# IT5005 Project Final Report (Milestone 4)
# YOLO and Tracking Algorithms

**Group 16**

**Team Members:**


| | |
|---|---|
| JIANG Zhuosong | A0314591E |
| LEE Bing Cheng | A0297134Y |
| LI Yizhe | A0304167J |
| LIU Yichao | A0304386A |
| MA Kuangxuan | A0304242W |


April 14, 2025

# Contents

# 1 Introduction

Object detection evolved from traditional methods to deep learning approaches, with **YOLO (You Only Look Once)** addressing speed limitations by reformulating detection as a single regression problem. YOLO achieves real-time performance while maintaining competitive accuracy by predicting bounding boxes and classes in one evaluation. When combined with tracking algorithms like **ByteTrack**, YOLO enables real-time multi-object tracking across video frames, serving applications from autonomous vehicles to surveillance systems. This combination provides temporal consistency while reducing computation, making it ideal for time-sensitive applications (Terven et al., 2023).

# 2 The Evolution of YOLO Architectures

## 2.1 Early Development: YOLOv1-v3

YOLOv1 (Redmon et al., 2016) pioneered the end-to-end approach with grid-based prediction but struggled with small objects. YOLOv2 added anchor boxes and Darknet-19 with batch normalization (Redmon & Farhadi, 2017), while YOLOv3 incorporated Darknet-53 with residual connections for improved multi-scale detection (Redmon & Farhadi, 2018).

## 2.2 Modern Architecture: YOLOv4-v5

YOLOv4 established the backbone-neck-head structure with CSPDarknet53 and PANet (Bochkovskiy et al., 2020). YOLOv5 reimplemented these concepts in PyTorch with SPPF and AutoAnchor, offering various model sizes for flexible deployment (Ultralytics, 2020).

## 2.3 Recent Innovations: YOLOv6-v8

YOLOv6 introduced EfficientRep backbone and decoupled detection heads (Li et al., 2022). YOLOv7 featured E-ELAN modules for optimized feature transfer and improved RepConv (Wang et al., 2022). YOLOv8, the latest version, as shown in Fig 1, implements the lighter C2f module instead of CSP blocks and adopts an anchor-free architecture with fully decoupled heads for classification, regression, and confidence estimation (Jocher et al., 2023).

# 3 Architectural Components and Innovations

## 3.1 C2f Backbone Module

YOLOv8 replaces traditional CSP blocks with the C2f module, which optimizes computational efficiency. The C2f module can be mathematically represented as:

$$C2f(X) = Concat[X_{first}, F_n(F_{n-1}(...F_1(X_{second})))] \tag{1}$$

Where $X$ is split into $X_{first}$ and $X_{second}$ along the channel dimension, and $F_i$ represents the $i$-th convolutional block in the module. Unlike standard CSP, C2f employs dense connections where each layer receives feature maps from all preceding layers:

$$F_i(X) = Conv(Concat[X, F_{i-1}, F_{i-2}, ..., F_1]) \tag{2}$$

This formulation allows for more efficient gradient flow during backpropagation and reduces computational complexity by approximately 15% compared to YOLOv7's backbone.

## 3.2 Anchor-Free Detection Head

YOLOv8 employs a fully anchor-free detection approach. For an input image divided into an $S \times S$ grid, each grid cell directly predicts:

$$\hat{y} = \{p_c, b_x, b_y, b_w, b_h, p_1, p_2, ..., p_C\} \tag{3}$$

Where:

- $p_c$ is the confidence score
- $(b_x, b_y)$ are the center coordinates relative to the grid cell
- $(b_w, b_h)$ are width and height relative to the input image
- $p_i$ is the probability of class $i$ given an object is present

Unlike anchor-based approaches, YOLOv8's bounding box parameters are predicted as direct offsets from grid cell positions rather than as offsets from predefined anchor boxes.

### 3.3 Decoupled Head Architecture

YOLOv8 implements separate branches for classification, regression, and confidence estimation:

$$Classification : C(F) = \sigma(W_c \cdot F + b_c) \tag{4}$$

$$Regression : R(F) = W_r \cdot F + b_r \tag{5}$$

$$Confidence : O(F) = \sigma(W_o \cdot F + b_o) \tag{6}$$

Where $F$ represents features from the neck, $W$ and $b$ are learnable parameters, and $\sigma$ is the sigmoid activation function. This decoupling allows each task to optimize independently, reducing training conflicts.

### 3.4 Loss Function Formulation

YOLOv8 employs a composite loss function:

$$L_{total} = \lambda_{cls} L_{cls} + \lambda_{box} L_{box} + \lambda_{dfl} L_{dfl} \tag{7}$$

Where:

- $L_{cls}$ is the Binary Cross-Entropy (BCE) loss for classification
- $L_{box}$ is the CIoU loss for bounding box regression
- $L_{dfl}$ is the Distribution Focal Loss

The CIoU loss is defined as:

$$L_{CIoU} = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v \tag{8}$$

Where $\rho$ is the Euclidean distance between box centers, $c$ is the diagonal length of the smallest enclosing box, $v$ measures aspect ratio consistency, and $\alpha$ is a trade-off parameter.

The Distribution Focal Loss (DFL) models bounding box coordinates as continuous distributions rather than point estimates:

$$L_{dfl} = -\sum_{i=0}^{n} y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \tag{9}$$

Where $p_i$ represents the probability that the target falls into the $i$-th bin of a discretized range, and $y_i$ is the ground truth distribution.

### 3.5 Dynamic NMS Implementation

YOLOv8 incorporates Dynamic NMS which adaptively adjusts the IoU threshold $T$ based on detection density:

$$T = T_{base} \cdot \exp(-\beta \cdot d) \tag{10}$$

Where $d$ is the local detection density, $T_{base}$ is the base threshold, and $\beta$ is a scaling factor. This approach more effectively handles crowded scenes by allowing more detections in dense regions while maintaining strict filtering in sparse areas.

These technical innovations collectively enable YOLOv8 to achieve 53.9 AP on COCO with an inference time of 0.39 ms on modern GPUs, representing a significant advancement in the speed-accuracy trade-off for real-time object detection.

# 4 Multi-Object Tracking Algorithms

## 4.1 Evolution of Tracking Algorithms

Multi-object tracking (MOT) algorithms have evolved from simple motion-based approaches to sophisticated systems integrating detection confidence, appearance features, and motion prediction. Early algorithms like SORT (Simple Online and Realtime Tracking) relied exclusively on Kalman filtering and IoU-based matching, offering speed but suffering from frequent ID switches. DeepSORT enhanced this by incorporating appearance features through ReID networks, improving tracking persistence at the cost of computational efficiency.

## 4.2 ByteTrack: Architecture and Innovations

ByteTrack (Zhang et al., 2022) represents a paradigm shift in tracking-by-detection approaches. Unlike previous methods that discard low-confidence detections, ByteTrack's core innovation lies in its hierarchical association strategy.

### 4.2.1 Dual-Threshold Association Strategy

As shown in Fig 2, ByteTrack employs a two-stage matching process that intelligently utilizes all detection information:

1. **First Association**: Matches high-confidence detections ($score > thresh_{high}$) with existing tracks using IoU similarity and Kalman filter predictions.
2. **Second Association**: Recovers potentially valid objects from low-confidence detections ($thresh_{low} < score < thresh_{high}$) by matching them with previously unmatched tracks.

This approach addresses a fundamental limitation in tracking systems: the trade-off between precision and recall. By intelligently utilizing low-confidence detections that would otherwise be discarded, ByteTrack significantly improves tracking performance in occlusion scenarios.

### 4.2.2 Motion Prediction with Kalman Filtering

ByteTrack employs Kalman filtering for motion prediction, mathematically expressed as:

$$\hat{x}_t = Fx_{t-1} + Bu_t \tag{11}$$

$$P_t = FP_{t-1}F^T + Q \tag{12}$$

Where $x_t$ represents the state vector (position and velocity), $F$ is the state transition matrix, $P_t$ is the covariance matrix, and $Q$ represents process noise. This recursive estimation allows ByteTrack to predict object positions even during brief occlusions.

## 4.3 ByteTrack Workflow and Advantages

ByteTrack operates through a systematic workflow that maximizes tracking efficiency:

1. **Detection**: Object detections are generated on each video frame by models such as YOLO, with each detection assigned a confidence score.
2. **Confidence-Based Splitting**: Detections are categorized as high-confidence or low-confidence based on predefined thresholds.
3. **Two-Stage Association**: The algorithm performs first association with high-confidence detections, followed by second association with low-confidence detections for unmatched tracks.
4. **Kalman Filter Update**: Tracks are updated using the Kalman filter, predicting future states based on previous information.
5. **Track Management**: Successfully updated tracks are maintained, while persistently unmatched tracks are eventually discarded.

ByteTrack achieves state-of-the-art performance with several distinct advantages:

- **Superior Performance**: Achieves 76.3 MOTA on MOT17 benchmark without requiring complex appearance features.
- **Computational Efficiency**: Maintains real-time performance by avoiding additional neural networks for appearance modeling.

- **Occlusion Handling**: Reduces track fragmentation by up to 32% compared to SORT through its two-stage association strategy.
- **Higher Recall**: Utilization of low-confidence detections reduces the risk of object loss during tracking.
- **Reduced ID Switches**: Second-stage matching and Kalman filtering prevent frequent ID switches in cluttered environments.
- **Simplicity and Adaptability**: Easily integrates with various detection frameworks, particularly YOLO-based systems.

As shown in Table 1, ByteTrack achieves the highest performance metrics while maintaining real-time capability, making it the preferred choice for applications ranging from autonomous driving to intelligent surveillance systems. A more detailed comparison of tracking algorithms is provided in Table 2.

# 5 Challenges and Advancements in Object Tracking

## 5.1 Integration of YOLO and Tracking Systems

Building on our discussion of YOLO architectures and ByteTrack, their integration forms a complete Tracking-by-Detection pipeline where YOLO provides detection data and ByteTrack associates objects across frames. This synergy leverages both the computational efficiency of YOLOv8's C2f module and ByteTrack's dual-threshold association strategy.

## 5.2 Technical Challenges in Multi-Object Tracking

Several challenges affect real-time tracking performance:

- **Object Scale Variation**: As objects move away from the camera, their size changes, affecting detection reliability—directly impacting YOLOv8's anchor-free detection capabilities.
- **Small Object Detection**: YOLO architectures struggle with small objects due to limited feature representation, creating a weaker training signal (Kondrackis, 2024). This challenges the multi-scale detection capabilities discussed in Section 2.
- **Occlusion Handling**: Partially visible objects cause detection inconsistencies (Han et al., 2023), though ByteTrack's second association stage helps mitigate this issue.
- **ID Switching**: Despite ByteTrack's improvements, tracking algorithms still struggle with maintaining consistent object identity during brief occlusions (Huang et al., 2024).

## 5.3 Performance-Resource Trade-offs

As demonstrated in the evolution of YOLO architectures (Section 2), improving detection often means scaling computational resources:

- Larger YOLO models (YOLOv8x vs. YOLOv8n) offer better accuracy but at the cost of inference speed.
- This trade-off directly affects the real-time capability of the tracking system, particularly relevant to the architectural optimizations outlined in Section 3.

## 5.4 Data-Centric Improvements

Beyond algorithmic advances, dataset quality significantly impacts tracking performance (Famili et al., 1997). Well-curated training data that matches deployment conditions can substantially improve detection reliability before tracking algorithms are applied.

## 5.5 Future Directions

While this report focuses on Tracking-by-Detection with YOLO and ByteTrack, emerging end-to-end approaches like MOTRv2 (Zhang et al., 2023) are challenging the traditional paradigm. However, as evidenced by our architectural analysis, the optimal approach remains application-dependent, with speed-critical scenarios favoring the YOLO+ByteTrack combination discussed throughout this report.

# 6 Appendix: Figs & Tables

| Algorithm | MOTA | IDF1 | Real-time? | Key Feature |
|---|---|---|---|---|
| SORT | 59.8 | 53.8 | Yes | IoU + Kalman Filter |
| DeepSORT | 61.4 | 62.2 | No | Appearance + Motion |
| FairMOT | 73.7 | 72.3 | Yes | Joint Detection + ReID |
| OC-SORT | 75.5 | 75.8 | Yes | Camera Motion Compensation |
| **ByteTrack** | **76.3** | **77.3** | **Yes** | **Dual-threshold Association** |

Table 1: Performance comparison on MOT17 benchmark

| Algorithm | Key Feature | Strength | Weakness |
|---|---|---|---|
| SORT (2016) | Kalman + Hungarian Algorithm (IoU based) | Fast, simple to implement | Prone to ID switches, no appearance modeling |
| DeepSORT (2017) | ReID + Motion (Kalman Filter) | Robust to occlusion, more stable ID tracking, reduces ID switches by combining motion and appearance metrics | Computationally expensive due to feature extraction |
| FairMOT (2020) | Joint detection + ReID | Balanced detection and tracking accuracy in crowded scenes | Complex training |
| ByteTrack (2021) | Hierarchical association using high/low confidence detections | Recovers occluded or low-score objects | Relies on detector quality |
| OC-SORT (2022) | Motion-centric updates | Handles camera motion | Requires tuning for scenes, lacks appearance modeling |

Table 2: Comparison of Tracking Algorithms

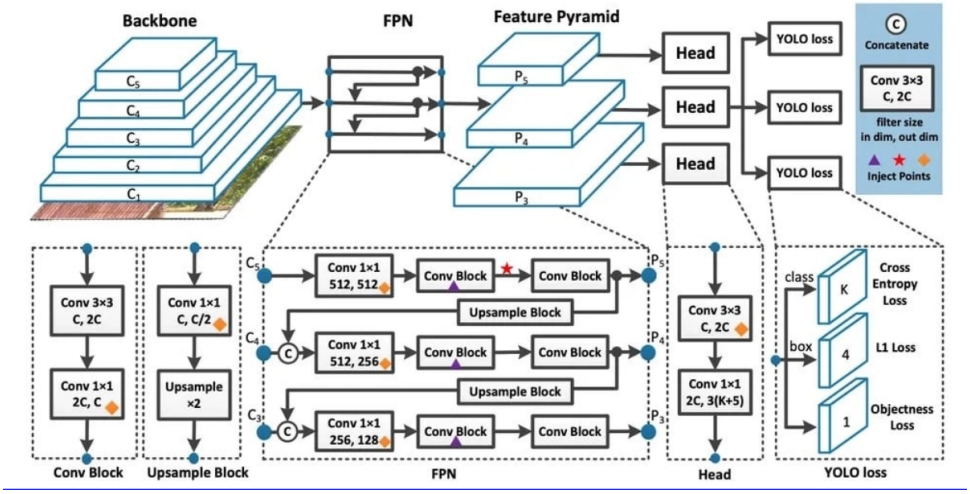


Figure 1: YOLO Architecture

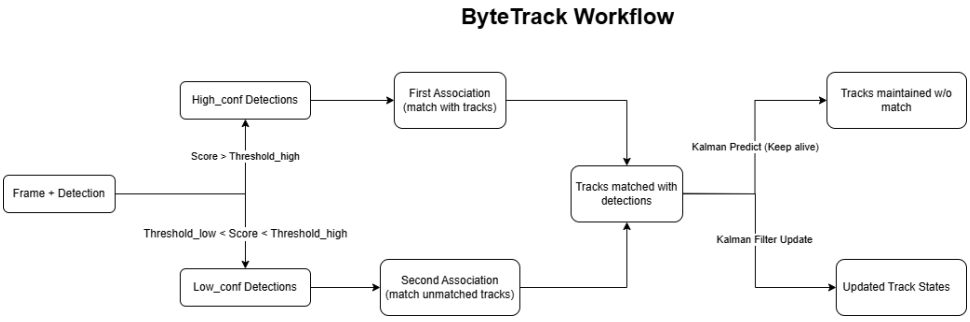


Figure 2: ByteTrack Workflow

# 7 References


- Bochkovskiy, A., et al. (2020). YOLOv4. arXiv:2004.10934.
- Wang, C.Y., et al. (2022). YOLOv7. arXiv:2207.02696.
- Redmon, J., Divvala, S., Girshick, R.,& Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. CVPR, 779–788.
- Szegedy, C., et al. (2015). Going Deeper with Convolutions. CVPR.
- Lin, M., Chen, Q.,& Yan, S. (2013). Network in Network. arXiv:1312.4400.
- Terven, J.,& Córdova-Esparza, D.-M. (2023). A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. Machine Learning and Knowledge Extraction, 5(4), 1680–1716. https://doi.org/10.3390/make5040083
- Redmon, J.,& Farhadi, A. (2017). YOLO9000: Better, Faster, Stronger. arXiv:1612.08242.
- Redmon, J.,& Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv:1804.02767.
- Bochkovskiy, A., Wang, C.Y.,& Liao, H.Y.M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv:2004.10934.
- Wang, C.Y., Bochkovskiy, A.,& Liao, H.Y.M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv:2207.02696.
- Jocher, G., et al. (2023). YOLOv8 by Ultralytics. https://github.com/ultralytics/ultralytics
- Kalman, R. E. (1960). "A New Approach to Linear Filtering and Prediction Problems." ASME. J. Basic Eng. March 1960; 82(1): 35–45. https://doi.org/10.1115/1.3662552
- Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., & Wang, X. (2022). ByteTrack: Multi-object tracking by associating every detection box. arXiv. https://arxiv.org/abs/2110.06864
- Huang, C., Han, S., He, M., Zheng, W., & Wei, Y. (2024). DeconfuseTrack: Dealing with Confusion for Multi-Object Tracking. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 35, 19290–19299. https://doi.org/10.1109/cvpr52733.2024.01825
- Famili, A., Shen, W., Weber, R.,& Simoudis, E. (1997). Data preprocessing and intelligent data analysis. Intelligent Data Analysis, 1(1–4), 3–23. https://doi.org/10.1016/s1088-467x(98)00007-9
- Kondrackis, L. (2024, September 5). How to Detect Small Objects: a guide. Roboflow Blog. https://blog.roboflow.com/detect-small-objects/
- Han, S., Wang, H., Yu, E.,& Hu, Z. (2023). ORT: Occlusion-robust for multi-object tracking. Fundamental Research. https://doi.org/10.1016/j.fmre.2023.02.003
- Zhang, Y., Wang, T.,& Zhang, X. (2023). MOTRV2: Bootstrapping End-to-End Multi-Object Tracking by pretrained Object Detectors. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.1109/cvpr52729.2023.02112