ABSTRACT

Title of Thesis: DEEP LEARNING ENSEMBLES
FOR LIGHTWEIGHT OBJECT DETECTION

Alexander Mattingly
Master of Science, 2023

Thesis Directed by: Professor Shuvra S. Bhattacharyya
Department of Electrical and Computer Engineering

Object detection, the task of identifying and localizing important objects within an image frame, is a critical task in automation, surveillance, and safety applications. Further, developments in lightweight sensor technologies, improved small-scale computing, and the widespread accessibility of well-labeled data have enabled numerous applications for object detection on inexpensive or low-power hardware. Many applications, such as self-driving and unmanned aerial vehicles, must process sensor data as it arrives (in real-time) using onboard hardware (at-the-edge) in order to continually inform systems such as navigation. Additionally, detection must be often achieved on platforms with limited Size, Weight, and Power (SWaP) since advanced computer hardware may not be possible to place nearby the sensor. This presents a unique challenge: how can we best provide accurate real-time object detection on limited SWaP systems while maintaining low power and computational cost?

A widespread approach for detection is using deep-learning. An object detection network is trained on a labeled dataset of images containing known objects and their location. After training, the network may be used to infer on new data, providing both bounding boxes and class identifiers for each box. Popular single-shot detectors have been demonstrated to achieve real-time performance on some systems while having acceptable detection accuracy.

An ensemble is a system comprised of several detectors. In theory, detectors with architectural differences, ones trained on different data, or detectors given different augmented data at inference time will discover and detect different features of an image. Unifying the results of several different detectors has been demonstrated to improve the detection performance of the ensemble compared to the performance of any component network at the expense of additional computational cost. Further, systems using an ensemble of detectors have been shown to be good solutions to object detection problems in limited SWaP applications such as surveillance and search-and-rescue.

Unlike tasks such as classification, where the output of a network describes the entire input, object detection is concerned both with localization and classification of one or multiple objects in an image. Two different bounding boxes for partially occluded objects may overlap, or highly similar bounding boxes may describe the same object. As a result, unifying the results of object detector networks is far more difficult than unifying classifier networks. Current works typically accomplish this by applying strategies that iteratively combine bounding boxes by overlap. However, little comparative study has been done to determine the effectiveness of

these approaches.

This thesis builds on current methods of ensembling object detector networks using novel approaches to combine bounding boxes. We first introduce current methods for ensembling and a dataflow-based framework for efficient, scalable computation of ensembles of detectors. We then contribute a novel method for ensembling and implement a practical system for scalable detection using an elastic neural network.

DEEP LEARNING ENSEMBLES FOR LIGHTWEIGHT OBJECT DETECTION

by

Alexander Mattingly

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Masters of Science
2023

Advisory Committee:
Professor Shuvra S. Bhattacharyya, Chair/Advisor
Professor Reza Monfaredi
Professor Timothy Horiuchi

Acknowledgements

This thesis was made possible by the generous support of many people.

I firstly would like to thank my advisor, Professor Shuvra Bhattacharyya, for his excellent support and mentorship.

I would like to thank my committee members, Professor Reza Monfaredi and Professor Timothy Horiuchi, for providing valuable feedback and guidance during the process of completing this thesis.

I would also like to extend my gratitude to Dr. Eung-Joo Lee, Dr. Heesung Kwon, and Dr. Rachel Kinard for being excellent collaborators and mentors over the course of my master's degree.

Finally, I would like to thank my family, who have always supported me.

This research was supported in part by the U.S. Army Research Laboratory under grant no. W911NF2110258.

Table of Contents

1 Introduction **1**

2 Object Detection on Resource-Constrained Platforms Using a Configurable Ensemble of Detectors **6**
2.1 Introduction . . . 6
2.2 Related Work . . . 10
2.3 Background . . . 12
2.4 Method . . . 16
2.4.1 Synchronous Dataflow Ensemble Network Design . . . 17
2.4.2 Selecting Unit . . . 19
2.4.3 Reconfigurable EOD System . . . 19
2.4.4 Weight-Focused Selection . . . 21
2.4.5 Combining Unit Strategy . . . 22
2.5 Experiments . . . 24
2.5.1 Maintaining FPS Constraints . . . 24
2.5.2 Exploiting Parallelism in the Dataflow Model . . . 26
2.6 Conclusion . . . 27

3 Scalable Real-time Object Detection Using an Ensemble formed by an Elastic Single Shot Detector Network **28**
3.1 Introduction . . . 28
3.2 Related Works . . . 31
3.3 Methodology . . . 33
3.3.1 Graph Clustering . . . 33
3.3.2 Proposed Design . . . 35
3.4 Experiments . . . 35
3.4.1 Training/Testing . . . 36
3.4.2 Evaluation . . . 37
3.4.3 Tuning of System Parameters . . . 37
3.4.4 Evaluation of Ensemble Scalability . . . 38
3.4.5 Discussion of Results . . . 38
3.5 Conclusion . . . 40

4 Conclusions and Future Work **43**

Bibliography **47**

Chapter 1

Introduction

In computer vision, object detection refers to the task of identifying and localizing important objects in an image or sequence of images. These identified objects are then described by bounding boxes and class identifiers, representing the minimum enclosing rectangular regions containing the objects and the types of the objects, respectively. This paradigm is highly versatile; automatic identification of targets in an image is desired for numerous use-cases in medical imaging, autonomy, security, and human-computer interaction, to name only a few.

Deep learning has become a popular method of approaching the problem of object detection. Deep learning encompasses a wide range of tools that involve a computational model, represented by a hierarchical construction of filters and operation, that allows for the model to be trained on known data. This way, the model may learn to identify features that indicate the presence of important objects over time and make inferences on new data.

Many applications of computer vision, especially those related to autonomous or unmanned aerial vehicles, surveillance, and security, require object detection calculated on-the-fly at a rate faster than or equal to the rate at which new data arrives. Such systems are called 'real-time,' and will typically involve trade-offs in detection accuracy to accommodate for this new requirement. A wide range of deep learn-

ing object detectors have been developed advertising real-time capability, indicating inference may occur at least as fast as new sensor data becomes available. In this work, the term 'lightweight' will refer to such systems, whose performance in both accuracy and speed are considered for real-time applications. To further complicate the problem of lightweight object detection, mobile real-time object detectors may be employed on systems with limited computational resources. Small UAVs and self-driving vehicles will typically only have a limited embedded processor and strict requirements on power consumption. As a result, object detection solutions must be designed from the ground-up with power and speed at the forefront, potentially to the detriment of detection accuracy.

Advancements in computer architecture to enable lightweight computation on limited platforms have allowed for increasingly powerful computer vision, but there still remains a performance gap between real-time detectors and their offline counterparts. However, detection challenges for typical UAV applications are extremely difficult, with critical objects occupying tiny fractions of an entire image supplied by a sensor. An example of an image from a UAV may be found in Figure 1.1.

A useful technique that can be applied to potentially improve object detection accuracy is ensembling. In an ensemble, multiple sources of object detections operate on a single image. Results are then combined so that the ensemble performs as a single source of detection. The ensemble may either involve multiple models, the same image with adjustments such as flipping or color correction, or a combination of both. Use of ensemble-of-detector (EOD) systems has been demonstrated to provide improved detection at the expense of additional computational cost. While



Figure 1.1: Image from the Vehicle Aerial Imaging from Drone dataset [1], a dataset of images captured by a UAV for traffic detection. From the perspective of the camera, vehicles are small and numerous, presenting a challenge for lightweight object detectors.

this may seem detrimental to the goal of lightweight object detection, numerous systems have been demonstrated that combine low-cost object detector networks to maintain real-time functionality while improving accuracy. However, questions still exist regarding best practices to ensembling in the context of real-time applications:

- How can we best utilize limited computational resources to enable EOD systems to be deployed to edge devices? Are there strategic tradeoffs that can be made across inference time, accuracy, and power requirements in such a system?

- What techniques can we use to combine outputs of object detectors?

To address the first questions, we investigate techniques for ensemble-of-detector

systems within the context of lightweight object detection. We first introduce current methods for ensembling and introduce a dataflow-based framework for efficient, scalable computation of ensembles of detectors. We then implement a practical system for scalable detection using an elastic neural network, identifying its strengths and limitations.

There are many challenges involved in combining outputs of detectors, our second question. Different detectors may pick up on different features, producing detections with slightly differing coordinates as well as false-positive and false-negative detections. A visualization of one image annotated by many detectors may be found in Figure 1.2, providing an intuition for this problem. To determine best practices for combining outputs of object detectors, we propose a novel method for ensembling, using a graph-based formulation, which we call a *similarity graph*, of nearby bounding boxes. Once this representation of bounding boxes is formed, we apply a graph clustering algorithm to group boxes into likely candidates. This method is compared with current established approaches.

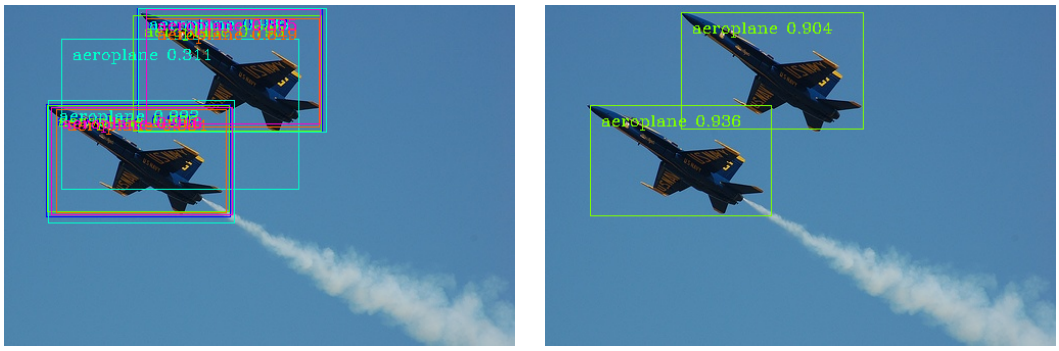


Figure 1.2: An image from the Pascal VOC dataset [2] annotated by five different detectors in different colors (left) and using combined detections (right). All detections label bounding boxes correctly as 'aeroplane,' however, one detector (teal) incorrectly identifies a third aeroplane. How should a combining algorithm handle this extra detection?

The remainder of this thesis will proceed with the following. Chapter 2 introduces a dataflow-based framework for scalable object detection that uses run-time information to inform the configuration of active detectors at run-time. Chapter 3 introduces a novel framework for object detection using an ensemble based on an elastic neural network. This *elastic-ensemble* design allows for easily-scalable object detection by allowing a single network to achieve a multitude of cost-accuracy tradeoffs. Finally, Chapter 4 will conclude this thesis and discuss results and future directions.

Chapter 2

Object Detection on Resource-Constrained Platforms Using a Configurable Ensemble of Detectors

In this chapter, we propose a new framework for flexible real-time object detection using an ensemble of toggle-able object detectors. Material in this chapter was published in partial, preliminary form in [3].

2.1 Introduction

Real-time object detection is of increasing interest on edge cloud platforms in both commercial and defense applications (e.g., see [4]). Because edge cloud platforms typically are much leaner compared to conventional cloud computing environments, edge-cloud-based deployment of object detection involves challenges of requiring accurate, real-time knowledge extraction from high-volume sensor data under stringent resource constraints. In particular, many edge cloud applications must operate within constraints of limited computational resources and power consumption. Methods based on deep neural networks (DNN) are of increasing interest for this class of applications due to their demonstrated effectiveness on many types of sensor data, and many types of knowledge extraction problems. However, DNN-based methods often face limitations in accuracy when confronted with highly unpredictable and varied operational scenarios in which defense-oriented edge-cloud

platforms and edge-cloud platforms in other mission-critical application areas must operate. Deployment of neural networks for such applications is further complicated by the heterogeneous capabilities of alternative DNN solutions to the same knowledge extraction problems. This diversity in capabilities presents potential opportunities to enhance object detection accuracy, but also leads to more complex design spaces that are challenging to navigate.

In this chapter, we are specifically interested in object detection from camera images — for example, the detection of people or vehicles in a geographical area from a UAV that is used to monitor the area. In the experiments presented in the chapter, we focus specifically on vehicle detection; however, it is anticipated that the methods developed in the chapter can be adapted readily to other types of object detection problems. In the remainder of the chapter, when we refer to object detection, we specifically mean image-based object detection, unless otherwise specified.

In addition to accuracy considerations, real-time performance is also an important concern in the design and implementation of object detection systems for edge-cloud platforms. Several works have been proposed for practical solutions to the problem of real-time deep-learning [5–8]. However, conventional approaches to real-time object detection have significant limitations when deployed in unpredictable environments.

To help address the limitations and complications described above, object detection systems can be designed using collections of independently-trained detectors whose inference results are aggregated at run-time to arrive at an overall detection result for each input instance (e.g., see [9]). In such systems, groups of detectors,

called ensembles, operate independently on each input image. By combining the detection results of an ensemble, accuracy may be improved at the cost of increased processing requirements, which may translate to degradation in metrics such as latency, throughput and energy consumption.

In addition to deciding on which detectors to use, the design of an ensemble object detection (EOD) system involves an important subsystem called a combining unit. Intuitively, the purpose of the combining unit is to transform the detection results of the different detectors in the ensemble into a single overall detection result for the system.

The objective of this chapter is to develop an EOD framework that delivers both accuracy and efficiency, and is capable of being adapted based on specific resource constraints and optimization objectives associated with a given edge-cloud-based deployment. To achieve this, a novel design for a dynamically reconfigurable EOD system is presented. An important aspect of the reconfigurable EOD design is the introduction of a subsystem called a *selecting unit* whose purpose is to determine which of the available detectors in an ensemble to use during a given period of time. By disabling some of the detectors, the selecting unit is able to dynamically trade-off processing accuracy and computational efficiency. We refer to the set of detectors that is in use (active) during a given period of time T as the *active ensemble subset* for T.

The reconfigurability of the proposed system is in terms of high-level system parameters — most notably, the active ensemble subset; dynamic configuration (or setting) of these EOD parameters is what we mean by "reconfigurability" in this

chapter. Such a reconfigurable EOD framework can be implemented in software, reconfigurable hardware (e.g., using field programmable gate array technology) or a co-designed hardware/software solution. In this chapter, we focus on software-based implementations. Investigation of reconfigurable-hardware and hardware/software implementations is an interesting direction for further study.

Our proposed system employs dataflow modeling techniques to both parallelize various object-detection-related tasks and allow for the system to adjust its active ensemble configuration at run-time through design of a subsystem for configuration management and scheduling. In the context of this work, dataflow can be viewed as a methodology for modeling and optimizing embedded signal and information processing systems using directed graphs [10]. By *scheduling*, we mean coordinating the execution of different functional components within a dataflow-based system model across available processing resources. We investigate the use of different scheduling patterns to optimize key performance aspects, such as frame rate and power consumption. We then use these patterns in our experiments with two relevant datasets — the Unmanned Aerial Vehicle Benchmark [11] and the KITTI Vision Benchmark Suite [12] — to demonstrate the advantages of our proposed EOD framework over conventional ensemble network designs in realistic scenarios.

The key contributions of this chapter are summarized as follows:

- We develop a framework for highly parallel, scalable, and efficiently-reconfigurable EOD system design using dataflow techniques. The framework is created to aid in object detection tasks that are deployed on resource-constrained edge-cloud

platforms. It is anticipated the framework can readily be adapted to other resource-constrained and real-time commercial- and defense-oriented sensing applications.

- We describe and discuss behavior patterns for active ensemble subsets and their possible use cases. These patterns are presented to optimize run-time performance metrics in limited-resource systems.

2.2 Related Work

As described in Section 2.1, an EOD system is an object detection system that is constructed by combining detection results from multiple detector networks. With an ensemble, both the accuracy and diversity of detectable object classes can be improved when compared to monolithic (non-ensemble) object detection methods. However, unless the complexity of individual detectors in an ensemble is reduced at least in proportion to the number of detectors in the ensemble (ensemble size), an ensemble is in general more computationally intensive when compared to single-detector systems.

A wide variety of object detection networks may be used in an ensemble. For example, one-stage detection networks (e.g., YOLO variants [13–15] and SSD [16]) are examples of networks that would fit naturally in an ensemble, due to their relatively low computational cost. one-stage detectors perform object detection by predicting bounding boxes and probabilities simultaneously without the use of a region proposal network (RPN). This is in contrast to two-stage detectors such as

Fast and Faster R-CNN [17, 18], which must first propose regions in an initial stage using an RPN and then classify the proposed bounding boxes in a second stage. These two methods of object detection simultaneously remain state-of-the-art; the two-stage detectors boast higher accuracy when compared to one-stage detectors but at the cost of lower speed.

Ensembles of one-stage and two-stage detectors have the potential to alleviate limitations that are unique to each individual detector. For example, combinations of Fast R-CNN and YOLO detectors have been tested, improving accuracy on the VOC 2007 dataset [14]. It is noted in [14] with this ensemble that the boost in accuracy is a result of the differing designs in these networks. This specific example uses Fast R-CNN as an initial stage that provides a high detection rate, and then uses the YOLO network as a post-processing stage to filter out background detections (false positives).

A major consideration for any ensemble of networks is the strategy employed to combine the results of individual detectors. As described in Section 2.1, we refer to the subsystem of an EOD system that performs this combination as the combining unit for the system. For a given input image to an EOD system, different object detection networks in general return different sets of bounding boxes for detected objects in the image. If there are n different detectors in an ensemble, the detectors will collectively produce n sets $\sigma_1, \sigma_2, \ldots, \sigma_n$ of bounding boxes, where each σ_i represents the set of objects detected by the i detector in the ensemble. The combining unit takes these n sets as input, and produces a single set of bounding boxes, which represents the overall detection results of the ensemble.

Combining strategies for detectors with expected unequal performance must be considered. Studies have been conducted that approach this problem by maintaining a list of estimated relative accuracy levels for each detector, and weighting the contributions of the individual detectors in favor of the more accurate detectors. This strategy is especially relevant to our design since our proposed system must make decisions to balance expected detection accuracy, speed, and power when combining results from detectors and selectively activating detectors [19]. Similarly, strategies have been proposed based on voting. Decisions are made depending on whether all detectors simultaneously report an object at approximately the same location (*unanimous voting*), if more than half report an object (*consensus voting*), or if any single detector reports an object (*affirmative voting*) [20].

2.3 Background

Our approach to EOD system design involves the application of dataflow models of computation, particularly signal-processing oriented dataflow modeling. In this section we provide a brief review of background on this form of dataflow modeling and design; for more details, we refer the reader to [10,21]. Dataflow modeling is commonly used in design and implementation of signal and information processing systems to provide a formal, graph-theoretic framework for working with high-level signal flow behaviors. As one of numerous example application areas, dataflow techniques have been applied to the design of novel systems for hyperspectral video processing [22].

The behavior of a dataflow-based application model is described as a directed graph $G = (V, E)$ where each vertex $v \in V$ describes a single computational element (*actor*) that performs operations on discrete units of data (*tokens*). Each directed edge $e \in E$ describes a first-in-first-out (FIFO) communication channel between two actors. Actors *fire* to perform a discrete unit of computation, which in general both *consumes* tokens on that actor's incoming edges and *produces* tokens on its outgoing edges. Actor firing is guarded by a set of *firing rules* that determine the amount of tokens required at each of its input edges, which may change between firings. Once an actor is able to fire (*enabled*) by its firing rules, it may fire at any time [10]. Actors can be of arbitary complexity and can be implemented in an programming language provided that they operate in terms of firing rules and production/consumption of tokens, as described above. The actors in a dataflow graph are typically fired repeatedly as long as input data is available for the graph to process or the graph is disabled or turned off through external control. A wide variety of methods for analyzing and optimizing dataflow-based hardware and software implementations can be found in the literature; for an extensive review, see [21].

A simple example of a dataflow graph is shown in Fig. 2.1. This graph operates on three infinite streams of integer tokens $A = \{a_1, a_2, \ldots\}$, $B = \{b_1, b_2, \ldots\}$, and $C = \{c_1, c_2, \ldots\}$ to produce a stream of output integer tokens $D = \{d_1, d_2, \ldots\}$, where each $d_k = (a_k + b_k) \times c_k, k \in \mathbb{N}$. Actors A, B and C are referred to as *source actors* since they have no inputs. These actors represent interfaces for injecting input values from the input stream into the dataflow graph for processing. For example, the actors may represent interfaces for retrieving data from three different sensors.

On each firing, each source actor produces a single token, which encapsulates the value of the next signal sample in the associated input stream. Similarly, actor D is a *sink* actor, which represents an interface for the dataflow graph to produce its outputs — e.g., an interface to a digital-to-analog converter.

Apart from the source actors and sink actors, the dataflow graph example contains an actor that performs addition and another that performs multiplication, as shown in Fig. 2.1. In the kinds of dataflow graphs that were are interested in, most of the computation is described in terms of actors that are significantly more complex than single addition and multiplication operations; this example incorporates primitive operations as computational actors for simplicity.

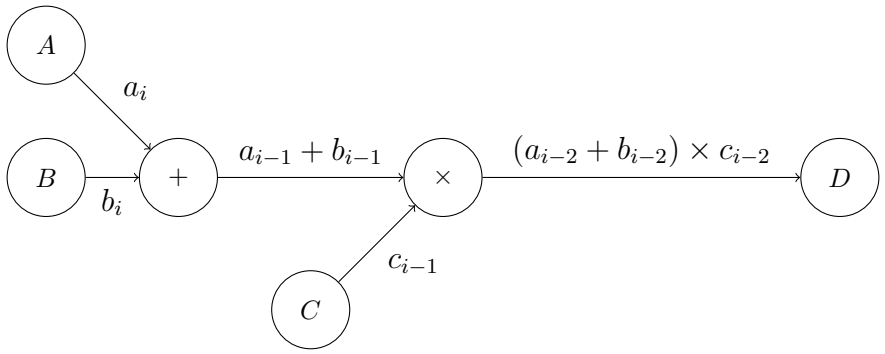


Figure 2.1: Dataflow Graph Operating on Stream of Tokens with Token Indexes Annotated at Iteration i

Enabled actors may fire independently while maintaining a consistent global order for the data being processed, so long as each actor's internal mechanisms behave in a deterministic manner. For example, dataflow actors may not share variables between each other, since the order of access to shared memory may change the behavior of the actors [10].

To further allow for efficient data transfer, a division between *passive* and

active elements in a dataflow model may be formulated, turning the graph into a passive-active dataflow model [23]. In these models, computational elements are active, while communication elements are passive. This distinction is important when considering communication channels that are not strictly FIFO buffers. For example, an actor may be created whose purpose is to read in tokens from an input FIFO buffer and copy the token to two different output FIFO buffers. This actor does not perform any processing, and only needs to implement the read-write interface typical of a communication channel. As a result, this actor can be implemented as a so-called *passive block*, which has read-write interfaces to its ports rather than having enable/invoke (firing-oriented) interfaces to the component as a whole, as actors have. Such a transformation (from active to passive form) eliminates overhead required for the scheduler to manage firings of the component and can greatly improve the efficiency of data communication between actors. In our EOD system design, we implement two passive blocks, which manage the routing of images to detectors and provide a simple interface to enable and disable detectors.

There exist many types of dataflow graphs depending on assumptions that are made on how the production and consumption rates of the actors can be characterized. One of the most widely used forms of dataflow is *synchronous dataflow* (*SDF*), where each actor consumes and produces a constant number of tokens on all of its edges on each firing [24]. In the prototype EOD system that we present and experiment with in this chapter, we apply a dataflow-based design model that is constructed in terms of SDF actors and passive blocks, where most of the passive blocks operate as dataflow edges, while one of the passive blocks is a more general

passive block design based on the passive-active flowgraph (PAFG) model presented in [23]. For detailed background on the SDF and PAFG modeling techniques that are applied in our EOD system design, we refer the reader to [23, 24].

Applying dataflow methods in our EOD system allows us to create a highly paralellizable, rapidly configurable design. Since a scheduler for the system may fire any actor that is enabled at any time, we can efficiently configure when and how each computation is allocated and executed. We implement a hybrid dataflow/Von-Neumann design [25], which features relatively complex actors that are programmed in an imperative language (C++). This allows us to develop using familiar imperative programming principles while applying the benefits of the dataflow model at higher levels of abstraction in the design.

2.4 Method

In this section, we present the design of our proposed EOD system for scalable, reconfigurable image-based object detection. The system provides a general framework for integrating an arbitrary collection of specialized detectors into a reconfigurable ensemble. While we demonstrate the system in our experiments using a specific set of detectors, the overall framework is not specific to these detectors and can readily be retargeted to alternative numbers and collections of detectors.

2.4.1 Synchronous Dataflow Ensemble Network Design

In this study, we first implement an SDF graph describing the high-level behavior of the ensemble. Given a set of n detectors $D = \{d_1, d_2, \ldots, d_n\}$, we create a dataflow graph as shown in Fig. 2.2.

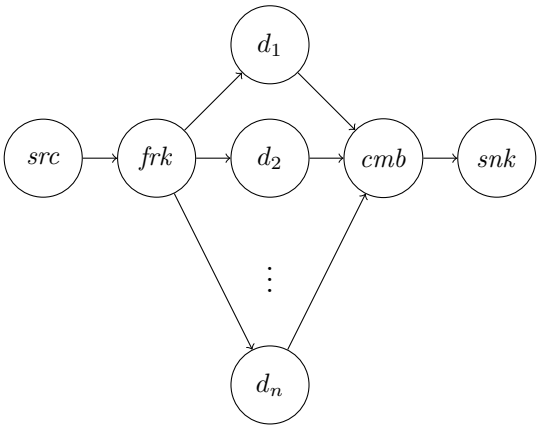


Figure 2.2: SDF model for the proposed EOD system.

Images are injected into the system for processing by the *SRC* actor; each firing of this actor produces a pointer to a block in memory that stores the next image to be processed by the system. The input images are forwarded to the n detectors using a *fork* actor (also called a broadcast actor), which produces a copy of each consumed input token (image pointer) on each of its output edges. On each firing, a given detector d_i consumes a pointer to an image, and produces a pointer to a list of bounding boxes of detected objects in the image. The results from each detector are merged using the *Combine* actor, which produces an overall detection result (list of bounding boxes) for the given input image based on the results produced by the individual detectors.

Like the individual detectors, the *Combine* actor produces a pointer to a list of

bounding boxes that is stored in memory. The *SNK* actor consumes the produced lists of bounding boxes and stores the bounding box information in a text file for further analysis. A text file is used for the output in our simulation experiments, as presented in Section 2.5; for a deployed system, the *SNK* actor could be replaced by an interface to a subsystem that interprets the bounding boxes in real-time to trigger actions, such as communication events or further image processing on the detection results, based on the observed detection results.

To enable flexible use of active ensemble subsets, we incorporate the concept of `null` inputs and outputs of the detectors. A special token value is used to represent an `null` input or output. When a detector fires and consumes an `null` input value, it bypasses its detection computations and immediately produces an `null` output value. Thus, `null` values can be passed to the input of a detector to disable it — that is, to remove it from the active ensemble subset (see Section 2.1). The combining unit (*Combine* actor) simply discards inputs from which it consumes `null` values, and performs its combination operation only on the non-null inputs that it consumes in a given firing.

The first-version design shown in Fig. 2.2 does not incorporate any disabling of detectors; this design uses all n detectors in its active ensemble subset. In Section 2.4.3, we present a second-version design that incorporates mechanisms that allow a selecting unit to disable selected detectors by sending `null` token values to their inputs.

2.4.2 Selecting Unit

The selecting unit, which was introduced briefly in Section 2.1, manages which detectors are enabled by generating a stream of vectors $\alpha_1, \alpha_2, \ldots$, where each α_i specifies the active ensemble subset that is to be used for processing the ith input image to the EOD system. We refer to each α_i as the *active-detector vector* for the ith input image. More specifically, each α_i is an n-dimensional vector of Boolean values such that $\alpha_i[j] = \mathit{true}$ if detector d_j is to be included in the active ensemble subset for processing the ith image.

In this chapter, we provide a general interface between our proposed reconfigurable EOD system and a selecting unit that can be connected to the system to generate active-detector vectors for dynamically configuring the active ensemble subset. We also experiment with one specific approach to selecting unit design. The applied approach involves assigning weights to the different detectors in relation to their computational costs and contribution to overall detection accuracy. More details on this selection approach are discussed in Section 2.4.3; the approach is used to demonstrate and study the reconfiguration capabilities of the proposed EOD system. Further investigation into the optimized design of selecting units is a useful direction for future work that builds naturally on the developments of this chapter.

2.4.3 Reconfigurable EOD System

In this section, we introduce a second-version design (see Fig. 2.3) of our EOSD system, which integrates functionality for processing active-detector vectors.

We also discuss the selecting unit, which is used to generate the active-detector vectors. To help reduce scheduler overhead and facilitate rapidly activating and disabling detectors, a passive block (see Section 2.3) is created to interpret active-detector vectors and broadcast incoming image frames (to active detectors) in an integrated manner. This passive block, which we call a *passive image fork* (*PIF*), consumes both an image-pointer token and an active-detector vector ν. The image token is then broadcast to all detectors that are marked as being active by the vector ν. On each output port of the *PIF* block that is connected to a disabled detector (as specified by ν), the *PIF* block outputs an `null` token. Since the *PIF* block is passive, it does not produce its output tokens as actors do; instead, its output tokens become available to consuming actors when they are read from the associated output ports of the *PIF* block. The modified flowgraph is illustrated in Fig. 2.3. The dashed outline around the *PIF* module in the figure highlights its design as a passive block.

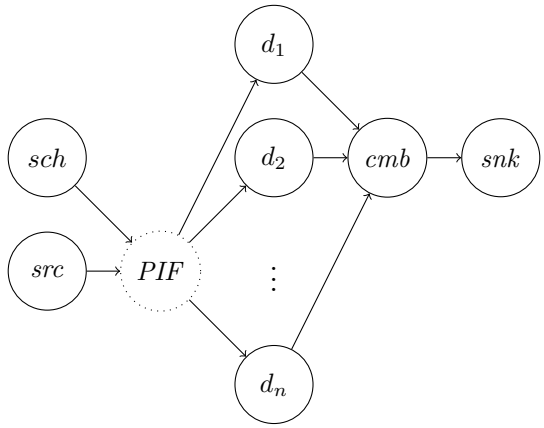


Figure 2.3: Reconfigurable EOD system.

Note that technically, due to use of PAFG modeling techniques in the *PIF*

module, the graph shown in Fig. 2.3 is not a dataflow graph. However, because PAFG models can be related precisely to functionally-equivalent dataflow models (see [23]), we ignore this distinction, and refer to the model illustrated in Fig. 2.3 interchangeably as a flowgraph and dataflow graph.

As described in Section 2.4.2, the design of the selecting unit is an important aspect in our proposed framework for EOD system design. In our prototype EOD system, we incorporate a specific algorithm as the basis for selecting unit design, which we refer to as *weight-focused selection*. However, the overall EOD framework that we propose is not dependent on weight-focused selection, and can incorporate a wide variety of alternative algorithms within the selecting unit. Exploring the design space for selecting unit design is an interesting direction for future research. Next, we describe details on the weight-focused selection approach used in our current prototype.

2.4.4 Weight-Focused Selection

This approach involves use of performance estimates for the detectors — both in terms of computational cost and contribution to overall ensemble accuracy. These estimates are derived off-line and stored as part of the state of the selecting unit to aid in its decision-making. We refer to these estimates as the computational weight and accuracy weight, respectively. Each detector in the ensemble is associated with real-valued weights of each type, where higher weights for the computational weight correspond to more computationally intensive detectors, and higher weights

for the accuracy weight indicate that the detector has a tendency to contribute more towards accurate detection results by the ensemble. Different methods are available for producing these estimates; detailed discussion of such methods is beyond the scope of this chapter.

Given a subset $S_d \subseteq D$ of detectors in the ensemble, we refer to $A(S_d)$ as the sum of the accuracy weights of the elements of S_d, and similarly, we denote by $C(S_d)$ the sum of the computational weights of the elements in S_d.

At a given point during system operation, the weight-focused selection approach aims to find a subset S_d of detectors that maximizes $A(S_d)$ subject to a dynamically variable constraint Z_c on $C(S_d)$ ($C(S_d) \leq Z_c$). Note that for an optimal solution, this strategy would need to implement a solution to the 0-1 knapsack problem, which is NP-complete. Moreover, due to the dynamic variability of the constraint Z_c, this solution would in general need to be applied at run-time or results would need to be stored off-line and accessed at run-time using a lookup table. For relatively small sets of detectors, the processing time for an optimal solution may be acceptable. However, for ensembles with relatively large numbers of detectors an approximation algorithm or other type of heuristic approach may be more appropriate [26].

2.4.5 Combining Unit Strategy

To effectively combine the results from the different detectors in D, our EOD framework applies a general combining algorithm based on the intersection-over-

union metric to determine if bounding boxes should be grouped. This algorithm is inspired by the weighted combining algorithm of Körez et al. [19] with the key difference being that our combining algorithm additionally tracks contribution metrics for individual bounding box groups. These contribution metrics are applied for use in voting- or contribution-based filtering. This approach provides an efficient means for discarding bounding boxes that are found only by relatively small numbers of detectors (e.g., by only one detector) or only by detector-subsets with low sum-of-weights.

We apply this algorithm to a combined list of bounding boxes from each detector after non-maximum suppression is applied to each individual list. Intuitively, non-maximum suppression in this context involves the derivation of a single bounding box out of multiple overlapping bounding boxes (e.g., see [27]). We refer to this process of merging overlapping bounding boxes as *grouping* of the bounding boxes; each grouping effectively replaces multiple bounding boxes with a single one. All bounding boxes in a group have a relatively high intersection-over-union value with all other boxes in that group.

Our combining algorithm allows each detector's overall contribution to the final results to be weighted according to its expected performance while avoiding the counting of multiple bounding boxes generated by the same detector for the same object. Since we save both the values of the total weight and whether or not a specific detector was involved in each grouping of bounding boxes, we are able to apply additional processing to remove both low-weight and low-contribution groups of bounding boxes.

The approach to combining unit design used in this chapter is not the only way to apply the combining unit within our proposed EOD framework. Indeed the EOD framework can be flexibly adapted to apply other combining unit algorithms. For example, we can easily adapt the design to filter results that are not met by a majority of detectors (consensus voting) or allow any detector to contribute a valid result (affirmative voting).

2.5 Experiments

In this section, we present experiments to demonstrate the effectiveness of the proposed framework for EOD system design. Our evaluation is carried out using the metric of speed (the average time required to process an image frame). We evaluate our system on a desktop computer containing an Intel i-7 3770k processor and an Nvidia 10-Series graphics processing unit (GPU). This platform has modest computational capabilities and is representative of the level of resource availability in many edge-cloud environments. In our experiments, we use an ensemble of YOLOv3-Tiny [15], YOLOv3 [13], and Faster R-CNN [17] detectors.

2.5.1 Maintaining FPS Constraints

To simulate a situation where the computational requirements of system are changing, we limit the availability of processing resources mid-execution. This situation mimics the realistic scenario where a system must dedicate additional resources to a different task that only occurs intermittently, such as wireless communication.

This scenario is a relevant use-case for the weight-focused selection approach described in Section 2.4. The selecting unit may optimize the expected combined detection accuracy using known weights and costs, subject to the additional constraint of maintaining a defined minimum FPS. When resources are limited or returned (for using by the EOD system), the scheduler is expected to disable and enable detectors, respectively, to keep the cost under the new resource limit while maintaining the objective of maximizing the total accuracy weight of the enabled detectors.

For this experiment, we use a 400-frame portion of the Unmanned Aerial Vehicle Benchmark dataset. Between frame indices 125 and 275, the CPU sleeps for 23 ms between each frame to simulate the time where the system is performing other tasks. We measure performance in speed compared to the ensemble without dynamic adjustments. Real-time performance and detection accuracy are also compared with the performance levels of the individual detectors working in isolation.

Fig. 2.4 shows graphs of the frame time (average time to process an image frame) of the ensemble system, both with and without dynamic adjustment using the selecting unit. The figure also shows the performance of the three component detectors. At frame index 125, additional CPU wait time is added, increasing the frame time for all configurations with the exception of the dynamic configuration, which disables selected detectors to maintain its target frame rate.

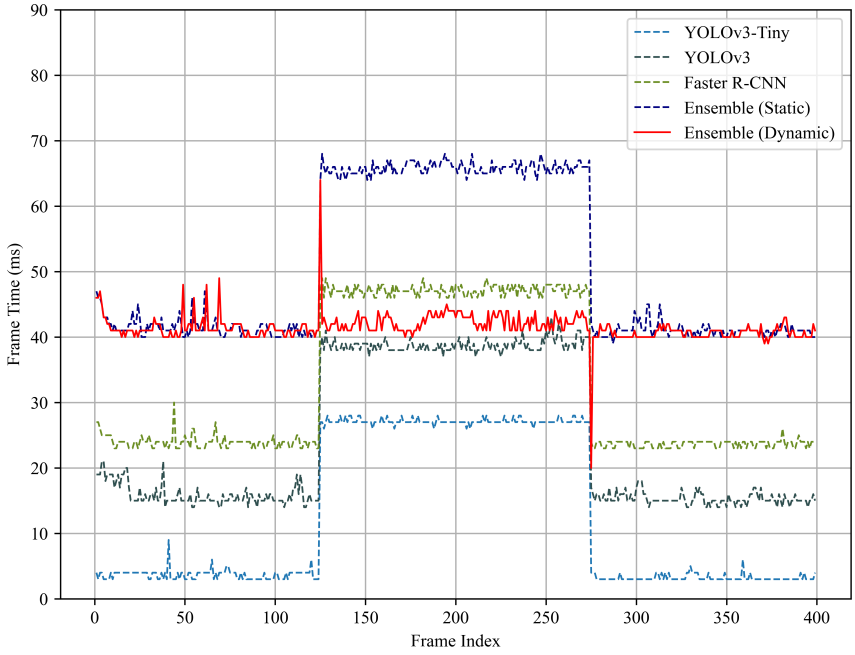


Figure 2.4: Frame time metric for EOD system in comparison with component networks.

When faced with a temporary scarcity of computational resources, our system is able to efficiently compensate for the new constraints. This provides for continued real-time performance while still allowing for the ensemble to return to its complete state (all detectors active) when sufficient computational resources become available.

2.5.2 Exploiting Parallelism in the Dataflow Model

Our EOD framework facilitates the exploitation of parallelism in the system based on the capabilities of the processing platform. To demonstrate this capability, we compare our prototype EOD system to a modified version of the EOD system that executes its computation with no parallelism between the detectors or between

frames. Unlike in the previous examples, this "serial" version fully processes a single frame before beginning any operations on the next frame. For this experiment, all detectors are enabled in both implementations. Since the two detection systems produce identical output, only the speed is considered in our comparison. We test our implementations using 18 trials of 100 images each from the KITTI Vision Benchmark Suite [12]. The parallel system performed faster than its serial counterpart to a significant degree while using the desktop computer. Table 2.1 displays our measured results as well as the improvement. Processing-speed performance is reported in terms of image frames per second (fps).

Parallel Dataflow Performance (fps)	Serial Performance (fps)
20.95	18.30

Table 2.1: Comparison of ensemble network performance between the dataflow-based parallel implementation and a conventional (non-dataflow) sequential implementation.

2.6 Conclusion

In our work, we present a novel methodology for highly parallel ensemble detection network design using dataflow modelling tools. We focus on creation of a system capable of accurate computer vision from embedded hardware for high-speed, low-power applications the edge. A scheduler algorithm is proposed that demonstrates improvements in computational resource utilization. In this study, we evaluate our proposed design by using the Unmanned Aerial Vehicle Benchmark and KITTI Vision Benchmark Suite.

Chapter 3

Scalable Real-time Object Detection Using an Ensemble formed by an Elastic Single Shot Detector Network

3.1 Introduction

Creating robust and lightweight sensor technologies has allowed various defense and civilian use cases, and Unmanned Aerial Vehicles (UAVs) remain highly useful tools in surveillance. These lightweight vehicles can be rapidly deployed to gather information efficiently in diverse scenarios. One critical component for UAV-based surveillance is object detection, the task of automatically recognizing and localizing important features of an image. To achieve real-time functionality, processing for object detection must often be conducted at the edge on the UAV itself. However, UAV-mounted computing systems must be designed with limited resources in mind to ensure optimal performance [28]. Various studies have explored embedded computer vision and UAV applications to tackle the issue of limited resources (e.g., see [8, 22, 29]).

Typical applications of UAV-based object detection include search-and-rescue [30], terrain mapping [31], crowd detection [32], and vehicle surveillance [33]. These tasks require the UAV to capture images of an environment from above, which can reduce detection accuracy [34, 35]. Furthermore, this introduces another challenge, as ob-

jects may be both small and numerous. In the UA-DETRAC [36] and UAVDT [11] datasets, the average number of objects per frame is 8.64 and 10.52 in the two datasets. Additionally, at altitudes above 70m, small objects in the UAVDT dataset may occupy as little as 0.005% of the total image pixels, making localization significantly more challenging.

Deep learning has become a popular method of approaching the object detection problem. Using lightweight single-shot object detectors, such as SSD [16], allows for real-time performance even in edge computing applications. However, accuracy is reduced compared to more complicated detectors. To achieve better accuracy performance in object detection tasks, outputs from multiple object detection models may be combined together. Using this method, the final system becomes more robust to individual features, noise, or artifacts that would otherwise cause failures in a single model. Such a system employing multiple models is considered an *ensemble* of object detectors. Extensive research has been conducted to create ensemble-based object detection systems [18, 37–39].

This work builds on the Elastic Neural Networks framework [40]. This framework introduces the concept of *early exits* and *post-exit subnetworks*. An early exit in a network is an alternative path for intermediate results to leave the network. Early exits are placed in specific points in a network and matched with a post-exit subnetwork, an extra network that produces results from features available at an early exit [40]. For our purposes with object detection, we use the Elasticized Single shot detector Network for Object detection (ESNO) [4]. The network structure as used in our system, showing the early exits and post-exit subnetworks, can be found

in Figure 3.1.

In this chapter, we propose a novel object detection system for UAV-based surveillance of automobile traffic using an ensemble of multiple early exits in the ESNO. By combining the output from multiple early exits in the network, we improve object detection accuracy while introducing minimal overhead compared to the original system — that is, the system that consists only of a single output, without any use of intermediate outputs or their ensembles . Further, our system achieves scalability by allowing for a subset of early exits to be used to provide trade-offs in inference time and detection accuracy. In this chapter, we introduce relevant techniques for ensembles of detectors, and specifically for the construction of ensembles using subsets of early exits from an elastic neural network. We then demonstrate the use of our proposed *ensemble-elastic* system using the Vehicle Aerial Imaging from Drone (VAID) [1] dataset.

Our key contributions in this chapter are listed below:

- We introduce a novel framework for scalable object detection using an elastic neural network to enable real-time processing.
- We develop and iterate upon methods for ensembling multiple distinct object detection models to enhance the robustness of detection accuracy. These methods are general, and not restricted to ensembles that are constructed from early exits of a given neural network model.
- We evaluate our proposed system and ensembling techniques using a public dataset representative of a typical application of embedded computer vision

in a real system.

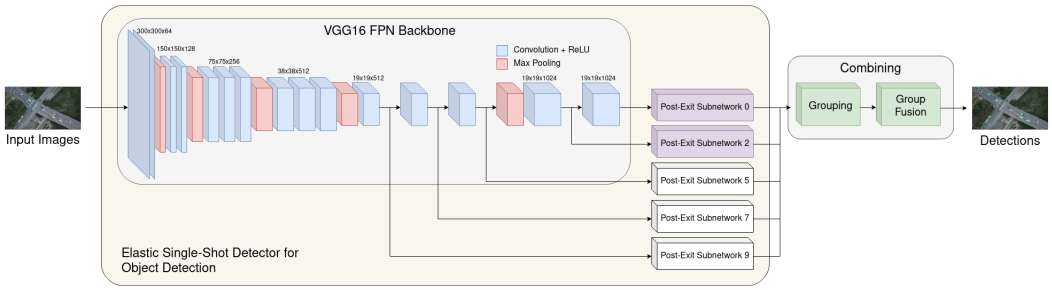


Figure 3.1: Diagram of the proposed system: Post-exit subnetworks 0 and 2 are active in this configuration.

3.2 Related Works

The Elastic Neural Networks concept is intended to maximize scalability while minimizing extra work at training time. Training for the original elasticized network only occurs once. After this, training of the post-exit subnetworks occurs individually with the original network frozen. The result is many separate functional networks with tradeoffs in performance. Early exits placed after more layers in the original network will yield improved accuracy compared to those placed after fewer. With ESNO, we refer to the complete detector formed by the FPN with an early exit n layers before the complete FPN as ESNO_n. This way, we may quickly denote which early exit is being used. With early exits at layers 34, 32, 29, 27, and 25, we create ESNO_0, ESNO_2, ESNO_5, ESNO_7, and ESNO_9. Note that ESNO_0 is equivalent to the original SSD.

Designing a robust method of combining results from multiple models poses a significant challenge. For object detection tasks, images are annotated using *detec-*

tions comprised of a bounding box, object class, and a confidence score. To produce a final result, finding appropriate *groups* of detections describing the same object must be conducted before the groups are *fused*.

Groups are typically formed by finding detections with high similarity. A widely accepted standard for measuring similarity in bounding box locations is the Intersection-over-Union (IoU) measure, listed in equation 3.1.

$$IoU_{A,B} = \frac{|A \cap B|}{|A \cup B|} \tag{3.1}$$

Groups may be formed as sets of detections where every bounding box shares a high IoU value with every other bounding box in the group. Further restrictions may be applied, such as requiring detections in a group to have the same object class and restricting each detector to one contribution per group. In our work, we group only bounding boxes of like class.

Current grouping approaches use IoU and make groups by greedily assigning each bounding box to a group. This is completed by selecting the highest confidence bounding box from a list of ungrouped bounding boxes and grouping it with all remaining bounding boxes with a high enough IoU value [20,41]. For the remainder of the chapter, we will refer to this as 'iterative' grouping.

Once groups have been identified, they must be fused into a single detection describing the object the detectors found. Many strategies to fuse groups have been proposed, such as assigning coordinates and confidences of fused detections to a weighted average of their component detections. Weights may be determined

by source detector [3], confidence scores [41], or other features. Optionally, previous works have investigated keeping the single highest confidence detection in a group [20], an approach similar to non-maximum suppression.

3.3 Methodology

In this chapter, we propose a method of grouping bounding boxes using a graph-theoretic approach using a graph formed by the pairwise similarity of all detections. We use the term *similarity graph* to describe an undirected, weighted graph that models detections as vertices. Between all pairs of vertices in a similarity graph, edge weights are derived by the IoU measure between the two bounding boxes represented by the incident vertices. With a similarity graph, clustering techniques may be applied to create groups of detections with high connectivity.

3.3.1 Graph Clustering

We find groups in a similarity graph by using a Highly Connected Subgraph (HCS) algorithm [42]. An unweighted graph G of order n is considered *highly connected* if all *minimum cuts* of the graph (smallest set of edges whose removal disconnects G) are at least of size $n/2$. Highly connected graphs are at least half as dense as a clique of equal size. This property relaxes the requirements of a group not to be as stringent as a clique while still requiring a high degree of connectivity. Grouping is then resilient to noise and artifacts since an HCS may contain vertices that do not share edges but share many neighbors. We utilize a recursive algorithm from [42]

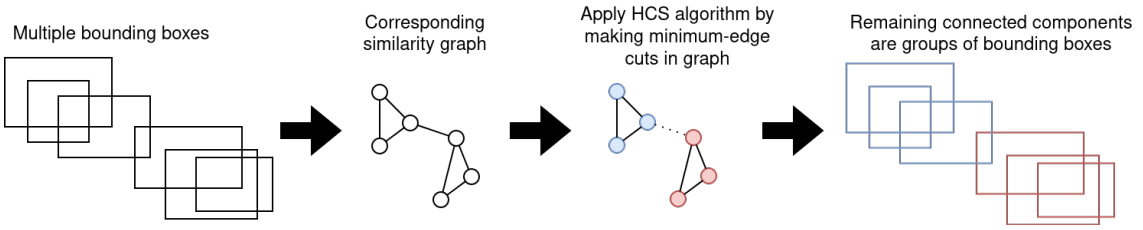


Figure 3.2: Example of similarity graph construction and use of HCS algorithm.

with a runtime of $O(2N * f(n,m))$ for N clusters, n nodes and m edges. $f(n,m)$ is the runtime complexity for finding a minimum edge cut in a graph. Randomized algorithms may accomplish this in $O(mlog^3n)$. Figure 3.2 shows a simple example of this process.

While algorithms exist to extend HCS techniques to weighted graphs [43], we find that it is more effective to make a similarity graph unweighted by removing edges below a minimum IoU value; edges above this threshold are considered unweighted edges, while all below are removed. This way, we may apply the unweighted algorithm. Grouping using the HCS algorithm will be referred to as 'HCS' grouping in the remainder of the chapter.

Finally, groups are unified into single bounding boxes by taking only the highest confidence bounding box in each group. Testing was conducted on other methods, such as averaging or using an average of only the highest confidence bounding box from each detector. For our system, we found non-maximum suppression to be the most effective.

3.3.2 Proposed Design

We use an elastic neural network for object detection built using the Single-shot Multibox Detector [16] using the VGG-16 Feature Pyramid Network (FPN) [44]. With each early exit, a post-exit subnetwork is added, which computes bounding boxes and confidences for the corresponding output from the FPN.

Each output ESNO_0 through ESNO_9 acts as its own independent detector in this ensemble. A diagram of the complete proposed system can be found in Figure 3.1. Each post-exit subnetwork may independently be included in an inference. Since each post-exit subnetwork requires additional computation, selectively choosing which subnetworks are used in an inference offers tradeoffs in inference time and accuracy. In our experiments, we investigate several configurations of our system.

3.4 Experiments

In this study, we conducted two experiments to analyze our system. First, we tune the parameters of our system to identify good grouping thresholds and methods. Then, to demonstrate the scalability of our system, we evaluated the tradeoffs between inference time and accuracy using different configurations of active early-exits and known optimal parameters from the previous experiment with both datasets.

3.4.1 Training/Testing

We evaluate our proposed framework using the Vehicle Aerial Imaging from Drone [1] dataset. We find this provides useful insight into our system's use in domains where scalable real-time detection is employed. This dataset contains 5,985 images of vehicle traffic captured from an overhead camera mounted on a UAV, containing an average of 8.1 objects per frame. We train our design on the dataset using a random selection of half of the image frames as our training set.

For both datasets, we first train ESNO_0. This trains the entire FPN, which will be re-used in ESNO_2 through ESNO_9. We then train each of the post-exit subnetworks. Since the entire FPN is already trained with ESNO_0, only training the post-exit subnetworks are necessary. Information regarding training hyperparameters may be found in Table 3.1.

To boost performance when using ensemble techniques with the VAID dataset, we used bagging techniques on the post-exit subnetworks [45]. Each post-exit subnetwork was trained on a random sample of one-third of the original training dataset. This had a negligible effect on the individual performance of each network.

	ESNO_0	ESNO_2	ESNO_5	ESNO_7	ESNO_9
Iterations	90k	30k	30k	30k	30k
Learning Rate	0.0025	0.0025	0.0025	0.0025	0.0025

Table 3.1: Training hyperparameters for VAID dataset.

3.4.2 Evaluation

Since many low-quality detections may negatively impact grouping, we evaluate our system at varying confidence thresholds by first removing all detections below the a given minimum. Since low-confidence boxes are typically discarded in practical systems, we find that this offers an accurate representation of the performance of our design. Performance in mAP is evaluated using the VOC2007 definition using code adapted from [4] and [46].

3.4.3 Tuning of System Parameters

First, we vary the threshold used in grouping in our system. This threshold is the lowest IoU value that may be considered an edge in a similarity graph. A value that is too high may allow the system to form groups of multiple occluded objects, while a value that is too low may generate multiple groups per object if bounding boxes from different exits are not perfectly overlapping.

We evaluate both our system and each ESNO layer at varying minimum confidence values and grouping IoU values. The performance of the individual ESNO outputs serves as a baseline for our experimentation. Results for the baseline may be found in Table 3.2, while the performance of the full ensemble may be found in Tables 3.3 and 3.4. When low-confidence detections are discarded and with a well-chosen grouping IoU value, our system performs better than any of its component networks.

Minimum Confidence	ESNO Output				
	$ESNO_0$	$ESNO_2$	$ESNO_5$	$ESNO_7$	$ESNO_9$
0.1	0.8768	0.8748	0.8738	0.8615	**0.8870**
0.2	0.8546	0.8341	0.8292	0.8506	**0.8670**
0.3	0.8062	0.8219	0.8170	0.7907	**0.8305**
0.4	0.7567	0.7987	0.7933	0.7418	**0.8069**
0.5	0.6941	0.7611	0.7436	0.7046	**0.7446**
0.6	0.6435	0.7253	0.6937	0.6541	**0.7191**
0.7	0.5669	0.6621	0.6435	0.6037	**0.6689**
0.8	0.5031	0.6244	0.5670	0.5530	**0.5925**
0.9	0.3359	0.4850	0.4907	0.4511	**0.5034**
0.99	0.0779	0.2312	0.2202	0.2461	**0.2335**

Table 3.2: mAP_{50} performance for ESNO with VAID dataset at varying minimum confidences. Bold values indicate highest performance at that minimum confidence.

3.4.4 Evaluation of Ensemble Scalability

With known optimal system parameters, we evaluate our system's inference time with varying active early exits. We chose a fixed grouping IoU value of 0.25 for all tests. Figures 3.3 and 3.4 plot the inference time and accuracy performance with mean average precision. In each plot, all points correspond to the performance of an ensemble comprised of one or more ESNO outputs. For instance, $ESNO_{0,2,5}$ denotes the system's performance using outputs from three early-exits. The results from non-active early exits are not calculated, reducing the performance cost. We find that adding additional early-exits to the ensemble, while increasing the inference time due to the higher computational load, substantially improves mAP.

3.4.5 Discussion of Results

During the initial experiment, where we tuned the system parameters, we utilized a fixed IoU for grouping. However, the results were inconclusive for a com-

Minimum Confidence	Grouping IoU 0.125	0.25	0.375	0.5	0.625	0.75
0.1	0.8866	**0.8868**	0.8845	0.8744	0.8085	0.7431
0.2	0.8666	***0.8776***	*0.8750*	0.8652	0.7995	0.7357
0.3	*0.8668*	***0.8668***	*0.8642*	*0.8547*	0.7897	0.7271
0.4	*0.8420*	***0.8422***	*0.8340*	*0.8314*	0.7686	0.7095
0.5	*0.8186*	***0.8188***	*0.8165*	*0.8078*	0.7468	0.6900
0.6	*0.7689*	***0.7690***	*0.7667*	*0.7585*	0.7020	0.6499
0.7	*0.7319*	***0.7320***	*0.7300*	*0.7215*	0.6664	0.6172
0.8	*0.6816*	***0.6817***	*0.6797*	*0.6725*	0.6203	0.5768
0.9	*0.5671*	***0.5671***	*0.5652*	*0.5582*	*0.5109*	0.4774
0.99	***0.2978***	***0.2978***	*0.2971*	*0.2938*	*0.2632*	*0.2524*

Table 3.3: mAP_{50} performance for ensemble of all ESNO early exits with VAID dataset using iterative grouping. Bold values indicate best performance and italicized values indicate better performance than baseline network. At low minimum confidence, no grouping IoU performs better than baseline.

pletely optimal value across all minimum confidence values. Moreover, since various object detection tasks may involve a different number of objects, varying levels of occlusion likelihood and severity, and different difficulty levels, it is necessary to tailor the process of grouping boxes for the specific task. Finding an approach that adaptively groups bounding boxes by incorporating run-time information in decision-making would help to generalize our system to other detection problems better.

In addition, we find that using a graph-based approach to grouping detections, while effective compared to the baseline ESNO network, does not improve mAP performance compared to existing methods. However, there may still be value in using a graph-based approach using graph clustering methods that have not been tested in this work. Compared to a greedy method, using a graph representation allows for grouping decisions to be made with all bounding boxes in mind. This information could help with the development of more flexible and comprehensive

Minimum Confidence	Grouping IoU 0.125	0.25	0.375	0.5	0.625	0.75
0.1	0.8801	**0.8835**	0.8788	0.8726	0.7845	0.6855
0.2	0.8630	***0.8756***	*0.8713*	0.8638	0.7759	0.6814
0.3	*0.8642*	***0.8649***	*0.8611*	*0.8534*	*0.7664*	0.6730
0.4	*0.8404*	***0.8411***	*0.8373*	*0.8304*	*0.7476*	0.6565
0.5	*0.8175*	***0.8186***	*0.8138*	*0.8068*	*0.7265*	0.6411
0.6	*0.7681*	***0.7689***	*0.7642*	*0.7574*	*0.6825*	0.6055
0.7	*0.7314*	***0.7319***	*0.7274*	*0.7207*	*0.6478*	0.5765
0.8	*0.6814*	***0.6816***	*0.6775*	*0.6717*	*0.6034*	0.5418
0.9	*0.5670*	***0.5671***	*0.5631*	*0.5575*	*0.4989*	0.4542
0.99	***0.2978***	***0.2978***	*0.2965*	*0.2930*	*0.2624*	*0.2498*

Table 3.4: mAP_{50} performance for ensemble of all ESNO early exits with VAID dataset using HCS grouping. Bold values indicate best performance and underlined values indicate better performance than baseline network. At low minimum confidence, no grouping IoU performs better than baseline.

approaches.

Ultimately, evaluating different ensembles demonstrates the scalability of a single system. Using an efficient training pipeline, we can generate numerous working ensemble configurations with tradeoffs in inference time and accuracy.

3.5 Conclusion

In this chapter, we have introduced an ensemble-of-detectors system formed by several outputs of a single elasticized neural network. We evaluated its use with a practical object detection dataset to demonstrate performance improvements over individual outputs of the same network. Further, we introduced additional methods toward more robust general ensembles of detector systems.

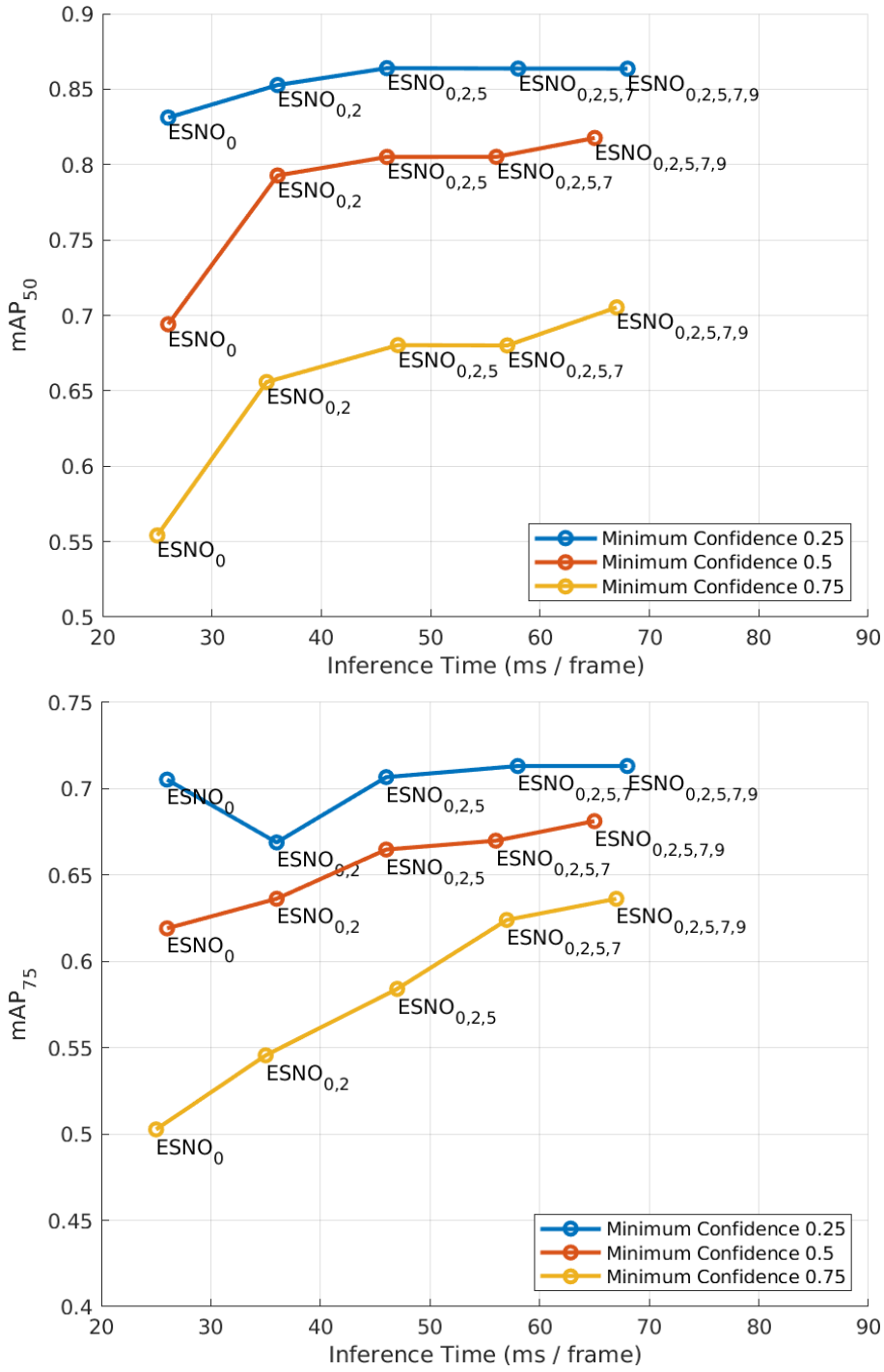


Figure 3.3: Performance of our system using iterative grouping using mAP_{50} and mAP_{75} evaluation, showing tradeoffs in inference time and accuracy for varying combinations of early exits.

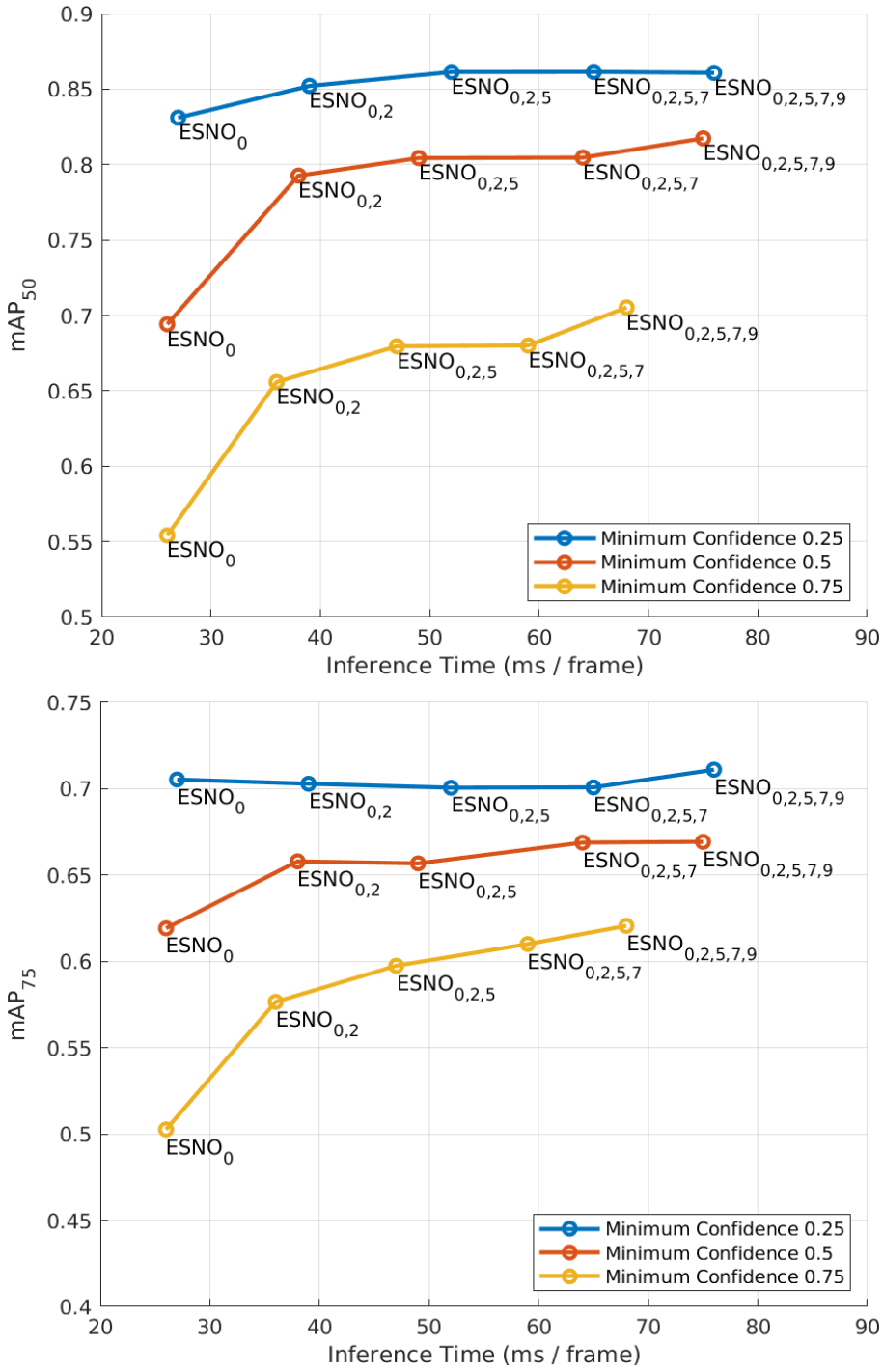


Figure 3.4: Performance of our system using HCS grouping using mAP_{50} and mAP_{75} evaluation, showing tradeoffs in inference time and accuracy for varying combinations of early exits.

Chapter 4

Conclusions and Future Work

In this thesis, we have investigated EOD systems for scalable real-time computation on lightweight platforms. To address the questions raised in Chapter 1, we analyzed methods to improve effective parallel computation of independent detectors and proposed two highly reconfigurable systems that offer useful tradeoffs in inference time, accuracy, and power consumption. In Chapter 2, the use of dataflow modeling techniques greatly improves resource utilization of ensemble-of-detector systems. We demonstrated a framework for a dynamic system built with resource usage in mind that allows for informed decision making for the active configuration of the system at run-time. This system leverages the natural parallelism in the dataflow model, allowing for improved inference time. In Chapter 3, we introduced an EOD system using early exits of an elastic neural network. We focused on scalability, creating a system offering numerous alternative configurations with varying trade-offs between accuracy and inference time. By allowing many possible systems to be unified and trained using a single pipeline, we allow for rapid, compact deployment to a wide range of target devices and flexible reconfiguration depending on application-specific requirements.

There are many potentially-useful research directions that build on our proposed new dataflow-based EOD systems. Embedded hardware may be designed to

further leverage the parallel nature of dataflow modeling and mitigate any overhead for scheduling and communication between actors. In addition, we introduce the notion of a selecting unit in our dataflow-based system, which makes closed-loop decisions about the active set of detectors based on run-time information. We propose one selecting unit algorithm, which optimizes expected performance subject to a dynamic cost constraint, based on available computational resources in the previous step. However, our system design allows for many other types of selecting unit strategies to be employed to optimize for specific use-cases. An investigation of such alternative selecting unit strategies is an interesting direction for future research.

The elastic-ensemble configurations we test (EODs formed by ESNO_0, $\text{ESNO}_{0,2}$, $\text{ESNO}_{0,2,5}$ and so on) are a small subset of the possible configurations for our system. Based on previous work with elastic networks for object detection [4], we expected these configurations to offer a good balance of speed and accuracy, since early exits placed further from the input side, such as ESNO_0, are expected to operate with higher detection accuracy compared to ones closer to the input, such as ESNO_9, due to layers being skipped in the latter. An exhaustive search of all configurations was not feasible in our work; further investigation is required to determine the optimal frontier for power-accuracy tradeoffs using an elastic-ensemble system. This may build on our notion of selecting units, as the active configuration of early-exits may be dynamic according to information only available at run-time. One useful set of elastic-ensemble configurations to investigate as part of future work are ones that progressively add more computationally intensive early-exits, e.g. $\text{ESNO}_{9,7}$, $\text{ESNO}_{9,7,5}$, and so on. This way, processing for unused layers of the FPN may be

avoided.

The optimal set of configurations may not be consistent across multiple datasets. As the expected size, quantity, and arrangement of objects in a dataset change, different early exits may lead to different levels of performance. Each layer in the ESNO's feature processing network may pick up on different information, allowing an attached post-exit subnetwork to identify different objects. Finding good guidelines for early-exit placement and use may greatly improve the performance of elastic-ensemble systems. Further, future work in this direction can significantly contribute to network explainability. Developing knowledge of how different early-exits interact within a single network provides significant insight into how the network detects individual objects.

Further, early-exits may be added to a multitude of different networks. While ESNO is based on the VGG-16 backbone [44], other object detection backbones, such as ResNet [47], may be used to form an elastic-ensemble. Additionally, beyond the Single Shot Detector, a diversity of object detectors may be adapted using the elastic neural networks framework to work as an elastic-ensemble. For a visual intuition of power and accuracy of current object detectors, we refer the reader to [48].

There also remain questions about training an elastic-ensemble system. In our work, we train post-exit subnetworks using a random third of our training data (a technique called *bagging* [45]). This technique is expected to boost performance of general EOD systems, since different networks will learn to identify different features. While this did improve performance for our elastic-ensemble system as

expected, a more comprehensive study of training approaches for elastic-ensemble systems may lead to further performance enhancement.

Finally, we introduced a novel approach to combining that utilizes a graph-based representation of bounding boxes and their relation by pairwise IoU values, which we call a similarity graph. While the method was determined to perform similarly to the state-of-the-art method when used with our elastic-ensemble system, there remain many directions to build on a graph-based approach to combining detections. One weakness with both our and previous methods is that they use a strict minimum IoU threshold to determine whether or not two bounding boxes should be considered for combination. The threshold is typically set to an experimentally-determined value. However, such a thresholding approach limits generalizability, possibly even within the same dataset, since overlapping objects are not guaranteed to occlude each other consistently. To both reduce the number of free parameters in EOD systems and improve general performance across many domains, a graph clustering algorithm could be employed that uses a weighted similarity graph. This avoids constructing an unweighted similarity graph by a strict threshold, as we do in our elastic-ensemble system after constructing a weighted representation using IoU values. Finding a suitable weighted graph clustering algorithm for our similarity graph approach remains an open question.

Bibliography

[1] Huei-Yung Lin, Kai-Chun Tu, and Chih-Yi Li, "VAID: An Aerial Image Dataset for Vehicle Detection and Classification," *IEEE Access*, vol. 8, pp. 212209–212219, 2020.

[2] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *Int. J. Comput. Vision*, vol. 88, no. 2, pp. 303–338, June 2010, Place: USA Publisher: Kluwer Academic Publishers.

[3] Eung-Joo Lee, Alexander Mattingly, Jing Xie, Heesung Kwon, and Shuvra S. Bhattacharyya, "Object detection on resource-constrained platforms using a configurable ensemble of detectors," in *Real-Time Image Processing and Deep Learning 2022*, Nasser Kehtarnavaz and Matthias F. Carlsohn, Eds. 2022, vol. 12102, p. 121020C, SPIE.

[4] R. Hector, M. Umar, A. Mehmood, Z. Li, and S. Bhattacharyya, "Scalable Object Detection for Edge Cloud Environments," *Frontiers in Sustainable Cities*, vol. 3, pp. 1–9, July 2021.

[5] G Chandan, Ayush Jain, Harsh Jain, and others, "Real time object detection and tracking using Deep Learning and OpenCV," in *2018 International Conference on inventive research in computing applications (ICIRCA)*. 2018, pp. 1305–1308, IEEE.

[6] Rashmika Nawaratne, Damminda Alahakoon, Daswin De Silva, and Xinghuo Yu, "Spatiotemporal anomaly detection using deep learning for real-time video surveillance," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 393–402, 2019, Publisher: IEEE.

[7] Xinchen Wang, Weiwei Zhang, Xuncheng Wu, Lingyun Xiao, Yubin Qian, and Zhi Fang, "Real-time vehicle type classification with deep convolutional neural networks," *Journal of Real-Time Image Processing*, vol. 16, no. 1, pp. 5–14, 2019.

[8] Eung-Joo Lee, Yi-Ting Shen, Lei Pan, Zhu Li, and Shuvra S Bhattacharyya, "DCT-based Hyperspectral Image Classification on Resource-Constrained Platforms," in *2021 11th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS)*. 2021, pp. 1–5, IEEE.

[9] Jiajia Liao, Yujun Liu, Yingchao Piao, Jinhe Su, Guorong Cai, and Yundong Wu, "GLE-Net: A Global and Local Ensemble Network for Aerial Object Detection," *International Journal of Computational Intelligence Systems*, vol. 15, no. 1, pp. 2, Jan. 2022.

[10] E. A. Lee and T. M. Parks, "Dataflow Process Networks," *Proceedings of the IEEE*, vol. 83, no. 5, pp. 773–799, 1995.

[11] Dawei Du, Yuankai Qi, Hongyang Yu, Yifan Yang, Kaiwen Duan, Guorong Li, Weigang Zhang, Qingming Huang, and Qi Tian, "The Unmanned Aerial Vehicle Benchmark: Object Detection and Tracking," 2018, _eprint: 1804.00518.

[12] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun, "Vision meets Robotics: The KITTI Dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[13] Joseph Redmon and Ali Farhadi, "YOLOv3: An Incremental Improvement," *CoRR*, vol. abs/1804.02767, 2018, arXiv: 1804.02767.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA, June 2016, pp. 779–788, IEEE Computer Society, ISSN: 1063-6919.

[15] Pranav Adarsh, Pratibha Rathi, and Manoj Kumar, "YOLO v3-Tiny: Object Detection and Recognition using one stage improved model," in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2020, pp. 687–694.

[16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg, "SSD: Single Shot MultiBox Detector," in *Computer Vision – ECCV 2016*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, Eds., Cham, 2016, pp. 21–37, Springer International Publishing.

[17] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," 2016.

[18] Richard Cobos, Jefferson Hernandez, and Andres G. Abad, "A fast multi-object tracking system using an object detector ensemble," in *2019 IEEE Colombian Conference on Applications in Computational Intelligence (ColCACI)*, 2019, pp. 1–5.

[19] Atakan Körez, Necaattin Barışçı, Aydın Çetin, and Uçman Ergün, "Weighted Ensemble Object Detection with Optimized Coefficients for Remote Sensing Images," *ISPRS International Journal of Geo-Information*, vol. 9, no. 6, 2020.

[20] Ángela Casado-García and Jónathan Heras, "Ensemble methods for object detection," in *ECAI 2020*. 2020, pp. 2688–2695, IOS Press.

[21] S. S. Bhattacharyya, E. Deprettere, R. Leupers, and J. Takala, Eds., *Handbook of Signal Processing Systems*, Springer, third edition, 2019.

[22] Honglei Li, Lei Pan, Eung Joo Lee, Zhu Li, Matthew J Hoffman, Anthony Vodacek, and Shuvra S Bhattacharyya, "Hyperspectral video processing on resource-constrained platforms," in *2019 10th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS)*. 2019, pp. 1–5, IEEE.

[23] Yaesop Lee, Yanzhou Liu, Karol Desnos, Lee Barford, and Shuvra Bhattacharyya, "Passive-Active Flowgraphs for Efficient Modeling and Design of Signal Processing Systems," *Journal of Signal Processing Systems*, vol. 92, Oct. 2020.

[24] E.A. Lee and D.G. Messerschmitt, "Synchronous data flow," *Proceedings of the IEEE*, vol. 75, no. 9, pp. 1235–1245, 1987.

[25] Fahimeh Yazdanpanah, Carlos Alvarez-Martinez, Daniel Jimenez-Gonzalez, and Yoav Etsion, "Hybrid Dataflow/von-Neumann Architectures," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 6, pp. 1489–1509, 2014.

[26] Michail G. Lagoudakis, "The 0-1 Knapsack Problem — An Introductory Survey," Tech. Rep., 1996.

[27] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2009, Publisher: IEEE.

[28] Paraskevi Nousi, Ioannis Mademlis, Iason Karakostas, Anastasios Tefas, and Ioannis Pitas, "Embedded UAV real-time visual object detection and tracking," in *2019 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. 2019, pp. 708–713, IEEE.

[29] AN Wilson, Abhinav Kumar, Ajit Jha, and Linga Reddy Cenkeramaddi, "Embedded sensors, communication technologies, computing platforms and machine learning for UAVs: A review," *IEEE Sensors Journal*, vol. 22, no. 3, pp. 1807–1826, 2021, Publisher: IEEE.

[30] Mesay Belete Bejiga, Abdallah Zeggada, and Farid Melgani, "Convolutional neural networks for near real-time object detection from UAV imagery in avalanche search and rescue operations," in *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. 2016, pp. 693–696, IEEE.

[31] Payal Mittal, Raman Singh, and Akashdeep Sharma, "Deep learning-based object detection in low-altitude UAV datasets: A survey," *Image and Vision Computing*, vol. 104, pp. 104046, 2020.

[32] Maria Tzelepi and Anastasios Tefas, "Human crowd detection for drone flight safety using convolutional neural networks," in *2017 25th European Signal Processing Conference (EUSIPCO)*. 2017, pp. 743–747, IEEE.

[33] Fatma Outay, Hanan Abdullah Mengash, and Muhammad Adnan, "Applications of unmanned aerial vehicle (UAV) in road safety, traffic and highway infrastructure management: Recent advances and challenges," *Transportation Research Part A: Policy and Practice*, vol. 141, pp. 116–129, 2020.

[34] Eung-Joo Lee, Damon M Conover, Shuvra S Bhattacharyya, Heesung Kwon, Jason Hill, and Kenneth Evensen, "Validation of object detection in UAV-based images using synthetic data," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III*. 2021, vol. 11746, pp. 584–601, SPIE.

[35] Yaesop Lee, Eung-Joo Lee, Damon M Conover, Yi-Ting Shen, Heesung Kwon, Shuvra S Bhattacharyya, Jason Hill, Kenneth Evensen, and G Jeremy Leong, "Archangel dataset: UAV-based imagery with position and pose metadata," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications IV*. 2022, vol. 12113, pp. 544–557, SPIE.

[36] Longyin Wen, Dawei Du, Zhaowei Cai, Zhen Lei, Ming-Ching Chang, Honggang Qi, Jongwoo Lim, Ming-Hsuan Yang, and Siwei Lyu, "UA-DETRAC: A New Benchmark and Protocol for Multi-Object Detection and Tracking," *Computer Vision and Image Understanding*, 2020.

[37] Pan Wei, John E Ball, and Derek T Anderson, "Fusion of an ensemble of augmented image detectors for robust object detection," *Sensors*, vol. 18, no. 3, pp. 894, 2018, Publisher: MDPI.

[38] Berat Mert Albaba and Sedat Ozer, "SyNet: An ensemble network for object detection in UAV images," in *2020 25th International Conference on Pattern Recognition (ICPR)*. 2021, pp. 10227–10234, IEEE.

[39] Rahee Walambe, Aboli Marathe, Ketan Kotecha, and George Ghinea, "Lightweight Object Detection Ensemble Framework for Autonomous Vehicles in Challenging Weather Conditions," *Computational Intelligence and Neuroscience*, vol. 2021, pp. 5278820, Oct. 2021.

[40] Yue Bai, Shuvra Bhattacharyya, Antti Happonen, and Heikki Huttunen, "Elastic Neural Networks: A Scalable Framework for Embedded Computer Vision," Sept. 2018, pp. 1472–1476.

[41] Roman Solovyev, Weimin Wang, and Tatiana Gabruseva, "Weighted boxes fusion: Ensembling boxes from different object detection models," *Image and Vision Computing*, vol. 107, pp. 104117, 2021.

[42] Erez Hartuv and Ron Shamir, "A clustering algorithm based on graph connectivity," *Information Processing Letters*, vol. 76, no. 4, pp. 175–181, 2000.

[43] Chuangxin Yang, Hong Peng, and Jiabing Wang, "A Clustering Algorithm for Weighted Graph Based on Minimum Cut," in *2008 First International Conference on Intelligent Networks and Intelligent Systems*, 2008, pp. 649–653.

[44] Karen Simonyan and Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," Sept. 2014, Publication Title: arXiv 1409.1556.

[45] David Opitz and Richard Maclin, "Popular ensemble methods: An empirical study," *Journal of artificial intelligence research*, vol. 11, pp. 169–198, 1999.

[46] Max deGroot and Ellis Brown, "SSD: Single Shot MultiBox Object Detector, in PyTorch," Mar. 2019.

[47] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep Residual Learning for Image Recognition," 2015, _eprint: 1512.03385.

[48] Alfredo Canziani, Thomas Molnar, Lukasz Burzawa, Dawood Sheik, Abhishek Chaurasia, and Eugenio Culurciello, "Analysis of deep neural networks," Sept. 2018.