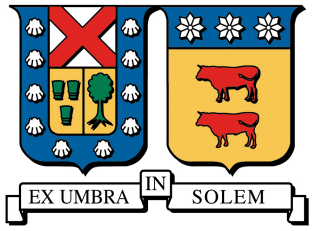
Universidad Técnica Federico Santa María
Departamento de Informática
Santiago, Chile



Design, Implementation, and Application of a Context Aggregation-Enhanced Feature Fusion Approach for Object Detection and Instance Segmentation

Camilo Esteban Núñez Fernández

Memoria para optar al Título de
Ingeniero Civil en Informática

Profesor Guía: Mauricio Solar Fuentes, Ph.D.
Profesor Correferente: Humberto Farias Aroca, Ph.D.

TBA

Dedication

To my mother Katthrinny, my sisters Javiera and Daniela, and my nieces Rafaela and Julieta.

Abstract

Feature fusion in Convolutional Neural Networks (CNNs) plays a critical role in improving the performance of specialized tasks, such as Object Detection and Instance Segmentation. This process integrates information from multiple hierarchical levels within the network to achieve a more comprehensive understanding of complex data. The incorporation of context features enriches the network's perspective, enabling more nuanced decision-making and thereby enhancing its ability to detect objects and segment instances in intricate scenes.

Despite the potential advantages of feature fusion, existing network architectures often exhibit limitations. While they are proficient in leveraging hierarchical data, these architectures frequently overlook the rich contextual information dispersed across different levels. Additionally, many feature fusion modules rely on elementary operations such as addition or concatenation, or employ rigid linear aggregation schemes for feature maps. Such practices can inadvertently impede the effective interplay between global and local context features, thereby constraining the network's capabilities.

To address these shortcomings, this research aims to design, implement, and evaluate a novel feature fusion neck architecture. Named CABiFPN, this architecture is engineered to seamlessly integrate contextual information from various receptive fields, with a particular focus on optimizing object recognition across different scales of feature maps. Our goal is to significantly boost performance in tasks like Object Detection and Instance Segmentation.

To rigorously assess its effectiveness, CABiFPN will be evaluated and validated using the COCO dataset. It will be compared against a range of existing feature extraction and feature fusion methodologies that represent the current state of the art. This comparative analysis is designed to deepen our understanding of how context feature fusion operates within Convolutional Neural Networks, and to determine whether CABiFPN can offer a significant performance boost in specialized tasks like Object Detection and Instance Segmentation.

Keywords: Object Detection, Instance Segmentation, Feature Fusion, Context Feature.

Contents

Abstract ii

Contents iii

1 Introduction 1
- 1.1 Introduction to *Context Features* in CNNs . . . 1
- 1.2 Problem Definition . . . 2
- 1.3 Objectives . . . 3
  - 1.3.1 General Objective . . . 3
  - 1.3.2 Specific Objectives . . . 3
- 1.4 Work Overview . . . 4

2 Background 5
- 2.1 Convolutional Neural Networks . . . 5
  - 2.1.1 Local Receptive Fields . . . 6
  - 2.1.2 Shared Weights . . . 7
  - 2.1.3 Spatial or Temporal Subsampling . . . 8
- 2.2 Computer Vision Tasks: Instance Segmentation and Object Detection . 9
  - 2.2.1 Object Detection . . . 9
  - 2.2.2 Instance Segmentation . . . 10
- 2.3 Metrics and CNN Explanation . . . 10
  - 2.3.1 Metrics for Object Detection and Instance Segmentation . . . 10
  - 2.3.2 CNN Model Explanation . . . 11

3 Related Work 15
- 3.1 Backbones . . . 16
  - 3.1.1 InternImage . . . 16
  - 3.1.2 ConvNeXt . . . 17
  - 3.1.3 EfficientNetV2 . . . 18
- 3.2 Necks . . . 20
  - 3.2.1 FPN . . . 20
  - 3.2.2 PANet . . . 21
  - 3.2.3 NAS-FPN . . . 22

3.2.4 BiFPN . . . 24
3.3 Object Instance Segmentation . . . 26
3.3.1 R-CNN . . . 26
3.3.2 Fast R-CNN . . . 27
3.3.3 Faster R-CNN . . . 29
3.3.4 Mask R-CNN . . . 31

4 Proposal **33**
4.1 Context Aggregator Units . . . 34
4.1.1 Global Context Operator . . . 34
4.1.2 Local Context Operator . . . 35
4.1.3 FMBConvCA Block . . . 36
4.2 Architecture of the CA-BIFPN Neck . . . 37

5 Results **39**
5.1 Experiments Definitions . . . 39
5.1.1 Hardware Configuration . . . 39
5.1.2 Specifications of Computational Environment . . . 39
5.1.3 Training Specifications . . . 40
5.2 Metric Evaluation Results . . . 43
5.3 Ablation CAM Results . . . 43
5.4 Imputation Results . . . 52

6 Discussion and Analysis **54**
6.1 Metric Evaluation Analysis . . . 54
6.2 Explanation with Ablation CAM . . . 54
6.2.1 Ablation CAM visual explanation Image `509403` . . . 55
6.2.2 Ablation CAM visual explanation Image `96825` . . . 56
6.2.3 Ablation CAM visual explanation Image `171382` . . . 57
6.3 Validation of Visual Explanations through Imputation Techniques . . . 58

7 Conclusion **60**
7.1 Future Work . . . 61

Bibliography **62**

Appendix A Supplementary Figures and Tables **70**
A.1 Predictions from models . . . 70
A.2 `Ablation CAM` over models . . . 73
A.3 Pixel Removal with `ROAD` strategy . . . 86

List of Figures **88**

List of Tables **91**

Chapter 1

Introduction

Over the past decade, there has been a marked escalation in both the development and application of deep learning methodologies across diverse disciplines, ranging from computer vision to natural language processing [LBH15]. A particularly noteworthy strategy that has been the focal point of significant academic and industrial attention within this domain is **feature fusion** [GLJ18]. Conceptually, feature fusion is the integration of information stemming from disparate sources or hierarchical levels, invariably aiming to augment the efficacy of a deep neural network [LAJ15].

In the specialized domain of computer vision, Convolutional Neural Networks (CNNs) have emerged as the preeminent mechanism for feature extraction [KSH12]. These networks are architected with multiple layers designed to discern hierarchical patterns inherent in visual data. Progressing through the layers of a CNN, one observes a gradual transition in feature complexity: initial layers typically capture rudimentary patterns, whereas deeper layers encapsulate more intricate, abstract representations of the data [ZF13].

The doctrine of feature fusion becomes instrumental within the context of CNNs [UL19]. By strategically merging features across varying layers, CNNs are adept at synthesizing representations that concurrently encapsulate both the fine-grained details discerned by the preliminary layers and the higher-order abstractions identified by the deeper strata [HLVDMW17]. This intricate melding ensures that during computational tasks, such as image classification or object detection, the models are equipped to factor in both the localized nuances and overarching patterns, thereby enhancing their predictive accuracy and robustness [Gir15].

1.1 Introduction to *Context Features* in CNNs

Context features encompass the auxiliary information that offers an expanded comprehension of the primary data features. Distinctly, primary data features are concerned

with immediate observable patterns. On the other hand, context features shed light on the encompassing details, thus presenting a more comprehensive perspective. The incorporation of such context allows models to arrive at more nuanced decisions. Taking image recognition as a case in point: while the primary feature might revolve around the shape of an object, the context feature delves into adjacent objects or the backdrop. This, in turn, can provide invaluable hints regarding the identity of the primary object.

As highlighted earlier in this chapter, Convolutional Neural Networks stand out as the zenith of feature extraction methods. Predominantly applied in image processing, CNNs have an innate capability to weave in context features due to their hierarchical architecture. This layered structure of CNNs is not merely a design, it's a deliberate strategy. While the initial layers grapple with basic features like edges, the deeper layers venture into the territory of intricate patterns, sometimes discerning whole objects [ZF13].

1.2 Problem Definition

The introduction of the Features Pyramid Networks (FPN) method in the realm of computer vision has precipitated a notable shift towards harnessing feature extraction through comprehensive pre-trained networks [LDG+17]. Concomitant to this trend is the increasing ubiquity of feature fusion within the segment of these architectures known as the **neck**. This growing trend is underpinned by the fusion capabilities inherent to the neck, where an amalgamation of both low-level and high-level features, drawn from the multi-tiered structure of the primary network, or **backbone**, takes place.

In light of this evolution, there has been a proliferation of architectural designs aiming to augment and hone the foundational tenets and mechanisms of the FPN. Prominent exemplars include the PANet [LQQ+18], NAS-FPN [GLPL19], and BiFPN [TPL19]. Such advancements echo the seminal influence of FPN in catalyzing multi-level feature fusion.

Yet, a discerning examination reveals a lacuna in these sophisticated architectures: *while they are adept at capitalizing on hierarchical levels for feature fusion, they frequently neglect the rich contextual information embedded in the input.* This context, which is palpable across all fusion strata, intimates a potential oversight of invaluable scene-level semantic cues. Furthermore, a significant portion of the feature fusion modules in the neck, often resort to elementary operations such as addition or concatenation [LDG+17, LQQ+18], or adhere to rigid linear aggregation schemas for feature maps [GLPL19, TPL19]. Such practices can inadvertently stymie the organic interplay between global and local context features.

The aim of this work is to meticulously design, implement, and evaluate the efficacy

of aggregating contextual information from disparate receptive fields tailored to objects of diverse scales. Specifically, we intend to ascertain whether such integration can enhance the feature fusion process across the multi-tiered hierarchy inherent in a neck architecture. This endeavor seeks not only to elucidate the potential benefits of this approach but also to provide a meaningful contribution to the ongoing discourse on optimizing neck structures within advanced architectural designs.

1.3 Objectives

1.3.1 General Objective

The general objective of this research is to design, implement, and evaluate a feature fusion neck architecture; this architecture integrates contextual information sourced from multiple receptive fields, thereby ensuring optimal recognition of objects across disparate scales of feature maps, and is especially tailored to enhance performance in tasks such as Object Detection and Instance Segmentation.

1.3.2 Specific Objectives

The specific objectives delineated for this research endeavor are as follows:

1. Determine the most suitable neck architecture capable of proficient multi-scale feature fusion.
2. Ascertain the ideal operator block for feature fusion, utilizing contextual information from diverse receptive fields.
3. Develop a neck architecture incorporating the previously identified operator block.
4. Construct a comprehensive, end-to-end model comprised of a backbone, neck, and the detection head using *Mask R-CNN*.
5. Implement the aforementioned end-to-end model using *Pytorch*, optimized for GPU acceleration.
6. Appraise the performance of the end-to-end model when paired with potential backbones such as InternImage, ConvNeXt, and EfficientNetV2.
7. Assess the efficacy of the end-to-end model in conjunction with potential necks, specifically the *BiFPN* and the identified neck architecture.

1.4 Work Overview

The organization of this document is systematically outlined as follows:

- **Introduction**: This section introduces the research, elucidates the underlying motivation, and defines the specific problem under investigation. It sets the context for the entire study by emphasizing the research's significance and its pertinence to contemporary discussions.
- **Background**: This section delves into the foundational knowledge that informs the research. It explicates key concepts and principles associated with Convolutional Neural Networks, provides an overview of Instance Segmentation and Object Detection tasks, and sets forth the metrics and explanation methods employed to assess the proposed architecture.
- **Related Work**: This section offers a review of prior research and studies pertinent to the subject matter. In particular, it elaborates on the utilization of backbones and necks in the journey to conceptualize the proposed architecture.
- **Proposal**: This segment presents the proposed **CA-BiFPN** architecture, detailing its design and implementation.
- **Results and Discussion**: This section furnishes the research outcomes, specifically focusing on the metrics and explanation method results. An extensive discussion follows, where the findings are scrutinized, juxtaposed, and interpreted, elucidating their broader implications and relevance.
- **Conclusion**: This concluding section encapsulates the principal insights derived from the research, underscores the salience of the findings, and contemplates potential trajectories for future research in this domain.

Chapter 2

Background

This chapter presents a comprehensive theoretical overview of the key Artificial Neural Network used in this study: **Convolutional Neural Networks**. It also outlines the two main tasks covered in this document: **Instance Segmentation** and **Object Detection**. Finally, the chapter will provide an extensive review of performance metrics and visual explanations.

2.1 Convolutional Neural Networks

To recap our understanding from Feed Forward Networks, each neuron unit is represented and connected in a continuous chain-based architecture. Each layer is described by the following equation:

$$\boldsymbol{h} = f\left(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}\right) \tag{2.1}$$

where $\boldsymbol{h}$ denotes the output of the layer, f is the activation function, $\boldsymbol{W}$ represents the weight vector, $\boldsymbol{b}$ is the bias vector, and $\boldsymbol{x}$ is the input vector of real values such that $\boldsymbol{x} \in \mathbb{R}^n$. In this formulation, the dimensionality $\mathbb{R}^n$ of $\boldsymbol{x}$ confines the input to a one-dimensional vector. Given this constraint, processing images in computer vision tasks using Feed Forward Networks leads to substantial computational complexity in matrix multiplication and results in vectors with high dimensionality of parameters.

Convolutional Neural Networks, as introduced by *Kunihiko Fukushima* [Fuk80], aim to address the challenge of dimensionality by utilizing a grid-like topology [GBC16]. *Le Cun et al.* further developed and elaborated on CNNs in their research [LBD+89, LBBH98]. They defined a Convolutional Neural Network based on three pivotal concepts: (I) local receptive fields, (II) shared weights, and (III) spatial or temporal subsampling [LBBH98].

2.1.1 Local Receptive Fields

As mentioned in the previous section, Feed Forward Networks consist of fully connected layers. The interaction is facilitated through connections between each neuron in the input layer and the corresponding neuron in the output layer, as illustrated in Figure 2.1.

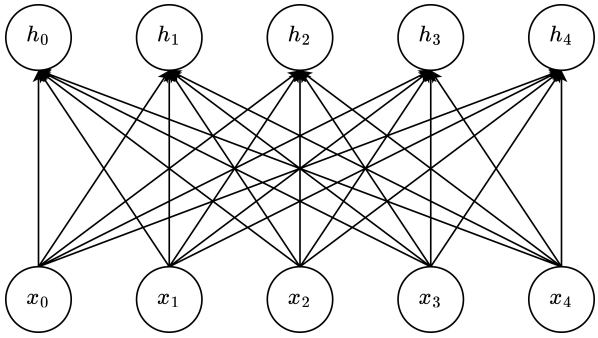


Figure 2.1: Fully connected layer. Given an input $\boldsymbol{x} \in \mathbb{R}^5 : \{x_i \in \boldsymbol{x}, i = 0, 1, ..., 4\}$, the hidden layer $\boldsymbol{h}$ is defined following the equation (2.1).

In CNNs, the connection is often restricted to a small neighborhood within the layer, as depicted in Figure 2.2. This limited neighborhood is termed the 'receptive field' and is employed in the convolution operation to compute sparse interactions in the input layer.

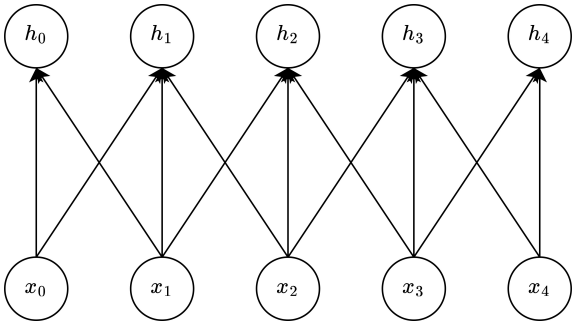


Figure 2.2: Convolutional Layer using neighborhoods: A variation of Figure 2.1

The standard convolutional operation can be represented as:

$$\boldsymbol{S}(t) = (\boldsymbol{x} * \boldsymbol{y})(t)$$

However, this function, being continuous in t, is not directly applicable in a discrete setting. To address this, a discrete form of the convolution operation can be adopted [Arn11]:

$$\boldsymbol{S}[t] = \boldsymbol{x}[t] * \boldsymbol{y}[t] = \sum_{k=-\infty}^{\infty} \boldsymbol{x}[k] \cdot \boldsymbol{y}[n-k]. \tag{2.2}$$

Here, Equation (2.2) alludes to a 1D convolutional operation, indicating that both $\boldsymbol{x}$ and $\boldsymbol{y}$ are 1D. For 2D operations [Arn11], the equation takes the form:

$$\boldsymbol{S}[m,n] = \boldsymbol{x}[m,n] * \boldsymbol{y}[m,n] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} \boldsymbol{x}[i,j] \cdot \boldsymbol{y}[m-i, n-j] \tag{2.3}$$

Equation (2.3) correlates directly with the receptive field concept: in this context, $\boldsymbol{x}$ represents the input, $\boldsymbol{y}$ is the kernel, and $\boldsymbol{S}$ denotes the feature map. Within this framework, connections remain localized within the 2D space of both the input vector and the kernel, as demonstrated in Figure 2.3.

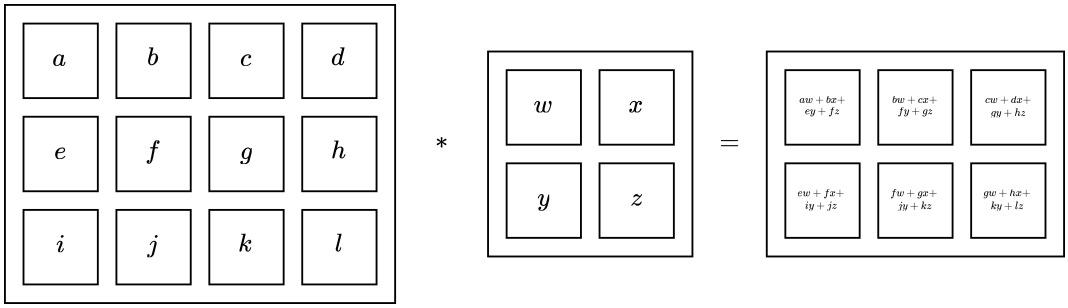


Figure 2.3: Example of a convolutional operation involving an input $\boldsymbol{x}$ and a kernel $\boldsymbol{y}$. Adapted image sourced from [GBC16].

2.1.2 Shared Weights

In CNNs, the convolutional layers use the equation (2.3) to operate over the two dimension. In this process, the kernel $\boldsymbol{y}$ is shared for more than one function in a model [GBC16]. This shared kernel is also know as the filter of the convolution that is convolved with the input. The Figure 2.3, display the a simple plane as input within which all the units share the same set of weights [LBBH98]. An example of this can see in the Figure 2.4.

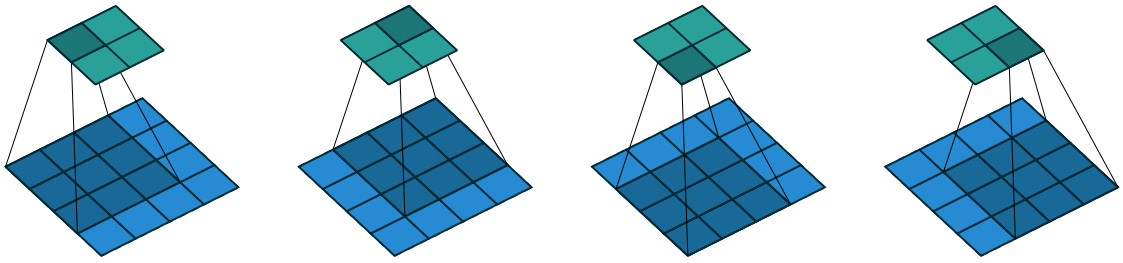


Figure 2.4: Convolving a 3×3 kernel over a 4×4 input using unit strides. Image and caption sourced from [DV16].

Convolutional Layer

Having defined the concepts of **Local Receptive Fields** and **Shared Weights**, we can express the convolutional layer in terms of equation (2.1) as:

$$\boldsymbol{h} = f\left(\boldsymbol{K} * \boldsymbol{x} + \boldsymbol{b}\right), \tag{2.4}$$

where the operator * represents the convolution between the kernel vector $\boldsymbol{K}$ (or the shared weights) and the input vector $\boldsymbol{x}$.

2.1.3 Spatial or Temporal Subsampling

After applying the convolution operation, the positions of the values in the input vector become less relevant in the output vector or feature map. This can be problematic, especially as these positions may vary across different instances [LBBH98]. This issue becomes even more pronounced when multiple convolutions are applied, as commonly occurs in layers. As a result, the extraction of distinctive features might shift in the location of the receptive field. One approach to address this problem is by reducing the precision with which the positions of distinctive features are encoded in a feature map. This reduction is akin to subsampling the output layer and is typically executed after one or more convolutional layers.

Pooling Operation

One operation used to reduce the feature map's dimensionality is the *Pooling* operation, typically applied after one or more convolutional layers. This operation modifies the layer's output by summarizing its values. Consequently, the spatial dimension is reduced, yielding a new representation of the output with fewer parameters and computations.

The two most common pooling operations are:

- *Max Pooling* [ZC88]: This operation selects the maximum value from defined patches of a feature map.
- *Average Pooling* [LBBH98]: This operation computes the average value over specified patches of a feature map.

Both operations result in a downsampled version of the output feature vector, helping to produce a representation that remains relatively invariant to minor shifts in the input vector [GBC16].

2.2 Computer Vision Tasks: Instance Segmentation and Object Detection

Convolutional Neural Networks have been instrumental in the realm of computer vision, initially applied to image classification tasks [LBD+89, LBBH98]. In these tasks, the network classifies images based on elementary visual features extracted, such as corners, edges, and endpoints. These features, processed across multiple layers of the network, are optimized to best represent the image in question.

Emerging from this foundational task, the versatility of CNN-based approaches has expanded to encompass a broader spectrum of tasks like object detection, semantic segmentation, instance segmentation, object tracking, and more [ZCS+23, ZAA+22]. The purpose of this work focuses on two critical tasks: Object Detection and Instance Segmentation.

2.2.1 Object Detection

Object detection is an advancement of the primary task of image classification. While image classification identifies the main subject of the image, object detection aims to classify and recognize multiple objects within the image. These individual objects are referred to as *instances*. The objective of object detection is to design a network that detects all of these instances annotated in an image. In this process, the network, also known as a *detector*, must identify the position of each instance and then assign a coordinate *bounding box* around it.

The foundation of the object detection task wasn't originally built upon the CNN architecture. It began with the pioneering work of *Viola and Jones* [VJ04, VJ01], who employed various techniques such as Haar-like features, integral images, Adaboost, and cascading classifiers [ZAA+22]. Subsequently, *Dalal and Triggs* [DT05] introduced the *Histogram of Oriented Gradients (HOG)* as a feature descriptor. Their detection approach utilized a grid-based decomposition of the image and created a gradient histogram for each grid element. These elements were then classified using a linear SVM.

However, prior models was based in the handcrafted features of the annotations. This problem was solved with the usage of the deep learning models, in more specific, the CNN models, due the ability to extract the high level features of the image. It's possible to identify two type of the CNN-based dectetor: **one-stage and two-stage detector**. In firs case, the two-stage detector, the model use a combination of backbone (in sometimes plus a neck) to extract the high level features and then pass it to a head that do the classify task. In the other hand, the one-stage network use an entire model to extract the feature and do the classify of the object in one pass-through stage.

In this work, we will focus on the two-stage model, which will be further described in Chapter 3.

2.2.2 Instance Segmentation

Much like the previously discussed tasks of classification and object detection, image segmentation leverages the high-level features of an image to classify its pixels based on annotations. This overarching task can be further subdivided, with the most prominent and those explored in this work being semantic segmentation and instance segmentation [MBP+22, GBK22]. Semantic segmentation classifies each pixel of the image into object categories such as cars, people, or traffic signs. Conversely, instance segmentation delves deeper, identifying and segmenting pixels corresponding to individual instances present in the image, distinguishing between each person, each car, or each traffic sign.

The concept of instance segmentation was pioneered by *Hariharan et al.* in their pioneer work titled *Simultaneous Detection and Segmentation* [HAGM14]. Here, a two-stage network was utilized for both object detection and instance segmentation. Following this groundbreaking work, several state-of-the-art instance segmentation architectures emerged, utilizing the two-stage approach. One such architecture will be elaborated upon in Chapter 3.

2.3 Metrics and CNN Explanation

To evaluated the performance of the models, it is necessary to summarise the principal metric commonly used for Object Detection and Instance Segmentation. On the other hand, in addition to performance metric of the models, we will define a strategy to evaluate the confidence of the proposal models and their respective variations, using visual and numerical evidence at the moment to compere these different models with each others.

2.3.1 Metrics for Object Detection and Instance Segmentation

Both Object Detection and Instance Segmentation metrics are based in the same metric reasoning: **where** and **what** a correct or incorrect area or object the model predicted. The most common way to measure the **where part**, is using the **Intersection over Union (IoU)**, or the **Jaccard Index** [Jac12, Jac01], and is define as the measure of the area of intersection between the predicted segmentation mask A (or the predicted bound box in the Object Detection) and the ground truth map B (or the ground truth bound box in the Object Detection), divided by the area of the union between them, and is defined as:

$$\text{IoU} = J(A, B) = \frac{||A \cap B||}{||A \cup B||}, \tag{2.5}$$

where $||\cdot||$ detonated the area of the set, and $J(A, B)$ the **Jaccard Index**, that is also in the range 0 and 1.

The most common way to measure the ***what part*** a correct or incorrect area or object the model predicted from the beginning above definition, is using the classical terms *True positive (TP), False positive (FP) and False negative (FN).* Also, the *True Negative (TN)* result are not used in the most command metrics, because indicate anywhere position where the model do not predict a mask or object and their respective annotation did not provide a mask or object; then the TN are vast and unnecessary to calculate. Whit this concepts, it is possible to define for each class, the **Precision** and **Recall** measure as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{2.6}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{2.7}$$

Whit this two measure, it is possible to define the **Precision-Recall (PR) curve**, where it is show the relationship between Recall on the x-axis and Precision on the y-axis. In this case, the simple form to define a discrete definitions of this is using the **Intersection over Union** as a threshold for the points in the Precision-Recall (PR) curve:

$$PR(A, B) = \sum_{i=0}^{|A|} \sum_{j=0}^{|B|} PR(A_i, B_j) \cdot I\left[\text{IoU}(A_i, B_i) > \alpha\right], \tag{2.8}$$

where $|\cdot|$ detonated the length of the set, $PR(A_i, B_j$ represent the point of the Precision-Recall (PR) curve for each element of the set A and B, $I\,[]$ the indicator function, and α the IoU threshold. Then, it's possible to define the **mean Average Precision (AP)** as the area under the Precision-Recall (PR) curve for each N classes as:

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} ||PR_i||. \tag{2.9}$$

2.3.2 CNN Model Explanation

In the realm of designing and implementing Convolutional Neural Networks, deeper networks aim to achieve more complex representations of inputs compared to their shallower counterparts [Ben09]. Specifically, for tasks in image processing or computer vision, such intricate representations enable enhanced hierarchical depictions of images [ZF13]. This is attributable to how the multiple layers in the network effectively capture pertinent information. However, these deeper networks often present challenges in interpretability, as their predictions can be difficult to elucidate due to limited insight into their internal operations [JO21].

To address this challenge, it is possible to employ advanced methods that offer improved interpretability and explanations of model performance. This facilitates a more robust evaluation when comparing different models. In this study, we aim to provide clearer insights into the behaviors of the proposed and tested models by utilizing a visual ***class-discriminative location technique*** and a novel *Remove and Debias* metric technique.

Explanation model using Ablation CAM

Visual explanations detailing how a model captures features across its layers can offer valuable insights into the model's internal behaviors. A novel avenue in this domain is the suite of **Class Activation Mapping (CAM)** based methods [ZKL$^+$15, SDV$^+$16, CSHB17, FHD$^+$20, WDYZ19, DR20]. These techniques produce a **visual explanation map** by retaining spatial information throughout convolutional layers and underscoring the significance of each neuron for specific decision regions.

Drawing inspiration from the pioneering work of *Bolei Zhou et al.* [ZKL$^+$15], the **visual explanation map**, given a model prediction f for a target class c of an input image x, can be formulated as:

$$L^c_{\text{CAM}}(A) = \text{ReLU}\left(\sum_{k=1}^{N_l} w^c_k A_k\right), \tag{2.10}$$

where $A = f^{[l]}(x)$ denotes the output of the l-th layer, A_k represents the k-th activation map of A, and w^c_k is the k-th weight corresponding to class c. The symbol N_l designates the count of activation maps in the l-th layer. Within the CAM paradigm, the coefficient w^c_k indicates the significance of A_k [ZKL$^+$15]. Applying the ReLU activation to the linear combination filters only the positive map values, thereby pinpointing pixels influencing the target class c.

In this research, we adopt the approach described by *Saurabh Desai et al.* in **Ablation CAM** [DR20]. This method is favored to avoid the gradient saturation issues faced by gradient-weighted strategies like *Grad-CAM* [SDV$^+$16]. Gradient saturation in these techniques leads to diminishing backpropagation gradients, consequently causing visualizations to falter in identifying pertinent image regions [DR20, FHD$^+$20].

Ablation CAM introduces a procedure wherein, for a given input image x, a first forward pass through the model yields a non-linear function termed the **class activation score** y^c for class c, derived from the activation map A_k of the final convolutional layer. In a subsequent forward pass with the same input image x, individual activation cell values of the activation map A_k are set to zero. This leads to the ablation of the k-th unit, generating a new **class activation score** y^c_k to serve as a baseline for the

activation map A_k. The slope produced by this unit's ablation is defined as:

$$\text{slope } = \frac{y^c - y_k^c}{\|A_k\|}. \tag{2.11}$$

Ablation CAM utilizes a modified version of this *slope* to avoid small values that arise when the norm $\|A_k\|$ is significantly larger than the difference $y^c - y_k^c$. This modification defines the coefficient w_k^c in equation (2.10) as:

$$w_k^c = \frac{y^c - y_k^c}{y^c}, \tag{2.12}$$

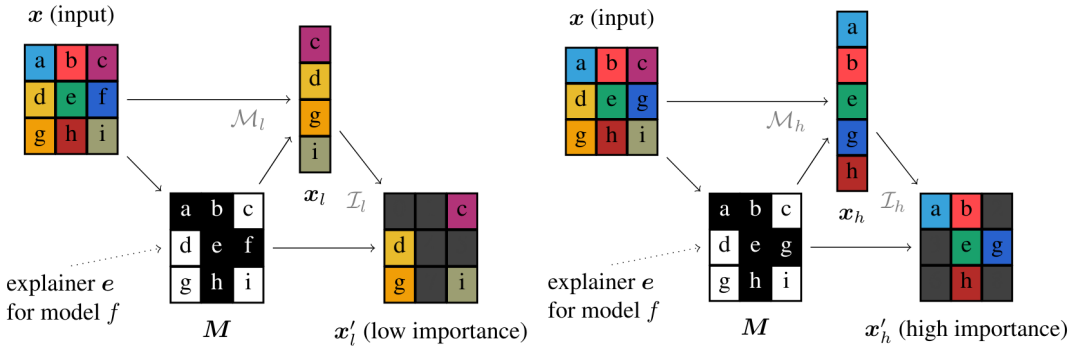
and this can be interpreted as the relative decrease in class activation score c upon the removal of activation map A_k [DR20].

Validate the explanation using imputation

A common method for validating explanation strategies is by perturbing the pixels of an image based on the importance assigned by these strategies. Specifically, the perturbation is determined by whether the pixel carries more or less critical information. Using this method, it is possible to measure the fidelity of the explanations [THC+19]. This validation technique is often referred to as pixel-perturbation strategies, where a pixel is removed and replaced with an imputed value [SBM+15, THC+19]. In our research, we employed a strategy named **ROAD (RemOve And Debias)** as proposed by *Yao Rong et al.* in their study [RLB+22].

The *ROAD* strategy outlines two orders for pixel removal: **MoRF** (Most Relevant First) and **LeRF** (Least Relevant First) [SBM+15]. In the **MoRF** approach, the k **most important features** of an instance are prioritized for removal. Given an explanation characterized by a choice of features through a binary mask, $\boldsymbol{M} = \boldsymbol{e}_k(f, \boldsymbol{x}) \in \{0,1\}^d$, a value is set to one if the corresponding feature ranks within the top-k, and zero otherwise. The selection operator for the **least important dimensions**, denoted as $\mathcal{M}_l : \{0,1\}^d \times \mathbb{R}^d \to \mathbb{R}^{d-k}$, extracts a vector, $\boldsymbol{x}_l$, that retains only the remaining features, preserving their order based on their original input indices. The modified output of these features is expressed as $\boldsymbol{x}'_l = \mathcal{I}_l(\boldsymbol{M}, \boldsymbol{x}_l)$, where $\mathcal{I}_l : \{0,1\}^d \times \mathbb{R}^{d-k} \to \mathbb{R}^d$ is an imputation operator. This operator repositions all inputs from $\boldsymbol{x}_l$ to their original places, with the remaining set to a fill value, discarding the top-k features. This procedure is illustrated in Figure 2.5a. For the **LeRF** approach, the k **least significant features** per instance are eliminated, with the selection operator defined for the **most important dimensions** as shown in Figure 2.5b.

The *ROAD* strategy introduces an innovative imputation operator, $\mathcal{I}_l$, termed **Noisy Linear Imputation**. This operator creates imputations that are not readily discernible, yielding a reduced $I(\boldsymbol{x}'_l; \boldsymbol{M})$. It approximates each pixel using a weighted mean of its neighbors, and a slight random noise is incorporated to ensure the model cannot easily learn the linear dependency.



(a) **MoRF** (Most Relevant First) order. (b) **LeRF** (Least Relevant First) order.

Figure 2.5: **MoRF** (Most Relevant First) and **LeRF** (Least Relevant First) pixel removal sequences utilized by the **ROAD** approach. Images sourced from [RLB+22].

Chapter 3

Related Work

Based on the background provided in Section 2.2, the extraction of semantic fractures in end-to-end detectors, such as the **one-stage detector** and **two-stage detector**, is accomplished using a *backbone*. This is followed by an attached *neck* that fuses features from various layers of the backbone, as depicted in Figure 3.1. In this chapter, we will delve into the combined use of the backbone and neck in one of the state-of-the-art **two-stage detector**, *Mask R-CNN* [HGDG17].

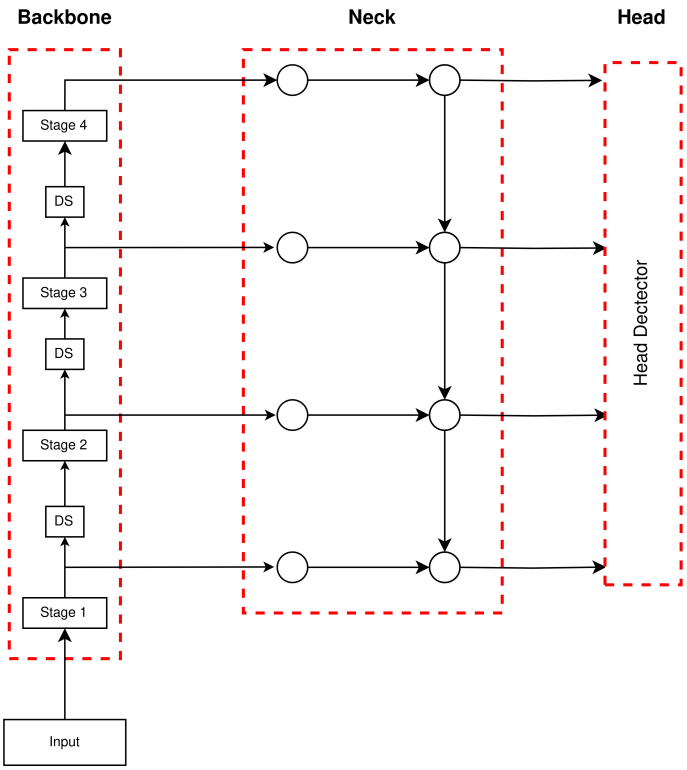


Figure 3.1: Diagram illustrating the architecture of an end-to-end detector, featuring a *backbone*, a *neck*, and a *head* for detection functions.

3.1 Backbones

Feature extraction from images serves as the foundational step in deep learning networks tailored for computer vision tasks, as elaborated in the previous chapter. This ability of the network to extract features can be generalized to various tasks through transfer learning [YCBL14]. This creates a robust network that can be adapted for multiple tasks without the need for retraining every time. Given this, a **backbone** can be defined as a pre-trained network dedicated to feature extraction and has been fine-tuned across multiple tasks prior to its designated use.

For the scope of this work, we will employ three cutting-edge backbones: InternImage [WDC+23], ConvNeXt [LMW+22], and EfficientNetV2 [TL21].

3.1.1 InternImage

Wenhai Wang et al. introduced their model, *InternImage* [WDC+23], designed to efficiently scale with larger parameter sizes and learn potent representations from vast training data. This approach achieved representations that are comparable to, if not better than, large-scale *ViTs* [DBK+21, LLC+21, WXL+22, ZKHB22].

In transformer-based architectures, the global attention mechanism employed by *ViT* [DBK+21] often incurs high computational and memory overheads, particularly with extensive feature maps. This burden restricts its utility in many downstream applications.

Addressing this issue, *InternImage* aims to resolve the challenges of long-range dependencies and adaptive spatial aggregation inherent to the multi-head self attention (MHSA) [VSP+23] operator frequently found in transformer-based architectures. Their solution harnesses an advanced version of the Deformable Convolution v2 (DCNv2) [ZHLD18] operator named DCNv3. This enhanced operator compensates for the limitations of standard convolution in handling long-range dependencies and adaptive spatial aggregation. Furthermore, it retains the inductive bias of traditional convolution and employs sparse sampling, proving to be more computationally memory-efficient compared to operators like *MHSA*.

The *InternImage* architecture comprises a series of stacked basic blocks, complemented by a stem and down-sampling layers, as illustrated in Figure 3.2. These basic blocks incorporate Layer Normalization [BKH16], a feed-forward network (akin to the one used in transformer-based architectures [VSP+23]), and are activated using GELU [HG23]. To generate hierarchical feature maps, the architecture employs a convolutional stem and down-sampling layers, which resize the feature maps across various scales.

For the purposes of this study, we will employ the *InternImage-S* backbone, which consists of approximately $50M$ parameters. This model has been trained on the classi-

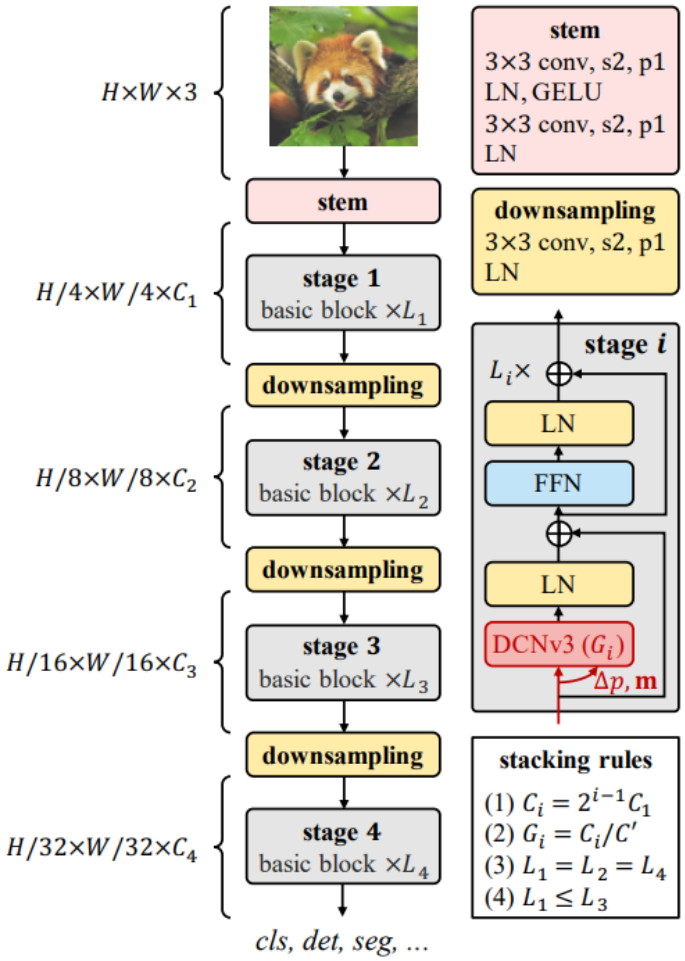


Figure 3.2: A schematic representation of the *InternImage* architecture detailing its four distinct stages. Image adapted from [WDC+23].

fication task using the *ImageNet-1K* dataset [DDS+09] for 300 epochs.

3.1.2 ConvNeXt

Zhuang Liu et al. undertook a thorough reassessment of the Convolutional Neural Network's design landscape in their pioneer work, introducing the ConvNeXt architecture [LMW+22]. Within this exploration, the authors meticulously examined the architectural differences that distinguish convolutional models from transformer-based counterparts. Stemming from their investigation, they presented the ConvNeXt, a novel *family of pure ConvNets*.

The ConvNeXt architectures share similarities at the micro level with transformer-based models such as *DeiT* [TCD+21] and *Swin* [LLC+21]. Common features include

the GELU activation [HG23], the integration of Layer Normalization [BKH16], and distinct down-sampling layers dedicated to feature resolution adjustments. On the other hand, when observed from a macro perspective, the ConvNeXt architectures draw parallels with the *ResNeXt* [XGD+17] structure. They predominantly employ techniques like grouped convolution and depth-wise convolution as the central operations within their stage blocks, a design which is visually detailed in Figure 3.3.

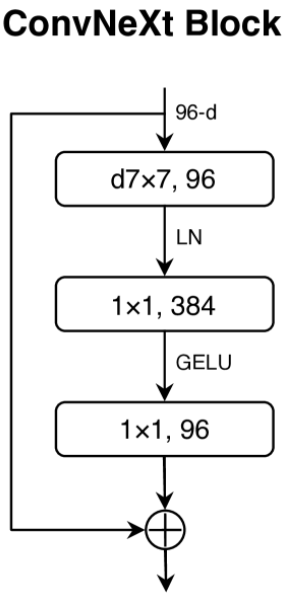


Figure 3.3: Block design for a *ConvNeXt*. Image adapted from [WDC+23].

In the context of our current research, we have opted to utilize the *ConvNeXt-S* backbone, a robust model encompassing roughly $50M$ parameters. Prior to its deployment, the model underwent an intensive pre-training phase focused on the classification task, leveraging the expansive *ImageNet-21K* dataset [DDS+09]. This pre-training lasted for 90 epochs. Subsequently, to further refine its capabilities, it was trained on the *ImageNet-1K* dataset for 300 epochs.

3.1.3 EfficientNetV2

In the domain of deep learning models, retraining often poses significant challenges, especially given the substantial processing resources and extended training durations necessitated by larger architectures such as *GPT-3* [BMR+20]. In response to these challenges, *Mingxing Tan and Quoc V. Le* introduced a novel solution in their research work [TL21]. They presented an architecture named *EfficientNetV2*. This architecture is notable for its enhanced method of progressive learning, which can adaptively adjust both the regularization parameters and image size during the training process. This

approach optimizes computational efficiency while also maintaining or enhancing model performance.

The *EfficientNetV2* architecture emerges as a refined iteration of the earlier *EfficientNet* [TL19] family of models. One of its key achievements is that it manages to boost the training speed without compromising on parameter efficiency. These enhancements primarily root in three strategic modifications. First, the architecture employs a technique of *training with progressive adjustment of image sizes and regularization*. Second, in its early stages, the architecture replaces the traditional *depth-wise convolutions operator* with a more efficient alternative. Lastly, the *non-uniform scaling strategy* is employed, allowing for the incremental addition of layers in the model's later stages, facilitating a more balanced distribution of complexity. A standout improvement in the *EfficientNetV2* architecture pertains to the convolutional operations. It swaps out the depth-wise convolution with a kernel of 3×3 and the expansion convolution with a kernel of 1×1 found in the *MBConv* [SHZ+18, TL19] block. Instead, it introduces a single regular convolution with a kernel size of 3×3. This modification is vividly illustrated in Figure 3.4. Furthermore, during the early stages, specifically stages 1 to 3, the conventional *MBConv* block is replaced by the more optimized *Fused-MBConv* [GT].

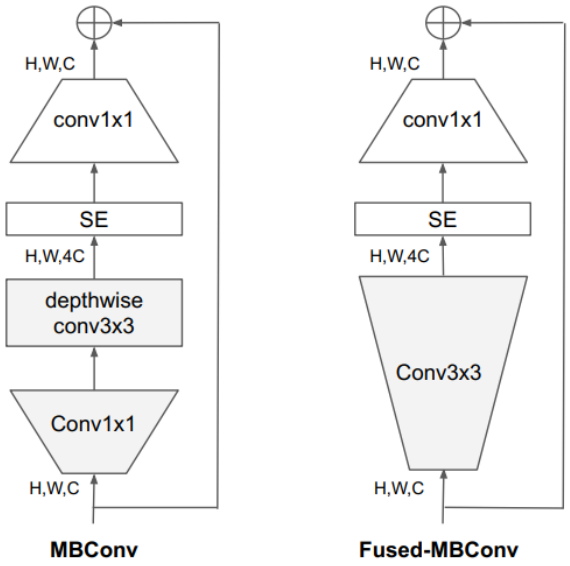


Figure 3.4: Diagram illustrating the architecture of MBConv and Fused-MBConv, as presented in [TL21].

In this study, we will utilize the *EfficientNetV2-M* backbone, which comprises roughly $54M$ parameters. This model underwent training for a classification task using the *ImageNet-1K* dataset [DDS+09] over 350 epochs. Subsequently, it was retrained on the *ImageNet-21K* dataset [DDS+09] for an additional 350 epochs.

3.2 Necks

As illustrated in Figure 3.1, the *neck* model plays a crucial role in aggregating features derived from the *backbone*. Its primary function is to gather feature maps from different stages of the backbone, facilitating interaction between high-level and low-level features. For this work, we will explore two types of *neck* models: the state-of-the-art *BiFPN* [TPL19] and a variant we introduce, the *CA-BiFPN*, which will be detailed in the Chapter 4. Before delving into the *BiFPN* [TPL19] model, it's essential to understand the pioneering neck models that laid the foundation for the *BiFPN* architecture: namely, the *FPN* [LDG+17], *PANet* [LQQ+18], and *NAS-FPN* [GLPL19].

3.2.1 FPN

The prior use of featurized image pyramids [AAB+84] in object detection was aimed at leveraging their scale-invariance. This unique property allows a model to detect objects over a vast scale range by examining in the pyramid levels. As a result, the detector can generate a multi-scale feature representation where every level, possesses robust semantic value. However, employing this multi-scale feature without an appropriate method of feature fusion can lead to significant semantic discrepancies due to varied depths. To address this challenge, *Tsung-Yi Lin et al.* introduced [LDG+17] an architecture that adeptly merges low-resolution features rich in semantics with high-resolution features that are semantically weaker. This is achieved through a top-down pathway combined with lateral connections.

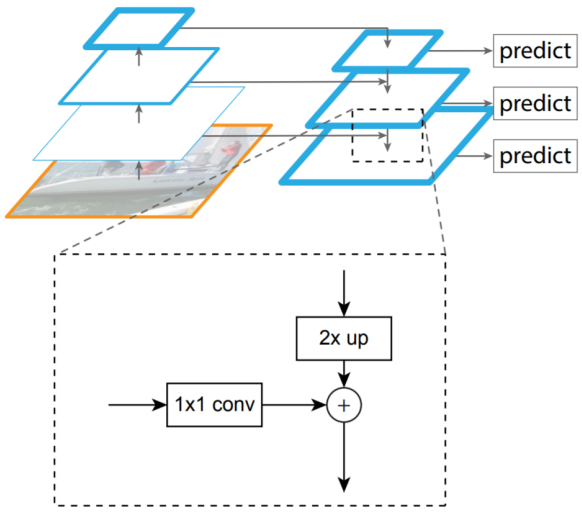


Figure 3.5: Diagram illustrating the architecture FPN, as presented in [LDG+17].

The architecture proposed, named *FPN* by its creators, incorporates two primary strategies: the Bottom-up pathway and the Top-down pathway with lateral connections. In the Bottom-up pathway, the network leverages feature maps extracted from various stages of the backbone. For each of these stages, the network designates a distinct pyramid level. This means that as one progresses through the stages of the backbone, different pyramid levels are determined, capturing features at various resolutions and scales. Conversely, in the Top-down pathway, each pyramid level undergoes upsampling with a spatial resolution factor of 2. This upsampling is then integrated with the lateral connections originating from the Bottom-up pathway. These lateral connections play a pivotal role in bridging the semantic gap between the features. Specifically, each lateral connection seamlessly combines feature maps, all of the same spatial dimensions, from both the Bottom-up and Top-down pathways. This combination is achieved through element-wise addition. A visual representation of this intricate process can be observed in Figure 3.5.

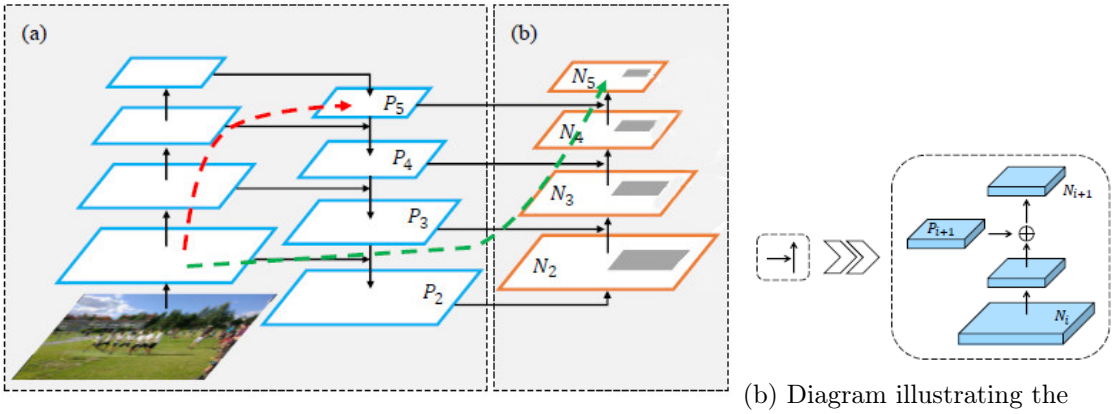
By intricately merging the Bottom-up and Top-down pathways through the use of lateral connections, the *FPN* architecture achieves a seamless integration of feature maps. These maps, which come from diverse scales and possess varying semantic strengths, are blended harmoniously. This unique integration technique provides the *FPN* with the capability for robust multi-scale object detection. After this fusion process, the various pyramid levels are then channeled into the head detector. Regardless of their original dimensions or scales, these feature maps are standardized to a fixed feature dimension in the final layer.

3.2.2 PANet

Building upon the architecture of the Feature Pyramid Network (FPN) [LDG+17], *Shu Liu et al.* extended their research [LQQ+18] to introduce a more sophisticated architecture termed the *Path Aggregation Network (PANet)*. They pinpointed a significant challenge in the foundational FPN design: a lengthened pathway from the basic structural elements to the most advanced features. This elongated route hinders the extraction of detailed localization information. Notably, accessing features at the foundational levels, which are crucial for pinpointing larger instances, becomes increasingly difficult.

In tackling the challenge, the *Bottom-up Path Augmentation* method is adopted. This method amplifies the entire feature hierarchy's localization ability by highlighting the dominant responses of foundational patterns, as show in Figure 3.6a. Within the *FPN*, feature maps P_i with consistent spatial sizes are utilized. The *PANet* structure then designs a segment. This segment integrates a detailed feature map N_i with a comparatively coarser map P_{i+1} using lateral pathways, resulting in the updated feature map N_{i+1}. Each of these maps first passes through a convolution with a kernel 3×3 and

a stride of 2, leading to a decrease in its spatial dimensionality. After this stage, elements of the feature map P_{i+1} are fused with the reduced map using lateral pathways. This integrated feature representation is then processed through an additional convolution with a kernel 3×3, giving rise to N_{i+1} for the following sub-network layers, as illustrated in Figure 3.6b.



(a) Illustration of the PANet Architecture: (a.a) The foundational FPN backbone; (a.b) The Bottom-up path augmentation.

(b) Diagram illustrating the PANet architecture for bottom-up path augmentation block.

Figure 3.6: Diagram illustrating the PANet architecture and its associated proposed fusion feature block, as detailed in [LQQ+18].

3.2.3 NAS-FPN

In the study presented by *Golnaz Ghiasi et al.*, they introduced a novel architecture termed *NAS-FPN* [GLPL19]. This architecture leverages an extensive search space to discern the optimal arrangement of unit merge routes. The technique they adopted is based on the Neural Architecture Search (NAS) methodology [ZL16], which was previously proposed by *Zoph et al.* [ZVSL17]. A distinctive feature of this algorithm is its capability to craft modular architectures. Such modular structures can be efficiently replicated and stacked, culminating in a scalable design. Inspired by this modular concept, *Golnaz Ghiasi et al.* designed a search space adept at churning out scalable architectures, especially those generating pyramidal representations.

The primary objective of the algorithm under discussion is to identify an enhanced FPN (Feature Pyramid Network) architecture suitable for a specific backbone. This relationship is clearly depicted in Figure 3.7. In the realm of FPN, a designated search space is formulated to generate feature pyramid representations. Within this framework, Neural Architecture Search (NAS) deploys a controller. This controller, utilizing reinforcement learning, is tasked with selecting the most optimal model architectures from the defined search space. A significant characteristic of the identified FPN model

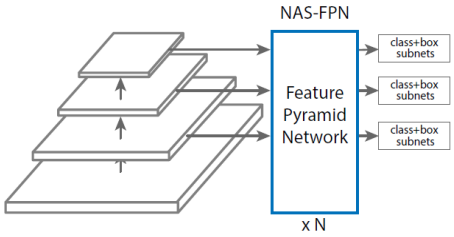


Figure 3.7: Diagram illustrating the interconnection between backbone stages and the NAS-FPN search spaces. The modular nature of the NAS-FPN architecture allows it to be stacked N times, potentially improving its accuracy. A comprehensive description of this stacking mechanism can be found in [GLPL19].

is its inherent scalability. To achieve larger, integrated architectures, the model can replicate itself N times, subsequently merging these instances into a unified, comprehensive structure.

In the NAS-FPN search space, the last layer of each group of feature layers from the backbone serves as the input to the initial pyramid network. Subsequently, the outputs from this first pyramid network become the inputs for the succeeding pyramid network. When considering the input feature within the search space, it is routed through a pyramid network composed of a sequence of merging cells. These cells facilitate cross-scale connections, as depicted in Figure 3.8. The final outputs are then expanded into multiscale feature representations. Notably, this architectural design allows for iterative stacking, which can potentially enhance accuracy.

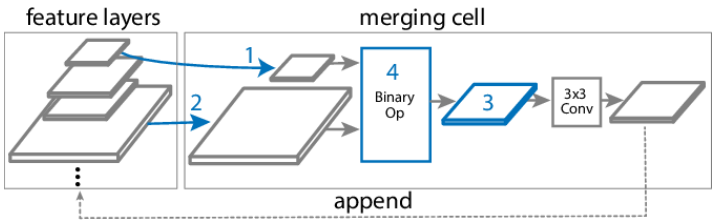


Figure 3.8: Diagram showcasing the merging cell utilized within the search space, further elaborated in [GLPL19].

The process employed by the merging cell within the search spaces can be delineated into four pivotal steps:

(I) Select a feature layer h_i from the available candidates.

(II) Choose another feature layer h_j from the candidates, ensuring no repetition.

(III) Determine the desired output feature resolution.

(IV) Opt for a binary operation, either summation or global pooling, to amalgamate h_i and h_j chosen in steps (I) and (II). This amalgamation produces a feature layer resonating with the resolution specified in step (III).

All these steps are illustratively represented in Figure 3.8.

Figure 3.9 presents the finalized search spaces. Notably, the architecture with the highest performance is represented by the last graph (f) within the figure.

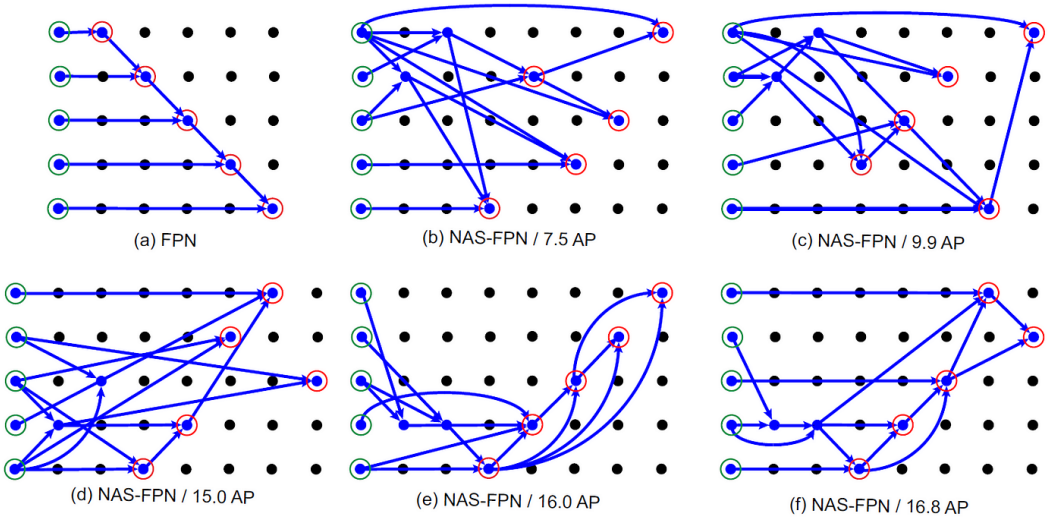


Figure 3.9: The diagram depicts various architecture graphs of NAS-FPN within the search space. Each dot symbolizes a feature layer, with feature layers in the same row sharing a uniform resolution. Arrows delineate connections between internal layers. The initial graph (a) corresponds to the baseline FPN architecture, while graphs (b-f) represent the 7-cell NAS-FPN architectures identified by the NAS controller. Image and caption sourced from [GLPL19].

3.2.4 BiFPN

The imperative nature of devising an architectural framework that offers scalability in detection, while simultaneously ensuring enhanced accuracy and superior efficiency across an expansive range of resource constraints, has driven researchers to seek innovative solutions. In light of this, *Mingxing Tan et al.* introduced the model termed *Bi-Directional Feature Pyramid Network (BiFPN)* [TPL19]. This model endeavors to achieve elevated performance metrics in one-stage detectors. It does so by harnessing the capabilities of an efficient multi-scale feature fusion coupled with learnable weights. These weights are strategically employed to discern and adapt to the significance of varied input features.

In this study, the researcher critically evaluated the *Multi-Scale Feature Representations* as proposed by the NAS-FPN model [GLPL19]. It was observed that the execution of this model necessitates a disproportionately high consumption of resources, primarily due to the vastness of its search space. Furthermore, when inspecting architectures such as FPN [LDG+17] and NAS-FPN [GLPL19], there seems to be a discrepancy in the manner in which different input features are fused, resulting in an inconsistent output feature fusion. To address these shortcomings and augment the model's efficiency, the BiFPN was introduced, which incorporates *cross-scale connections*. These connections are specifically designed to rectify the aforementioned inefficiencies in the previously discussed architectures. The optimizations encapsulated within BiFPN include:

1. The elimination of nodes possessing only a singular input edge. This is orchestrated with the objective of eschewing nodes that do not actively participate in the fusion of different feature maps, thus ensuring that only the significant nodes with multiple connections contribute to feature amalgamation.

2. The integration of an additional edge that extends from the original input feature maps to the corresponding output node, provided they reside on the same hierarchical level. This is pursued to facilitate a more comprehensive feature fusion without necessitating the introduction of a plethora of operators.

3. The incorporation of a bi-directional pathway (both top-down and bottom-up) within a singular feature network layer. This layer is then replicated numerous times to foster an enriched high-level feature fusion process.

A graphic representation detailing these optimizations, showcasing BiFPN as the feature network, can be found in Figure 3.10.

In the preceding architectures, there is a discernible uniformity in the treatment of input features. This homogeneity often overlooks the nuanced significance that some features might possess over others. Contrary to this general approach, BiFPN employs a strategy that allocates a unique weight to each input. This allocation is not arbitrary; rather, it is meticulously designed to facilitate the network's learning of the relative importance of each input feature. This weighting mechanism is encapsulated in the methodology termed *Fast Normalized Fusion*, which is described as:

$$O = \sum_i \frac{w_i}{\epsilon + \sum_j w_j} \cdot I_i, \tag{3.1}$$

In the above equation, I_i delineates the i-th fused feature map. The term w_i represents the weight of the corresponding feature map, and to ensure its non-negativity, a Rectified Linear Unit (ReLU) is applied post determination of each w_i. The constant $\epsilon = 0.0001$ is judiciously incorporated to circumvent potential numerical instabilities that might arise during calculations.

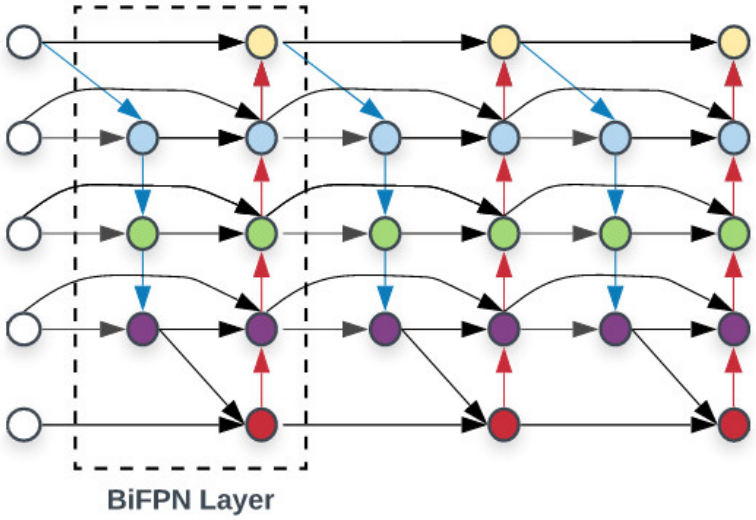


Figure 3.10: Diagram illustrating of BiFPN as the feature network. Image adapted from [WDC+23].

3.3 Object Instance Segmentation

Segmenting and identifying distinct objects within images is a foundational pillar of computer vision. This process, often termed as Object Instance Segmentation, is instrumental in equipping machines with the capability to delineate and recognize individual entities, as explained in the last chapter. This section traces this evolutionary path, elucidating the mechanisms and novelties of R-CNN and its descendants: Fast R-CNN, Faster R-CNN, and Mask R-CNN. Each algorithm brought forth incremental refinements, ensuring faster processing times, enhanced accuracy, and the ability to tackle more nuanced tasks like instance segmentation.

3.3.1 R-CNN

In the realm of object detection, the R-CNN (Regions with Convolutional Neural Networks) methodology stands as a pioneering contribution. It was introduced by *Ross Girshick et al.* in their work [GDDM13], and it skillfully integrates region proposals with the functionalities of convolutional neural networks. The architecture's essence is captured within three primary modules:

1. *Region Proposal*: This module allow to generate around 2000 category-independent region proposals for each image. This is achieved using the *Selective Search* [UvdSGS13] algorithm, which is based on a bottom-up segmentation approach.

2. *Feature Extraction*: Each of these region proposals is then warped to a fixed size and passed through a pre-trained CNN to extract a 4096-dimensional feature

vector.

3. *Classification*: A set of class-specific linear Support Vector Machines (SVMs) is trained using the above feature vectors. For each region, the SVM outputs a classification score, and bounding box regressors are used to fine-tune the object's location.

A visual representation encapsulating these modules is presented in Figure 3.11.

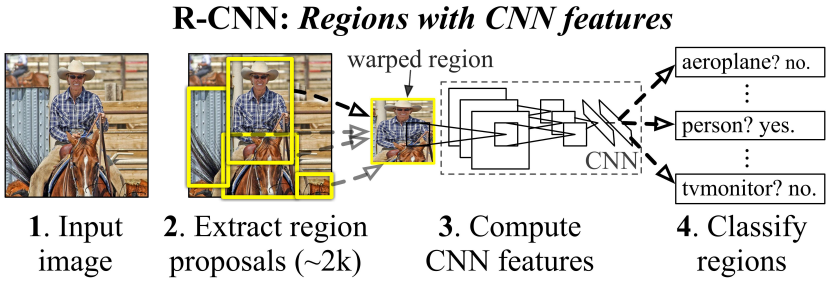


Figure 3.11: The object detection framework as delineated by R-CNN. Image adapted from [GDDM13].

3.3.2 Fast R-CNN

Despite the groundbreaking innovations introduced by the architecture of R-CNN [GDDM13], it exhibits several notable limitations. Chief among these are its space and time-intensive multi-stage pipeline training, the prolonged detection times during the inference phase, and the absence of shared computation in the forward pass for each object proposal. Recognizing these challenges, Ross Girshick introduced the Fast R-CNN architecture [Gir15], which employs a single-stage training algorithm. This new algorithm concurrently learns to classify object proposals and finetune their spatial positions.

In the operational flow of the Fast R-CNN architecture, the input image initially undergoes processing through multiple convolutional and max pooling layers, leading to the generation of a feature map. Subsequently, for each object proposal, the architecture incorporates a Region Of Interest (RoI) pooling layer, which is pivotal in extracting a fixed-length feature vector from the aforesaid feature map. This feature vector is then forwarded to a series of fully connected layers, culminating in two distinct output layers. The first output layer is tasked with generating softmax probability estimates over K object classes, plus a separate class for the background. The second layer, conversely, outputs four real-valued coordinates for each of the K object classes. These coordinates effectively refine the bounding-box positions corresponding to each class. A schematic representation of this intricate process is delineated in Figure 3.12.

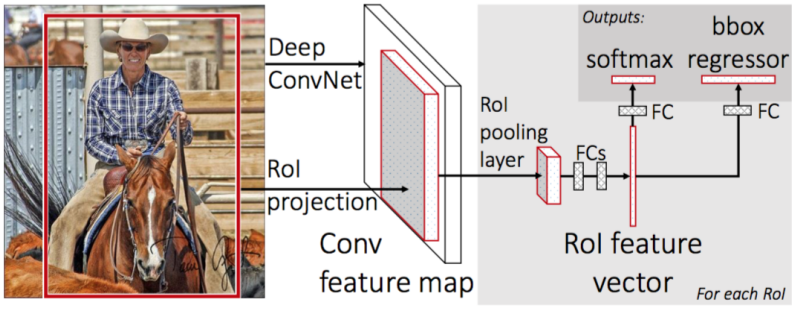


Figure 3.12: An overview of the Fast R-CNN architecture, as sourced from [Gir15].

Within the scope of the described processes, the introduction of the *Region of Interest (RoI) pooling layer* emerges as a crucial innovation. This RoI pooling layer employs the max pooling operator to transform features encapsulated within any valid region of interest into a compact feature map of a consistent spatial dimension, represented as $H \times W$. Notably, the dimensions H and W function as layer hyperparameters and maintain independence from any specific RoI. Every individual RoI is characterized by a quadruple (r, c, h, w), which demarcates its top-left corner with coordinates (r, c) and further defines its dimensions with height h and width w. Here, the operational principle of the max pooling technique entails partitioning the $h \times w$ RoI window into an $H \times W$ sub-window grid. Each of these sub-windows approximates a size of $h/H \times w/W$, and the values within them are subsequently max-pooled into the corresponding grid cell of the output feature map.

The epithet *fast* in the architecture alludes to its refinement optimized for detection. The underlying objective is the holistic training of all weights during the back-propagation phase. Fast R-CNN achieves this by harnessing a single-stage training process in tandem with a multi-task loss. This multi-task loss is intrinsically tied to the aforementioned dual output layers. For the primary output layer, a discrete probability distribution, symbolized as $p = (p_0, ..., p_K)$, is formulated for each RoI. This distribution is derived from the softmax function applied over the $K + 1$ outputs emanating from a fully connected layer. Pertaining to the secondary output layer, bounding-box regression offsets are parameterized as $t^k = \left(t^k_{\mathrm{x}}, t^k_{\mathrm{y}}, t^k_{\mathrm{w}}, t^k_{\mathrm{h}}\right)$ for the k-th object class. Then, given a labeled RoI associated with a ground-truth class u and a corresponding bounding-box regression target v, the encompassing multi-task loss L for both classification and bounding-box regression can be expressed as:

$$L\left(p, u, t^u, v\right) = L_{\mathrm{cls}}(p, u) + \lambda[u \geq 1] L_{\mathrm{reg}}\left(t^u, v\right). \tag{3.2}$$

In this equation, λ functions as a hyperparameter modulating the equilibrium between the dual task losses. The Iverson bracket indicator function, denoted by $[u \geq 1]$, evaluates to 1 when the condition $u \geq 1$ is met and 0 otherwise. Further, $L_{\mathrm{cls}}(p, u) = -\log p_u$ signifies the log loss for the genuine class u. The bounding-box regression

loss, L_{reg}, is ascribed to the target $u, v = (v_{\text{x}}, v_{\text{y}}, v_{\text{w}}, v_{\text{h}})$ and its predicted counterpart $t^u = \left(t^u_{\text{x}}, t^u_{\text{y}}, t^u_{\text{w}}, t^u_{\text{h}}\right)$, and is given by:

$$L_{\text{reg}}\left(t^u, v\right) = \sum_{i \in \{\text{x,y,w,h}\}} \text{smooth}_{L_1}\left(t^u_i - v_i\right) \tag{3.3}$$

The function $\text{smooth}_{L_1}(x)$ is defined as:

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}. \tag{3.4}$$

3.3.3 Faster R-CNN

Leveraging the innate potential of Fast R-CNN for generating region proposals, *Shaoqing Ren et al.* formulated an integrated architecture, termed as Faster R-CNN [RHGS15]. This architecture comprises two pivotal modules: (I) Region Proposal Networks (RPNs) and (II) the Fast R-CNN detector, as depicted in Figure 3.13. The overarching objective of these networks is to engender region proposals potentially encapsulating objects.

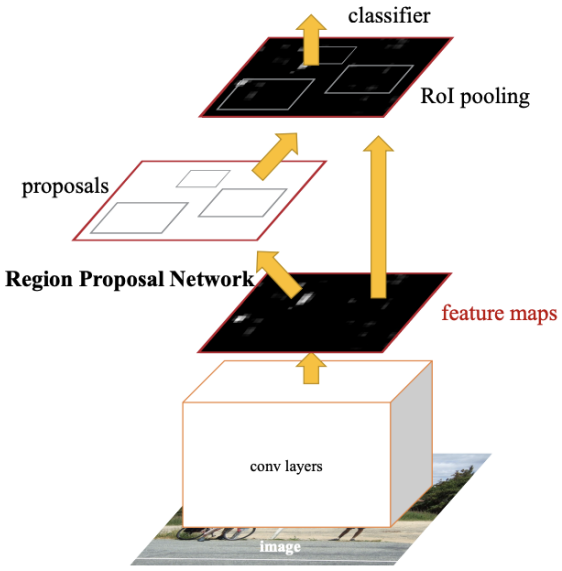


Figure 3.13: Faster R-CNN represents a unified network dedicated to object detection, wherein the RPN module acts as the network's *attention*. Image and caption have been sourced from [RHGS15].

Delving deeper, the Region Proposal Networks are designed to intake an image of arbitrary dimensions and subsequently output an array of rectangular object proposals, each quantified with an objectness score. To achieve this, the RPN processes a diminutive network atop the convolutional feature map that emerges from the terminal shared

convolutional layer. This concise network imbibes an $n \times n$ spatial window from the originating convolutional feature map. Every sliding window is then transposed into a lower-dimensional feature, subsequently funneled into two sibling fully-connected layers. These layers are tailored for box-regression and box-classification, respectively. For each sliding window locale, simultaneous predictions are made for multiple region proposals, with the upper bound of possible proposals for any given location symbolized by k. These k proposals are parameterized in correlation to k reference boxes, denominated as *anchors*. Notably, every anchor is centralized at the respective sliding window, possessing a defined scale and aspect ratio, as illustrated in Figure 3.14.

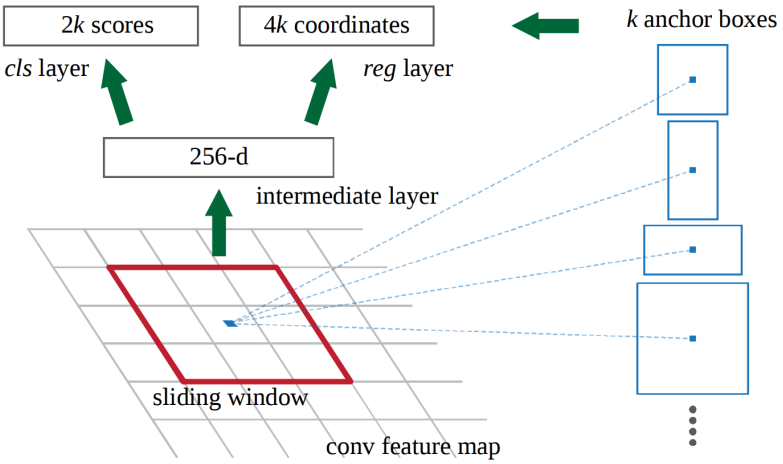


Figure 3.14: The architecture of the Region Proposal Network (RPN). Image and caption sourced from [RHGS15].

In a quest to uphold the *fast* essence intrinsic to the Fast R-CNN [Gir15], the Faster R-CNN architecture embodies a revamped multi-task loss function for the Region Proposal Network. This is achieved by introducing a binary class assignment for every anchor, distinguishing it as either an object or a non-object. The resultant loss to be minimized can be represented as:

$$L\left(\{p_i\},\{t_i\}\right) = \frac{1}{N_{\text{cls}}}\sum_i L_{\text{cls}}\left(p_i, p_i^*\right) + \lambda\frac{1}{N_{\text{reg}}}\sum_i p_i^* L_{\text{reg}}\left(t_i, t_i^*\right) \tag{3.5}$$

Here, i signifies the anchor's index within a mini-batch and p_i denotes the predicted likelihood of anchor i being classified as an object. The ground-truth label, p_i^*, is set to 1 if the anchor is identified as positive, and 0 if negative. Meanwhile, t_i stands for a vector detailing the four parameterized coordinates of the projected bounding box. In contrast, t_i^* designates the ground-truth box linked with a positive anchor. The terms L_{cls} and L_{reg} mirror those described in equation (3.2). They are normalized by factors N_{cls} (referring to the mini-batch size) and N_{reg} (pertaining to the total anchor locations). Lastly, λ operates as a hyperparameter orchestrating the balance between the twin task losses.

3.3.4 Mask R-CNN

In their research, *Kaiming He et al.* sought to adapt the architecture of the Faster R-CNN to cater to instance segmentation tasks. The method they introduced, named Mask R-CNN [HGDG17], integrates an additional branch designed specifically to predict segmentation masks for each Region of Interest (RoI). This branch operates concurrently with the existing branches focused on classification and bounding box regression, as depicted in Figure 3.15.

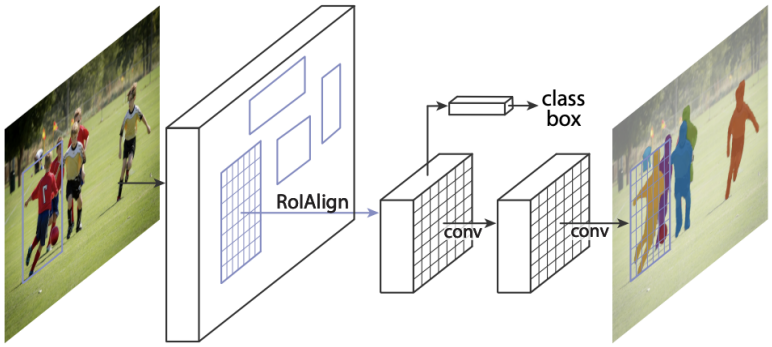


Figure 3.15: The architecture of Mask R-CNN, as sourced from *Kaiming He et al.* [HGDG17].

The architecture of Mask R-CNN builds upon the foundational structure of Faster R-CNN. While the Faster R-CNN is responsible for the class label and bounding-box prediction, the Mask R-CNN integrates a novel branch dedicated to outputting the object mask. This mask is distinct from the class and bounding box outputs, necessitating a more intricate spatial layout extraction of an object. The branch is designed to encode K binary masks, each of resolution $m \times m$, one corresponding to each of the K classes. The intricate spatial structure of the masks is addressed by the inherent pixel-to-pixel correspondence that convolutions offer. Within this architecture, an $m \times m$ mask is predicted for each RoI using a Fully Convolutional Network (FCN). This approach ensures that each layer in the mask branch retains the explicit $m \times m$ object spatial layout, avoiding its reduction into a vector representation devoid of spatial dimensions. One significant innovation in the Mask R-CNN is its acknowledgment of the pixel-to-pixel alignment between network inputs and outputs, an aspect overlooked by the RoIPool utilized in the Fast R-CNN architecture. To address this, Mask R-CNN incorporates a novel RoI layer, termed RoIAlign. RoIAlign serves to rectify the severe quantization inherent in RoIPool, ensuring a precise alignment of the extracted features with the input. The principal objective of this layer is to eschew any quantization of the RoI boundaries or bins. Instead, it employs bilinear interpolation to accurately compute the values of the input features at four systematically sampled locations within each

RoI bin, as depicted in Figure 3.16.

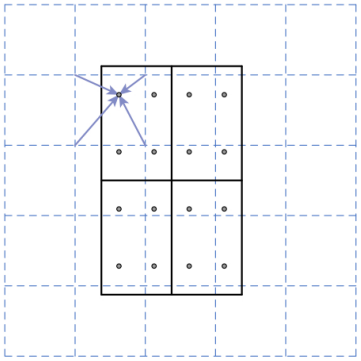


Figure 3.16: RoIAlign: The dashed grid symbolizes a feature map, with the solid lines representing an RoI (comprising 2×2 bins in this instance), and the dots marking the four sampling points in each bin. RoIAlign determines the value of each sampling point through bilinear interpolation from the adjacent grid points on the feature map. Crucially, no quantization is applied to any coordinates related to the RoI, its bins, or the sampling points. Image and caption sourced from [HGDG17].

The training phase for the Mask R-CNN method introduces a composite multi-task loss for each sampled RoI, formulated as $L = L_{\text{cls}} + L_{\text{reg}} + L_{\text{mask}}$. Here, L_{cls} and L_{reg} remain consistent with those detailed in equation (3.5). L_{mask} is the average binary cross-entropy loss. Notably, for an RoI correlated with the ground-truth class k, L_{mask} is solely defined on the k-th mask. This unique definition of the loss L_{mask} empowers the network to produce masks for each class without any class competition.

Chapter 4

Proposal

In alignment with the objectives delineated in Section 1.3.2, this chapter presents the newly proposed architectural framework termed **CA-BiFPN**. This design draws inspiration from the *two-stage detector* paradigm, as comprehensively elucidated in Chapter 3. The architecture is strategically segmented into three primary components:

1. **Backbone**: This component represents the refined network specifically tailored for feature extraction. Within this segment, we harness feature maps from diverse stages of the network. The intent is to meticulously capture feature maps both from advanced (high-level) stages and foundational (low-level) stages of the network.

2. **Neck**: The *neck* merges connections from various stages of the backbone. It incorporates the structure of the *BiFPN* [TPL19]. Within this framework, we introduced two new components: the **Local Context Aggregator** and the **Global Context Aggregator**. A deeper exploration of these components will be provided in Section 4.1.

3. **Head Detector**: This segment integrates a *Mask R-CNN* head [HGDG17], equipped with two distinct *RoI Pooler* operators. While one is dedicated to the Object Detection task, the other is tailored for the Instance Segmentation task. These poolers subsequently receive inputs from the varied fused levels extended by the *neck*.

A comprehensive visualization of the final architecture is presented in Figure 4.1.

The subsequent sections will delve into the design intricacies of the *Context Aggregator Units* located within the *neck*, emphasizing their integration within the *BiFPN* structure.

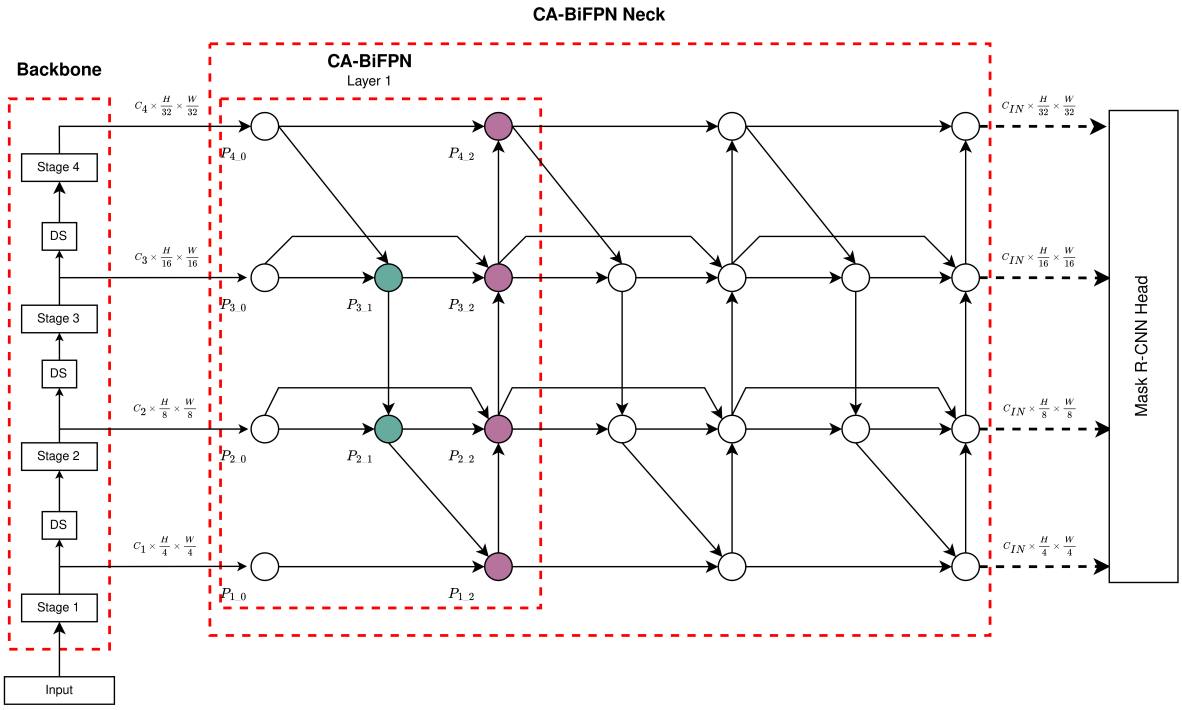


Figure 4.1: Detailed schematic of the CA-BiFPN architecture.

4.1 Context Aggregator Units

Drawing parallels with *ParseNet* [LRB15], the *Context Aggregator* unit is designed to synergize both local and global features emanating from the backbone stage. Within the scope of this research, the distinctions between local and global contextual features are articulated as follows:

1. **Local Context Feature**: These originate from the deeper stage layers in the backbone, often referred to as the bottom-up pathway of the network. Possessing a minimal receptive field, these features preserve intricate spatial information, thereby facilitating the generation of high-level features.

2. **Global Context Feature**: The features extracted from the shallower stages of the backbone, characterized by their expansive receptive field, are adept at capturing robust semantic information. These are categorized as low-level features.

4.1.1 Global Context Operator

In the *Squeeze-and-Excitation* networks study [HSS17], the convolutional vector operation presented in Equation (2.4), denoted as $\boldsymbol{K} * \boldsymbol{x}$, involves a summation across all channels C. The output from this operation is closely interwoven with the local spatial correlations captured by the filters $\boldsymbol{K}$. To address the convolutional complexities,

the squeeze-and-excitation block integrates global spatial information throughout the channels and subsequently consolidates this data through channel-wise dependencies. Then, for an input $\mathbf{X} \in \mathbb{R}^{C\times H\times W}$, the *global feature context operator* of the squeeze-and-excitation block is defined as follows:

$$\mathbf{G}(\mathbf{X}) = \mathbf{W}_2 \left(\delta \left(\mathbf{W}_1(\mathbf{g}(\mathbf{X}))\right)\right), \tag{4.1}$$

where both $\mathbf{W}_1 \in \mathbb{R}^{\frac{C}{r}\times C}$ and $\mathbf{W}_2 \in \mathbb{R}^{C\times\frac{C}{r}}$ represent learnable weights. The channel reduction factor is denoted as r and usually takes values within the set $\{2, 4\}$. Here, δ signifies the activation function. The squeeze operator, symbolized as $\mathbf{g}(\mathbf{X})$, is formulated as:

$$g(\mathbf{X})_c = \frac{1}{H\times W}\sum_{i=1}^{H}\sum_{j=1}^{W} x_{c,i,j}. \tag{4.2}$$

In this equation, $g(\mathbf{X})_c$ is a component of $\mathbf{g}(\mathbf{X}) \in \mathbb{R}^C$, and $x_{c,i,j}$ corresponds to individual units of the feature map $\mathbf{X}$.

Building upon the concept of point-wise channel representation introduced by the *global feature context operator*, we focus on the interaction of the global spatial information throughout the channels. This approach mirrors that of the *MS-CAM* block presented by *Yimian Dai et al.* in their research titled *Attentional Feature Fusion* [DGO+20]. Guided by their insights, we have chosen to implement the *point-wise convolution* for channel aggregation, specifically targeting the learnable weights $\mathbf{W}_1$ and $\mathbf{W}_2$.

As a result, the **Global Context Operator** formulated in our study can be expressed through the following equation:

$$\sigma\left(\mathbf{G}(\mathbf{X})\right) = \sigma\left(\text{PW-Conv}_{1\times1}\left(\delta\left(\text{PW-Conv}_{1\times1}(\mathbf{g}(\mathbf{X}))\right)\right)\right). \tag{4.3}$$

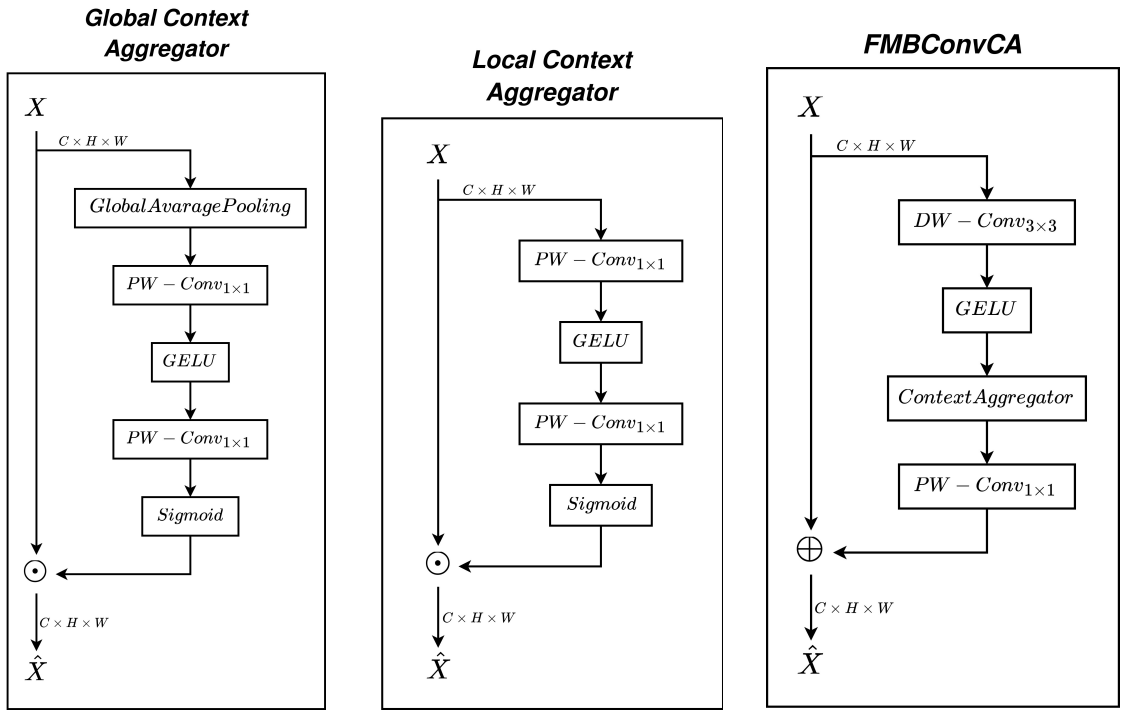
Within this equation, σ denotes the *Sigmoid* function, while δ refers to the GELU [HG23] activation function. The term $\text{PW-Conv}_{1\times1}$ signifies the point-wise convolution. Following this, a channel-wise multiplication is performed between the output $\sigma\left(\mathbf{G}(\mathbf{X})\right)$ and the input $\mathbf{X}$. A graphical representation of the **Global Context Operator** is available in Figure 4.2a.

4.1.2 Local Context Operator

Given the understanding that channel relationships shaped by convolution are inherently implicit and localized, the *Local Context Operator* functions as a simplified version of equation (4.3). Notably, this operator excludes the *global feature context operator*:

$$\sigma\left(\mathbf{L}(\mathbf{X})\right) = \sigma\left(\text{PW-Conv}_{1\times1}\left(\delta\left(\text{PW-Conv}_{1\times1}(\mathbf{X})\right)\right)\right). \tag{4.4}$$

Mirroring the process in equation (4.3), a channel-wise multiplication is carried out between the output $\sigma\left(\mathbf{L}(\mathbf{X})\right)$ and the input $\mathbf{X}$. A graphical illustration of the **Local Context Operator** can be found in Figure 4.2b.



(a) Detailed schematic Global Context Aggregator Unit.

(b) Detailed schematic Local Context Aggregator Unit.

(c) Detailed schematic of the merge unit node.

Figure 4.2: Diagram illustrating the Context Aggregator Units and merge unit node FMBConvCA, proposed in this work.

4.1.3 FMBConvCA Block

After establishing the architecture of the *Context Aggregator Units*, we incorporate them into a refined version of the *Fused-MBConv* block [GT], hereafter referred to as **FMBConvCA**. This modification is analogous to the enhanced block delineated in the study by *Hatamizadeh et al.* [HYH+23]. The purpose of this novel block is to imbue the network with advantageous characteristics, such as inductive bias and the capacity to model inter-channel dependencies, particularly at the various stages within the neck of the architecture. The bottleneck equation for the *FMBConvCA* block can be formally defined as follows:

$$\text{FMBConvCA}\left(\mathbf{X}\right) = \text{PW-Conv}_{1\times1}\left(\text{CA}\left(\delta\left(\text{DW-Conv}_{3\times3}\left(\mathbf{X}\right)\right)\right)\right), \tag{4.5}$$

In Equation 4.5, the term $\text{DW-Conv}_{3\times3}$ denotes a 3×3 depth-wise convolution operation [HSS17]. The symbol δ is employed to represent the Gaussian Error Linear Unit (GELU) activation function [HG23]. Moreover, $\text{PW-Conv}_{1\times1}$ signifies the point-wise convolution operation. Within this equation, CA stands for the *Context Aggregator Units*, which can take the form of either $\sigma\left(\mathbf{L}(\mathbf{X})\right)$ or $\sigma\left(\mathbf{G}(\mathbf{X})\right)$. Lastly, the output of the $\text{FMBConvCA}\left(\mathbf{X}\right)$ operation is subject to direct summation with the input vector

X. A visual representation of the FMBConvCA block is provided in Figure 4.2c.

The incorporation of the *Context Aggregator Units* into the **FMBConvCA** block provides enhanced capabilities for contextual understanding and feature extraction. By elegantly combining the depth-wise and point-wise convolutions with the *Context Aggregator Units* and GELU activation function, the block aims to achieve a balance between computational efficiency and representational power. This design fosters a robust architecture capable of handling intricate data patterns and inter-channel dependencies, thereby amplifying the overall performance of the neck.

4.2 Architecture of the CA-BIFPN Neck

As alluded to in the introductory section of this chapter, the salient innovations proposed in this work predominantly reside in the *neck* component of the overall neural network architecture. The schematic representation of this enhanced *neck*, denoted as CA-BIFPN, is illustrated in Figure 4.1. The proposed architecture incorporates the node structures layer from the *BiFPN* [TPL19] and substitutes the *Fast Normalized Fusion* weighting mechanism with the *FMBConvCA* block within each node.

The input nodes for this specialized neck region are labeled as P_{1_0}, P_{2_0}, P_{3_0}, and P_{4_0}, each of which corresponds to the lateral output from the backend of the network. The intermediate nodes[1], namely P_{3_1} and P_{2_1}, serve as internal context aggregation fusion points. Similarly, the terminal nodes[1], denoted as P_{1_2}, P_{2_2}, P_{3_2}, and P_{4_2}, represent the ultimate context aggregation fusion nodes and are analogous in function to their counterparts in the traditional *BiFPN* architecture layer.

The formal definitions for each of these feature fusion nodes are enumerated below:

$$P_{3\_1} = \text{FMBConvCA-}\mathbf{g}\left(P_{3\_0}\right) \oplus \text{FMBConvCA-}\mathbf{L}\left(\text{Resize}\left(P_{4\_0}\right)\right) \tag{4.6}$$

$$P_{2\_1} = \text{FMBConvCA-}\mathbf{g}\left(P_{2\_0}\right) \oplus \text{FMBConvCA-}\mathbf{L}\left(\text{Resize}\left(P_{3\_1}\right)\right) \tag{4.7}$$

$$P_{1\_2} = \text{FMBConvCA-}\mathbf{g}\left(P_{1\_0}\right) \oplus \text{FMBConvCA-}\mathbf{L}\left(\text{Resize}\left(P_{2\_1}\right)\right) \tag{4.8}$$

$$P_{2\_2} = \text{FMBConvCA-}\mathbf{L}\left(P_{2\_0}\right) \oplus \text{FMBConvCA-}\mathbf{L}\left(P_{2\_1}\right) \tag{4.9}$$

$$\oplus\, \text{FMBConvCA-}\mathbf{g}\left(\text{Resize}\left(P_{1\_2}\right)\right) \tag{4.10}$$

$$P_{3\_2} = \text{FMBConvCA-}\mathbf{L}\left(P_{3\_0}\right) \oplus \text{FMBConvCA-}\mathbf{L}\left(P_{3\_1}\right) \tag{4.11}$$

$$\oplus\, \text{FMBConvCA-}\mathbf{g}\left(\text{Resize}\left(P_{2\_2}\right)\right) \tag{4.12}$$

$$P_{4\_2} = \text{FMBConvCA-}\mathbf{L}\left(P_{4\_0}\right) \oplus \text{FMBConvCA-}\mathbf{g}\left(\text{Resize}\left(P_{3\_2}\right)\right) \tag{4.13}$$

In the above equations, FMBConvCA-**L** and FMBConvCA-**g** represent variants of the *FMBConvCA Block* configured for both *Local Context Operator* and *GLobal Context*

[1]The colors JungleGreen and DarkOrchid are consistent with the color scheme used to denote intermediate and terminal nodes, respectively, in Figure 4.1.

Operator, and $\oplus$ indicates a direct summation operation.

The CA-BIFPN neck design presents a nuanced approach to achieving efficient feature fusion and contextual understanding. By embedding the *FMBConvCA Block* within the nodes, the architecture enhances its capabilities for capturing complex relationships and spatial dependencies across channels. These intricacies are efficiently managed, thus contributing to the overall robustness and performance of the neck.

Chapter 5

Results

5.1 Experiments Definitions

The primary aim of this study is to rigorously investigate the efficacy of our proposed end-to-end neural network model in addressing challenges related to object detection and instance segmentation. To achieve this, a meticulous experimental setup has been established to ensure both the computational rigor and the reproducibility of results.

5.1.1 Hardware Configuration

For the purpose of this study, a specialized hardware configuration was utilized during the experimental phase to facilitate accelerated computation. This hardware was instrumental in training the end-to-end neural network with the assistance of GPU acceleration. A comprehensive list of the hardware components and their respective models is provided in Table 5.1.

Component	**Model**
CPU	*AMD Ryzen 9 5950X 16-Core Processor*
GPU	*NVIDIA GeForce RTX 3090 24GB*
RAM	*64GB*
Data Storage	*NVMe 1TB*

Table 5.1: Detailed hardware configuration employed for model training.

5.1.2 Specifications of Computational Environment

The architecture of the proposed network was realized through the utilization of the PyTorch framework [PGM+19]. To ensure compatibility and leverage optimizations, the computational environment was configured using NVIDIA NGC's official PyTorch

container, version 23.05 [1]. This containerized approach facilitates reproducibility and enhances the overall robustness of the experimental setup.

For the quantitative assessment of model performance, metric evaluations were conducted as delineated in Section 2.3.1. Specifically, the PyTorch model underwent evaluation via the TorchMetrics API library [2]. This library was chosen for its comprehensive set of metrics that are well-aligned with the evaluation criteria set forth in this study.

Furthermore, the methodologies for CNN explainability were discussed in Section 2.3.2. To elucidate the behavior of the trained model, two strategies were employed: Ablation CAM and ROAD strategy. These strategies were implemented using the *Advanced AI Explainability for PyTorch* package [Gc21].

The official repository, which contains the source code, required libraries, configuration files, and checkpoints, can be accessed via the following URL: `https://github.com/camilo-nunez/ca-bifpn`.

5.1.3 Training Specifications

Dataset

In order to rigorously assess the performance of our proposed model in the realms of object detection and instance segmentation, we employed the Common Objects in Context (COCO) dataset, as outlined in [LMB+14]. Specifically, the 2017 version of this dataset was utilized for both the training and evaluation phases. The COCO dataset is a large-scale, multi-purpose resource that encompasses various tasks within computer vision, including image classification, object detection, semantic segmentation, and instance segmentation. It boasts an extensive collection of 328,000 images, encompassing 2.5 million labeled instances across 91 object classes, 80 of which are designated for the task of instance segmentation.

One salient feature of the COCO dataset that aligns well with the objectives of this research is its proclivity for scale variation and small object instances. A significant portion of object instances in the dataset occupies less than 1% of the overall image area, as noted in [SD17]. This characteristic makes the COCO dataset an ideal candidate for evaluating how well our proposed model can generalize and perform when confronted with scale variability and diminutive object instances.

Nevertheless, it is imperative to acknowledge the inherent class imbalance within the COCO dataset. Notably, the class labeled `'person'` is disproportionately represented, boasting a far greater number of instances compared to other classes.

[1] `https://docs.nvidia.com/deeplearning/frameworks/pytorch-release-notes/rel-23-05.html`

[2] `https://torchmetrics.readthedocs.io/en/stable/`

Training Phase

As delineated in Chapter 4, the architecture of the proposed end-to-end model is compartmentalized into three integral components: (I) the backbone, (II) the neck, and (III) the head detector.

Backbone Selection and Configuration For the backbone, we leverage three pre-trained models: InternImage [WDC+23], ConvNeXt [LMW+22], and EfficientNetV2 [TL21]. The configurations of these models employed during training are detailed in Table 5.2. The backbones are categorized into three distinct groups, each of which is organized based on the similarity in the number of parameters. Due to constraints in hardware capabilities and the computational time required for training, our experiments focus exclusively on the models belonging to the group labeled as $G1$. However, configuration files for all the proposed groups of backbones are publicly available in the official repository.

Group	Model Name	**# of Parameters (Millions)**
G0	InternImage-T	28M
	ConvNeXt-T	27M
	EfficientNetV2-S	19M
G1	InternImage-S	48M
	ConvNeXt-S	49M
	EfficientNetV2-M	52M
G2	InternImage-B	94M
	ConvNeXt-B	87M
	EfficientNetV2-L	116M

Table 5.2: Configuration groups of the backbones employed in the proposed model, alongside their respective number of parameters in millions.

Each selected backbone model has been pre-trained on an image classification task using the ImageNet dataset [DDS+09]. In accordance with established best practices in transfer learning [YCBL14], the weights of these models are frozen to serve as feature extractors.

Neck Configuration The neck component in the training phase adopts two distinct configurations: one based on the original BiFPN model and another based on the proposed CA-BiFPN model. In adherence to the standard configurations utilized in the InternImage work [3], we employ a five-layered architecture for each neck with an output channel size of 256[4].

[3] `https://github.com/OpenGVLab/InternImage/tree/master/detection/configs/coco`

[4] C_{IN} in output nodes as depicted in Figure 4.1

Head Detector Configuration As previously stated in the preceding chapter, the head detector employed in the end-to-end model is Mask R-CNN. The configuration for the anchor sizes is set to $(32, 64, 128, 256)$ and aspect ratios are set to $(0.5, 1.0, 1.5, 2.0)$. Both the RoI Pooler Object and the RoI Pooler Mask receive feature outputs labeled as `'P0', 'P1', 'P2', 'P3'` from the neck component.

Implementation of Training Procedures Building upon the aforementioned definitions of backbones and necks, the ultimate configurations utilized for the training implementation in this study comprise the following combinations:

- InternImage-S [WDC+23] + BiFPN [TPL20]
- ConvNeXt-S [LMW+22] + BiFPN [TPL20]
- EfficientNetV2-M [TL21] + BiFPN [TPL20]
- InternImage-S [WDC+23] + CABiFPN (ours)
- ConvNeXt-S [LMW+22] + CABiFPN (ours)
- EfficientNetV2-M [TL21] + CABiFPN (ours)

In alignment with established best practices exemplified by InternImage [WDC+23], the backbones components were initialized with weights pretrained on classification tasks. Subsequent training was conducted on the neck components as well as the Mask R-CNN head detector, adhering to a $1x$ schedule comprising 12 epochs[5].

The experiments utilized the `train2017` dataset from COCO. Moreover, a series of image augmentations were employed using the Albumentations library [BIK+20]. The specific transformations and their respective probabilities are delineated in Table 5.3. These augmentations were applied both to the input images and the associated ground-truth annotations, including bounding boxes and binary segmentation masks.

To optimize the networks, the AdamW optimizer [KB17] was utilized, conforming to the $1x$ schedule. The learning rate was set at 0.001, with a weight decay coefficient of 0.00001. Training was executed with a batch size of two to mitigate the risk of `out-of-memory` errors. It should be noted that, to further avert memory-related complications, the default image size from the ImageNet dataset was employed, as opposed to the larger InternImage dimensions of 1333×800.

[5]For further insights into common scheduling practices for object detection networks trained from scratch, refer to `https://github.com/facebookresearch/detectron2/blob/main/MODEL_ZOO.md`

Transform	**Probability**	**Description**
Resize	$p = 1$	Rescale to 224×224
Random Brightness Contrast	$p = 0.4$	Alter brightness and contrast randomly
RGB Shift	$p = 0.5$	Shift RGB channels randomly
Invert Image	$p = 0.5$	Invert colors by subtracting pixel values
Blur	$p = 0.3$	Apply blur
Gauss Noise	$p = 0.4$	Add Gaussian noise
Flip	$p = 0.4$	Flip image
Random Rotate 90°	$p = 0.4$	Rotate 90°
Normalize	$p = 1$	Normalize image

Table 5.3: Image augmentation transforms from Albumentations [BIK+20] along with their associated probabilities.

5.2 Metric Evaluation Results

In accordance with the evaluation conducted on the six trained models delineated in Section 5.1.3, we assessed their performance using the metric of mean average precision (AP) at Intersection over Union (IoU) thresholds of 0.50 and 0.75. Furthermore, a global average performance metric was derived for the entire validation set `val2017` from the COCO dataset. These empirical results are systematically presented in Table 5.4.

	#params		**Mask R-CNN 1x schedule**					
Model	**Neck**	**Backbone**	AP^{bbox}	AP^{bbox}_{50}	AP^{bbox}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
InternImage-S [WDC+23] + BiFPN [TPL20]	$2.3M$	$48.5M$	0.1654	0.2914	0.1715	0.0341	0.1239	0.0080
ConvNeXt-S [LMW+22] + BiFPN [TPL20]	$2.4M$	$49.4M$	0.1884	0.3215	0.1966	0.0405	0.1420	0.0100
EfficientNetV2-M [TL21] + BiFPN [TPL20]	$2.2M$	$52.2M$	0.2050	0.3507	0.2177	0.0416	0.1531	0.0092
InternImage-S [WDC+23] + CABiFPN (ours)	$9.7M$	$48.5M$	0.2071	0.3548	0.2168	0.0421	0.1508	0.0108
ConvNeXt-S [LMW+22] + CABiFPN (ours)	$9.8M$	$49.4M$	0.2297	0.3906	0.2416	0.0475	0.1663	0.0117
EfficientNetV2-M [TL21] + CABiFPN (ours)	$9.6M$	$52.2M$	0.2470	0.4203	0.2592	0.0433	0.1648	0.0083

Table 5.4: Performance metrics for object detection and instance segmentation on the `val2017` dataset from `COCO2017`. AP denotes the overall mean average precision. AP_{50} and AP_{75} indicate the mean average precision at $IoU = 0.50$ and $IoU = 0.75$ respectively. Meanwhile, AP^{bbox} and AP^{mask} represent the mean average precision for bounding box and mask results, respectively.

5.3 Ablation CAM Results

Due to the limitations of the measure provided in the previous section, which does not offer a comprehensive understanding of the neck model behavior, we aim to augment our model evaluation. To achieve this, we will examine the Class Activation Mapping

(CAM) across five layers of the necks, for each of the six trained models, utilizing the Ablation CAM visual explanation technique.

To provide a robust assessment, three disparate images from the COCO dataset have been selected at random, identifiable by their respective IDs: `96825`, `171382`, and `509403`. These images serve as the ground truth and are accompanied by their respective bounding boxes and segmentation masks, all of which are depicted in Figure 5.1.

For the purposes of this section, our primary results will center around the visual results generated for the image with ID `509403`. However, the visual results corresponding to images with IDs `96825` and `171382` have also been generated and can be found in the supplementary sections. Specifically, these supplementary figures are located in Section A.1 and Section A.2 of Appendix A.

Upon selection of these test images, we have conducted predictions to generate bounding boxes and segmentation masks for each. The results of these predictions are systematically represented in Figures 5.2, A.1, and A.2.

After applying the Ablation CAM visual explanation technique to the five layers of the neck in each trained model, we generated a series of visual results. For image ID `509403`, these are shown in Figures 5.3 to 5.8. Similarly, the visual outcomes for image ID `96825` are displayed in Figures A.3 to A.8, while those for image ID `171382` appear in Figures A.9 to A.14.

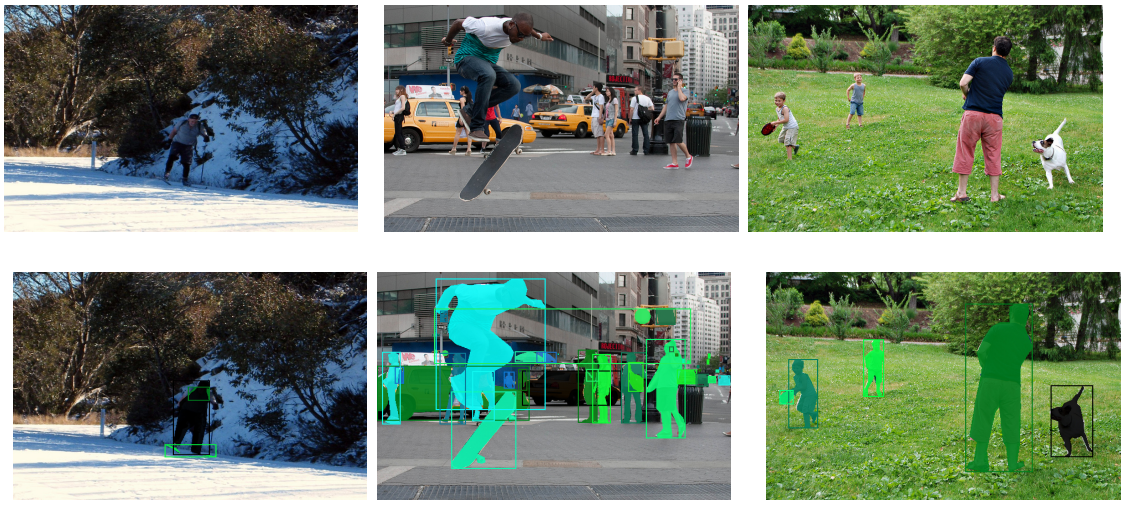
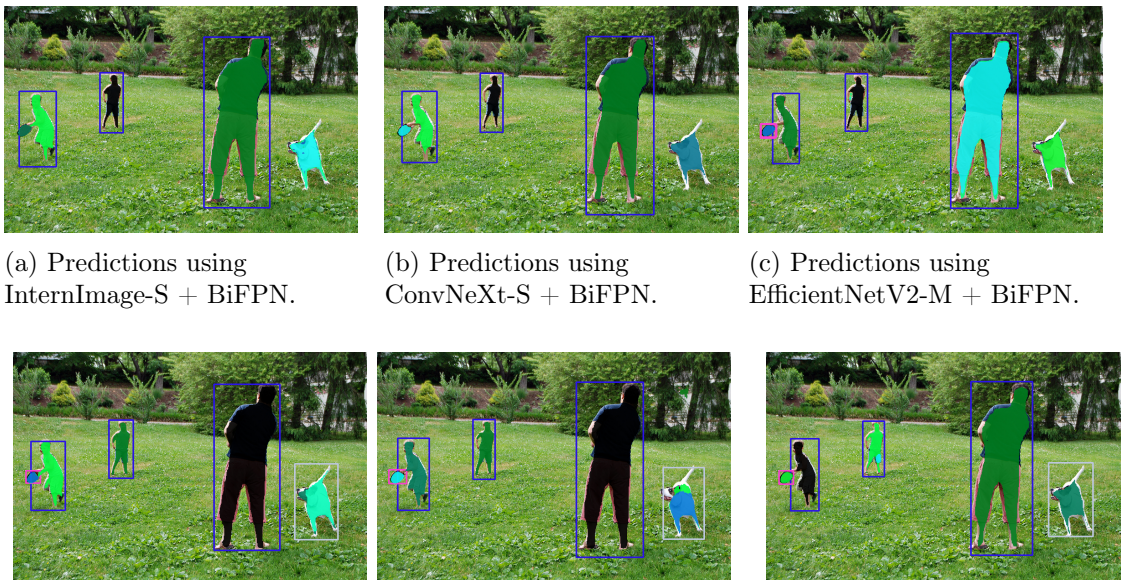


Figure 5.1: Original images from `COCO2017` utilized for testing. The first row displays images with IDs `96825`, `171382`, and `509403`. The second row showcases the bounding boxes and instance segmentation masks derived from the annotations of these images.



(a) Predictions using InternImage-S + BiFPN.

(b) Predictions using ConvNeXt-S + BiFPN.

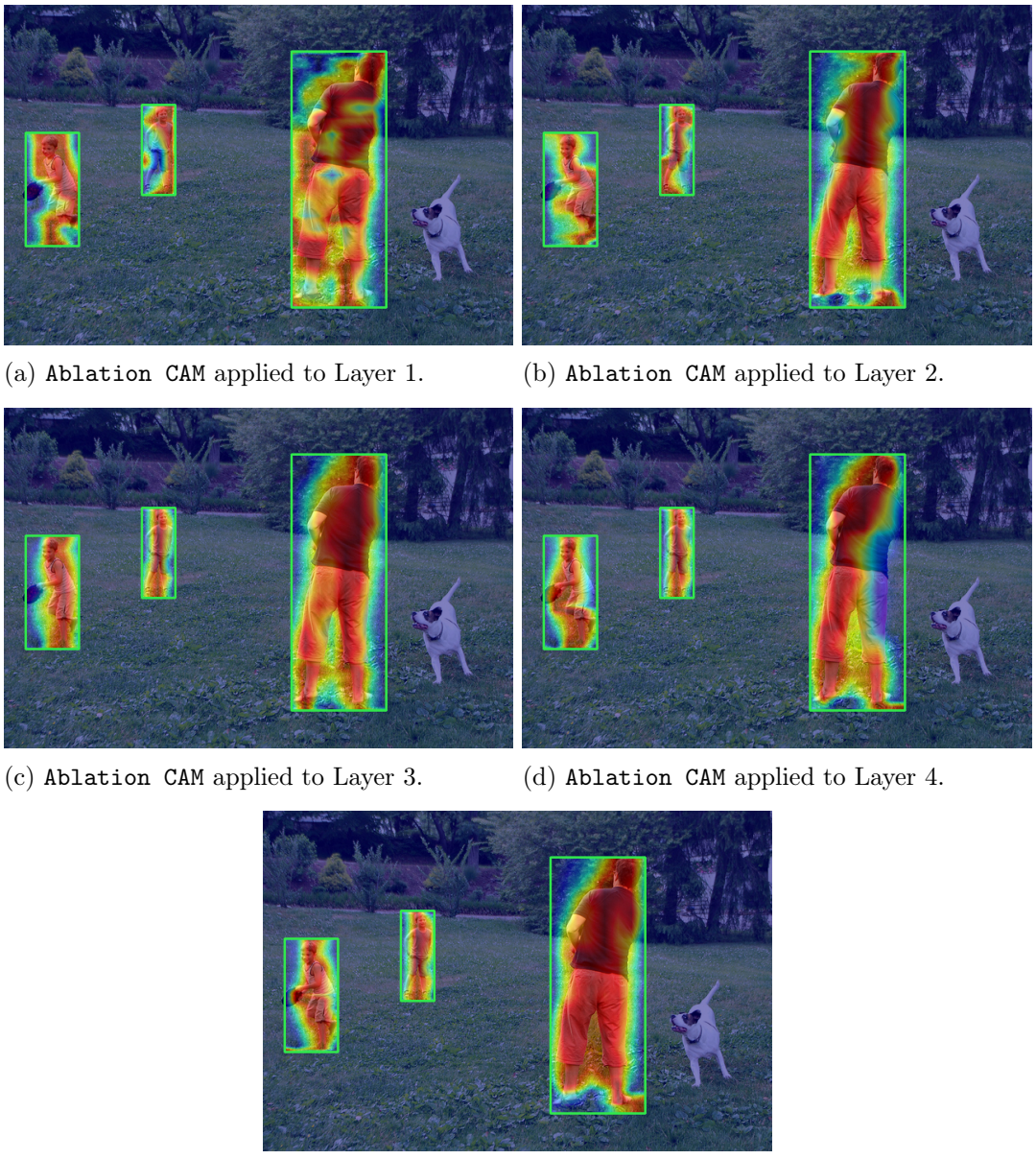
(c) Predictions using EfficientNetV2-M + BiFPN.

(d) Predictions using InternImage-S + CABiFPN.

(e) Predictions using ConvNeXt-S + CABiFPN.

(f) Predictions using EfficientNetV2-M + CABiFPN.

Figure 5.2: Object detection and instance segmentation predictions for image `ID 509403` based on evaluations of the six tested models.



(a) `Ablation CAM` applied to Layer 1.

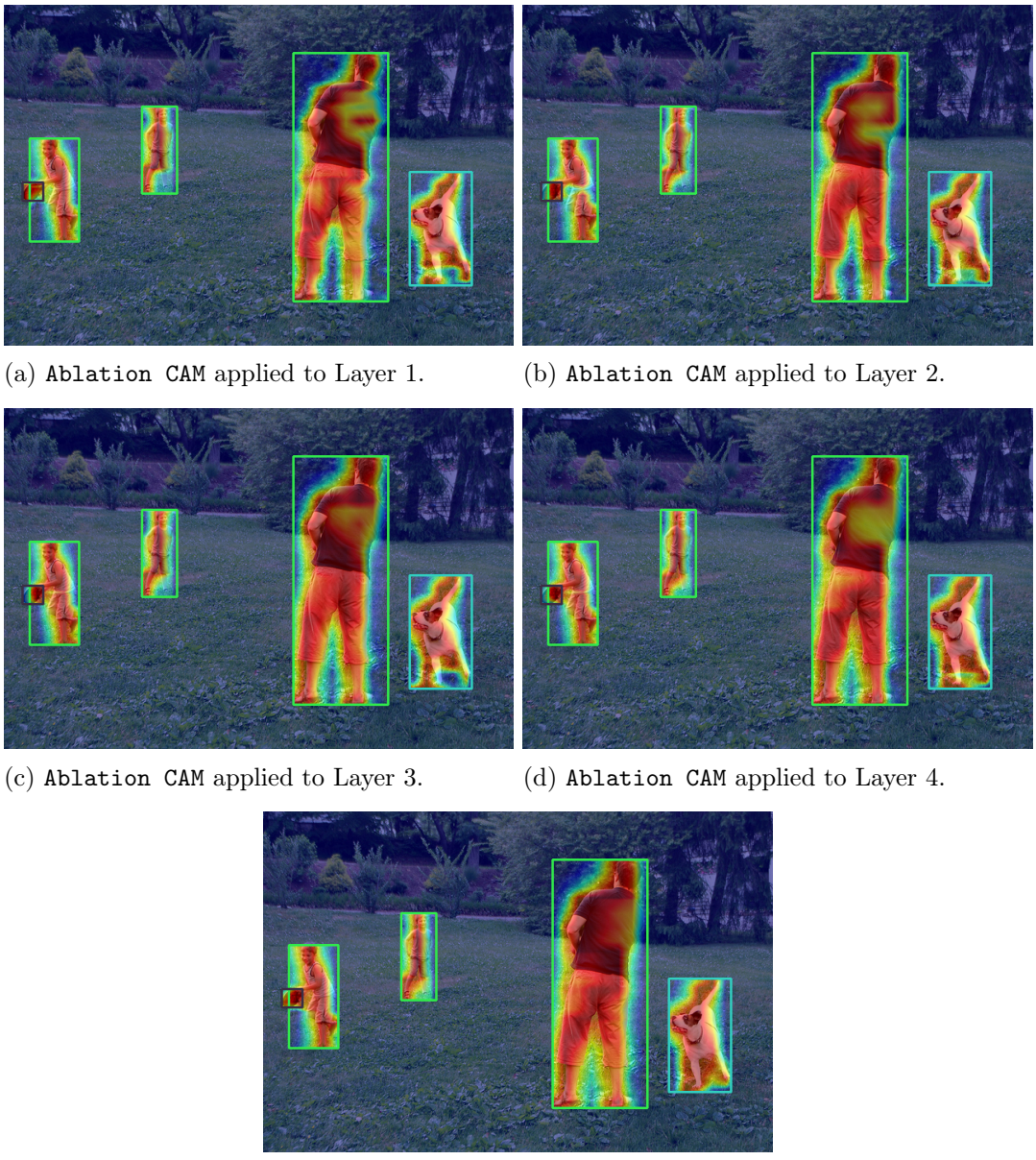
(b) `Ablation CAM` applied to Layer 2.

(c) `Ablation CAM` applied to Layer 3.

(d) `Ablation CAM` applied to Layer 4.

(e) `Ablation CAM` applied to Layer 5.

Figure 5.3: `Ablation CAM` applied to the five layers in the neck of the InternImage-S + BiFPN model, with corresponding object detection predictions for image `ID 509403`.



(a) `Ablation CAM` applied to Layer 1.

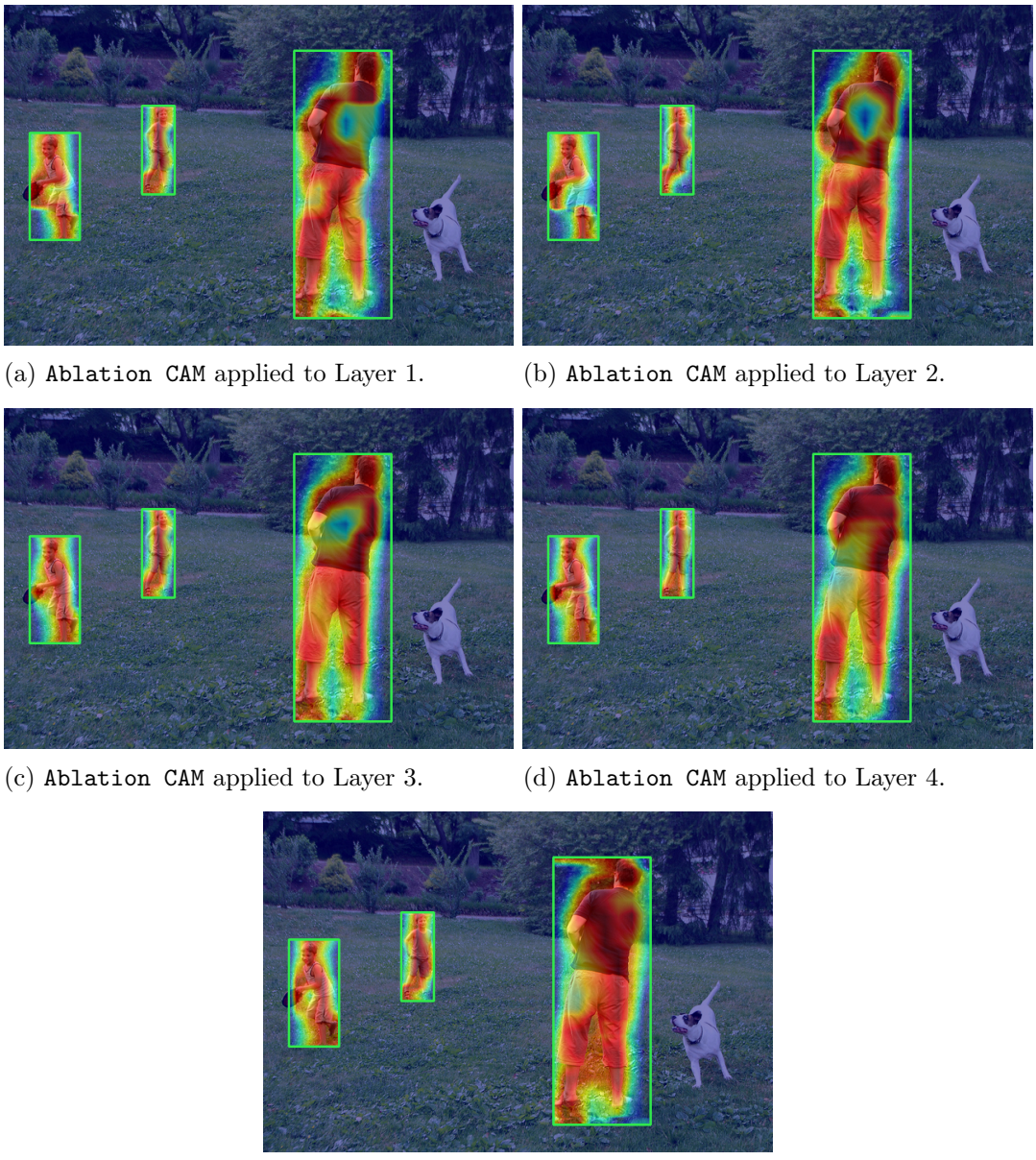
(b) `Ablation CAM` applied to Layer 2.

(c) `Ablation CAM` applied to Layer 3.

(d) `Ablation CAM` applied to Layer 4.

(e) `Ablation CAM` applied to Layer 5.

Figure 5.4: `Ablation CAM` applied to the five layers in the neck of the InternImage-S + CABiFPN model, with corresponding object detection predictions for image `ID 509403`.



(a) `Ablation CAM` applied to Layer 1.

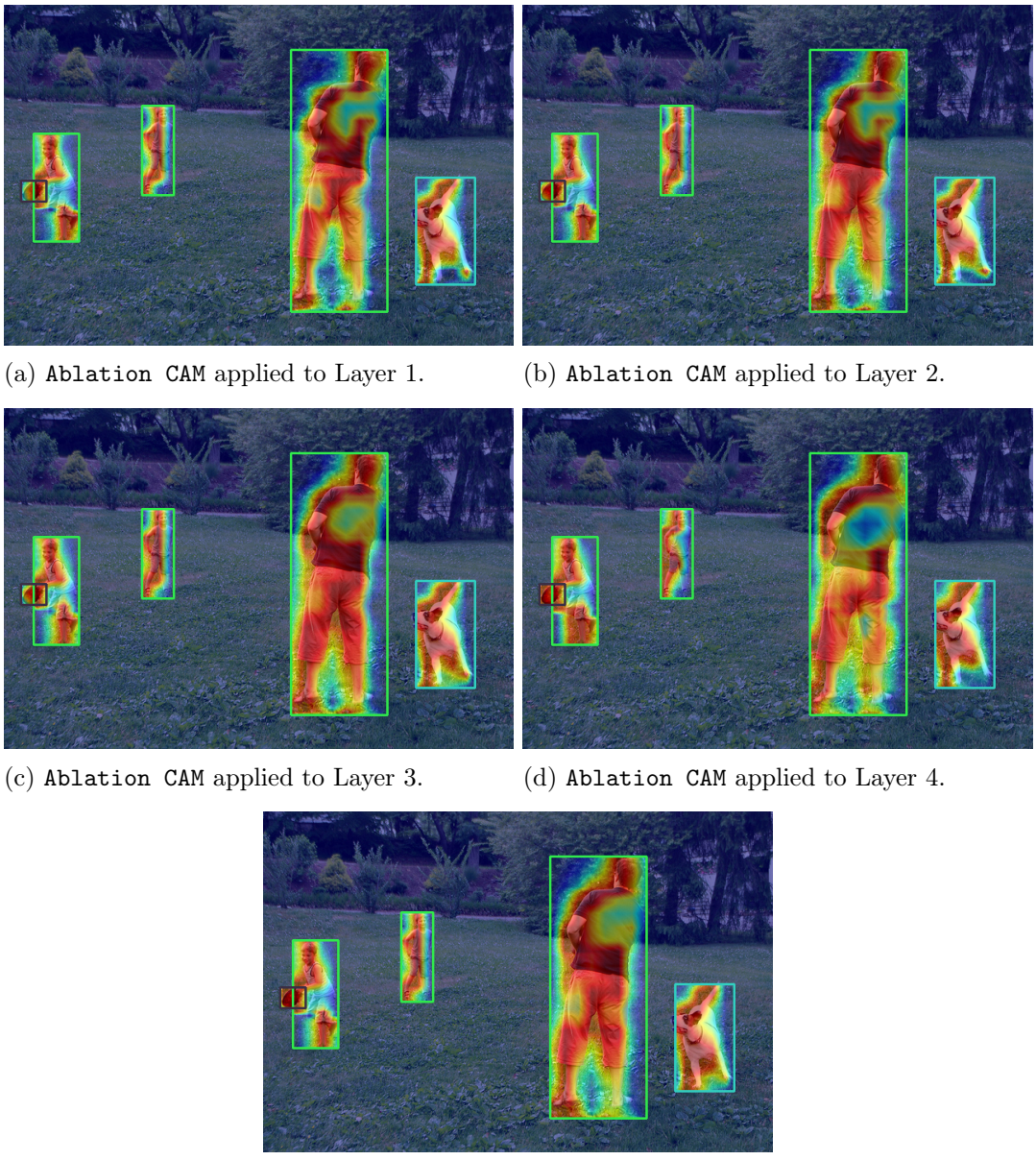
(b) `Ablation CAM` applied to Layer 2.

(c) `Ablation CAM` applied to Layer 3.

(d) `Ablation CAM` applied to Layer 4.

(e) `Ablation CAM` applied to Layer 5.

Figure 5.5: `Ablation CAM` applied to the five layers in the neck of the ConvNeXt-S + BiFPN model, with corresponding object detection predictions for image `ID 509403`.



(a) `Ablation CAM` applied to Layer 1.

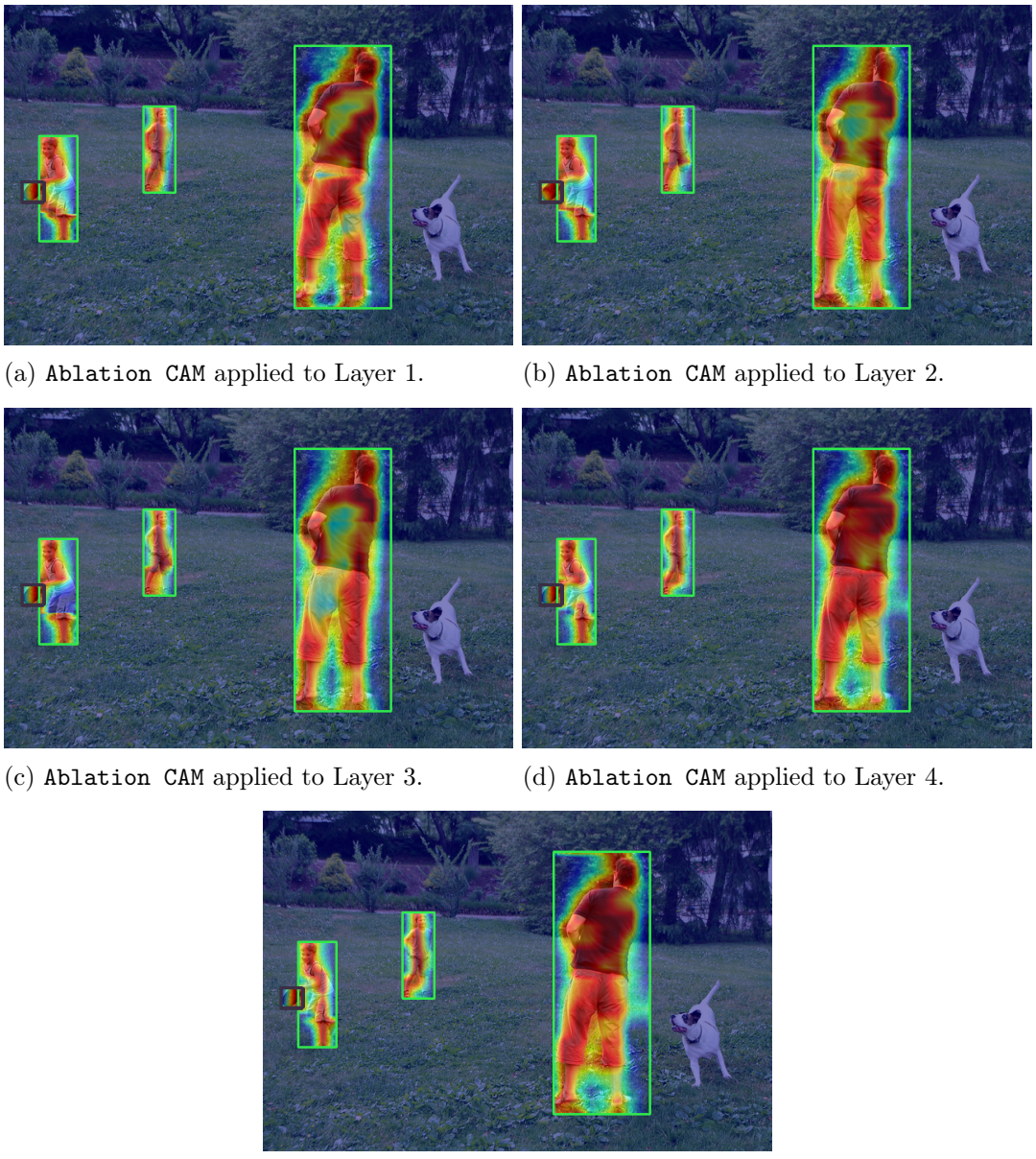
(b) `Ablation CAM` applied to Layer 2.

(c) `Ablation CAM` applied to Layer 3.

(d) `Ablation CAM` applied to Layer 4.

(e) `Ablation CAM` applied to Layer 5.

Figure 5.6: `Ablation CAM` applied to the five layers in the neck of the ConvNeXt-S + CABiFPN model, with corresponding object detection predictions for image `ID 509403`.



(a) `Ablation CAM` applied to Layer 1.

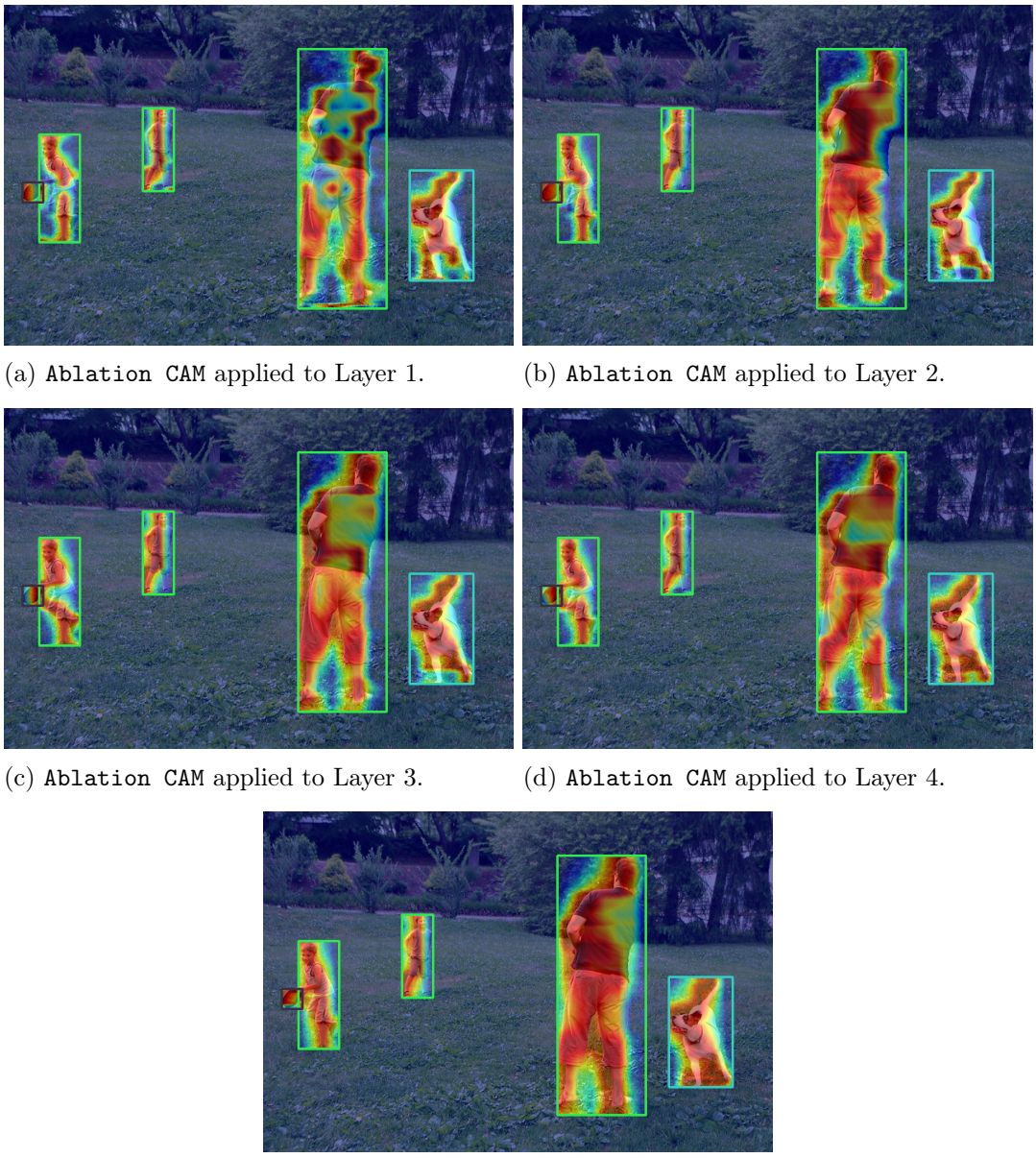
(b) `Ablation CAM` applied to Layer 2.

(c) `Ablation CAM` applied to Layer 3.

(d) `Ablation CAM` applied to Layer 4.

(e) `Ablation CAM` applied to Layer 5.

Figure 5.7: `Ablation CAM` applied to the five layers in the neck of the TF-EfficientNetV2-M + BiFPN model, with corresponding object detection predictions for image `ID 509403`.



(a) `Ablation CAM` applied to Layer 1.

(b) `Ablation CAM` applied to Layer 2.

(c) `Ablation CAM` applied to Layer 3.

(d) `Ablation CAM` applied to Layer 4.

(e) `Ablation CAM` applied to Layer 5.

Figure 5.8: `Ablation CAM` applied to the five layers in the neck of the EfficientNetV2-M + CABiFPN model, with corresponding object detection predictions for image `ID 509403`.

5.4 Imputation Results

To complement the insights gained from the Ablation Class Activation Mapping (CAM) visual explanations, we utilized the pixel-perturbation technique known as ROAD [THC+19]. This approach is detailed in Section 2.3.2 and was applied to each of the five layers in the neck across all six trained models for a comprehensive analysis. We considered two distinct strategies for pixel removal: Most Relevant First (MoRF) and Least Relevant First (LeRF) [SBM+15]. These strategies were crafted to either target the most relevant or least relevant pixels, thereby offering a more nuanced understanding of the model's decision-making processes.

To evaluate the impact of these perturbations on model performance, we utilized a custom confidence metric derived from both MoRF and LeRF. The equation for this metric is given by:

$$\text{Confidence Metric} = \frac{(LeRF - MoRF)}{2},$$

This metric was applied across multiple percentile thresholds, specifically at 20%, 40%, 60%, and 80% levels of pixel removal, to provide a multi-faceted view of model sensitivity to pixel perturbations.

The resultant data are tabulated and can be found in Table 5.5 for image ID `509403`, Table 5.6 for image ID `96825`, and Table 5.7 for image ID `171382`. These tables collectively present a comprehensive overview of how the imputation strategies impact the interpretability and reliability of the trained models under study.

	Confidence				
Model	**L1**	**L2**	**L3**	**L4**	**L5**
InternImage-S [WDC+23] + BiFPN [TPL20]	**2.842369**	2.6254954	2.6084025	1.4631767	*1.2592103*
ConvNeXt-S [LMW+22] + BiFPN [TPL20]	*2.5395045*	**2.8010774**	2.564676	2.7891173	2.7836413
EfficientNetV2-M [TL21] + BiFPN [TPL20]	3.8836775	3.9961786	4.127801	*3.7277918*	**4.129478**
InternImage-S [WDC+23] + CABiFPN (ours)	4.4633584	4.4853263	*4.432617*	**4.67398**	4.667679
ConvNeXt-S [LMW+22] + CABiFPN (ours)	**3.980652**	3.742455	3.9141197	3.9185126	*3.6784308*
EfficientNetV2-M [TL21] + CABiFPN (ours)	*3.916544*	3.9983945	4.111134	4.141871	**4.179732**

Table 5.5: Combined MoRF and LeRF avarage confidence using ROAD strategy for the five neck layers across six models, evaluated at percentiles 20, 40, 60, and 80 on image `ID 509403`. The highest values are in **bold**, indicating the most confident layer in the neck, while the lowest values are in *italic*.

Model	Confidence				
	L1	**L2**	**L3**	**L4**	**L5**
InternImage-S [WDC+23] + BiFPN [TPL20]	0.95029116	**0.95994985**	0.48607662	-0.24469143	*-0.48681438*
ConvNeXt-S [LMW+22] + BiFPN [TPL20]	**0.94896114**	0.93078464	0.93930924	0.933002	*0.9292339*
EfficientNetV2-M [TL21] + BiFPN [TPL20]	**0.9650256**	0.9558013	0.9592131	*0.95095545*	0.9558286
InternImage-S [WDC+23] + CABiFPN (ours)	*0.*	0.9606638	**0.969138**	0.9632113	0.96664834
ConvNeXt-S [LMW+22] + CABiFPN (ours)	*0.93161523*	**0.9501411**	0.9351118	0.9351068	0.93464565
EfficientNetV2-M [TL21] + CABiFPN (ours)	0.9696959	0.9621654	*0.9494796*	**0.97354376**	0.9564767

Table 5.6: Combined MoRF and LeRF avarage confidence using ROAD strategy for the five neck layers across six models, evaluated at percentiles 20, 40, 60, and 80 on image `ID 96825`. The highest values are in **bold**, indicating the most confident layer in the neck, while the lowest values are in *italic*.

Model	Confidence				
	L1	**L2**	**L3**	**L4**	**L5**
InternImage-S [WDC+23] + BiFPN [TPL20]	2.9169397	3.3592114	**3.4150054**	2.9640887	*1.8052661*
ConvNeXt-S [LMW+22] + BiFPN [TPL20]	*6.385799*	6.8318634	**7.1888366**	6.8878274	6.6595736
EfficientNetV2-M [TL21] + BiFPN [TPL20]	6.1930113	**6.560341**	6.082729	6.4222465	*5.624841*
InternImage-S [WDC+23] + CABiFPN (ours)	*7.4974775*	7.8757086	8.453202	**8.4683075**	8.111521
ConvNeXt-S [LMW+22] + CABiFPN (ours)	*5.323591*	5.753009	6.464163	6.36069	**6.803749**
EfficientNetV2-M [TL21] + CABiFPN (ours)	*6.8131247*	7.8556514	**8.355728**	7.8893404	8.054105

Table 5.7: Combined MoRF and LeRF avarage confidence using ROAD strategy for the five neck layers across six models, evaluated at percentiles 20, 40, 60, and 80 on image `ID 171382`. The highest values are in **bold**, indicating the most confident layer in the neck, while the lowest values are in *italic*.

Chapter 6

Discussion and Analysis

6.1 Metric Evaluation Analysis

Upon scrutiny of the aggregated data from the Table 5.4, it becomes evident that networks employing the proposed CABiFPN neck architecture demonstrated superior performance in both bounding box and mask predictions, when compared to those networks utilizing the conventional BiFPN neck. More specifically, the architecture that most proficiently predicted bounding boxes was a combination of EfficientNetV2-M [TL21] with CABiFPN. Conversely, the least effective architecture for this particular task was identified as InternImage-S [WDC+23] combined with BiFPN [TPL20]. In terms of mask prediction, the ConvNeXt-S network [LMW+22] paired with CABiFPN produced the most favorable outcomes, while the least effective results were similarly attributed to the InternImage-S network [WDC+23] with neck BiFPN [TPL20].

It is noteworthy to address the implications of architectural choices on model complexity. The integration of a greater number of parameters in the proposed CABiFPN neck substantially elevates the model's depth-related complexity as compared to its BiFPN counterpart. This suggests that a more expansive neck architecture, such as CABiFPN, manifests a higher degree of computational sophistication than a less expansive structure like BiFPN. However, this elevated complexity is accompanied by more intricate internal representations within the neck architecture. Consequently, this demands additional training time and computational resources, aligning with the observations put forth in previous work [Ben09].

6.2 Explanation with Ablation CAM

In an examination of the predictive capabilities of six computational models, variety of differentiated elements were noted with respect to object detection and mask generation in images with IDs `96825`, `171382`, and `509403`. These outcomes are elucidated in Figures 5.2, A.1, and A.2, showcasing key differences between models that implement

CABiFPN and BiFPN as their respective neck architectures.

Focusing initially on the predictions related to image ID `509403`, as presented in Figure 5.2, it was observed that the models equipped with the CABiFPN neck architecture successfully identified all bounding boxes for the object instances present (comprising three instances of `'person'`, one of `'dog'`, and one of `'frisbee'`). Conversely, models utilizing the BiFPN neck were less proficient, particularly in identifying objects with smaller surface areas such as instances of `'frisbee'` and `'dog'`. It is worth noting that both sets of models (those with CABiFPN and those with BiFPN neck architectures) were able to accurately generate masks for the aforementioned instances.

Turning to the predictive outcomes for image ID `96825`, as illustrated in Figure A.1, all models uniformly succeeded in identifying the mask and the associated bounding box for the `'person'` instance that occupied the largest area. However, the models collectively failed to detect smaller object instances, notably those labeled as `'backpack'` and `'skis'`. Additional scrutiny revealed that models incorporating InternImage-S + CABiFPN and ConvNeXt-S + CABiFPN architectures produced an increased number of segmentation artifacts.

Lastly, concerning the two most frequently occurring instances in image ID `171382` (delineated in Figure A.2) none of the models were able to precisely pinpoint the bounding box for the most prevalent instance, labeled as `'traffic light'`. Nevertheless, these models demonstrated a higher success rate in generating corresponding masks. With regard to the second most frequently instance, the `'person'` instances, models employing the CABiFPN neck architecture predominantly excelled in identifying both the bounding boxes and associated masks. Recurring issues were observed with the ConvNeXt-S + CABiFPN architecture, which continued to generate artifacts in segmentation tasks.

While visualizations of the predictions may initially clarify the behavior of the models, it is necessary to delve deeper into the visual explanation provided by the Ablation CAM method.

6.2.1 Ablation CAM visual explanation Image `509403`

Figures 5.3 and 5.4 present visualizations generated via Ablation CAM for models equipped with InternImage-S backbones and BiFPN and CABiFPN necks, respectively. A salient observation, corroborated by previous accuracy measurements, is the exclusive visualization of `'person'` instances by the model incorporating the BiFPN neck. Conversely, the model featuring the CABiFPN neck demonstrates the capability to identify multiple instances across classes. Specific attention is drawn to layer 1 of Figure 5.3, which belongs to the BiFPN neck model. Here, discriminative regions (marked in red) appear to be sporadically distributed across the instance, most prominently in

the larger 'person' instances. Such a pattern suggests that the BiFPN neck initially fails to leverage the substantial contributions that context features can offer. This limitation, however, appears to be ameliorated beginning with layer 2. In contrast, the CABiFPN neck model evidences the effective utilization of context features from layer 1 itself, accentuating elements derived from local context features, such as the limbs of 'person' instances or smaller objects.

In the context of the ConvNeXt-S + BiFPN and ConvNeXt-S + CABiFPN models, Figures 5.5 and 5.6 reveal that both networks initially exhibit a low scattered distribution of smaller sub-regions. This is particularly noticeable in the network employing the BiFPN neck. This low dispersion implies that the fusion of context features transpires earlier in models equipped with a ConvNeXt-S backbone as compared to those with an InternImage-S backbone. Nevertheless, the CABiFPN neck model continues to outperform in highlighting discriminative regions imbued with high-level context features, such as the limbs of 'person' instances or small objects like the 'frisbee' instance.

Lastly, the EfficientNetV2-M + BiFPN and EfficientNetV2-M + CABiFPN models, visualized in Figures 5.7 and 5.8, diverge from preceding findings. For the BiFPN neck model, the fusion of context features is apparent from layer 1, a feature not mirrored in the CABiFPN neck model where such fusion is discernible only in the innermost layers, ranging from layer 2 to layer 4. Notwithstanding these differences, both models converge towards highlighting similar discriminative sub-regions by Layer 5.

6.2.2 Ablation CAM visual explanation Image 96825

As mentioned in the begening of this section, image ID 96825 contains only three instances: 'person', 'backpack', and 'skis'. The first being the instance with the largest area in the image, and the last two with much smaller areas compared to the first instance.

In this case, the visualizations of the models with the InternImage-S backbone and the BiFPN and CABiFPN necks are illustrated in Figures A.3 and A.4, respectively. It can be observed and confirmed that the model with the BiFPN neck rapidly identifies the largest 'person' instance in the first three layers without major issues. However, in the last two layers, it can be observed that both layer 4 and layer 5 start to generate dispersed regions in the instance detection, indicating that these latter layers generate more artifacts compared to the first three. In contrast, in Figure A.4, it can be seen that the model with the CABiFPN neck identifies the instance homogeneously, despite not detecting it in the first layer. Again, it is noticeable that in this model, the details of the 'person' instances's limbs are better defined compared to the detections made by the model with the BiFPN neck.

Regarding the visualizations of the models with the ConvNeXt-S backbone, presented in Figures A.5 and A.6 for the BiFPN and CABiFPN necks, respectively, a

similar behavior can be observed as described in Section 6.2.1 for the same image, and as described above. A homogeneous distribution of regions of interest is observed across all five layers in both models with BiFPN and CABiFPN necks. The model with the CABiFPN neck achieves a more accurate identification of local context features of the instance's limbs in layer 2 compared to the model with the BiFPN neck.

A similar effect occurs in Figures A.7 and A.8 for the EfficientNetV2-M backbone, as all layers of both models with BiFPN and CABiFPN necks homogeneously identify the instance in their respective regions of interest, without showing significant representative differences in their visualization.

6.2.3 Ablation CAM visual explanation Image `171382`

In the present study, Image ID `171382` serves as a particularly challenging example for evaluating the efficacy of instance detection across six pre-trained models. This complexity arises from a confluence of factors such as the sheer number of instances, variability in their dimensions and geometries, or the degree of overlapping among them. Figures A.9 to A.14 provide visual representations to support our observations. Notably, it is exclusively the models incorporating the CABiFPN neck that demonstrate a superior ability in detecting a diverse array of instances, beyond the overrepresented `'person'` class.

In the specific of Figures A.9 and A.10, models employing both InternImage-S backbone and BiFPN and CABiFPN necks primarily identify instances of the `'person'` class. Yet, the model featuring the CABiFPN neck exhibits the capability to discern `'person'` instances across a range of scales. In contrast, the model with the BiFPN neck only effectively identifies `'person'` instances when they are confined to a similar area and pose. This suggests that the BiFPN-based model is more prone to neglecting underrepresented contextual features within the image.

Further scrutiny of Figures A.11 and A.12 reveals nuanced behavior from the ConvNeXt-S + BiFPN and ConvNeXt-S + CABiFPN models. Again, the model with the CABiFPN neck outperforms its BiFPN counterpart in terms of instance detection. Interestingly, the CABiFPN model initiates with dispersed focal regions in its initial layers, particularly noticeable in the `'person'` instance depicted performing a jumping gesture over a `'skateboard'` instance, unlike other `'person'` instances where the gestures are more homogenous.

Lastly, an examination of Figures A.13 and A.14, which correspond to the EfficientNetV2-M + BiFPN and EfficientNetV2-M + CABiFPN models, respectively, corroborates the aforementioned pattern. Here again, the CABiFPN model identifies a greater number of instances. However, it is only in the model's advanced layers that the fuse of high-level context features becomes more manifest, thereby accentuating key discriminative

regions such as the limbs of the `'person'` instances.

6.3 Validation of Visual Explanations through Imputation Techniques

As elaborated upon in Sections 2.3.2 and 5.4, a robust visual explanation is expected to result in an escalation of output confidence. To assess this, Tables 5.5, 5.6, and 5.7 present the variations in confidence scores for image IDs `509403`, `96825`, and `171382`, respectively. The employed strategy for this assessment is the Removal and Ordering Aware Decomposition (*ROAD*), in conjunction with Noisy Linear Imputation as the imputation operator $\mathcal{I}_l$ for the function $\mathcal{I}_l(\boldsymbol{M}, \boldsymbol{x}_l)$. Here, $\boldsymbol{M}$ represents the binary masks derived from the Ablation CAM visual explanations. The pixel elimination sequences applied are Least Relevant First (**LeRF**) and Most Relevant First (**MoRF**), as detailed in Section 2.3.2. Instances of these removal orders are illustrated in Figures A.15 and A.16.

It is crucial to highlight that a higher confidence score does not inherently imply superior model performance. An elevated confidence score solely corroborates that the obtained visual explanation's mask is in alignment with the pixel modification strategy and, by extension, adapts well to the focal visual fields under investigation across different layers.

For image ID `509403`, Table 5.5 corroborates that the initial layers offer more precise visual explanations in the InternImage-S + BiFPN model, thus substantiating the observations made in Section 6.2.1. In contrast, the InternImage-S + CABiFPN model demonstrates stable confidence values across all layers. Comparable stability in confidence values is also noted for both the ConvNeXt-S + BiFPN and ConvNeXt-S + CABiFPN models. For the EfficientNetV2-M + BiFPN and EfficientNetV2-M + CABiFPN models, it is unequivocally observed that the final layers produce the highest confidence scores. This observation reiterates the layer fusion occurrences in the ConvNeXt-S + CABiFPN model and further substantiates that the terminal layer contributes most substantially to the visual explanation in both BiFPN and CABiFPN neck configurations.

Concerning image ID `96825`, as per Table 5.6, the confidence value distribution across all models is largely homogenous, with a notable exception in the final two layers of the InternImage-S + BiFPN model. The latter possibly encompasses elements missed by the visual explanations, such as particular artifacts.

Finally, in the context of image ID `171382`, Table 5.7 exhibits trends akin to those observed for the InternImage-S + BiFPN model in Table 5.5. Specifically, the most

compelling confidence scores are registered in the initial layers, particularly within the first three layers of the network's neck architecture. This sharply contrasts with the elevated confidence values observed for the concluding layers in the InternImage-S + CABiFPN model, specifically layers 4 and 5. For the ConvNeXt-S + BiFPN model, the intermediate layers, specifically layers 2 to 4, yield the most refined visual explanations. Concurrently, the ConvNeXt-S + CABiFPN model records the apex of its confidence values in the final layer. In both EfficientNetV2-M + BiFPN and EfficientNetV2-M + CABiFPN configurations, it is discernible that the intermediate layers (layer 2 for the BiFPN model and layer 3 for the CABiFPN model) garner the most substantial confidence scores.

Chapter 7

Conclusion

This research undertakes a comprehensive analysis of feature fusion behaviors, particularly in the local and global context feature variables, within the specialized domains of object detection and instance segmentation. Central to this investigation is the development and implementation of the *Context Aggregation with Bi-Feature Pyramid Network*(CABiFPN), a novel fusion neck architecture designed to enhance object detection efficacy. Through a rigorous comparison with the conventional *Bi-Feature Pyramid Network* (BiFPN), the CABiFPN demonstrates markedly superior performance in various object detection tasks.

Utilizing visual explanation methodologies such as Ablation Class Activation Mapping (Ablation CAM) and imputation-based techniques, we substantiate that CABiFPN-enabled models exhibit a higher proficiency in recognizing and exploiting contextual features, thereby contributing to enhanced object detection and segmentation accuracies. Our empirical findings reveal that models incorporating the CABiFPN architecture surpass those utilizing the standard BiFPN architecture, particularly in the realm of bounding box and mask predictions. For example, the EfficientNetV2-M model outfitted with neck CABiFPN excels in bounding box predictions, whereas the ConvNeXt-S architecture coupled with neck CABiFPN yields superior outcomes in mask predictions. However, it should be noted that the increased number of parameters in CABiFPN introduces a higher level of computational complexity. While this confirms earlier research, the added complexity appears to be a contributing factor to the model's enhanced capability to capture intricate patterns.

Furthermore, the CABiFPN architecture manifests a robust capacity for identifying objects across a range of scales. Contrarily, models built upon the BiFPN neck encounter difficulties, especially when faced with objects possessing smaller surface areas. CABiFPN neck models exhibit a broader efficacy, identifying instances across diverse contexts and poses. In terms of feature fusion strategies, different models reveal heterogeneous approaches. For instance, when configured with both necks BiFPN and CABiFPN, the EfficientNetV2-M model displays distinct feature fusion behaviors in

its early and later layers. CABiFPN tends to capitalize more effectively on high-level contextual features compared to BiFPN. Despite these advances, certain limitations persist across all evaluated models and architectures. One universal issue is the suboptimal detection of objects with smaller surface areas, such as `'backpacks'` and `'skis'` particularly in cluttered or crowded scenes. Additionally, certain configurations like InternImage-S + CABiFPN and ConvNeXt-S + CABiFPN produce increased artifacts in mask segmentation tasks.

In summary, both the general and specific objectives of this research were fully realized. Quantitative findings were further bolstered through the utilization of Ablation CAM and imputation techniques, providing a nuanced understanding of variations in model performance.

7.1 Future Work

Several avenues for future research are delineated below:

- Future investigations should assess the feasibility of sibling training with different optimization algorithms, such as a 1x schedule training using AdamW and another employing Stochastic Gradient Descent with Warm Restarts [LH16].
- As discussed in Section 5.1.3, we opted to resize images from the COCO dataset to the original dimensions of 224×224 as used in the ImageNet dataset. A more comprehensive analysis should explore varying input sizes to understand the behavior of multi-scale variations in the images. Alternatively, one may choose to not resize the dataset, using the original COCO image dimensions.
- It is imperative to optimize the complexity of Context Aggregator Units and FMBConvCA Blocks. A potential approach might be to follow the research conducted by *Zhuang Liu et al.*, as discussed in their work on ConvNeXt [LMW+22] and outlined in Section 3.1.2.
- Future work should also evaluate the performance of CABiFPN architectures using more balanced datasets, such as Open Images [KDA+17].

Bibliography

[AAB+84] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden. 1984, Pyramid methods in image processing. *RCA Engineer*, 29(6):33–41, 1984.

[Arn11] Jörg Arndt. *Matters Computational*. Springer Berlin Heidelberg, 2011.

[Ben09] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends® in Machine Learning*, 2(1):1–127, 2009.

[BIK+20] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020.

[BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[BMR+20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

[CSHB17] Aditya Chattopadhyay, Anirban Sarkar, Prantik Howlader, and Vineeth N. Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. *CoRR*, abs/1710.11063, 2017.

[DBK+21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[DGO+20] Yimian Dai, Fabian Gieseke, Stefan Oehmcke, Yiquan Wu, and Kobus Barnard. Attentional feature fusion. *CoRR*, abs/2009.14082, 2020.

[DR20] Saurabh Desai and Harish G. Ramaswamy. Ablation-cam: Visual explanations for deep convolutional network via gradient-free localization. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 972–980, 2020.

[DT05] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005.

[DV16] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *ArXiv e-prints*, mar 2016.

[FHD+20] Ruigang Fu, Qingyong Hu, Xiaohu Dong, Yulan Guo, Yinghui Gao, and Biao Li. Axiom-based grad-cam: Towards accurate visualization and explanation of cnns. *CoRR*, abs/2008.02312, 2020.

[Fuk80] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, April 1980.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[GBK22] Wenchao Gu, Shuang Bai, and Lingxing Kong. A review on 2d instance segmentation based on deep neural networks. *Image and Vision Computing*, 120:104401, 2022.

[Gc21] Jacob Gildenblat and contributors. Pytorch library for cam methods. `https://github.com/jacobgil/pytorch-grad-cam`, 2021.

[GDDM13] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[Gir15] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[GLJ18] Qishuo Gao, Samsung Lim, and Xiuping Jia. Hyperspectral image classification using convolutional neural networks and multiple feature learning. *Remote Sensing*, 10(2), 2018.

[GLPL19] Golnaz Ghiasi, Tsung-Yi Lin, Ruoming Pang, and Quoc V. Le. NAS-FPN: learning scalable feature pyramid architecture for object detection. *CoRR*, abs/1904.07392, 2019.

[GT] Suyog Gupta and Mingxing Tan. Efficientnet-edgetpu: Creating accelerator-optimized neural networks with automl.

[HAGM14] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 297–312, Cham, 2014. Springer International Publishing.

[HG23] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023.

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[HLVDMW17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017.

[HSS17] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *CoRR*, abs/1709.01507, 2017.

[HYH+23] Ali Hatamizadeh, Hongxu Yin, Greg Heinrich, Jan Kautz, and Pavlo Molchanov. Global context vision transformers, 2023.

[Jac01] Jaccard, Paul. Étude comparative de la distribution florale dans une portion des alpes et du jura. 1901.

[Jac12] Paul Jaccard. THE DISTRIBUTION OF THE FLORA IN THE ALPINE ZONE.1. *New Phytologist*, 11(2):37–50, February 1912.

[JO21] Hyungsik Jung and Youngrock Oh. Towards better explanations of class activation mapping. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1336–1344, October 2021.

[KB17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[KDA+17] Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Shahab Kamali, Matteo Malloci, Jordi Pont-Tuset, Andreas Veit, Serge Belongie, Victor Gomes, Abhinav Gupta, Chen Sun, Gal Chechik, David Cai, Zheyun Feng, Dhyanesh Narayanan, and Kevin Murphy. Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://storage.googleapis.com/openimages/web/index.html*, 2017.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[LAJ15] Dana Lahat, Tülay Adali, and Christian Jutten. Multimodal data fusion: An overview of methods, challenges, and prospects. *Proceedings of the IEEE*, 103(9):1449–1477, 2015.

[LBBH98] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LBD+89] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[LDG+17] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[LH16] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983, 2016.

[LLC+21] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021.

[LMB+14] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[LMW+22] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s, 2022.

[LQQ+18] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. *CoRR*, abs/1803.01534, 2018.

[LRB15] Wei Liu, Andrew Rabinovich, and Alexander C. Berg. Parsenet: Looking wider to see better. *CoRR*, abs/1506.04579, 2015.

[MBP+22] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3523–3542, 2022.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[RHGS15] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[RLB+22] Yao Rong, Tobias Leemann, Vadim Borisov, Gjergji Kasneci, and Enkelejda Kasneci. Evaluating feature attribution: An information-theoretic perspective. *CoRR*, abs/2202.00449, 2022.

[SBM+15] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Bach, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *CoRR*, abs/1509.06321, 2015.

[SD17] Bharat Singh and Larry S. Davis. An analysis of scale invariance in object detection - SNIP. *CoRR*, abs/1711.08189, 2017.

[SDV+16] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016.

[SHZ+18] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018.

[TCD+21] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention, 2021.

[THC+19] Richard Tomsett, Dan Harborne, Supriyo Chakraborty, Prudhvi Gurram, and Alun D. Preece. Sanity checks for saliency metrics. *CoRR*, abs/1912.01451, 2019.

[TL19] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 09–15 Jun 2019.

[TL21] Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training, 2021.

[TPL19] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection. *CoRR*, abs/1911.09070, 2019.

[TPL20] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection, 2020.

[UL19] Md Azher Uddin and Young-Koo Lee. Feature fusion of deep spatial features and handcrafted spatiotemporal features for human action recognition. *Sensors*, 19(7), 2019.

[UvdSGS13] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, April 2013.

[VJ01] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, 2001.

[VJ04] Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, May 2004.

[VSP+23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[WDC+23] Wenhai Wang, Jifeng Dai, Zhe Chen, Zhenhang Huang, Zhiqi Li, Xizhou Zhu, Xiaowei Hu, Tong Lu, Lewei Lu, Hongsheng Li, Xiaogang Wang, and Yu Qiao. Internimage: Exploring large-scale vision foundation models with deformable convolutions, 2023.

[WDYZ19] Haofan Wang, Mengnan Du, Fan Yang, and Zijian Zhang. Score-cam: Improved visual explanations via score-weighted class activation mapping. *CoRR*, abs/1910.01279, 2019.

[WXL+22] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. PVT v2: Improved baselines with pyramid vision transformer. *Computational Visual Media*, 8(3):415–424, mar 2022.

[XGD+17] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5987–5995, 2017.

[YCBL14] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *CoRR*, abs/1411.1792, 2014.

[ZAA+22] Syed Sahil Abbas Zaidi, Mohammad Samar Ansari, Asra Aslam, Nadia Kanwal, Mamoona Asghar, and Brian Lee. A survey of modern deep learning based object detection models. *Digital Signal Processing*, 126:103514, June 2022.

[ZC88] Yi-Tong Zhou and Rama Chellappa. Computation of optical flow using a neural network. *IEEE 1988 International Conference on Neural Networks*, pages 71–78 vol.2, 1988.

[ZCS+23] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *Proceedings of the IEEE*, 111(3):257–276, March 2023.

[ZF13] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

[ZHLD18] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results, 2018.

[ZKHB22] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers, 2022.

[ZKL+15] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. *CoRR*, abs/1512.04150, 2015.

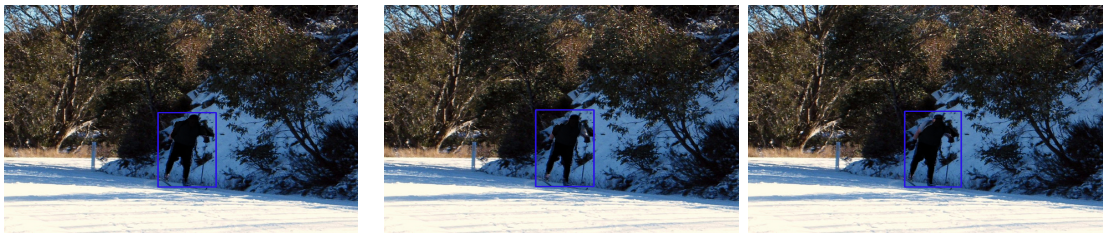
[ZL16] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *CoRR*, abs/1611.01578, 2016.

[ZVSL17] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017.

Appendix A

Supplementary Figures and Tables

A.1 Predictions from models



(a) Predictions using InternImage-S + BiFPN.

(b) Predictions using ConvNeXt-S + BiFPN.

(c) Predictions using EfficientNetV2-M + BiFPN.

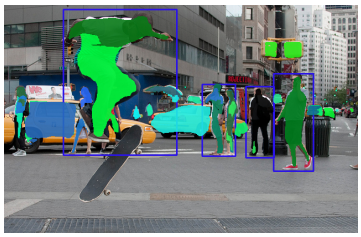


(d) Predictions using InternImage-S + CABiFPN.

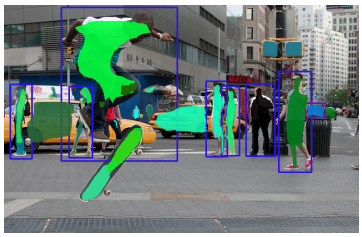
(e) Predictions using ConvNeXt-S + CABiFPN.

(f) Predictions using EfficientNetV2-M + CABiFPN.

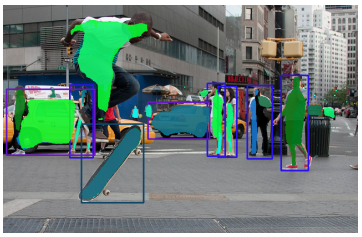
Figure A.1: Object detection and instance segmentation predictions for image `ID 96825` based on evaluations of the six tested models.



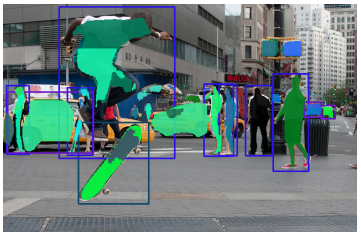
(a) Predictions using InternImage-S + BiFPN.



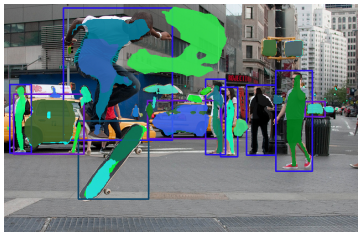
(b) Predictions using ConvNeXt-S + BiFPN.



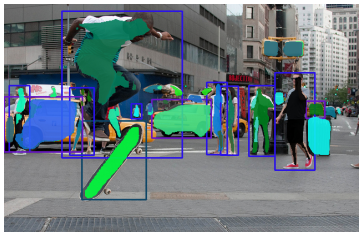
(c) Predictions using EfficientNetV2-M + BiFPN.



(d) Predictions using InternImage-S + CABiFPN.



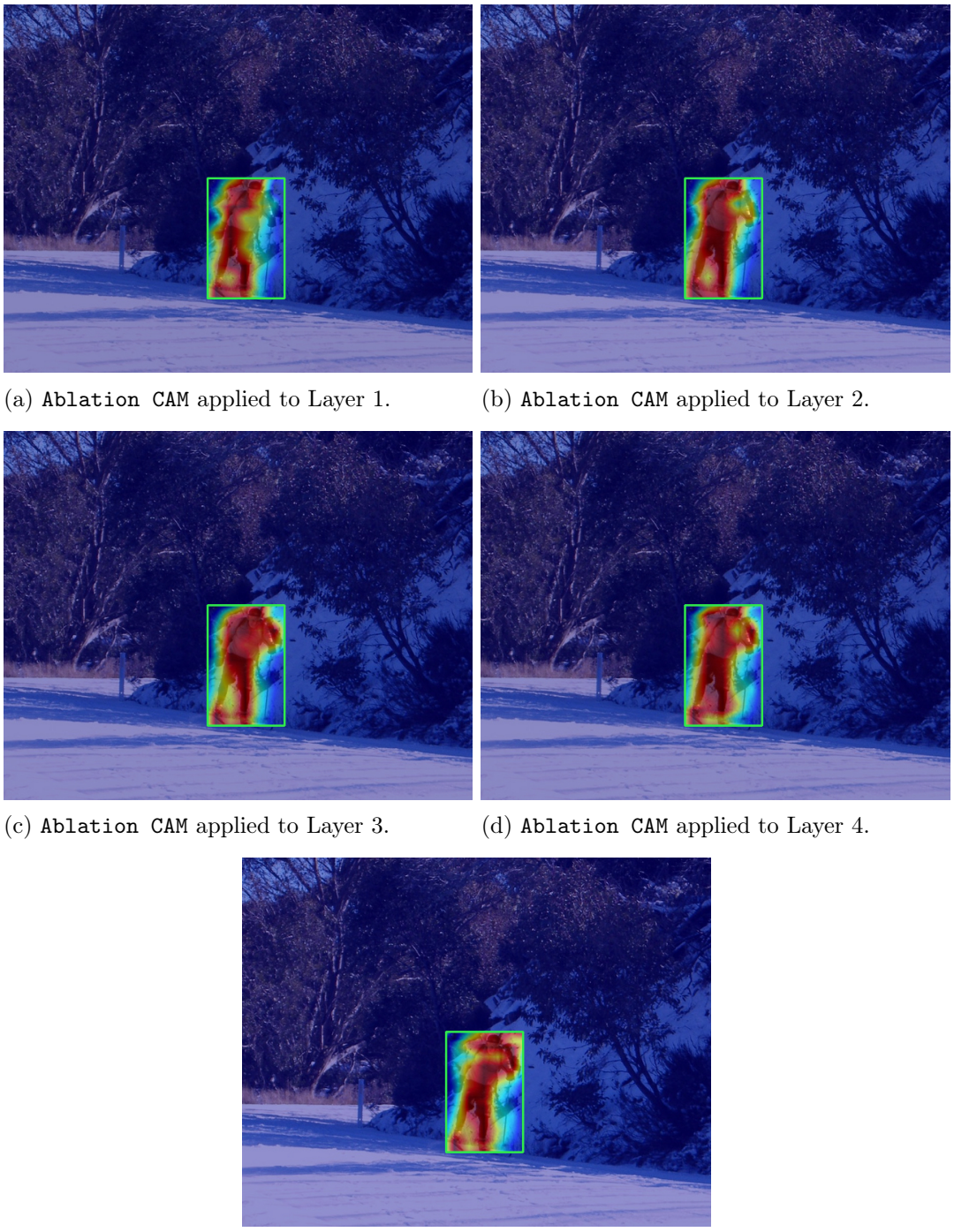
(e) Predictions using ConvNeXt-S + CABiFPN.



(f) Predictions using EfficientNetV2-M + CABiFPN.

Figure A.2: Object detection and instance segmentation predictions for image `ID 171382` based on evaluations of the six tested models.

A.2 `Ablation CAM` over models



(a) `Ablation CAM` applied to Layer 1.

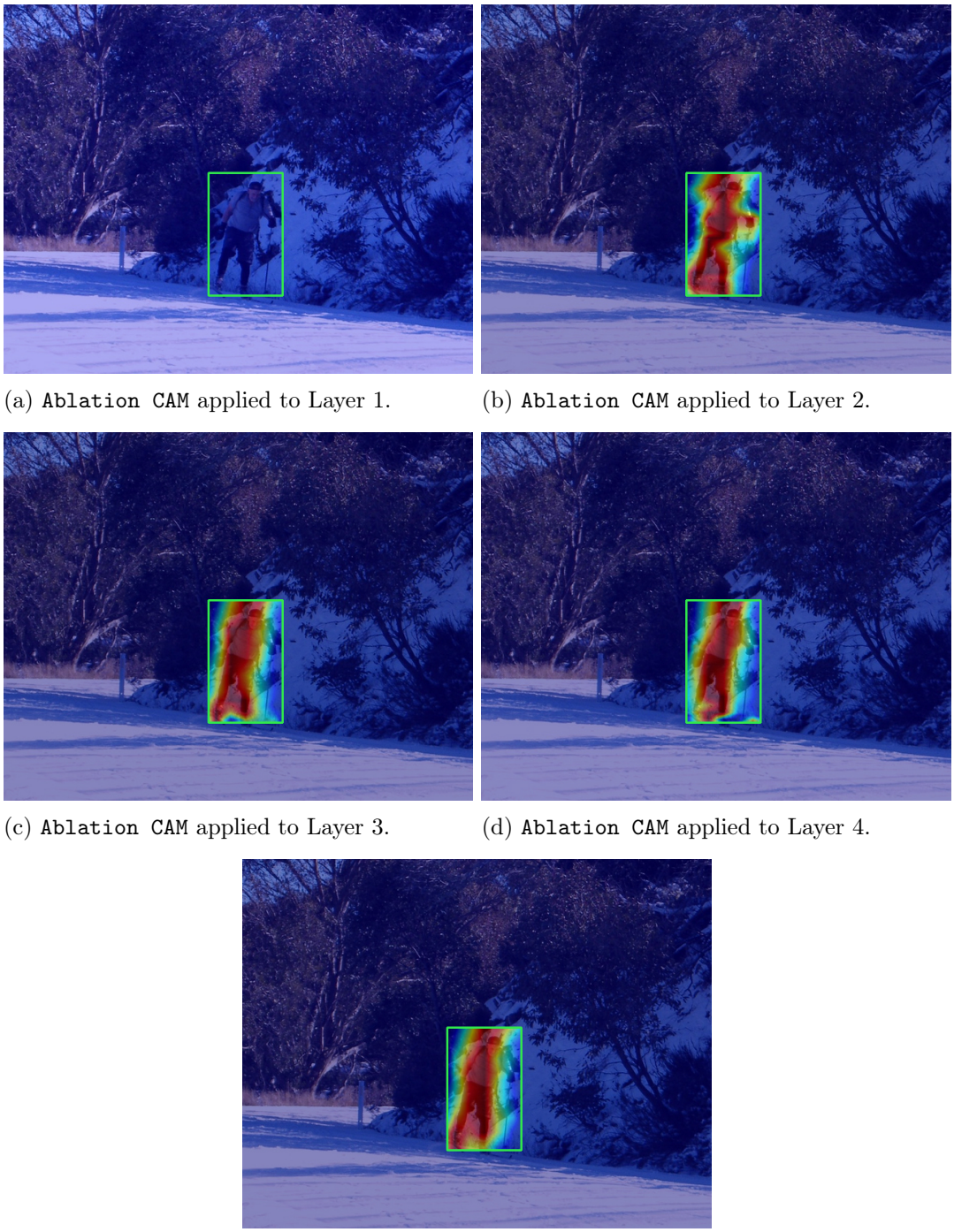
(b) `Ablation CAM` applied to Layer 2.

(c) `Ablation CAM` applied to Layer 3.

(d) `Ablation CAM` applied to Layer 4.

(e) `Ablation CAM` applied to Layer 5.

Figure A.3: `Ablation CAM` applied to the five layers in the neck of the InternImage-S + BiFPN model, with corresponding object detection predictions for image `ID 96825`.



(a) `Ablation CAM` applied to Layer 1.

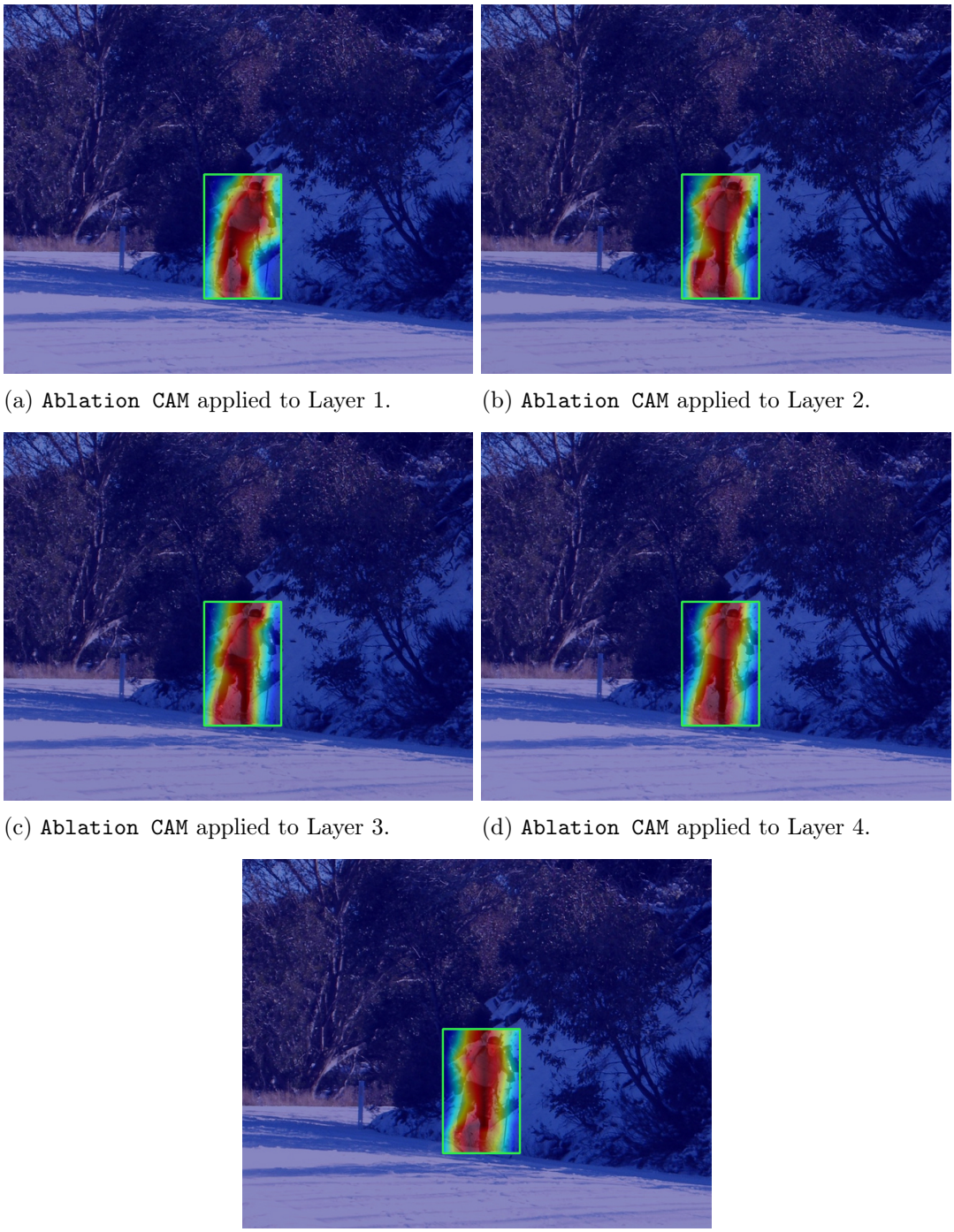
(b) `Ablation CAM` applied to Layer 2.

(c) `Ablation CAM` applied to Layer 3.

(d) `Ablation CAM` applied to Layer 4.

(e) `Ablation CAM` applied to Layer 5.

Figure A.4: `Ablation CAM` applied to the five layers in the neck of the InternImage-S + CABiFPN model, with corresponding object detection predictions for image `ID 96825`.



(a) `Ablation CAM` applied to Layer 1.

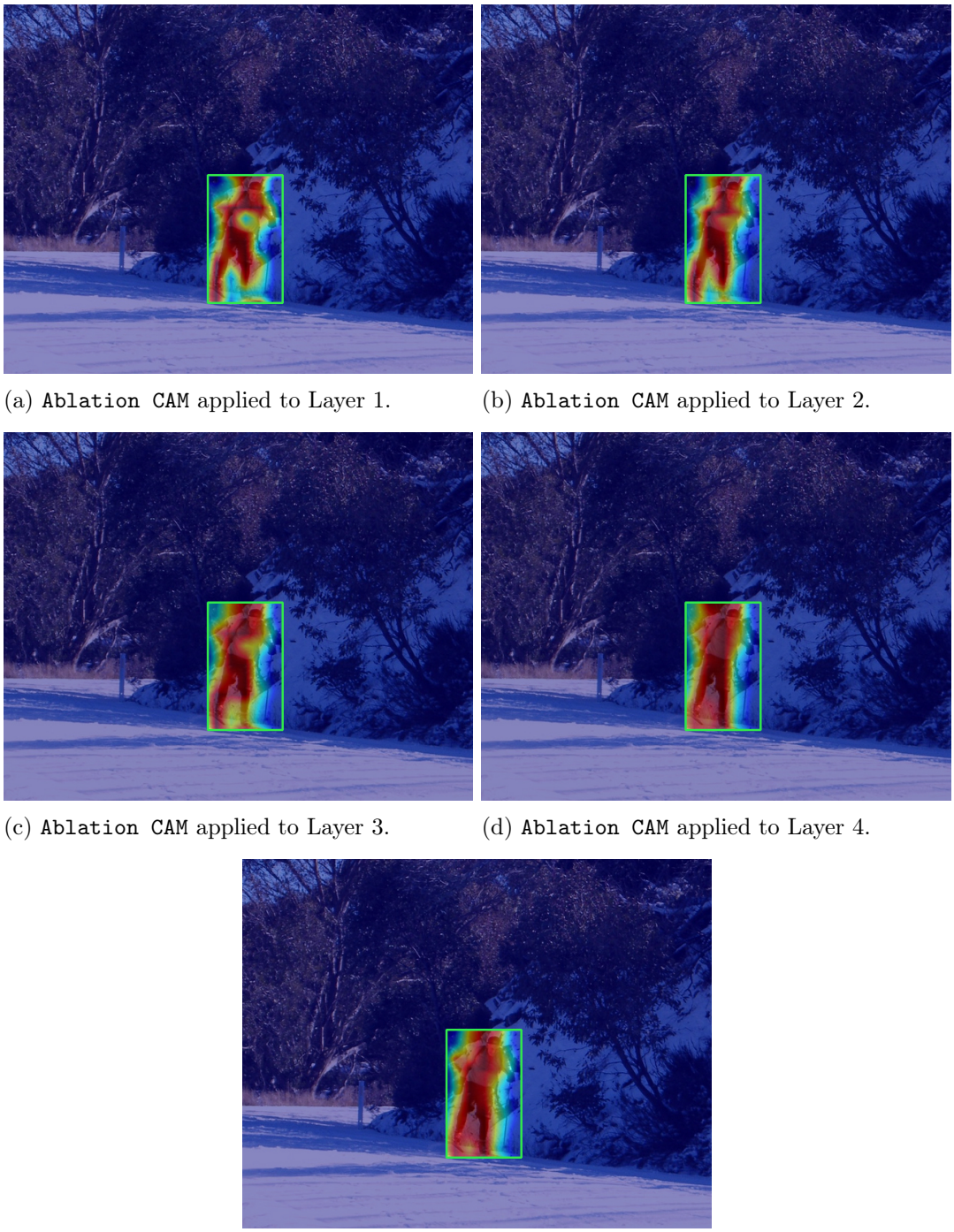
(b) `Ablation CAM` applied to Layer 2.

(c) `Ablation CAM` applied to Layer 3.

(d) `Ablation CAM` applied to Layer 4.

(e) `Ablation CAM` applied to Layer 5.

Figure A.5: `Ablation CAM` applied to the five layers in the neck of the ConvNeXt-S + BiFPN model, with corresponding object detection predictions for image `ID 96825`.



(a) `Ablation CAM` applied to Layer 1.

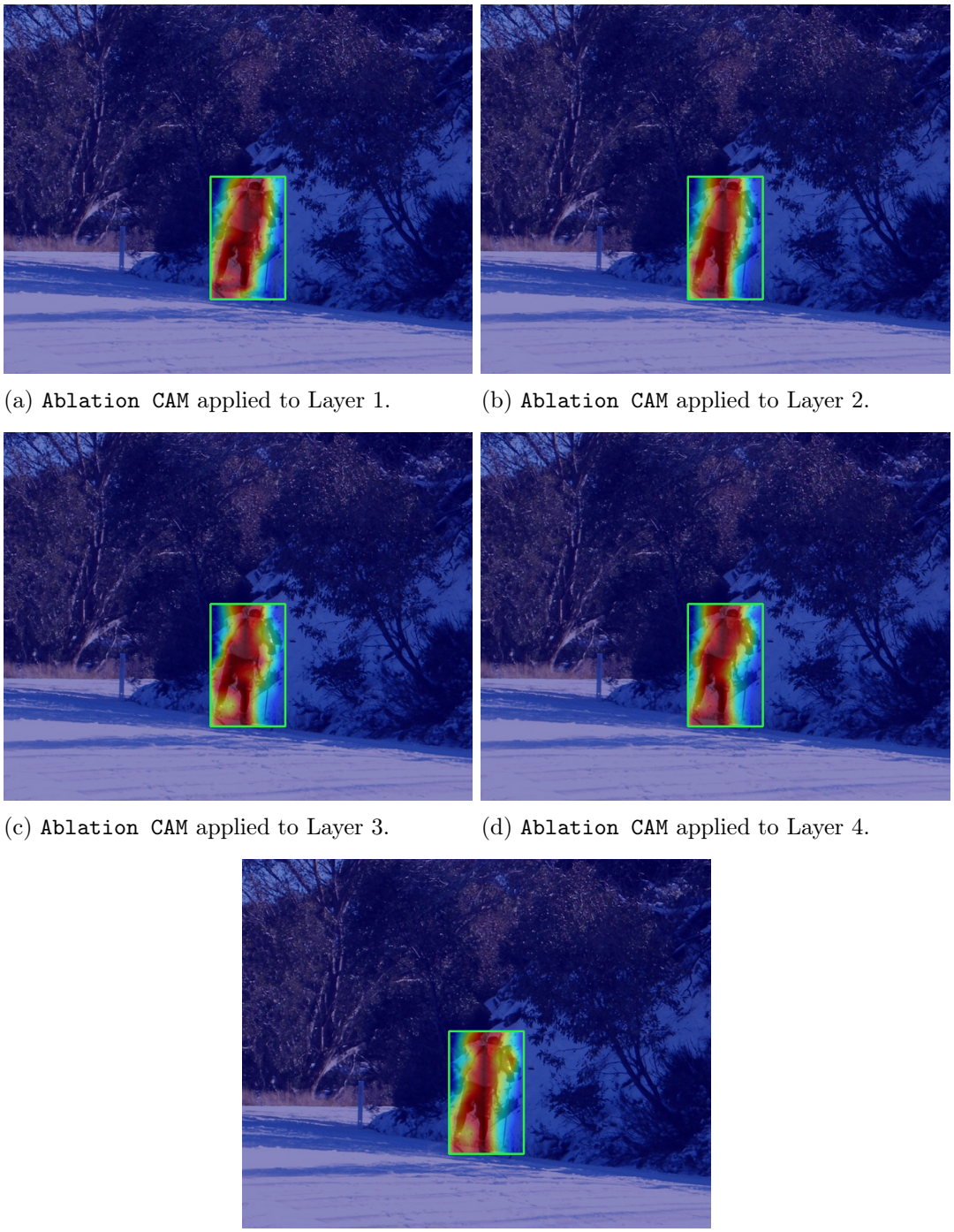
(b) `Ablation CAM` applied to Layer 2.

(c) `Ablation CAM` applied to Layer 3.

(d) `Ablation CAM` applied to Layer 4.

(e) `Ablation CAM` applied to Layer 5.

Figure A.6: `Ablation CAM` applied to the five layers in the neck of the ConvNeXt-S + CABiFPN model, with corresponding object detection predictions for image `ID 96825`.



(a) `Ablation CAM` applied to Layer 1.

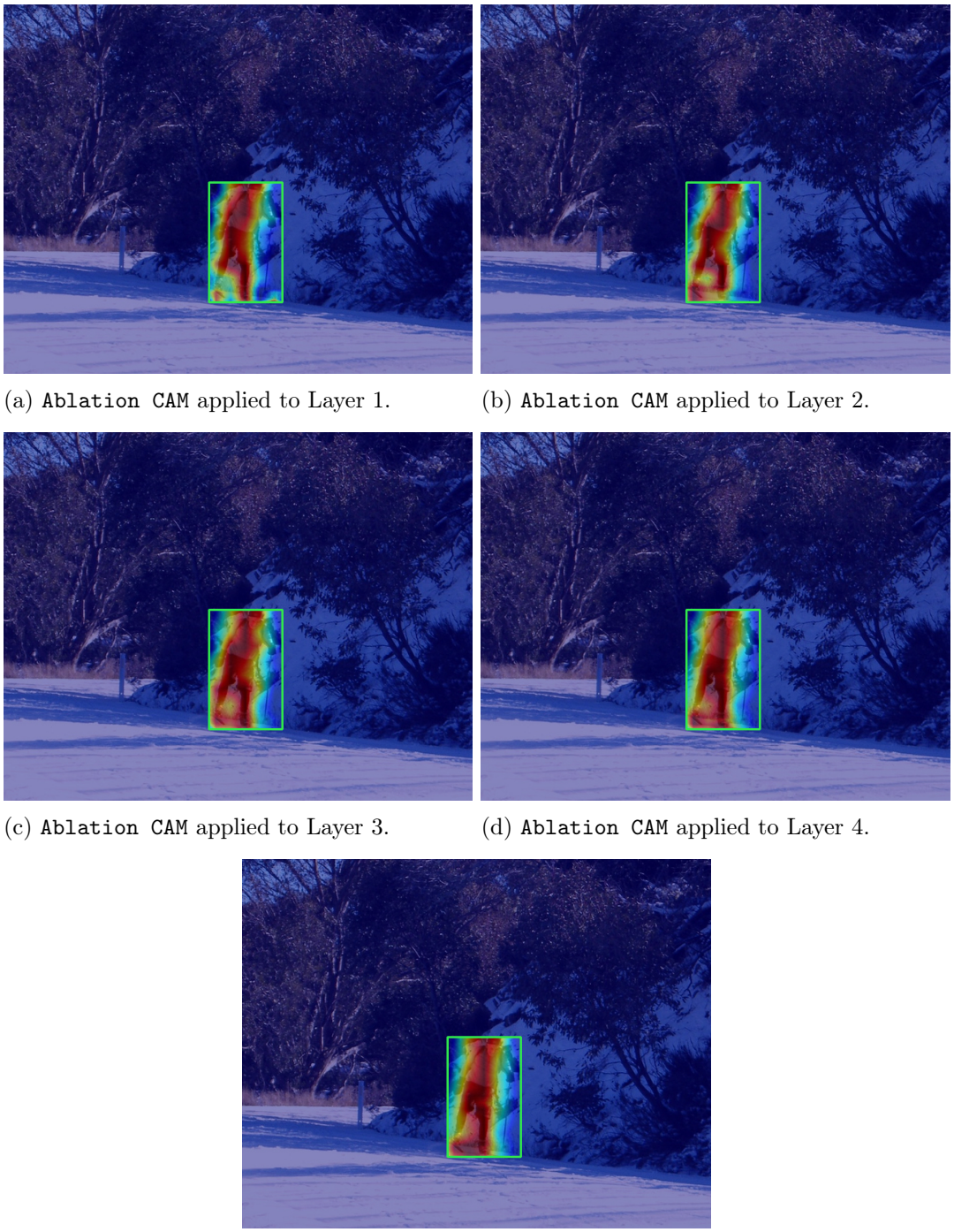
(b) `Ablation CAM` applied to Layer 2.

(c) `Ablation CAM` applied to Layer 3.

(d) `Ablation CAM` applied to Layer 4.

(e) `Ablation CAM` applied to Layer 5.

Figure A.7: `Ablation CAM` applied to the five layers in the neck of the EfficientNetV2-M + BiFPN model, with corresponding object detection predictions for image `ID 96825`.



(a) `Ablation CAM` applied to Layer 1.

(b) `Ablation CAM` applied to Layer 2.

(c) `Ablation CAM` applied to Layer 3.

(d) `Ablation CAM` applied to Layer 4.

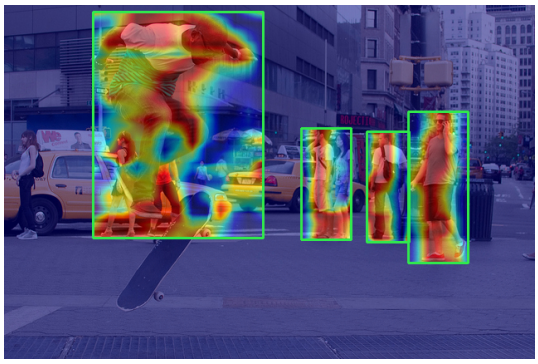
(e) `Ablation CAM` applied to Layer 5.

Figure A.8: `Ablation CAM` applied to the five layers in the neck of the EfficientNetV2-M + CABiFPN model, with corresponding object detection predictions for image `ID 96825`.

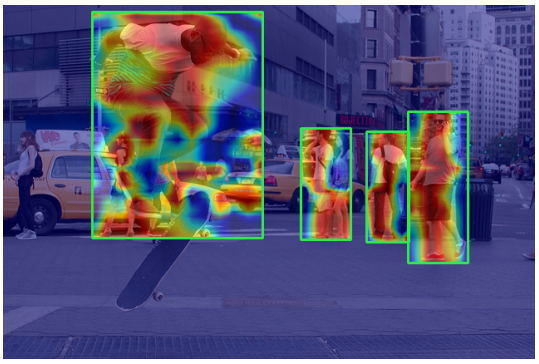


(a) `Ablation CAM` applied to Layer 1.

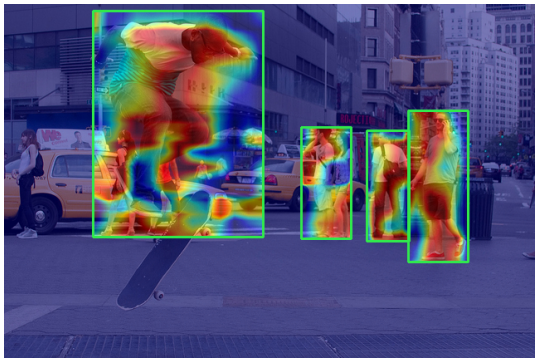
(b) `Ablation CAM` applied to Layer 2.



(c) `Ablation CAM` applied to Layer 3.

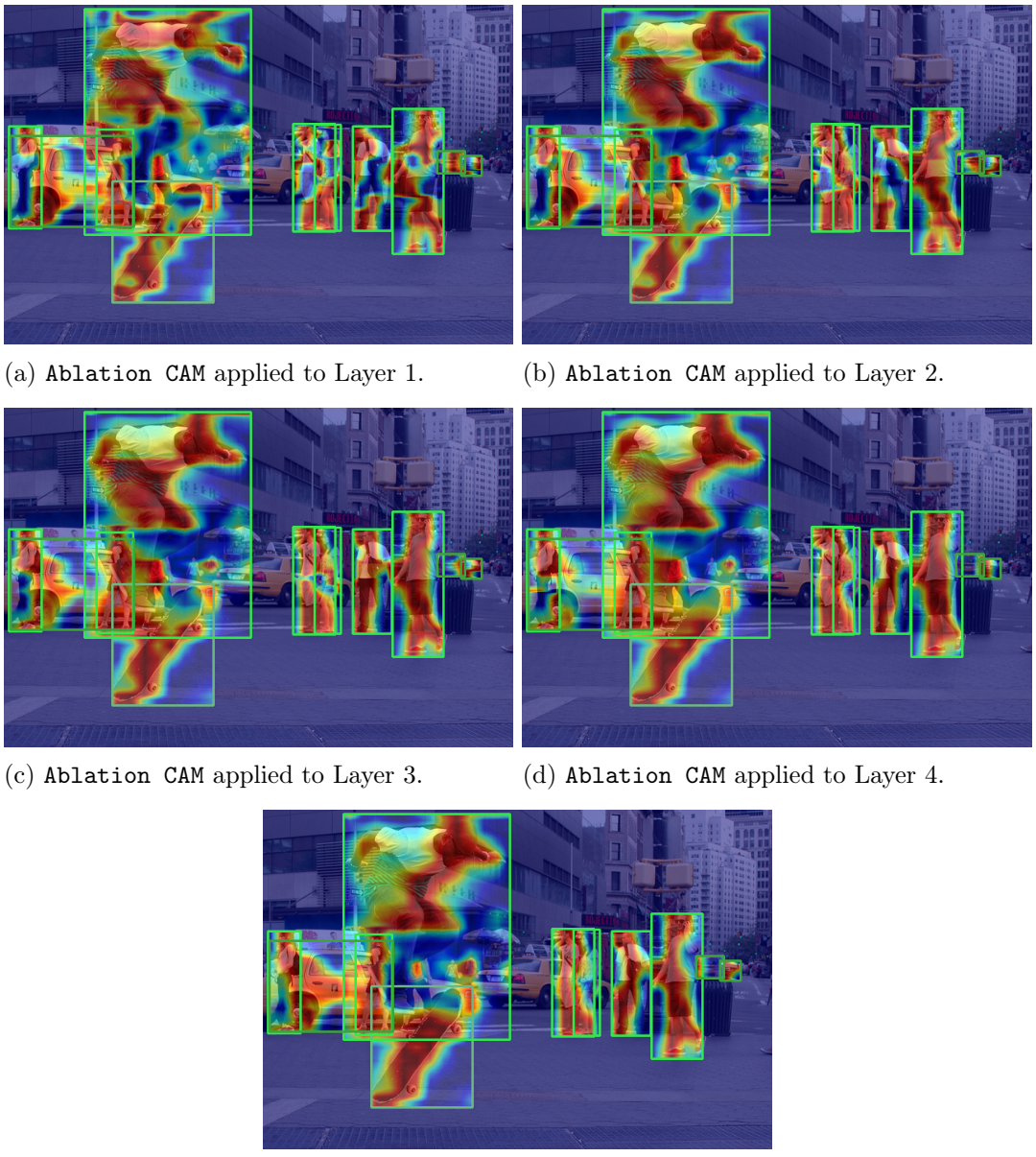


(d) `Ablation CAM` applied to Layer 4.



(e) `Ablation CAM` applied to Layer 5.

Figure A.9: `Ablation CAM` applied to the five layers in the neck of the InternImage-S + BiFPN model, with corresponding object detection predictions for image `ID 171382`.



(a) `Ablation CAM` applied to Layer 1.

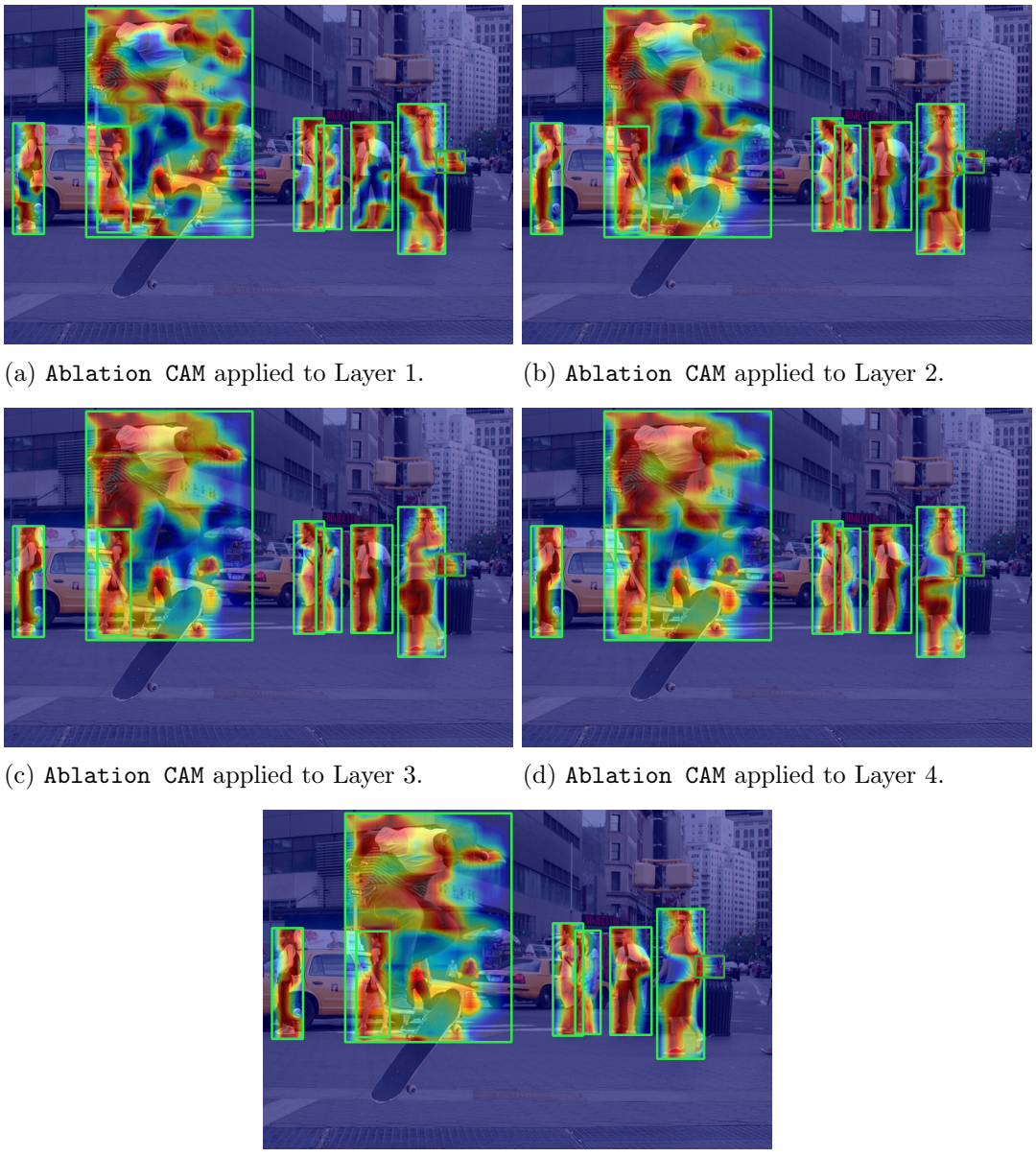
(b) `Ablation CAM` applied to Layer 2.

(c) `Ablation CAM` applied to Layer 3.

(d) `Ablation CAM` applied to Layer 4.

(e) `Ablation CAM` applied to Layer 5.

Figure A.10: `Ablation CAM` applied to the five layers in the neck of the InternImage-S + CABiFPN model, with corresponding object detection predictions for image `ID 171382`.



(a) `Ablation CAM` applied to Layer 1.

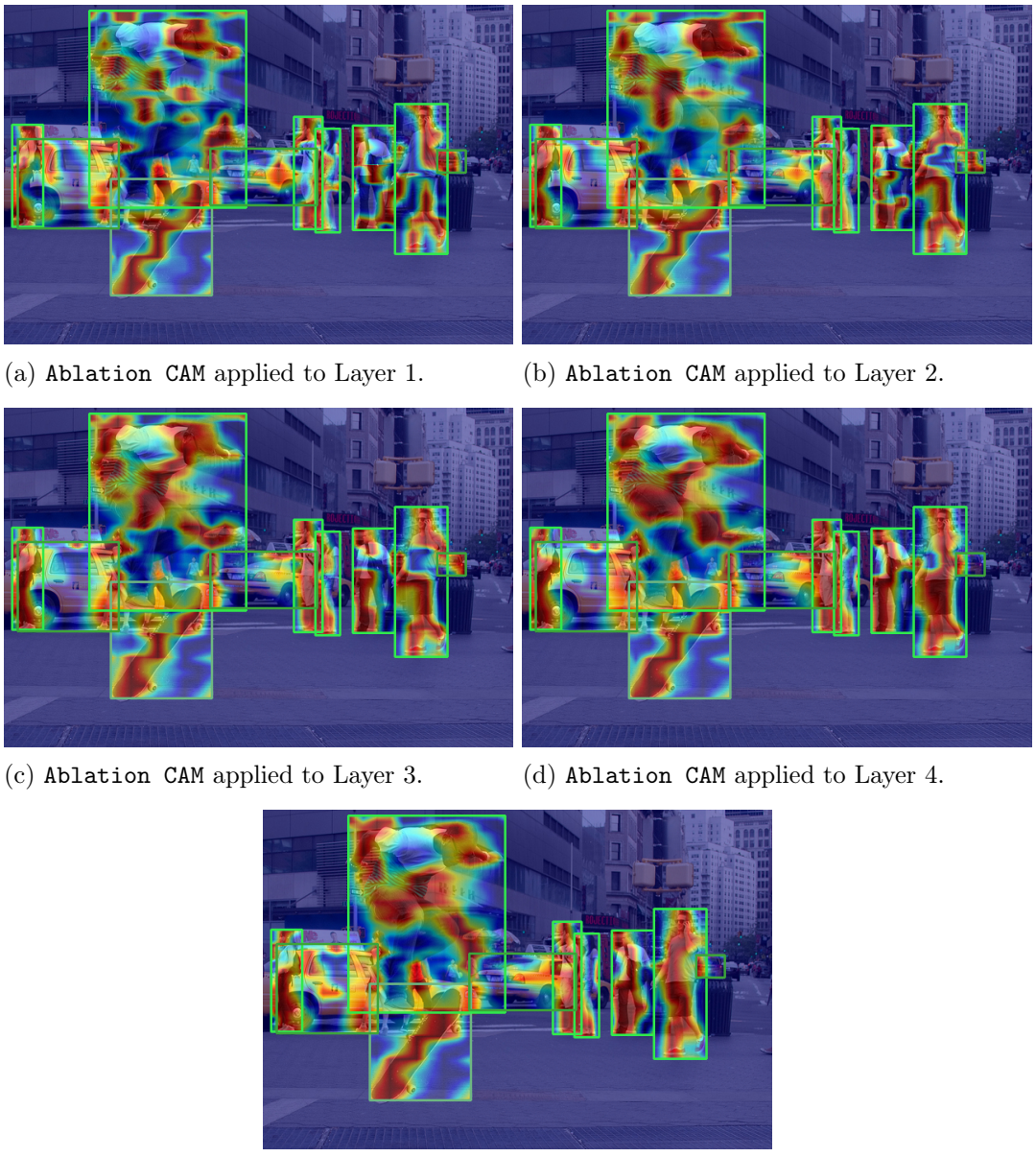
(b) `Ablation CAM` applied to Layer 2.

(c) `Ablation CAM` applied to Layer 3.

(d) `Ablation CAM` applied to Layer 4.

(e) `Ablation CAM` applied to Layer 5.

Figure A.11: `Ablation CAM` applied to the five layers in the neck of the ConvNeXt-S + BiFPN model, with corresponding object detection predictions for image `ID 171382`.



(a) `Ablation CAM` applied to Layer 1.

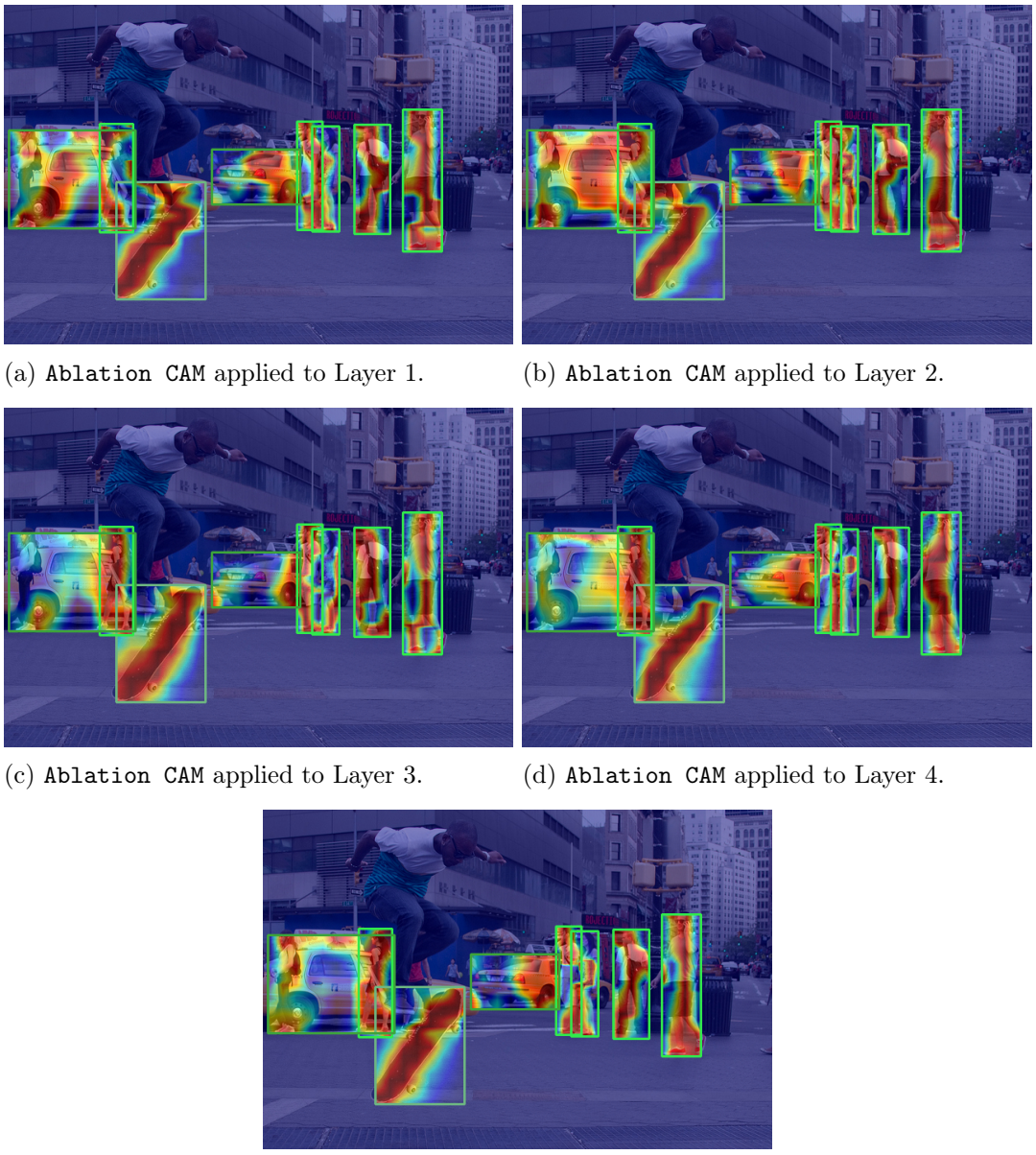
(b) `Ablation CAM` applied to Layer 2.

(c) `Ablation CAM` applied to Layer 3.

(d) `Ablation CAM` applied to Layer 4.

(e) `Ablation CAM` applied to Layer 5.

Figure A.12: `Ablation CAM` applied to the five layers in the neck of the ConvNeXt-S + CABiFPN model, with corresponding object detection predictions for image `ID 171382`.



(a) `Ablation CAM` applied to Layer 1.

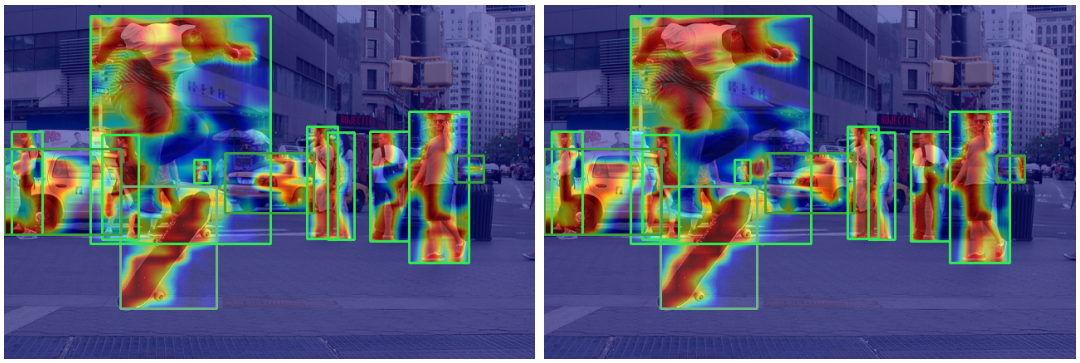
(b) `Ablation CAM` applied to Layer 2.

(c) `Ablation CAM` applied to Layer 3.

(d) `Ablation CAM` applied to Layer 4.

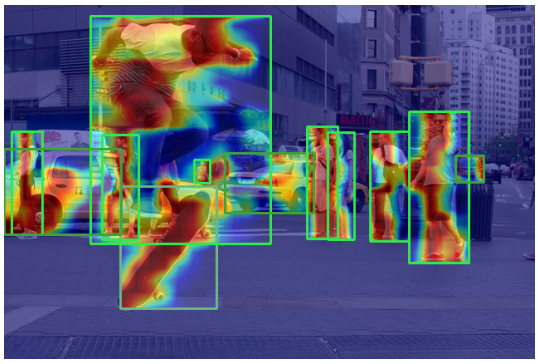
(e) `Ablation CAM` applied to Layer 5.

Figure A.13: `Ablation CAM` applied to the five layers in the neck of the EfficientNetV2-M + BiFPN model, with corresponding object detection predictions for image `ID 171382`.

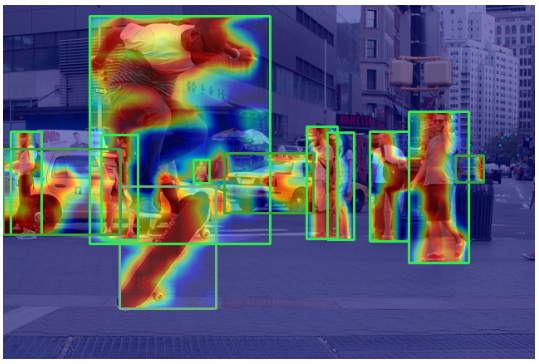


(a) `Ablation CAM` applied to Layer 1.

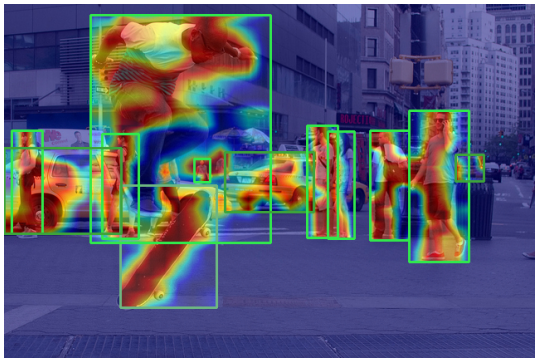
(b) `Ablation CAM` applied to Layer 2.



(c) `Ablation CAM` applied to Layer 3.



(d) `Ablation CAM` applied to Layer 4.



(e) `Ablation CAM` applied to Layer 5.

Figure A.14: `Ablation CAM` applied to the five layers in the neck of the EfficientNetV2-M + CABiFPN model, with corresponding object detection predictions for image `ID 171382`.

A.3 Pixel Removal with ROAD strategy



Figure A.15: Visualization of layers 1, 3, and 5 in the neck for the six tested models after applying the ROAD Least Relevant First method and removing 20% of the image (at the 80th percentile). Evaluated using image ID 171382.

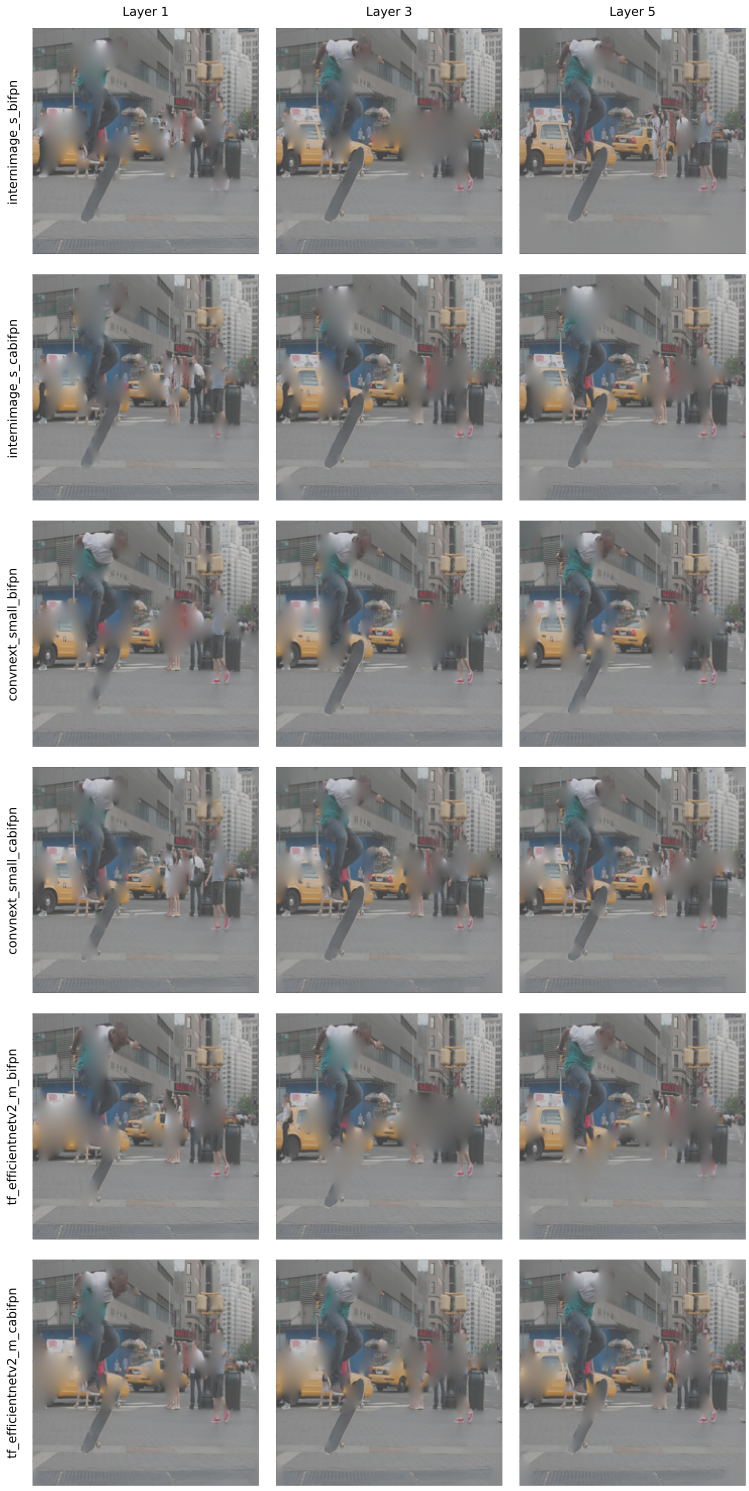


Figure A.16: Visualization of layers 1, 3, and 5 in the neck for the six tested models after applying the ROAD Most Relevant First method and removing 20% of the image (at the 80th percentile). Evaluated using image `ID 171382`.

List of Figures

2.1 Fully connected layer. 6
2.2 Neighbourhood Convolutional Layer 6
2.3 Example of a convolutional operation involving an input $\boldsymbol{x}$ and a kernel $\boldsymbol{y}$. 7
2.4 Convolving a 3×3 kernel over a 4×4 input using unit strides. 7
2.5 **MoRF** (Most Relevant First) and **LeRF** (Least Relevant First) pixel removal sequences utilized by the **ROAD** approach 14

3.1 Diagram illustrating the architecture of an end-to-end detector, featuring a *backbone*, a *neck*, and a *head* for detection functions. 15
3.2 *InternImage* architecture 17
3.3 Block design for a *ConvNeXt*. Image adapted from [WDC+23]. 18
3.4 Diagram illustrating the architecture of MBConv and Fused-MBConv, as presented in [TL21]. 19
3.5 Diagram illustrating the architecture FPN, as presented in [LDG+17]. . 20
3.6 Diagram illustrating the PANet architecture and its associated proposed fusion feature block, as detailed in [LQQ+18]. 22
3.7 Diagram illustrating the interconnection between backbone stages and the NAS-FPN search spaces. 23
3.8 Diagram showcasing the merging cell utilized within the search space, further elaborated in [GLPL19]. 23
3.9 The diagram depicts various architecture graphs of NAS-FPN within the search space. 24
3.10 Diagram illustrating of BiFPN as the feature network. Image adapted from [WDC+23]. 26
3.11 The object detection framework as delineated by R-CNN. Image adapted from [GDDM13]. 27
3.12 An overview of the Fast R-CNN architecture, as sourced from [Gir15]. . 28
3.13 Faster R-CNN represents a unified network dedicated to object detection. 29
3.14 The architecture of the Region Proposal Network (RPN). Image and caption sourced from [RHGS15]. 30
3.15 The architecture of Mask R-CNN, as sourced from *Kaiming He et al.* [HGDG17]. 31
3.16 RoIAlign design. 32

4.1 Detailed schematic of the CA-BiFPN architecture 34
4.2 Diagram illustrating the Context Aggregator Units and merge unit node FMBConvCA, proposed in this work. 36

5.1 Original images from `COCO2017` utilized for testing. 44
5.2 Object detection and instance segmentation predictions for image `ID 509403` based on evaluations of the six tested models. 45
5.3 `Ablation CAM` applied to the five layers in the neck of the InternImage-S + BiFPN model. 46
5.4 `Ablation CAM` applied to the five layers in the neck of the InternImage-S + CABiFPN model. 47
5.5 `Ablation CAM` applied to the five layers in the neck of the ConvNeXt-S + BiFPN model. 48
5.6 `Ablation CAM` applied to the five layers in the neck of the ConvNeXt-S + CABiFPN model. 49
5.7 `Ablation CAM` applied to the five layers in the neck of the TF-EfficientNetV2-M + BiFPN model. 50
5.8 `Ablation CAM` applied to the five layers in the neck of the EfficientNetV2-M + CABiFPN model. 51

A.1 Object detection and instance segmentation predictions for image `ID 96825`. 70
A.2 Object detection and instance segmentation predictions for image `ID 171382`. 71
A.3 `Ablation CAM` applied to the five layers in the neck of the InternImage-S + BiFPN model. 73
A.4 `Ablation CAM` applied to the five layers in the neck of the InternImage-S + CABiFPN model. 74
A.5 `Ablation CAM` applied to the five layers in the neck of the ConvNeXt-S + BiFPN model. 75
A.6 `Ablation CAM` applied to the five layers in the neck of the ConvNeXt-S + CABiFPN model. 76
A.7 `Ablation CAM` applied to the five layers in the neck of the EfficientNetV2-M + BiFPN model. 77
A.8 `Ablation CAM` applied to the five layers in the neck of the EfficientNetV2-M + CABiFPN model. 78
A.9 `Ablation CAM` applied to the five layers in the neck of the InternImage-S + BiFPN model. 79
A.10 `Ablation CAM` applied to the five layers in the neck of the InternImage-S + CABiFPN model. 80
A.11 `Ablation CAM` applied to the five layers in the neck of the ConvNeXt-S + BiFPN model. 81

A.12 `Ablation CAM` applied to the five layers in the neck of the ConvNeXt-S + CABiFPN model. 82
A.13 `Ablation CAM` applied to the five layers in the neck of the EfficientNetV2-M + BiFPN model. 83
A.14 `Ablation CAM` applied to the five layers in the neck of the EfficientNetV2-M + CABiFPN model. 84
A.15 Visualization of layers 1, 3, and 5 in the neck for the six tested models after applying the ROAD Least Relevant First method and removing 20% of the image (at the 80th percentile). Evaluated using image `ID 171382`. 86
A.16 Visualization of layers 1, 3, and 5 in the neck for the six tested models after applying the ROAD Most Relevant First method and removing 20% of the image (at the 80th percentile). Evaluated using image `ID 171382`. 87

List of Tables

5.1 Detailed hardware configuration employed for model training. 39
5.2 Configuration groups of the backbones employed in the proposed model, alongside their respective number of parameters in millions. 41
5.3 Image augmentation transforms from Albumentations [BIK+20] along with their associated probabilities. 43
5.4 Performance metrics for object detection and instance segmentation on the `val2017` dataset from `COCO2017`. 43
5.5 Combined MoRF and LeRF avarage confidence using ROAD strategy on image `ID 509403`. 52
5.6 Combined MoRF and LeRF avarage confidence using ROAD strategy on image `ID 96825`. 53
5.7 Combined MoRF and LeRF avarage confidence using ROAD strategy on image `ID 171382`. 53