

Article

An Approach to Incorporating Implicit Knowledge in Object Detection Models

Wenbo Peng¹  and Jinjie Huang^{1,2,*}¹ School of Automation, Harbin University of Science and Technology, Harbin 150080, China; wbopeng@163.com² Heilongjiang Provincial Key Laboratory of Complex Intelligent Systems and Integration, Harbin 150080, China

* Correspondence: jjhuang@hrbust.edu.cn

Abstract: Current object detection methods typically focus on addressing the distribution discrepancies between source and target domains. However, solely concentrating on this aspect may lead to overlooking the inherent limitations of the samples themselves. This study proposes a method to integrate implicit knowledge into object detection models, aiming to enhance the models' effectiveness in identifying target features within images. We analyze the sources of information loss in object detection models, treating this loss as a form of implicit knowledge and modeling it in the form of dictionaries. We explore potentially effective ways of integrating latent knowledge into the models and then apply it to object detection models. The models demonstrate a 1% and 0.8% improvement in mean average precision (mAP) in the UA-DETRAC and KITTI datasets, respectively. The results indicate that the proposed method can effectively enhance the relevant metrics of object detection models without significantly increasing the parameter or computational overhead. By excavating and utilizing implicit knowledge, we enhance the performance and efficiency of the models, offering new perspectives and methods for addressing challenges in practical applications.

Keywords: multi-task; implicit knowledge; dictionary parameters; object detection



Citation: Peng, W.; Huang, J. An Approach to Incorporating Implicit Knowledge in Object Detection Models. *Appl. Sci.* **2024**, *14*, 2124. <https://doi.org/10.3390/app14052124>

Academic Editor: Andrea Prati

Received: 9 February 2024

Revised: 29 February 2024

Accepted: 29 February 2024

Published: 4 March 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In deep learning models, sharing a common underlying network structure is a commonly employed approach when simultaneously handling and optimizing multiple distinct yet related tasks [1]. However, due to the varying feature requirements among different tasks, the sharing of the underlying network may lead to information interference between tasks. The object detection task can be characterized as achieving both object classification and object position regression, and these two tasks may mutually influence each other.

Furthermore, during the training process of deep learning models, when influenced by data quality and the model structure, the model may lose some valuable information relevant to the task. In convolutional neural networks, due to constraints on parameters such as the size and stride of convolutional kernels, certain local information may be discarded or blurred [2]. This information loss is particularly pronounced in deep neural networks, where each layer of convolutional operations may introduce a certain degree of information loss, thereby decreasing the overall performance and generalization capability of the entire network [3].

To alleviate the impact of information loss in the training process of object detection models, in this paper, we investigate the causes of model information loss by combining the algorithmic principles of single-stage object detection models. We propose a single-stage object detection method based on implicit knowledge. As shown in Figure 1, we present a method for modeling implicit knowledge that is distinguishable from conventional approaches in object detection, such as knowledge distillation or attention mechanisms. Specifically, to parameterize the implicit knowledge in the model, we introduce a set of

dictionaries into the model to extract implicit knowledge. Experimental results on the UA-DETRAC dataset [4] and KITTI dataset [5] demonstrate that our approach performs well in detecting objects in road scenes.

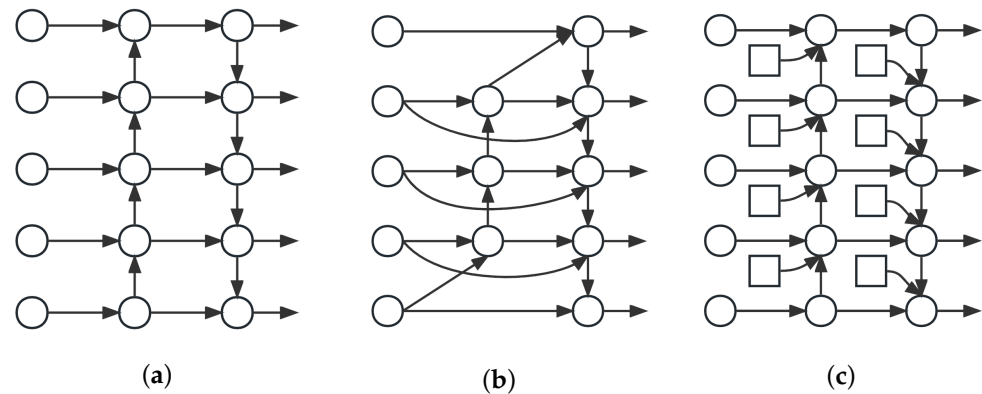


Figure 1. (a) FPN + PAN; (b) BiFPN; (c) FPN + PAN implicit knowledge.

In summary, our study makes the following contributions:

1. The causes of information loss in object detection tasks are thoroughly investigated, and the sources of implicit knowledge are fully analyzed;
2. A method to model implicit knowledge through parameter dictionaries is proposed, integrating the principles of dictionary learning. The experimental section validates the different effects of implicit knowledge on various structures of object detection models;
3. An approach is designed to incorporate implicit knowledge into the model structure of object detection, demonstrating, in the experimental results, that the cost of integrating implicit knowledge into the object detection model is negligible.

The remaining content of this paper is organized as follows. In Section 2, we introduce related work from three aspects: explicit and implicit knowledge, knowledge dictionaries, and the multi-scale structure of object detection models. In Section 3, we discuss the sources of implicit knowledge, theoretically analyze the feasibility of modeling implicit knowledge, and outline the methods for incorporating implicit knowledge into the model based on the structure of object detection models. Experimental validation of the ways in which implicit knowledge affects the model is conducted in Section 4 to confirm the effectiveness of the proposed approach. Finally, Section 5 concludes the paper and discusses future work.

2. Related Works

In this section, we will present the methods employed in this study and review the relevant research.

2.1. Explicit Knowledge and Implicit Knowledge

Convolutional neural networks (CNNs) extract the shallow features of objects as explicit knowledge, serving as the basis for observing the model's performance. Therefore, explicit knowledge can be understood as the parameters of the model learned from the data. These parameters are explicitly represented and quantifiable, such as the weights and biases of a neural network. The transformer [6,7] is a commonly used explicit deep learning method.

During the process of feature extraction, due to the lack of effective constraints, some deep-level information may be lost. We refer to this easily discarded portion of feature information as implicit knowledge. Implicit knowledge is knowledge that cannot be directly quantified or explicitly represented, such as when using periodic activation functions for implicit neural representations [8]. For example, an image classification model may learn how to recognize certain features in images (such as edges or textures), but this knowledge

is not directly reflected in the model's parameters; instead, it is implicit in the model's structure and parameters.

Both explicit knowledge and implicit knowledge are crucial components of deep learning models for understanding data and making predictions. In the field of object detection, methods based on knowledge distillation [9–11] are frequently employed to transfer knowledge from complex models to simpler models with slight performance loss. Attention mechanisms [12–14] are also commonly used to focus on relevant information and suppress irrelevant details. Wang et al. [15] proposed a method using kernel space alignment to obtain implicit knowledge, arguing that convolutional neural networks (CNNs) predominantly emphasize the extraction of explicit knowledge within the network, neglecting the impact of implicit knowledge.

2.2. Multi-Task Learning

The objective of multi-task learning is to enhance overall performance by enabling a model to leverage the correlations between different tasks through the sharing of model parameters or hierarchical structures. Multi-task models can simultaneously address various types of tasks, including classification, regression, and so on. In order to address feature differences among different tasks, prevalent methods include knowledge transfer through transfer learning [16], where shared knowledge is utilized to boost model performance in another task. Jocher et al. [17] balanced the contributions of different tasks in object detection models by weighting the loss functions for various tasks. Li et al. [18] mitigated the interference between different tasks by decoupling tasks. This enables the model to focus on individual tasks, reducing the impact of inter-task interference.

2.3. FPN + PAN Structure

The feature pyramid network (FPN) [19] enhances the accuracy of object detection and segmentation by constructing a feature pyramid across multiple scales, extracting object features at different resolutions. This network architecture effectively merges feature maps with strong semantic information at lower resolutions and feature maps with weaker but spatially rich semantic information at higher resolutions. This integration addresses the multi-scale challenges in object detection, improving the detection performance of smaller objects without significantly increasing computational cost. EfficientDet [20] introduced a bi-directional feature pyramid network (BiFPN), employing a fast multi-scale fusion approach to enhance the integration of information across scales.

By utilizing a bottom-up reinforcement path, the path aggregation network (PAN) [21] enhances the precise localization signals throughout the entire feature hierarchy, thereby shortening the information path between the bottom and top-most features. Both of these architectures effectively enhance the network's ability to detect objects at different scales, merging feature maps with strong semantic information at lower resolutions and feature maps with weaker but spatially rich semantic information at higher resolutions, all while minimizing the increase in computational complexity.

3. Problem Formulation and Preliminaries

Suppose that there is a noise data matrix $\tilde{X}$, which can be represented as

$$\tilde{X} = X + S + E \quad (1)$$

where X represents the noise-free data matrix, S represents the sparse noise matrix, and E represents the Gaussian noise matrix. If a neural network model is used to solve the data matrix, it can be represented as

$$\begin{aligned} y &= f_{\theta}(x) + \varepsilon \\ \text{minimize } &\varepsilon \end{aligned} \quad (2)$$

In the equation, ε represents the error term, which describes the disparity between the predicted values of the model and the actual observed values. θ denotes the parameter

vector, taking values in the n -dimensional Euclidean space. $f_{\theta}(x)$ represents the output values of the model f under the parameter vector θ when the input data are x . y signifies the output of the neural network. For a neural network model, our objective is to ascertain a parameter vector θ that satisfies

$$y = f_{\theta}(x) \quad (3)$$

During the model training process, the neural network's forward propagation extracts features by transforming raw input data into more abstract and meaningful representations. Neurons in different layers of the network learn to map input data to higher-level representation spaces, gradually extracting useful feature information from the data. In the neural network's backward propagation, optimization algorithms [22,23] are employed to minimize ε based on the feature information extracted during forward propagation, aiming to approximate the optimal solution for the parameter vector θ . This optimization theory can be summarized by the following formula:

$$\arg \min_{\theta} \|y - f_{\theta}(x)\|_2^2 \quad (4)$$

From an optimization perspective, we seek to find the optimal parameters to minimize prediction error. This involves defining a loss function to measure the disparity between the model's predictions and the actual values and employing optimization algorithms to adjust the model parameters. From the standpoint of deep learning, our aim is to enhance the model's generalization capability, meaning its ability to make accurate predictions on new, unseen data.

Therefore, in deep learning, we often do not strictly adhere to minimizing the error term to train the model. Moreover, the error term might reflect the complex patterns presented in the data, but this is not captured by the model. These patterns could arise due to insufficient model capacity or because the model's structure is inadequate for handling certain types of data. Thus, Equation (2) can be modified to

$$y = f_{\theta}(x) + \varepsilon + \mu \quad (5)$$

where μ represents the complex patterns presented in the data but that are not captured by the model, while ε still denotes the error term. We believe that incorporating this implicit knowledge can assist the model in achieving a better performance.

3.1. Information Loss in the Object Detection Model

In the process of feature extraction in object detection models, the hierarchical structure of neural networks and parameter adjustments lead to transformations and selections of input data, resulting in a certain degree of information loss. The information loss in object detection models stems from various aspects. This study aims to effectively explore and utilize the implicit knowledge within the model to enhance its performance.

3.1.1. Sampling Information Loss

The design and structure choices of neural network models may restrict their expressive power. For instance, using a smaller network depth, fewer convolutional or fully connected layers, etc., might fail to adequately capture the details and complex relationships within the input data, leading to information loss. Additionally, downsampling is often performed in neural network models to reduce computational complexity and memory consumption [24]. These operations result in a decrease in the dimensionality of input features, leading to the loss of some detailed information. Nonlinear activation functions (such as ReLU, sigmoid, etc.) are employed to perform nonlinear transformations on features; while these activation functions introduce nonlinear factors that aid the model in learning more complex feature representations, they may also introduce information loss, especially when features are compressed into a minimal range. As shown in Figure 2,

as the depth of the neural network increases, the loss of information in the model becomes more apparent.

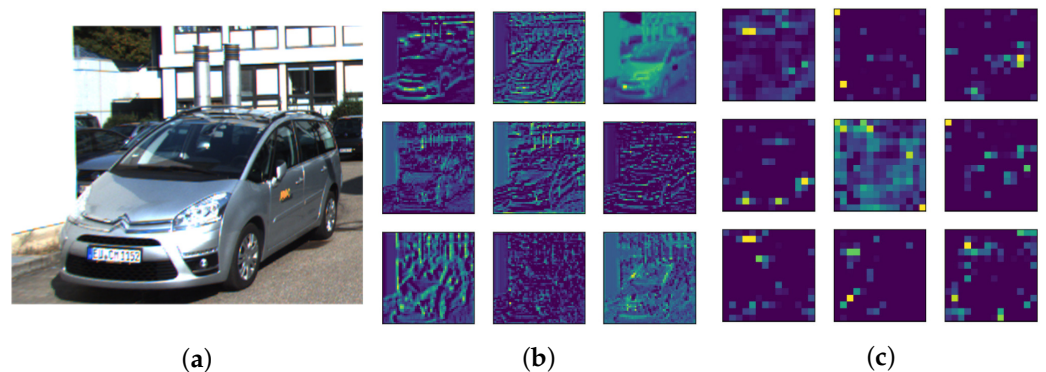


Figure 2. Visualization of feature maps at different layers in neural networks. (a) Input images; (b) shallow layer feature maps; (c) deep layer feature maps.

In real-time object detection tasks, we often opt to enhance the model's inference speed by reducing its parameter count and computational complexity [25]. However, due to the decrease in model complexity, the information loss during feature extraction tends to increase further.

In single-stage object detection models, features are first extracted using convolutional neural networks. Subsequently, multi-scale feature fusion operations are applied to the extracted features, followed by the prediction of object positions and categories using a series of convolutional and fully connected layers. Feature fusion involves utilizing techniques such as feature pyramid networks (FPNs) and path aggregation networks (PANs) to obtain both global and local information, thereby acquiring features with broader receptive fields at different scales.

However, during the feature fusion process in object detection, models using feature pyramid network (FPN) upsampling and fusion operations are performed on features from different levels. Due to the upsampling operation, the spatial resolution of lower-level features increases, but this may result in the loss of some fine-grained details. Upsampling typically employs interpolation methods (such as nearest neighbor or bi-linear interpolation), which can lead to the loss or blurring of fine details. Additionally, feature fusion operations may also contribute to information blurring. The fusion of lower-level and higher-level features might mix some semantic information, causing the details of lower-level features to be affected by blurring.

In the path aggregation network (PAN) structure, similar issues arise during the downsampling process of feature information in object detection models.

3.1.2. Information Loss Caused by Multi-Tasking

In addition to sampling information loss, multi-task learning is another significant factor leading to information loss in single-stage object detection models. In multi-task learning, the model needs to handle multiple tasks simultaneously, which may result in mutual interference between tasks, leading to information loss, as shown in Figure 3.

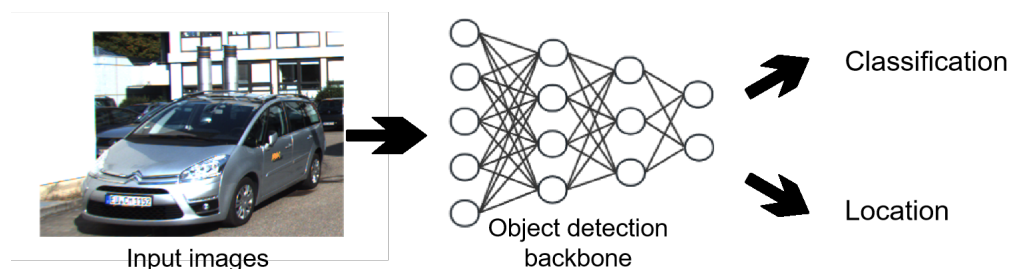


Figure 3. Schematic diagram of multi-task single-stage object detection model.

Currently, the main approach to address this issue is by designing loss functions to balance the importance of each task. Different tasks may have varying learning speeds and difficulties, and excessive focus on one task may lead to a decline in performance in others. During model training, to adapt to multiple tasks, the model may prioritize certain tasks while sacrificing the learning effectiveness for others, thereby causing information loss.

The shared underlying network structure in multi-task learning may also have an impact on the feature representation for each task. The shared representation may not be optimal for every task, causing some tasks to inadequately express their specific information, resulting in information loss.

3.2. Modeling Implicit Knowledge

In the above discussions, we explored the implicit knowledge embedded within the error term of the model and analyzed the parts of object detection models where information loss is prone to occur, along with potential causes. In this section, we will outline how to incorporate this implicit knowledge into the model.

In the transformer model [6,26], the self-attention mechanism captures the dependencies between different positions, whereas the introduction of the *cls_token* directs the model's focus to the entire sequence, thereby better capturing global semantic information. The *cls_token* serves as a special token for input sequences and is typically used to represent the semantic information of the entire input sequence or as input for classification tasks. It is transformed into a vector representation through an embedding layer and passed as part of the input sequence to the transformer encoder. The *cls_token* itself is not directly related to the classification target but serves as a learned special representation during the model's design and training processes, providing contextual information and semantic representation for the overall sequence. Constructing a structure similar to '*cls_token*' in object detection models enables the extraction of implicit knowledge obtained during the sampling process or influenced by model structural factors.

Suppose that there exists a set of parameters, z , that interact with the model and that aid in capturing the complex patterns that exist in the data but are not captured by the model, thereby storing the implicit knowledge, μ , present in the data. These parameters should be capable of storing and processing various types of information, dynamically updating and modifying during the training process. They should adjust their content and structure dynamically based on different input data and task requirements.

We believe that this set of parameters can be structured in the form of a dictionary and applied to the model. However, the exact mechanism of their influence is currently unknown. Therefore, we temporarily use the symbol " $\otimes$ " to denote the operational relationship between explicit and implicit knowledge, as follows:

$$y = f_{\theta}(x) \otimes g_{\varphi}(z) + \varepsilon \quad (6)$$

which can be represented as

$$\arg \min_{\theta, \varphi} \|y - f_{\theta}(x) \otimes g_{\varphi}(z)\|_2^2 \quad (7)$$

We use g_{φ} to represent the implicit knowledge gained by the model under the influence of parameter z . We attempt to use the features obtained from sampling in the object detection model as the "keys" of the dictionary and set multiple sets of mutually independent parameter vectors as the "values" of the dictionary, storing task-related implicit knowledge.

3.3. Modeling Implicit Knowledge in Object Detection Models

A single-step object detection model generally consists of three parts: the backbone, responsible for feature extraction; the neck, which performs multi-scale feature fusion; and the head, which detects the position and class of objects based on the features extracted by the previous two parts.

During the feature fusion process, the object detection model concatenates feature maps or feature vectors from different layers along the channel dimension, transforming features from different layers to the same size through interpolation or convolution. Shallow features, closer to the input data, typically have higher resolution and more detailed information, while deeper features contain higher-level semantic information and global context, albeit with lower spatial resolution but capturing broader contextual information. Since the feature maps from different layers contain varying information, sampling and concatenation may lead to feature aliasing issues [27], resulting in information loss. Therefore, we chose to incorporate implicit knowledge into the model during the feature fusion process to optimize feature representation, as illustrated in Figure 4.

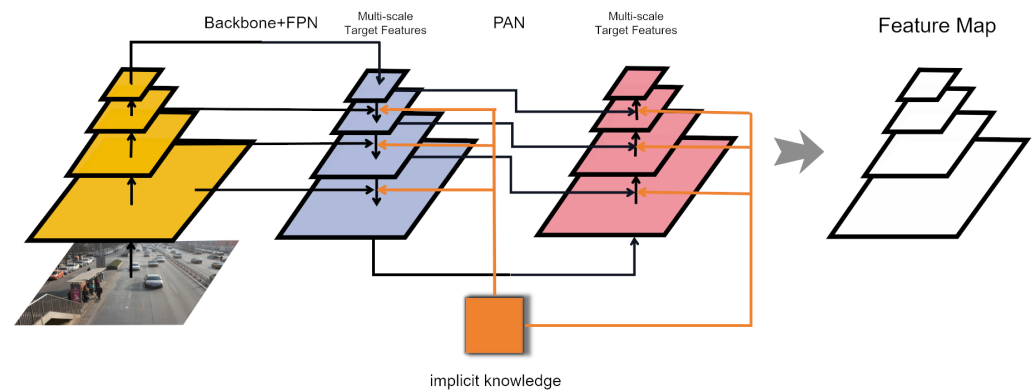


Figure 4. Architecture of n single-stage object detection model with implicit knowledge integration in feature extraction and fusion stages.

Let $C_i (i \in 1, \dots, n)$ denote the original feature maps input into the feature pyramid network (FPN) and the pyramid attention network (PAN) structures. The fused feature maps after fusion that compute h are represented as $P_i (i \in 1, \dots, n)$, where i represents the level of the feature maps. Therefore, the FPN and PAN stages can be expressed as

$$P_{i+n} = h(C_i, C_{i+1}, \dots, C_{i+n}) \quad (8)$$

The incorporation of the fusion operation, which integrates implicit knowledge into the model, can be represented as

$$P_{i+n} = h(C_i \otimes g_\varphi(z_i), C_{i+1} \otimes g_\varphi(z_{i+1}), \dots, C_{i+n} \otimes g_\varphi(z_{i+n})) \quad (9)$$

4. Experiment

In this section, we conducted experiments to validate the effectiveness of the proposed method. Our experiments were conducted on the UA-DETRAC and KITTI datasets.

4.1. Datasets and Evaluation Metrics

4.1.1. Datasets

As shown in Figure 5, the UA-DETRAC dataset comprises video clips captured from traffic surveillance cameras on urban roads. These clips are recorded in high definition, depicting real-world urban traffic scenarios involving various types of vehicles, pedestrians, and other road users. We selected three categories from the dataset, namely sedans, buses, and vans. We split the *Insight – MVT_Annotation_Train* part into training and validation sets in an 8:2 ratio and utilized all images from the *Insight – MVT_Annotation_Test* part as the test set.

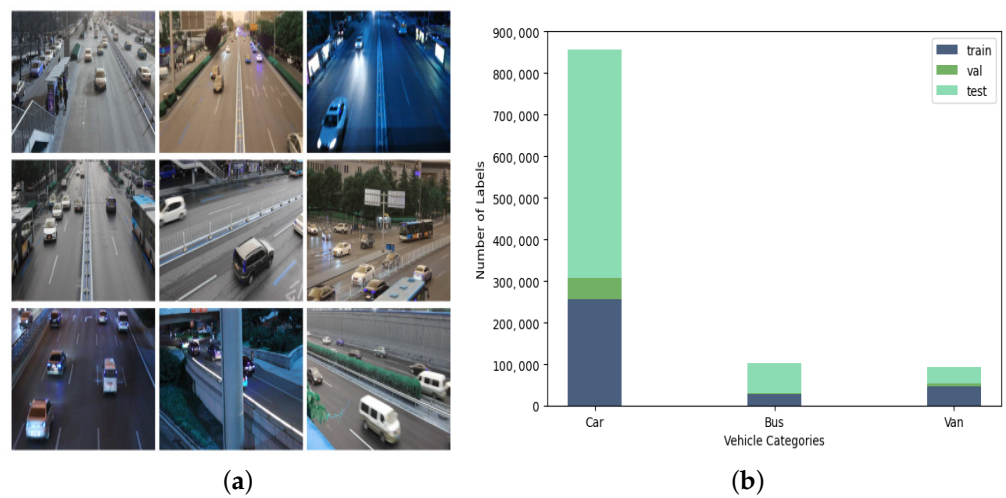


Figure 5. UA-DETRAC dataset. (a) Example images; (b) the distribution of labels in the dataset.

As shown in Figure 6, the KITTI dataset is a widely used public dataset for research in autonomous driving and computer vision. It provides high-resolution RGB image sequences captured in various scenes, such as urban streets, rural areas, and highways. These images often present various challenges, including different weather conditions, complex traffic situations, and lighting variations. For our experiments, we randomly split the training set into training and validation sets at a ratio of 8:2, and we selected four vehicle categories from the dataset: Van, Tram, Car, and Truck.

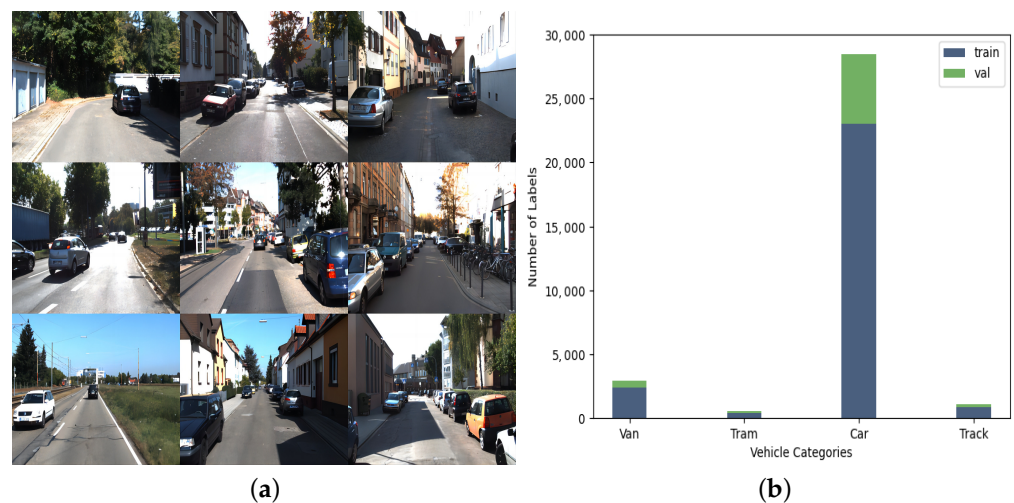


Figure 6. KITTI dataset. (a) Example images; (b) the distribution of labels in the dataset.

4.1.2. Evaluation Metrics

In the experiments, precision (P), recall (R), average precision (AP), and mean average accuracy (mAP) were employed as performance evaluation metrics. Precision is defined as the ratio of the number of true positive samples to the total number of positive samples predicted, serving to assess the model's accuracy. Precision (P) can be computed using the following formula:

$$P = \frac{TP}{TP + FP} \quad (10)$$

where TP represents the true positive cases where the actual condition is positive and the predicted condition is also positive, and FP represents the false positive cases where the actual condition is negative, but the predicted condition is positive. Recall, also known as sensitivity, is defined as the ratio of the number of true positive samples to the total number

of positive samples in the ground truth, which is used to evaluate the comprehensiveness of the model's detection. R can be calculated using the following formula:

$$R = \frac{TP}{TP + FN} \quad (11)$$

where FN represents the false negative cases where the actual condition is positive but the predicted condition is negative. Average precision (AP) is defined as the area enclosed by precision, recall, and axis. AP can be calculated using the following formula:

$$AP = \int_0^1 P(x) dx \quad (12)$$

where $P(x)$ in Formula (11) represents the smoothed precision–recall curve.

To calculate the mean average precision mAP , sum the AP values for each class and divide by the number of classes, as follows:

$$mAP = \frac{\sum_{i=1}^n AP_i}{n} \quad (13)$$

In the formula, n represents the number of classes and AP_i represents the average precision for the i -th class.

In the experiments, $mAP50$ and mAP are adopted as evaluation metrics to assess the effectiveness of the model. $mAP50$ calculates the average AP at IoU [24] = 0.5. mAP is computed across IoU values ranging from 0.5 to 0.95, with a step size of 0.05, and mAP is calculated at these IoU thresholds.

4.2. Implement Details

In this study, we conducted training on an Inspur brand computer equipped with an Intel 5220R XeonCPU @ 2.2 GHz processor, 32 GB of memory, and Tesla V100s graphics cards using the PyTorch [28] 1.10.1 with CUDA 11.3.

For a fair comparison, all methods were trained under the same settings. We chose YOLOv5s [17] as the baseline model for experiments, and all hyperparameters were based on the pre-trained model on the COCO dataset [29].

During the training process, we initialized the model parameters using those trained on the COCO dataset, and they were trained for 200 epochs. After 200 epochs, we froze the backbone part of the model and continued training for another 100 epochs without any other modifications, using the model after 200 epochs as the baseline model. In the experimental group, we attempted to incorporate implicit knowledge into the model by adding dictionary parameters to the model that had already been trained for 200 epochs. We then froze the model and continued training for another 100 epochs. The batch size during training remained constant at 32. Data augmentation was performed using normalization and random cropping. The weight decay of the Adam optimizer was set to 1×10^{-3} . The initial learning rate was set to 1×10^{-2} , and was then decayed to 1×10^{-4} , using the cosine annealing scheduling algorithm [30] over the iterations. The momentum was set to 0.937.

4.3. Experimental Design

In this study, the object detection model adopts an FPN+PAN structure. In the FPN stage, feature fusion involves feature maps from both the backbone and the upsampled feature maps from FPN. The feature maps from the backbone are obtained through convolutional modules [31], while the feature maps from the FPN are obtained through bi-linear interpolation. In the PAN stage, feature fusion involves feature maps from both the FPN and the downsampling stage of PAN itself. The former comes from the convolutional modules in the FPN stage, while the latter is derived from convolution in the PAN stage.

We assume two ways in which implicit knowledge affects the model, denoted as operation “ $\otimes$ ”: addition and multiplication. In the experiment, we set the initial parameters z of the dictionary to a set of random numbers following a normal distribution. If an

addition operation is used, the initial dictionary parameter is denoted as $z \sim N(0, 0.02)$; if a multiplication operation is used, the initial parameter is denoted as $z \sim N(1, 0.02)$. As shown in Figure 7, the “ $\otimes$ ” symbol represents an addition or multiplication operation.

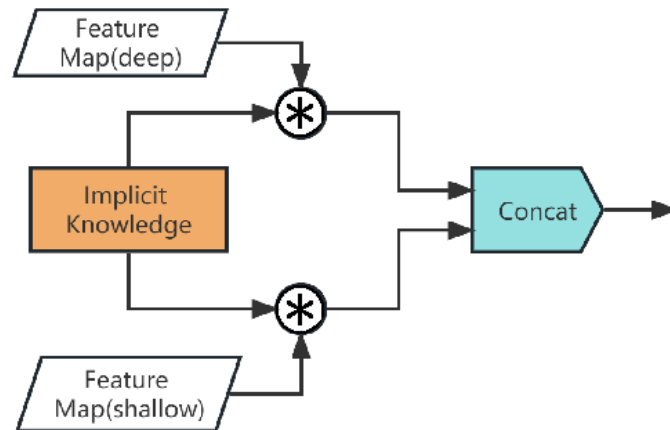


Figure 7. Parameter dictionaries in the feature fusion stage.

4.4. The Experimental Results and Analysis

In the comparative experiment results, as shown in Table 1, there are four feature fusion operations in the model. We categorized them into two groups based on how the feature maps from two parts are processed: the FPN part and the PAN part. We performed addition or multiplication operations with feature maps and parameter dictionaries processed differently, or we did not make any modifications. When comparing the experimental group with the baseline model indicators, it can be observed that incorporating implicit knowledge can enhance the effectiveness of the object detection model.

Table 1. Comparative experiments on the effects of parameter dictionaries on UA-DETRAC datasets.

	FPN Conv Upsample ¹		PAN Conv Conv		P	R	AP50	AP75	mAP
Baseline	○	○	○	○	0.731	0.669	0.721	0.68	0.571
1	○	+	○	○	0.735	0.671	0.728	0.687	0.578
2	○	×	○	○	0.736	0.67	0.725	0.683	0.576
3	+	×	○	○	0.736	0.67	0.722	0.68	0.573
4	×	○	○	○	0.738	0.669	0.725	0.684	0.575
5	+	×	○	○	0.738	0.665	0.721	0.681	0.572
6	×	+	○	○	0.735	0.673	0.729	0.687	0.579
7	×	×	○	○	0.734	0.672	0.723	0.682	0.574
8	+	+	○	○	0.735	0.673	0.723	0.68	0.574
9	○	○	○	+	0.741	0.669	0.723	0.68	0.573
10	○	○	○	×	0.738	0.667	0.727	0.687	0.577
11	○	○	+	×	0.735	0.668	0.723	0.681	0.574
12	○	○	×	×	0.734	0.674	0.725	0.682	0.575
13	○	○	×	+	0.736	0.669	0.723	0.682	0.574
14	○	○	+	×	0.742	0.668	0.725	0.682	0.575
15	○	○	×	×	0.737	0.672	0.729	0.688	0.579
16	○	○	+	+	0.742	0.666	0.723	0.681	0.574

¹ Improvements made after convolution/upsampling in FPN/PAN. “○” denotes no modification, “+” denotes the addition of parameter dictionaries for additive operations at this stage, and “×” denotes the addition of parameter dictionaries for multiplicative operations at this stage.

In the FPN part, we performed both addition and multiplication operations separately on the same feature fusion region. When comparing the AP50, AP75, and mAP in experiments 3 and 4, introducing implicit knowledge into the model using a multiplication

operation in the convolution part resulted in improvements of 0.3%, 0.4%, and 0.2%, respectively, when compared to using the addition operation. Similarly, when comparing the AP indicators in experiments 1 and 2 in the upsampling part, introducing implicit knowledge into the model using the addition operation resulted in improvements of 0.3%, 0.4%, and 0.2%, respectively, compared to using the multiplication operation. Likewise, in the PAN part, when comparing the AP indicators in experiments 9 and 10 and experiments 11 and 12, we can conclude that, for the feature fusion stage after the convolutional operation, using a multiplication operation for the parameter dictionary yields better results than using an addition operation in this dataset scenario. Therefore, we hypothesize that, for the feature fusion stage after linear operation, using the addition operation for the parameter dictionary yields better results than using the multiplication operation in this dataset scenario.

Based on the conclusions above, we compared the results of the following experiments: experiments 6 and 8, experiments 6 and 7, and experiments 15 and 16. We found that the results were consistent with the hypothesis, indicating that linear interpolation operations for implicit knowledge fusion should use an addition operation, while convolution operations for implicit knowledge fusion should use a multiplication operation.

Table 2 records the mAP (mean average precision) of three classes of targets in the experimental groups with significantly higher improvements compared to the baseline. When compared to the baseline, the integration of latent knowledge resulted in improved mAP values for all three categories of labels. In the dataset, the number of labels for the “van” category is significantly lower than those for the “car” and “bus” categories. The incorporation of latent knowledge notably enhanced the mAP for the “van” category, showing a 1.5% improvement in Experiment 15. This improvement suggests that the parameter dictionary successfully stored complex patterns existing in the data but not captured by the model, thereby enhancing the model’s expressive capacity.

Table 2. The mAP values of categories in certain experimental groups.

mAP	Baseline	1	4	6	10	15
car	0.612	0.613	0.614	0.615	0.614	0.615
bus	0.646	0.649	0.647	0.647	0.647	0.649
van	0.457	0.471	0.457	0.474	0.471	0.472

Further experiments were conducted, building upon the conclusions drawn from the previous sections, as shown in Table 3. The enhancement of model AP metrics by latent knowledge became more pronounced. In Experiment 22, the feature maps obtained through convolution were multiplied by the parameter dictionary, while the feature maps obtained through upsampling were added by the parameter dictionary. As a result, AP50, AP75, and mAP were improved by 1.1%, 1%, and 1%, respectively, compared to the baseline. With an increase in the number of parameter dictionaries, there was also an increase in the latent knowledge stored within these dictionaries.

Table 3. Comparative experiments on the effects of parameter dictionaries.

	FPN Conv Upsample		PAN Conv Conv		P	R	AP50	AP75	mAP
Baseline	○	○	○	○	0.731	0.669	0.721	0.68	0.571
17	×	×	×	×	0.749	0.665	0.729	0.688	0.578
18	+	+	+	+	0.746	0.67	0.731	0.69	0.58
19	+	+	×	×	0.746	0.667	0.731	0.69	0.58
20	×	×	+	+	0.744	0.669	0.728	0.685	0.578
21	+	×	+	+	0.745	0.669	0.731	0.69	0.58
22	×	+	×	×	0.743	0.668	0.732	0.69	0.581

As shown in Table 4, when analyzing the information regarding model parameters and computational complexity, the incorporation of implicit knowledge resulted in an increase of approximately 0.12% in model parameters and an increase of 0.08% to 0.11% in computational complexity. However, there was a significant improvement in the model's performance. It can be observed from Figure 8 that adding dictionary parameters did not have a substantial impact on the model training.

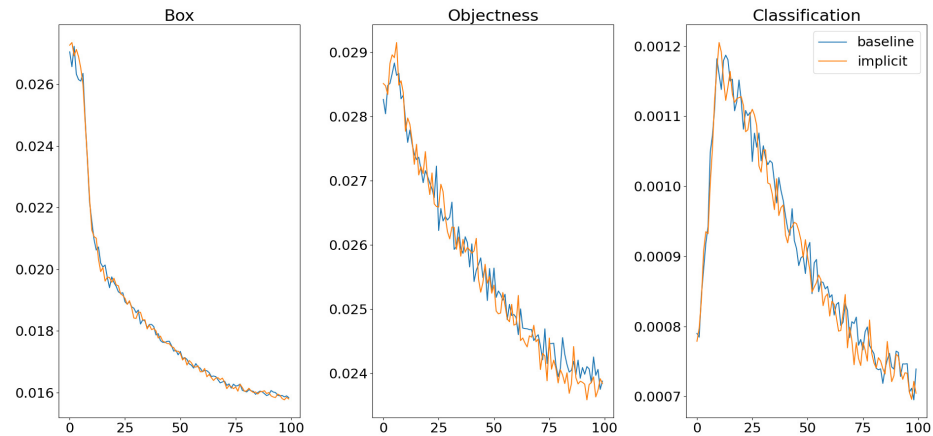


Figure 8. Model learning curve.

Table 4. Comparison of model parameters and computational complexity.

Model	# Parameters	MFLOPS
Baseline	7,016,287	158,429.6371
Implicit1 ¹	702,357 (+0.12%)	158,556.3808 (+0.08%)
Implicit2	7,025,791 (+0.13%)	158,603.9097 (+0.11%)

¹ The notation implicit{1, 2} indicates that the number of locations for adding parameter dictionaries is 1 and 2, respectively.

We compare the effectiveness of our proposed method with commonly used approaches in the field of object detection based on three aspects: mAP, computational complexity, and processing time per image; the experimental results are shown in Table 5. For the entire training process, we followed the YOLOv5 [17] training process. Furthermore, we opted to first train for 80 epochs and then fine-tune for an additional 40 epochs.

Table 5. Comparison of performance among different methods on UA-DETRAC dataset.

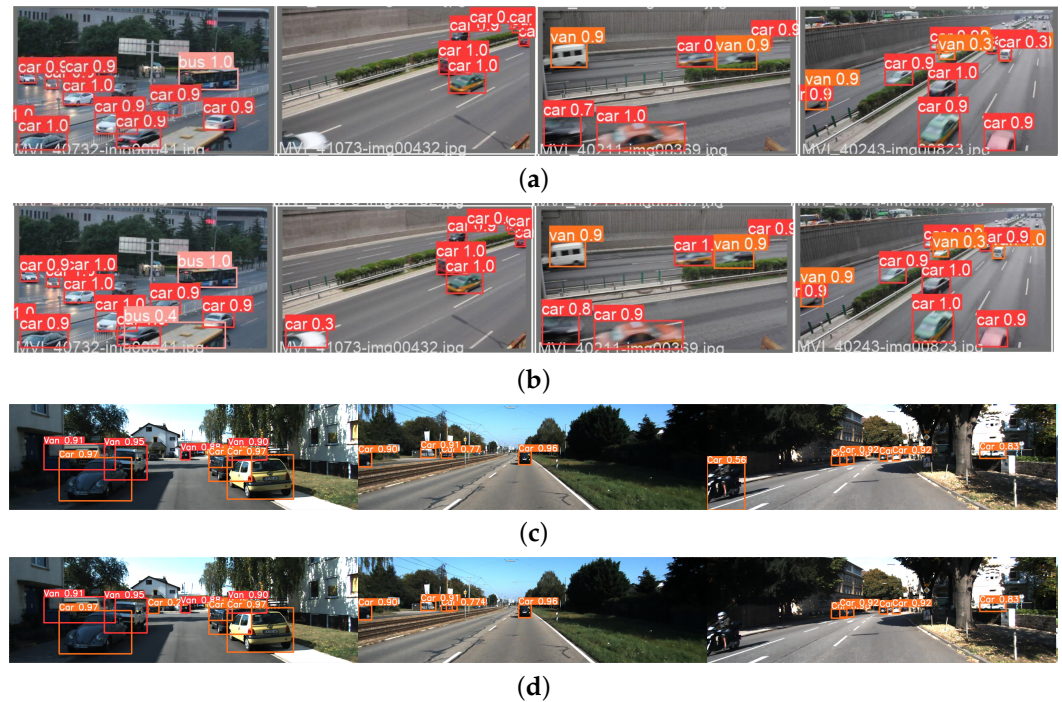
Method	mAP	Gflops	Time (ms)
CBAM [23]	0.598	15.8	2.0
MobileNetv2 [32]	0.583	10.4	2.2
MobileNetv3 [33]	0.559	37.9	2.3
ShuffleNet [34]	0.505	3.6	1.5
Ours	0.596	15.8	2.0

In order to validate the effectiveness of our proposed method on other datasets, experiments were conducted using the same configuration on the KITTI dataset. The results indicate that our proposed method successfully integrates latent knowledge into object detection models.

As shown in Table 6, compared to the baseline model, our proposed method effectively improves the relevant metrics of the object detection model. The performance of the proposed method on two datasets is shown in Figure 9.

Table 6. Experiment on KITTI dataset.

Model	P_{val}	R_{val}	$AP50_{val}$	mAP_{val}
Baseline	0.951	0.96	0.973	0.818
Ours	0.955	0.96	0.978	0.826

**Figure 9.** Detection performance. (a,c) Baseline model; (b,d) Model with implicit knowledge dictionary.

5. Conclusions

In this paper, we present the principles and implementation methods of modeling implicit knowledge. Specifically, we propose a method of modeling implicit knowledge using parameter dictionaries based on the idea of knowledge dictionary learning. Through these parameter dictionaries, we store complex patterns that exist in the data but are not captured by the model. In the future, we will investigate the role of implicit knowledge in transfer learning [35] and explore the ways in which prior knowledge influences deep learning.

Author Contributions: Conceptualization, W.P. and J.H.; methodology, J.H.; software, W.P.; validation, W.P.; writing—original draft preparation, W.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets used in this study are freely available and can be accessed from <https://detrac-db.rit.albany.edu> (accessed on 8 February 2024) and <https://www.cvlibs.net/datasets/kitti> (accessed on 8 February 2024).

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

References

1. Crawshaw, M. Multi-task learning with deep neural networks: A survey. *arXiv* **2020**, arXiv:2009.09796.
2. Liu, Y.; Qin, Z.; Anwar, S.; Caldwell, S.; Gedeon, T. Are deep neural architectures losing information? Invertibility is indispensable. In Proceedings of the Neural Information Processing: 27th International Conference, ICONIP 2020, Bangkok, Thailand, 23–27 November 2020; Proceedings, Part III 27; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; pp. 172–184.
3. Conneau, A.; Kiela, D.; Schwenk, H.; Barrault, L.; Bordes, A. Supervised learning of universal sentence representations from natural language inference data. *arXiv* **2017**, arXiv:1705.02364.
4. Wen, L.; Du, D.; Cai, Z.; Lei, Z.; Chang, M.C.; Qi, H.; Lim, J.; Yang, M.H.; Lyu, S. UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. *Comput. Vis. Image Underst.* **2020**, *193*, 102907. [\[CrossRef\]](#)
5. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets robotics: The kitti dataset. *Int. J. Robot. Res.* **2013**, *32*, 1231–1237. [\[CrossRef\]](#)
6. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 1–11.
7. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-end object detection with transformers. In Proceedings of the European Conference on Computer Vision, Online, 23–28 August 2020; Springer International Publishing: Cham, Switzerland, 2020; pp. 213–229.
8. Sitzmann, V.; Martel, J.; Bergman, A.; Lindell, D.; Wetzstein, G. Implicit neural representations with periodic activation functions. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 7462–7473.
9. Wang, T.; Yuan, L.; Zhang, X.; Feng, J. Distilling object detectors with fine-grained feature imitation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 4933–4942.
10. Sun, R.; Tang, F.; Zhang, X.; Xiong, H.; Tian, Q. Distilling object detectors with task adaptive regularization. *arXiv* **2020**, arXiv:2006.13108.
11. Zhang, L.; Ma, K. Improve object detection with feature-based knowledge distillation: Towards accurate and efficient detectors. In Proceedings of the International Conference on Learning Representations, Virtual, 26 April–1 May 2020.
12. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7132–7141.
13. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. Cbam: Convolutional block attention module. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.
14. Hou, Q.; Zhou, D.; Feng, J. Coordinate attention for efficient mobile network design. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Virtual, 19–25 June 2021; pp. 13713–13722.
15. Wang, C.Y.; Yeh, I.H.; Liao, H.Y.M. You only learn one representation: Unified network for multiple tasks. *arXiv* **2021**, arXiv:2105.04206.
16. Reed, C.J.; Yue, X.; Nrusimha, A.; Ebrahimi, S.; Vijaykumar, V.; Mao, R.; Li, B.; Zhang, S.; Guillory, D.; Metzger, S.; et al. Self-supervised pretraining improves self-supervised pretraining. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Waikoloa, HI, USA, 3–8 January 2022; pp. 2584–2594.
17. Glenn Jocher; et al. Ultralytics/yolov5: V6.0 - Yolov5n 'nano' Models, Roboflow Integration, Tensorflow Export, Opencv DNN Support. v6.0, Zenodo, October 2021. [\[CrossRef\]](#)
18. Li, C.; Li, L.; Jiang, H.; Weng, K.; Geng, Y.; Li, L.; Ke, Z.; Li, Q.; Cheng, M.; Nie, W.; et al. YOLOv6: A single-stage object detection framework for industrial applications. *arXiv* **2022**, arXiv:2209.02976.
19. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.
20. Tan, M.; Pang, R.; Le Q.V. Efficientdet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Online, 23–28 August 2020; pp. 10781–10790.
21. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path aggregation network for instance segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8759–8768.
22. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
23. Robbins, H.; Monro, S. A stochastic approximation method. *Ann. Math. Stat.* **1951**, *22*, 400–407. [\[CrossRef\]](#)
24. Gholamalinezhad, H.; Khosravi, H. Pooling methods in deep neural networks, a review. *arXiv* **2020**, arXiv:2009.07485.
25. Zou, Z.; Chen, K.; Shi, Z.; Guo, Y.; Ye, J. Object detection in 20 years: A survey. *Proc. IEEE* **2023**, *111*, 257–276. [\[CrossRef\]](#)
26. Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Virtual, 19–25 June 2021; pp. 10012–10022.
27. Luo, Y.; Cao, X.; Zhang, J.; Guo, J.; Shen, H.; Wang, T.; Feng, Q. CE-FPN: Enhancing channel information for object detection. *Multimed. Tools Appl.* **2022**, *81*, 30685–30704. [\[CrossRef\]](#)
28. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 1–12.

29. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In Proceedings of the Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, 6–12 September 2014; Proceedings, Part V 13; Springer International Publishing: Berlin/Heidelberg, Germany, 2014; pp. 740–755.
30. He, T.; Zhang, Z.; Zhang, H.; Zhang, Z.; Xie, J.; Li, M. Bag of tricks for image classification with convolutional neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 558–567.
31. Wang, C.Y.; Liao, H.Y.M.; Wu, Y.H.; Chen, P.Y.; Hsieh, J.W.; Yeh, I.H. CSPNet: A new backbone that can enhance learning capability of CNN. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 14–19 June 2020; pp. 390–391.
32. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520.
33. Howard, A.; S.; ler, M.; Chu, G.; Chen, L.C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; Le, Q.V. Searching for mobilenetv3. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 1314–1324.
34. Ma, N.; Zhang, X.; Zheng, H.T.; Sun, J. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 116–131.
35. Zhuang, F.; Qi, Z.; Duan, K.; Xi, D.; Zhu, Y.; Zhu, H.; Xiong, H.; He, Q. A comprehensive survey on transfer learning. *Proc. IEEE* **2020**, *109*, 43–76. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.