Understanding, Selecting, and Optimizing Object Detectors for Edge Applications

Md Nasir Uddin Laskar

Staff Machine Learning Engineer

Walmart Global Tech

Highlights

Introduction

Evolution of OD models

Two-stage models: R-CNNs

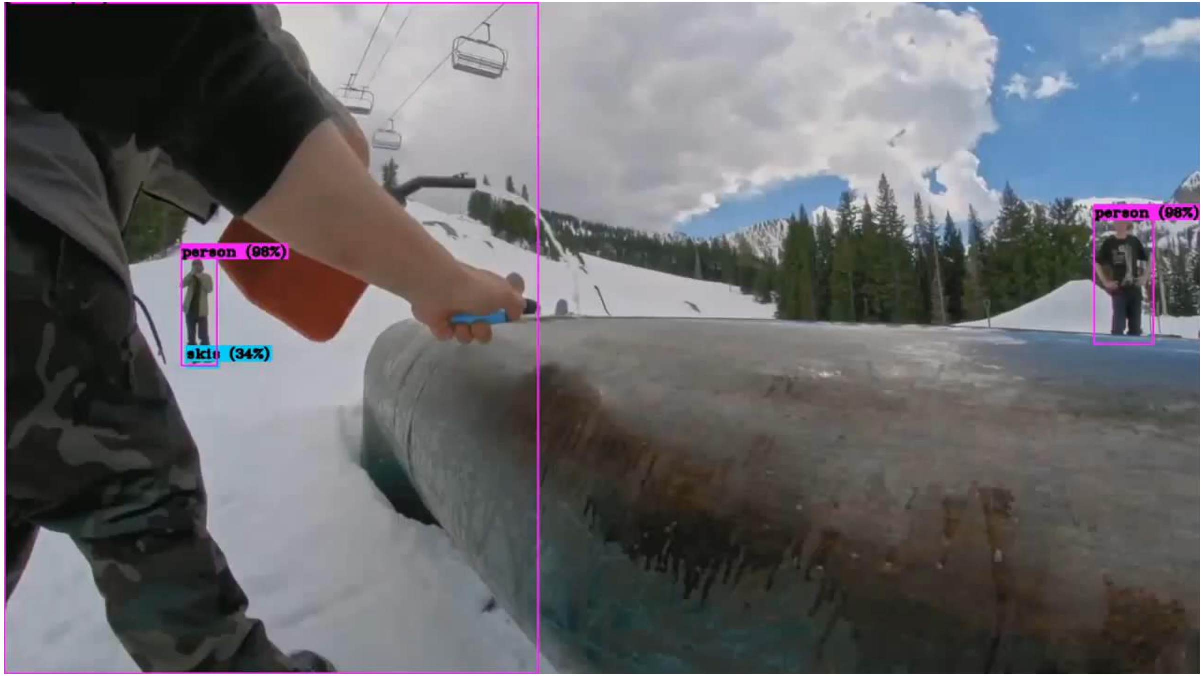
One-stage models: YOLO, SSD

Transformer-based: DETR

OD for edge devices

OD future directions

Object Detection: Introduction



https://www.youtube.com/embed/1_SiUOYUoOI

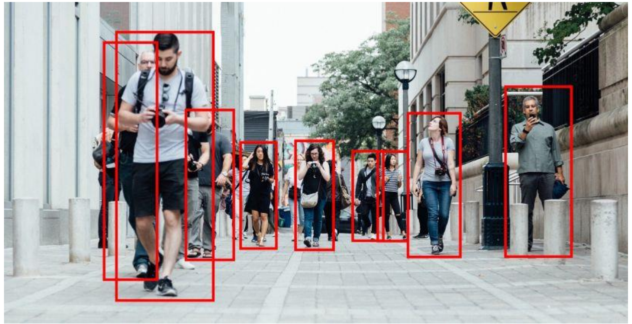
[Bochkovskiy A. et al]

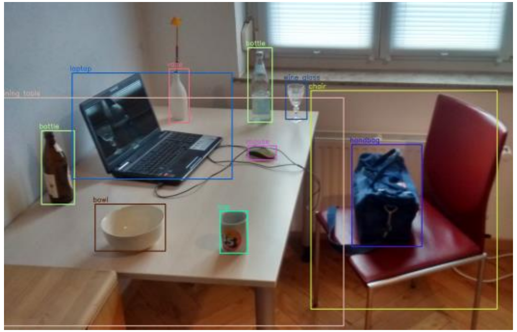
Object Detection: Task

Input: A single image (typically RGB)

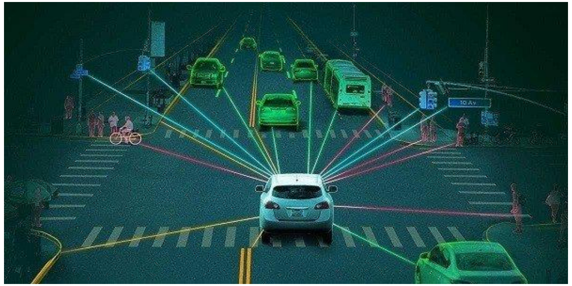
Output: A set of detected objects as class label and bounding box

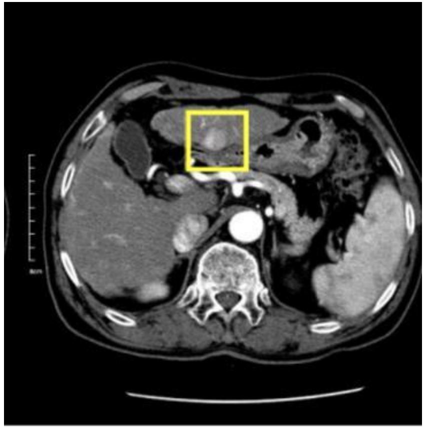
Objects: From a set of classes. Person, things, even Texts





Object Detection: **Applications**





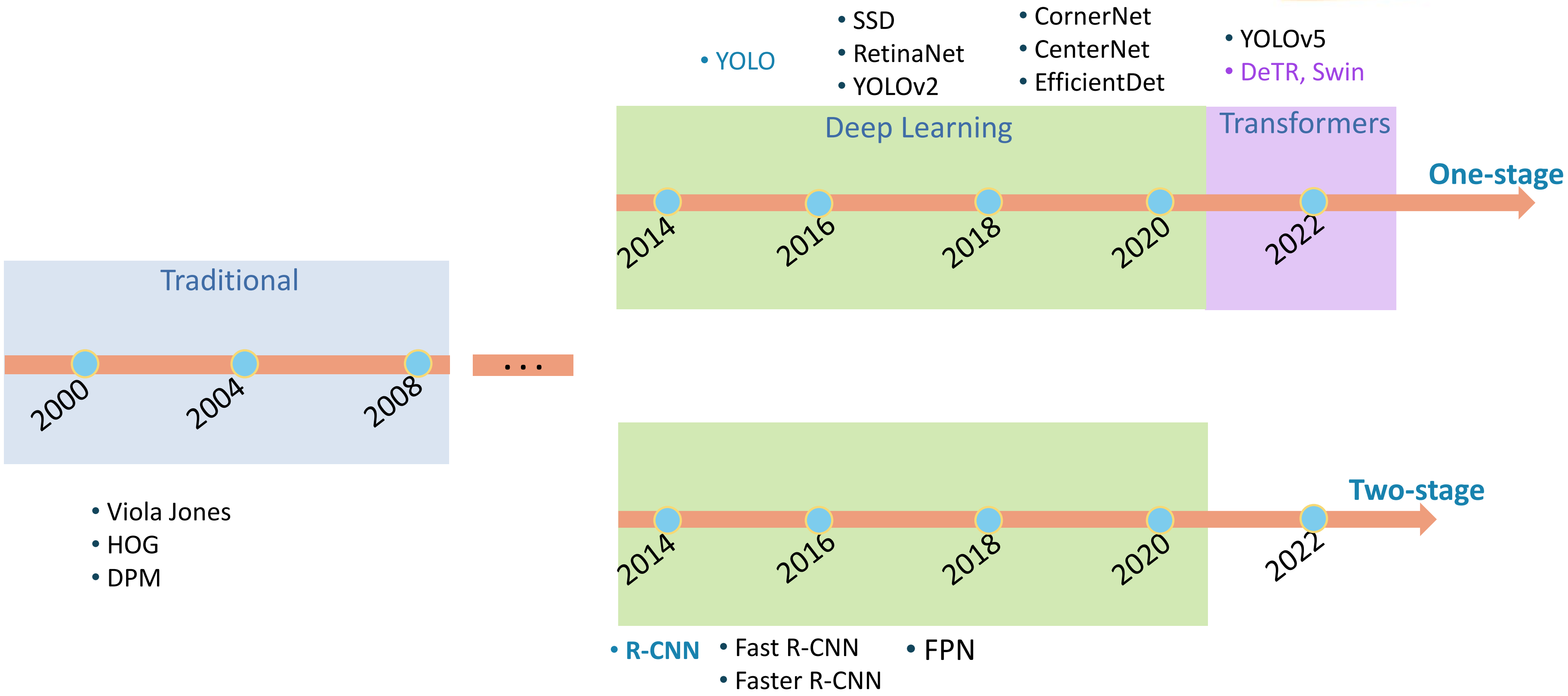
[researchleap.com/ | psimagazine.co.uk/]
[Sang-gil Lee et al, MICCAI 2018]
[learn.arcgis.com/ | vectorstock.com/]

Object Detection: Challenges

- **Multiple Outputs**
  - Image can have variable number of objects from various classes
  - Can also have high overlap between objects in the image
- **Multiple Types of Outputs**
  - Need to output **what** (class label) and **where** (bounding box)
- **High Resolution Images**
  - Classification works at 224x224. Higher resolution is needed for detection.

[image credit Bochkovskiy A.]

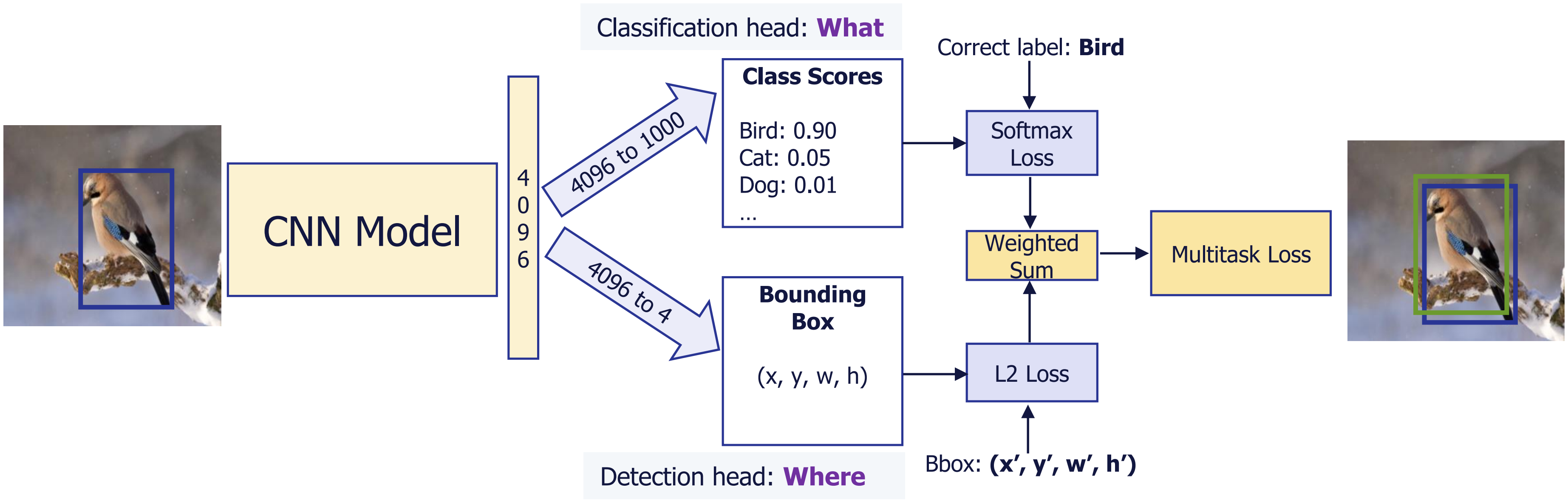
Object Detection: Evolution of Models



Object Detection: Simple Approach

- **Question**: What is the problem with this setup?

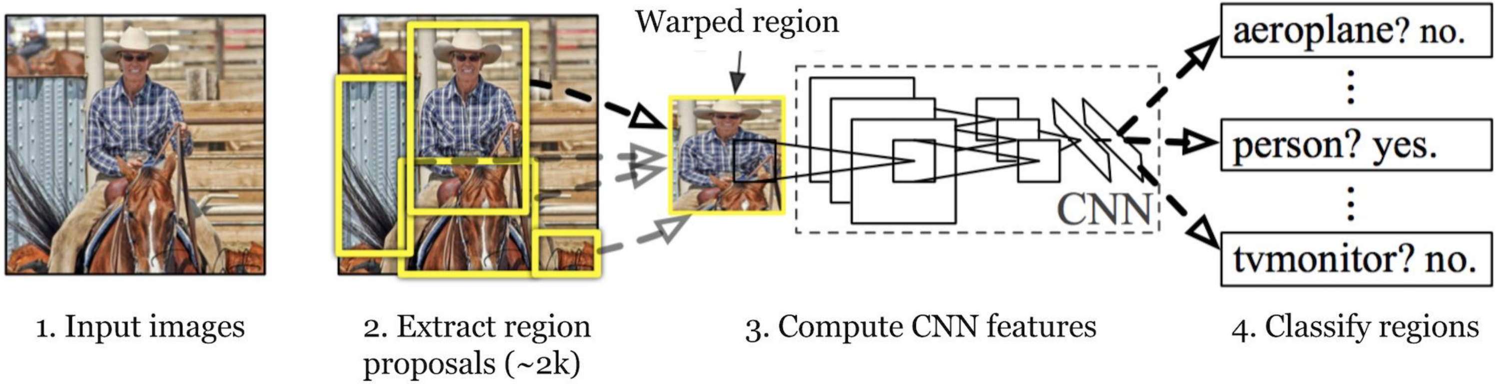
  It cannot detect if the image has multiple objects.



[Bird picture: https://pixabay.com/]

R-CNN Class of Models

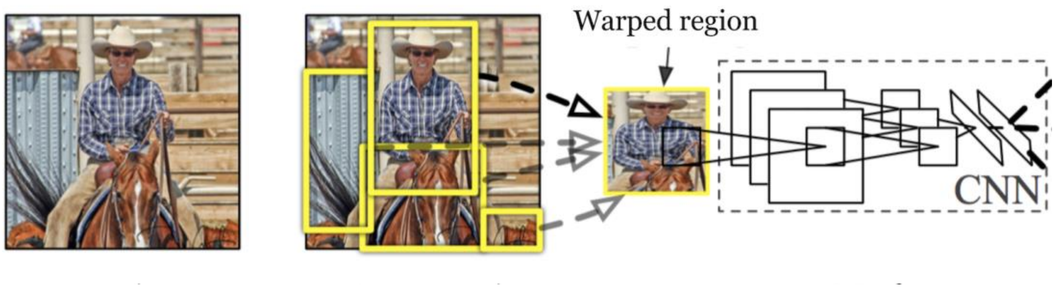
- Use selective search to identify a manageable number of object region candidates (region of interest or RoI).
- Extracts CNN features from each region independently for classification.

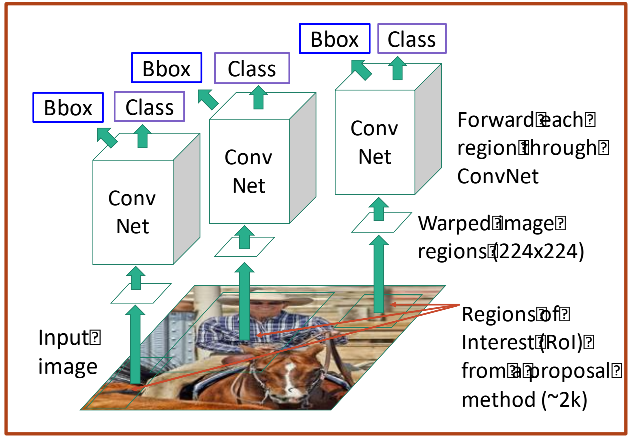


[Girshick et al, CVPR 2014]

R-CNN Steps in Detail

1. Propose category-independent RoIs by selective search
2. Warp region candidates to a fixed size as required by CNN, e.g. 224x224
3. Generate potential bounding boxes, and then run a classifier on these proposed boxes, e.g. SVM
4. Refine the bounding boxes, eliminate duplicate detections, and rescore the boxes based on other objects in the scene





[Girshick et al, CVPR 2014]

R-CNN: Impacts / Limitations

Pioneered the CNN for object detection

Sets the stage to evolve the field

- 30K citations
- 4K papers with title "R-CNN" [1]

Cannot be trained end-to-end

Requires 100s of GB of storage space

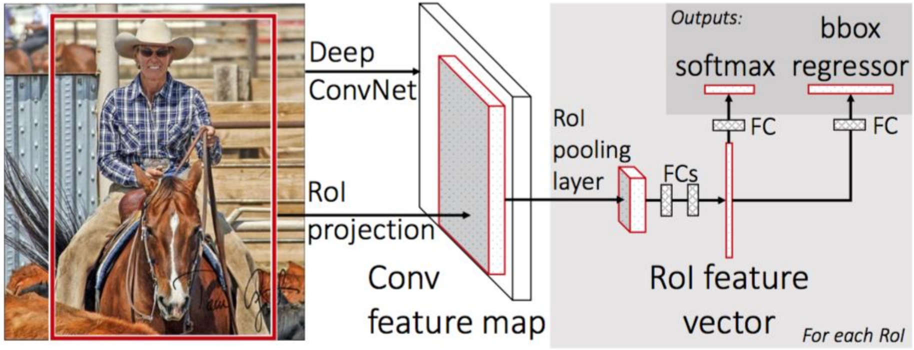
Selective search is not optimized for object detection

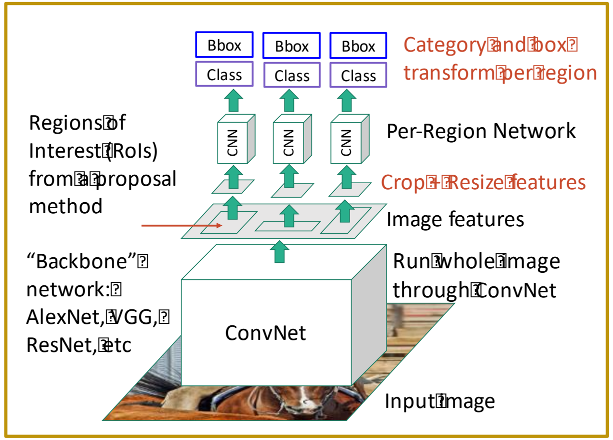
Not suitable to run real-time applications

[[1]Google Scholar advanced search. allintitle:"R-CNN"]

Fast R-CNN

- Run a single CNN on the entire image. Get RoIs from the image features instead of the image itself.
- Share computations across all ROIs rather than doing calculations for each proposal independently.
- Does not need to cache extracted features in the disk. The architecture is trained end-to-end with a multi-task loss.

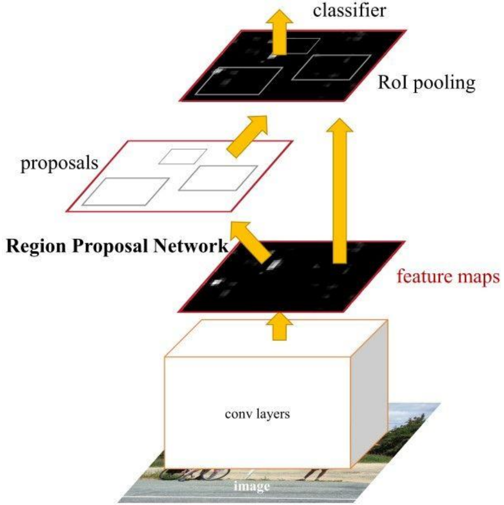


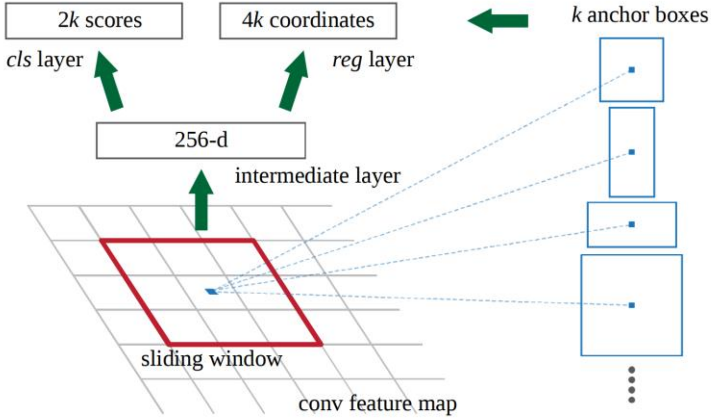


[Paper: Girshick , ICCV 2015]
[Image: https://www.mathworks.com/]

Faster R-CNN

- Nearly cost-free region proposals using Region Proposal Network (RPN), that shares convolutional features with the detection network.
- The convolutional computations are shared across the RPN and the Fast R-CNN, effectively reducing the computation time.



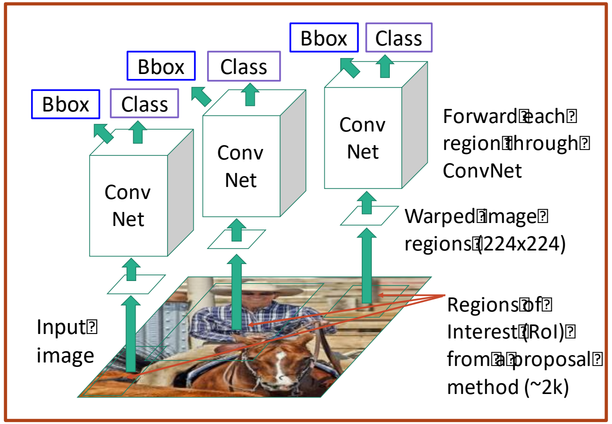


- Introduced multi-scale anchor boxes to detect objects of various sizes.

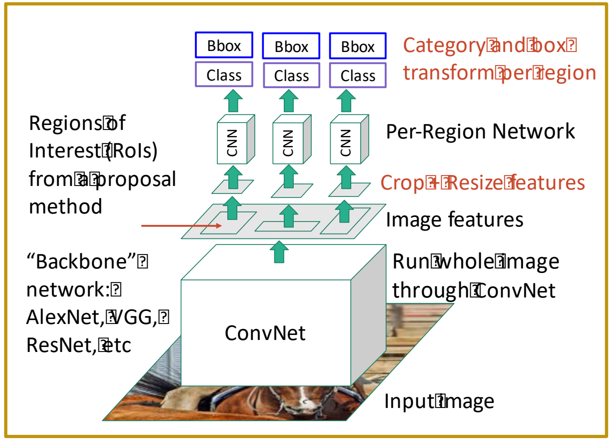
[Ren et al, NeurIPS 2015]

Slow, Fast, and Faster R-CNN

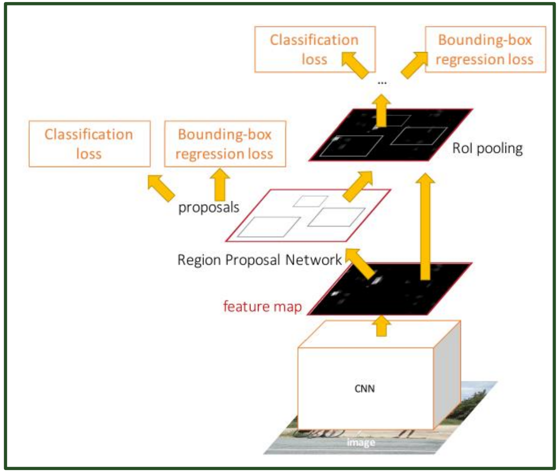
Run CNN independently for each region



Differentiable cropping to shared image features



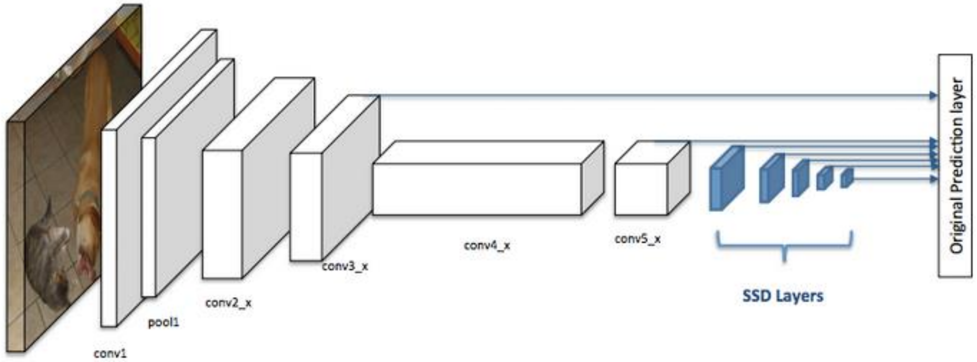
Compute region proposals with CNNs

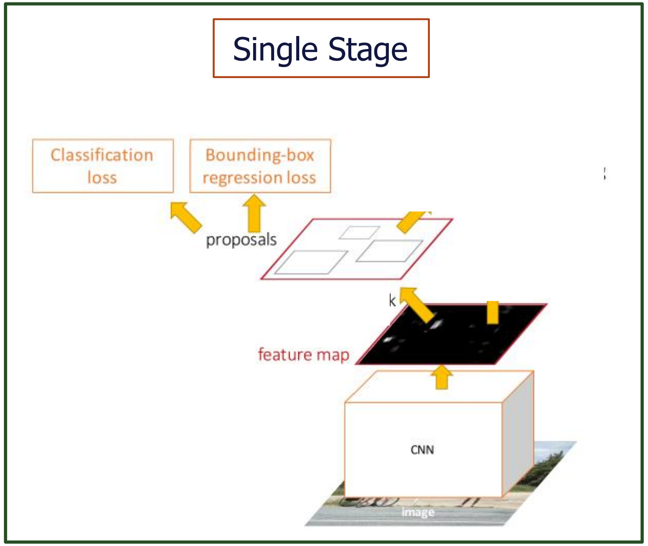


[image credit: Justin Johnson, University of Michigan]

Single Shot Detector: SSD

- Use pyramidal feature hierarchy for efficient detection of objects of various sizes.
- Model Architecture: Backbone model (VGG) and SSD head. SSD head outputs the bounding box and object classes.
- Large **fine-grained** feature maps (lower level) at are good at capturing small objects and small **coarse-grained** feature maps detect large objects well (higher level).



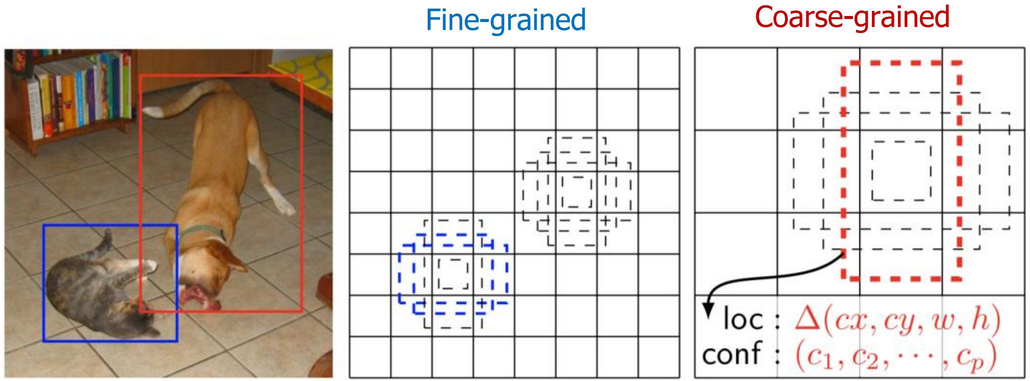


[Wei Liu et al, ECCV 2016]

SSD: Steps

- Eliminate RPN. Use grid cells technique to detect object of various sizes.
- Predicts offset of predefined anchor (default) boxes for every location of the feature map.
- The anchor boxes on different levels are rescaled so that one feature map is only responsible for objects at one particular scale.

- Cat (Small Object) is captured by the 8x8 feature map (lower level).
- Dog (Large Object) can only be detected in the 4x4 feature map (higher level)

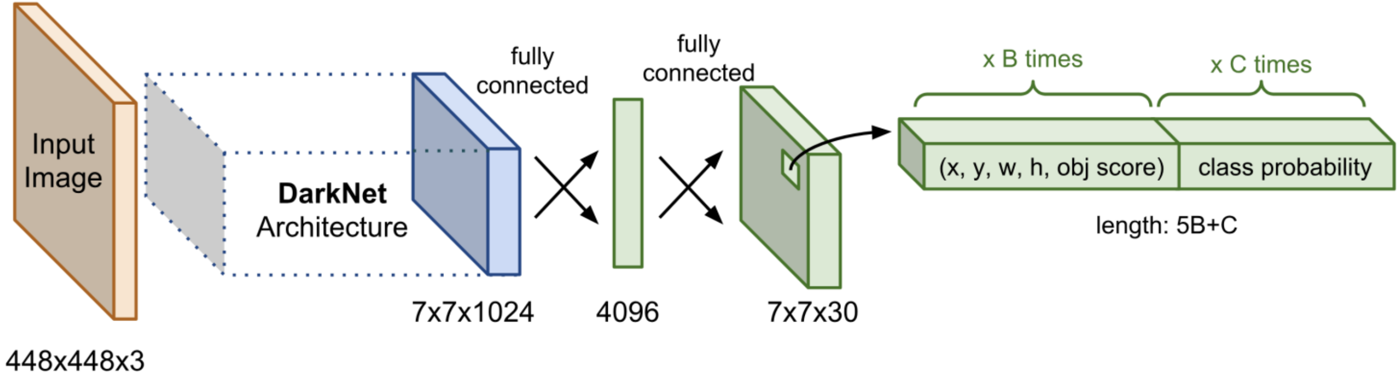


(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

[Wei Liu et al, ECCV 2016]

YOLO Class of Models

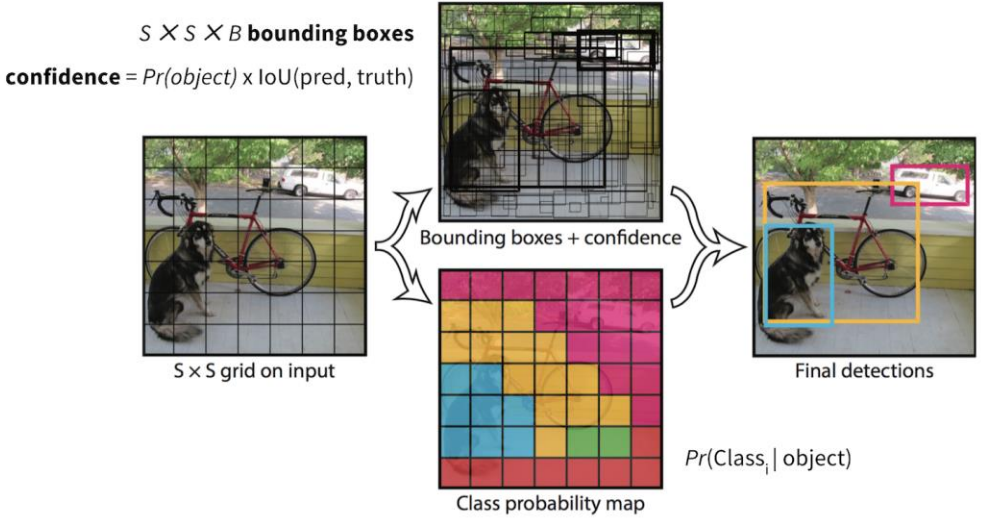
- One of the first attempts to build a fast, real-time object detector.
- YOLO Frames the object detection as a single regression problem, straight from image pixels to bounding box and class probabilities. Hence, YOLO, You Only Look Once.
- The final prediction of shape $S \times S \times (5B + C)$ is produced by two fully connected layers over the whole conv feature map.



[Paper: Redmond et al, CVPR 2016.]
[image: https://lilianweng.github.io/]

YOLO: Steps and Limitations

- Split the image into SxS cells. Each cell predicts
  - The **location** of bounding boxes as (x, y, w, h), a **confidence score**, and a **probability** of object class
- Final prediction is $S \times S \times (5B + C)$. For PASCAL VOC S=7, B=2, C=20. That is why the final map is 7x7x30



- Cannot detect group of small objects. Maximum B (here, 2) objects per cell
- Irregular shaped objects

YOLOv2 and Beyond

YOLOv2

- Light-weight base model, DarkNet-19
- BatchNorm on conv layers
- Conv layers to predict anchor boxes
- Direct location prediction

YOLOv3

- Logistic regression for confidence scores
- Multiple independent classifiers instead of one softmax
- Skip-layer concatenation

. . .

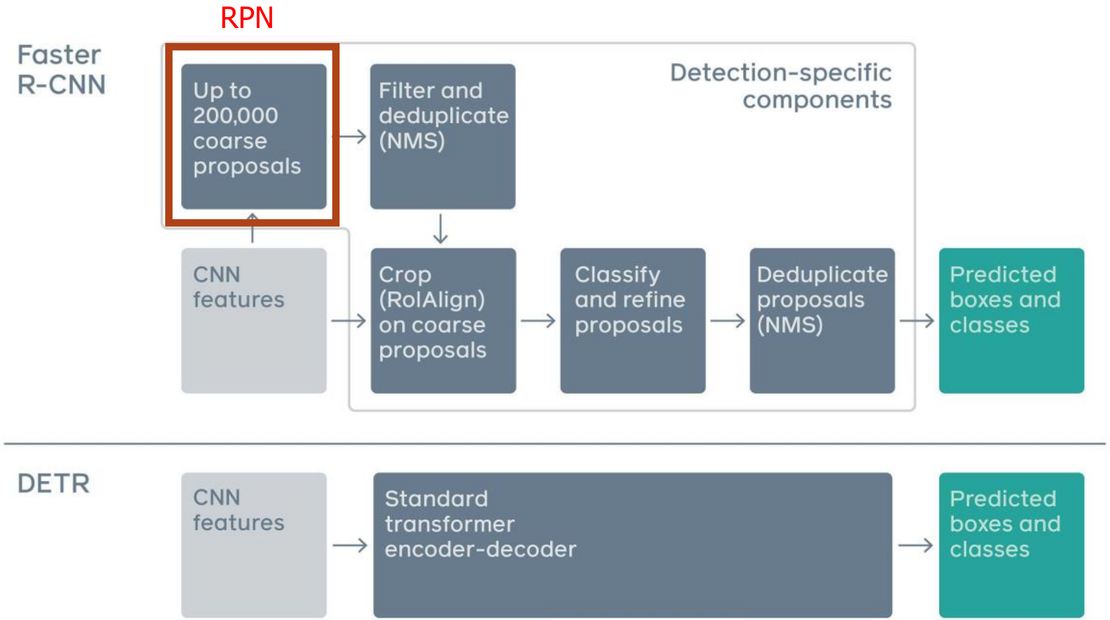
YOLOv8

- Latest in the series

[Redmond et al, CVPR 2017]

Transformer-based Detectors: DETR

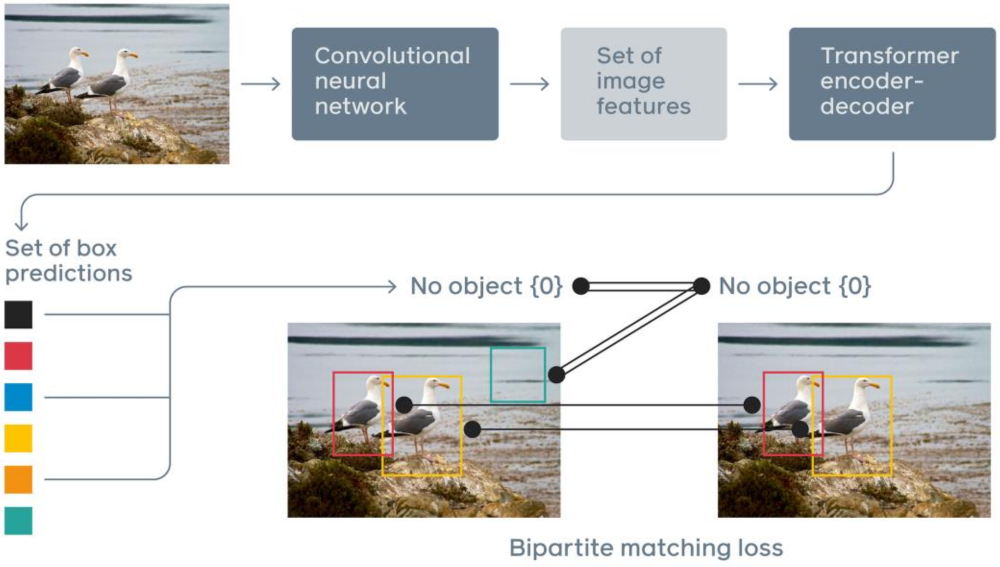
- DETR frames the object detection task as an image-to-set problem. Given an image, the model predicts an unordered set of all the objects present.
- Existing methods have number of components that make them complicated.



[Carion N, Massa F et al]

Transformer-based Detectors: **DETR**

- Directly predicts the final set of detections in parallel
- During training, bipartite matching uniquely assigns predictions with ground truth boxes.
- Predictions with no match yield a “no object” class prediction.



- Slow convergence, 5x slower than Faster R-CNN
- Poor detection on small objects

[Carion N, Massa F et al]

Object Detection in Video

- Task of detecting objects from a video, such as in autonomous driving scenario

- Challenges
  - Appearance deterioration
  - Changes of video frames, e.g., motion blur, part occlusion, camera re-focous, rare poses etc.

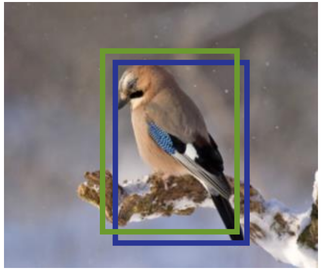
- Aggregate temporal cues from different frames. Two-step baseline models (Faster R-CNN, R-FCN)
  - **Box-level.** Post-processing of temporal information.
  - **Feature-level.** Improve features of the current frame by aggregating that of adjacent frames.

- Recent. Use one-step models such as YOLO / DETR to build end-to-end detectors.

Evaluation Metrics

poor

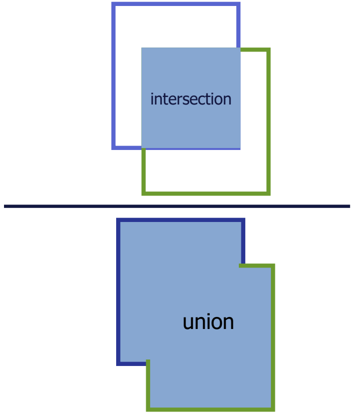
poor



good

- **Precision** measures how accurate are the predictions of the detector, aka, percentage of correct predictions.
- **Recall** measures how good the object detector can detect all the positives.
- **IoU** measures the overlap between GT and predicted boundaries.

IoU =



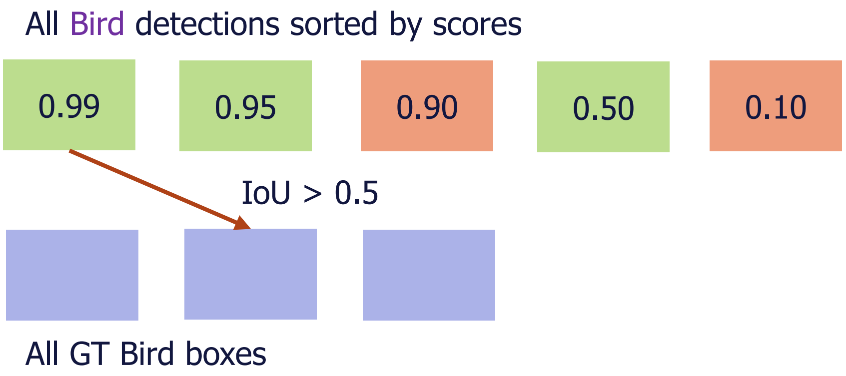
- **Average Precision (AP)** computes the mean precision value for recall value over 0 to 1.

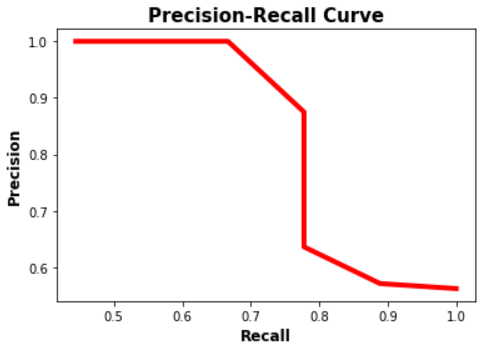
[Bird picture: https://pixabay.com/]

Mean Average Precision (mAP)

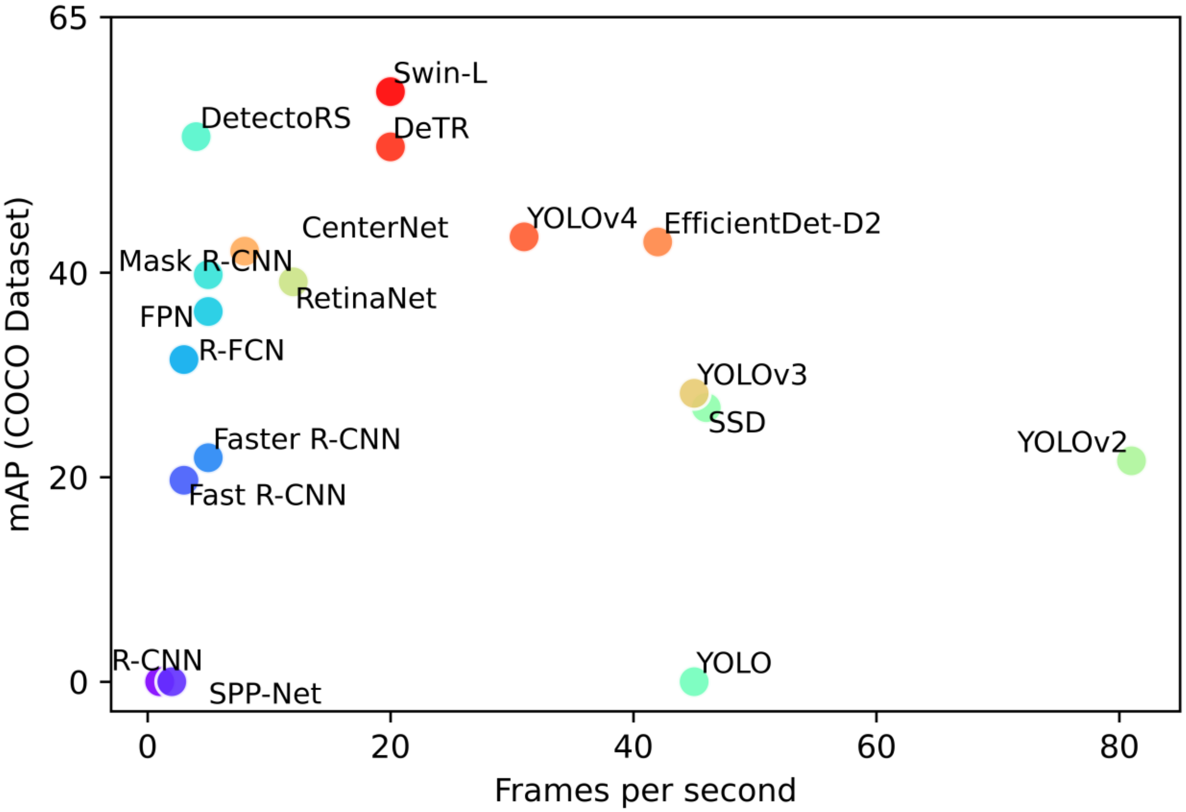
1. Run the detector for all test images
2. For each category: for each detection
   1. Compute the AP, which is area under PR curve
   2. Plot a point on PR curve if IoU > 0.5
3. mAP = average of AP for each category
4. COCO mAP : average AP for IoU from 0.5 to 0.95 with a step size of 0.05.

- **Speed** of the detection is usually quantified with FPS





Benchmark Analysis



Object Detection at the Edge: Considerations / Tradeoffs

Considerations and Challenges

Compute / Speed

- CPU / GPU / NPU
- Real-time applications
- High resolution images

Memory

- Model size / #Params
- RAM / Flash
- Imbalanced memory distribution in first conv layers

Post-Process

- Some edge devices do not support NMS

Tradeoffs

Accuracy

- Single-stage models have lower mAP

FPS

- Higher precision models usually have lower FPS

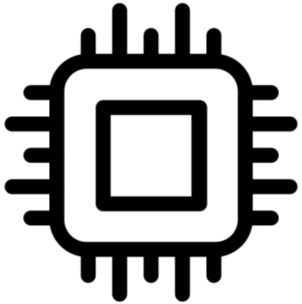
Object Detection at the Edge: Develop and Optimize

- **Design New Model**
  - Design new model architecture that runs on your target device and train it [Not Recommended]
  - Smaller version of an existing model and train it, such as FOMO, MCUNetV2

- **Transfer Learning**
  - Fine-tune an existing model on your custom data. For example, TF Detection Model Zoo.
  - Pick a model that works best for your use-case and target hardware.

- **Pre-training Optimizations**
  - Quantization-aware training of existing models

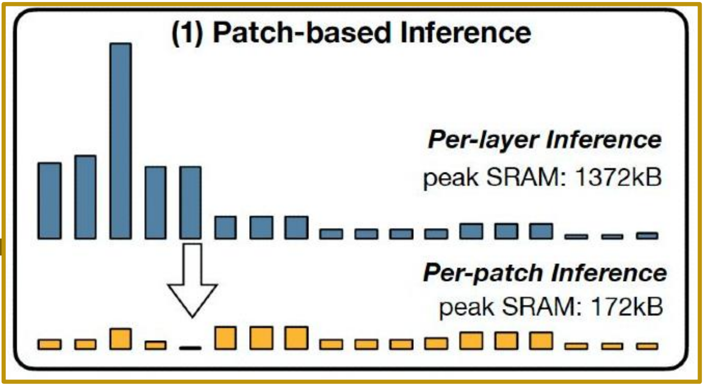
- **Post-training Optimizations**
  - Model pruning / quantization
  - Hardware specific optimizations: TFLite / TensorRT / ONNX / similar



Object Detection at the Edge: Example

MCUNetV2

- MobileNetV2 base model
- Patch-by-patch inference to solve imbalanced memory distribution
- Receptive Field redistribution to reduce computation overhead



On Pascal VOC

- 68.3% (+16.9) with 438kB SRAM
- 64.6% (+13.2) with 247kB SRAM

- Only 7 FPS
- Not tested on high resolution images

[MCUNetV2: Lin et all, NeurIPS 2021]

Object Detection: What is Next?

Fastest R-CNN

- Accuracy of two-stage
- Speed of one-stage

Transformers

- More algorithms/ models
- Compatibility towards edge devices

3D Obj Detection

- Particularly critical for autonomous driving

Detection in Video

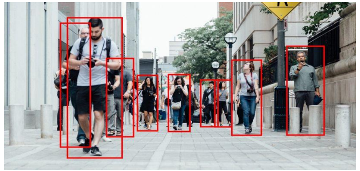
- Efficient detection in video
- Has so many real-world applications

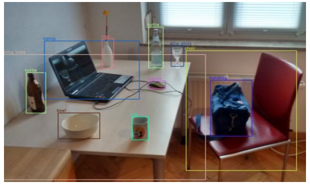
On-device Training

- Training at the edge devices
- Adapt to data drifts

Conclusion

- Object detection applications and challenges
- Evolution of object detection systems
- Some of the popular object detection models
- Considerations and tradeoffs of object detection for edge applications
- Optimizing object detection systems for edge devices





Questions / Discussions

Resources

- Off the shelf object detection models:
  - TensorFlow OD model Zoo
  - TensorFlow Mobile Optimized Detectors
  - Detectron 2: object detection using PyTorch and model zoo
- Object detection training datasets
  - Pascal VOC dataset
  - MS COCO Dataset
- Object detection training frameworks
  - TensorFlow Lite , Example object detection for mobile devices
  - PyTorch example object detection using pre-trained models
- Get hands-on
  - Train YOLOv4 using Google Colab