information

MDPI

Article

Object Detection Post Processing Accelerator Based on Co-Design of Hardware and Software

Dengtian Yang [1,2], Lan Chen [1,*], Xiaoran Hao [1] and Yiheng Zhang [1]

[1] Institute of Microelectronics of the Chinese Academy of Sciences, Beijing 100029, China; yangdengtian@ime.ac.cn (D.Y.); haoxiaoran@ime.ac.cn (X.H.); zhangyiheng@ime.ac.cn (Y.Z.)
[2] University of Chinese Academy of Sciences, Beijing 100049, China
* Correspondence: chenlan@ime.ac.cn; Tel.: +86-010-8299-5745

Abstract: Deep learning significantly advances object detection. Post processes, a critical component of this process, select valid bounding boxes to represent the true targets during inference and assign boxes and labels to these objects during training to optimize the loss function. However, post processes constitute a substantial portion of the total processing time for a single image. This inefficiency primarily arises from the extensive Intersection over Union (IoU) calculations required between numerous redundant bounding boxes in post processing algorithms. To reduce these redundant IoU calculations, we introduce a classification prioritization strategy during both training and inference post processes. Additionally, post processes involve sorting operations that contribute to their inefficiency. To minimize unnecessary comparisons in Top-K sorting, we have improved the bitonic sorter by developing a hybrid bitonic algorithm. These improvements have effectively accelerated the post processing. Given the similarities between the training and inference post processes, we unify four typical post processing algorithms and design a hardware accelerator based on this framework. Our accelerator achieves at least 7.55 times the speed in inference post processing compared to that of recent accelerators. When compared to the RTX 2080 Ti system, our proposed accelerator offers at least 21.93 times the speed for the training post process and 19.89 times for the inference post process, thereby significantly enhancing the efficiency of loss function minimization.

Keywords: deep learning; object detection; post process; accelerator

Academic Editor: Ognjen Arandjelović

Received: 3 December 2024
Revised: 11 January 2025
Accepted: 13 January 2025
Published: 17 January 2025

Citation: Yang, D.; Chen, L.; Hao, X.; Zhang, Y. Object Detection Post Processing Accelerator Based on Co-Design of Hardware and Software. *Information* **2025**, *16*, 63. https://doi.org/10.3390/info16010063

1. Introduction

Object detection, a fundamental task in deep learning, has found widespread application in fields such as autonomous driving [1,2], agricultural monitoring [3,4], and geological exploration [5,6]. The complete training process for object detection includes feature extraction through the backbone [7,8], feature fusion via the neck [9,10], detection heads [11–13], post processing [14], loss calculation [12,13,15,16], and back propagation. During inference, the steps of loss calculation and back propagation are omitted. The backbone, neck, and head are typically implemented using Convolutional Neural Networks (CNNs). Post processing mainly involves selecting valid bounding boxes. In the training phase, post processing is responsible for assigning labels to the bounding boxes produced by the detection head. This step involves matching the detected bounding boxes with ground truth targets and is commonly referred to as label assignment (LA). During the inference phase, post processing involves choosing bounding boxes that represent the true targets using Non-Maximum Suppression (NMS).

Figure 1a shows how in applying NMS to inference, box2 is selected as the optimal bounding box based on the IoU and confidence. Figure 1b illustrates Faster-RCNN's training post process, assigning the dog label and box2 to the true target. Both NMS and label assignment aim to match the bounding boxes with the true targets to enhance the model's accuracy. However, while previous CNN hardware accelerators [17–20] have focused on the feature process, post processing acceleration has been explored less due to algorithmic differences. For instance, Faster-RCNN [21], YOLOv8s, and GFL [22] use different assigners, such as a Max IoU Assigner, Task-Aligned Assigner [23], and Dynamic Soft Label Assigner, in training, whereas NMS is consistently used in inference. Thus, creating a unified post processing workflow is challenging.

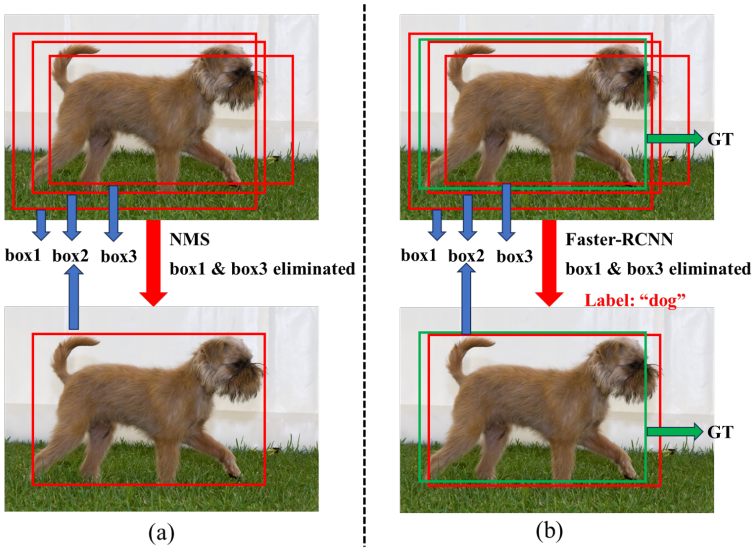

Figure 1. The (**a**) inference and (**b**) training post processes. The box represents the bounding box, while GT denotes the ground truth, which corresponds to the box that accurately delineates the true target. Red boxes denote the bounding boxes of the detected objects (indicated by blue arrows pointing to specific numbered boxes), while green boxes denote the GT (indicated by green arrows pointing to the GT). Red arrows indicate post processing outcomes.

Previous post processing accelerators have primarily focused on NMS due to its significance in edge inference tasks. However, as edge devices require more advanced detection capabilities, they perform local fine-tuning using real-time camera data [24,25], eliminating the need for cloud-based training and the complexities of data transmission. This highlights the need for hardware acceleration in the training post process. Despite this, research on training post processing acceleration is limited due to challenges in its standardization and a steep learning curve. Our observations show that both the training and inference post processes consume a significant portion of the total processing time for a single image. As Figure 2 illustrates, using an RTX 2080 Ti to measure the processing times on the COCO dataset, the average time dedicated to post processing during training and inference accounted for 20.5% and 22.1% of the total processing time, respectively.

Post processing involves fewer parameters than feature processing but lacks a unified statistical approach. We propose two methods. Unlike feature processing, which focuses on weights and biases, post processing emphasizes bounding boxes, thresholds, and empirical parameters. The number of bounding boxes, which depends on the input image size, includes confidence scores, categories, and coordinates, while the thresholds and empirical parameters are fewer. The first method includes all of the bounding box data, thresholds, and empirical parameters. The second method only focuses on the thresholds and empirical parameters, aligning with feature processing. As shown in Figure 3, if the feature map size

is considered in post processing, the feature process's parameters are 2.28 to 4.92 times greater than those in post processing. If not, they can be up to 787,000 times greater for training and 21.4 million times greater for inference. Accelerating post processing is crucial, with the main bottleneck being the repeated bounding box filtering. Reducing ineffective filtering is the key to improving its efficiency.

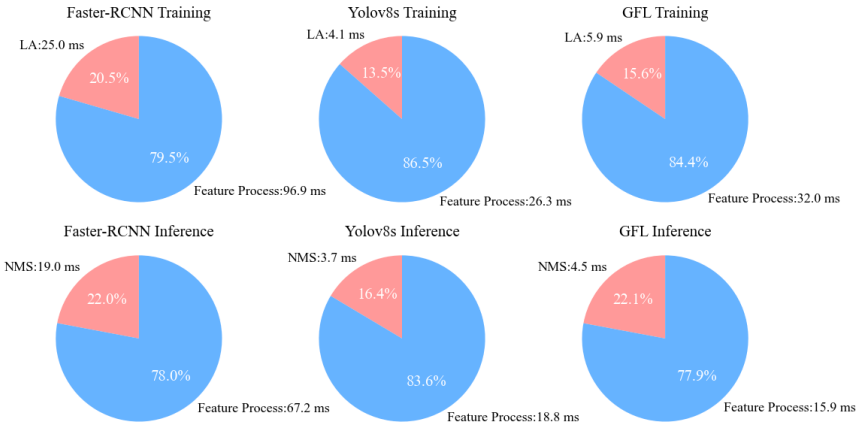

Figure 2. The time proportions for training and inference post processes in three detectors. LA is the training post process, NMS is the inference post process, and Feature Process includes the backbone, neck, and head processing. For Faster-RCNN training, feature processing averages 96.9 ms and LA 25.0 ms per image, accounting for 79.5% and 20.5% of the total time, respectively.

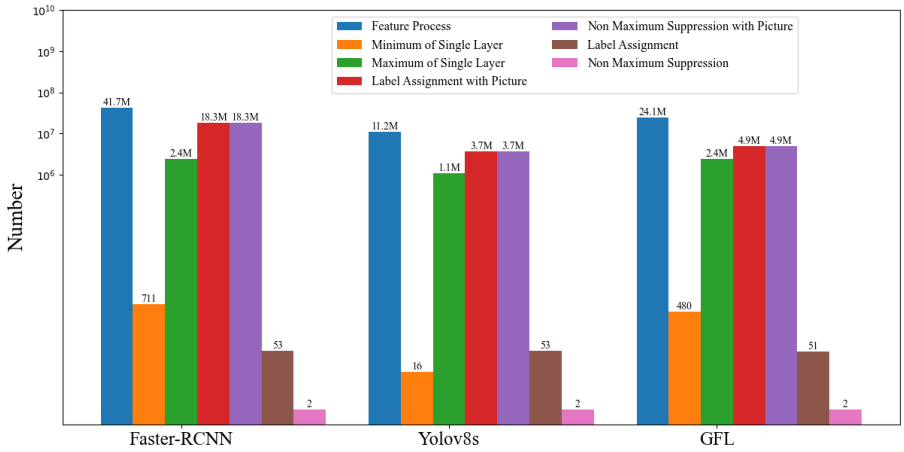

Figure 3. Parameter count statistics: Feature Process includes parameters from the model's backbone, neck, and head. Minimum of Single Layer is the parameter count of the convolutional layer with the fewest parameters, and Maximum of Single Layer is that for the layer with the most. Label Assignment with Picture and Non-Maximum Suppression with Picture count the parameters when feature maps are considered in the post processes, while Label Assignment and Non-Maximum Suppression do not include the feature map parameters.

Reducing redundancy in post processing involves minimizing unnecessary IoU calculations and comparisons.

- Minimizing redundant IoU calculations: The traditional methods often include redundant IoU calculations. We introduce the priority of confidence threshold checking combined with classification (CTC-C) into NMS, effectively reducing the computation of ineffective IoUs. Given the operational similarities between the inference post process and the training post process, we also implement the priority of CTC-C in the training post process of Faster-RCNN. For YOLOv8s and GFL, we incorporate

the priority of center point position checking combined with classification (CPC-C) to further reduce redundant IoU calculations.

- Reducing redundant comparisons: Traditional Top-K selection methods involve full sorting, leading to unnecessary comparisons. We introduce a hybrid bitonic sorter to avoid this, efficiently sorting the bounding boxes by the IoUs and identifying the Top-K elements, optimizing the selection of the best bounding boxes during training.

The core research content of this paper is post processing software–hardware acceleration. In terms of the software, we use the priorities of CTC-C and CPC-C to implement the reduction in redundant IoU calculations, employ a hybrid bitonic sorter to reduce redundant comparisons, and unify four post processing algorithms. In terms of the hardware, we design the hardware accelerator under a unified algorithmic framework.

The main contributions of this paper are as follows:

- Post processing software–hardware acceleration: We highlight the importance of hardware acceleration in both the training and inference post processes, proposing a co-design strategy to integrate the software and hardware for improved efficiency.
- Unified algorithm framework and hardware design: We explore unifying the training and inference post processing algorithms, presenting a hardware design framework that achieves significant speedups. Compared to recent accelerators, our design offers at least 7.55 times faster NMS acceleration.
- Priorities of CTC-C and CPC-C: We address the redundant IoU calculations by introducing the priorities of CTC-C and CPC-C into the training and inference post processes, reducing the bounding boxes to 12.9% and 11.9% of their original counts, respectively, and achieving a speedup of 1.10 to 1.19 times.
- Hybrid bitonic sorter: We introduce a hybrid bitonic sorter that enables efficient parallel comparisons and Top-K selection, reducing redundant comparisons. This design achieves a speedup of 1.05 to 1.25 times compared to the speed of recent sorters.

2. The Related Works

2.1. Object Detection and Post Process Algorithm

Traditional object detectors, with their complex handcrafted features and lower accuracy, are being replaced by deep learning models [26]. These models use large network architectures with parameters of up to hundreds of megabytes [27,28], allowing for extensive feature learning without manual intervention. They offer end-to-end solutions that integrate image processing, feature learning, object classification, and localization into a unified framework. Object detectors are categorized into one-stage and two-stage detectors. One-stage detectors, such as YOLO and SSD, directly regress boxes in images, making them faster but traditionally less accurate [29,30]. In contrast, two-stage detectors like Faster-RCNN first generate candidate regions and then classify and refine them, achieving higher accuracy at the cost of speed [21,31]. Recent advancements, including anchor-free methods like CornerNet and CenterNet [32–34]; improved feature fusion in neck architectures [19,27]; and adaptive post processing algorithms, have enhanced the accuracy of one-stage detectors. These improvements have made one-stage detectors increasingly popular in applications such as intelligent transportation [35] and fire detection [36], where speed and efficiency are crucial.

Object detection post processing algorithms are categorized into training and inference processes. The training process aims to classify samples as positive or negative, compute the losses using ground truth boxes, and guide learning through supervisory signals. Label assignment is crucial, divided into static and dynamic methods based on the threshold variability. Static methods use fixed thresholds, such as the distance and IoU, exemplified by FCOS [37], Faster-RCNN's Max IoU Assigner [21], and RFLA [38]. Dynamic methods

adjust the thresholds dynamically, including ATSS [14], PAA [39], OTA[40], the Dynamic Soft Label Assigner [22], SimOTA [41], and the Task-Aligned Assigner [23]. While ATSS uses dynamic IoU thresholds, others combine classification predictions and IoU scores to align the training and inference processes, improving the accuracy. The shift from the Max IoU Assigner to the Task-Aligned Assigner reduces the reliance on the empirical parameters, enhancing the model's generalizability. Despite this, post processing remains complex. Approaches like DETR [15,42,43] simplify the training post process with one-to-one matching, but other Transformer-based detectors [44,45] using Swin Transformer [7] maintain one-to-many processes for higher accuracy and faster convergence. Although DETR claims to eliminate complex post processing, NMS is still required during inference.

In contrast to the training post process, which has seen significant evolution, the inference post process has developed more incrementally, with most of the advancements focusing on the IoU calculation paradigm [46–49]. Standard NMS remains the predominant technique, ranking the candidate bounding boxes by confidence scores and iteratively removing those that overlap significantly with higher-IoU candidates [50]. However, it uses fixed empirical thresholds and does not adapt to each input image's unique characteristics, potentially limiting the detector performance. To address this, some research has redefined the NMS threshold selection into a convex optimization problem, applying swarm intelligence optimization algorithms to find the optimal solutions [51]. Adaptive NMS strategies have also been developed for crowded scenarios, such as pedestrian and vehicle detection [52,53]. Despite these innovations, methods aiming to eliminate NMS by improving the network architectures (i.e., NMS-free mechanisms) often exhibit a lower performance in practical applications compared to that of detectors that utilize NMS [15,54,55].

In this paper, we focus on the latest advancements in the label assignment algorithms, selecting both the Dynamic Soft Label Assigner and the Task-Aligned Assigner. While both are derived from the paradigms established by OTA and SimOTA, they exhibit notable algorithmic differences. Additionally, we use the Max IoU Assigner from Faster-RCNN as a representative of static label assignment methods. For the inference post process, we retain the standard NMS as our experimental approach.

2.2. The Hardware Accelerator in Inference Post Processing Algorithms

Hardware accelerators for the inference post process in edge object detection are crucial, with NMS being a key step. It selects the highest-confidence bounding box, computes the IoU using the other boxes, and filters those above a certain threshold, then repeating this until no boxes remain. Due to the computational complexity of IoU calculation, which involves multiple operations per comparison, significant power consumption and hardware resource usage occur. To address this, approximations can replace multiplications and divisions with simpler operations like additions and shifts, reducing the power and hardware demands. For instance, Shi et al. [56] approximate the IoU using coordinate maximums, and Fang et al. [57] introduce boundary constraints to eliminate division. However, these methods may reduce the model's accuracy.

To address the loss of model accuracy with IoU approximation, enhancements to the NMS algorithm can minimize redundant calculations and leverage parallelism without accuracy loss. Chen et al. [58] enhance NMS by routing the bounding boxes to three output head branches, each for a different ratio, and selecting the most suitable box after merging. This method increases the parallelism but faces challenges in maintaining consistent parallelism across different ratios. Choi et al. [59] improve NMS by limiting the number of valid bounding boxes in each IoU calculation, though this reduces adaptability in target detection.

Optimizing NMS by maintaining its core principles and using the threshold parameters to reduce IoU redundancy is another strategy for improving the algorithm's performance. Research [60–63] shows that applying confidence thresholds can significantly reduce the number of boxes involved in the IoU calculations. Anupreetham et al. [64] innovate by adapting NMS to SSD detectors, employing a pipelined system with three criteria for the box selection: class membership, IoU thresholds, and confidence levels. This approach incorporates class information into the selection process, unlike Shi et al.'s [56] method, which focused on coordinate-based clustering without addressing the class distribution. However, Anupreetham et al.'s approach does not fully address the IoU redundancy.

This work introduces an approach that prioritizes both class and confidence threshold checking to significantly reduce the IoU calculations between bounding boxes of different classes before performing the IoU computation. Table 1 provides a detailed comparison of recent studies highlighting these differences.

Table 1. Hardware accelerators for post processing.

	Standard IoU	Priority of Confidence Threshold Checking	Priority of Classification	Support for Training Post Process
2019-TCASII [56]	×	×	×	×
2021-ASSCC [57]	×	×	×	×
2021-FPL [60]	✓	✓	×	×
2021-ICCE [59]	✓	✓	*	×
2022-DATE [58]	×	✓	×	×
2023-FPT [61]	✓	✓	×	×
2023-ISCAS [62]	✓	✓	×	×
2023-TCASII [63]	×	✓	×	×
2024-TReTS [64]	✓	✓	*	×
Ours	✓	✓	✓	✓

* indicates that in this work, classification is typically prioritized after the IoU calculation, which leads to redundant IoU computations between bounding boxes of different classes. ✓ indicates that this work supports the technology. × indicates that this work does not support the technology.

2.3. The Hardware Accelerator in the Sorter Algorithm

Efficient sorting techniques are crucial for NMS and label assignment algorithms due to the need for sorting by the confidence scores and IoUs, as well as Top-K selection. Fang et al. [57] address sorting storage challenges using a comparator tree, which efficiently manages elements by focusing on the maximum confidence values in each iteration. Sun et al. [63] use parallel bitonic sorting to order the IoU values between bounding boxes. Bitonic sorting [65] offers faster sorting than traditional bubble sorting through parallel comparisons. It consists of two phases: the merge phase, where the input elements are organized into bitonic sequences with both ascending and descending subsequences, creating two subsequences at the end, and the sort phase, which recursively subdivides these subsequences until each is reduced to a length of one. While bitonic sorting improves the efficiency, its hardware implementation becomes costly with more elements due to the need for multiple comparators. Chen et al. [66] reduce the resource consumption by using dual insertion sorting after the merge phase, avoiding further recursion. Zhao et al. [67] introduce a smaller merge sort after the merge phase to facilitate Top-K extraction, particularly for K = 2. Alternatively, Fang et al. [68] use insertion sorting for Top-K identification, though this is limited by the sorting scale and longer delays.

In line with the hybrid bitonic sorting methods proposed by [66,67], our work introduces hybrid bitonic sorting that supports both full sorting and Top-K extraction. However, our method differs by incorporating multi-comparator trees during the merge phase to directly obtain the Top-K elements, instead of the traditional approach, which only identifies

the maximum value. Additionally, we overcome the K = 2 limitation present in previous methods, enabling greater parallelism due to the parallelizable nature of retrieving K elements. Table 2 presents the sorting schemes of recent sorting accelerators and provides a comparison.

Table 2. Sorting schemes.

	Sorter Type	Support for Full Sorting	Support for Top-K Sorting
-	Bitonic Sorter	✓	✓
2023-TCAD [68]	Insertion Sorter	×	✓
2023-TCASI [66]	Bitonic Sorter + Insertion Sorter	✓	✓
2024-TCAD [67]	Variant of Bitonic Sorters	×	✓
Ours	Bitonic Sorter + Sorter Tree	✓	✓

✓ indicates that this work supports the technology. × indicates that this work does not support the technology. Note: Supporting full sorting implies supporting any Top-K sorting.

3. The Unification of the Post Process Algorithm and Optimization

3.1. Analysis of the Uniformity of Post Processing Algorithms

In Faster-RCNN, the Max IoU Assigner begins by filtering out unsuitable bounding boxes based on a confidence threshold. It then calculates the IoU between the remaining bounding boxes and the ground truth boxes, selecting the bounding boxes that match the ground truth boxes according to predefined thresholds. Similarly, in Yolov8s, the Task-Aligned Assigner and GFL's Dynamic Soft Label Assigner first eliminate unsuitable bounding boxes based on whether their center points fall within the ground truth boxes. They then compute the alignment scores for each bounding box by combining the classification scores and IoUs. The Task-Aligned Assigner assigns a fixed number of bounding boxes to each ground truth target, whereas the Dynamic Soft Label Assigner adaptively determines the number of bounding boxes by selecting and summing the Top-K alignment scores for each ground truth target, making it more flexible.

For NMS, the process starts by selecting the bounding box with the highest confidence score from all of the candidate boxes. The IoU is then calculated between this highest-confidence box and the remaining boxes. Bounding boxes with an IoU exceeding a predefined threshold are removed. This process of selecting the highest-confidence box and recalculating the IoU is repeated until no bounding boxes remain.

From a procedural standpoint, as illustrated in Figure 4, we categorize the workflow into four stages: Pre Process, Alignment, Dynamic Sampling, and Sorter:

- Step 1. Pre Process: This step involves extracting the bounding box with the highest confidence or verifying whether a bounding box's center point is within the ground truth boxes. To be detailed, as shown in Figure 4, the light green background represents the Pre Process stage, where the input is data related to the bounding boxes, and the output is qualified bounding boxes that have undergone classification and confidence judgment and whose centers have been determined. The core element is the Pre Process Engine.
- Step 2. Alignment: This step calculates the IoU between the bounding boxes and the ground truth. Both the Task-Aligned Assigner and the Dynamic Soft Label Assigner additionally compute the alignment scores by integrating the classification values with the IoUs. As shown in Figure 4, the light pink background represents the Alignment stage, where the input includes the coordinate and category information on the qualified bounding boxes, as well as the coordinate and category information on the ground truth. Possible outputs include the IoU matrix and the alignment score matrix constructed based on the qualified bounding boxes and the ground truth. The

core elements are the IoU Engine, the Score Engine, the and Cost Engine, with the IoU Engine responsible for calculating the IoU matrix, the Score Engine responsible for calculating the classification score matrix, and the Cost Engine responsible for calculating the alignment score matrix.

- Step 3. Dynamic Sampling: This step determines the number of bounding boxes corresponding to each ground truth target, a requirement specific to the Dynamic Soft Label Assigner based on its alignment scores. In Figure 4, the light yellow background represents the Dynamic Sampling stage, where the input is the alignment score matrix, and the output is the maximum number of bounding boxes that each ground truth can be assigned to. The core element is the Top-K Sampling Engine, used only for the Dynamic Soft Label Assigner.
- Step 4. Sorter: This final step involves sorting and selecting the bounding boxes based on the IoU matrix or the alignment score matrix. In Figure 4, the light blue background represents the Sorter stage, where possible inputs include the IoU matrix, the alignment score matrix, and the number of bounding boxes that each ground truth can be assigned to, with the output being the assignment of the bounding boxes to each ground truth or true target.

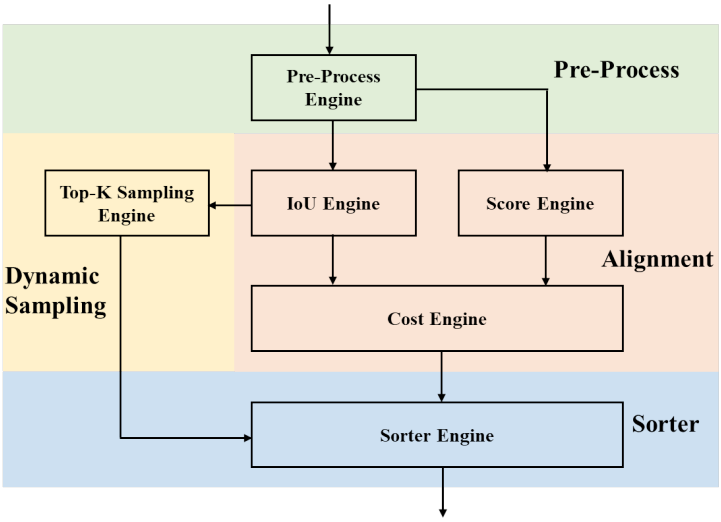

Figure 4. Unified implementation process for post processing algorithms.

Figure 5 illustrates the re-expression of the four types of algorithms within a unified workflow. The orange boxes highlight the enabled steps, while the light blue arrows indicate the flow of data.

- NMS: In the Pre Process stage, NMS identifies the bounding box with the highest confidence score. During the Alignment stage, it calculates the IoU between the selected bounding box and the remaining ones. In the Sorter stage, the IoUs are sorted, and the bounding boxes are filtered based on predefined IoU thresholds. This iterative process continues until no bounding boxes remain.
- Max IoU Assigner: This algorithm filters the bounding boxes based on the confidence in the Pre Process stage. It then computes the IoU matrix between the bounding boxes and the ground truth boxes in the Alignment stage. In the Sorter stage, the IoUs are sorted and used to filter the bounding boxes according to the IoU thresholds, mapping each bounding box to its corresponding ground truth box.
- The Task-Aligned Assigner: In the Pre Process stage, this algorithm filters the bounding boxes based on confidence. The Alignment stage involves calculating the IoU matrix and the alignment score matrix. During the Sorter stage, the alignment scores

are sorted, and bounding boxes are selected based on a fixed number of boxes per ground truth target, mapping them accordingly.

- The Dynamic Soft Label Assigner: This method filters the bounding boxes based on the confidence in the Pre Process stage. It calculates the IoU matrix and the alignment score matrix, using both the classification and IoU results, during the Alignment stage. The number of allowable bounding boxes per ground truth target is determined based on the alignment scores in the Dynamic Sampling stage. In the Sorter stage, the alignment scores are sorted, and bounding boxes are selected according to the number required for each ground truth target, effectively mapping them to the ground truth boxes.

Based on our analysis of the post processing algorithms, we propose a unified algorithmic workflow. By investigating the implementation details and characteristics of each algorithm, we examine their specific functionalities within this unified framework. This approach establishes a solid foundation for the hardware design.

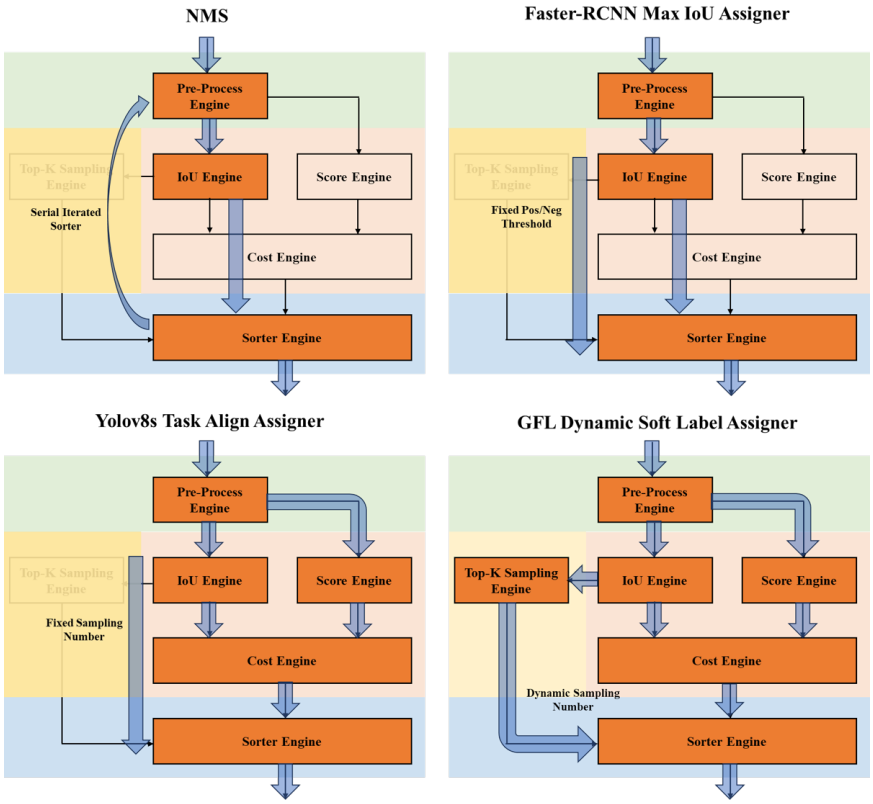

Figure 5. Re-expression of post processing algorithms in a unified implementation process.

3.2. Redundancy Analysis of Post Processing Algorithms

Both the training and inference post processes involve extensive IoU calculations. The IoU calculation involves computing the areas of two bounding boxes, A and B, and their overlapping region. The IoU is then defined as the ratio of the overlapping area to the total area covered by both bounding boxes, as expressed by Equation (1). Here, Equation (2) calculates the area S_A of A based on the width w_A and height h_A of A, Equation (3) calculates the area S_B of B based on the width w_B and height h_B of B, and Equation (4) marks the overlapping area of A and B as S_{cross}. Each IoU calculation requires three multiplications and one division. Consequently, optimizing the IoU calculation is crucial for efficiency. Unlike the methods proposed in [56,57], which simplify the IoU calculation to reduce the number of multiplications and divisions, our approach retains the standard

IoU computation. This choice ensures the accuracy of the alignment scores and maintains the algorithm's generalizability in the training post process.

$$\text{IoU} = \frac{S_{\text{cross}}}{S_A + S_B - S_{\text{cross}}} \tag{1}$$

$$S_A = w_A \times h_A \tag{2}$$

$$S_B = w_B \times h_B \tag{3}$$

$$S_{\text{cross}} = S_A \cap S_B \tag{4}$$

In post processing algorithms, IoU calculations are performed either between different bounding boxes or between bounding boxes and ground truth boxes. As depicted in Figure 6, the volume of such IoU calculations in post processing can reach the order of millions. Given the previously discussed principles of post processing, these calculations often involve redundancy. Consequently, eliminating redundant IoU calculations can significantly reduce the system's overall power consumption and improve its acceleration efficiency.

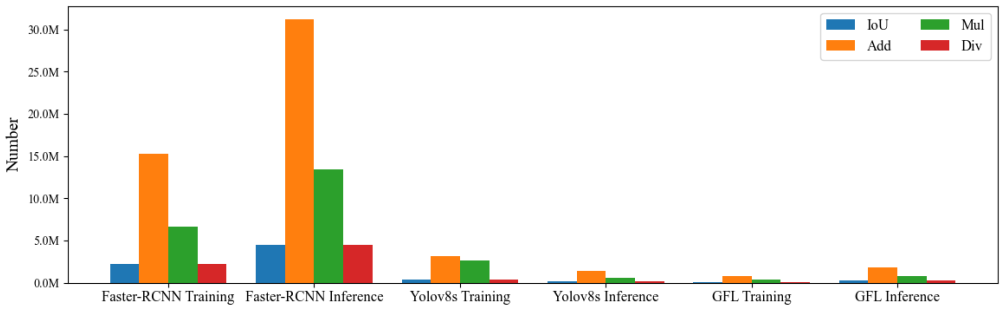

Figure 6. Operator count statistics for post processing algorithms.

In standard post processing algorithms, a substantial number of IoU calculations are performed between bounding boxes of different classes or between bounding boxes and ground truth boxes of different classes. However, many of these IoU calculations do not aid in the selection process. To address this inefficiency, distinguishing bounding boxes by class through a prior classification step can be advantageous. Unlike the approach taken in [64], our method incorporates classification before performing the IoU calculations. Specifically, this classification strategy is applied after confidence filtering or bounding box qualification but prior to the IoU computation.

3.3. Redundancy Optimizations for Post Processing Algorithms

Employing early classification prior to computing the IoUs can significantly reduce the number of redundant IoU calculations. For instance, in the inference phase, as depicted in Figure 7, the object detector's classification head generates eight bounding boxes, necessitating pairwise IoU calculations. This results in a total of 64 IoU computations. By incorporating early classification and limiting the IoU calculations to bounding boxes of the same class, the number of required IoU computations is reduced to 22. Consequently, this approach decreases the total IoU calculations to 34.4% of the original amount. Similarly, during the training phase, if 8 bounding boxes are returned and need to be matched with 5 ground truth boxes, the number of IoU calculations would otherwise total 40. However, with early classification, this number is reduced to 13, which is 32.5% of the original number.

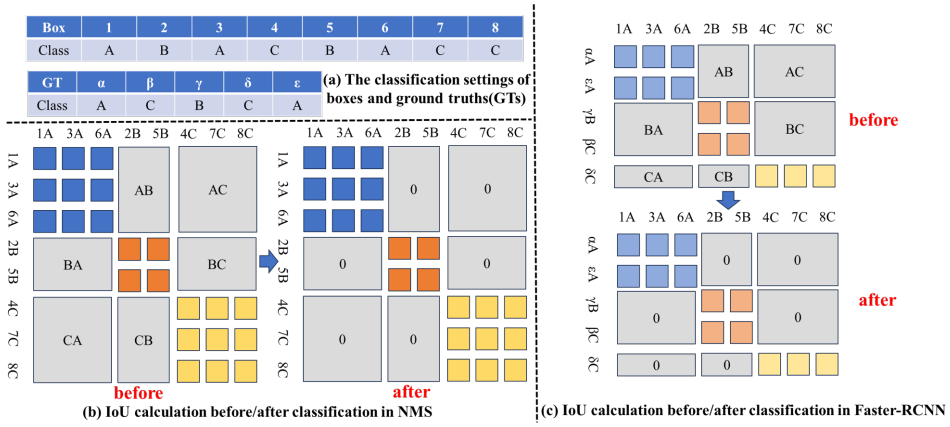

Figure 7. IoU calculation before and after classification: (**a**) classification settings for 8 boxes and 5 GTs, with categories A, B, and C. (**b**) In NMS inference, the IoUs are calculated between all boxes initially (64 calculations) but only within the same category after classification (22 calculations). (**c**) In Faster-RCNN training, the IoU is calculated between each box and GT initially (40 calculations) but only for the same category after classification (13 calculations).

In contrast to [56,57], our approach, consistent with several other studies, retains the confidence filtering in the NMS algorithm and preserves the bounding box center point position checking in the post processing phase. Despite incorporating early classification, we have not modified the post processing algorithms. Instead, we have reduced the redundant calculations by leveraging early classification.

4. Accelerator Framework Design

4.1. Overall Framework Design of the Accelerator

Based on the previous analysis of the uniformity and redundancy in the post processing algorithms, we identify that the post processes can be effectively structured into four key stages: the Pre Process, Alignment, Dynamic Sampling, and Sorter.

We propose a hardware accelerator architecture for the post processes, as illustrated in Figure 8. The system consists of a CPU for managing the read/write control registers, a DMA for transferring data to BRAM, BRAM which supplies data to the accelerator IP, the post processing accelerator IP, and a bus system for control and data transfer. The post processing accelerator IP is designed to implement the four stages: the Pre Process, Alignment, Dynamic Sampling, and Sorter. Control signals are utilized to identify whether the algorithm is in training or inference mode. In training mode, additional differentiation is made among the Max IoU Assigner, the Task-Aligned Assigner, and the Dynamic Soft Label Assigner. Data pathways are selected through enabling mechanisms, as depicted in Figure 5.

In our design, the data width is set to 8 bits, primarily because the precision loss during post processing is considerably less significant compared to that in the feature processing network layers. Consequently, this choice does not markedly affect the final detection accuracy, with an observed average accuracy loss of only 0.02%. We process 8 ground truth boxes and 64 bounding boxes in a single pass. The input data comprise two coordinates for each GT, center point coordinates, width and height dimensions for each bounding box, and the category data and confidence scores for each box. These data are written to the corresponding BRAM write ports via DMA, while the read ports of BRAM supply data to the post processing accelerator IP.

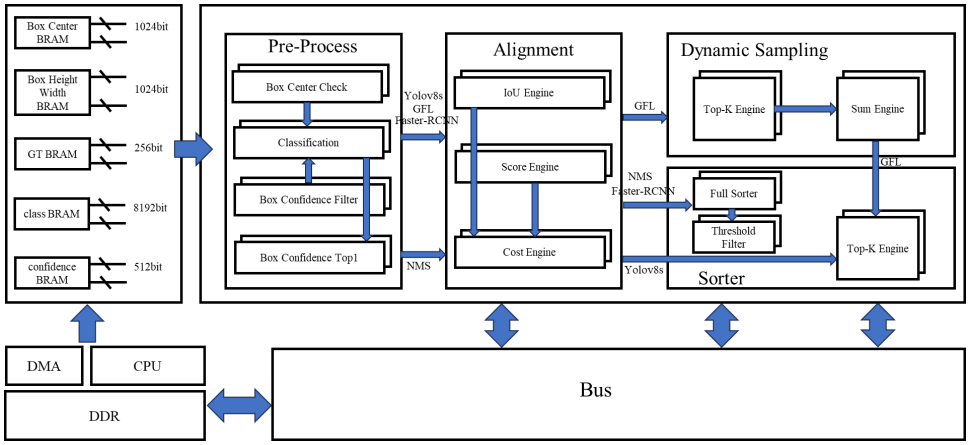

Figure 8. Hardware accelerator architecture for training and inference post processes: Blue arrows show data flow direction. Labeled arrows (YOLOv8s, GFL, Faster-RCNN, and NMS) relate to these algorithms. Unlabeled arrows are unrelated to any of the four algorithms.

4.2. The Design of the Hybrid Bitonic Sorter

The Top-K Engine and the Full Sorter within the post processing accelerator (shown in Figure 8) all require sorting. In this paper, we propose an innovative hybrid sorting approach based on bitonic sorters. During the merge phase of bitonic sorting, the ascending and descending subsequences can be utilized to pinpoint the maximum values. Moreover, leveraging the known order relationships within these subsequences, a comparator tree can efficiently determine the Top-2 and Top-3 maximum values. This approach allows us to replace the traditional sorting phases with a comparator tree, thus minimizing unnecessary sorting operations and facilitating faster retrieval of the Top-K data. Our sorter design features a fundamental unit comprising a 32-input sorter, capable of managing up to 16 Top-K elements, as demonstrated in Figure 9.

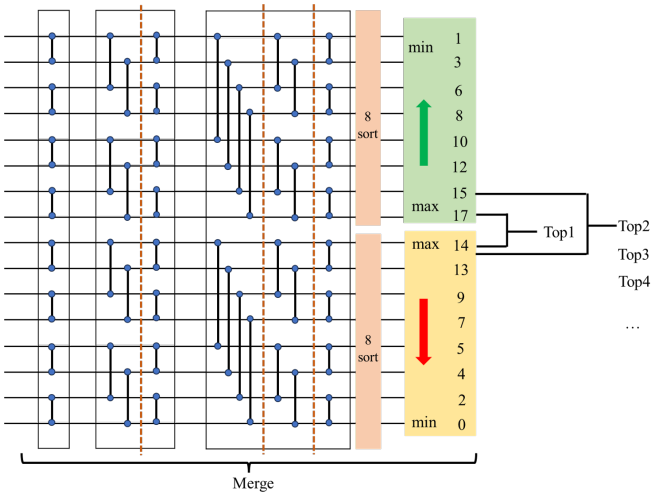

Figure 9. Hybrid bitonic sorter. An ascending subsequence and a descending subsequence (each containing 8 elements) obtained through the merge phase can be used to determine the Top-1 element by comparing the maximum values of the two subsequences. The Top-2 element can be found by comparing the second largest values in both subsequences, excluding the Top-1 element.

4.3. The Design of the Pre Process Submodule

As illustrated in Figure 10, the bounding boxes are derived from different ratio layers centered at various points (denoted as × on the left side of Figure 10). By sliding across the image frames (i.e., moving × from left to right as shown), the number of bounding boxes that needs to be compared or filtered can be significantly reduced. This method is effective

because a bounding box in the bottom-right corner of the image is unlikely to overlap with one in the top-left corner. Moreover, this approach tends to group bounding boxes near the same center point into a limited number of categories, thereby facilitating the formation of clusters of bounding boxes within the same class. We utilize this sliding technique for bounding boxes around the input center points in our approach.

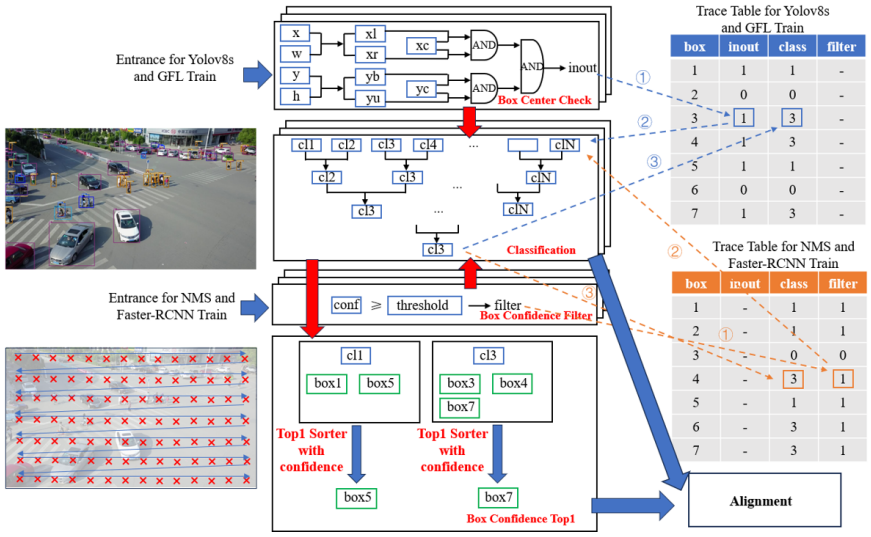

Figure 10. The design of the Pre Process submodule.

In the Pre Process submodule, the data path varies according to the algorithm type, as depicted in Figure 10. The input to the Pre Process submodule can come from bounding-box-related data in both the YOLOv8s training post process and the GFL training post process, as well as from the NMS and Faster-RCNN training post process. For the former, the data are processed in the Box Center Check to determine whether the bounding box's center point lies within the ground truth box. Here, x, w, y, and h represent the x-coordinate of the ground truth center point, the ground truth's horizontal width, the y-coordinate of the ground truth center point, and the ground truth's vertical height, respectively. xl, xr, yb, and yu represent the left x-coordinate, right x-coordinate, bottom y-coordinate, and top y-coordinate of the ground truth, calculated based on x, w, y, and h. xc and yc represent the x-coordinate and y-coordinate of the bounding box's center point, respectively. The output results are entered into the Trace Table's inout column—for example, ①, where 1 indicates that the bounding box's center point is within the ground truth, and 0 indicates it is not. For the latter, the bounding box's confidence is filtered using the Box Confidence Filter using a confidence threshold, and the results are filled into the filter column of the Trace Table—for example, ①, where 1 indicates confidence above the threshold, and 0 indicates confidence below the threshold. Subsequently, all four algorithms proceed to Classification, where they are categorized using a classification tree. For example, both ② and ② send the probability of each category for qualified bounding boxes into Classification, and the results are written into the Trace Table, such as ③ and ③, which record the category judgment results into the corresponding class column of the Trace Table. With this, the Pre Process for the post processes of YOLOv8s, GFL, and Faster-RCNN training is complete, and the Trace Table can be passed on to the Alignment submodule. NMS, however, still needs to send bounding boxes of the same category to Box Confidence Top-1 to obtain the bounding box with the highest confidence for each category, such as box5 and box7, these being the bounding boxes with the highest confidence in the first and third categories, respectively, and then passing the identifier and the Trace Table to the Alignment submodule. The Top-1 Sorter with confidence still employs the classification tree. Figure 11 shows the circuit

implementation of the corresponding key modules. Data Supply provides the input data for the four algorithms. Yolov8s and GFL use the Box Center Check to calculate xr, xl, yu, and yb, comparing them with xc and yc. Three AND gates determine whether the GT box contains the center point (xc, yc). Faster-RCNN and NMS employ the Box Confidence Filter to compare conf with the threshold, outputting filter=1 if conf exceeds this threshold. The Classification and Box Confidence Top-1 modules share multiple 32-input comparator trees. The first input port is linked to Module A and the other 31 ports to Module B. To handle cases with over 32 comparison objects and find the maximum value, the current round's maximum value is re-input into the first port for subsequent rounds until all objects are compared. Module A uses a counter and a priority arbiter to decide between re-inputting the previous round's maximum or a new comparison object (new conf/new cls) into the first port. Module B decides between inputting the remaining or new comparison objects (conf/cls or new conf/new cls) into the other 31 ports using counters and priority arbiters.

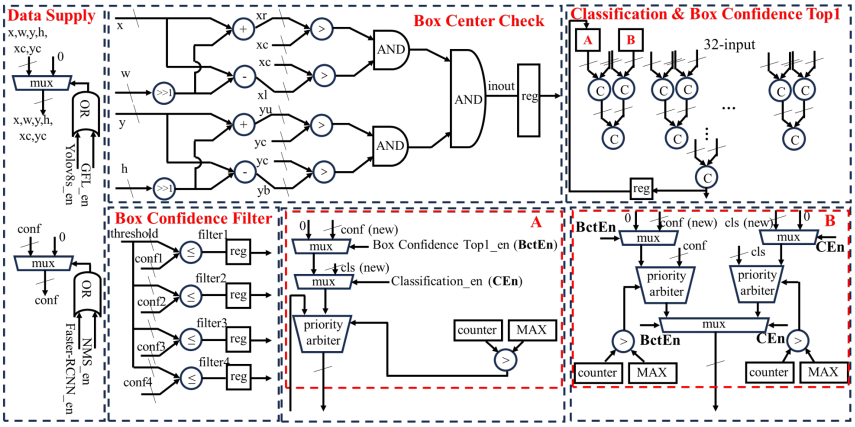

Figure 11. The circuit implementation of the Pre Process submodule.

4.4. The Design of the Alignment Submodule

As shown in the left part of Figure 12, the data flow within the Alignment submodule differs based on the algorithm used. Taking the training post processes of YOLOv8s and GFL as examples, in the IoU Engine, the GT coordinates for class 1 and class 3 serve as one input for the IoU calculation, while the other input comes from the coordinates of bounding boxes of the same class, thereby obtaining the IoU between the GT and the bounding boxes. Subsequently, in the Score Engine, the category of the bounding box is input to obtain the cross-entropy category score. Due to the computational complexity of the logarithmic operations in the cross-entropy function, a lookup table is utilized within the module for this calculation. Following this, in the Cost Engine, the alignment score for GFL is derived by summing the classification score and the IoU. For YOLOv8s, the alignment score is obtained by multiplying the classification score and the IoU after applying a power function with an exponent of 1, which does not significantly affect the model's accuracy. Finally, GFL sends the alignment score to the Dynamic Sampling submodule, while YOLOv8s sends the alignment score to the Sorter submodule. In contrast, for NMS and Faster-RCNN's training post process, the Score Engine and the Cost Engine are not required. NMS uses the coordinates of the highest-confidence bounding boxes output by the Pre Process submodule as one input for IoU calculation, with the other input being the coordinates of the other bounding boxes of the same class. Faster-RCNN, on the other hand, uses the GT coordinates for class 1 and class 3 as one input for the IoU calculation, with the other input coming from the coordinates of bounding boxes of the same class, thereby obtaining the IoU between the GT and the bounding boxes. The IoU obtained from NMS and Faster-RCNN's post processes through the IoU Engine is directly sent to the Sorter submodule. As shown in

the right part of Figure 12, the IoU Unit, essential for all four algorithms, calculates areas $S1$ and $S2$ from the dimensions of two bounding boxes, determines the overlap area's dimensions (Δh and Δw), and computes the overlap area $Scross$. The IoU is then calculated. Yolov8s and GFL use the Score Engine to find scores in the LUT based on class probabilities. Yolov8s employs the Cost Engine for element-wise multiplication, while GFL performs element-wise addition, with the selection paths controlled by enable signals.

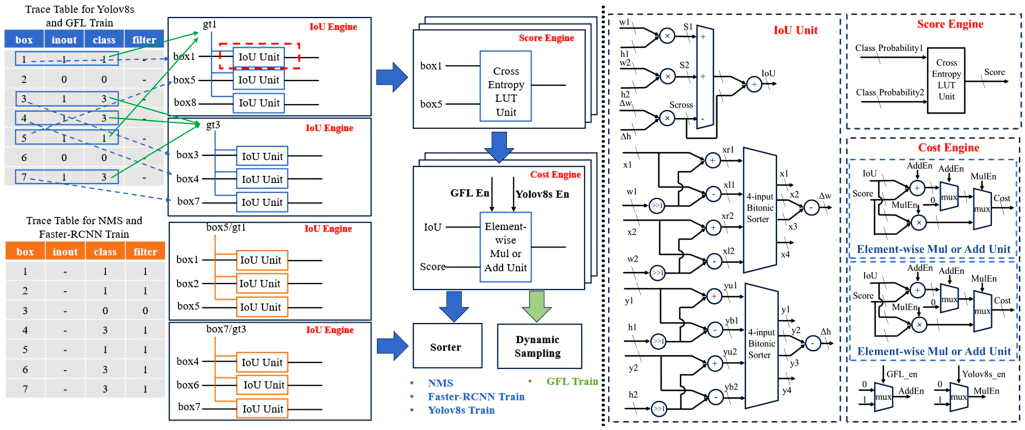

Figure 12. The design and circuit implementation of the Alignment submodule.

4.5. The Design of the Dynamic Sampling and Sorter Submodules

As shown in the left part of Figure 13, the Dynamic Sampling submodule is exclusively used for the training post process in GFL. The alignment scores between each bounding box and ground truth (i.e., the Cost Table) are obtained from the Cost Engine of the Alignment submodule. These alignment scores are then processed by the Top-K Engine and the Sum Engine within the Dynamic Sampling module, as shown in Figure 8, yielding the results presented in the blue table of Figure 13. The Top-K Engine first selects the top-K maximum cost values between the ground truth and the bounding boxes, and then the Sum Engine calculates the Sum values for each gt in the blue table in Figure 13. For example, for gt1, the top-K maximum numbers are only 0.8 and 0.3, so they sum up to 1.1, and thus the Sum for gt1 is filled with 1.1. To determine the maximum number of bounding boxes, Num, that can be assigned to each gt, a decision is made based on whether the decimal part of the fixed-point number is zero—if it is zero, the integer part is taken, such as 1.0 being taken as 1; if it is not zero, the integer part is incremented by one, such as 1.1, where the integer part is incremented by one to get 2. This result is stored in the buffer of the Dynamic Sampling submodule and subsequently passed to the Sorter submodule.

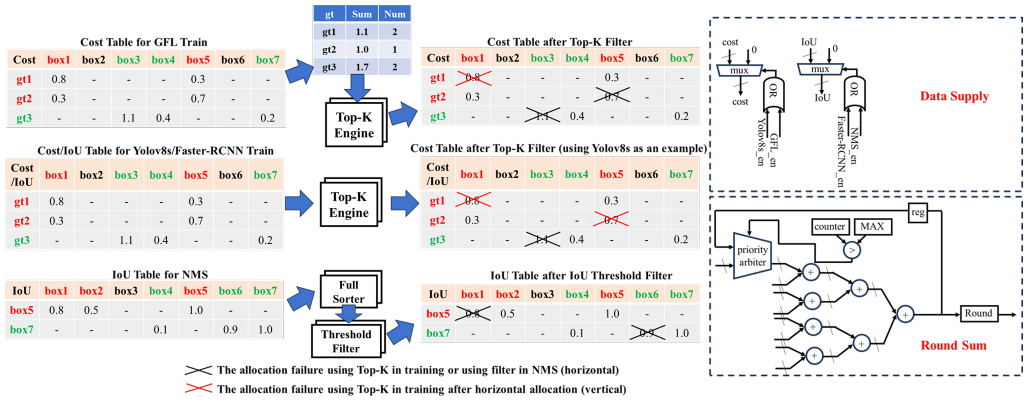

Figure 13. The design and circuit implementation of the Dynamic Sampling and Sorter submodules.

For the Sorter submodule, the inputs vary among the four types of algorithms. For GFL, the number of boxes, Num, corresponding to each gt and the Cost Table is used as the input, with boxes being assigned to the appropriate gt through the Top-K Engine; for YOLOv8s, the Cost Table is used as the input, and boxes are assigned to the appropriate gt through the Top-K Engine; for Faster-RCNN and NMS, the IoU Table is used as the input. For Faster-RCNN, boxes are assigned to the appropriate gt using the Top-K Engine, while for NMS, the most suitable boxes for representing the true targets are selected through the Full Sorter and the Threshold Filter. Taking GFL as an example, gt1, gt2, and gt3 are allowed to be assigned up to 2, 1, and 2 boxes, respectively. Therefore, gt1 can initially be assigned to box1 and box5; gt2 can be assigned to box1 (box5, due to its large cost value and the fact that gt2 can be assigned to at most one box, is not assigned to gt2, which is referred to as the allocation failure using Top-K in training or using a filter in NMS); and gt3 can be assigned to box4 and box7. Since the same box cannot be assigned to two gts, gt1 and gt2 compete for box1. According to the principle of the smallest cost, gt2 obtains box1, and gt1 loses its box1 (this is referred to as the allocation failure using Top-K in training after horizontal allocation). The same applies to YOLOv8s and Faster-RCNN. For NMS, the Full Sorter first completes the IoU sorting between each high-confidence box and the other boxes within the same category, and then the Threshold Filter removes the bounding boxes above the IoU threshold. As shown in the right part of Figure 13, the Data Supply feeds the data, with Yolov8s and GFL needing the cost and NMS and Faster-RCNN using the IoU. GFL determines the max bounding boxes per gt via Round Sum. The Top-K Engine employs a hybrid bitonic sorter, and the Full Sorter uses a bitonic sorter. The Threshold Filter adapts the Box Confidence Filter from Figure 11, applying the IoU threshold and using the IoU input instead of confidence.

5. Experiments and Result Analysis

5.1. The Experimental Setup

We deployed the solution on the ZCU102 platform. Vivado was used to build the IPs and SoCs, and the Vitis SDK qA utilized to implement the post processing algorithms.

For the IP design and SoC construction, we used the Xilinx Central DMA IP for the data transfer and the Xilinx AXI BRAM Controller for memory management. Radix-8 Booth multipliers replace simple Verilog multiplications, using shift and add operations. The post processing accelerator is an AXI Slave module connected via AXI Interconnect, configured by the CPU for NMS, Faster-RCNN, Yolov8s, and GFL. The FPGA resource constraints limit the parameter configurability for GFL and Yolov8s, allowing only specific parameters. However, the system remains software-programmable.

For algorithm implementation, the Vitis SDK implements the post processing algorithms in CNN object detection. Data are exported from PyTorch-based CNN layers using the *print* API, written to the main memory using Vitis SDK's write commands, and loaded into the accelerator using read commands. During acceleration, the post processing algorithms are managed with write and read commands for the registers to ensure accurate computations.

In terms of the data distribution and memory mapping, the accelerator's memory space allocates separate address spaces to the confidence, coordinates, and classes of the bounding boxes and ground truths. Storing them as intact units maintains their integrity, simplifying memory management and data access. Consecutive storage of these values streamlines the data retrieval and optimizes the acceleration performance.

5.2. Redundancy Result Analysis

By employing the priorities of CTC-C and CPC-C, we effectively reduce redundant calculations. As shown in Figure 14, we assess the impact of classification on the redundancy reduction using the COCO dataset. Applying the priority of CTC-C and CPC-C alone reduces the total number of bounding boxes to 12.9% and 11.9% of the original counts (original), respectively. Furthermore, incorporating classification further reduces the maximum number of bounding boxes per category (the maximum reduction in the category) to between 4.4% and 19.1% of the original number (original). These results confirm the effectiveness of our methods.

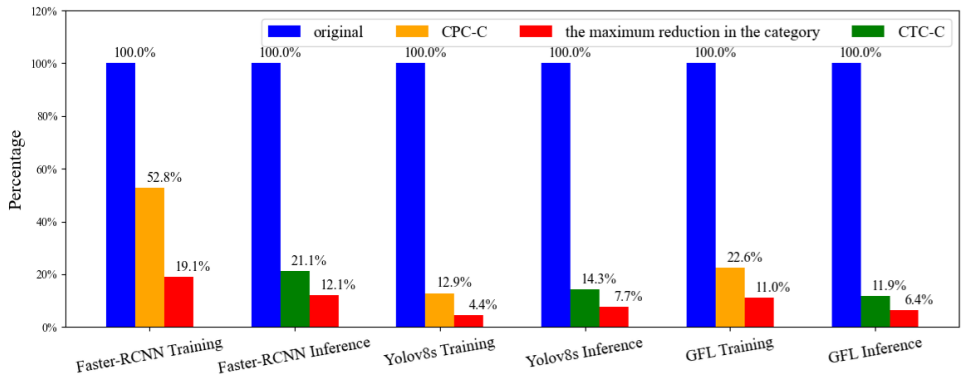

Figure 14. Analysis of reducing redundant bounding boxes using priorities of CTC-C and CPC-C.

5.3. Analysis of the Results for Sorter Optimization

We conduct a comparative analysis of several sorting algorithms, focusing on the sorting cycles and resource utilization. Figure 15 presents a comparison of the resource consumption between our proposed sorter and other existing sorters. Notably, parallel bubble sorting exhibits substantially higher resource requirements. In contrast, our proposed sorter maintains a moderate level of resource usage, demonstrating more efficient use of resources.

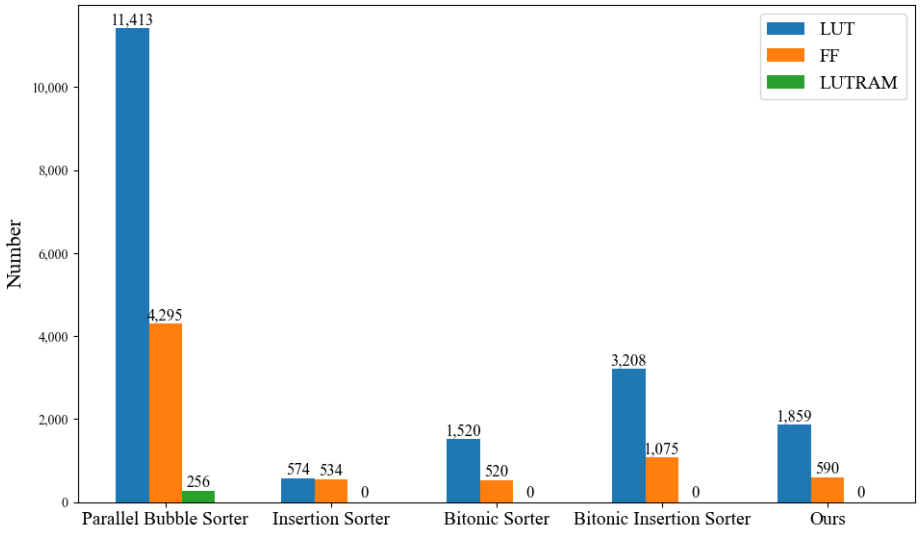

Figure 15. Comparison of resource utilization between our proposed sorter and existing sorter solutions.

Table 3 presents the cycle data, which detail the number of cycles needed for full sorting, finding the maximum value among 32 elements, and identifying the top 16 values among 32 elements. Although parallel bubble sorting demonstrates shorter sorting cycles, it requires substantial hardware resources due to its complexity. In contrast, our

proposed sorting solution uses only a modest amount of additional resources for Top-K element extraction compared to a basic bitonic sorter. When evaluating both the resource consumption and cycle efficiency, our proposed sorting solution offers a well-balanced performance across all of the compared schemes.

Table 3. Comparison of sorting cycles between our proposed sorter and existing sorting solutions.

Sorter	Sorter Type	Cycle
Parallel Bubble Sorter	Bubble Sorter	7, 7@32:1, 7@32:16
2023-TCAD [68]	Insertion Sorter	261, 261@32:1, 261@32:16
Bitonic Sorter	Bitonic Sorter	21, 21@32:1, 21@32:16
2023-TCASI [66]	Bitonic Insertion Sorter	106, 15@32:1. 106@32:16
Ours	Hybrid Bitonic Sorter	21, 15@32:1, 21@32:16

Note : In the context of cycle data, 7, 7@32:1, 7@32:16 represent the number of cycles required for full sorting, selecting the top-1 out of 32, and selecting the top-16 out of 32, respectively.

5.4. Analysis of the Results for the Post Processing Accelerator

We compare the performance, power, and resource usage of various post processing accelerators within the same dataset, as detailed in Table 4. In Table 4, the variations among the compared items are due to the types of sorters, with the hardware architecture differing only in the sorter implementations. The power data are directly derived from the power results provided by Vivado for the specified simulation activity files after the implementation, which include static power and dynamic power. Our proposed hybrid bitonic sorter achieves performance improvements of up to 1.25 times and no less than 1.05 times compared to that of the solution presented in [66]. Relative to the original bitonic sorter, our implementation provides faster post processes for both inference and training.

Table 4. Comparison of performance between our proposed sorter and existing sorting solutions.

	2023-TCAD [68]	**Bitonic Sorter**	**2023-TCASI [66]**	**Ours**
Sorter Type	Insertion Sorter	Bitonic Sorter	Bitonic Insertion Sorter	Hybrid Bitonic Sorter
Frequency (MHz)	200	200	200	200
Time (ms)	0.416@F 0.208@G 0.206@Y 0.388@F-NMS 0.220@G-NMS 0.210@Y-NMS	0.252@F 0.184@G 0.186@Y 0.255@F-NMS 0.192@G-NMS 0.188@Y-NMS	0.309@F 0.193@G 0.193@Y 0.304@F-NMS 0.202@G-NMS 0.196@Y-NMS	0.248@F 0.182@G 0.187@Y 0.253@F-NMS 0.190@G-NMS 0.186@Y-NMS
Static Power (W)	0.222	0.218	0.228	0.216
Dynamic Power (W)	2.806	2.010	3.512	2.105
LUT	204,200	158,329	351,365	166,417
FF	154,203	135,901	181,732	136,652

F denotes Faster-RCNN, G denotes GFL, and Y denotes Yolov8s in the training post process. F-NMS denotes Faster-RCNN's NMS, G-NMS denotes GFL's NMS, and Y-NMS denotes Yolov8s's NMS in the inference post process.

Due to the differing degrees of redundancy among the various solutions, we evaluate the effectiveness of the redundancy reduction across different algorithms using our proposed accelerator scheme. Table 5 shows the differences in the types of redundancy optimization algorithms, with the overall hardware architecture remaining the same across all items. Our solution provides a maximum speedup of 1.19 times and a minimum speedup of 1.10 times.

Table 5. Comparing different redundancy schemes on our proposed post processing accelerator.

	2023-TCASII [63]	**2024-TReTS [64]**	**Ours**
Confidence Threshold Checking in Inference	✓	✓	✓
Box Center Position Checking in Training	✓	✓	✓
Classification Priority	×	*	✓
Frequency (MHz)	200	200	200
Time (ms)	0.295@F 0.217@G 0.209@Y 0.368@F-NMS 0.228@G-NMS 0.209@Y-NMS	0.272@F 0.214@G 0.212@Y 0.287@F-NMS 0.226@G-NMS 0.209@Y-NMS	0.248@F (1.10×) 0.182@G (1.18×) 0.187@Y (1.13×) 0.253@F-NMS (1.13×) 0.190@G-NMS (1.19×) 0.186@Y-NMS (1.12×)

✓ indicates that this work supports the technology. × indicates that this work does not support the technology. * Classification is typically prioritized after the IoU calculation, which leads to redundant IoU computations between regression boxes of different classes. Note: The meaning of the data highlighted in red is the ratio of "2024-TReTS" to the corresponding items in "Ours"—for example, 0.272 divided by 0.248 equals 1.10.

Table 6 compares our approach with recent studies. Notably, our work is among the few that retrieve data from DDR and load it onto the accelerator while adhering to the official bounding box specifications. We considered two modes: IP simulation, assuming data are always available in the accelerator, and DMA transfer from DDR, aligning with system testing. Our solution achieves at least a 7.55× speedup in NMS using DDR. For the IP simulation alone, it offers greater acceleration, especially with many candidate boxes. Compared to systems using an RTX 2080 Ti, our solution provides at least a 21.93× speedup in the training post process and 19.89x in the inference post process.

Table 6. Comparison with other post processing accelerators.

	2020-ISVLSI [69]	**2021-ICCE [59]**	**2022-DATE [58]**	**2023-FPT [61]**	**Ours**	
Support Train	×	×	×	×	✓	
IoU Accuracy Expression	✓	✓	✓	✓	✓	
Testing Mode	IP Simulation	Data From DDR	IP Simulation	IP Simulation	Data Fetched by DMA From DDR	IP Simulation
Frequency (MHz)	100	-	400	150	200	
Time (ms)	0.032@NMS	1.91@NMS 5.19@NMS	0.05@NMS	0.014@NMS	0.248@F 0.182@G 0.187@Y 0.253@F-NMS 0.190@G-NMS 0.186@Y-NMS	0.080@F 0.014@G 0.015@Y 0.083@F-NMS 0.022@G-NMS 0.017@Y-NMS
Box Number	3000	-	1000	2880	22,000@F 8400@G 6300@Y	
Selected Boxes	5	11/48	5	24	10	
Platform	ZYNQ7-ZC706	ZCU106	Gensys2	Virtex7 690T	ZCU102	
LUT	-	-	14,188	26,404	166,417	
FF	11,139	-	2924	21,127	136,652	
LUTRAM	714	-	7504	2	0	
DSP	22	-	-	450	0	

✓ indicates that this work supports the technology. × indicates that this work does not support the technology. - indicates that data cannot be obtained.

6. Conclusions

This paper provides a detailed analysis of the training and inference post processes, accelerating the algorithms from a software–hardware co-design perspective. On the software side, we introduce the priorities of CPC-C and CTC-C to address the inefficiency of the IoU calculations, design a hybrid bitonic sorter to reduce the number of redundant comparisons, and unify four post processing algorithms. On the hardware side, we have designed a hardware accelerator for these post processing algorithms.

Author Contributions: Conceptualization: D.Y. and L.C. Methodology: D.Y. Software: D.Y. and Y.Z. Validation: D.Y., X.H. and Y.Z. Formal analysis: D.Y., L.C. and X.H. Investigation: D.Y. Resources: L.C. Writing—original draft preparation: D.Y. Writing—review and editing: L.C. Visualization: D.Y. Supervision: L.C. Project administration: L.C. Funding acquisition: L.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Key Research and Design Plan of Chinese Academy of Sciences under Grant 2022YFB4400404 (Corresponding author: Lan Chen).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. These data can be found here: https://cocodataset.org/ (accessed on 1 May 2014).

Acknowledgments: The first author, D.Y., hereby acknowledges the Institute of Microelectronics of the Chinese Academy of Sciences (IMECAS) and the EDA Center.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Hu, Y.; Yang, J.; Chen, L.; Li, K.; Sima, C.; Zhu, X.; Chai, S.; Du, S.; Lin, T.; Wang, W.; Lu, L.; Jia, X.; Liu, Q.; Dai, J.; Qiao, Y.; Li, H. Planning-oriented Autonomous Driving. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–23 June 2023; pp. 17853–17862.
2. Zhou, X.; Lin, Z.; Shan, X.; Wang, Y.; Sun, D.; Yang, M.H. DrivingGaussian: Composite Gaussian Splatting for Surrounding Dynamic Autonomous Driving Scenes. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 17–21 June 2024; pp. 21634–21643.
3. Weiss, M.; Jacob, F.; Duveiller, G. Remote sensing for agricultural applications: A meta-review. *Remote Sens. Environ.* **2020**, *236*, 111402. [CrossRef]
4. Vandome, P.; Leauthaud, C.; Moinard, S.; Sainlez, O.; Mekki, I.; Zairi, A.; Belaud, G. Making technological innovations accessible to agricultural water management: Design of a low-cost wireless sensor network for drip irrigation monitoring in Tunisia. *Smart Agric. Technol.* **2023**, *4*, 100227. [CrossRef]
5. Chakraborty, R.; Kereszturi, G.; Pullanagari, R.; Durance, P.; Ashraf, S.; Anderson, C. Mineral prospecting from biogeochemical and geological information using hyperspectral remote sensing—Feasibility and challenges. *J. Geochem. Explor.* **2022**, *232*, 106900. [CrossRef]
6. Wang, R.; Chen, F.; Wang, J.; Hao, X.; Chen, H.; Liu, H. Prospecting criteria for skarn-type iron deposits in the thick overburden area of Qihe-Yucheng mineral-rich area using geological and geophysical modelling. *J. Appl. Geophys.* **2024**, 105442.
7. Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 11–17 October 2021; pp. 10012–10022.
8. Lee, Y.; Hwang, J.; Lee, S.; Bae, Y.; Park, J. An energy and GPU-computation efficient backbone network for real-time object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR), Long Beach, CA, USA, 16–20 June 2019.
9. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path aggregation network for instance segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 8759–8768.
10. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.

11. Duan, K.; Bai, S.; Xie, L.; Qi, H.; Huang, Q.; Tian, Q. Centernet: Keypoint triplets for object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27–30 October 2019; pp. 6569–6578.
12. Li, X.; Wang, W.; Wu, L.; Chen, S.; Hu, X.; Li, J.; Tang, J.; Yang, J. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 21002–21012.
13. Li, X.; Wang, W.; Hu, X.; Li, J.; Tang, J.; Yang, J. Generalized focal loss v2: Learning reliable localization quality estimation for dense object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 16–22 June 2021; pp. 11632–11641.
14. Zhang, S.; Chi, C.; Yao, Y.; Lei, Z.; Li, S. Z. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 9759–9768.
15. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-end object detection with transformers. In Proceedings of the European Conference on Computer Vision (ECCV), Glasgow, UK, 23–28 August 2020; pp. 213–229.
16. Zhang, S.; Li, C.; Jia, Z.; Liu, L.; Zhang, Z.; Wang, L. Diag-IoU loss for object detection. *IEEE Trans. Circuits Syst. Video Technol.* **2023**, *33*, 7671–7683. [CrossRef]
17. Basalama, S.; Sohrabizadeh, A.; Wang, J.; Guo, L.; Cong, J. FlexCNN: An end-to-end framework for composing CNN accelerators on FPGA. *ACM Trans. Reconfigurable Technol. Syst.* **2023**, *16*, 1–32. [CrossRef]
18. Jia, X.; Zhang, Y.; Liu, G.; Yang, X.; Zhang, T.; Zheng, J.; Xu, D.; Liu, Z.; Liu, M.; Yan, X. XVDPU: A High-Performance CNN Accelerator on the Versal Platform Powered by the AI Engine. *ACM Trans. Reconfigurable Technol. Syst.* **2024**, *17*, 1–24. [CrossRef]
19. Wu, C.; Wang, M.; Chu, X.; Wang, K.; He, L. Low-precision floating-point arithmetic for high-performance FPGA-based CNN acceleration. *ACM Trans. Reconfigurable Technol. Syst. (TRETS)* **2021**, *15*, 1–21.
20. Wu, D.; Fan, X.; Cao, W.; Wang, L. SWM: A high-performance sparse-winograd matrix multiplication CNN accelerator. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2021**, *29*, 936–949. [CrossRef]
21. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *39*, 1137–1149. [CrossRef] [PubMed]
22. NanoDet-Plus: Super Fast and High Accuracy Lightweight Anchor-Free Object Detection Model. Available online: https://github.com/RangiLyu/nanodet (accessed on 8 January 2024).
23. Feng, C.; Zhong, Y.; Gao, Y.; Scott, M.; Huang, W. Tood: Task-aligned one-stage object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021; pp. 3490–3499.
24. Lin, J.; Zhu, L.; Chen, W.; Wang, W.; Gan, C.; Han, S. On-device training under 256kb memory. In Proceedings of the Advances in Neural Information Processing Systems (NIPS), New Orleans, LA, USA, 28 November–9 December 2022; pp. 22941–22954.
25. Lyu, B.; Yuan, H.; Lu, L.; Zhang, Y. Resource-constrained neural architecture search on edge devices. *IEEE Trans. Netw. Sci. Eng.* **2021**, *9*, 134–142. [CrossRef]
26. Zhao, Z.; Zheng, P.; Xu, S.; Wu, X. Object detection with deep learning: A review. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3212–3232. [CrossRef]
27. Tan, M.; Pang, R.; Le, Q. Efficientdet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 10781–10790.
28. Zhao, Y.; Lv, W.; Xu, S.; Wei, J.; Wang, G.; Dang, Q.; Liu, Y.; Chen, J. Detrs beat yolos on real-time object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 17–21 June 2024; pp. 16965–16974.
29. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.; Berg, A. SSD: Single shot multibox detector. In Proceedings of the 14th European Conference on Computer Vision (ECCV), Amsterdam, The Netherlands, 10–16 October 2016; pp. 21–37.
30. Ross, T.Y.; Dollár, P.; He, K.; Hariharan, B.; Girshick, R. Focal loss for dense object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2980–2988.
31. Cai, Z.; Vasconcelos, N. Cascade R-CNN: Delving into high quality object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 6154–6162.
32. Law, H.; Deng, J. CornerNet: Detecting objects as paired keypoints. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 734–750.
33. Zhou, X.; Zhuo, J.; Krahenbuhl, P. Bottom-up object detection by grouping extreme and center points. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 850–859.
34. Liu, Z.; Zheng, T.; Xu, G.; Yang, Z.; Liu, H.; Cai, D. Training-time-friendly network for real-time object detection. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), New York, NY, USA, 7–12 February 2020; pp. 11685–11692.
35. Ghahremannezhad, H.; Shi, H.; Liu, C. Object detection in traffic videos: A survey. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 6780–6799. [CrossRef]
36. Geng, X.; Su, Y.; Cao, X.; Li, H.; Liu, L. YOLOFM: An improved fire and smoke object detection algorithm based on YOLOv5n. *Sci. Rep.* **2024**, *14*, 4543. [CrossRef] [PubMed]

37. Tian, Z.; Shen, C.; Chen, H.; He, T. FCOS: Fully convolutional one-stage object detection. *arXiv* **2019**, arXiv:1904.01355.
38. Xu, C.; Wang, J.; Yang, W.; Yu, H.; Yu, L.; Xia, G. RFLA: Gaussian receptive field based label assignment for tiny object detection. In Proceedings of the European Conference on Computer Vision (ECCV), Tel-Aviv, Israel, 23–27 October 2022; pp. 526–543.
39. Kim, K.; Lee, H. Probabilistic anchor assignment with IoU prediction for object detection. In Proceedings of the 16th European Conference on Computer Vision (ECCV), Glasgow, UK, 23–28 August 2020; pp. 355–371.
40. Ge, Z.; Liu, S.; Li, Z.; Yoshie, O.; Sun, J. OTA: Optimal transport assignment for object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 19–25 June 2021; pp. 303–312.
41. Ge, Z.; Liu, S.; Wang, F.; Li, Z.; Sun, J. YOLOX: Exceeding YOLO series in 2021. *arXiv* **2021**, arXiv:2107.08430.
42. Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; Dai, J. Deformable DETR: Deformable transformers for end-to-end object detection. *arXiv* **2020**, arXiv:2010.04159.
43. Pu, Y.; Liang, W.; Hao, Y.; Yuan, Y.; Yang, Y.; Zhang, C.; Hu, H.; Huang, G. Rank-DETR for high quality object detection. In Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023), New Orleans, LA, USA, 10–16 December 2023.
44. Giroux, J.; Bouchard, M.; Laganiere, R. T-fftradnet: Object detection with Swin vision transformers from raw ADC radar signals. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France, 2–6 October 2023; pp. 4030–4039.
45. Zeng, C.; Kwong, S.; Ip, H. Dual Swin-transformer based mutual interactive network for RGB-D salient object detection. *Neurocomputing* **2023**, *559*, 126779. [CrossRef]
46. Bodla, N.; Singh, B.; Chellappa, R.; Davis, L. S. Soft-NMS–improving object detection with one line of code. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 5561–5569.
47. Zheng, Z.; Wang, P.; Liu, W.; Li, J.; Ye, R.; Ren, D. Distance-IoU loss: Faster and better learning for bounding box regression. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), New York, NY, USA, 7–12 February 2020; pp. 12993–13000.
48. Zheng, Z.; Wang, P.; Ren, D.; Liu, W.; Ye, R.; Hu, Q.; Zuo, W. Enhancing geometric factors in model learning and inference for object detection and instance segmentation. *IEEE Trans. Cybern.* **2021**, *52*, 8574–8586. [CrossRef] [PubMed]
49. Rezatofighi, H.; Tsoi, N.; Gwak, J.; Sadeghian, A.; Reid, I.; Savarese, S. Generalized intersection over union: A metric and a loss for bounding box regression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 658–666.
50. Neubeck, A.; Van G.L. Efficient non-maximum suppression. In Proceedings of the 18th International Conference on Pattern Recognition (ICPR), Hong Kong, China, 20–24 August 2006; pp. 850–855.
51. Song, Y.; Pan, Q.; Gao, L.; Zhang, B. Improved non-maximum suppression for object detection using harmony search algorithm. *Appl. Soft Comput.* **2019**, *81*, 105478. [CrossRef]
52. Chu, X.; Zheng, A.; Zhang, X.; Sun, J. Detection in crowded scenes: One proposal, multiple predictions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 12214–12223.
53. Liu, S.; Huang, D.; Wang, Y. Adaptive NMS: Refining pedestrian detection in a crowd. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 6459–6468.
54. Gao, P.; Zheng, M.; Wang, X.; Dai, J.; Li, H. Fast convergence of DETR with spatially modulated co-attention. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021; pp. 3621–3630.
55. Sun, P.; Zhang, R.; Jiang, Y.; Kong, T.; Xu, C.; Zhan, W.; Tomizuka, M.; Li, L.; Yuan, Z.; Wang, C. Sparse R-CNN: End-to-end object detection with learnable proposals. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 19–25 June 2021; pp. 14454–14463.
56. Shi, M.; Ouyang, P.; Yin, S.; Liu, L.; Wei, S. A fast and power-efficient hardware architecture for non-maximum suppression. *IEEE Trans. Circuits Syst. II Express Briefs* **2019**, *66*, 1870–1874. [CrossRef]
57. Fang, C.; Derbyshire, H.; Sun, W.; Yue, J.; Shi, H.; Liu, Y. A sort-less FPGA-based non-maximum suppression accelerator using multi-thread computing and binary max engine for object detection. In Proceedings of the IEEE Asian Solid-State Circuits Conference (A-SSCC), Busan, Republic of Korea, 7–10 November 2021; pp. 1–3.
58. Chen, C.; Zhang, T.; Yu, Z.; Raghuraman, A.; Udayan, S.; Lin, J.; Aly, M.M.S. Scalable hardware acceleration of non-maximum suppression. In Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 21–25 March 2022; pp. 96–99.
59. Choi, S.B.; Lee, S.S.; Park, J.; Jang, S.J. Standard greedy non maximum suppression optimization for efficient and high speed inference. In Proceedings of the IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), Gangwon, Republic of Korea, 1–3 November 2021; pp. 1–4.
60. Anupreetham, A.; Ibrahim, M.; Hall, M.; Boutros, A.; Kuzhively, A.; Mohanty, A.; Nurvitadhi, E.; Betz, V.; Cao, Y.; Seo, J. End-to-end FPGA-based object detection using pipelined CNN and non-maximum suppression. In Proceedings of the 31st International Conference on Field-Programmable Logic and Applications (FPL), Dresden, Germany, 30 August–3 September 2021; pp. 76–82.

61. Guo, Z.; Liu, K.; Liu, W.; Li, S. Efficient FPGA-based Accelerator for Post-Processing in Object Detection. In Proceedings of the International Conference on Field Programmable Technology (ICFPT), Yokohama, Japan, 11–14 December 2023; pp. 125–131.
62. Chen, Y.; Zhang, J.; Lv, D.; Yu, X.; He, G. O3 NMS: An Out-Of-Order-Based Low-Latency Accelerator for Non-Maximum Suppression. In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Monterey, CA, USA, 21–25 May 2023; pp. 1–5.
63. Sun, K.; Li, Z.; Zheng, Y.; Kuo, H.W.; Lee, K.P.; Tang, K.T. An area-efficient accelerator for non-maximum suppression. *IEEE Trans. Circuits Syst. II Express Briefs* **2023**, *70*, 2251–2255. [CrossRef]
64. Anupreetham, A.; Ibrahim, M.; Hall, M.; Boutros, A.; Kuzhively, A.; Mohanty, A.; Nurvitadhi, E.; Betz, V.; Cao, Y.; Seo, J. High Throughput FPGA-Based Object Detection via Algorithm-Hardware Co-Design. *ACM Trans. Reconfigurable Technol. Syst.* **2024**, *17*, 1–20. [CrossRef]
65. Batcher, K. E. Sorting networks and their applications. In Proceedings of the Spring Joint Computer Conference, Atlantic City, NJ, USA, April 30–May 2 1968; pp. 307–314.
66. Chen, Y.R.; Ho, C.C.; Chen, W.T.; Chen, P.Y. A Low-Cost Pipelined Architecture Based on a Hybrid Sorting Algorithm. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2023**, *71*, 717–730. [CrossRef]
67. Zhao, J.; Zeng, P.; Shen, G.; Chen, Q.; Guo, M. Hardware-software co-design enabling static and dynamic sparse attention mechanisms. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2024**, *43*, 2783–2796. [CrossRef]
68. Fang, C.; Sun, W.; Zhou, A.; Wang, Z. Efficient N: M Sparse DNN Training Using Algorithm, Architecture, and Dataflow Co-Design. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2023**, *43*, 506–519. [CrossRef]
69. Zhang, H.; Wu, W.; Ma, Y.; Wang, Z. Efficient hardware post processing of anchor-based object detection on FPGA. In Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Limassol, Cyprus, 6–8 July 2020; pp. 580–585.