AMNIL TECHNOLOGIES

Presentation on Object Detection using YOLOv8n Architecture

Prepared by: Sejal Panta

4th Nov, 2025

What is Object Detection?

Object Detection is a computer vision task that identifies and localizes multiple objects within an image or video frame.

Difference from Image Classification:

Classification: single label for entire image

Detection: class + location (bounding boxes)

Complexity: Combines classification and localization

Real-life Applications:

Self-driving cars: detecting pedestrians, vehicles

Security surveillance: detecting intruders

Medical imaging: detecting tumors or anomalies

Robotics: object manipulation and navigation

Components of Object Detection

- Input Image/Video: source to analyze
- Feature Extraction: extract important visual features (edges, textures, shapes)
- Object Localization: find where objects are (bounding boxes)
- Classification: identify what the object is
- Post-processing: techniques like Non-Maximum Suppression (NMS) to remove duplicate detections

Key Outputs of Object Detection

- Bounding Boxes: Rectangles around detected objects
- Class Labels: e.g., "car", "dog"
- Confidence Scores: Probability indicating model certainty

Bounding Boxes in Object Detection

Bounding boxes are rectangular boxes drawn around objects in an image to localize them. They are essential for object detection, as they tell the model where an object is.

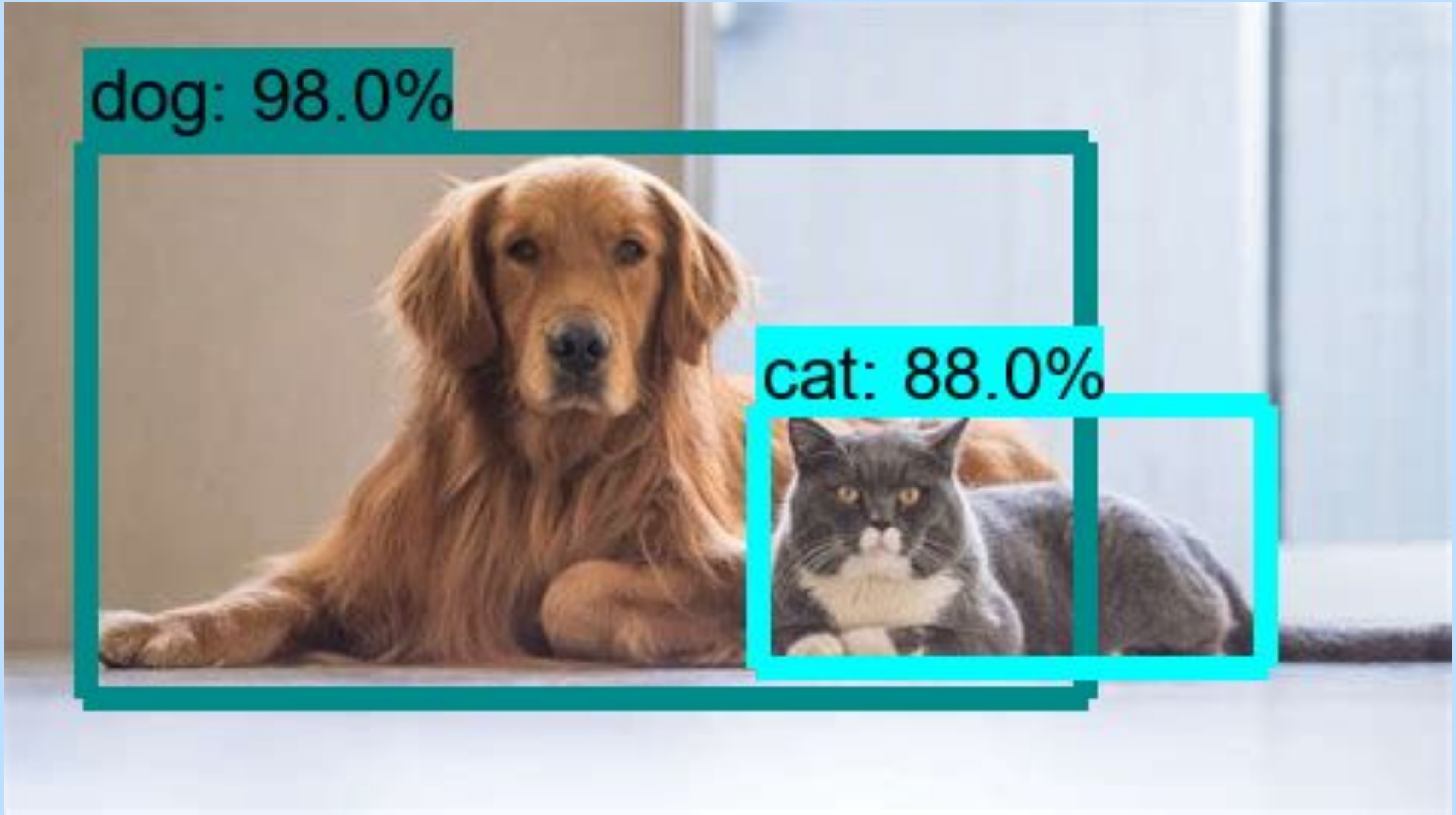
Types:

1. Axis-aligned (corner-based):
   Defined by top-left and bottom-right coordinates: (x_min, y_min, x_max, y_max)

2. Center-based:
   Defined by center coordinates and dimensions: (center_x, center_y, width, height)
   Common in YOLO models

Best Practices:

- Normalize coordinates to [0,1] relative to image size
- Tight boxes to improve detection precision
- Looser boxes to help when objects are partially occluded


dog: 98.0%
cat: 88.0%

Object Detection Architectures Overview:

a. Two-Stage Detectors (Region Proposal + Classification)
Examples: R-CNN, Fast R-CNN, Faster R-CNN
Characteristics: Accurate but slower

b. Single-Stage Detectors (Direct prediction)
Examples: YOLO (You Only Look Once), SSD (Single Shot Detector)
Characteristics: Real-time but slightly less accurate

c. Modern / Efficient Detectors
Examples: EfficientDet, DETR, YOLOv5-8
Characteristics: Efficient, transformer-based

ML Pipeline:

- Data Collection & Annotation: label objects with boxes & classes
- Data Preprocessing: resize, normalize, format boxes
- Data Augmentation: flips, rotations, color jitter
- Model Selection: YOLO, Faster R-CNN, SSD, DETR
- Training: optimize detection & localization
- Validation & Evaluation: tune hyperparameters
- Deployment: optimize for speed & efficiency

Challenges of Object Detection:

1. Varying Object Sizes: Detecting very small or very large objects in the same image
2. Occlusion: Objects partially hidden behind others
3. Cluttered Backgrounds: Hard to distinguish objects from busy scenes
4. Class Imbalance: Rare objects may be missed
5. Real-Time Performance: High accuracy can require more computation

Advantages of Object Detection:

1. Localization + Classification: Knows what the object is and where it is
2. Automation: Useful in self-driving cars, surveillance, retail, and medical imaging
3. Real-Time Detection: Modern models (YOLO, SSD) can detect objects fast
4. Versatile Applications: Works on images and video for various industries

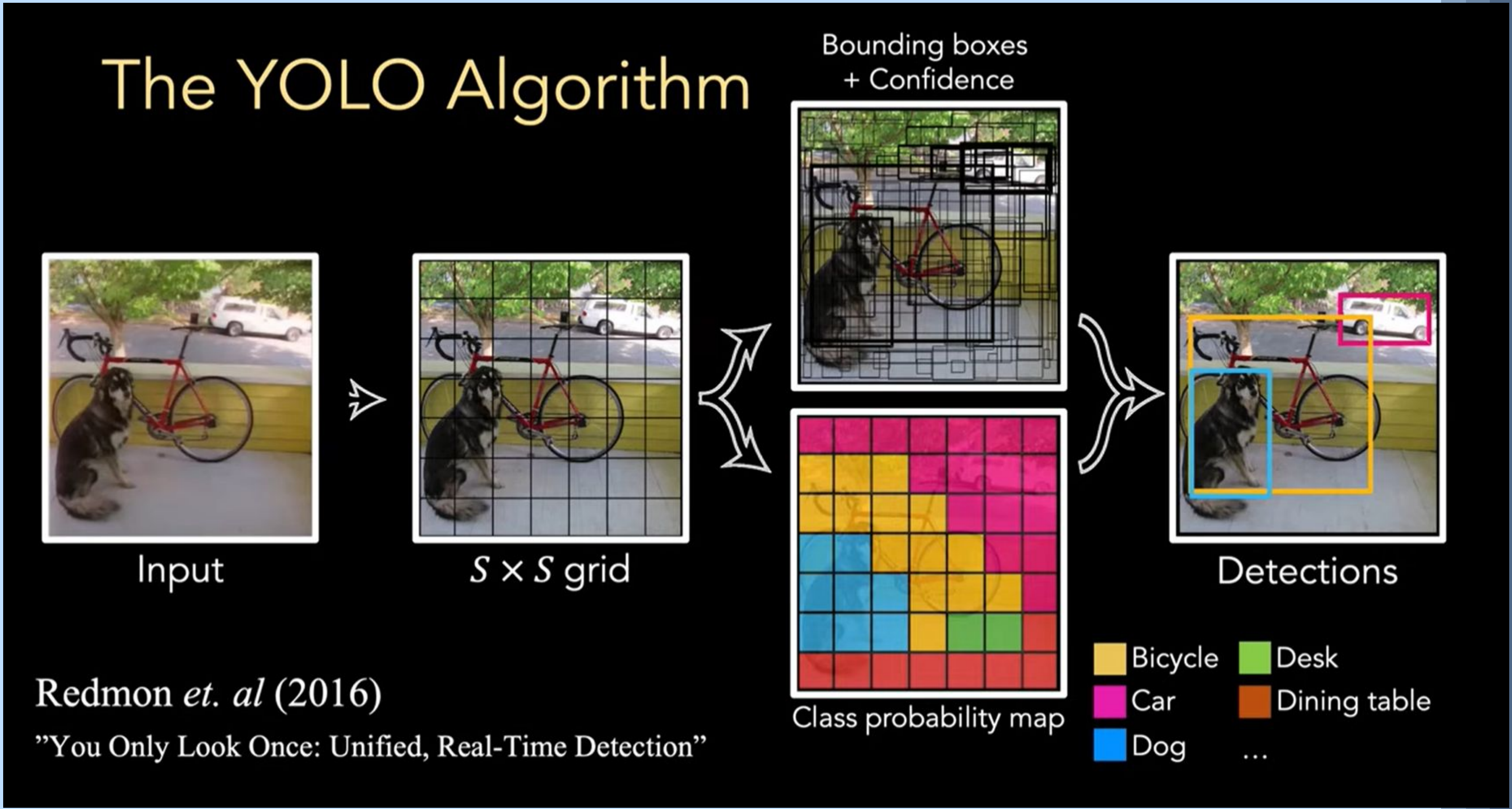
YOLO

It stands for You Only Look Once. It's a real-time object detection system. Detects objects and their bounding boxes in a single forward pass of the network.

YOLO Architecture Overview:

1. YOLO resizes and divides the input image into a grid of S x S cells.
2. Each grid cell predicts:
   - B bounding boxes (coordinates + width/height)
   - Class probabilities for objects
   - Confidence score for each box

   $\text{Confidence} = \Pr(\text{Object}) \times \text{IoU}_{\text{pred,truth}}$
3. Architecture has three main parts:
   - Backbone: Extracts features from input images (like CSPDarknet, or for YOLOv8 – a custom convolutional backbone).
   - Neck: Combines features at different scales (FPN / PAN) to detect objects of various sizes.
   - Head: Makes final predictions (bounding boxes, class probabilities, confidence scores).
4. Predictions are filtered using Non-Maximum Suppression (NMS) to remove overlapping boxes.


The YOLO Algorithm
Bounding boxes
+ Confidence
Input
S × S grid
Detections
Class probability map
Bicycle
Desk
Car
Dining table
Dog
...
Redmon et. al (2016)
"You Only Look Once: Unified, Real-Time Detection"

YOLOv8

- It stands for You Only Look Once, version 8, developed by Ultralytics (2023).
- It's the latest and most advanced version in the YOLO family (YOLOv5 → YOLOv8).
- It includes multiple model sizes — nano (n), small (s), medium (m), large (l), extra-large (x), all sharing the same architecture but with different depth and width.
- Designed for real-time object detection, image segmentation, and classification.
- Uses a single neural network that predicts bounding boxes and class probabilities in one pass.
- Fully rewritten in Python and PyTorch for flexibility and ease of use.

Features & Improvements:

1. Anchor-free detection: Faster, more flexible bounding box prediction.
2. Better feature fusion: Improved PAN-FPN neck for multi-scale detection.
3. Lighter and faster: Optimized for real-time performance.
4. Stronger accuracy: Higher mAP than YOLOv5/YOLOv7 on benchmarks.
5. Modular design: Easy to fine-tune, extend, or train from scratch.
6. Supports export to formats: ONNX, TensorRT, CoreML, OpenVINO, etc.

YOLOv8 Model Variants:

YOLOv8n (Nano): Fastest, lightest, best for beginners.
YOLOv8s (Small): Good balance of speed and accuracy.
YOLOv8m (Medium): Moderate size for general projects.
YOLOv8l (Large): Higher accuracy, slower speed.
YOLOv8x (X-Large): Most accurate, needs powerful GPU.

YOLOv8n:

1. A Specific Model Variant
2. The "n" stands for nano, meaning it's the smallest and fastest model in the YOLOv8 family.
3. It's designed for:
   - Lower GPU/CPU usage
   - Faster inference
   - Lightweight deployment (mobile, edge devices)
4. Trades some accuracy for speed and efficiency.
5. Ideal for beginner-level object detection projects (like Pascal VOC 2012).
6. Built with fewer parameters (~3.2 M) and smaller layers, reducing memory usage.
7. Often used for real-time tasks such as webcam object detection or IoT applications.
8. Despite being the smallest, it inherits all modern YOLOv8 features (like auto-shape, mosaic augmentation, and advanced loss functions).

THANK YOU!