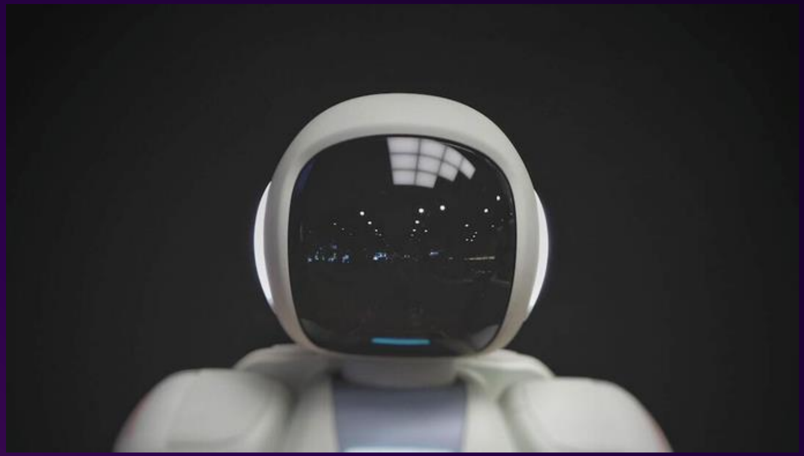
INVESTIGATION OF SOLUTIONS FOR 2D OBJECT DETECTION MODELS WITH ORIENTATION INFORMATION USING YOLO ARCHITECTURE

ZHANGYLAY KALYBAY

AGENDA

1. Introduction
2. Technical advancements
3. Application
4. Architecture
5. MOBILNET
6. YOLOv5 result
7. Further plans

INTRODUCTION TO YOLOV5



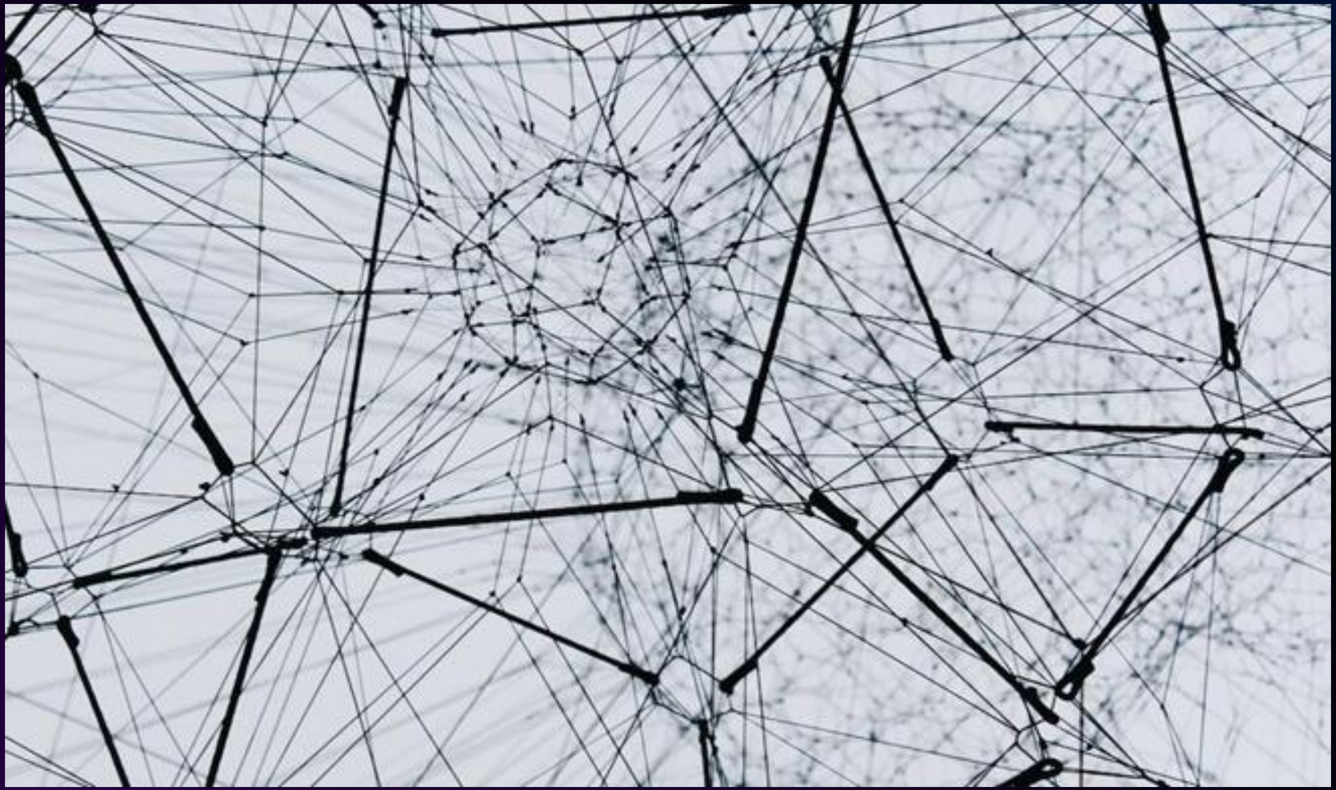
What is Object Detection?: Object detection is a computer vision technique for identifying and locating objects within digital images and videos. It serves as the basis for many applications, from security surveillance to autonomous driving.

Key Concepts: Involves detecting the presence, location, and classification of multiple objects in an image using predefined categories.

Applications: Widely used in face recognition, traffic control systems, and real-time threat detection

TECHNICAL ADVANCEMENTS IN YOLOV5

1. Architecture Improvements: YOLOv5 introduces a more streamlined architecture that increases the inference speed while maintaining high accuracy, making it suitable for real-time applications.
2. Performance Metrics: It excels in both speed and accuracy metrics, achieving impressive benchmarks on standard datasets like COCO, with significant improvements over its predecessors.
3. Model Scalability: The model is designed for scalability and efficiency, offering various sizes (small to large) to accommodate different computational and accuracy needs.



APPLICATIONS OF YOLOV5

Automotive Industry: YOLOv5 is deployed for real-time vehicle and pedestrian detection to enhance safety features in advanced driver-assistance systems (ADAS).

Security and Surveillance: In security, YOLOv5 aids in efficient monitoring and threat detection by recognizing suspicious activities or unauthorized individuals quickly.

Industrial Automation: Factories utilize YOLOv5 for defect detection and assembly line monitoring, significantly reducing errors and increasing production efficiency.

YOLOV5: ARCHITECTURE

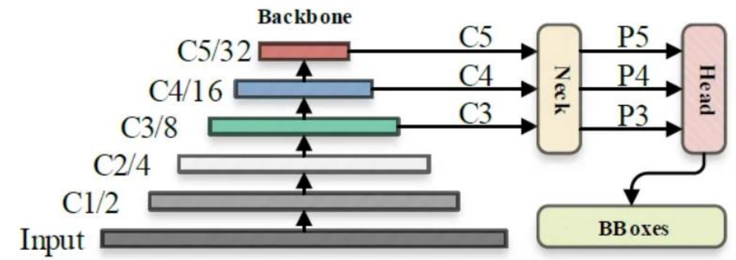


Figure 1. The default inference flowchart of YOLOv5.

YOLOv5: Overall Architecture

1. **Initial Processing:** The image passes through an input layer and a backbone network that extracts varying sizes of feature maps.

2. **Feature Fusion:** These features are fused into three key maps (P3, P4, P5) by the neck network, enabling detection of objects of different sizes.

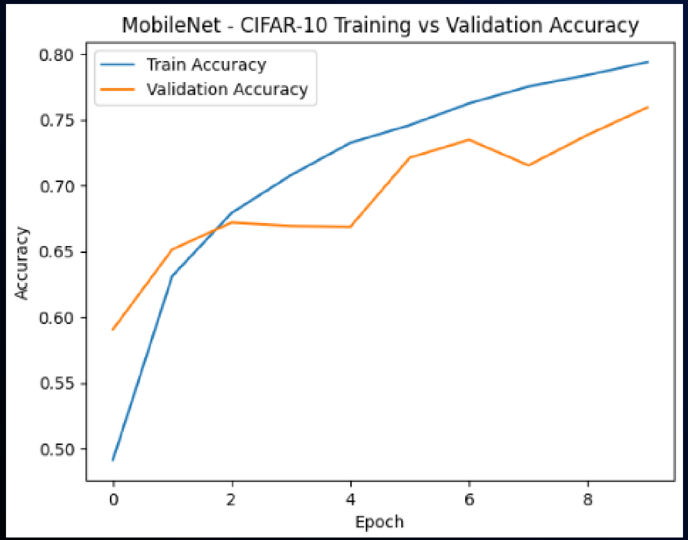
3. **Detection and Analysis:** The feature maps proceed to the prediction head for confidence scoring and bounding box calculations, producing a detailed array of detection data.

4. **Refinement:** Settings filter irrelevant data, and non-maximum suppression (NMS) finalizes the output by eliminating overlapping detections.

MOBILNET

Cifar10

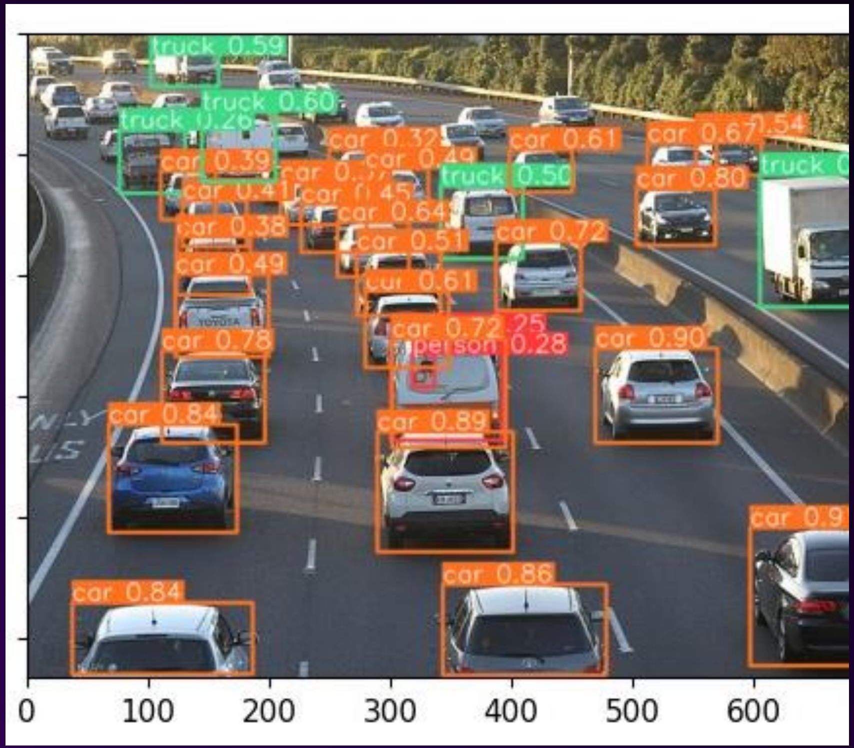
```python
class MobileNet(nn.Module):
    """MobileNet architecture for CIFAR-10."""
    def __init__(self, num_classes=10):
        super(MobileNet, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, kernel_size=3, stride=1, padding=1)

        # First Inverted Residual Block
        self.block1 = nn.Sequential(
            DepthwiseSeparableConv(32, 64, kernel_size=3, padding=1),
            nn.ReLU(inplace=True)
        )

        self.block2 = nn.Sequential(
            DepthwiseSeparableConv(64, 128, kernel_size=3, stride=1, padding=1),
            nn.ReLU(inplace=True)
        )

        self.block3 = nn.Sequential(
            DepthwiseSeparableConv(128, 128, kernel_size=3, padding=1),
            nn.ReLU(inplace=True)
        )

        self.block4 = nn.Sequential(
            DepthwiseSeparableConv(128, 128, kernel_size=3, padding=1),
            nn.ReLU(inplace=True)
        )
        self.avgpool = nn.AdaptiveAvgPool2d((1, 1))
        self.fc = nn.Linear(128, num_classes)

    def forward(self, x):
        x = self.conv1(x)
        x = self.block1(x)
        x = self.block2(x)
        x = self.block3(x)
        x = self.block4(x)
        x = self.avgpool(x)
        x = x.view(x.size(0), -1)
        x = self.fc(x)
        return x
```




```
Epoch 1/10 - Train Acc: 0.491 - Val Acc: 0.591
Epoch 2/10 - Train Acc: 0.631 - Val Acc: 0.651
Epoch 3/10 - Train Acc: 0.679 - Val Acc: 0.672
Epoch 4/10 - Train Acc: 0.708 - Val Acc: 0.669
Epoch 5/10 - Train Acc: 0.732 - Val Acc: 0.669
Epoch 6/10 - Train Acc: 0.746 - Val Acc: 0.721
Epoch 7/10 - Train Acc: 0.762 - Val Acc: 0.735
Epoch 8/10 - Train Acc: 0.775 - Val Acc: 0.715
Epoch 9/10 - Train Acc: 0.784 - Val Acc: 0.739
Epoch 10/10 - Train Acc: 0.794 - Val Acc: 0.759
```


www.werpol.pl
RELOKACJE - MONTAŻE PRZEMYSŁOWE
MAN
KOS 0922E
DAF
WI 460KA
OB 89134

RESULT OF APPLICATION OF YOLOV5



FURTHER PLANS:

- Train the model on a custom dataset
- Calculate all metrics for different hyperparameters
- Add object orientation
- Calculate all metrics & define best result

THANK YOU

- Zhangylay Kalybay
- Janylay.kalybay@gmail.com