Enhancing Real-Time Detection Transformer (RT-DETR) for Handgun Detection on Nvidia Jetson

Luis A. Bustamante
Universidad Nacional de San Agustin, Escuela de Ciencia de la Computación ,
Arequipa, Perú, 04001,
lbustamantet@unsa.edu.pe

and

Juan C. Gutiérrez
Universidad Nacional de San Agustin, Escuela de Ciencia de la Computación,
Arequipa, Perú, 04001,
jgutierrezca@unsa.edu.pe

Abstract

Recent studies have highlighted the rise in violence and criminal activities, particularly those involving firearms. In response to the growing demand for effective firearm detection systems in public safety applications, this paper explores advancements in real-time object detection using Transformer-based models. Building on the RT-DETR architecture and its latest version, RT-DETR v2, we introduce modifications such as the Bidirectional Feature Pyramid Network (BiFPN), achieving an improvement of over 17% in small object detection specifically firearms compared to previous work. Additionally, we employ dynamic batch processing to maximize computational efficiency and resource utilization on edge devices like the Nvidia Jetson AGX Xavier, enabling efficient real-time deployment. We also compare our model with state-of-the-art alternatives such as YOLOv10, demonstrating the superior accuracy and performance of Transformer-based models. For the comparative study, we use a benchmark that includes three datasets with challenging conditions. The code and trained models are available at `https://github.com/labt1/GunDetection-RTDETR`.

Keywords: Transformers, RT-DETR, Gun Detection, Edge Computing, Nvidia Jetson.

1 Introduction

This paper is an extension of our previous work presented at the CLEI conference [1], where we introduced a real-time handgun detection system based on the Real-Time Detection Transformer (RT-DETR) architecture [2]. Our prior implementation demonstrated the feasibility of deploying Transformer-based models on embedded devices, achieving state-of-the-art performance. However, several challenges remained unaddressed, particularly in detecting small firearms and evaluating the model under broader and more realistic datasets.

Firearm-related crimes have been on the rise in recent years. According to the latest survey conducted by INEI in 2024, between May and October, the percentage of crimes involving firearms in urban areas increased to 27.5%, reflecting a 2.5% rise from 2023. Firearms continue to be the most commonly used weapons in criminal acts across all contexts, as illustrated in Fig. 1. These alarming statistics underscore the urgent need for robust and efficient weapon detection systems, particularly in real-time surveillance applications.

Traditional object detection methods, such as Faster R-CNN and YOLO, have achieved significant advancements in real-time applications. However, these models still face notable limitations when applied to firearm detection in surveillance scenarios. Faster R-CNN, as a two-stage detector, provides high accuracy but suffers from increased inference time, making it impractical for real-time deployment. On the other hand, YOLO-based architectures have demonstrated faster inference speeds but often struggle with small object detection and occlusion, which are critical challenges in crime scene surveillance footage.

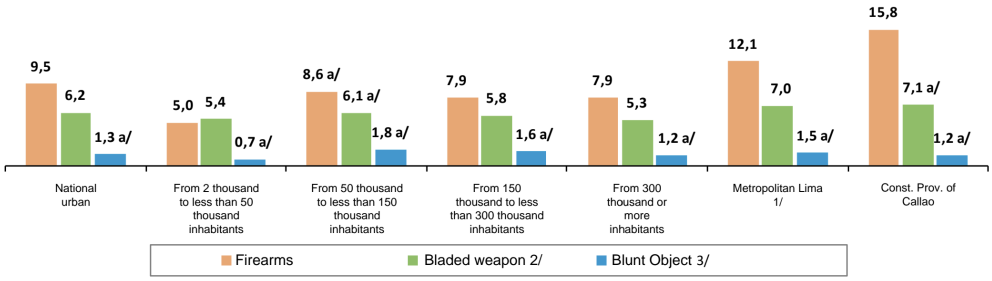


Figure 1: Victims of criminal acts committed with some weapon, according to the size of urban population centers [3].

Transformer-based object detectors, such as DETR and its variants, have emerged as promising alternatives by leveraging self-attention mechanisms to capture long-range dependencies and improve feature representation. RT-DETR, in particular, was designed as an end-to-end real-time detection model that eliminates the need for Non-Maximum Suppression (NMS), leading to faster and more efficient inference. However, its performance in handgun detection, particularly on edge devices, has not been extensively explored.

In this paper, we utilize an improved version of RT-DETR, previously used in our work: RT-DETR v2. This version enhances the accuracy of RT-DETR. Additionally, we modify the backbone by incorporating a Bidirectional Feature Pyramid Network (BiFPN) [4] to improve small object detection. The study in [5] achieved better results by integrating this modification. Firearms, being small objects, are often challenging to detect in surveillance footage. We also adopt the benchmark proposed in [6], which includes diverse datasets covering crime scenes, surveillance recordings, and general-purpose action detection contexts, ensuring a comprehensive evaluation of our approach.

Furthermore, we evaluate the deployment of our improved RT-DETR model on the Nvidia Jetson AGX Xavier, optimizing its performance using TensorRT to achieve high-speed inference while maintaining accuracy. Our experiments analyze critical metrics such as detection precision, inference speed, and throughput at varying batch sizes, demonstrating the feasibility of deploying RT-DETR v2 for real-time firearm detection in edge computing environments.

Despite significant advances in deep learning for object detection, existing approaches—particularly CNN-based models such as Faster R-CNN and YOLO—continue to face challenges in firearm detection. These models struggle with detecting small objects like weapons, especially in surveillance scenarios where occlusion and varying lighting conditions further complicate the task. Transformer-based detectors, such as DETR, ViT, and their variants, have shown promise in addressing these limitations by leveraging global attention mechanisms. However, their effectiveness when deployed on edge devices for real-time applications remains insufficiently explored. To bridge this gap, this work evaluates the performance of RT-DETR and its improved version, RT-DETR v2, along with a modified variant incorporating a BiFPN, which has been successfully applied in other small object detection tasks to enhance firearm detection. A comparative analysis is conducted against state-of-the-art models to assess their effectiveness in this domain. The following section reviews existing object detection approaches, focusing on CNN-based methods and recent advancements in transformer-based architectures applied to firearm detection.

2 Related Work

Object detection has evolved from traditional computer vision techniques to modern deep learning approaches. Convolutional Neural Networks (CNNs) have dominated the field, offering robust solutions for detecting objects in images and videos. However, recent advancements in Transformer-based architectures have demonstrated significant improvements in performance and scalability, especially for complex tasks like real-time detection. Specifically, handgun detection has emerged as an application of these technologies, addressing security concerns by leveraging state-of-the-art models tailored to the unique challenges of weapon detection in surveillance scenarios. This section reviews the progress in object detection using CNNs and Transformers, as well as prior research focused on handgun detection.

2.1 Object Detection With CNN

2.1.1 Two Stage Detectors

Two-stage object detectors divide the detection task into two sequential steps: generating Regions of Interest (RoI) and classifying the content within each region.. This methodology faces significant challenges due to the variability in the spatial locations and aspect ratios of RoIs. As a result, a large number of candidate regions must be analyzed, leading to high computational costs and latency, particularly for real-time applications.

The introduction of Regions with Convolutional Neural Networks (R-CNN) [7] marked a significant milestone in object detection. R-CNN employed a selective search algorithm to generate approximately 2000 candidate RoIs per image, which were then individually classified using a Convolutional Neural Network (CNN). Despite achieving high detection accuracy, R-CNN was computationally expensive, requiring up to 47 seconds per image, making it impractical for real-time scenarios.

Building on this, Fast R-CNN [8] optimized the process by performing convolutional operations once for the entire image, generating a shared feature map. RoIs were then projected onto this feature map for classification, reducing the processing time to approximately 2.3 seconds per image. Additionally, Fast R-CNN introduced the RoI pooling layer to normalize RoI sizes for consistent processing. However, it still relied on the selective search algorithm, which remained a bottleneck for speed.

Faster R-CNN [9] eliminated the dependency on selective search by introducing a Region Proposal Network (RPN) that learned to generate RoIs directly from feature maps. This integration of region proposal and classification within a single network significantly improved both speed and accuracy, establishing Faster R-CNN as a foundational model in object detection. Further improvements were made with the incorporation of Feature Pyramid Network (FPN) [10], which enhanced the model's ability to detect objects at multiple scales by effectively combining features from different levels of the network hierarchy. This advancement was particularly beneficial for detecting small objects, as it preserved high-resolution details.

2.1.2 One Stage Detectors

The You Only Look Once (YOLO) algorithm [11] revolutionized object detection by introducing a single convolutional network that simultaneously predicts bounding boxes and class probabilities. While the original YOLO achieved impressive speeds of up to 45 frames per second (FPS), it struggled with detecting small objects due to its fixed grid structure and limited feature resolution. Subsequent versions addressed these limitations. YOLOv3 [12] introduced a FPN to enable multi-scale object detection, and YOLOv4 [13] improved upon this with advanced feature integration techniques, achieving better performance in challenging scenarios.

EfficientDet [4] introduced a Bidirectional Feature Pyramid Network (BiFPN), which optimizes multi-scale feature fusion using a weighted, bi-directional structure. This design enhances the detection of small objects by effectively integrating high-resolution features while maintaining computational efficiency. BiFPN's success in EfficientDet underscores its potential for improving small object detection in real-time systems.

YOLOv5 [14] further improved efficiency by prioritizing small object detection through a "small objects first". Similarly, YOLOv7 [15] optimized the trade-off between speed and accuracy, introducing innovations that reduced parameters and computational requirements. YOLOv8 [16] adopted an anchorless architecture, eliminating predefined anchor boxes for better adaptability to varying object sizes and proportions, further enhancing precision and efficiency.

YOLOv9 [17] introduces the concept of Programmable Gradient Information (PGI), enabling better utilization of input information for objective function calculations and improving gradient updates, achieving high parameter efficiency without relying on depth-wise convolutions.

YOLOv10 [18] eliminates reliance on Non-Maximum Suppression (NMS) through the introduction of consistent dual assignments for NMS-free training, this innovation reduces inference latency. YOLOv10 also employs a holistic efficiency-accuracy driven design strategy, optimizing various architectural components to reduce computational overhead while maintaining or improving accuracy.

2.2 Object detection with Transformers

Vision Transformers (ViT) [19] marked a significant shift in computer vision by introducing Transformer [20] architectures, originally designed for natural language processing, to image recognition. Address the limitations of ViT by introducing a hierarchical architecture with shifted windows. Swin Transformers [21], divide images into non-overlapping windows and compute attention within each window, reducing computational overhead, enhancing both efficiency and scalability for high-resolution inputs.

DETR [22] (DEtection TRansformer) introduced an end-to-end approach to object detection by framing it as a set prediction problem. DETR utilizes bipartite matching to directly predict object instances

without relying on traditional anchors or non-maximum suppression. Despite its simplicity, DETR requires substantial computational resources and long training times to converge, particularly for large-scale datasets.

Deformable DETR [23] addresses the limitations of DETR by incorporating deformable attention mechanisms. This modification enables the model to focus on a sparse set of key features across the image, significantly improving training efficiency and convergence speed while maintaining competitive performance. Deformable DETR is particularly effective for tasks requiring high-resolution feature aggregation.

2.3 Handgun detection

The works [24, 25] implemented alarm systems using Faster R-CNN with a VGG16 backbone for weapon detection. Their system achieved an F1-score of 91.43% but was limited to 5.3 FPS on an Nvidia GTX 980, highlighting its unsuitability for real-time applications. In [26] improved upon this by incorporating an FPN and a ResNet-50 backbone, achieving a mean Average Precision (mAP) of 68.3% and 11 FPS on an Nvidia GTX 1080. However, the reliance on synthetic datasets emphasized the need for evaluations in real-world contexts.

YOLO-based architectures have shown significant promise for real-time weapon detection. Bhatti et al. [27] demonstrated the effectiveness of YOLOv3 over Faster R-CNN for handgun detection, achieving faster inference using a custom dataset. The work [28] compared YOLOv3 and YOLOv4, highlighting YOLOv4's superior sensitivity and processing time for diverse weapon sizes and backgrounds. In [29] further advanced this line of research with YOLOv5, achieving 98.56% accuracy at 33 FPS on an Nvidia Jetson AGX Xavier, demonstrating its suitability for edge computing.

Recent innovations have extended YOLO architectures to tackle specific challenges. The work [30] proposed YOLOv7-DarkVision, integrating a brightening algorithm for nighttime weapon detection, achieving an F1-score of 93.41%. Using quantized YOLOV8, the work [31] reduces inference time by 15% while maintaining a mAP of 90.1%, demonstrating the potential of lightweight models for real-time surveillance.

Alternative approaches have also been explored. In [32] utilized CenterNet on HiSilicon embedded devices, combining a two-stage classification and tracking mechanism to reduce false positives, achieving 11 FPS but with limited accuracy (40.0% mAP). The work [6] introduced the CCTV-Gun benchmark, focusing on handgun detection in challenging CCTV scenarios. Their benchmark provides a comprehensive evaluation framework for detectors, considering factors such as blur, occlusion, and diverse real-world conditions. While significant progress has been made, challenges persist in detecting small objects like firearms, maintaining high precision in diverse environments, and ensuring real-time performance.

3 Real Time Detection Transformer RT-DETR

Transformers, introduced by [20], revolutionized sequence modeling tasks and were originally designed for natural language processing. Their adaptability has recently extended to the domain of computer vision, enabling groundbreaking advancements in object detection. This shift is exemplified by models like DETR [22], which depart from traditional detection pipelines by adopting an end-to-end framework. Despite its innovative approach, DETR suffers from limitations in real-time performance, largely due to its computational complexity and lengthy convergence time during training. These challenges have motivated the development of more efficient variants, including real-time detection frameworks that adapt Transformer architectures for faster inference without compromising accuracy. RT-DETR builds upon these advancements, addressing key bottlenecks in speed and efficiency, making it particularly suitable for real-time object detection tasks.

RT-DETR [2] is the first real-time end-to-end object detector, and by generating predictions directly, it greatly reduces the post-processing time. In Fig. 2, it can be seen that it's outperforming current state-of-the-art YOLO detectors of the same scale in both speed and accuracy.

3.1 End-to-end Object Detector and Analysis of NMS

Traditional object detection models, such as Faster R-CNN or YOLO, (except YOLOV10), rely on a multi-stage pipeline that includes RPN, RoI pooling, and post-processing steps like NMS to filter redundant detections. In contrast, RT-DETR reframes object detection as a direct set prediction problem, where a fixed number of object queries are used to predict the presence and location of objects in an image.

This approach simplifies the detection process by removing hand-crafted components and relying solely on the Transformer architecture to model global dependencies. The model directly outputs a fixed set of predictions $\{(\mathbf{b}_i, \mathbf{c}_i)\}_{i=1}^{N}$, where $\mathbf{b}_i$ represents the bounding box coordinates and $\mathbf{c}_i$ represents the class probabilities. A bipartite matching loss is employed during training to associate predicted boxes with ground truth annotations, defined as Eq. (1):

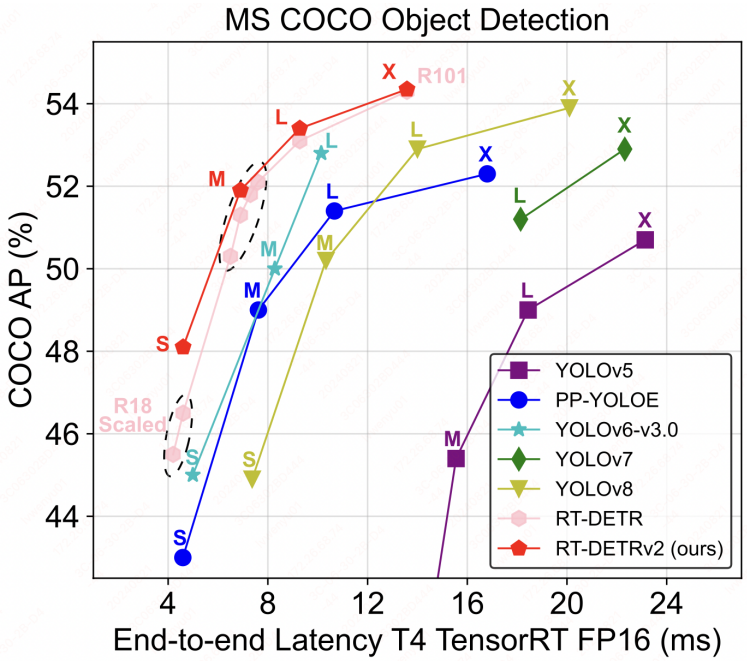


Figure 2: Performance Comparison of Real-Time Object Detectors on an Nvidia T4 GPU at FP16 Precision [2]

$$\mathcal{L}_{match} = \min_{\sigma \in \Sigma_N} \sum_{i=1}^{N} \mathcal{L}_{det}(\hat{\mathbf{b}}_{\sigma(i)}, \mathbf{b}_i) + \mathcal{L}_{cls}(\hat{\mathbf{c}}_{\sigma(i)}, \mathbf{c}_i) \tag{1}$$

Where Σ_N represents all possible permutations of N predictions, $\mathcal{L}_{det}$ is the box regression loss, and $\mathcal{L}_{cls}$ is the classification loss.

By eliminating NMS, RT-DETR avoids these pitfalls. Instead, the model's set-based prediction ensures that each object is detected exactly once, removing the need for redundancy filtering. This is achieved by the inherent design of the Transformer decoder, which assigns each object query to a unique detection through the bipartite matching mechanism.

3.2 Architecture

RT-DETR is an advanced object detection framework designed for real-time applications, leveraging the efficiency of Transformers while maintaining high detection accuracy. The architecture, Fig. 3, consists of three main components: the backbone, the encoder-decoder Transformer, and the prediction head. Below, we detail each stage from input processing to the final output.

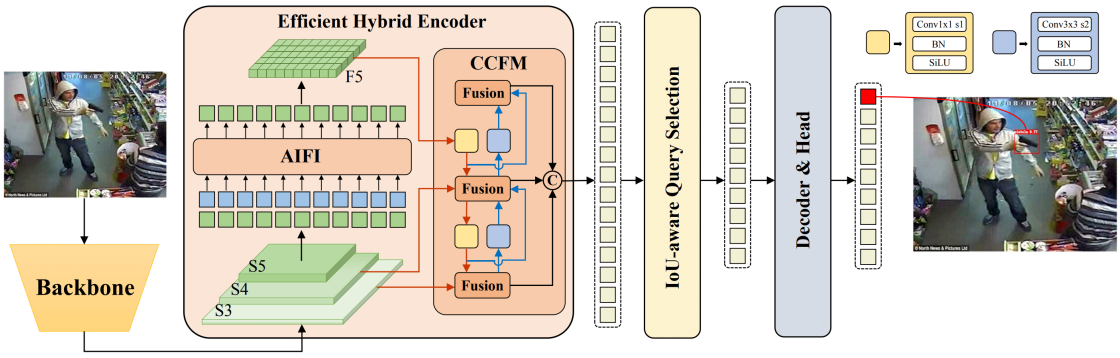


Figure 3: Baseline RT-DETR architecture for gun detection, adapted from [2]

3.2.1 Backbone

The backbone of RT-DETR is responsible for extracting multi-scale feature representations from the input image $\mathbf{I} \in R^{H \times W \times 3}$. These features provide a hierarchical structure with different levels of abstraction, which are crucial for detecting objects of varying sizes and complexities. The backbone consists of multiple convolutional layers organized into stages. The output features from the last three stages are denoted as $\{\mathbf{S}_3, \mathbf{S}_4, \mathbf{S}_5\}$, where:

- $\mathbf{S}_3 \in R^{H_3 \times W_3 \times C_3}$: Low-level features with fine spatial resolution, capturing detailed information about edges and textures. Small objects benefit from the detailed spatial resolution of $\mathbf{S}_3$.
- $\mathbf{S}_4 \in R^{H_4 \times W_4 \times C_4}$: Mid-level features with reduced spatial resolution but higher semantic abstraction, capturing information about object parts. Which balances spatial resolution and semantic richness.
- $\mathbf{S}_5 \in R^{H_5 \times W_5 \times C_5}$: High-level features with coarse spatial resolution and rich semantic information, suitable for identifying complete objects.

Each stage of the backbone is computed as:

$$\mathbf{S}_l = f_l(\mathbf{S}_{l-1}) \quad \forall l \in \{3, 4, 5\} \tag{2}$$

Where f_l represents the operations (e.g., convolution, normalization, and activation) performed in stage l.

The backbone architecture can be adapted based on the specific requirements of the application. Commonly used backbones for RT-DETR include ResNet variants, which are pre-trained on large-scale datasets like ImageNet. These backbones are chosen for their efficiency in balancing computational cost and feature quality.

3.2.2 Efficient Hybrid Encoder

The Efficient Hybrid Encoder is a key component of RT-DETR, designed to process multi-scale features $\{\mathbf{S}_3, \mathbf{S}_4, \mathbf{S}_5\}$ from the backbone. It addresses the computational bottleneck of traditional multi-scale encoders by decoupling intra-scale feature interaction and cross-scale feature fusion. This design improves efficiency while preserving accuracy.

The encoder consists of two main modules:

(1) Attention-based Intra-scale Feature Interaction (AIFI): This module operates on the high-level feature $\mathbf{S}_5$, which contains rich semantic information. By limiting the self-attention mechanism to $\mathbf{S}_5$, the AIFI captures relationships between objects and their contexts without redundant computations on lower-level features. The process can be described as:

$$\mathbf{F}_5 = AIFI(\mathbf{S}_5) \tag{3}$$

Where $\mathbf{F}_5$ is the refined representation of $\mathbf{S}_5$ after intra-scale interaction. The choice to focus on $\mathbf{S}_5$ is based on its strong semantic representation, which is more relevant for object detection tasks. Lower-level features $\mathbf{S}_3$ and $\mathbf{S}_4$ are bypassed for intra-scale attention to reduce redundancy and latency.

(2) CNN-based Cross-scale Feature Fusion (CCFF): This module fuses features from all three scales $\{\mathbf{S}_3, \mathbf{S}_4, \mathbf{F}_5\}$ to produce a unified representation $\mathbf{O}$. The fusion process is optimized for efficiency using convolutional layers and element-wise operations. Each pair of adjacent scale features $\mathbf{S}_l$ and $\mathbf{S}_{l+1}$ is combined using a fusion block, which includes:

- 1×1 convolutions to adjust the channel dimensions.
- Depthwise convolutions for efficient feature aggregation.
- Element-wise addition to combine the refined features.

Encoder Output: The final output of the encoder is a unified feature map $\mathbf{O}$, which serves as input to the subsequent uncertainty-minimal query selection module. This output combines the strengths of multi-scale features while minimizing computational overhead. The entire process of the hybrid encoder can be summarized as:

$$\begin{aligned} Q = K = V &= Flatten(\mathbf{S}_5), \\ \mathbf{F}_5 &= Reshape(AIFI(Q, K, V)), \\ \mathbf{O} &= CCFF(\{\mathbf{S}_3, \mathbf{S}_4, \mathbf{F}_5\}), \end{aligned} \tag{4}$$

Where *Flatten* and *Reshape* adjust feature dimensions for compatibility with subsequent layers.

Advantages of the Hybrid Design: The hybrid design achieves a balance between efficiency and accuracy by:

- Limiting self-attention computations to high-level features ($\mathbf{S}_5$) with the AIFI module, which captures long-range dependencies and object-level semantics.
- Using lightweight convolutional operations in the CCFF module to aggregate information from different scales, ensuring that fine-grained details from $\mathbf{S}_3$ and $\mathbf{S}_4$ are effectively combined with the semantic richness of $\mathbf{F}_5$.

3.2.3 Uncertainty-Minimal Query Selection

The Uncertainty-Minimal Query Selection module is a critical innovation in RT-DETR, designed to address the limitations of conventional query selection strategies in Transformer-based object detectors. It improves the initialization of object queries for the decoder by explicitly minimizing the uncertainty of encoder features.

Motivation: Traditional query selection methods often rely on classification confidence scores to initialize object queries. However, object detection requires accurate modeling of both classification and localization, and ignoring the uncertainty in localization can lead to suboptimal performance. The uncertainty-minimal query selection approach ensures that the selected queries have high confidence in both dimensions.

Feature Uncertainty: The uncertainty of a feature $\mathbf{X}$ is defined as the discrepancy between the predicted classification confidence $C(\mathbf{X})$ and the localization confidence $P(\mathbf{X})$. Formally:

$$U(\mathbf{X}) = \|P(\mathbf{X}) - C(\mathbf{X})\| \tag{5}$$

Where:

- $C(\mathbf{X})$ represents the probability that the feature corresponds to a foreground object (classification confidence).
- $P(\mathbf{X})$ represents the confidence in the predicted location of the object (localization confidence).

The goal is to minimize $U(\mathbf{X})$ during the query selection process to ensure high-quality queries.

Optimization Objective: To integrate uncertainty minimization into the query selection process, the following loss function is used:

$$\mathcal{L}_{query} = \mathcal{L}_{cls} + \mathcal{L}_{box} + \mathcal{L}_{uncertainty}, v \tag{6}$$

Where:

- $\mathcal{L}_{cls}$ is the classification loss, measuring the difference between predicted and ground truth class probabilities.

- $\mathcal{L}_{box}$ is the bounding box regression loss, capturing errors in object localization.
- $\mathcal{L}_{uncertainty}$ penalizes high uncertainty, defined as:

$$\mathcal{L}_{uncertainty} = \sum_{i=1}^{N} U(\mathbf{X}_i) \tag{7}$$

Where N is the number of selected queries.

3.2.4 Decoder and Prediction Head

The decoder and prediction head of RT-DETR play a critical role in transforming the encoder's unified feature map $\mathbf{O}$ into object detection predictions. This stage consists of iterative refinements performed by the Transformer decoder and the final mapping of object queries to bounding boxes and class probabilities.

Transformer Decoder: The decoder processes the encoder output $\mathbf{O}$ using a set of N object queries $\mathbf{Q} \in R^{N \times d}$, where d is the dimensionality of the queries. These object queries are initialized using the uncertainty-minimal query selection module, ensuring high-quality starting points for the decoder. The output of the last decoder layer $\mathbf{Q}^*$ contains the refined object queries, which are passed to the prediction head.

Prediction Head: The prediction head maps each refined object query $\mathbf{q}_i^* \in \mathbf{Q}^*$ to a bounding box $\mathbf{b}_i$ and class probabilities $\mathbf{c}_i$. This process involves two parallel MLPs:

- **Bounding Box Prediction:** The MLP maps each query to a bounding box $\mathbf{b}_i \in R^4$, which represents the normalized coordinates (x, y, w, h):

$$\mathbf{b}_i = MLP_{box}(\mathbf{q}_i^*) \tag{8}$$

- **Class Prediction:** The MLP predicts the class probabilities for each object query:

$$\mathbf{c}_i = Softmax(MLP_{class}(\mathbf{q}_i^*)) \tag{9}$$

Where $\mathbf{c}_i \in R^C$ represents the probabilities across C object classes.

Iterative Refinement: The decoder operates iteratively, with each layer refining the queries based on the encoder output and previous decoder layers. The number of decoder layers can be adjusted to balance accuracy and inference speed. Fewer layers reduce latency with minimal accuracy loss, making the architecture adaptable for real-time applications.

Output: The final output of the decoder and prediction head is a set of N predictions $\{(\mathbf{b}_i, \mathbf{c}_i)\}_{i=1}^{N}$, where:

- $\mathbf{b}_i$: The predicted bounding box for the i-th object.
- $\mathbf{c}_i$: The predicted class probabilities for the i-th object.

These predictions are directly compared to the ground truth during training using a bipartite matching loss, eliminating the need for post-processing steps like Non-Maximum Suppression (NMS). The entire decoding process can be summarized as:

$$\mathbf{b}_i = MLP_{box}(\mathbf{q}_i^*), \quad \mathbf{c}_i = Softmax(MLP_{class}(\mathbf{q}_i^*) \tag{10}$$

3.3 RT-DETRv2

RT-DETRv2 [33], introduces several enhancements over RT-DETRv1 to improve accuracy and flexibility, particularly in models with lightweight backbones like ResNet18. These improvements fall into three main categories: modifications in the deformable attention module, changes to the training scheme, and increased practicality for deployment. Below, we explain each improvement.

3.3.1 Deformable Attention with Scale-Specific Sampling Points

In RT-DETRv1, the deformable attention module used a fixed number of sampling points across all feature scales, which limited its ability to adapt to the intrinsic differences in features at different scales. RT-DETRv2 addresses this limitation by introducing a distinct number of sampling points for each scale, allowing the model to focus on the most relevant features for each scale. This change enhances the decoder's ability to extract multi-scale features effectively, particularly for lightweight backbones like ResNet18, which rely more heavily on precise feature extraction.

3.3.2 Discrete Sampling for Deployment Flexibility

RT-DETRv1 relied on the `grid_sample` operator, which imposed deployment constraints due to its reliance on bilinear interpolation. RT-DETRv2 introduces a novel discrete sampling operator, which replaces the `grid_sample` operator with a rounding operation. This change eliminates the need for bilinear interpolation and significantly improves deployment flexibility.

3.3.3 Dynamic Data Augmentation

RT-DETRv2 introduces a dynamic data augmentation strategy to enhance the generalizability of the model during training. Unlike RT-DETRv1, which applied static data augmentation throughout training, RT-DETRv2 dynamically adjusts the augmentation strength based on the training stage:

- **Early Training:** Strong data augmentation (e.g., `RandomPhotometricDistort`, `RandomZoomOut`, `RandomIoUCrop`, and `MultiScaleInput`) is applied to improve the model's robustness.
- **Late Training:** Augmentation is progressively reduced to allow the model to adapt to the target domain.

This strategy improves the model's ability to generalize to unseen data while maintaining high performance in the target domain.

3.3.4 Scale-Adaptive Hyperparameter Customization

RT-DETRv1 used the same optimizer hyperparameters for all backbone sizes, leading to suboptimal performance for lightweight backbones like ResNet18. RT-DETRv2 introduces scale-adaptive hyperparameter customization, which adjusts the learning rate based on the backbone's feature quality. For lightweight backbones (e.g., ResNet18), the learning rate is increased to compensate for lower feature quality. This customization ensures that each model variant achieves optimal performance relative to its architecture.

4 Experiments and Analysis

4.1 Bidirectional FPN (BiFPN)

While RT-DETR provides high performance in real-time object detection and general precision, detecting small objects in surveillance scenarios remains a significant challenge. This is particularly evident in small weapons detection at a distance or partially occluded in complex environments. The architecture of RT-DETR can be optimized to improve its precision for small object detection while maintaining its real-time efficiency.

Proposed Enhancement with BiFPN and Additional Stages: One potential improvement involves modifying the "neck" of the RT-DETR model by introducing a Bidirectional Feature Pyramid Network (BiFPN), similar to EfficientDet [4]. BiFPN is a powerful feature fusion mechanism that efficiently combines multi-scale information to enhance the detection of small objects. This approach was successfully utilized in a study aimed at detecting safety helmets under foggy conditions [5], where small objects presented similar challenges.

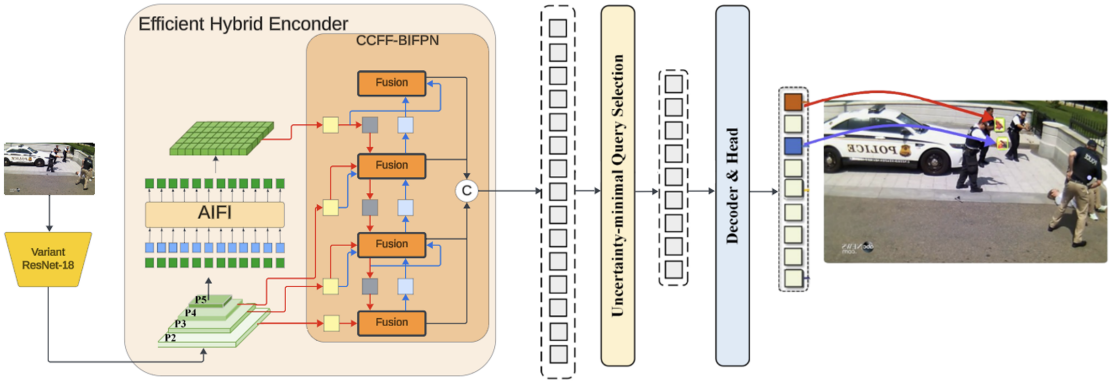


Figure 4: RT-DETR with 4 stages and BIFPN Fusion

The BiFPN module operates by aggregating features from different resolutions, ensuring that fine details from high-resolution feature maps and semantic information from low-resolution maps are effectively merged. The new hybrid encoder output would be:

$$\mathbf{O} = BiFPN(\{\mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4, \mathbf{F}_5\}) \tag{11}$$

In addition to integrating BiFPN, expanding RT-DETR's feature extraction stages from three to four, see Fig. 4.1 further enhances its ability to capture detailed information from small objects. The additional stage introduces higher-resolution feature maps, providing more granular details essential for small object detection.

4.2 Deployment

4.2.1 Optimization

For model optimization, we begin by converting the model to *Open Neural Network Exchange* (ONNX), which facilitates the optimization of models for execution on various hardware platforms, such as CPUs, GPUs, FPGAs, or mobile devices. Next, we utilize the *ONNX Simplifier* tool, which simplifies the ONNX model graph. This tool infers the entire computation graph and replaces redundant operators with their constant outputs, a process known as *constant folding*, which reduces model complexity and improves inference efficiency.

To further enhance performance, we leverage *TensorRT*, a powerful deep learning inference optimizer and runtime library developed by NVIDIA. Using the *trtexec* tool, we perform model quantization to a half-precision floating point (FP16) format, reducing memory bandwidth requirements and accelerating computation, which is particularly well-suited for edge computing scenarios.

Additionally, *TensorRT* offers the ability to simplify or modify the computation graph through operation fusions, combining multiple operations into a single, optimized operation tailored for specific hardware. This improves efficiency without sacrificing model accuracy. Mixed precision quantization is also employed, where different layers of the graph are executed with different precisions (e.g., FP16 for some layers and FP32 for others) in order to maximize performance across hardware platforms while preserving model accuracy.

This approach not only enables significant inference acceleration but also improves energy efficiency and adapts well to the resource constraints of edge devices such as mobile or embedded platforms.

4.2.2 Inference

Once the model is optimized and serialized, it is deployed on the Jetson AGX Xavier using CUDA, as shown in Fig. 5. The implementation utilizes *Cuda Streams*, where each stream processes a frame independently. The serialized model is dynamic, allowing it to accept a predefined number of input frames, although this parameter cannot be modified during runtime. The steps of the implementation are detailed below:

1. The input consists of frames obtained from one or more cameras. A *batch* of frames can be accumulated from a single camera, or each camera can be assigned a thread to process its frames. The frames are retrieved using the RTSP protocol.

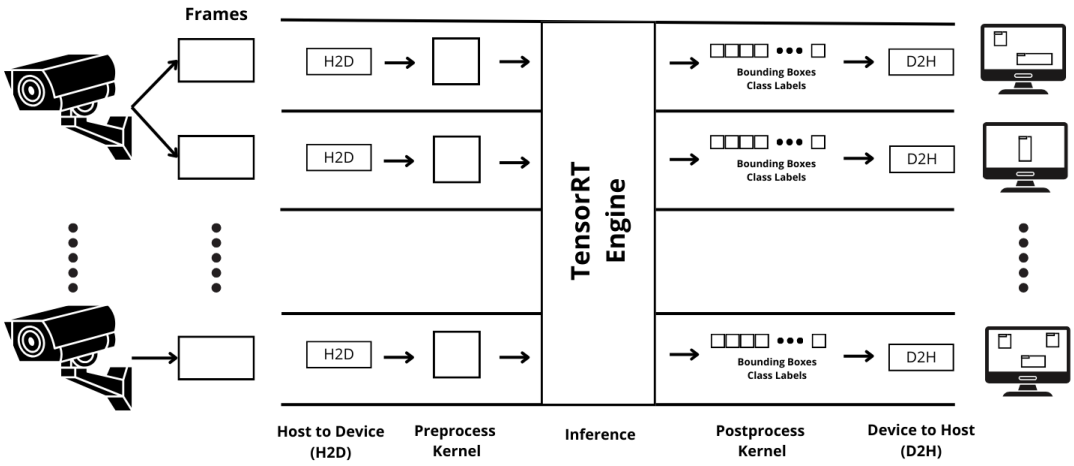


Figure 5: Implementation in CUDA [1]

2. When a frame is captured, it resides in the CPU-accessible RAM. To preprocess the input in parallel using the GPU, the image or batch of images must be transferred to the GPU memory (VRAM) via the *Host to Device (H2D)* process.

3. Preprocessing includes resizing images to 640×640 using bilinear interpolation, which estimates pixel values in the resized image based on neighboring pixels from the original image. This algorithm is parallelized to reduce preprocessing time. The resized image is then normalized using MeanStd (Mean and Standard Deviation), which adjusts the data to have a mean of zero and a standard deviation of one. Each image is processed in blocks and threads to fully leverage the parallel computation capabilities of the GPU.

4. After preprocessing, the *forward* method is called to propagate the data through the neural network, generating raw outputs (*logits*).

5. Postprocessing is streamlined in the end-to-end RT-DETR model, as it does not require the computationally expensive Non-Maximum Suppression (NMS) used by YOLO. The output consists of 300 object queries, each containing a *bbox(left, top, right, bottom)* and a confidence score for the detected class (gun). To simplify the output, the *bbox* is concatenated with the class, resulting in a 300×5 vector. Object queries are filtered based on a user-defined confidence threshold, and the bounding boxes and confidence scores of detections that pass this threshold are stored. This process is also parallelized.

6. The filtered results remain in GPU memory and are not directly accessible by the CPU. To address this, the data is transferred back from GPU memory to CPU memory via the *Device to Host (D2H)* process. This allows the output to be utilized for triggering alerts or displaying detections on-screen.

4.2.3 Edge Computing and Nvidia Jetson

The application of Edge Computing in weapon detection focuses on processing and analyzing data closer to its source, specifically at the edge of the network, near the capture devices. This paradigm offers several advantages for real-time weapon detection systems:

- **Reduced Latency:** Data processing at the edge enables faster weapon detection by eliminating the need to transmit data through external communication networks. This is critical in scenarios where every millisecond counts.

- **Enhanced Privacy and Security:** Sensitive images or data can be processed locally, minimizing the need to send them to external servers and reducing potential exposure to security vulnerabilities.

- **Bandwidth Conservation:** By reducing the amount of data transmitted over the network, edge computing minimizes bandwidth usage. This is particularly valuable in environments with limited or costly network infrastructure.

Nvidia Jetson platforms [34] are designed for real-time vision and deep learning applications at the edge, leveraging CUDA to provide a high degree of customization and control. In contrast, Intel Movidius Neural Compute Stick (NCS) [35] is a low-power vision processor designed for AI model inference, utilizing the OpenVINO toolkit for high-level abstraction and ease of use. For the purposes of this study, CUDA is preferred, as it enables more efficient utilization of hardware resources for the proposed solution, the device used for testing is the following.

Nvidia Jetson AGX Xavier specifications:

- AI Performance: 32 TOPS
- GPU: 512-core NVIDIA Volta architecture GPU with 64 Tensor Cores
- CPU: 8-core NVIDIA Carmel Arm®v8.2 64-bit CPU 8MB L2 + 4MB L3
- Memoria: 32GB 256-bit LPDDR4x 136.5GB/s
- Power: 10W, 15W y 30W

An additional advantage of the Nvidia Jetson AGX Xavier is its unified memory architecture, where the 32GB of LPDDR4x RAM can also be utilized by the GPU. This shared memory design allows for the deployment of larger models or the processing of higher batch sizes, providing a significant advantage in scenarios requiring real-time inference.

4.3 Setup

4.3.1 Datasets

Datasets used in previous work [1]:

- **JoinDatabase [29]:** This dataset consists of 7,000 images of firearms captured by video surveillance cameras. It was created by consolidating multiple datasets, with the authors removing duplicates, low-resolution images, and those depicting unrealistic contexts. This dataset will be employed to enhance the model's performance in video surveillance scenarios.

The [32] dataset will not be considered because it does not include real-world scenarios. In our previous work [1], the RTDETR-R18A model trained with this dataset performed poorly on the [dextre] dataset, which includes such scenarios, as also noted in [6]. Similarly, the [24] dataset will not be included because it was used as a preliminary step to evaluate the models' precision and speed. Additionally, many of its images are already present in the [29, 6] datasets.

The following datasets used in the benchmark [6] were extracted and transformed from public datasets, we will use the modified versions in [6] and not the original datasets for comparison.

- **Monash Gun Dataset (MGD):** Each annotated CCTV image in this dataset contains the presence of a handgun in various outdoor and indoor conditions. The dataset includes images of enacted crime scenes. The images are of size 512×512. Additionally, 2,857 images were extracted from 250 recorded CCTV videos in diverse indoor and outdoor settings.
- **US Real-Time Gun Detection Dataset (USRT):** This dataset contains images of enacted crime scenes, comprising 3,294 images. Both MGD and USRT are mock datasets, meaning the creators acted out the attack scenes. The images in this dataset are of size 1920×1080. Although the dataset also includes annotations for knives and shotguns, these were ignored and considered as background.
- **UCF Crime Scene Dataset (UCF):** This dataset originates from a large-scale collection of 128 videos, primarily intended for anomaly detection rather than gun detection. The subset used consists of robbery and shooting videos (with a resolution of 320×240), recorded by CCTV cameras during real-world crime scenes. A total of 57 robbery and 17 shooting videos were selected, from which handgun images were extracted at a rate of 2 frames per second, resulting in 1,616 handgun frames.

Table 1: Hyperparameters for RT-DETR-R18v1 and RT-DETR-R18v2.

Component	**Hyperparameter**	**RT-DETR-R18v1**	**RT-DETR-R18v2**
Model	Backbone	PResNet18	PResNet18
	Number of stages	3	3, (4 BiFPN)
	Transformer layers	3	3
Optimizer	Type	AdamW	AdamW
	Learning rate (backbone)	1×10^{-5}	1×10^{-5}
	Learning rate (encoder/decoder)	1×10^{-4}	1×10^{-4}
	Weight decay (encoder/decoder)	0	0
	Betas	[0.9, 0.999]	[0.9, 0.999]
	EMA decay	0.9999	0.9999
Scheduler	Learning rate scheduler	MultiStepLR	MultiStepLR
	Milestones	[1000]	[1000]
	Gamma	0.1	0.1
	Warmup duration	-	2000
Dataloader	Total batch size (train)	4	16
	Total batch size (validation)	8	32
	Data augm. policy stop epoch	-	117
	Number of workers	4	4
Loss	Loss functions	vfl, boxes	vfl, boxes
	Loss weights	vfl:1, bbox:5, giou:2	vfl:1, bbox:5, giou:2
	Matcher	HungarianMatcher	HungarianMatcher
	Matcher weights	class:2, bbox:5, giou:2	class:2, bbox:5, giou:2
	Focal loss parameters	α: 0.75, γ: 2.0	α: 0.75, γ: 2.0
Transformer	Feature map strides	[8, 16, 32]	[8, 16, 32]
	Number of queries	300	300
	Label noise ratio	0.5	0.5
	Box noise scale	1.0	1.0
Training	Total epochs	72	120

4.3.2 Experimental Setup

RTDETRv2 incorporates new hyperparameters, the training configurations for both models are compared in Table 1, all models will be trained on a 24 GB Nvidia RTX 4090 GPU. Since this work is designed for implementation on embedded systems, a lightweight model, such as RT-DETR with a ResNet18 backbone, has been selected. Furthermore, as demonstrated in [**?**], this model achieves a superior AP/FPS ratio, making it well-suited for the target application.

- **Data Augmentation:** Both models apply similar augmentations such as `RandomPhotometricDistort`, `RandomZoomOut`, and `RandomIoUCrop`, but RT-DETR-R18v2 dynamically adjusts augmentation strategies after epoch 117.
- **Batch Collation:** RT-DETR-R18v2 uses multi-scale input resizing until epoch 71, while RT-DETR-R18v1 does not specify a multi-scale policy.
- **Evaluation Resolution:** Both models use a fixed evaluation resolution of 640×640 pixels.
- **EMA:** Exponential Moving Average (EMA) is enabled in both configurations to stabilize training.

4.4 Results

First, to evaluate the performance of BiFPN when applied to RT-DETR, we will assess AP_s, the metric commonly used for small object detection. The Tab. 2 shows the combined result of previous work in the same dataset. Results highlighted in black show a model's best performance on each metric.

The models used in the benchmark [6] are not designed for real-time processing, and the work presented in [29] uses an outdated version of YOLO. Therefore, we will train a more recent version, YOLOv10, for 120 epochs using the same datasets. In the results presented in [6], AP_{50} was used, applied to each class (handgun, person), for the Tab. 3, we will use the average of both in each dataset.

The table 4 provides a comparison of the number of parameters and computational cost (in GFLOPs) for various models. This information highlights the trade-off between model complexity and computational

Table 2: Comparison of small object detection with RT-DETR versions

Model	**Weights**	**AP**$_{50:95}$	**AP**$_{50}$	**AP**$_{S}$
Test Dataset [29] *Mainly real situations*				
RT-DETR R18A [1]	COCO+Obj365+[32]	28.5	58.6	21.3
RT-DETR R18B [1]	COCO+Obj365	82.2	**98.8**	79.5
RT-DETR V2 R18	COCO	**94.8**	98.6	97.0
RT-DETR V2 R18 BiFPN	COCO	94.6	98.5	**97.7**

Table 3: Comparison using Average Precision at IoU=0.5, the models were trained and tested on their respective dataset

Model	**Weights**	**MGD**	**USRT**	**UCF**
FasterRCNN+FPN ResNet50 [6]	COCO	90.80	61.85	65.30
DeformableDETR ResNet50 [6]	COCO	93.05	58.55	67.45
DetectoRS ResNet50 [6]	COCO	91.25	65.25	71.95
FasterRCNN+FPN Swin-T [6]	COCO	92.60	65.40	73.40
FasterRCNN+FPN ConvNext-T [6]	COCO	91.85	**65.60**	73.10
RT-DETR R18	COCO	95.11	62.52	67.83
RT-DETR R18	COCO + Obj365	94.89	63.06	**78.93**
RT-DETR V2 R18	COCO	93.68	65.00	75.62
RT-DETR V2 R18 BiFPN	COCO	**95.92**	64.39	71.70
YOLOV10 S	COCO	87.90	55.30	60.80
YOLOV10 M	COCO	89.90	51.70	62.10

requirements, which is critical for resource-constrained applications such as real-time systems or embedded devices.

Table 4: Comparison of the number of parameters in millions and GFLOPs for each model, The first 2 are non-real-time models

Model	**Params (M)**	***GFLOP*s**
Faster R-CNN-FPN R50 [26]	41.7	134.4
Faster R-CNN [24]	138	-
YOLOV5-S [29]	7.3	17
YOLOV5-M [29]	21.4	51.3
YOLOV10-S	7.2	21.6
YOLOV10-M	15.4	59.1
RT-DETR R18	20.0	60.0
RT-DETR R18 V2	20.0	60.0
RT-DETR R18 V2 BiFPN	22.2	65.4
RT-DETR R34	31.0	92.0
RT-DETR R50	42.0	136.0

In section 4.2.2, a CUDA implementation for inference was presented, which allows both batch processing and real-time processing (batch size = 1). In batch processing, the CUDA H2D and D2H operations are performed only once per batch, which reduces the overhead of frequent data transfers. While latency increases due to processing multiple frames simultaneously, performance improves significantly when measured over a time slice. For example, processing 60 input frames with a batch size of 1 requires 60 CUDA I/O operations, while with a batch size of 10, only 6 operations are needed, Tab. 5 shows this behavior. Consequently, batch processing achieves higher efficiency and faster processing per second compared to real-time processing, however, the latency per inference increases.

The work by [6] also included images with challenging attributes, such as occlusion, blur, and gun-like

Table 5: Comparison of inference speed and throughput for each batch size, with results including pre-processing and post-processing time

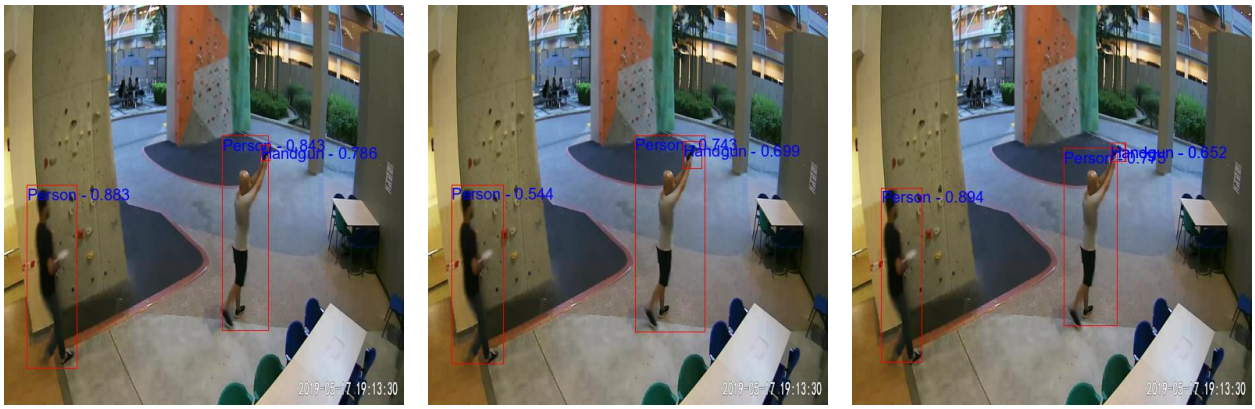
Model	BS=1		BS=4		BS=8	
	Inf. (ms)	Throughput img/s	Inf. (ms)	Throughput img/s	Inf. (ms)	Throughput img/s
RT-DETR R18	26,32	37.99	65.11	61.43	123.96	64.54
RT-DETR V2 R18	26,16	38.22	64.95	61.58	123.75	64.64
RT-DETR R18 BiFPN	28.81	34.21	71.01	56.33	135.19	59.27

objects that can easily cause confusion. The images in Fig. 6 were selected from the blur subset and tested with RT-DETR BiFPN models trained on MGD, USRT, and UCF, respectively. Among these, only the image in Fig. 6b contains a detection error: the person was not detected, but the weapon was successfully identified.

(a) (b) (c)

Figure 6: Inference results for a blurred image, confidence threshold = 0.50

In the images shown in 7, no errors are found in any case. However, the confidence in detecting people is considerably lower and is close to the detection threshold in the USRT dataset compared to the other datasets.



(a) (b) (c)

Figure 7: Inference results for an image of similar and small objects, confidence threshold = 0.50

4.5 Discussion

The integration of BiFPN into RT-DETR, as presented in Table 2, resulted in a marginal enhancement in small object detection. However, considering that the precision already exceeds 97%, achieving further

improvements becomes increasingly challenging. Notably, BiFPN did not surpass the standard RT-DETR v2 in terms of Average Precision (AP). When comparing benchmark results, RT-DETR-based models outperformed others in two datasets. For the standard model, pre-trained weights from COCO and Objects365 are available, whereas these are absent in RT-DETR v2. Therefore, the observed improvement may be attributed to the dataset used for training rather than the model itself.

In the benchmark (Table 3), RT-DETR-based models excelled on the MGD dataset, which comprises images of recreated scenes, and on UCF, derived from real scenes. In MGD, minimal differences were observed among models, except for the YOLO variants, which remained below 90%. Conversely, in UCF, the standard RT-DETR with COCO + Objects365 weights demonstrated a notable advantage. Given that these are real images, the pre-trained weights enhance performance. This is followed by RT-DETR v2; however, the BiFPN variant underperforms compared to the original RT-DETR v2. This further underscores the significance of pre-trained weights, as adding an extra stage initialized with random weights degrades performance in real-world scenarios with more varied contexts than recreated situations. For the USRT dataset, which, like MGD, consists of recreated images but at a much higher resolution (1920×1080), three models proposed in [6] outperformed those based on RT-DETR. Analyzing the input size, these larger models accommodate variable input sizes ranging from 800 to 1333, compared to RT-DETR's 640. This issue arises from the information loss associated with image resizing. However, unlike RT-DETR, these models do not operate in real-time.

Regarding inference speed, YOLOv10-based models are lighter in terms of parameters and GFLOPs (see Table 4), as this version omits the NMS algorithm used in its predecessors, making them suitable candidates for embedded system implementation. It is noteworthy that, despite YOLOv10 M having considerably fewer parameters, it is slower than RT-DETR R18. As both are end-to-end models, this discrepancy cannot be attributed to the NMS algorithm but rather to the architecture. RT-DETR is more parallelizable, whereas YOLOv10 may encounter bottlenecks due to its pipeline, which includes numerous residual connections and a complex architecture. Thus, a model's efficiency depends not only on the number of parameters but also on its architecture and the management of internal operations during processing.

Even in its most basic version, RT-DETR outperforms the two YOLO versions implemented for comparison in terms of accuracy. Depending on the application or desired result speed, batch processing can be utilized to increase the number of images processed per second and optimize NVIDIA Jetson resources, particularly VRAM. As shown in Table 5, with a batch size of 4, performance is multiplied by 1.6; however, this improvement does not persist with a batch size of 8. The Nvidia Jetson capitalizes on this; the test equipment contains 32GB of RAM/VRAM, an uncommon feature for a low-power device. This design supports multiple models and batch processing, which tends to consume more memory depending on the batch size used.

5 Conclusion

In this work, we explored the advancements introduced in RT-DETRv2 for real-time firearm detection, including the integration of BiFPN, and evaluated its performance on more realistic datasets. Additionally, updated model configurations, such as dynamic augmentation and multi-scale resizing, further enhance detection accuracy, particularly for small-object datasets like MGD, USRT, and UCF. The model's performance on the MGD and USRT datasets demonstrates its strong generalization across diverse environments, including both indoor and outdoor crime scenes. RT-DETRv2 maintains competitive performance, outperforming even the latest YOLO versions. This study also highlights its flexible deployment through dynamic batching on edge devices such as the Nvidia Jetson, optimizing both performance and resource efficiency. However, while the datasets used provide a wide range of scenarios, they do not fully capture the context of Latin American countries, as they were not specifically designed for them. Moreover, many of the images, despite resembling real-world conditions, do not originate from actual crime situations, which may introduce errors when applying the model in real-world scenarios. Finally, the input image resolution (640×640) is uncommon in surveillance cameras, necessitating transformations that lead to information loss and potentially degrading the model's performance.

6 Future Work

The future work of the proposed weapon and violence detection system will involve implementing a two-stage approach. In the first stage, both weapons and people will be detected, leveraging the fact that people, due to their larger size, are easier to identify. In the second stage, the images of the detected people will be cropped, and a classification model will be applied to verify the presence of weapons and violence. This

approach aims to optimize the efficiency of the surveillance system, providing more accurate and reliable detections.

For example, the USRT dataset contains images with a resolution of 1920x1080, while the implemented model accepts images of 640x640. Resizing the image results in the loss of crucial information, especially when weapons are small objects, making their detection more difficult. Additionally, datasets [6] include the class 'person,' which could facilitate initial identification and enhance weapon and violence detection accuracy.

References

[1] L. A. Bustamante and J. C. Gutiérrez, "Real-time handgun detection using transformers on nvidia jetson agx xavier," in *2024 L Latin American Computer Conference (CLEI)*, 2024, pp. 1–6. [Online]. Available: https://doi.org/10.1109/CLEI64178.2024.10700426

[2] Y. Zhao, W. Lv, S. Xu, J. Wei, G. Wang, Q. Dang, Y. Liu, and J. Chen, "Detrs beat yolos on real-time object detection," 2024, Conference paper, p. 16965 – 16974, cited by: 381; All Open Access, Green Open Access. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85216283501&doi=10.1109%2fCVPR52733.2024.01605&partnerID=40&md5=a8a554beb4d310aecbef9f5866840942

[3] "Estadísticas de seguridad ciudadana del semestre móvil: mayo – octubre 2024," Instituto Nacional de Estadística e Informática INEI, , 2024. [Online]. Available: https://www.gob.pe/institucion/inei/informes-publicaciones/6244024-informe-tecnico-estadisticas-de-seguridad-ciudadana-del-semestre-movil-mayo-octubre-2024

[4] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 778–10 787. [Online]. Available: https://doi.org/10.1109/CVPR42600.2020.01079

[5] Z. Liu, C. Sun, and X. Wang, "Dst-detr: Image dehazing rt-detr for safety helmet detection in foggy weather," *Sensors*, vol. 24, no. 14, 2024, cited by: 2; All Open Access, Gold Open Access, Green Open Access. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85199779170&doi=10.3390%2fs24144628&partnerID=40&md5=a0d12619271734bc51069f3a182ea1b8

[6] S. Yellapragada, Z. Li, K. B. Doshi, P. Mhasakar, H. Fan, J. Wei, E. Blasch, and H. Ling, "Cctv-gun: Benchmarking handgun detection in cctv images," *ArXiv*, vol. abs/2303.10703, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257632553

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2014, Conference paper, p. 580 – 587, cited by: 26176; All Open Access, Green Open Access. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84911400494&doi=10.1109%2fCVPR.2014.81&partnerID=40&md5=91442ce64988c082a419975369a2a615

[8] R. Girshick, "Fast r-cnn," vol. 2015 International Conference on Computer Vision, ICCV 2015, 2015, Conference paper, p. 1440 – 1448, cited by: 23006. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84964588182&doi=10.1109%2fICCV.2015.169&partnerID=40&md5=a20bd3b46a2dd9c7893e12e9c19e5e4c

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, p. 1137 – 1149, 2017, cited by: 25804; All Open Access, Green Open Access. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85019258369&doi=10.1109%2fTPAMI.2016.2577031&partnerID=40&md5=d7b4c992730c8e0c7b63f78b01ad4bfb

[10] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 936–944. [Online]. Available: https://doi.org/10.1109/CVPR.2017.1060

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," vol. 2016-December, 2016, Conference paper, p. 779 – 788, cited by: 37402; All Open Access, Green Open Access. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84986308404&doi=10.1109%2fCVPR.2016.91&partnerID=40&md5=79a23b9cc271b6f314eea447fb88cc7d

[12] J. Redmon and A. Farhadi, "YOLOV3: An incremental improvement," *arXiv (Cornell University)*, 4 2018. [Online]. Available: https://arxiv.org/abs/1804.02767

[13] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOV4: Optimal speed and accuracy of object detection," 4 2020. [Online]. Available: https://arxiv.org/abs/2004.10934

[14] G. Jocher, "YOLOv5 by Ultralytics," May 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[15] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7464–7475, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:250311206

[16] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO," Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[17] C.-Y. Wang, I.-H. Yeh, and H. Liao, "Yolov9: Learning what you want to learn using programmable gradient information," *ArXiv*, vol. abs/2402.13616, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:267770251

[18] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding, "Yolov10: Real-time end-to-end object detection," *ArXiv*, vol. abs/2405.14458, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:269983404

[19] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021, Conference paper, cited by: 9729. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85150208907&partnerID=40&md5=e9616b2256a7db273c2ed80b6f498721

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Kaiser, and I. Polosukhin, "Attention is all you need," vol. 2017-December, 2017, Conference paper, p. 5999 – 6009, cited by: 79385. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85043317328&partnerID=40&md5=3e5a5c2b862c8979ffea845bb707b3c3

[21] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 9992–10 002. [Online]. Available: https://doi.org/10.1109/ICCV48922.2021.00986

[22] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I.* Berlin, Heidelberg: Springer-Verlag, 2020, p. 213–229. [Online]. Available: https://doi.org/10.1007/978-3-030-58452-8_13

[23] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," 2021, Conference paper, cited by: 1138. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85144432695&partnerID=40&md5=4e20aa4c5921aca153676add3c3dd8cf

[24] R. Olmos, S. Tabik, and F. Herrera, "Automatic handgun detection alarm in videos using deep learning," *Neurocomputing*, vol. 275, p. 66 – 72, 2018, cited by: 185; All Open Access, Green Open Access. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85019947234&doi=10.1016%2fj.neucom.2017.05.012&partnerID=40&md5=e74598f026f6d7485aee108974cd2649

[25] A. Castillo, S. Tabik, F. Pérez, R. Olmos, and F. Herrera, "Brightness guided preprocessing for automatic cold steel weapon detection in surveillance videos with deep learning," *Neurocomputing*, vol. 330, pp. 151–161, 2 2019. [Online]. Available: https://doi.org/10.1016/J.NEUCOM.2018.10.076

[26] J. L. S. González, C. Zaccaro, J. A. Álvarez García, L. M. S. Morillo, and F. S. Caparrini, "Real-time gun detection in cctv: An open problem," *Neural Networks*, vol. 132, pp. 297–308, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608020303361

[27] A. Warsi, M. Abdullah, M. N. Husen, M. Yahya, S. Khan, and N. Jawaid, "Gun detection system using yolov3," in *2019 IEEE International Conference on Smart Instrumentation, Measurement and Application (ICSIMA)*, 2019, pp. 1–4. [Online]. Available: https://doi.org/10.1109/ICSIMA47653.2019.9057329

[28] T. S. S. Hashmi, N. U. Haq, M. M. Fraz, and M. Shahzad, "Application of deep learning for weapons detection in surveillance videos," in *2021 International Conference on Digital Futures and Transformative Technologies (ICoDT2)*, 2021, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ICoDT252288.2021.9441523

[29] M. Dextre, O. Rosas, J. Lazo, and J. C. Gutiérrez, "Gun detection in real-time, using yolov5 on jetson agx xavier," *Proceedings - 2021 47th Latin American Computing Conference, CLEI 2021*. [Online]. Available: https://doi.org/10.1109/CLEI53233.2021.9640100

[30] P. Yadav, N. Gupta, and P. K. Sharma, "Robust weapon detection in dark environments using yolov7-darkvisionimage 1," *Digital Signal Processing*, vol. 145, p. 104342, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1051200423004372

[31] M. Pullakandam, K. Loya, P. Salota, R. M. R. Yanamala, and P. K. Javvaji, "Weapon object detection using quantized yolov8," in *2023 5th International Conference on Energy, Power and Environment: Towards Flexible Green Energy Technologies (ICEPE)*, 2023, pp. 1–5. [Online]. Available: https://doi.org/10.1109/ICEPE57949.2023.10201506

[32] D. Qi, W. Tan, Z. Liu, Q. Yao, and J. Liu, "A dataset and system for real-time gun detection in surveillance video using deep learning," 2021, Conference paper, p. 667 – 672, cited by: 14; All Open Access, Green Open Access. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85124274513&doi=10.1109%2fSMC52423.2021.9659207&partnerID=40&md5=f149df98d6f5c4946dd81f5feca66d43

[33] W. Lv, Y. Zhao, Q. Chang, K. Huang, G. Wang, and Y. Liu, "Rt-detrv2: Improved baseline with bag-of-freebies for real-time detection transformer," 2024. [Online]. Available: https://arxiv.org/abs/2407.17140

[34] Nvidia jetson xavier. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-xavier-series/

[35] Intel movidius neural compute stick. [Online]. Available: https://www.intel.com/content/www/us/en/products/sku/125743/intel-movidius-neural-compute-stick/specifications.html