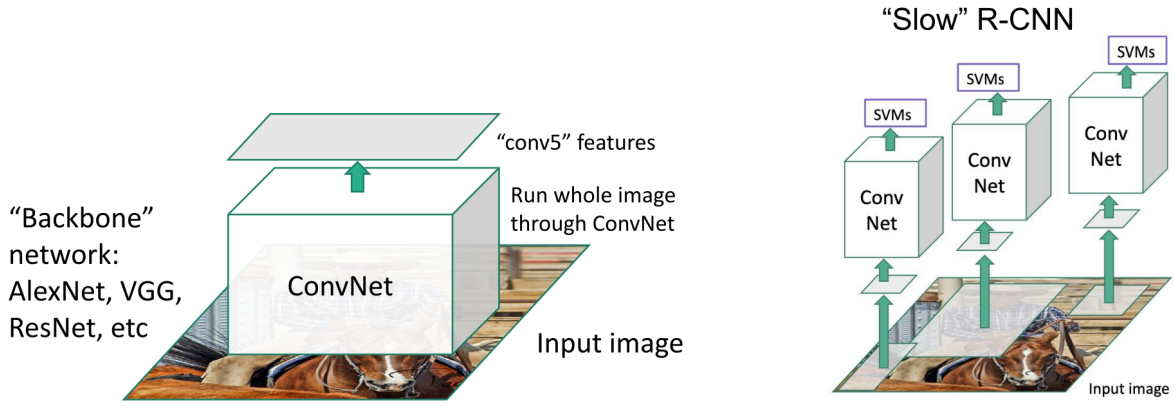

Figure 15: Source: Johnson (2019).

Then, the region proposals generated by a separate method (e.g., selective search) are projected onto the extracted activation maps (see Figure 16).

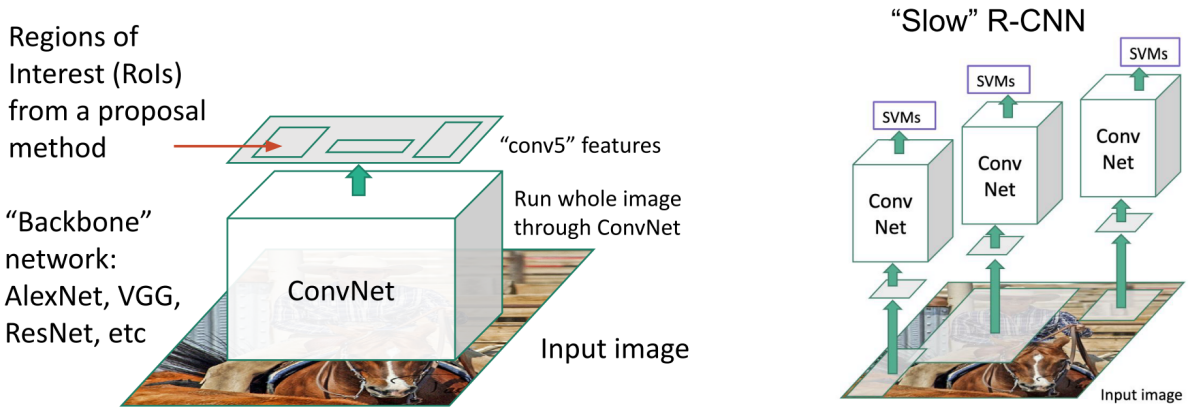

Figure 16: Source: Johnson (2022).

Next, the extracted features are warped and processed through a small network of region of interest pooling and fully connected layers (see Figure 17).

Finally, a classification and adjustment of the region of interest are output. Figure 18 shows the entire architecture and the losses. During model training, classification loss and bounding box regression loss can be calculated on these outputs.

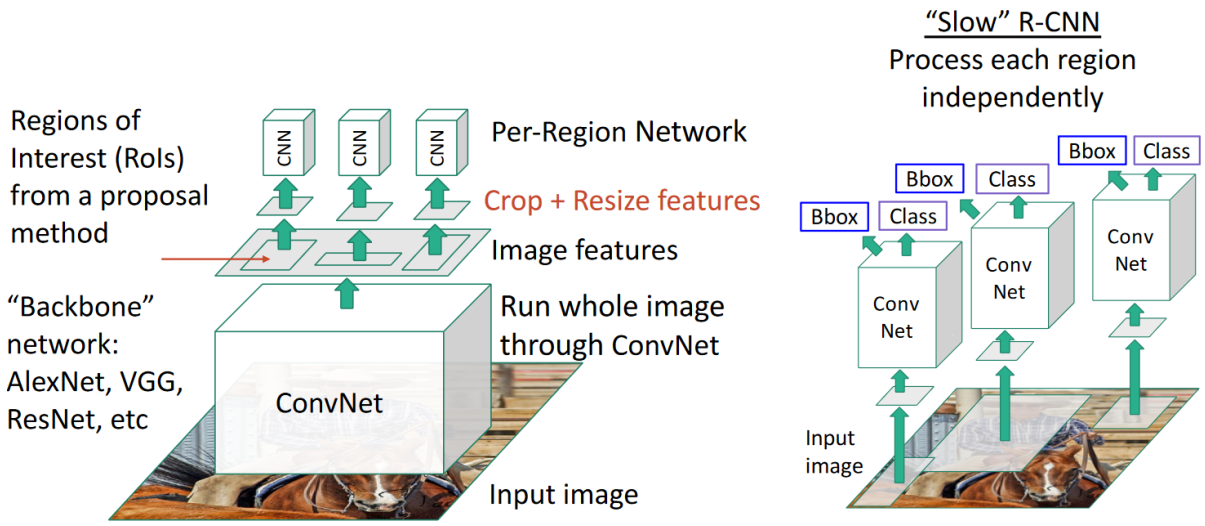

Figure 17: Source: Johnson (2022).

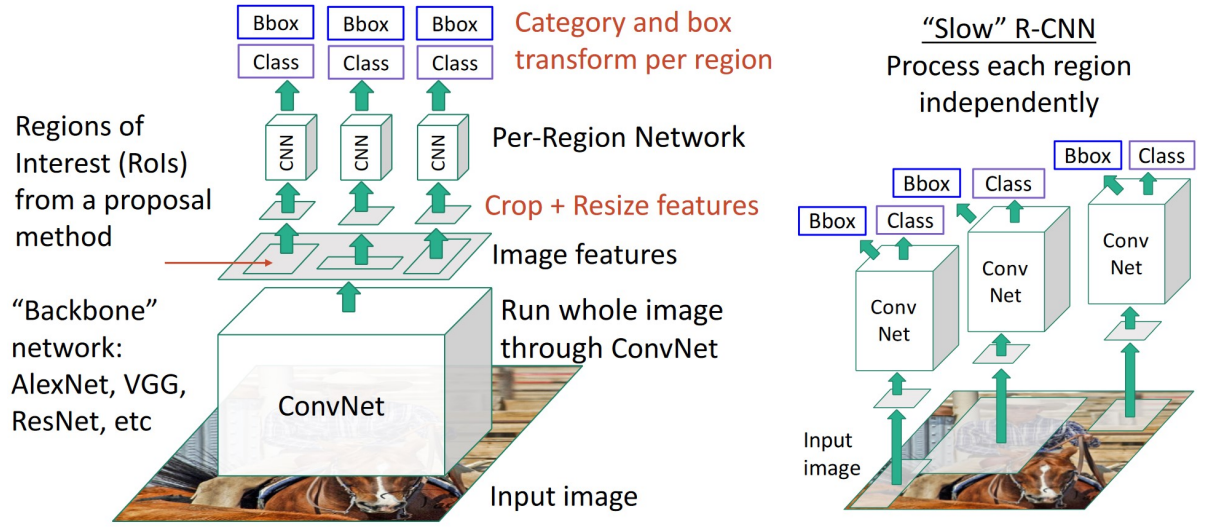

Figure 18: Source: Johnson (2019).

Figure 19 shows the architecture of Fast R-CNN with a ResNet backbone.

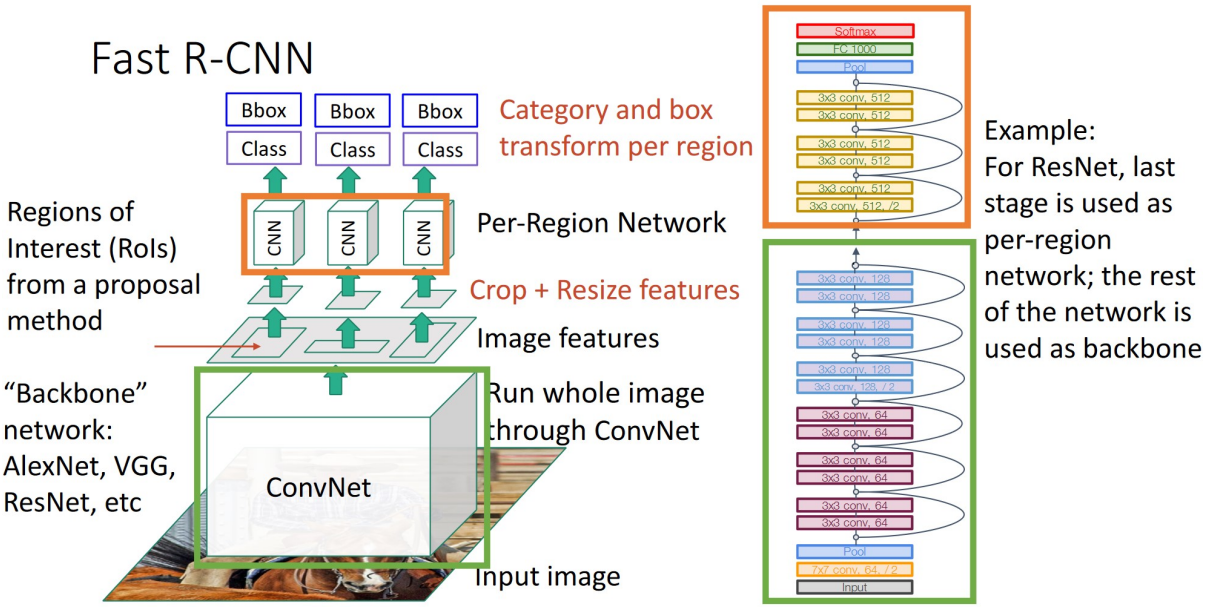

Figure 19: Source: Johnson (2019).

The region proposals from the proposal method must be projected onto the activation maps. An important innovation of Fast R-CNN was ROI pooling, see Figure 20. Here, the spatial dimension is reduced, for example, with max pooling so that all ROIs have the same dimensionality. This is necessary so that all can be further processed with the same network for classification and regression.

Figure 21 shows the training and test time for the model, respectively for a single image. In Fast-RCNN, the test time is dominated by the region proposals generated by a separate method.

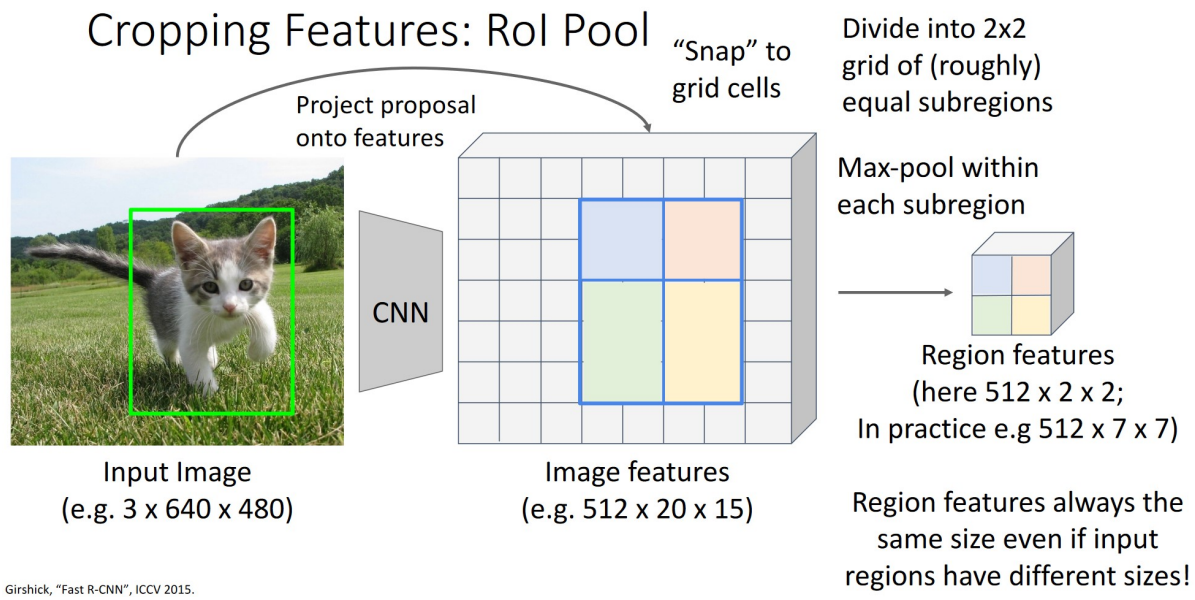

Figure 20: Source: Johnson (2019).

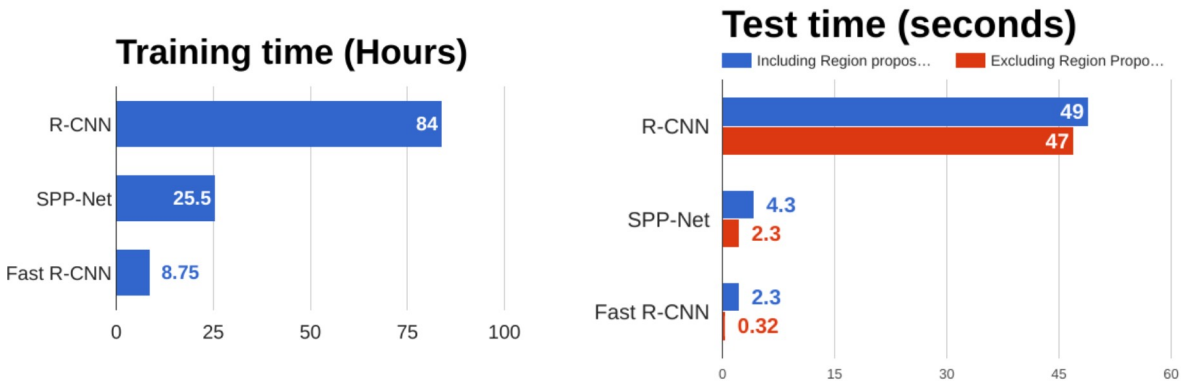

Figure 21: Source: Johnson (2019).

Note
The loss function of Fast R-CNN is as follows:

$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u > 1]L_{\text{loc}}(t^u, v) \tag{3}$$

where u represents the true class, p the modeled probability for u. t^u are the modeled bounding box coordinates (a tuple with 4 numbers) for the class u, and v are the true bounding box coordinates for the class u. $L_{\text{cls}}(p, u)$ is the cross-entropy loss. $L_{\text{loc}}(t^u, v)$ is only evaluated for non-background classes with $[u > 1]$, as there is no bounding box for the background class. $L_{\text{loc}}(t^u, v)$ is a slightly modified L_1 loss (absolute distance). For bounding-box regression, a smooth-L1 loss is used.

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L_1}(t_i^u - v_i), \tag{4}$$

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases} \tag{5}$$

Faster R-CNN

With Faster R-CNN Ren et al. (2017), the R-CNN family was further improved. In particular, the generation of region proposals was integrated into the method by creating them with a Region Proposal Network (RPN). In line with the end-to-end learning principle, the aim was to use as few heuristics as possible, such as selective search.

The rest of the architecture corresponds to Fast R-CNN. Figure 22 shows the architecture.

The RPN generates object proposals (bounding boxes) in a sliding window approach (implemented as a convolution) on the activation maps of the backbone CNN. These proposals are locations where an object is likely to be found. Figure 23 illustrates an example image (left) with dimensions $3 \times 640 \times 480$, the resolution of the activation map (right) with $512 \times 5 \times 4$ on which the RPN operates. Each point/grid cell represents the spatial coverage on the input image. It becomes apparent that the spatial resolution has been reduced by the CNN backbone (e.g., with pooling layers or convolutions with stride > 2). It is illustrated that the activation map (in this case) has 512 channels, i.e., complex and rich features that represent each region defined by the grid cells. As a comparison: In the paper by Ren et al. (2017), they write that an image with a spatial resolution of 1000×600 results in activation maps with 60×40 resolution.

The RPN now models whether an object is present at each point/for each grid cell and whether a correction of a reference bounding box is necessary. Figure 24 illustrates the reference box (blue) for one point. In this case, there is no object nearby. This reference box is also called an anchor.

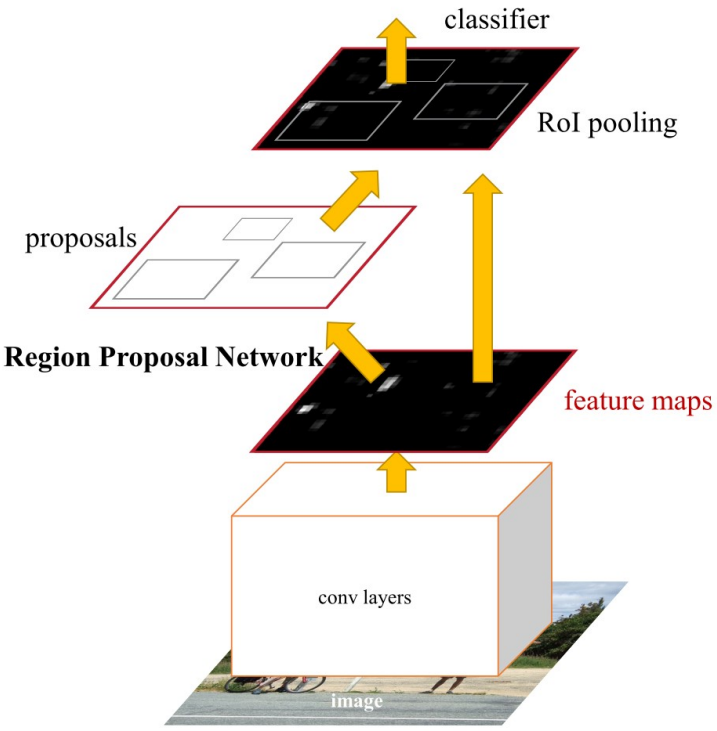

Figure 22: Source: Ren et al. (2017)

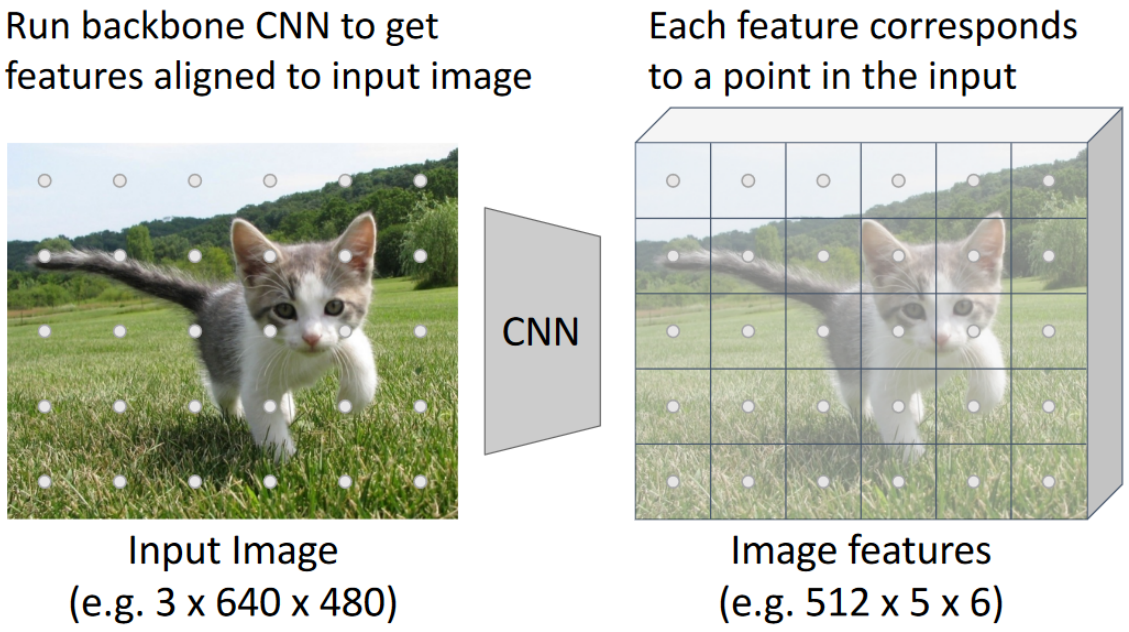

Figure 23: Source: Johnson (2022).

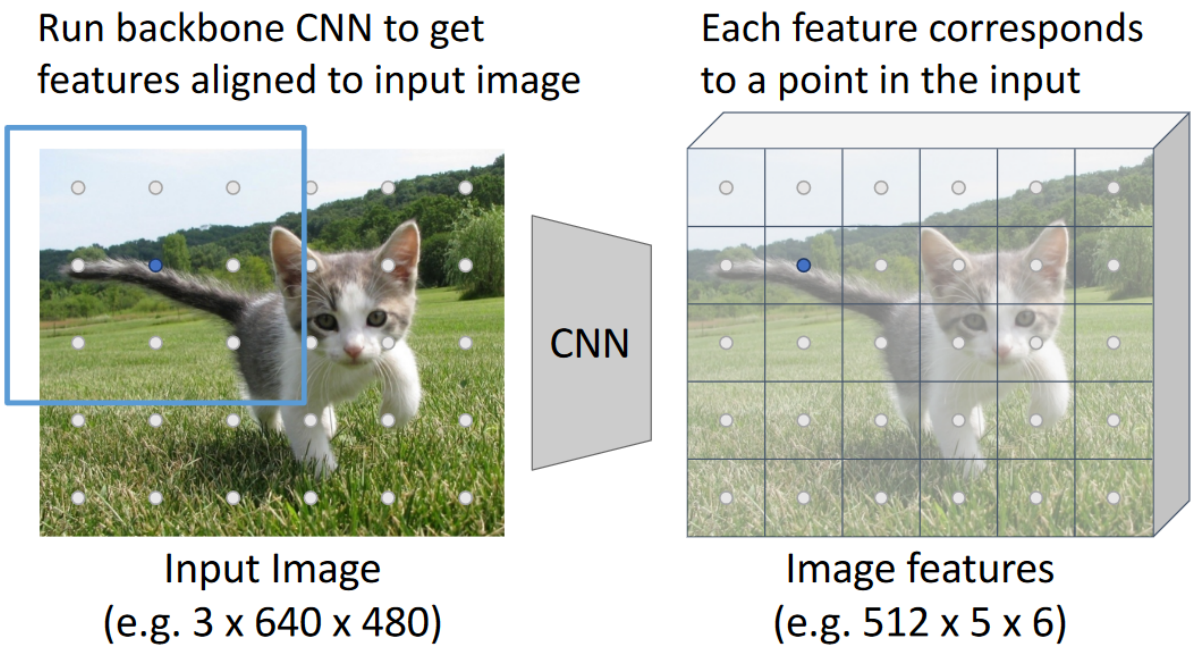

Figure 24: Source: Johnson (2022).

In Figure 25, you see a positive (green) anchor (with an object) and a negative (red) one without an object. The RPN models an objectness score that is high for positive anchors and low for negatives.

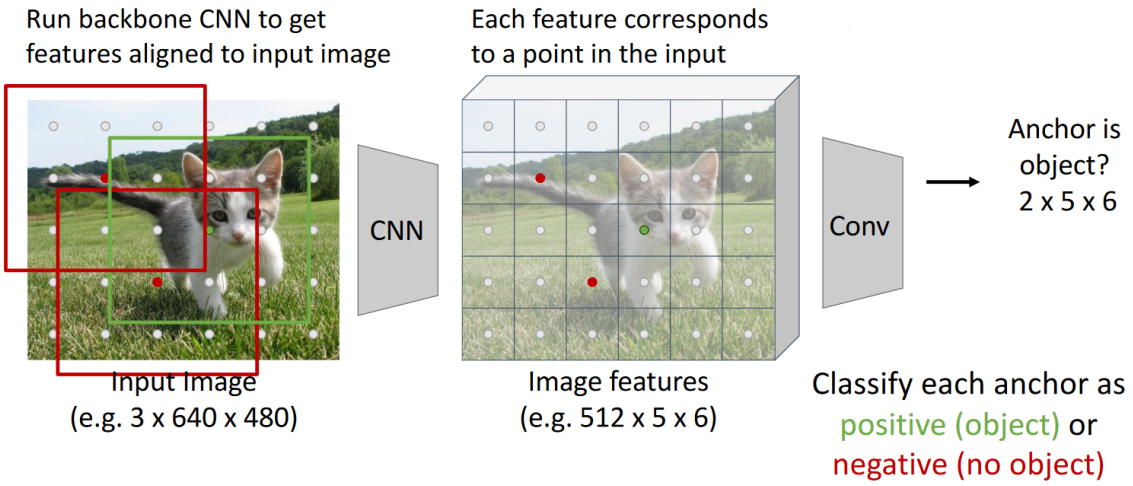

Figure 25: Source: Johnson (2022).

In addition to objectness scores, transformations for the anchors are also modeled (bounding box regression) so that they fully cover the object. During RPN training, transformations for the positive anchors (green box) are calculated/modelled relative to the ground truth box (orange).

> **Question:**
> What happens if 2 or more objects are in the same place?

If two or more objects are in the same place, often the objects overlapping have a different shape. Figure 27 illustrates two objects with almost the same center but with significantly different bounding boxes. If anchor boxes with different aspect ratios are defined, e.g., for long and tall/narrow objects, this problem can be partially circumvented.

Figure 28 shows that the RPN in Faster R-CNN models k anchors per location. This allows almost all possible objects, even those close to each other, to be assigned to an anchor

and detected.

Overall, Faster R-CNN is trained with 4 different losses, as seen in Figure 29.

Faster R-CNN is a two-stage detector because the RPN is conceptually separated from the final classification/bounding box regression. In particular, the found regions of the RPN must

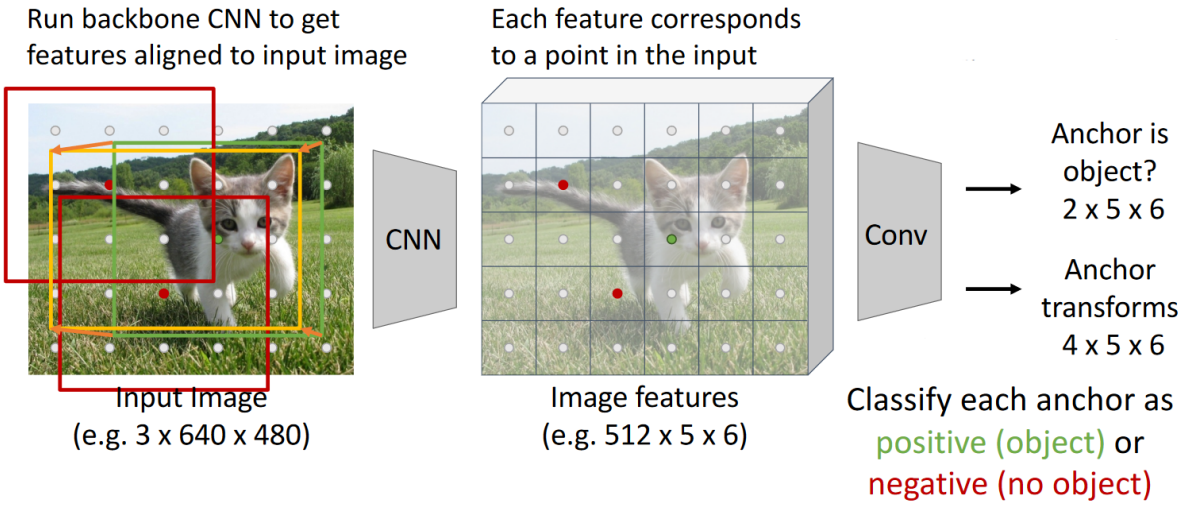

Figure 26: Source: Johnson (2022).

Figure 27: Anchor boxes.

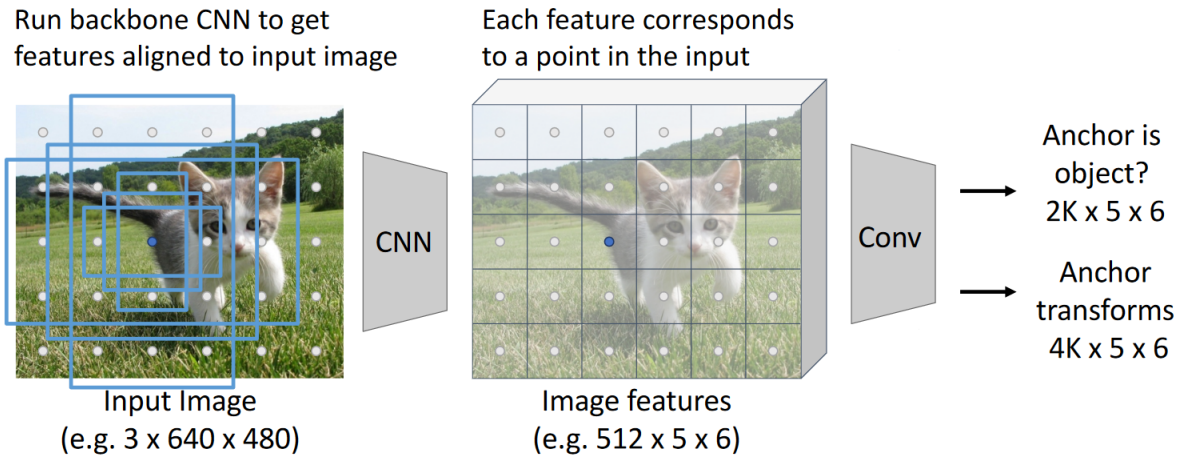

Figure 28: Source: Johnson (2022).

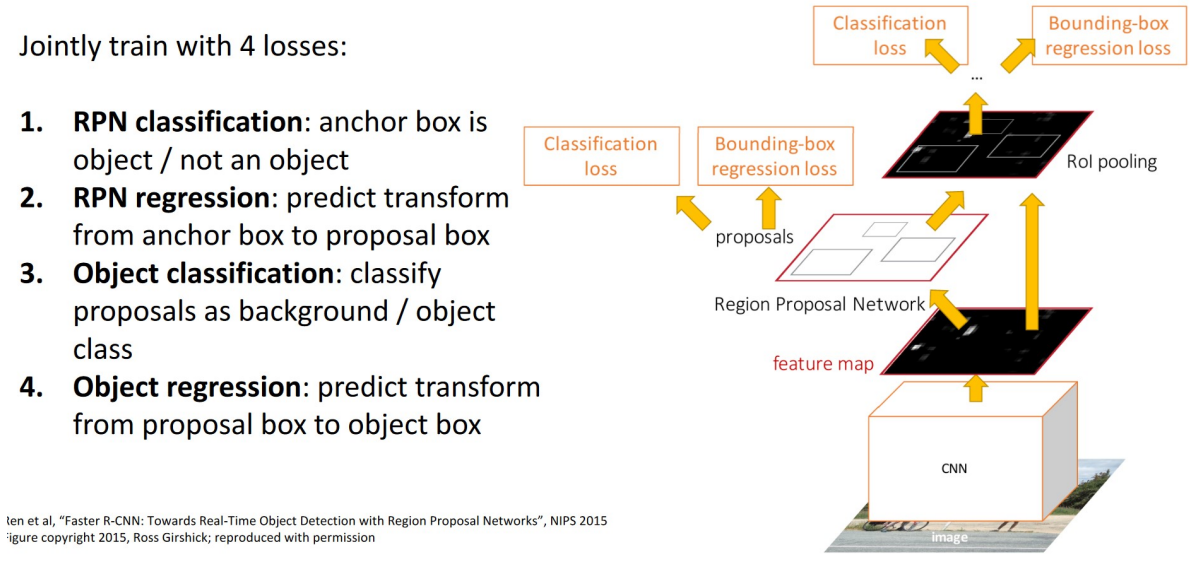

Figure 29: Source: Johnson (2019).

be warped and arranged in a batch of samples so they can then be processed through the second stage.

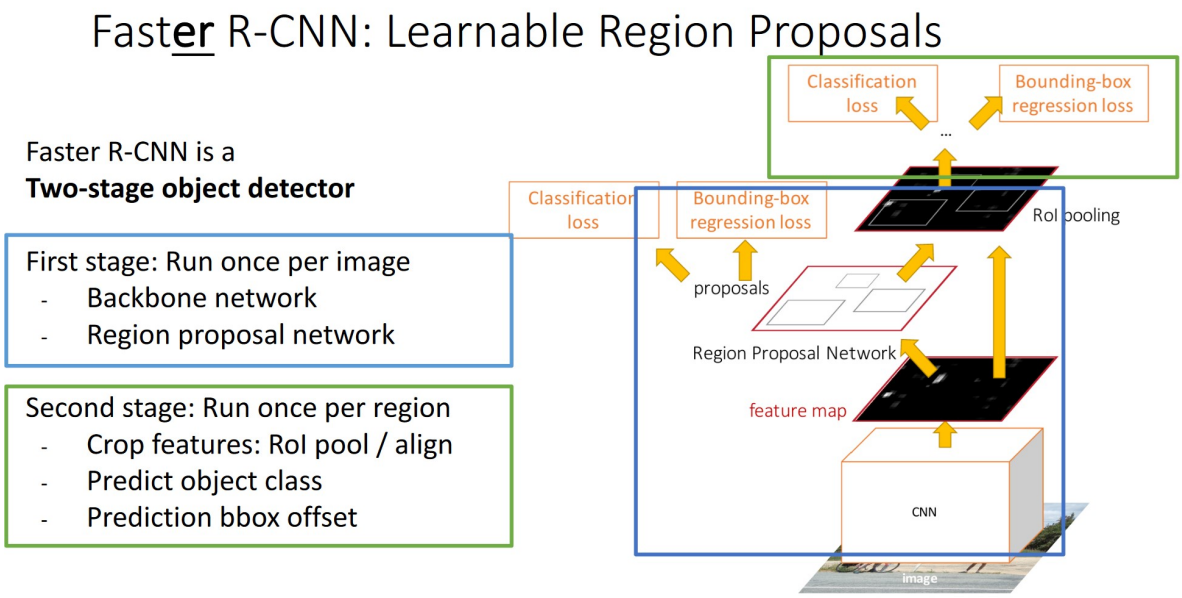

Figure 30: Source: Johnson (2019).

Single-Stage Detectors

Two-stage detectors process an image with a two-step approach: In the first step, the goal is to detect as many possible objects as possible, thus maximizing recall. In the second step, the detections are refined, focusing more on classification and distinguishing different objects. This approach achieves high accuracy with little effort. However, a problem is their slow inference speed and complexity due to the two stages. Single-stage detectors detect objects in one step, making them inherently more elegant and faster. Such models can therefore be used on mobile devices. Often, single-stage detectors have problems detecting small or closely spaced objects.

YOLO

A well-known representative is YOLO (You Only Look Once) Redmon et al. (2016a) and its variants. An image is reduced to the spatial dimensionality of SxS with a CNN (see Figure 31).

Then, a classification is performed for each grid cell (value on the activation map) (see Figure 32).

Figure 31: Source: Redmon et al. (2016a)

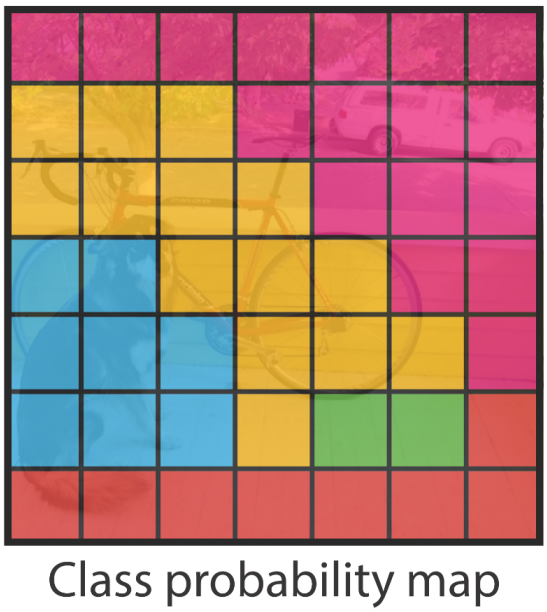

Figure 32: Source: Redmon et al. (2016a)

Finally, for each grid cell, the following values are modeled for each of the B bounding boxes: a bounding box regression (4 parameters) and an object score that models whether the center of an object is in the grid cell.

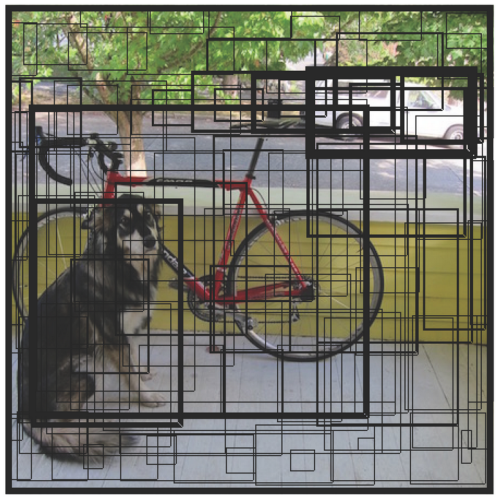

Figure 33: Source: Redmon et al. (2016a)

Then, the classification and bounding box regression are merged. Figure 34 shows the whole picture.

The full architecture of YOLO is shown in Figure 35. The output is a tensor of dimension $(C + 1)xKxHxW$ for the classification of detections and a tensor of $Cx4KxHxW$ for the bounding box regression, where C is the number of classes ($C + 1$ with included background class), K is the number of anchor boxes, and HxW is the spatial dimension of the output activation maps.

Since objects often occupy multiple grid cells, it may be that multiple cells detect the same object. Figure 36 shows an example where two cells detect the dog and create two bounding boxes accordingly. This is problematic because exactly one bounding box is needed per detection, and no duplicates.

With non-max suppression, such duplicates can be avoided. Figure 37 shows the effect of NMS on our example image. More on NMS in {ref}`non-max-suppression`.

Another difficulty is detecting objects that have their center in the same grid cell. Therefore, YOLO uses B bounding boxes per cell. This allows B objects per cell to be detected, as illustrated in Figure 38. A better variant is to work with anchor boxes.

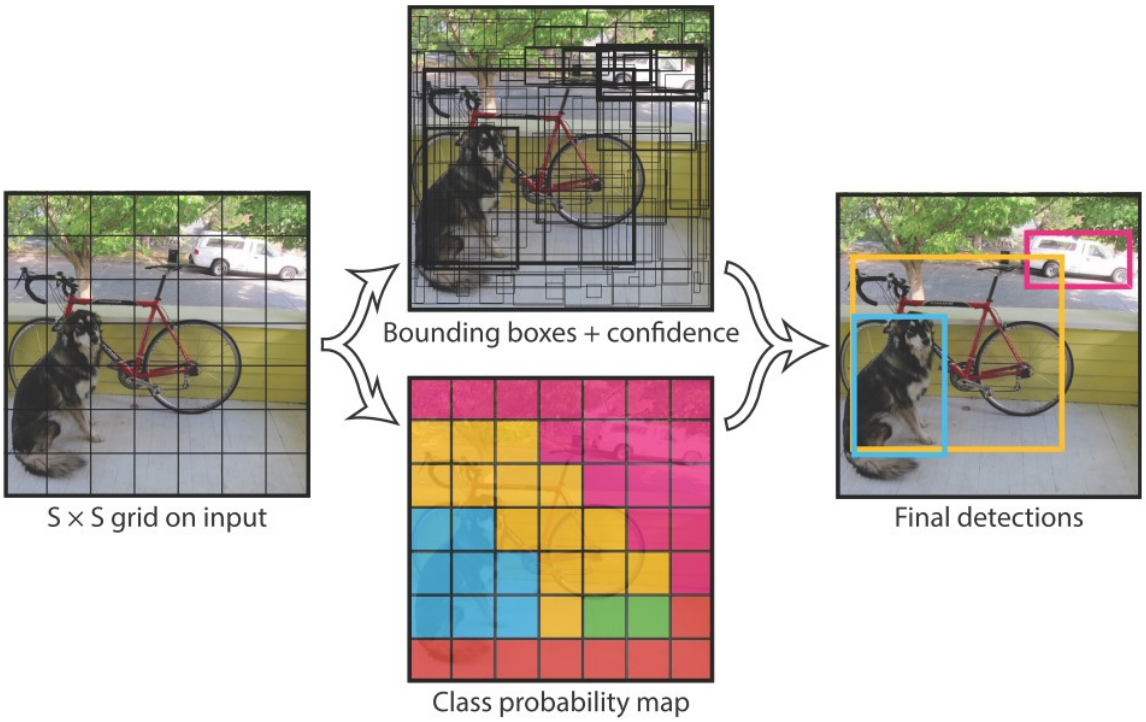

Figure 34: Source: Redmon et al. (2016a)

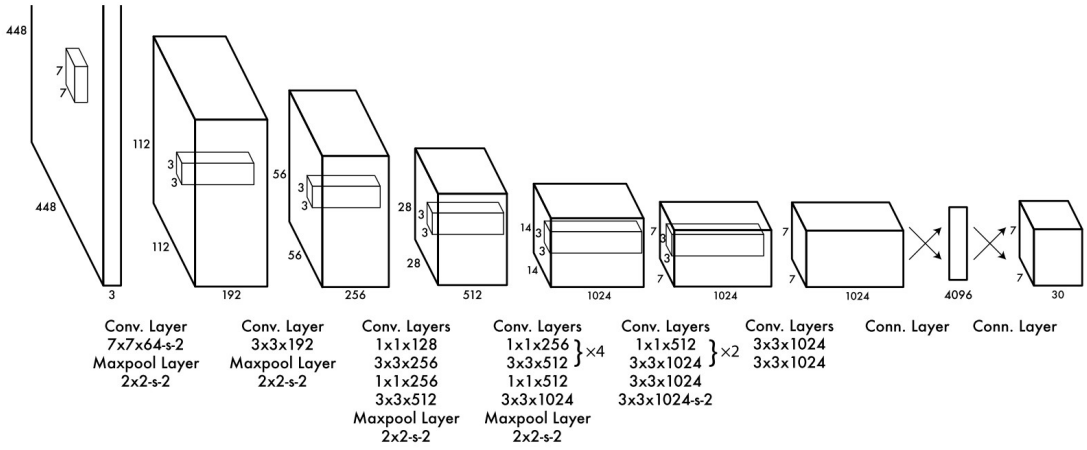

Figure 35: Source: Redmon et al. (2016a)

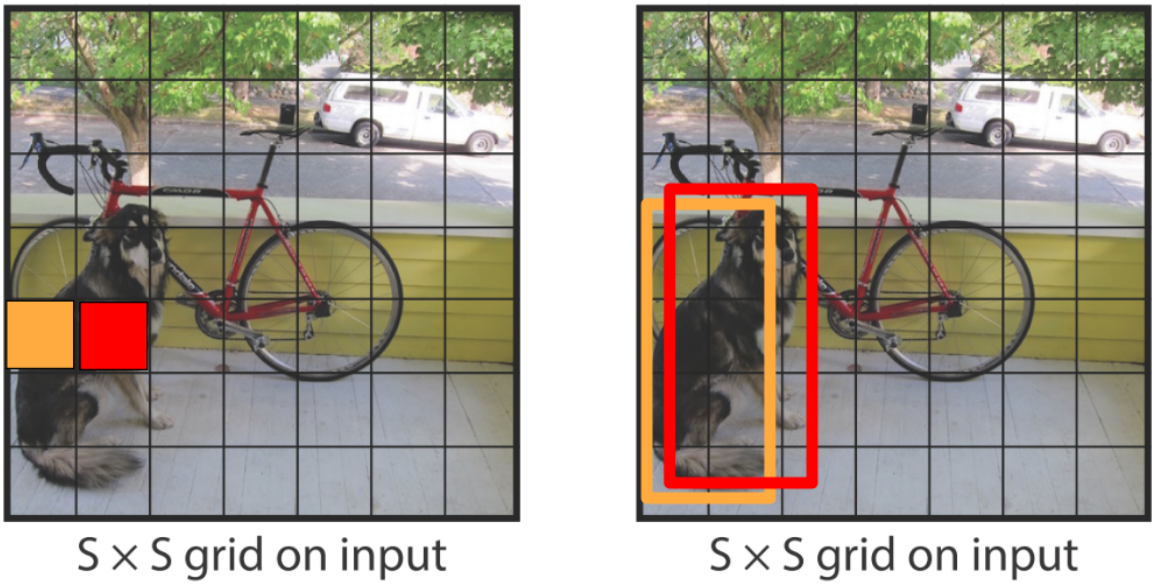

Figure 36: Inspired by Austin et al. (2022)

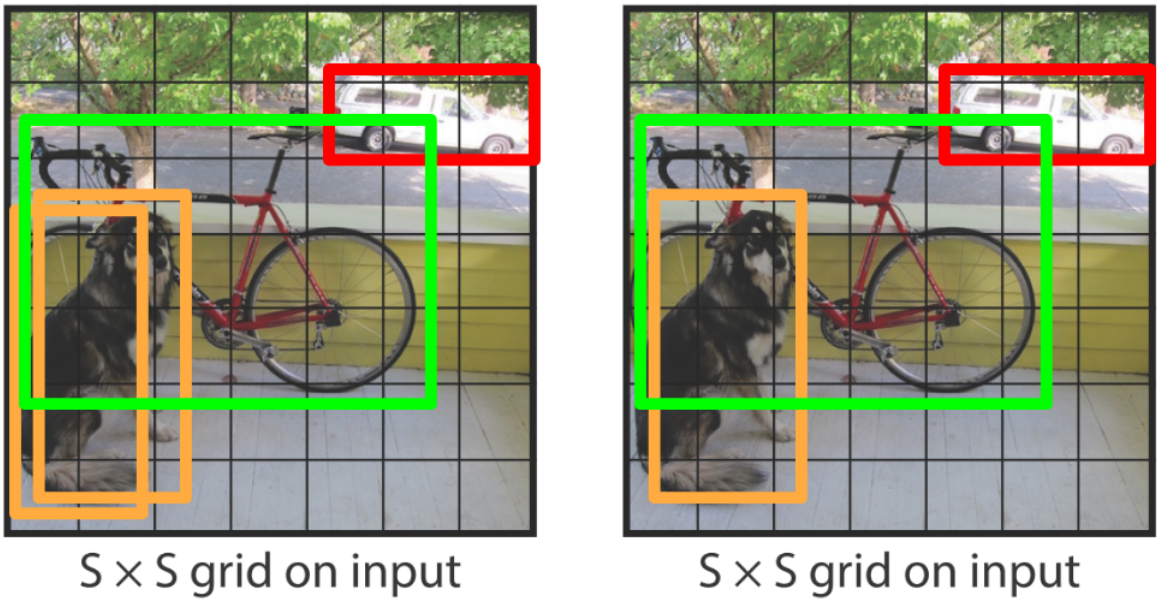

Figure 37: Inspired by Austin et al. (2022)

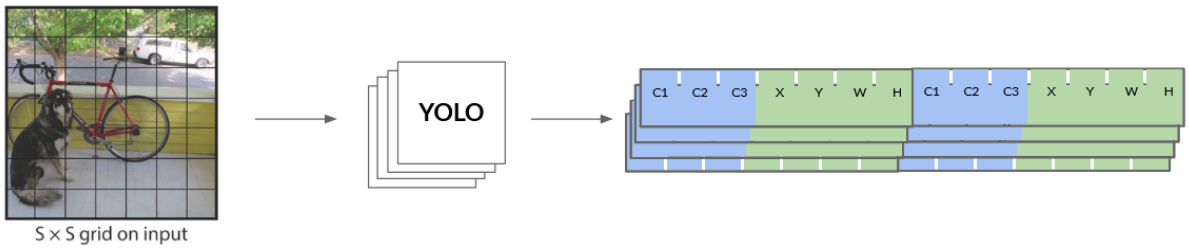

Figure 38: Source: Austin et al. (2022)

Note

The cost function of YOLO is shown below. λ weights are for the different terms, and $\mathbb{1}$ turns certain loss terms on and off, depending on whether an object is present or not. $\hat{C}_i$ models the IoU for the predicted box, and $\hat{p}_i(c)$ the presence of class c in a grid cell. It is interesting that the classification part is penalized with a squared error and not, for example, with a cross-entropy loss.

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \tag{6}$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2 \tag{7}$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \tag{8}$$

YOLO did not work with anchors. More modern versions of single-stage detectors sometimes use anchors, just like many two-stage detectors. This allows a single-stage detector to directly output a tensor of dimensions $(C+1)xKxHxW$ for classification of detections and a tensor of $Cx4KxHxW$ for bounding box regression, where C is the number of classes ($C+1$ with included background class), K is the number of anchor boxes, and HxW is the spatial dimension of the output activation maps. Figure 39 illustrates single-shot detection.

CenterNet - Objects as Points

A modern representative of single-shot detectors is CenterNet Zhou, Wang, and Krähenbühl (2019). CenterNet divides an image into a finer grid compared to YOLO. The global stride is about 4, meaning the spatial resolution of the grid is 4 times smaller than that of the image. CenterNet assigns each object, according to its center, a grid point. Then, all object

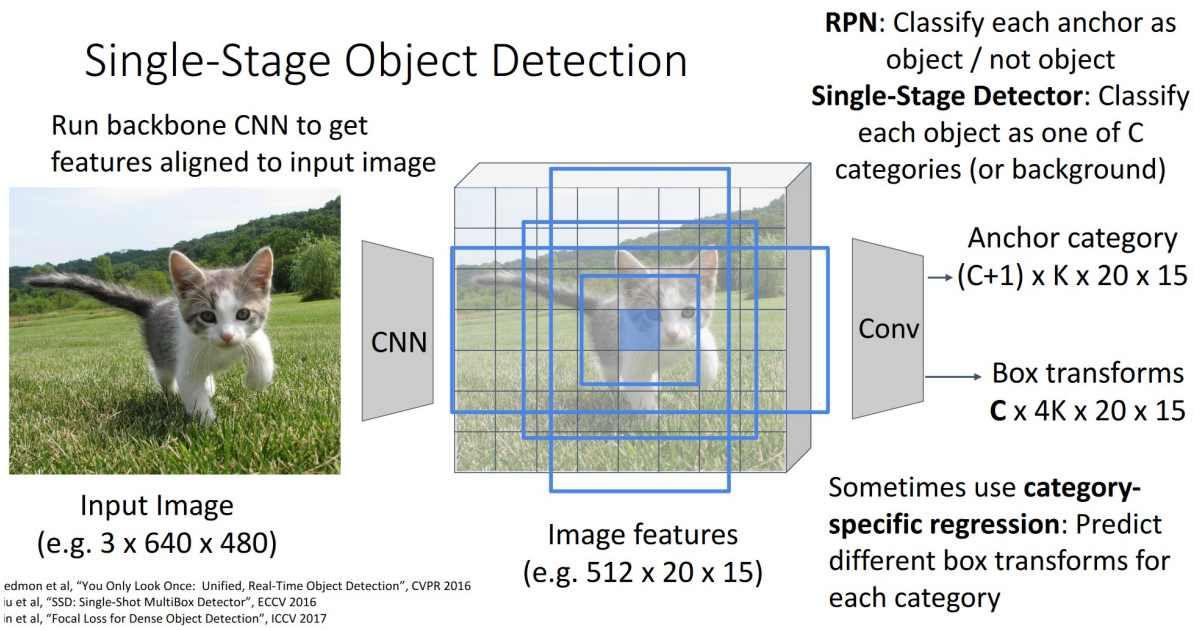

Figure 39: Source: Johnson (2019).

properties, such as the bounding box coordinates, are modeled based on the features from the center. Figure 40 illustrates various objects, their centers, and their height and width, which are modeled.

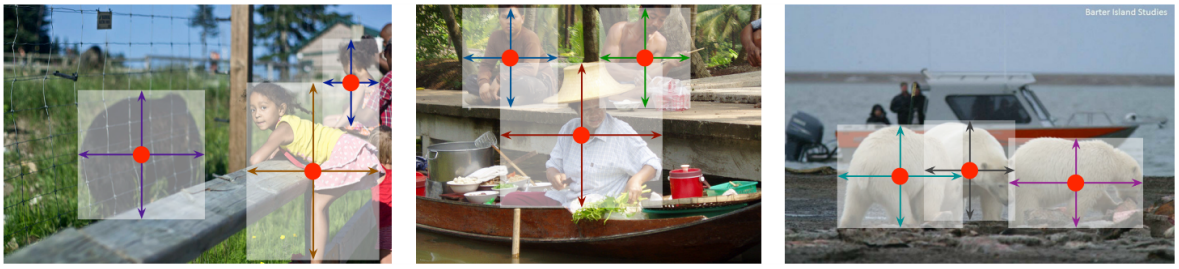

Figure 40: Source: Zhou, Wang, and Krähenbühl (2019).

The output of CenterNet is a keypoint heatmap $\hat{Y} \in [0,1]^{\frac{W}{R} \times \frac{H}{R} \times C}$, where H, W are the spatial resolution of the image, R the global stride, and C the number of object classes to be detected. $\hat{Y}_{x,y,c} = 1$ corresponds to a keypoint (center of an object in object detection), $\hat{Y}_{x,y,c} = 0$ corresponds to the background. Additionally, an offset is modeled: the deviation of the object center from the center of the grid cell: $\hat{O} \in \mathbb{R}^{\frac{W}{R} \times \frac{H}{R} \times 2}$, and the bounding box coordinates (height/width): $\hat{S} \in \mathbb{R}^{\frac{W}{R} \times \frac{H}{R} \times 2}$.

Figure 41 illustrates the keypoint heatmap, the offset prediction, and the object size with an

example.

Figure 41: Source: Zhou, Wang, and Krähenbühl (2019).

Figure 42 contrasts anchor-based methods with center points. Anchors are divided into positive (green) and negative (red), or ignored (gray), depending on the overlap with ground truth objects. These are then used in model training. CenterNet does not use anchors and thus does not need manually selected positive and negative anchors. Non-max suppression is also unnecessary. This results in fewer manual hyperparameters and heuristics.

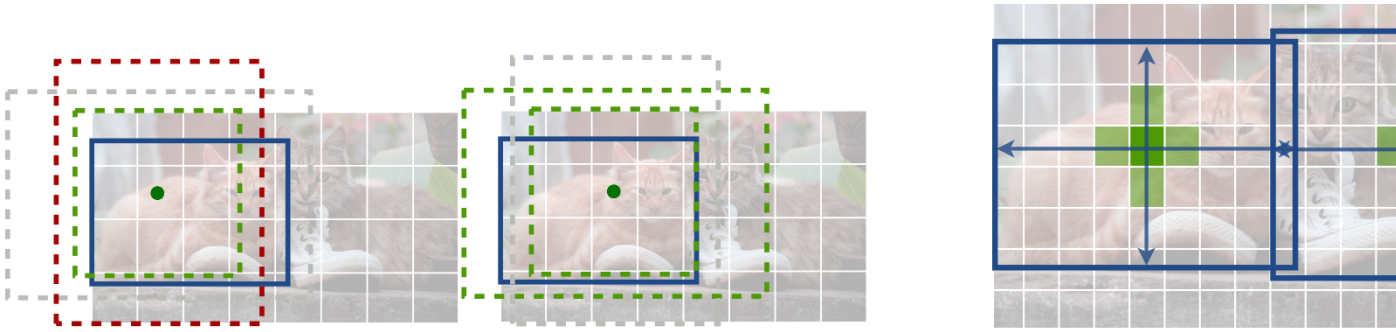

Figure 42: Source: Zhou, Wang, and Krähenbühl (2019).

Question
What could be an inherent limitation of CenterNet?

Further Aspects

There are many architectures and tricks used when training object detection models. Some of these are listed below.

Class Imbalance

An important topic is class imbalance when training models: Often, there is much more background than object classes (e.g., a ratio of 1:1000). This can lead to problems during learning, as the model may have a strong bias towards the background class, and the gradient during model training may be dominated by simple predictions (for the background class). In this context, the **focal loss** is an important milestone Lin et al. (2018). This reduces the loss for simple samples and increases the relative loss for difficult samples. Figure 43 shows the effect of focal loss (compared to cross-entropy) for different values of the parameter γ, which regulates the strength of the focal loss.

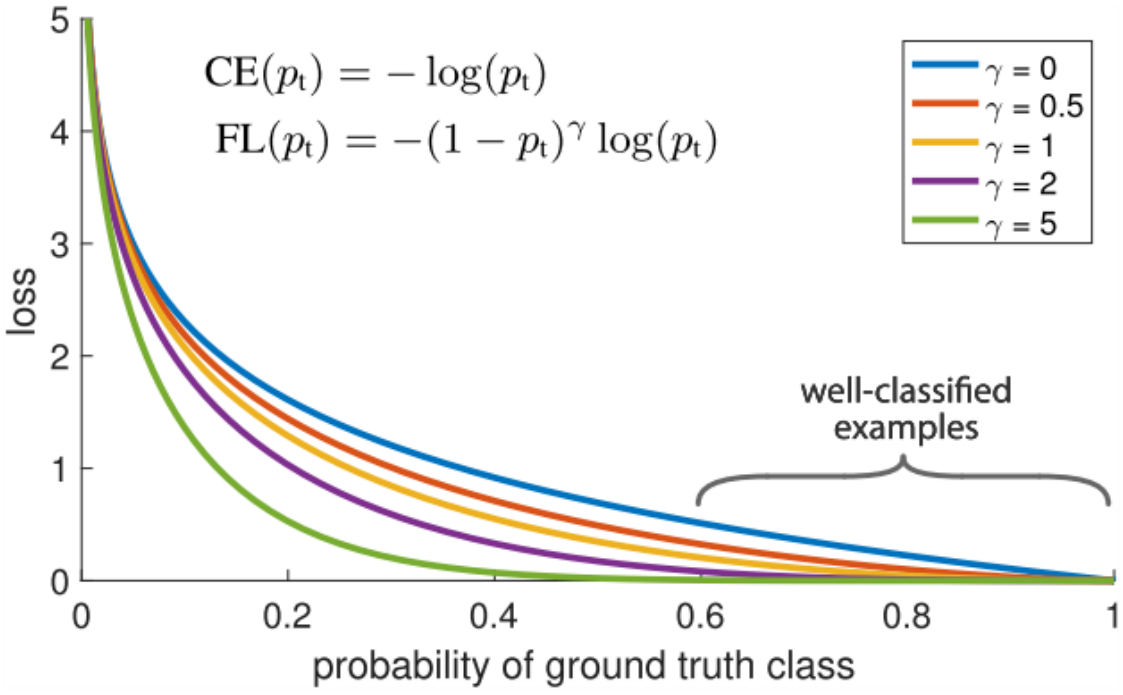

Figure 43: Source: Lin et al. (2018).

Feature Pyramid Networks

One challenge in object detection is the different spatial scaling of various objects. Both relatively small and relatively large objects need to be detected. Global features that provide important contextual information are helpful for classifying objects. Additionally, fine (pixel-accurate), more local features are important for accurately modeling bounding boxes. An innovation is feature pyramid networks (FPNs) Lin et al. (2017). This approach laterally combines and aggregates features from different layers. This allows global and local information to be combined. Additionally, it is possible to model objects of different sizes on different layers in the network. Figure 44 shows how features are increasingly condensed (left) as the spatial resolution decreases. You can see (right) that global information flows back to deeper levels, allowing smaller objects to be better classified with global information.

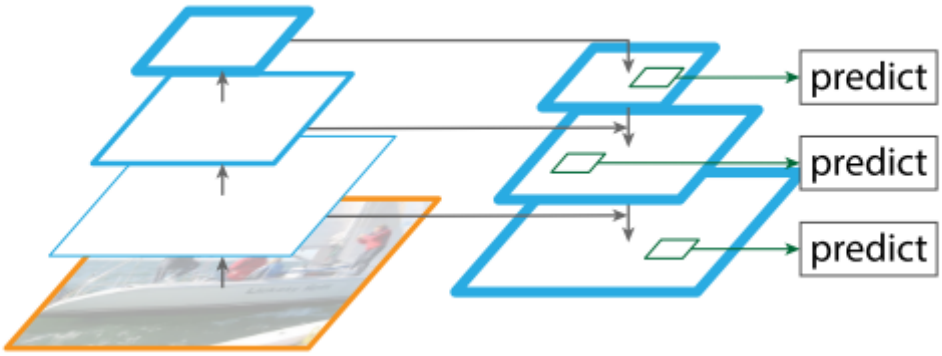

Figure 44: Source: Lin et al. (2017).

Transformers

A newer architecture is the transformer. This was successfully used in natural language processing (NLP) and has also taken an important place in object detection. With a transformer, object detection can be reformulated as a set-prediction problem Carion et al. (2020). This allows object detection to be trained end-to-end, and all objects can be detected in a single forward pass. This makes hand-designed features like anchor boxes or heuristics like non-max suppression unnecessary. Many state-of-the-art models are now based on the transformer architecture, although CNN-based models are still well represented.

Evaluation

The following describes how object detection models are evaluated.

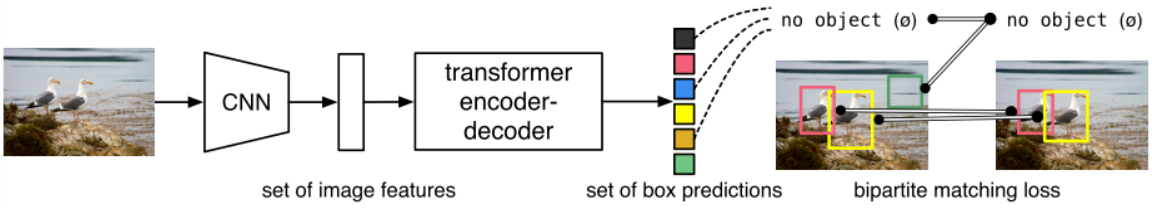

Figure 45: Source: Carion et al. (2020).

Intersection over Union (IoU)

Intersection over Union (IoU) is a metric to compare two bounding boxes. Figure 46, Figure 47, and Figure 48 illustrate the concept. An IoU > 0.5 is usually considered just acceptable.

Non-Max Suppression

In many methods, such as Faster R-CNN, the same objects can be detected multiple times. Therefore, potential duplicates must be eliminated during test time (inference or applying the model to an image). In practice, such cases are often resolved with non-max suppression (NMS).

Then, using the following heuristic (NMS), duplicates can be removed:

1. Select the box with the highest score (probability of a certain class (excluding background))
2. Eliminate boxes with a lower score that have an IoU $> \epsilon$, where ϵ is an arbitrary threshold (e.g., 0.7).
3. Repeat until no overlapping boxes remain.

Figure 49 illustrates NMS with an example:

This becomes problematic when many objects are densely packed or there is a lot of overlap, as shown in Figure 50. Here, many valid objects would be removed.

Mean Average Precision (mAP)

Evaluating object detection models is not easy. The most commonly used metric is mean average precision (mAP).

The following metrics are important for understanding mAP. These are based on the confusion matrix, which can be created for all classes. Figure 51 shows a confusion matrix.

Precision is the proportion of positively classified samples that are actually positive:

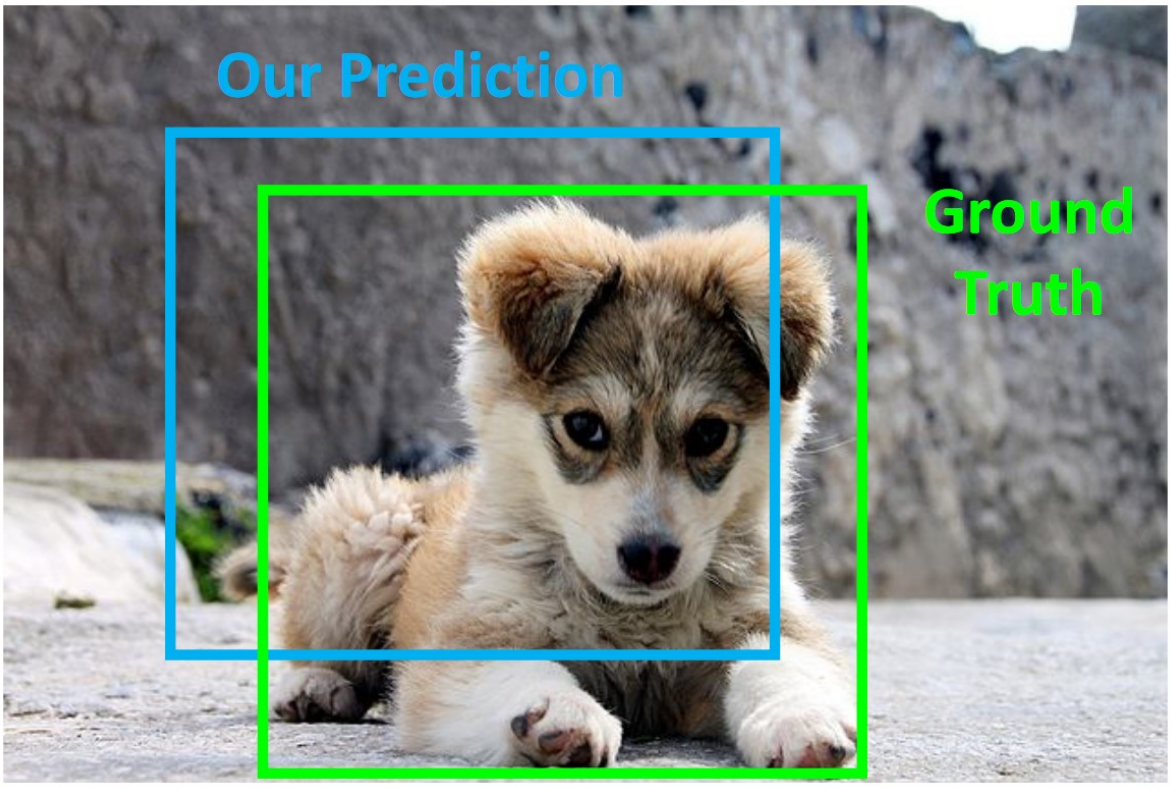

Figure 46: Source: Johnson (2022).

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU)
(Also called "Jaccard similarity" or "Jaccard index"):

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$

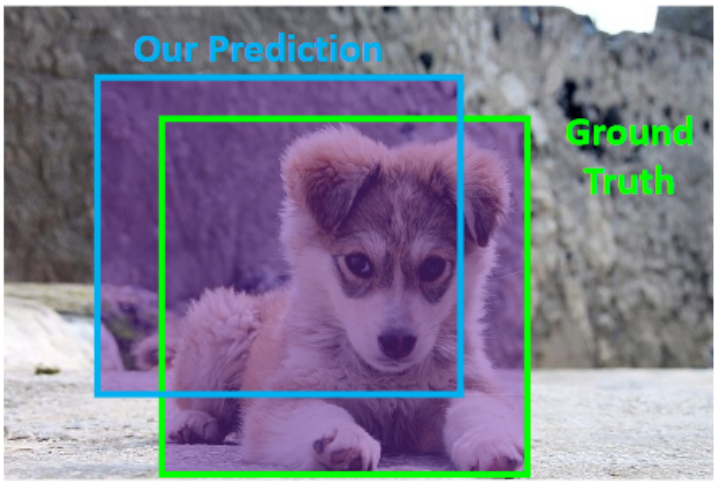

Figure 47: Source: Johnson (2022).

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU)
(Also called "Jaccard similarity" or "Jaccard index"):

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$

IoU > 0.5 is "decent",
IoU > 0.7 is "pretty good",
IoU > 0.9 is "almost perfect"

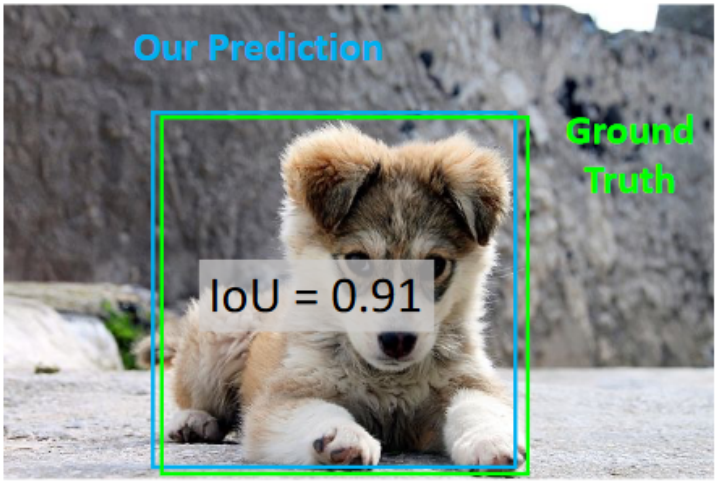

Figure 48: Source: Johnson (2022).

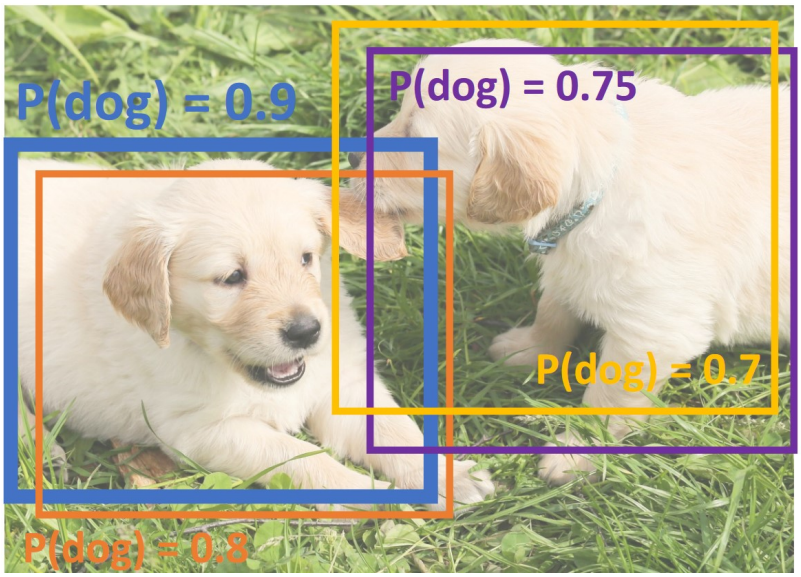

Figure 49: Source: Johnson (2019).

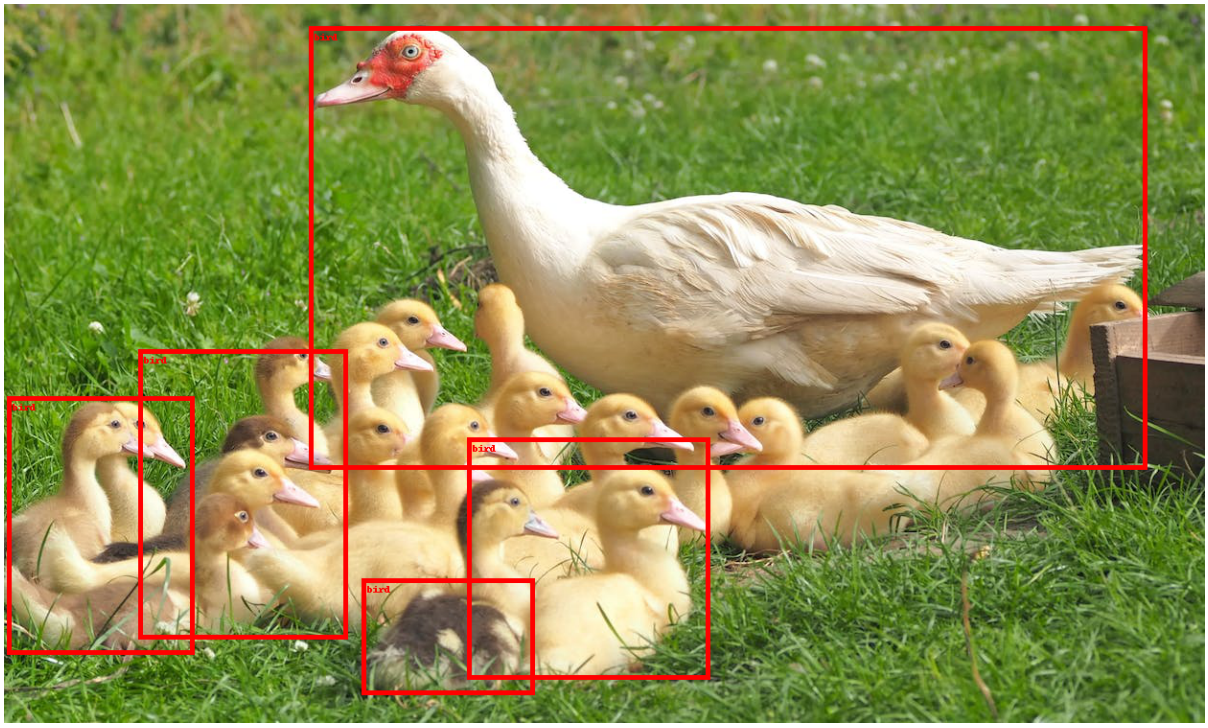

Figure 50: Source

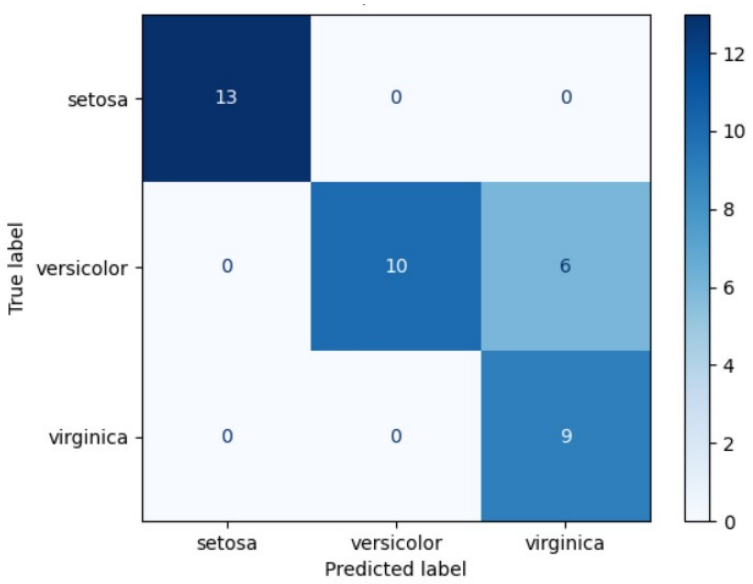

Figure 51: Source

$$\text{Precision} = \frac{TP}{TP + FP} \tag{9}$$

Recall is the proportion of positive samples that were correctly identified as such:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{10}$$

Average precision is the area under the precision/recall curve for a particular class. All detections of this class are sorted in descending order of their confidence (class score), and precision and recall are calculated after each sample. It is determined how many detections were correct and had a minimum IoU with a ground-truth box. These points can then be plotted. Figure 52 shows the calculation of average precision with an example.

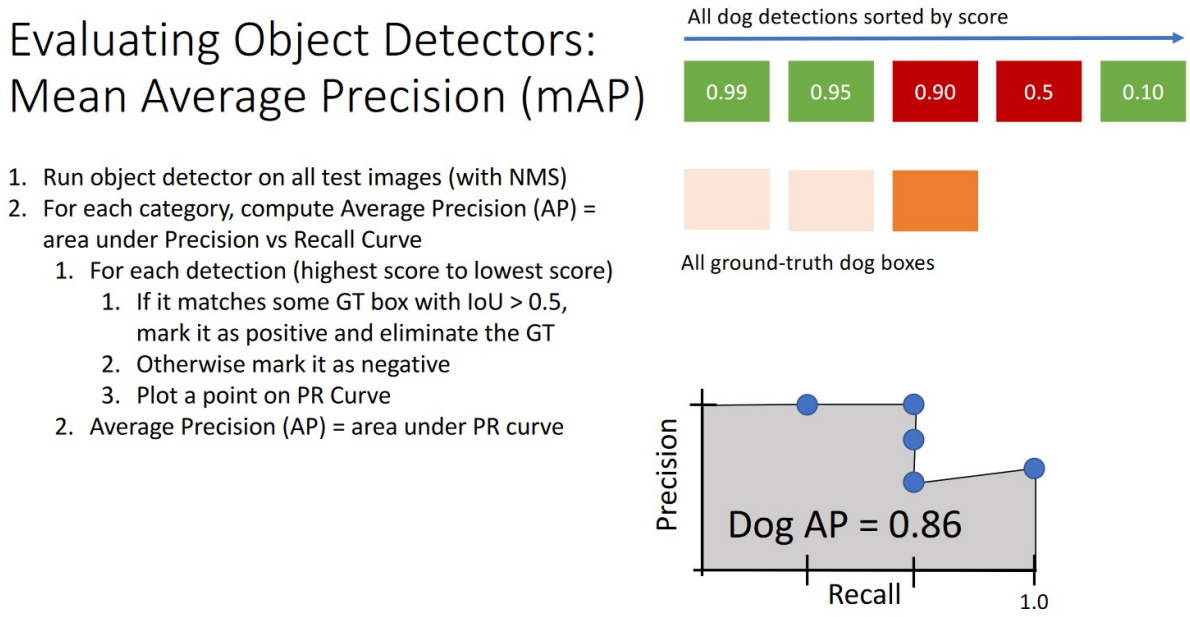

Figure 52: Source: Johnson (2019).

Mean average precision (mAP) is the average of all average precisions across all classes. Sometimes, an average over different IoU thresholds is also calculated, which a model must achieve for a hit.

More details can be found in this blog: Link

PyTorch

There are various ways to apply object detection in PyTorch. It is recommended to use an object detection framework.

An example is Detectron2. There are pre-trained models that can be used directly or adapted to your dataset.

Object detection can also be performed with torchvision.

References

Austin, Jake, Arvind Rajaraman, Aryan Jain, Rohan Viswanathan, Ryan Alameddine, and Verona Teo. 2022. "Modern Computer Vision and Deep Learning (CS 198-126)." Lecture {Notes} / {Slides}. https://fluff-armadillo-037.notion.site/Modern-Computer-Vision-and-Deep-Learning-CS-198-126-b11006739378470fa67a9cf6594201e0.

Carion, Nicolas, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. "End-to-End Object Detection with Transformers." arXiv. http://arxiv.org/abs/2005.12872.

Girshick, Ross. 2015. "Fast R-CNN." arXiv. http://arxiv.org/abs/1504.08083.

Girshick, Ross, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation." arXiv. http://arxiv.org/abs/1311.2524.

Johnson, Justin. 2019. "EECS 498-007 / 598-005: Deep Learning for Computer Vision." Lecture {Notes} / {Slides}. https://web.eecs.umich.edu/~justincj/teaching/eecs498/FA2019/.

———. 2022. "EECS 498.008 / 598.008 Deep Learning for Computer Vision." Lecture {Notes} / {Slides}. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/.

Lin, Tsung-Yi, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. 2017. "Feature Pyramid Networks for Object Detection." arXiv. http://arxiv.org/abs/1612.03144.

Lin, Tsung-Yi, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2018. "Focal Loss for Dense Object Detection." arXiv. http://arxiv.org/abs/1708.02002.

Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016a. "You Only Look Once: Unified, Real-Time Object Detection." *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2016-Decem: 779–88. https://doi.org/10.1109/CVPR.2016.91.

———. 2016b. "You Only Look Once: Unified, Real-Time Object Detection." arXiv. http://arxiv.org/abs/1506.02640.

Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. 2017. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (6): 1137–49. https://doi.org/10.1109/TPAMI.2016.2577031.

Uijlings, J. R. R., K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. 2013. "Selective Search for Object Recognition." *International Journal of Computer Vision* 104 (2): 154–71. https://doi.org/10.1007/s11263-013-0620-5.

Zhou, Xingyi, Dequan Wang, and Philipp Krähenbühl. 2019. "Objects as Points." arXiv. http://arxiv.org/abs/1904.07850.

Zou, Zhengxia, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. 2023. "Object Detection in 20 Years: A Survey." arXiv. http://arxiv.org/abs/1905.05055.