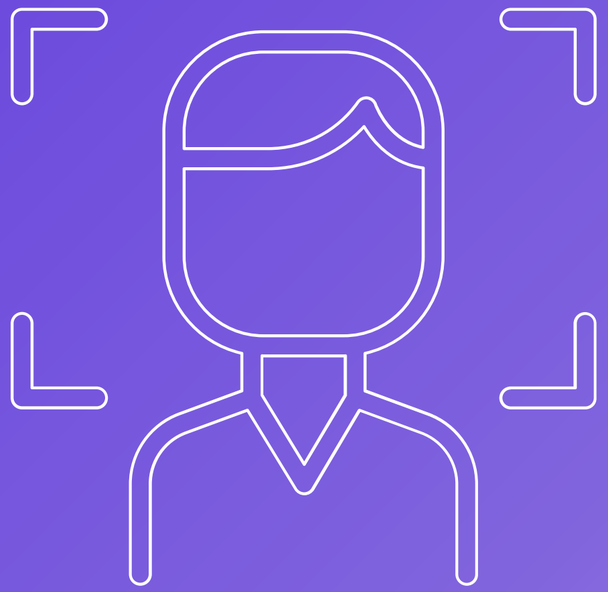
# PERFORMANCE BENCHMARKING OF YOLO ARCHITECTURES

## FOR REAL – TIME OBJECT DETECTION




**Team 6**

- **Department of Computer Engineering**
- **Ataturk University**
- **Artificial Intelligence Systems - Resul TUGAY**
- **30.12.2025**

# OUR TEAM

## Development Team

### Abdelrahman MOHAMED

**YOLOv5 Specialist & Web Dev.**

- Trained YOLOv5 Model
- Analyzed YOLOv5 Metric Results
- Developed Web Interface
- Front-end Integration

### Ramazan YILDIZ

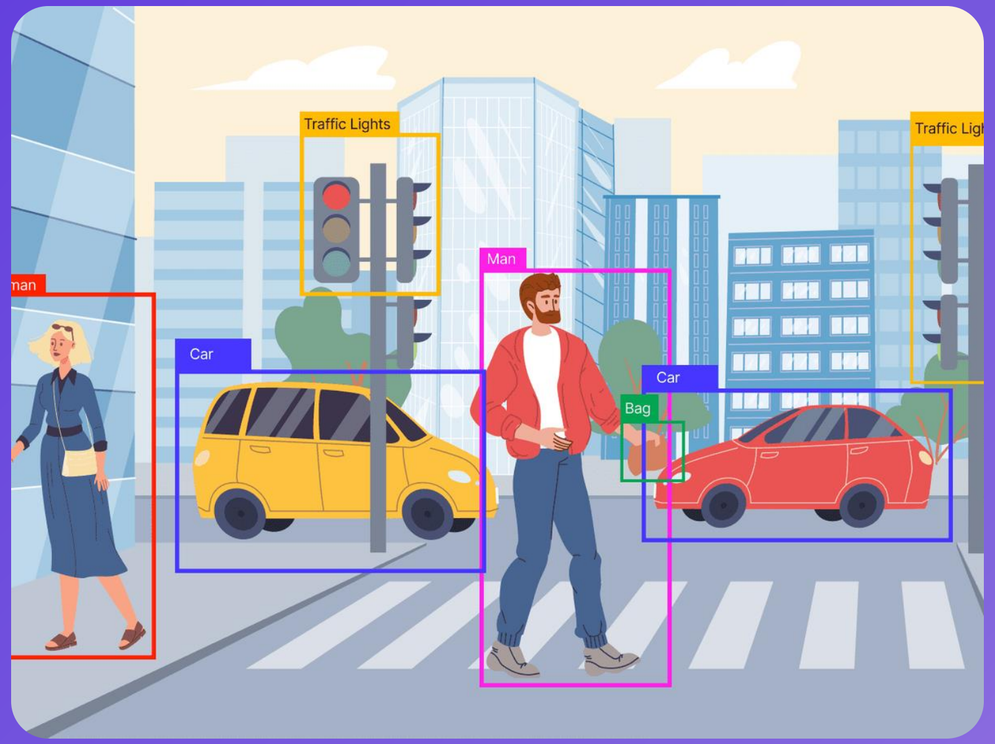
**Project Planning & AI Research**

- Trained YOLOv8m & YOLOv8n Models
- Project Initialization & Prep
- Research & Dataset Preparation
- Technical Documentation
- Model Comparison Analysis

### Beyza GÜLER

**YOLOv11 Specialist & Reporting**

- Trained YOLOv11 Model
- Analyzed YOLOv11 Metric Results
- Technical Documentation
- Presentation Slides Design

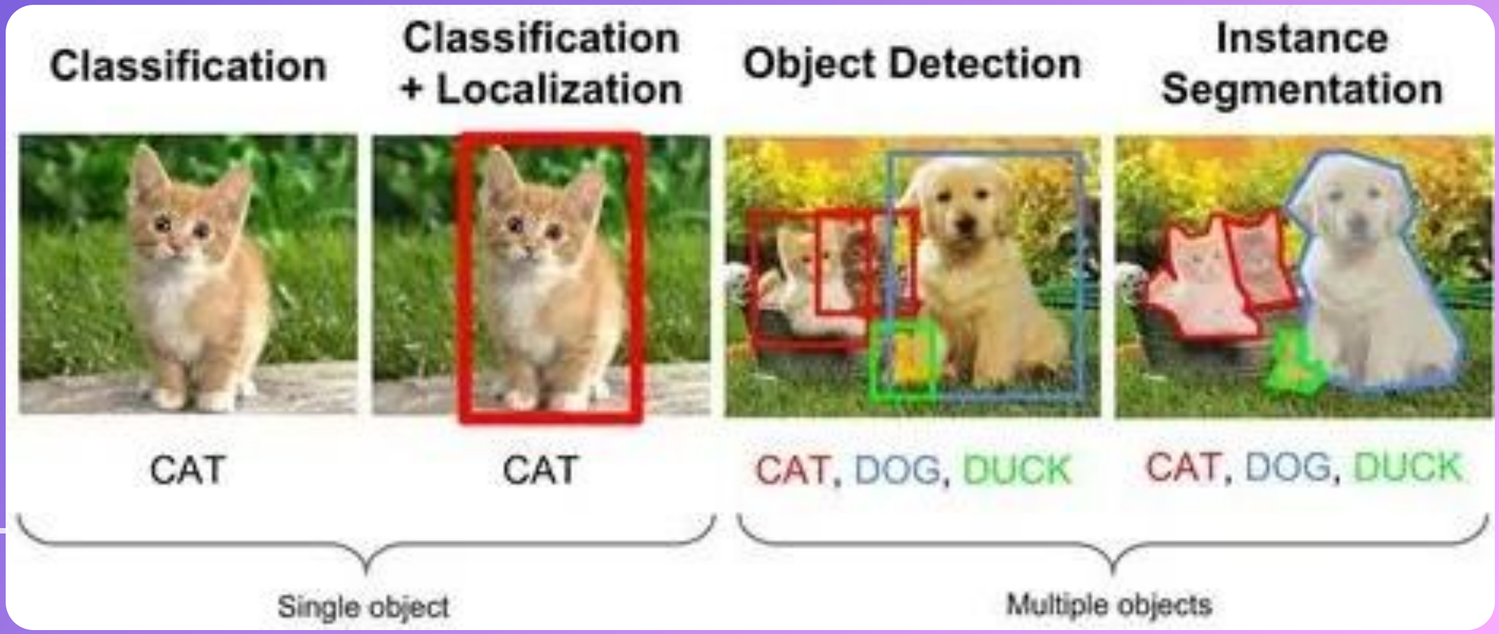



# Abstract

- Real-time object detection using YOLO
- Pre-trained YOLO models + Roboflow pre-labeled dataset
- Streamlit-based web deployment

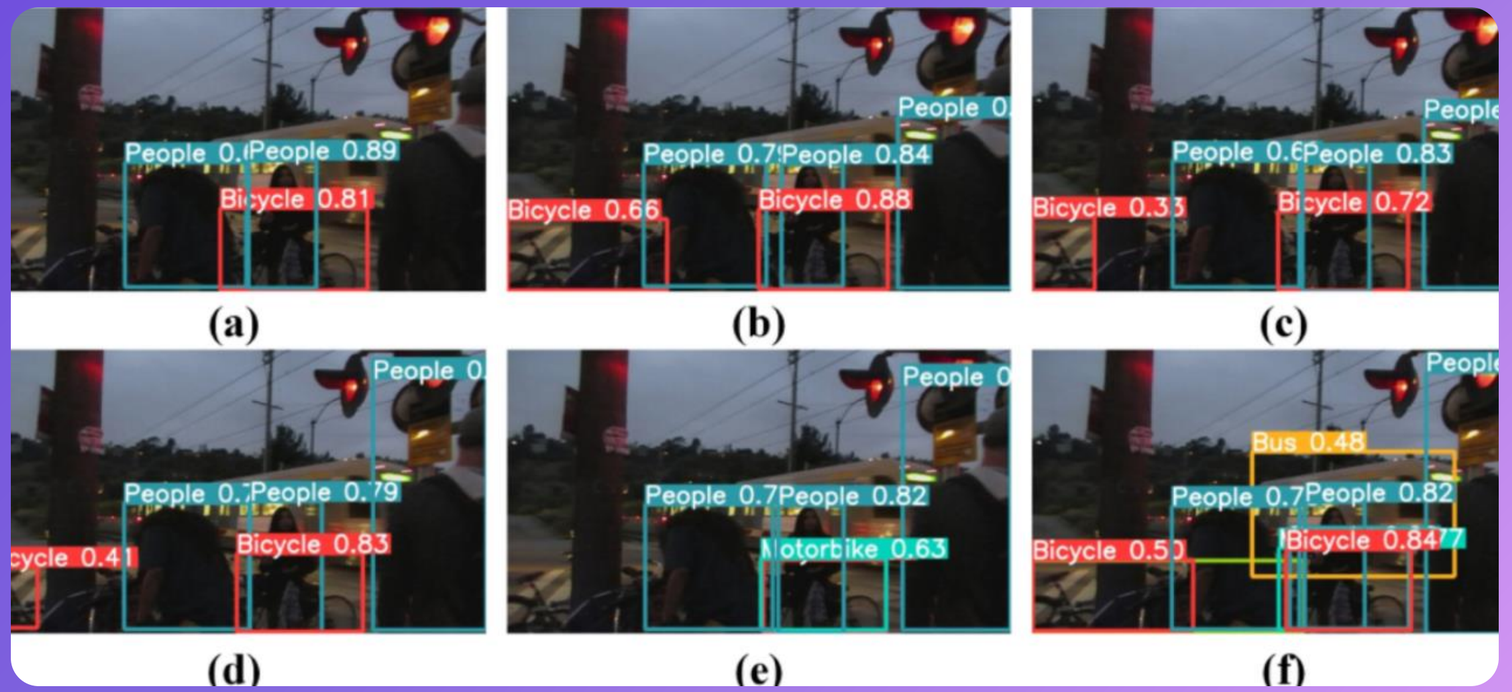
# Introduction – Object Detection

- What is object detection?
- Difference from image classification
- Real-time requirement



# Problems

Title: Problem Definition




- Real-time deployment challenges
- Computational cost of high-accuracy models
- Limited accessibility of web-based systems

# Input – Output

**Input:**

- Images
- Video streams
- Webcam input

**Output:**

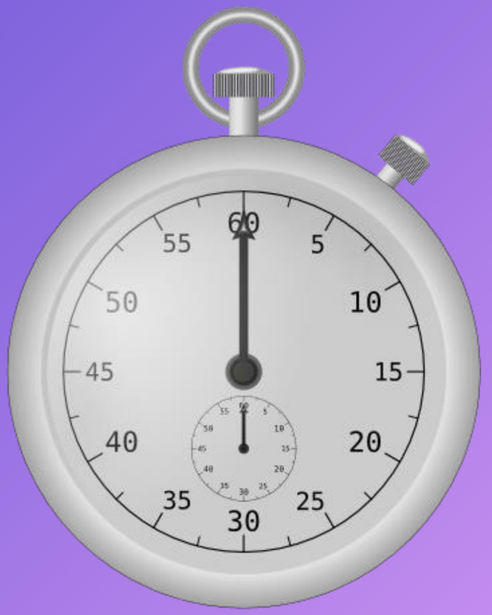
- Bounding boxes
- Class labels
- Confidence scores

# Input – Output

**Input:**

**Output:**

# Goals




- Real-time object detection
- High FPS performance
- Easy-to-use web interface
- Scalable architecture for future extensions

# Methodology (Overview)

Title: Methodology

- Train YOLO model with Labeled Data
- Capture video using OpenCV
- Perform object detection
- Visualize results
- Deploy using Streamlit

1 Input

2 Preprocessing

3 Detection Model (YOLO)

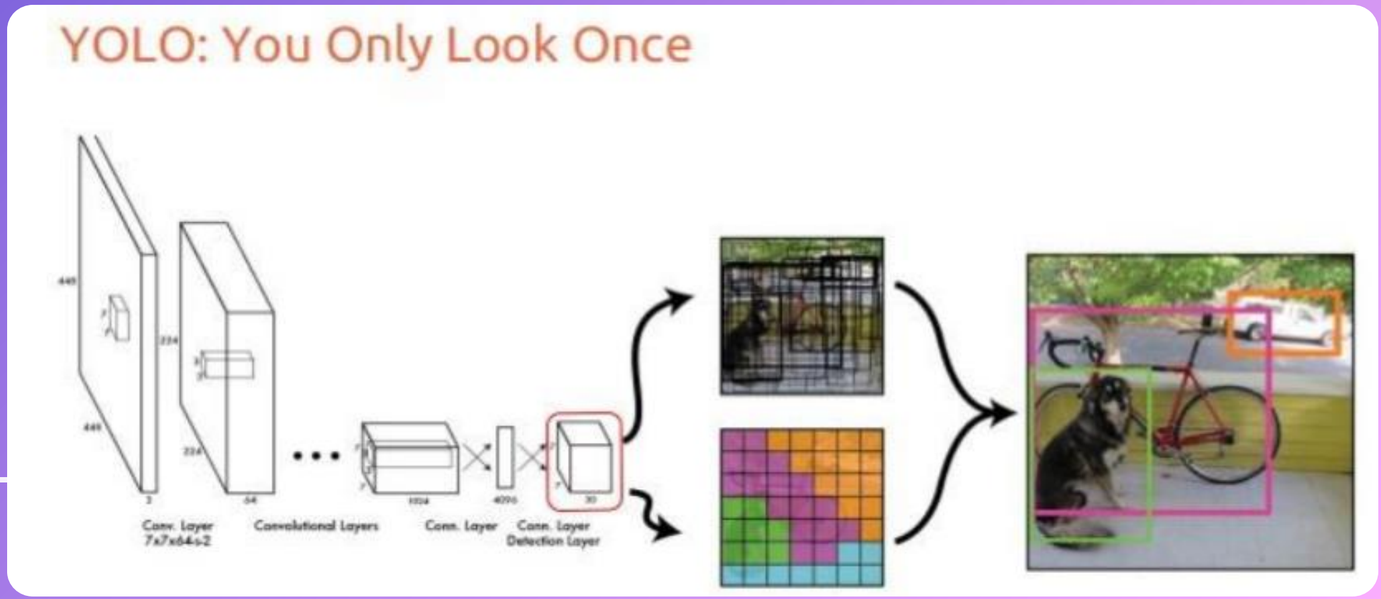
4 Output Visualization

5 Web Deployment

# Methodology (Model)

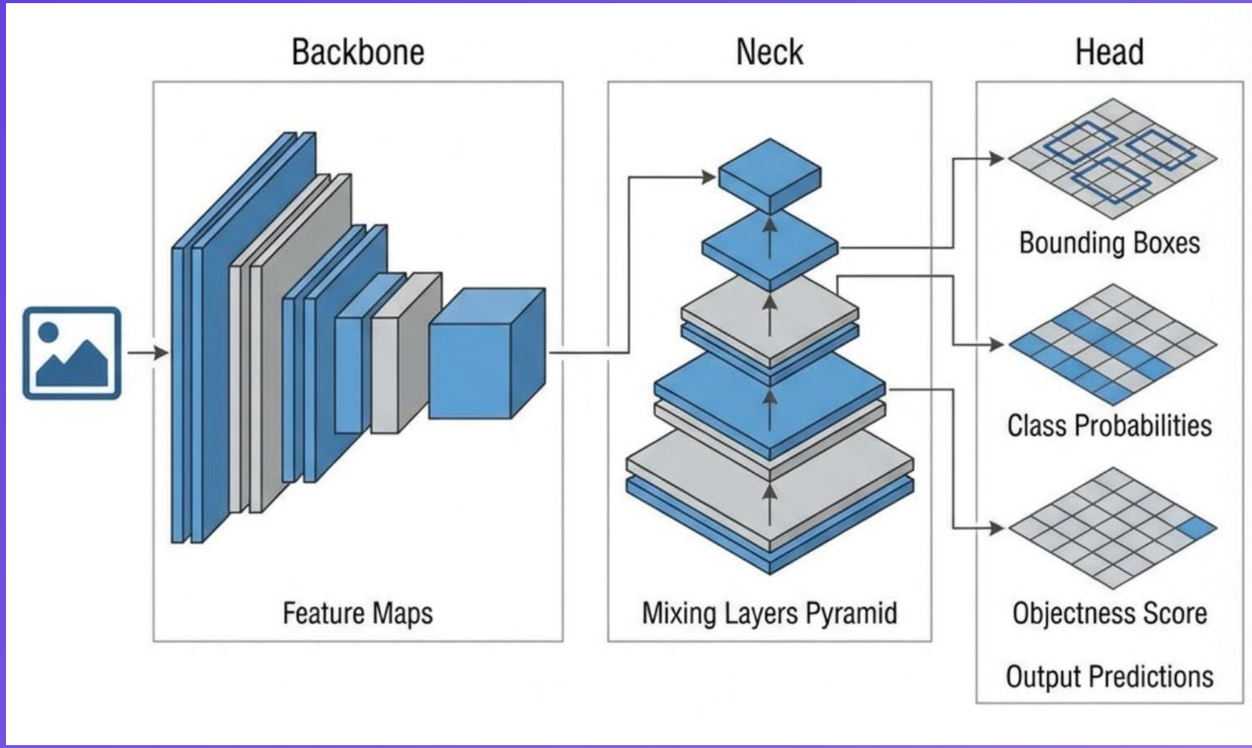
Title: Detection Model – YOLO

- Single-stage detector
- Custom Trained on Fruit Dataset
- 3 YOLO Models used for real-time inference



# Model Architecture (YOLO)

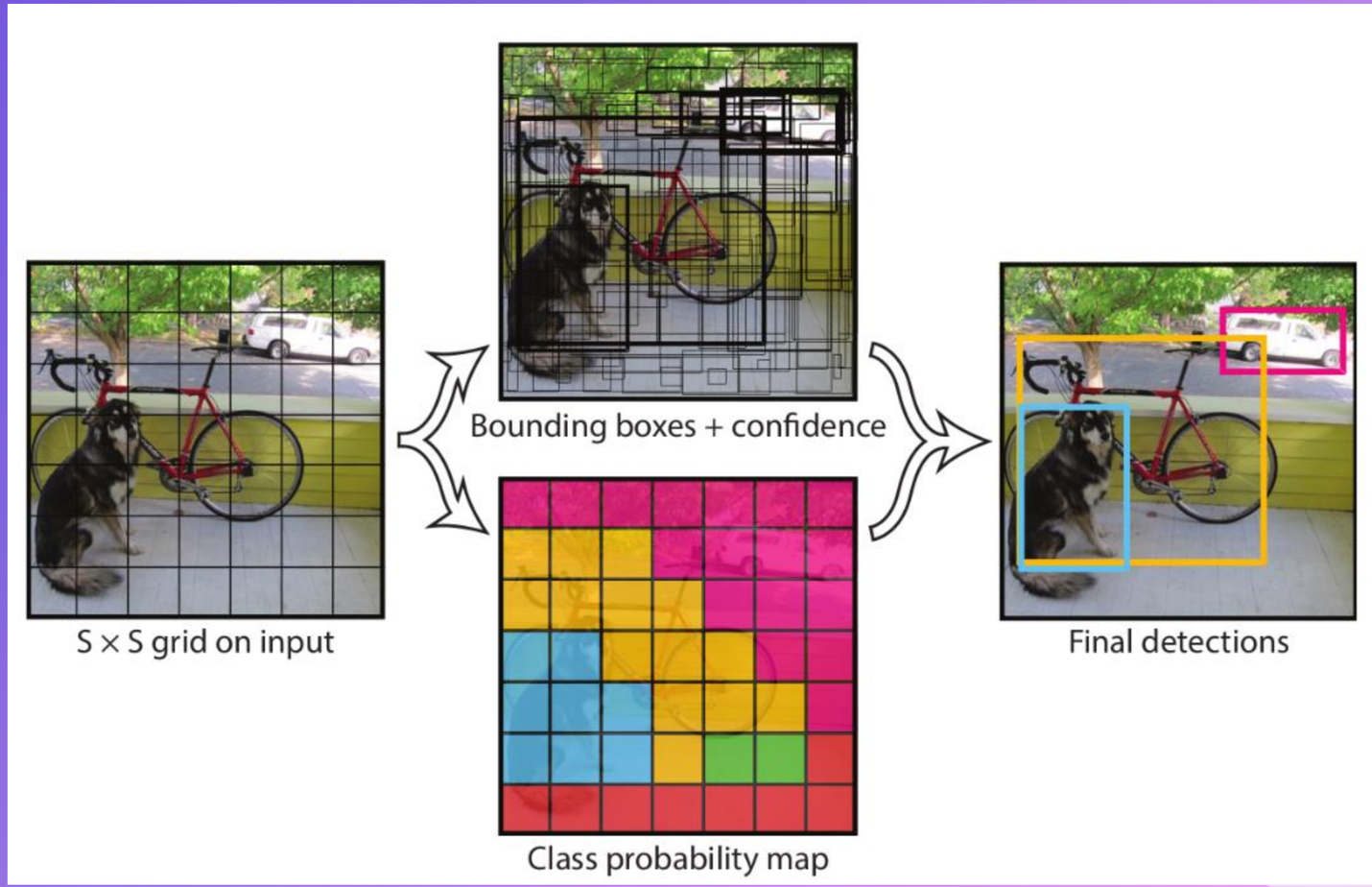
Title: Detection Model – YOLO




- **Backbone** – Feature extraction
- **Neck** – Feature aggregation
- **Head** – Bounding box & class prediction

# How the YOLO Detection System Works ?

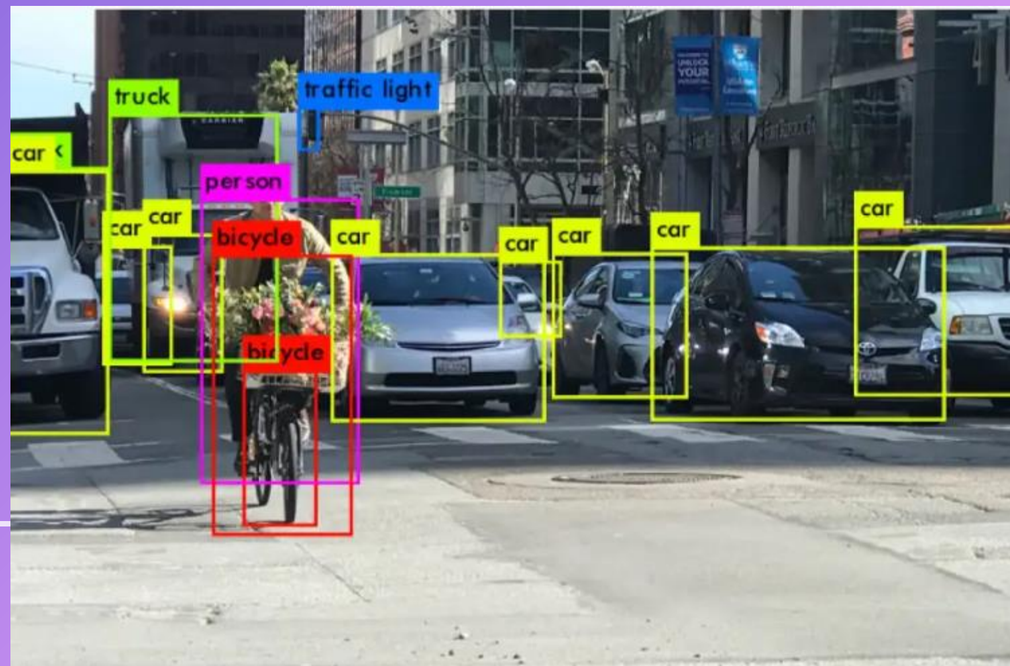
- Image divided into grid cells
- Bounding box and class prediction
- Single forward pass detection



# Model Selection: Why YOLO?

Title: Detection Model – YOLO

- Single-stage detector for real-time performance
- High inference speed with competitive accuracy
- End-to end object detection in one forward pass
- Scalable model sizes (Nano – Medium)
- Strong community & pre-trained weights



# Methodology (Dataset)

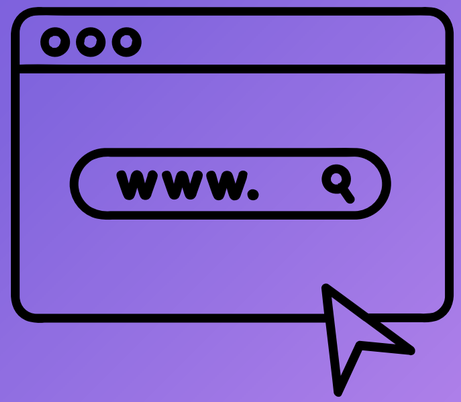
Title: Dataset

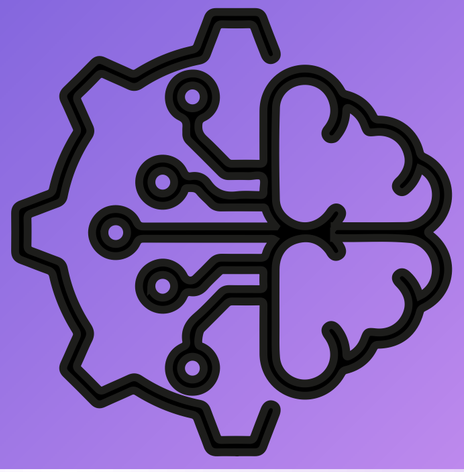
- A Pre – Labeled Dataset from Roboflow
- 9 fruit classes
- 2697 images for training, 187 for validation, 90 for test

# Methodology (Tools)

Title: Tools & Technologies

- Python
- YOLO
- OpenCV
- Streamlit
- PyTorch



# Training Phase

Title : Dataset Configuration

- Custom dataset with **9 fruit classes**
- Dataset paths defined via **data.yaml**

```yaml
# data.yaml
train: ../train/images
val: ../valid/images
test: ../test/images
nc: 9
names: ['Apple', 'Banana', 'Grape', 'Kiwi', 'Mango',
        'Orange', 'Pineapple', 'Strawberry', 'Watermelon']
```

# Training Phase

Title : Annotation Format (Bounding Boxes)

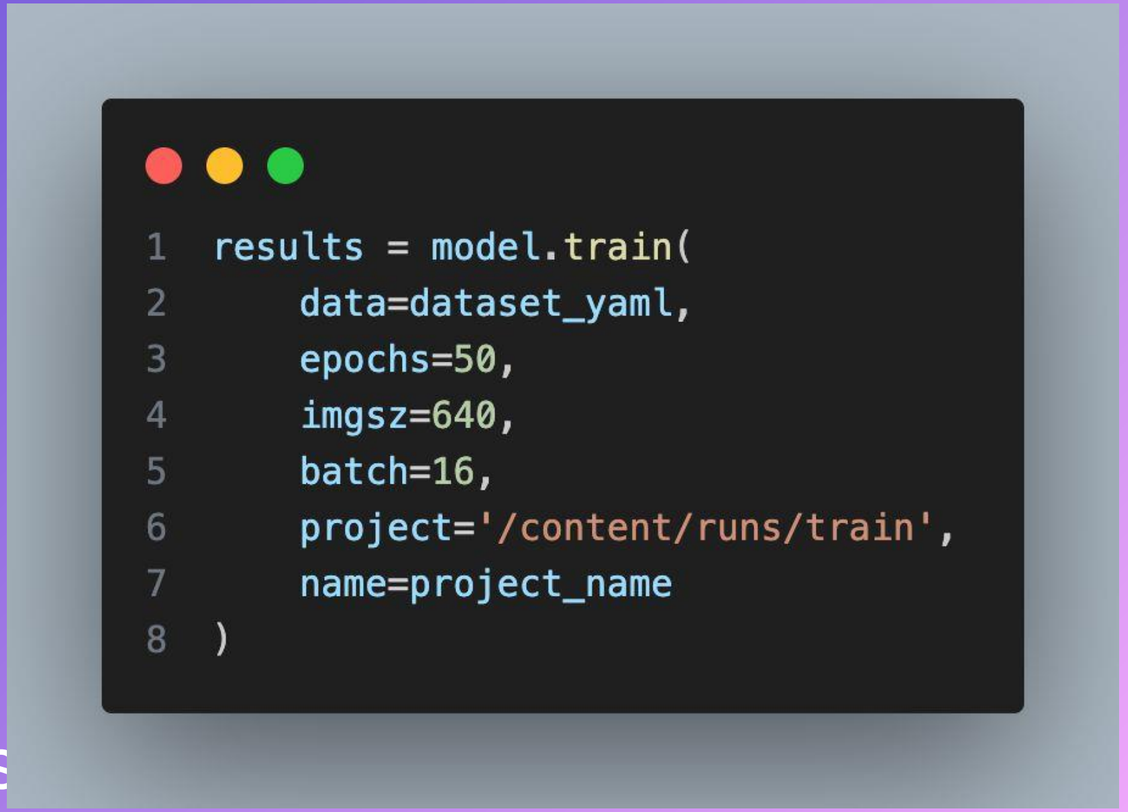
```
# labels/*.txt
# Class_ID  Center_X  Center_Y  Width  Height
0           0.71      0.84      0.15   0.23
4           0.32      0.45      0.10   0.12
```

- Class ID (0) : Represents the object category (Apple,defined in data.yaml)
- Center X (0.71) : Horizontal position of the object’s center
- Center Y (0.84) : Vertical position of the object’s center
- Width (0.15) : Relative width of the bounding box
- Height (0.23) : Relative height of the bounding box

# Training Phase

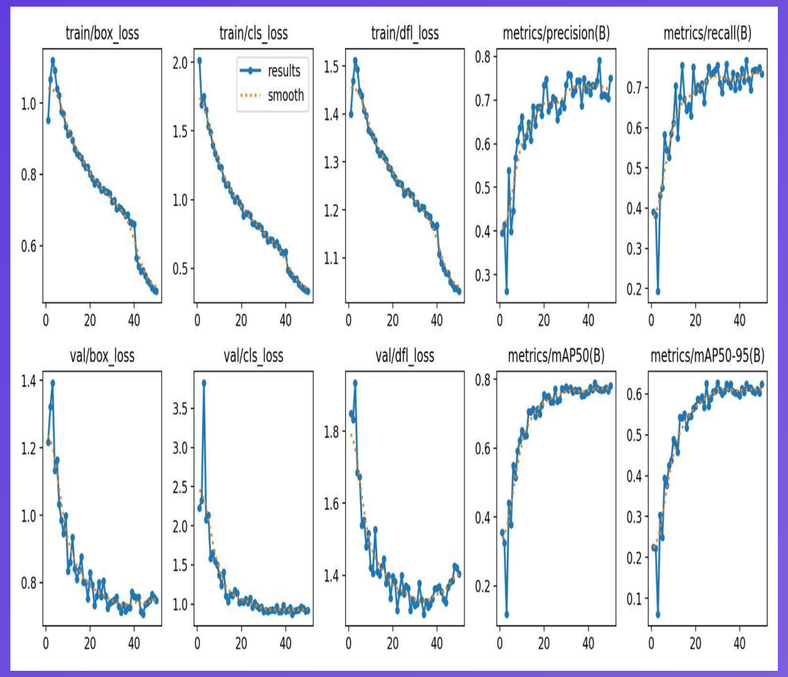
Title : Model Training

- **Training Set (70%)** – used to update model weights
- **Validation Set (20%)** – monitors performance during training
- **Test Set (10%)** – reserved for final evaluation only
- Ensures **generalization** and prevents **overfitting**

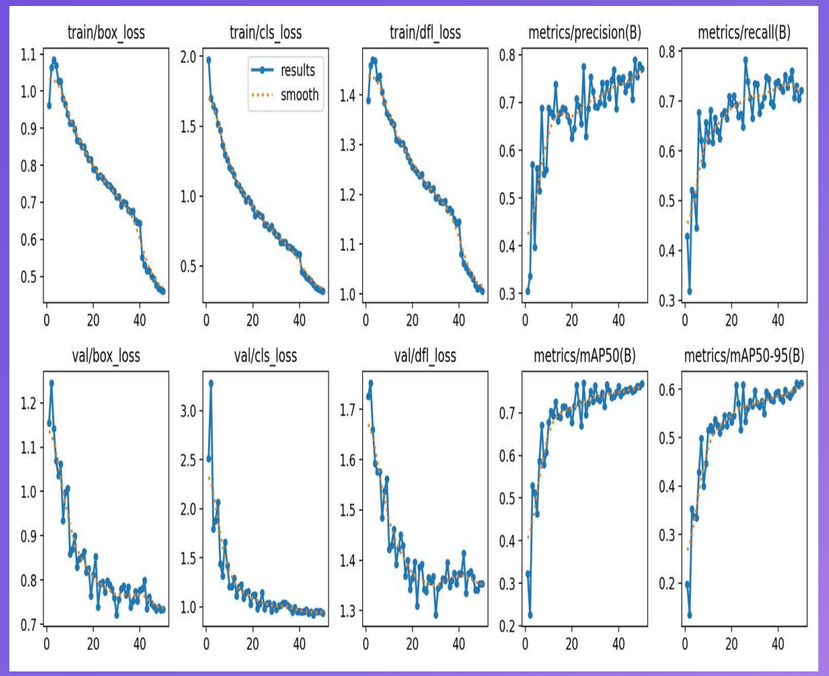


```
results = model.train(
    data=dataset_yaml,
    epochs=50,
    imgsz=640,
    batch=16,
    project='/content/runs/train',
    name=project_name
)
```

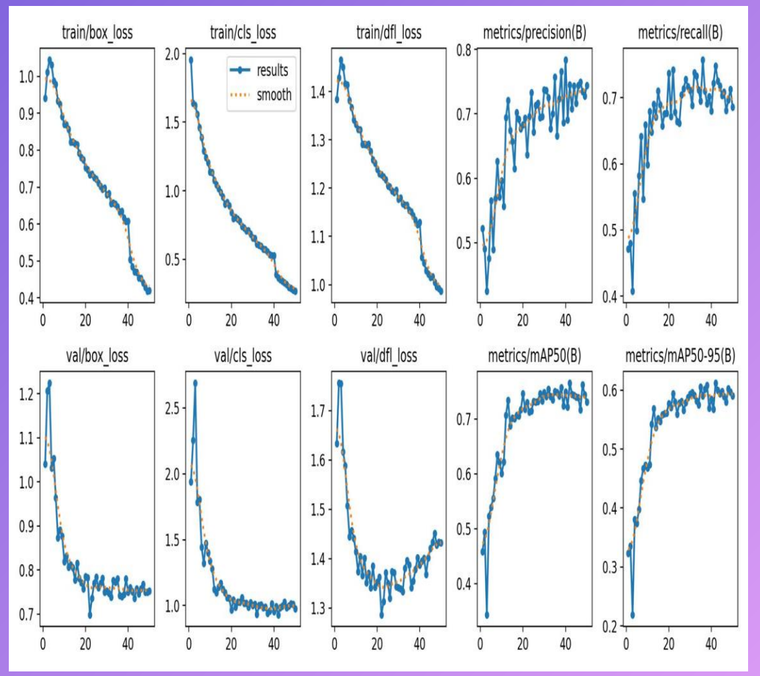
# Results Graphs For Each Model Training




**YOLOv11**



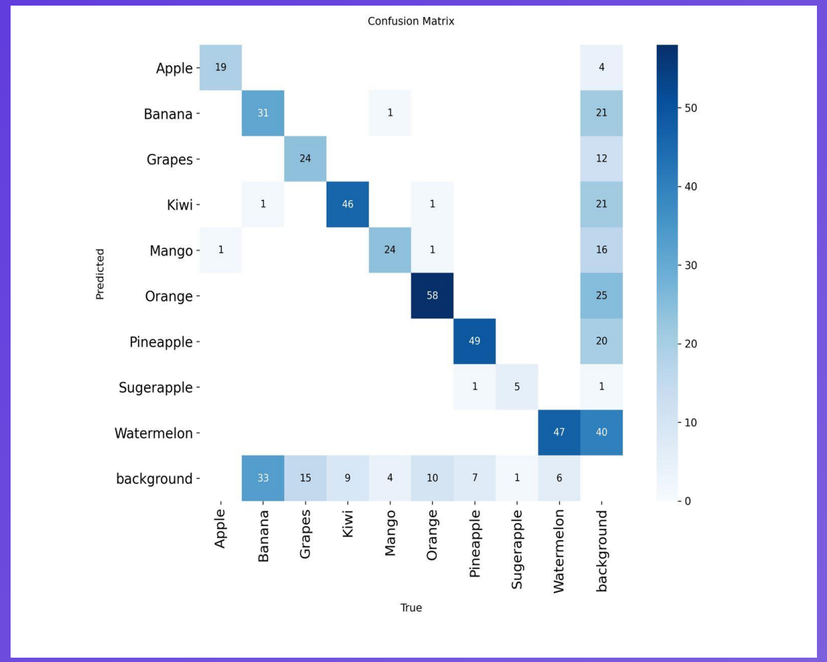

**YOLOv5**



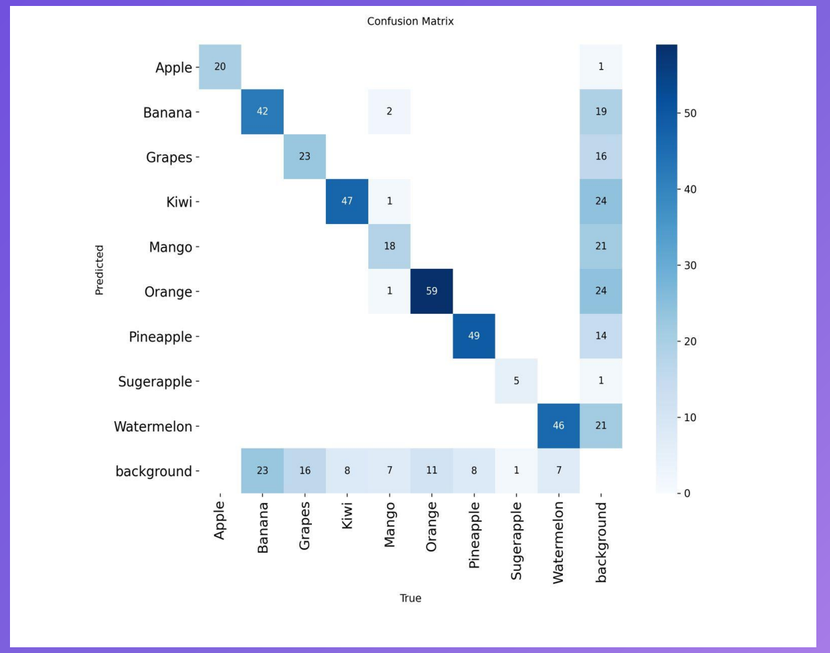

**YOLOv8**

The results demonstrate consistent learning, with steadily decreasing loss and increasing accuracy (mAP) over each training iteration.

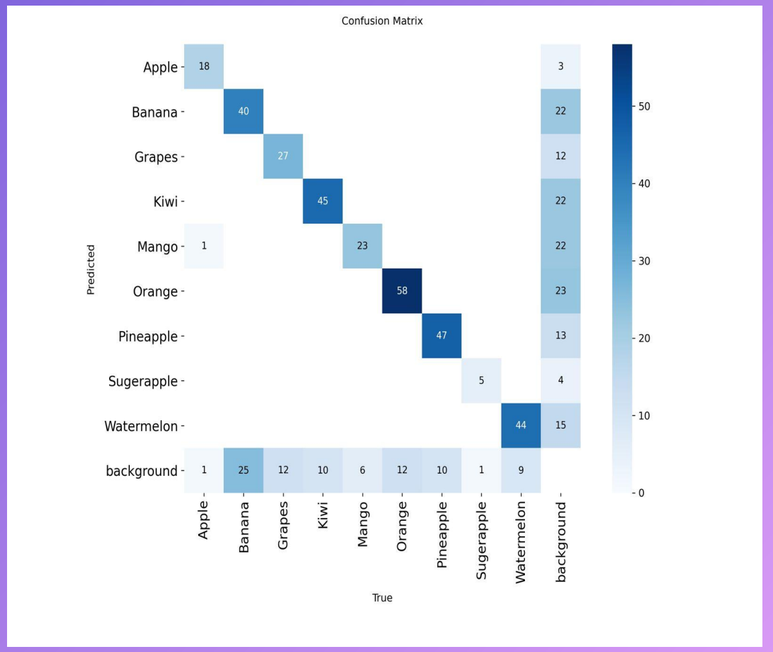
# Confusion Matrixes For Each Model




**YOLOv11**




**YOLOv5**



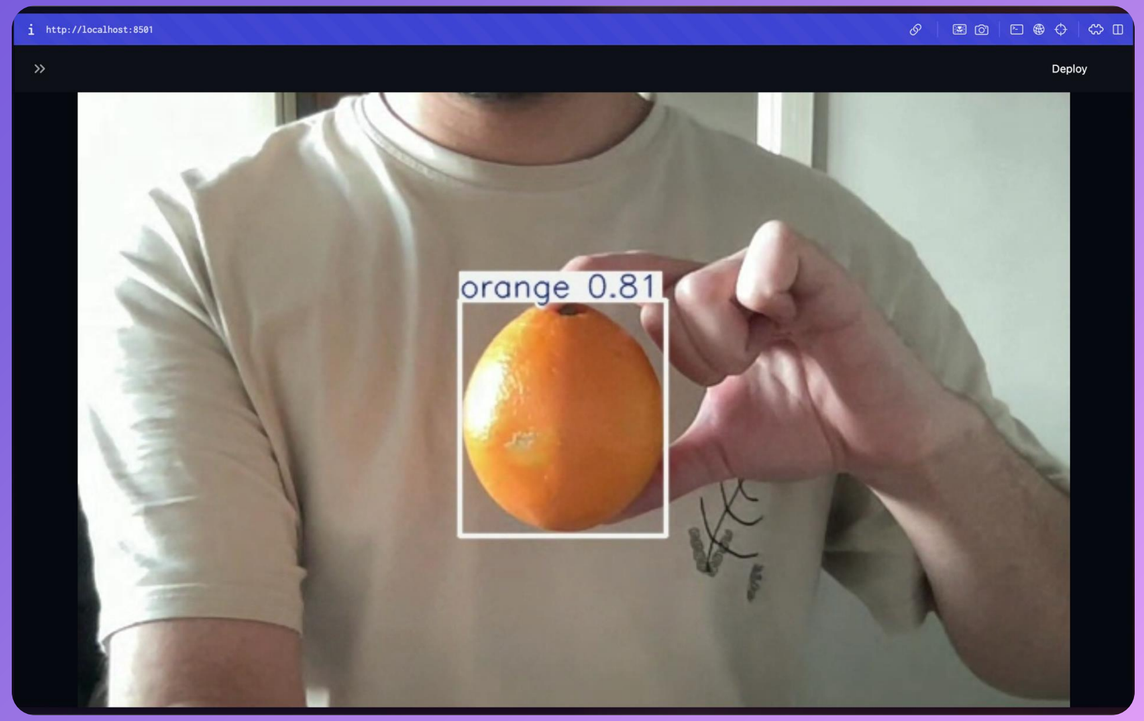

**YOLOv8**

The strong diagonal concentration confirms that the model correctly identifies each fruit class with high accuracy and minimal confusion

# Experiment Results

Title: Experimental Results

- Real-time detection achieved
- Approx. 30 FPS
- Qualitative results available



# Experiment Results (Evaluation – Planned)

Title: Evaluation Metrics

| Model | Precision | Recall | mAP@0.5 | mAP@0.5: 0.95 | Inference |
|---|---|---|---|---|---|
| YOLOv5m | 0.77 | 0.72 | 0.767 | 0.614 | ~10 ms |
| YOLO11m | 0.74 | **0.76** | **0.778** | **0.627** | ~13 ms |
| YOLOv8 | 0.75 | 0.75 | 0.765 | 0.612 | ~11 ms |

YOLOv11m achieved the highest detection accuracy, while YOLOv5m provided the fastest inference time.

# Demo

Title: Live Demo

- Webcam-based detection
- Bounding boxes & confidence scores
- Real-time performance

# Conclusion

Title: Conclusion

- A working real-time prototype was developed
- Stable performance achieved
- Strong foundation for future improvements

# Recommendation (Future Work)

Title: Recommendations & Future Work

- Training with a Bigger Custom Dataset
- 3D Localization
- Mobile App Deployment

# THANK YOU
# For Attention