Thesis for the Degree of Masters in Science

Gated YOLOv6 Dynamic Channel Gating for Efficient Object Detection

School of Computer Science and Engineering

The Graduate School

Hector Andres Acosta Pozo

June, 2024

The Graduate School

Kyungpook National University

Gated YOLOv6 Dynamic Channel Gating for Efficient Object Detection

Hector Andres Acosta Pozo

School of Computer Science and Engineering

The Graduate School

Supervised by professor Soon Ki Jung

Approved as a qualified thesis of Hector Andres Acosta Pozo

for the degree of Masters of Science

by the Evaluation Committee

June, 2024

Chairman: Prof. AAA

Prof. BBB

Prof. CCC

Prof. DDD

Prof. EEE

The Graduate School

Kyungpook National University

Contents

Acknowledgement

I would like to express my deepest appreciation to Professor Soon Ki Jung for his invaluable guidance, patience, and expertise throughout my research. His insights and encouragement were crucial to the success of this work.

I am deeply grateful to my family, whose unwavering support and belief in me have been my constant source of strength and motivation. To my mother, Soneyda Pozo, and my father, Andres Acosta, thank you for your endless love, encouragement, and sacrifices, which have not gone unnoticed. Your belief in my abilities has been a significant driving force in my pursuit of academic excellence.

I also wish to extend a special thanks to my sister, Diana Acosta, for her support, understanding, and companionship. Your presence and belief in my work have been a great source of comfort and encouragement.

To all of you, I am eternally grateful for your support and love. Thank you for being my guiding lights.

List of Tables

List of Figures

Abstract

In the evolving landscape of object detection, the advent of the YOLO (You Only Look Once) methodology has marked a significant leap forward for real-time applications, streamlining the process with its single-pass detection capabilities. Despite its advancements, the dynamic and often unpredictable nature of real-world environments, especially when operating on edge devices like security cameras, presents a pressing challenge for optimization. Addressing this, our research introduces the ”Gated Scene-Specific YOLO,” a novel adaptation of the YOLO framework, incorporating a dynamic gating mechanism aimed at enhancing computational efficiency with minimal compromise on the model’s detection prowess.

Traditional YOLO architectures, while robust, are predisposed to processing vast amounts of data, much of which may be extraneous or irrelevant to the task at hand. This not only leads to unnecessary computational overhead but also hinders the deployment of such models in environments where resources are limited. Our Gated Scene-Specific YOLO methodology seeks to alleviate this issue by integrating a mechanism that dynamically adjusts the activation of neural pathways based on the relevance to the observed scene. Through a meticulous process of gate generation and analysis during the training phase, our approach identifies and deactivates neural pathways that are consistently inactive across specific environmental conditions. This strategic deactivation allows the model to shed redundant computational weight, thus becoming more streamlined and efficient for the task it is deployed to perform.

The core of our research lies in demonstrating the practicality of

dynamically tuning deep learning models to their operational context, significantly reducing the computational load while minimally impacting the detection accuracy. Our empirical results showcase that the Gated Scene-Specific YOLO not only elevates processing speeds but also upholds a high standard of accuracy, making it a compelling solution for real-time object detection across a diverse array of settings. This contribution is particularly relevant for the deployment of object detection models in resource-constrained devices, where optimizing the balance between efficiency and performance is paramount.

In summary, the Gated Scene-Specific YOLO represents a meaningful stride towards more adaptable and resource-efficient object detection solutions. By tailoring model processing pathways to the specific demands of the environment, this research paves the way for the development of highly optimized, context-aware deep learning models, thereby enhancing the applicability and effectiveness of real-time object detection systems in fixed and scene specific settings.

1 Motivation and Objectives

1.1 Introduction

Object detection stands as a foundational pillar within the field of computer vision, influencing an extensive range of applications from advanced surveillance systems to the dynamic realm of autonomous driving technologies. The rapid evolution of deep learning methodologies has significantly propelled the field forward, introducing architectures capable of not only enhancing the accuracy of object detection but also its efficiency and adaptability in real-time processing environments. Among these innovations, the YOLO (You Only Look Once) architecture, introduced by Redmon et al. [1], has emerged as a significant contribution, revolutionizing the way real-time object detection is approached by enabling swift and efficient processing without the need for iterative detection stages. This architecture underscores the potential of modern object detection systems, demonstrating remarkable versatility across a variety of computing environments, from high-powered servers to more constrained devices such as smartphones and embedded systems.

Despite the advancements brought forth by YOLO and its subsequent iterations, challenges remain, particularly in the context of edge computing where computational resources are limited, and the demand for real-time processing is paramount. Recognizing the potential for further optimization, our study delves into the exploration of YOLOv6 [2], [3], a variant designed with a keen focus on hardware efficiency and optimized for real-time applications. The architecture of YOLOv6 [2], [3] serves as an ideal foundation for our

investigation, given its emphasis on performance in hardware-constrained environments. Within such contexts, minor enhancements can lead to substantial gains in processing speed and overall system efficiency, thereby improving the applicability of YOLO-based models in edge devices.

To advance the capabilities of YOLOv6 [2], [3], our research introduces the ”Gated Scene-Specific YOLO,” a novel framework that marries the concept of dynamic gating with the principle of model pruning specifically tailored to the YOLO architecture. Model pruning, a technique aimed at reducing the computational burden of neural networks, achieves this by eliminating superfluous or insignificant parameters, thereby refining the network’s structure with minimal detriment to its performance. Our approach innovates beyond traditional model pruning by implementing a dynamic gating mechanism that adjusts in real-time to the distinctive characteristics of the input scene, thereby optimizing the efficiency of the object detection process with minimal sacrifice in accuracy.

A cornerstone of our methodology is the adaptation of Improved SemHash [4], a technique initially proposed by Kaiser and Bengio and further refined by Chen et al. [5] This approach facilitates the generation of binary gates during the model training phase, enabling selective activation or deactivation of network filters in response to variations in the input. Through dynamic gate generation and subsequent analysis tailored to specific scenes, our method identifies filters that consistently remain inactive. These filters are then statically pruned from the network for deployment, allowing the model to operate more efficiently by focusing computational resources on active, scene-relevant pathways. The Gater Network [5], integral during the training phase for gate determination, is thus

rendered unnecessary during actual deployment, replaced by the pre-determined, statically applied gates that ensure both efficiency and specificity in detection.

Our contributions to the YOLO architecture, through the integration of a gating network and the innovative use of Improved SemHash [4], not only enable precise control over network activity but also herald significant improvements in computational efficiency and detection accuracy. The effectiveness of our approach is substantiated through rigorous experimental validation, focusing on key performance metrics such as floating-point operations per second (FLOPs), frames per second (FPS), and mean Average Precision (mAP@0.5:0.95). Our findings reveal a notable increase in FPS for the Gated Scene-Specific YOLO model in comparison to its YOLOv6 counterpart, with a slight compromise on detection robustness, as evidenced by stable mAP scores. This research thus presents a significant step forward in optimizing deep learning models for object detection, especially in scenarios where computational resources are at a premium.

1.2 Motivation

The relentless pursuit of advancements in computer vision, specifically within the domain of object detection, has been driven by the escalating demands of modern applications. These range from enhancing public safety through sophisticated surveillance mechanisms to propelling the future of mobility with autonomous vehicles. The inception of deep learning architectures like YOLO has significantly narrowed the gap between theoretical possibility and practical implementation, offering a glimpse into the potential of real-time object

detection systems. Yet, as these technologies are increasingly deployed in real-world scenarios, particularly on the edge, the limitations of current models under resource-constrained conditions become apparent. The motivation behind our work is rooted in the desire to transcend these limitations, pushing the boundaries of what is possible with existing object detection frameworks.

Our focus on YOLOv6 [2], [3], known for its balance of speed and accuracy, stems from a recognition of the critical need for optimization in edge computing scenarios where resources are scarce yet the demand for high-performance computing is incessant. The drive to refine and enhance the efficiency of such models without compromising their detection capabilities underlines our research. We are particularly inspired by the potential impact of our work on a wide array of applications, from low-power IoT devices to mobile applications requiring real-time analysis and feedback, envisioning a future where advanced object detection is not only possible but also practical and pervasive, regardless of computational limitations.

1.3 Objectives

The primary objective of our research is to develop an optimized version of the YOLOv6 [2], [3] architecture, termed ”Gated Scene-Specific YOLO,” which incorporates a dynamic gating mechanism to enhance computational efficiency in object detection tasks, particularly in edge computing environments. To achieve this, we aim to:

Implement Dynamic Gating: Integrate a dynamic gating mechanism that adapts to the unique characteristics of input scenes, thereby selectively activating

relevant neural pathways and improving model efficiency.

Optimize Through Model Pruning: Apply model pruning techniques in conjunction with dynamic gating to eliminate redundant parameters and streamline the model, focusing computational resources on critical tasks.

Leverage Improved SemHash [4]: Utilize the Improved SemHash technique for effective gate generation during training, allowing for precise control over network activity and further optimization of the model for specific scenarios.

Demonstrate Practical Efficacy: Validate the effectiveness of the Gated Scene-Specific YOLO model through extensive testing, focusing on key performance metrics such as processing speed (FPS), computational efficiency (FLOPs), and detection accuracy (mAP@0.5:0.95).

Enhance Real-World Applicability: Ensure that the optimized model maintains high detection accuracy while significantly reducing computational load, making it suitable for deployment in real-world, resource-constrained environments.

Through these objectives, our research seeks to address the pressing challenges of deploying sophisticated object detection models in edge computing scenarios, offering a pathway to more efficient, accurate, and accessible real-time object detection technologies.

2 Related Work

The exploration of efficiency within neural network architectures, particularly for object detection in computationally constrained environments, constitutes a significant area of research. This section delves into various methodologies and developments that have shaped the current landscape of efficient neural network design, highlighting the relevance and novelty of our approach within this context.

2.1 Sparsity and Conditional Computation

A fundamental concept in the pursuit of neural network efficiency is the integration of sparsity and conditional computation mechanisms. Sparsity in neural networks refers to the idea that not all neurons or connections (weights) are necessary for every input. By identifying and activating only a subset of the neural pathways for a given input, significant reductions in computational overhead can be achieved without sacrificing the model's ability to represent complex functions. This principle is akin to how decision trees operate, where at each node a decision is made that leads to a subset of the next possible states, thus not exploring all branches of the tree for a given input.

The concept of conditional computation extends this idea further by introducing mechanisms that allow a neural network to adapt its computation pathways dynamically based on the input. This adaptability ensures that only the most relevant parts of the network are engaged during the forward pass, which not only saves computational resources but also helps in reducing overfitting by

limiting the effective capacity of the model based on the complexity of the input.

Introduced by Bengio et al. [6], the idea of selectively activating neural pathways has been a cornerstone in the development of more efficient neural network architectures. The authors suggest that such mechanisms could allow for deeper and more complex models by allocating computational resources more judiciously. By simulating sparsity and conditional computation, networks can potentially achieve a balance between depth and width, optimizing the computational cost without compromising the benefits of distributed representations.

Several approaches have been proposed to implement sparsity and conditional computation in neural networks. One method involves pruning, where connections between neurons are removed based on their importance to the model's performance. Another approach is the use of gating mechanisms, where gates control the flow of information in the network, effectively turning on or off certain pathways based on the input.

Furthermore, research in this area has explored the use of dropout as a form of introducing dynamic sparsity during training, encouraging the network to develop more robust and efficient representations. Additionally, recent advancements have introduced techniques such as dynamic routing between capsules in Capsule Networks, where the network learns complex spatial hierarchies in data by activating only the relevant parts of the network for a given input.

In summary, the integration of sparsity and conditional computation into neural networks offers a promising avenue for enhancing computational

efficiency. By leveraging these concepts, researchers aim to develop models that can process information more effectively, adapting their structure to the demands of the input while preserving the representational power that characterizes deep learning. This ongoing exploration holds the potential to significantly impact the development of neural network architectures, especially in contexts where computational resources are limited.

2.2 Dynamic Sparse Training

Dynamic Sparse Training (DST), as proposed by Liu et al. [7], represents a significant leap forward from static sparsity models towards a more flexible and efficient neural network architecture. DST addresses one of the primary concerns in neural network efficiency—how to use the minimal number of parameters without sacrificing the model's ability to learn complex patterns. Unlike traditional training methods that rely on a fixed architecture, DST allows the network to adjust its architecture dynamically during the training process. This is achieved by periodically redistributing the network's connections to focus more on those that are most beneficial for learning the current task.

The key innovation of DST lies in its ability to identify and leverage the most informative connections within a network based on the training data. By doing so, it optimizes both the model capacity and computational resources, directing them towards the aspects of the data that are most crucial for performance. This method not only improves the efficiency of the network but also has the potential to enhance its generalization ability by preventing overfitting to less relevant features. However, implementing DST effectively

requires sophisticated mechanisms for deciding when and how to adjust the network's sparsity. This dynamic adjustment process introduces additional complexity and computational overhead, which can be challenging to manage, particularly in environments where computational resources are strictly limited.

2.3 Dynamic Filter Selection

Building upon the concept of dynamic adjustment in neural networks, Chen et al. [5] introduced an approach specifically designed for Convolutional Neural Networks (CNNs), dubbed GaterNet. This method revolutionizes the way CNNs are structured by implementing a dynamic filter selection mechanism that adapts in real-time to the input. GaterNet operates by employing a gating network that runs parallel to the main network. The gating network analyzes the input and determines which filters in the main network are most relevant for processing that particular input. By activating only a subset of the filters, GaterNet significantly reduces the computational load required for each forward pass through the network.

The brilliance of dynamic filter selection lies in its capacity to maintain high levels of accuracy while dramatically enhancing computational efficiency. This is especially pertinent for applications requiring real-time processing, such as video analysis or mobile computing, where resource constraints are a major consideration. Moreover, the adaptability of GaterNet enables the CNN to focus its computational power on the most informative features of the input, potentially leading to better performance on complex tasks. Despite its advantages, the design and training of a gating network that can accurately predict the most

effective filters for any given input pose substantial challenges, requiring careful consideration of the trade-offs between complexity, efficiency, and accuracy.

2.4 Neural Network Pruning

Parallel to dynamic selection techniques, neural network pruning strategies focus on reducing network complexity without severely impacting performance. Zhang et al. [8] presented a methodical approach to pruning using gradient descent, streamlining the network in a manner that minimizes performance degradation. This technique aligns with the overarching goal of developing efficient, robust neural network models suitable for a wide range of applications.

2.5 Efficient Object Detection Models

The quest for efficiency extends into the domain of object detection, with MobileNet YOLO by Howard et al. [9] setting a precedent. By leveraging depth-wise separable convolutions, MobileNet YOLO offers a significant reduction in computational demands while sustaining performance levels, demonstrating the viability of real-time object detection on embedded systems. This integration of MobileNet with the YOLO framework exemplifies the practical application of efficiency-driven design principles in object detection models.

2.6 Adverse Conditions Object Detection

Emerging research by Kalwar et al. [10], titled 'GDIP: Object-Detection in Adverse Weather Conditions Using Gated Differentiable Image Processing,' explores object detection under challenging environmental conditions. Utilizing a gated image processing technique, this work provides a novel perspective on efficiency and adaptability, presenting a potential area for future comparative studies with our gated scene-specific approach.

2.7 Our Contribution

Building upon these foundational advancements, our work introduces the "Gated Scene-Specific YOLO," which incorporates dynamic gating and model pruning tailored specifically for the YOLO architecture. Unlike existing methodologies, our approach leverages Improved SemHash to establish static gating configurations post-training, significantly reducing computational requirements during inference. This innovation sets our work apart in the pursuit of efficient, real-time object detection, particularly in environments where computational resources are at a premium.

3 Methodology

Our study introduces an innovative approach to optimize object detection within the constraints of limited computational resources, specifically through the development of the Gated Scene-Specific YOLO architecture. This methodology leverages a dynamic gating mechanism, a novel adaptation within the established YOLO framework, to enhance the model's efficiency and adaptability by selectively processing only the neural pathways relevant to the observed scene. The primary focus of this section is to delineate the systematic approach employed in the design, implementation, and validation of this architecture. We outline the steps taken to integrate dynamic gating with the YOLO architecture, including the development and deployment of the Gater Network, the modifications applied to the YOLO architecture for accommodating gating mechanisms, and the analytical methods used to assess the model's performance in varied scene-specific contexts. The subsequent paragraphs will detail each component of our methodology, elucidating the technical strategies employed to achieve a balance between computational efficiency and detection accuracy in resource-constrained environments.

3.1 Gater Network

The cornerstone of our Gated Scene-Specific YOLO model is the Gater Network, a novel component designed to enhance the model's computational efficiency by dynamically modulating the activation of neural pathways based on the specific features of the input scene. Utilizing the principles of conditional computation, the Gater Network strategically deactivates certain parts of the neural network that are

deemed irrelevant for a given scene, thereby reducing unnecessary computational overhead without compromising the model's object detection capabilities.

The Gater Network operates in two main phases: gate generation during training and gate application during inference. During the training phase, the network learns to identify and generate a set of binary gates based on the distinguishing features of the input images. These gates serve as indicators for activating or deactivating specific channels within the YOLO architecture, depending on their relevance to the task at hand. The process leverages Improved Semantic Hashing, a technique inspired by the work of Kaiser and Bengio [4] and further explored by Chen et al. [5], to ensure that the gate generation is both efficient and effective.

Gate Generation and Application. In practice, the Gater Network employs a specialized architecture, typically based on a lightweight convolutional neural network (CNN) like ResNet-18 [11], to process input images and extract relevant features. These features are then passed through a series of fully connected layers, culminating in the generation of binary gates. Each gate corresponds to a specific channel or filter in the subsequent layers of the YOLO architecture, dictating whether it should be activated (1) or deactivated (0) based on the input scene's characteristics.

During inference, the pre-determined gates are applied to the YOLO architecture, enabling the model to focus its computational resources only on the parts of the network that are essential for detecting objects in the current scene. This selective processing significantly enhances the model's efficiency, particularly in scenarios where computational resources are limited, such as edge

devices.

Dynamic Adaptation to Scene Features. One of the key advantages of the Gater Network is its ability to dynamically adapt to various scenes without the need for manual tuning. By analyzing the input scene and applying the appropriate gates, the network ensures that only the most relevant features are processed. This adaptability is crucial for applications such as surveillance and traffic monitoring, where the scene's characteristics may vary significantly but remain consistent over time.

In summary, the Gater Network introduces a significant advancement in the YOLO architecture by incorporating a dynamic gating mechanism that intelligently deactivates filters based on the unique features of each input scene. This approach not only enhances computational efficiency but also maintains high detection accuracy, demonstrating the potential of selective gating in improving real-time object detection applications.

3.1.1 Feature Extraction

The feature extraction process within the Gater Network plays a pivotal role in our Gated Scene-Specific YOLO model, laying the foundation for the subsequent gating mechanism by identifying the critical features from the input images. This process utilizes the ResNet-18 architecture, renowned for its efficiency and effectiveness in capturing salient features from images with minimal computational resources [11].

ResNet-18 for Efficient Feature Representation. ResNet-18 is chosen for its shallow architecture compared to deeper variants, striking an optimal balance between computational efficiency and the ability to extract rich, discriminative features. This balance is crucial for our model's application in resource-constrained environments, where maintaining high accuracy without excessive computational burden is essential. The feature extraction is formalized as follows:

$$F_{\text{extract}}(x) = \text{Flatten}(\text{AdaptivePool}(f_{\text{net}}(x))), \tag{1}$$

where $x \in \mathbb{R}^{h_0 \times w_0 \times c_0}$ represents the input image, and $f_{\text{net}}(x)$ denotes the feature representation extracted by the ResNet-18 network. The $AdaptivePool$ operation ensures that the output from f_{net} is standardized, facilitating uniformity across different input dimensions. The final feature vector $F_{\text{extract}}(x)$ is obtained by flattening the pooled features, making it suitable for further processing by the fully connected layers leading to gate generation.

Adaptive Pooling for Dimensionality Reduction. Adaptive pooling plays a crucial role in our feature extraction process by dynamically adjusting the size of the feature maps to a fixed dimension, thereby enabling a consistent input size for the fully connected layers regardless of the original input image size. This step is critical for ensuring that the feature extraction process remains efficient and scalable across varying image dimensions.

Feature Flattening for Gate Generation. After adaptive pooling, the feature map undergoes a flattening operation, transforming it into a one-dimensional vector suitable for analysis by the subsequent layers responsible for gate generation. This flattened feature vector encapsulates the essential information required for determining the relevance of specific neural pathways in the YOLO architecture, forming the basis for the dynamic gating mechanism.

By employing ResNet-18 for feature extraction, we leverage its proven capabilities in efficient feature representation while ensuring that the Gater Network remains computationally manageable. The combination of adaptive pooling and feature flattening further streamlines the process, preparing the extracted features for the critical task of gate generation, which ultimately drives the selective activation and deactivation of neural pathways in our model.

3.1.2 Binary Gates

Binary gates within the Gater Network are pivotal for modulating the activity of neural pathways in the YOLO architecture, enabling selective processing based on the input scene's characteristics. These gates are generated through a series of operations that map the high-dimensional feature vector obtained from the feature extraction phase to a binary vector, where each element corresponds to a specific channel in the YOLO architecture. The generation and application of these binary gates are fundamentally rooted in the concept of Improved Semantic Hashing [4], [5], which ensures the differentiability of the gating process during training while maintaining binary decisions during inference.

Mapping to Binary Space. The process begins with mapping the extracted features to the binary space. This mapping is achieved through a dual-layer architecture comprising two fully connected layers that introduce a bottleneck layer to efficiently manage the parameter space while ensuring the representational capacity:

$$f_0 = \text{ReLU}(\text{BatchNorm}(\text{FC1}(f))), \tag{2}$$

$$g_0 = \text{FC2}(f_0). \tag{3}$$

In Equation (2), f denotes the flattened feature vector, and f_0 represents the intermediate representation at the bottleneck. $FC1$ and $FC2$ are the two fully connected layers, with ReLU activation and Batch Normalization applied after the first layer to enhance training stability and non-linearity. Equation (3) maps f_0 to g_0, the pre-activation gate vector, poised for binary conversion.

Improved Semantic Hashing. The cornerstone of our binary gate generation process is the Improved Semantic Hashing technique, which allows the network to generate binary gates in a differentiable manner during training. This adaptability is crucial for integrating the gating mechanism within the end-to-end training process of the YOLO architecture. The method involves adding a noise component to the pre-activation gate vector and applying a sigmoid function to obtain a soft binary gate:

$$g_{\text{noisy}} = g_0 + \epsilon, \tag{4}$$

$$g_\alpha(i) = \text{clamp}(1.2 \times \sigma(g_{\text{noisy}}(i)) - 0.1, 0, 1), \tag{5}$$

where ϵ represents a noise vector sampled from a Gaussian distribution. g_{noisy} denotes the noisy gate vector, and g_α is the soft binary gate vector, with each element being clamped between 0 and 1 to ensure binary-like behavior. This approach facilitates the backpropagation of gradients during training, allowing for the optimization of the gating mechanism.

Binary Decision Making. During inference, the soft binary gates g_α are converted into hard binary decisions g_β, which directly control the activation of corresponding channels in the YOLO architecture:

$$g_\beta(i) = \begin{cases} 1, & \text{if } g_\alpha(i) \geq 0.5, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

This binary decision-making process ensures that only the relevant features contributing to the object detection task are processed, thereby enhancing the model's computational efficiency without compromising its detection performance.

In essence, the Binary Gates subsection outlines the sophisticated methodology employed to generate and apply binary gates within the Gater

Network, leveraging Improved Semantic Hashing to marry the model's need for differentiability during training with the necessity for discrete decision-making during inference. This delicate balance enables the Gated Scene-Specific YOLO model to dynamically adapt to various scene features, optimizing computational resources while maintaining high accuracy in object detection tasks.

3.1.3 Loss Function

The loss function of our Gated Scene-Specific YOLO model is meticulously designed to optimize both the object detection performance and the efficiency of the gating mechanism. It is an amalgamation of multiple components that jointly guide the training process towards achieving high accuracy in object detection while ensuring computational efficiency through effective gate generation. The overall loss function is formulated as follows:

$$L = \alpha_{\text{cls}} L_{\text{cls}} + \alpha_{\text{iou}} L_{\text{iou}} + \alpha_{\text{dfl}} L_{\text{dfl}} + \lambda \cdot L_{\text{gate}}, \tag{7}$$

where L_{cls}, L_{iou}, and L_{dfl} represent the classification loss, Intersection over Union (IoU) loss, and distance-IoU loss, respectively. These components are weighted by their respective coefficients α_{cls}, α_{iou}, and α_{dfl}, ensuring a balanced contribution to the overall loss. L_{gate} signifies the gating loss, which encourages the sparsity of the gating mechanism, and λ is the regularization coefficient controlling its influence on the total loss.

Classification and IoU Losses. The classification loss L_{cls} and IoU loss L_{iou} are fundamental to object detection models, ensuring accurate classification and localization of objects within the scene [2], [3]. The distance-IoU loss L_{dfl} further refines the bounding box predictions, enhancing the precision of object localization.

Gating Loss. The gating loss L_{gate} is pivotal in training the Gater Network, promoting the generation of efficient and effective gates. It is designed to encourage the model to minimize the number of active gates, thereby reducing the computational load during inference. The gating loss is formulated as an L_1 regularization term over the gate vector g, encouraging sparsity:

$$L_{\text{gate}} = \frac{1}{c}\|g\|_1 = \frac{1}{c}\sum_i |g_i|, \tag{8}$$

where c is the total number of gates, and g_i represents the individual gate values. By penalizing the sum of the absolute values of the gate vector, the model is incentivized to deactivate unnecessary channels, thereby streamlining the network for increased efficiency.

Balancing Detection Performance and Efficiency. The coefficients α_{cls}, α_{iou}, α_{dfl}, and the regularization term λ play crucial roles in balancing the model's object detection performance with the efficiency of the gating mechanism. By adjusting these parameters, we can fine-tune the model to prioritize either detection accuracy or computational efficiency, depending on the application requirements.

In conclusion, the loss function of the Gated Scene-Specific YOLO model is a comprehensive formulation that encompasses the dual objectives of maintaining high accuracy in object detection and optimizing computational efficiency through an effective gating mechanism. This strategic combination of loss components ensures that the model can be effectively trained to meet the demands of real-world object detection tasks in resource-constrained environments.

3.2 YOLO Architecture

In the design of our Gated Scene-Specific YOLO model, we leverage the strengths of the YOLOv6 architecture, as proposed by Li et al. [2], [3], renowned for its exceptional real-time performance and optimal utilization of hardware resources. The YOLOv6 framework serves as the foundational backbone, onto which we have integrated the GaterNet, an auxiliary network that introduces a dynamic gating mechanism. This integration enables our model to perform selective feature processing, substantially improving computational efficiency without sacrificing detection accuracy. The following figure illustrates the modified YOLO architecture augmented with the GaterNet functionality:

The incorporation of GaterNet into the YOLO architecture facilitates a more selective and efficient processing approach, primarily through three key components:

Gated Efficient Reparameterizable Backbone (EfficientRep): This component forms the core of our feature extraction mechanism. By employing gating mechanisms, the EfficientRep selectively emphasizes critical features while minimizing attention to redundant information. This selectivity ensures

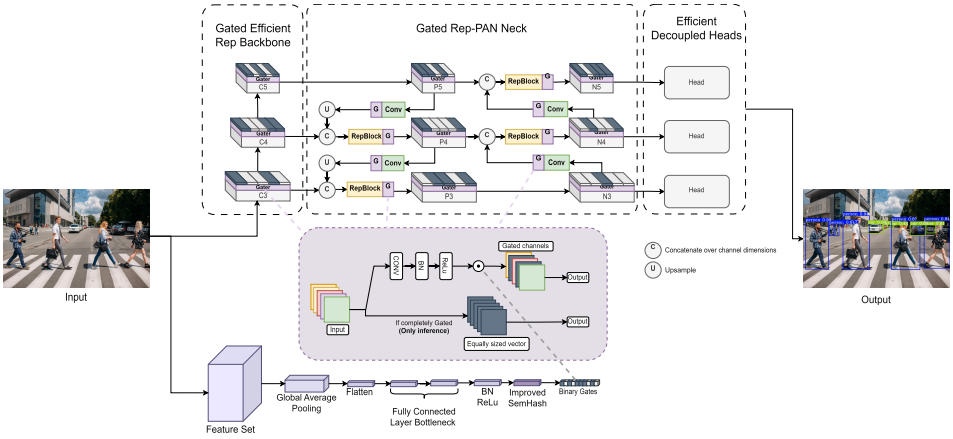

Fig. 1: Illustration of the YOLOv6 architecture enhanced with the GaterNet, featuring sections like 'Gated Efficient Reparameterizable Backbone (EfficientRep)', 'Gated Rep-Pan Neck', and 'Gated Efficient Decoupled Heads', each engineered for maximized performance and efficiency.

that subsequent processing layers focus computational resources on analyzing features of utmost relevance to the detection task at hand.

Gated Rep-Pan Neck: Inspired by the PANet topology and enhanced with Rep blocks for added efficiency, the Gated Rep-Pan Neck dynamically adjusts feature resolution and scale. It optimizes the integration and refinement of features passed from the EfficientRep to the detection heads, playing a vital role in ensuring the accuracy of object detection.

Gated Efficient Decoupled Heads: Drawing on the original design of YOLOv6, this component employs a hybrid-channel strategy to reduce the convolutional layers count. Gates within these heads dynamically prioritize the processing of pertinent features, thus significantly curtailing unnecessary computations and reducing inference time.

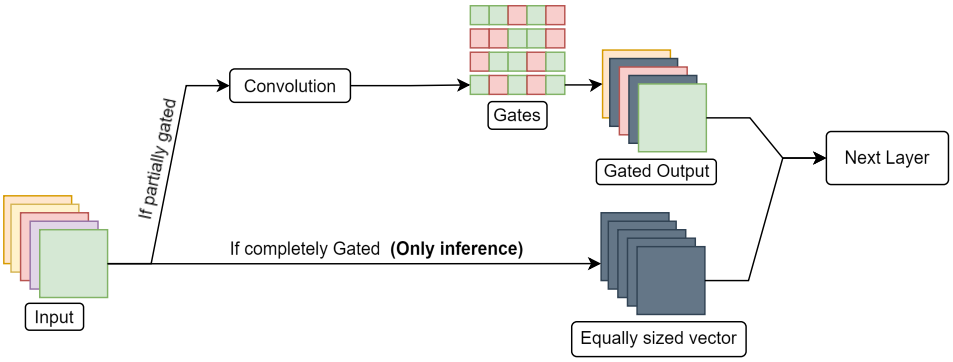

Fig. 2: Illustration of the YOLOv6 architecture enhanced with the GaterNet, featuring sections like 'Gated Efficient Reparameterizable Backbone (EfficientRep)', 'Gated Rep-Pan Neck', and 'Gated Efficient Decoupled Heads', each engineered for maximized performance and efficiency.

3.2.1 YOLO Gate Module

The YOLO Gate Module significantly enhances the YOLO architecture by integrating the dynamic gating functionality, offering a novel approach to managing feature processing both during training and inference phases. Through this module, the network learns to modulate its output by performing element-wise multiplication of the convolutional outputs with the gating signals generated by the GaterNet:

$$\mathbf{G} = [g_1, g_2, \ldots, g_c], \tag{9}$$

$$g_{\text{closed}} = \frac{\sum_{i=1}^{c} \mathbf{1}(G_i = 0)}{c}, \tag{10}$$

$$g(x) = \begin{cases} 0, & \text{if } g_{\text{closed}} > 0.99, \\ \text{Conv}(x) \odot \mathbf{G}, & \text{otherwise.} \end{cases} \tag{11}$$

Here, $\mathbf{G}$ represents the gate vector, indicating the activation status of corresponding feature maps within the network's layers. The module evaluates the proportion of active gates to decide between bypassing certain convolutional operations or allowing them, thus tailoring the network's processing power to the task's specific requirements. This selective gating mechanism, visualized in Figure 2, ensures optimal resource allocation during feature propagation, enhancing the model's overall efficiency and effectiveness in real-time object detection tasks.

3.3 Analysis Step

The analysis step represents a critical phase in our Gated Scene-Specific YOLO methodology, where the model's adaptability and efficiency are fine-tuned for specific scene conditions. Through a detailed analysis process, we evaluate the model's performance over a designated scene for a set duration, allowing us to observe and record the operational status of the gating mechanism in real-time scenarios. This process involves a meticulous examination of the gating decisions made by the GaterNet for each frame processed, focusing on identifying which filters are essential for maintaining detection accuracy and which can be deactivated without compromising performance.

Monitoring Gating Decisions. The essence of this analysis lies in monitoring the closure rate of each gate across the network, a task accomplished by systematically tracking the activation state of each gate throughout the inference process:

$$\Gamma(t) = \{\gamma_1(t), \gamma_2(t), \ldots, \gamma_n(t)\}, \tag{12}$$

where $\Gamma(t)$ represents the set of gating states at time t, and $\gamma_i(t)$ indicates the state (active or inactive) of the i-th gate. By analyzing these states over time, we can identify patterns of gate usage that correlate with specific features or elements within the scene.

Grouping and Analysis of Gating Decisions. To facilitate a clearer understanding and interpretation of the gating decisions, we aggregate these decisions by network sections, as visualized in the following figure:

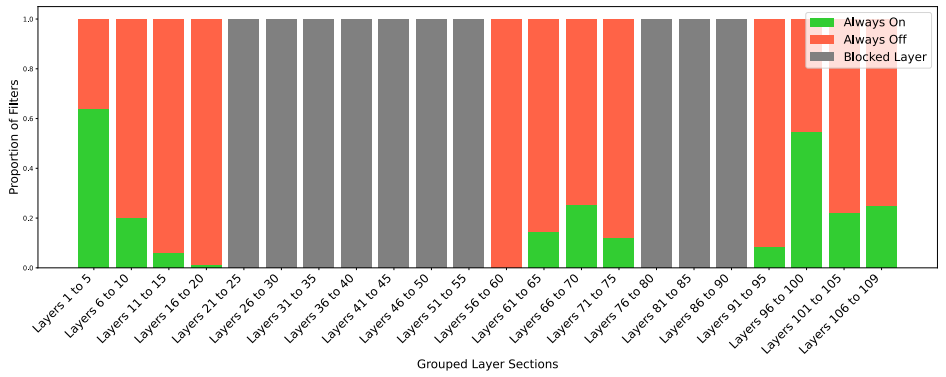

Fig. 3: Graphical representation of the distribution of gating states across various sections of the network, highlighting the sections with consistently active (”Always On”) or suppressed (”Always Off”) filters, as well as those that are completely deactivated (”Blocked Layer”).

Distillation of Static Gating Configuration. The ultimate objective of this analytical step is to derive a static gating configuration that can be consistently applied to the designated scene and similar scenarios in future inferences. This involves discerning which gates remain predominantly active or inactive throughout the analysis period and categorizing them accordingly:

$$G_{\text{static}} = \text{distill}(\Gamma(t), \forall t \in T), \tag{13}$$

where G_{static} denotes the derived static gating configuration, and T is the duration of the analysis period. This static configuration enables the model to bypass the need for dynamic gate computations by the GaterNet in subsequent inferences, significantly reducing computational overhead.

Practical Implications and Future Directions. By implementing this analysis step, we establish a methodical approach to optimizing the Gated Scene-Specific YOLO model for specific environmental conditions, ensuring that the model's computational resources are allocated efficiently. The findings from this phase not only enhance our understanding of the dynamic interaction between scene features and the gating mechanism but also pave the way for future research into adaptive and efficient object detection methodologies.

In conclusion, the analysis step is a pivotal component of our methodology, enabling the Gated Scene-Specific YOLO model to achieve an optimal balance between detection performance and computational efficiency. Through this process, we refine the model's gating mechanism to suit specific scene characteristics, thereby ensuring high accuracy and reduced resource

consumption in real-world object detection tasks.

4 Experiments and Results

5 Conclusion

Appendices

A

B

Abstract in Korean

References

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[2] C. Li, L. Li, H. Jiang, *et al.*, "Yolov6: A single-stage object detection framework for industrial applications," *arXiv preprint arXiv:2209.02976*, 2022.

[3] C. Li, L. Li, Y. Geng, *et al.*, "Yolov6 v3. 0: A full-scale reloading," *arXiv preprint arXiv:2301.05586*, 2023.

[4] Ł. Kaiser and S. Bengio, "Discrete autoencoders for sequence models," *arXiv preprint arXiv:1801.09797*, 2018.

[5] Z. Chen, Y. Li, S. Bengio, and S. Si, "You look twice: Gaternet for dynamic filter selection in cnns," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9172–9180.

[6] Y. Bengio, "Deep learning of representations: Looking forward," in *International conference on statistical language and speech processing*, Springer, 2013, pp. 1–37.

[7] J. Liu, Z. Xu, R. Shi, R. C. Cheung, and H. K. So, "Dynamic sparse training: Find efficient sparse network from scratch with trainable masked layers," *arXiv preprint arXiv:2005.06870*, 2020.

[8] Z. Zhang, R. Tao, and J. Zhang, "Neural network pruning by gradient descent," *arXiv preprint arXiv:2311.12526*, 2023.

[9] A. Howard, M. Sandler, G. Chu, *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1314–1324.

[10] S. Kalwar, D. Patel, A. Aanegola, K. R. Konda, S. Garg, and K. M. Krishna, "Gdip: Gated differentiable image processing for object detection in adverse conditions," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 7083–7089.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.