Journal of Computer Science and Cybernetics, V.42, N.2 (2026), 119–136
DOI no. 10.15625/1813-9663/21114

ENHANCING OBJECT DETECTION EFFICIENCY WITH TRANSFORMERS THROUGH MULTI-LEVEL FEATURE INTEGRATION

DUNG NGUYEN[1], VAN-DUNG HOANG[2*], VAN-TUONG-LAN LE[3]

[1]*Hue University of Sciences, Hue University, No. 77, Nguyen Hue Street, Thuan Hoa Ward, Hue City*

[2]*Ho Chi Minh City University of Technology and Engineering, No. 1, Vo Van Ngan Street, Thu Duc Ward, Ho Chi Minh City*

[3]*Department of Academic and Students' Affairs, Hue University, No. 3, Le Loi Street, Thuan Hoa Ward, Hue City*

Abstract. This paper presents a novel approach to enhancing object detection efficiency by integrating multi-level features within a transformer architecture. Traditional object detection methods often rely on single-level feature representations, which may limit their ability to accurately detect objects of varying sizes and complexities. By leveraging multi-level feature integration within the transformer framework, our method captures a richer set of spatial and semantic information, leading to more precise and robust object detection. The powerful attention mechanisms of transformers are utilized to effectively combine these features, improving detection accuracy and localization. The proposed approach is evaluated on the PASCAL VOC benchmark dataset, demonstrating superior performance over conventional single-level feature-based methods. Experimental results show that our model achieves an mAP@0.5 of 87% on PASCAL VOC, outperforming recent state-of-the-art methods while maintaining computational efficiency. These findings highlight the potential of multi-level feature integration within transformers in advancing the field of object detection.
Keywords. high resolution, multi-level features, object detection, transformer.

1. INTRODUCTION

Object detection is a fundamental task in computer vision with widespread applications in fields such as autonomous driving [1], surveillance [2], and medical imaging [3]. The goal of object detection algorithms is to identify and localize objects of interest within an image, providing crucial information for downstream tasks. Traditional object detection methods often rely on handcrafted features and manually designed architectures, which may struggle to handle the complexities and variations present in real-world images. However, recent advancements in deep learning, particularly with the introduction of Transformer architectures, have revolutionized the field by offering powerful tools for feature extraction and representation learning.

Transformers [4], initially proposed for natural language processing tasks, have shown remarkable success in computer vision tasks, including image classification, segmentation, and

*Corresponding author.
E-mail addresses: nguyendung@hueuni.edu.vn (D. Nguyen); dunghv@hcmute.edu.vn (V.D. Hoang); lvtlan@hueuni.edu.vn (V.T.L Le)

object detection. Unlike traditional convolutional neural networks (CNNs) [5, 6], Transformers leverage self-attention mechanisms to capture long-range dependencies and relationships between different image regions effectively. In addition to the Transformer architecture, the integration of multi-level features has emerged as a crucial strategy for improving object detection performance. Objects in images can vary significantly in scale, orientation, and appearance, making it challenging for traditional detectors to achieve accurate localization and classification. By incorporating features from multiple levels of abstraction, ranging from low-level edges to high-level semantic representations, models can better capture the rich contextual information necessary for robust object detection. Furthermore, the importance of high-resolution input data cannot be understated, as it enables detectors to capture fine-grained details and subtle visual cues that are essential for accurate localization and classification.

In this paper, we propose a novel object detection approach that uniquely integrates multi-level features extracted from a Swin Transformer backbone with Positional Encoding, combined with the computational efficiency of Linformer in the Transformer encoder. Unlike existing methods such as Deformable DETR [7], or Conditional DETR [8], our method effectively captures contextual information across multiple feature scales while significantly reducing computational complexity. By leveraging the expressive power of Transformers, the rich representation of multi-level features, and the efficiency gains from Linformer, our approach aims to improve detection accuracy, especially for objects at varying scales, without incurring high computational costs. We evaluate the proposed method on standard benchmark datasets, and the results demonstrate its effectiveness and competitive performance compared to state-of-the-art approaches. Extensive experiments and analysis provide insights into the advantages and potential impact of our method for advancing object detection.

The remainder of this paper is organized as follows. Section 2 reviews related work on object detectors, Transformers, and hierarchical vision Transformers. Section 3 describes the proposed S-DETR architecture, including multi-level feature fusion, positional embedding, Linformer-based encoding, and the bipartite matching loss. Section 4 presents the experimental settings, datasets, evaluation results, and ablation studies. Finally, Section 5 concludes the paper and discusses potential directions for future research.

2. RELATED WORKS

2.1. Object detector

In recent years, object detection has advanced significantly thanks to deep learning. Among proposed models, DETR (Detection Transformer) [9, 10] stands out due to its novel architecture and strong performance. Traditional two-stage methods like R-CNN and its variants [11, 12] achieve high accuracy but suffer from computational inefficiency due to their sequential pipeline of region proposal and classification. This limitation has led to single-stage models such as YOLO [13] and SSD [14], which enable real-time detection but often struggle with small objects and precise localization. Figure 1 compares the workflows of Faster R-CNN and DETR.

DETR [9], proposed in 2020, reformulates object detection as a direct set prediction problem using the Transformer architecture. It eliminates handcrafted components like region

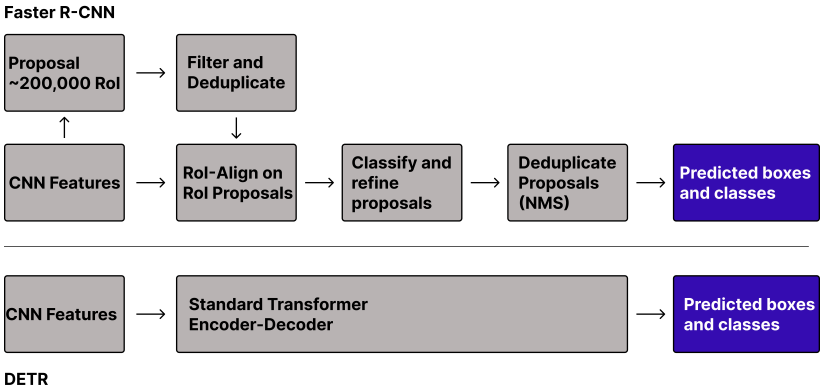


Figure 1: Comparison of the workflow between Faster R-CNN and DETR models

proposals, anchor boxes, and NMS, instead leveraging attention mechanisms to directly predict object classes and bounding boxes in an end-to-end manner. This simplifies the pipeline and improves flexibility in handling objects of varying sizes and shapes. Experiments on the COCO [15] dataset demonstrate competitive performance with faster inference compared to strong two-stage models.

DETR has inspired many extensions, including Deformable DETR and Conditional DETR, which improve spatial modeling and contextual reasoning. It has also been applied to tasks such as instance segmentation, panoptic segmentation, and video object detection, highlighting its adaptability. However, DETR still has limitations, including high computational cost due to self-attention and reduced performance in cluttered or occluded scenes where local context is critical.

2.2. Transformer

Transformers [4] have revolutionized the field of natural language processing (NLP), with significant applications in speech recognition [16], speech synthesis [17], and natural language generation [18]. Unlike recurrent neural networks (RNNs) such as LSTM [19], which process sequential data step by step, Transformers utilize stacked self-attention layers to effectively capture long-range dependencies. This design enables Transformers to process entire sequences in a single computation while leveraging deep architectures to enhance performance.

The Transformer architecture consists of two main components: the Encoder and the Decoder. The Encoder is composed of multiple layers, each containing two key components: Multi-Head Self-Attention and a Feed-Forward Neural Network. The Multi-Head Self-Attention mechanism allows the model to focus on different parts of the input sequence simultaneously rather than relying on a single attention layer. The output from this mechanism is then passed through a Feed-Forward Neural Network, which consists of two linear layers with a non-linear activation function in between, enhancing the model's representation capability. Similarly, the Decoder has multiple layers but includes an additional attention mechanism to focus on the output of the Encoder, enabling it to utilize input information when generating the output sequence. To improve stability and efficiency, both the Encoder and Decoder incorporate Layer Normalization and Residual Connections.

2.3. Hierarchical vision transformers in object detection

Recent advancements in vision transformers have introduced several hierarchical architectures that aim to improve feature representation and computational efficiency in object detection. Swin Transformer [20, 21] introduces a shifted window attention mechanism, which enables efficient local and global feature interactions while maintaining a hierarchical structure similar to CNNs. This design significantly reduces computational complexity compared to vanilla Vision Transformer (ViT) [22], which relies on global self-attention across all image tokens, making it less efficient for high-resolution inputs.

Another competitive hierarchical model is ConvNeXt [23], which revisits standard CNN architectures with Transformer-inspired modifications, such as depthwise convolutions and LayerNorm. While ConvNeXt achieves strong performance with efficient inductive biases, it lacks the adaptive feature learning capabilities of vision transformers like Swin Transformer. Compared to these models, Swin Transformer v2 enhances stability through post-norm attention and improved normalization techniques, making it a well-suited backbone for object detection.

Table 1: Computational complexity and accuracy of vision backbones

Backbone	GFLOPs	Parameters (M)	Top-1 Accuracy (%)
ResNet-50	4.1	25M	76.2
ResNet-101	7.8	45M	77.4
ViT-B/16	17.6	86M	84.0
ConvNeXt-Tiny	4.5	28M	82.1
Swin-T v2	4.7	29M	82.8
Swin-S v2	8.7	50M	83.5

Table 1 presents a comparative analysis of different vision backbones in terms of FLOPs, parameter count, and Top-1 accuracy on ImageNet-1K. Traditional CNN-based models like ResNet provide a good trade-off between efficiency and accuracy, but cannot capture long-range dependencies. ViT, on the other hand, achieves higher accuracy but comes with a significant computational cost due to its self-attention mechanism. ConvNeXt introduces improvements over CNNs, achieving better accuracy while maintaining moderate complexity. Swin Transformer v2, with its shifted window attention mechanism, balances computational efficiency and accuracy, outperforming ConvNeXt and ResNet while being more scalable than ViT. The comparison in Table 1 highlights the advantages of Swin Transformer v2 as a backbone for object detection.

3. PROPOSAL METHOD

3.1. Model overview

S-DETR consists of a backbone for extracting multi-level features, an encoder with attention mechanisms focusing on important components of the image, and a conventional decoder in the standard Transformer architecture with a series of object queries. An overview of S-DETR is illustrated in Figure 2. Specifically, the input image feature map is extracted using the Swin-Transformer v2 backbone pretrained on the large ImageNet-1K [24] dataset. The resulting features, denoted as S1, S2, S3, S4, are then fed into the encoder.

Multi-Level Feature Integration enhances object detection performance by effectively combining feature representations from different network layers. Lower-level features capture fine-grained spatial details, which are crucial for detecting small objects, while higher-level features provide rich semantic context for robust classification. By integrating multi-scale features using techniques such as Feature Pyramid Networks (FPN) or Bidirectional Feature Pyramid Networks (BiFPN), our model maintains high-resolution spatial information while leveraging deep semantic cues. This integration significantly improves detection accuracy, particularly for small and occluded objects, ensuring a more comprehensive feature representation across different object scales.

The encoder transforms multi-scale features into a sequence of image features through interactions among features at the same scale and fusion among features at different scales, see also Figure 4. Subsequently, the features from the encoder are passed to the decoder.

Attention mechanisms improve feature selection by emphasizing important regions in an image while suppressing irrelevant background noise. Our model incorporates both channel attention and spatial attention to enhance object localization and classification. Channel attention dynamically reweights feature maps to prioritize informative channels, whereas spatial attention focuses on the most relevant regions within an image. By leveraging these mechanisms, our model achieves better feature discrimination, reducing false positives and improving detection robustness in cluttered or complex environments. This approach ensures that the model efficiently allocates computational resources to the most significant object features.

Finally, the decoder, along with the object queries, performs predictions to generate class labels and bounding boxes for objects in the image.

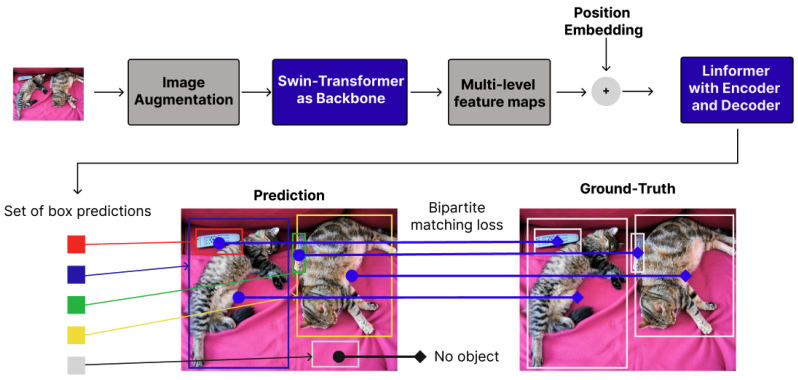


Figure 2: Overview of S-DETR

3.2. Image augmentation

Generating new variations of training data by performing transformations on images helps the model learn richer representations of object variations. Methods such as rotation, scaling, resizing, image flipping, and cropping are commonly used in the augmentation process, as illustrated in Figure 3. These methods are often applied before the data is fed into deep learning-based object detection models to improve the quality and generalization ability of the model. Research related to combining these techniques and applying them

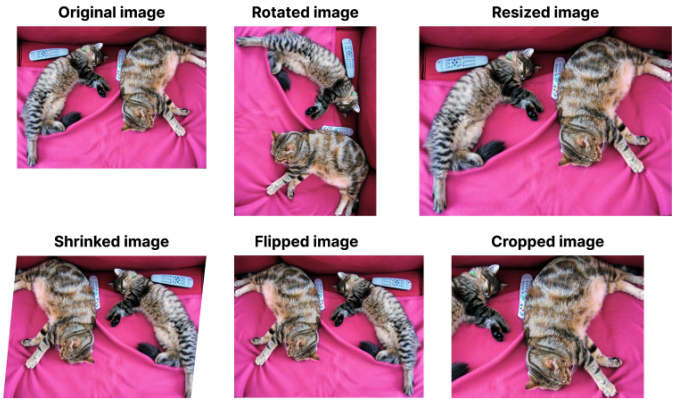


Figure 3: Data augmentation

to specific object detection models may provide deeper insights into their performance and practical applications.

3.3. Data extraction model

The Swin Transformer V2 model [21] is an improved version of Swin Transformer for large-scale vision tasks. Compared with the original design, it improves training stability and scalability while preserving the efficiency of window-based attention. Figure 5 presents the overall architecture.

Key improvements include:

- Better training stability via residual-post-norm and cosine attention.
- Better cross-resolution transfer using log-spaced continuous position bias.

Swin Transformer V2 can scale to very large models and high-resolution inputs (e.g., SwinV2-G), making it suitable for detail-sensitive applications. The network is organized into four hierarchical stages. Across stages, patch merging progressively reduces spatial resolution and increases channel capacity, while local window attention preserves important contextual information.

In early stages, the model captures fine-grained local features; in later stages, it learns more abstract semantic representations with broader receptive fields. This hierarchical design provides a strong balance between accuracy and computational efficiency, enabling robust multi-scale feature extraction for downstream detection tasks.

3.4. Fusion with multi-level features

Each stage of the Swin Transformer extracts features from different regions of the image, with varying resolutions and levels of detail. Assuming there are N stages in the Swin Transformer, each stage i extracts a feature matrix F_i , where H_i and W_i are the height and width of the feature at stage i, and C_i is the number of channels of the feature.

After features are extracted from different stages of the Swin Transformer, they can be merged through a convolutional layer or a feature fusion module. This helps to enhance information on different scales, from fine details to broader context. Feature fusion can be

represented by a function F, where F is a fusion function. Equation 1, the function we use for fusion is concatenation.

$$F = F_1 + F_2 + ... + F_N. \tag{1}$$

The output size of this function will be $H_{\text{target}} \times W_{\text{target}} \times C_{\text{target}}$.

However, in each stage of the Swin Transformer, features have different numbers of channels and sizes. Therefore, combining features using addition requires some processing steps to adjust the size and number of channels so that they can be merged appropriately.

Normalizing feature size

- To concatenate features from different stages, they need to be the same size. This can be achieved using convolutional layers, pooling, or interpolation.
- For example, if features from stage 1 have size $H_1 \times W_1$ and features from stage 2 have size $H_2 \times W_2$, we need to adjust their sizes to the same size, for instance $H_{\text{target}} \times W_{\text{target}} \times C_{\text{target}}$.

Adjusting the number of channels

- Features from different stages may have different numbers of channels. To merge them, we can use a 1×1 convolutional layer to adjust the number of channels of each feature to a specified number of channels.
- For example, if features from stage 1 have C_1 channels and features from stage 2 have C_2 channels, we can use a 1×1 convolutional layer to adjust their number of channels, as shown in Equation 2.

$$F_1' = W_1 \times F_1; F_2' = W_2 \times F_2, \tag{2}$$

where $W_1 \in \mathbb{R}^{C_{\text{target}} \times C_1}$ and $W_2 \in \mathbb{R}^{C_{\text{target}} \times C_2}$ are the weights of the 1×1 convolutional layers. The above process can be illustrated as shown in Figure 4.

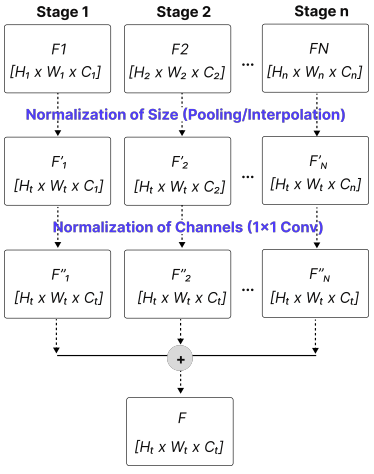


Figure 4: Fusion with multi-level features

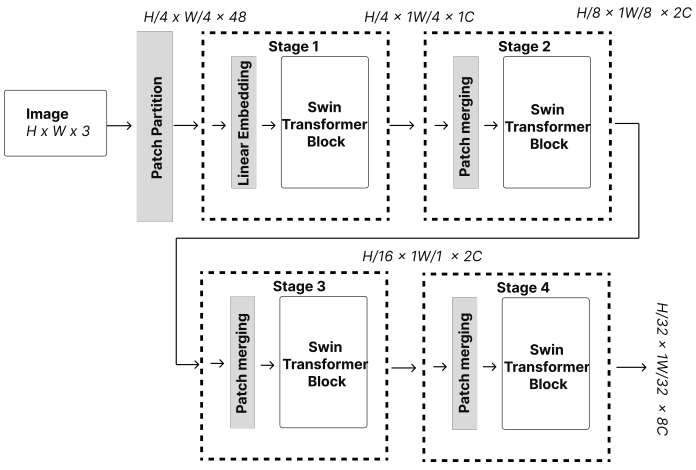


Figure 5: Architecture of Swin-Transformer v2

3.5. Position embedding

Position embedding (PE) in Transformer models is a crucial technique that helps the model understand the spatial positions of regions within an image. Transformer models process pixels or regions in an image in parallel, which leads to a loss of spatial position information. PE provides this information by adding position vectors to the embedding vectors of the regions, helping the model maintain the spatial structure of the image.

There are two main types of PE: fixed position embedding and learnable position embedding. Fixed position embedding uses fixed formulas, such as sine and cosine functions, to encode positions. These formulas generate position vectors based on predetermined frequencies and resolutions, ensuring consistency across different inputs without adaptation to specific data patterns. Equations 3 and 4 illustrate the computation of PE using the fixed position embedding method.

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right), \tag{3}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right), \tag{4}$$

where pos is the position in the sequence, i is the dimension of the position vector, and d_{model} is the size of the input vector.

In contrast, learnable position embedding initializes position vectors randomly and adjusts them during training based on the input data. Each position in the image is associated with a distinct position vector, and these vectors are optimized to enhance the model's performance on the training dataset. This approach allows the model to adaptively represent position information, tailoring its representation to best suit the requirements of the task at hand.

Suppose $X \in R^{n\times d}$ is the embedding matrix of regions in the image, where X_i is the embedding of the i_{th} region in the image. Suppose a position embedding matrix $P \in R^{n\times d}$, where P_i is the position vector for the i_{th} position in the image. The values of P are either fixed or randomly initialized and can be learned during training. The embedding vector of the i_{th} region after adding the position information is

$$Z_i = X_i + P_i, \tag{5}$$

where Z_i is the combined embedding vector of the region and the position at the i_{th} position.

For the entire image, we have

$$Z = X + P, \tag{6}$$

where X is the embedding matrix of the regions, P is the learnable position embedding matrix, and Z is the combined embedding matrix containing both region and position information.

The advantage of learnable position embedding is its high adaptability to various types of data and different tasks, improving the model's performance in understanding the spatial structure and positions of regions within an image. This is particularly important in applications such as object detection, image segmentation, and image recognition.

3.6. Linformer

Linformer [25] is a variant of the Transformer model designed to enhance the computational efficiency of traditional attention mechanisms. The original Transformer faces challenges with its $O(n^2)$ complexity, where n represents the input sequence length, particularly impacting the processing of long sequences or large images due to memory and speed limitations. Linformer addresses these issues by reducing the complexity to $O(n)$ through strategic optimizations.

Attention complexity in the standard transformer. In the conventional Transformer, the self-attention mechanism computes an attention matrix A using the formula

$$A = \text{softmax}\left(\frac{QK^T}{\sqrt{d_m}}\right)V, \tag{7}$$

where $Q, K, V \in \mathbb{R}^{n\times d_m}$ are the query, key, and value matrices, and d_m is the embedding dimension. The computation of the attention scores QK^T has a complexity of

$$O(n \cdot d_m \cdot n) = O(n^2 d_m), \tag{8}$$

which dominates the overall complexity of the model.

Complexity reduction in Linformer. Linformer introduces a more efficient approach by projecting the key and value matrices into a lower-dimensional space using a learned projection matrix $E \in \mathbb{R}^{k\times n}$, where $k \ll n$. The transformation is defined as

$$K' = EK, \quad V' = EV, \tag{9}$$

where $K' \in \mathbb{R}^{k\times d_m}$ and $V' \in \mathbb{R}^{k\times d_m}$. Additionally, Linformer applies learned weight matrices $W^K, W^V \in \mathbb{R}^{d_m\times d_k}$ to transform the projected keys and values

$$K'' = K'W^K, \quad V'' = V'W^V, \tag{10}$$

where $K'' \in \mathbb{R}^{k\times d_k}$ and $V'' \in \mathbb{R}^{k\times d_k}$. The attention mechanism in Linformer then computes

$$A' = \text{softmax}\left(\frac{QK''^T}{\sqrt{d_k}}\right)V''. \tag{11}$$

Since K'' has size $k \times d_k$, the computational cost of QK''^T is

$$O(n \cdot d_k \cdot k) = O(nkd_k), \tag{12}$$

Similarly, computing $A'V''$ has complexity

$$O(nkd_k). \tag{13}$$

Final complexity analysis. From Equations (12) and (13), both major computations in Linformer operate in $O(nkd_k)$, leading to an overall complexity of $O(nkd_k)$. For a small fixed k (typically $k = O(d_k)$ in practical implementations), this results in an overall complexity of $O(nd_k)$, which is linear with respect to n. This confirms that Linformer effectively reduces the computational complexity from $O(n^2 d_m)$ in the standard Transformer (Equation (8)) to $O(nd_k)$, making it significantly more efficient for handling long sequences and large-scale data. Additionally, by leveraging the projection matrix E and learned weight matrices W^K, W^V, Linformer ensures that memory consumption scales linearly with n, as it no longer needs to store the full $n \times n$ attention matrix. This significantly reduces memory overhead, making Linformer well-suited for real-world applications involving long sequences and large datasets.

3.7. Bipartite matching loss

Bipartite matching loss [9] is a loss function used in machine learning and computer vision, particularly in tasks such as object detection and image segmentation. It is designed to handle the assignment of predicted outputs to ground truth targets in an optimal way to minimize the total cost of assignments. Before delving into the application of this function, let's review some foundational concepts:

- Bipartite graph: In a bipartite graph, nodes can be divided into two disjoint sets such that no two vertices in the same set are adjacent. In the context of bipartite matching loss, one set of nodes represents the predicted outputs, and the other set represents the ground truth targets.
- Matching: The goal is to find a match between the predicted outputs and the ground truth targets such that the total cost of these assignments is minimized.
- Hungarian algorithm: Often used to efficiently solve the bipartite matching problem, the Hungarian algorithm finds the optimal matching that minimizes the total cost.

In object detection, the bipartite matching loss is used to match predicted bounding boxes with ground truth bounding boxes. The process involves:

- Cost calculation: Compute a cost matrix where each entry represents the cost of matching a predicted bounding box to a ground truth bounding box. This cost can be a combination of factors such as Intersection over Union (IoU), classification scores, and location differences.
- Optimal matching: Use the Hungarian algorithm to find the optimal assignment of predicted boxes to ground truth boxes that minimizes the total matching cost.
- Loss computation: After obtaining the optimal matching, compute the loss for each matched pair. This can include classification loss (e.g., cross-entropy [26] or focal-loss [27]) and localization loss (e.g., L1 or smooth L1 loss for bounding box coordinates).

This loss function ensures that each ground truth target is accurately assigned to one prediction, helping to address issues such as multiple predictions for the same object. Additionally, it can handle varying numbers of objects in different images, making it suitable for dense prediction tasks.

The following mathematical formulas describe the problem. Let:

- y be the set of ground truth objects.
- $\hat{y}$ be the set of predicted objects.
- $\hat{y}_i$ be the ith predicted object, consisting of class $\hat{c}_i$ and bounding box $\hat{b}_i$.
- y_j be the jth ground truth object, consisting of class c_j and bounding box b_j.

The cost matrix $C \in \mathbb{R}^{N \times M}$ is defined, where each element represents the cost of matching predicted object $\hat{y}_i$ to ground truth object y_j. It consists of classification cost and bounding box cost

$$c_{ij} = L_{\text{cls}}(\hat{c}_i, c_j) + L_{\text{box}}(\hat{b}_i, b_j), \tag{14}$$

where L_{cls} is the classification cost (e.g., cross-entropy loss) and L_{box} is the bounding box cost.

Using the Hungarian algorithm, we find the optimal assignment σ

$$\sigma = \arg\min_{\sigma \in \Sigma_N} \sum_{i=1}^{N} c_{i,\sigma(i)}, \tag{15}$$

where Σ_N is the set of all possible permutations of N elements.

The total loss L is calculated as

$$L(y, \hat{y}) = \sum_{i=1}^{N} \left[L_{\text{cls}}(\hat{c}_i, c_{\sigma(i)}) + L_{\text{box}}(\hat{b}_i, b_{\sigma(i)}) \right]. \tag{16}$$

4. EXPERIMENTS

4.1. Datasets

The Pascal VOC 2012 (PASCAL visual object classes) dataset [28] is a widely recognized benchmark in computer vision, primarily used for object detection and classification. This dataset comprises a diverse array of real-world images, each meticulously annotated with objects from 20 different categories, including people, animals (such as birds, cats, and dogs), vehicles (such as airplanes, bicycles, and cars), and objects (such as bottles, chairs, and TVs). Each image comes with an XML annotation file that provides detailed information about the object type, its location (bounding box), and occasionally its precise shape (segmentation mask). Pascal VOC 2012 supports a range of computer vision tasks like image classification, object detection, object and scene segmentation, and action recognition. Methods are evaluated using metrics such as precision, recall, and F1 score, with Average Precision (AP) and Mean Average Precision (mAP) being specifically used for object detection. The dataset is divided into training and test sets, allowing researchers and developers to validate and compare new algorithms. Pascal VOC 2012 serves as a crucial tool and a standard platform for benchmarking various methods in computer vision, playing a significant role in the field's development and innovation. However, this is an imbalanced dataset where the distribution among classes is uneven. For example, classes like "person" and "car" have many samples, while others like "sheep" or "sofa" have significantly fewer samples.

4.2. Parameters

The hyperparameters used for training the model are listed in Table 2. During the model training process, the parameters used play a crucial role in optimizing performance and avoiding overfitting. First, Weight Decay with a value of 0.0001 is a regularization technique that adds a penalty to the loss function related to the model's weights, thereby preventing overfitting. Dropout at 20% is another regularization technique, where 20% of the units in the network are randomly dropped during training to improve the model's generalization ability. The AdamW optimization algorithm combines Adam and Weight Decay, effectively optimizing the adjustment of the model's weights. Position Embeddings use the sine function to encode positions, helping the model recognize the order and position of elements in the data sequence. Finally, the Learning rate with a value of 0.00001 is a small learning rate, ensuring stable training and preventing overlearning. These parameters are carefully chosen to ensure effective learning and to avoid issues related to overfitting.

Table 2: Training parameters

Parameter	Value
Weight Decay	0.0001
Dropout	20%
Optimizer	AdamW
Position Embeddings	Sin
Learning Rate	0.00001
Activation	GELU
Epochs	100
Encoder/Decoder Layers	6
Attention	Linear attention
Object Queries	100

4.3. Experimental results

As illustrated in Figure 6, our model shows a significant reduction in error early in the epochs and steadily converges in subsequent epochs. The error rate decreases rapidly in the initial epochs, reflecting the model's rapid learning as it explores and learns complex data representations. Later epochs demonstrate a more stable error rate, indicating that the model has begun to converge after initial training exploration. Compared with the original DETR model, our proposed model achieves faster convergence. This enhancement is attributed to the integration of high-level image feature extraction, particularly leveraging Swin Transformer v2's capability to process large input sizes and integrating features at multiple levels. The Linear Attention mechanism within Linformer reduces computational complexity, enabling accelerated training while maintaining high performance.

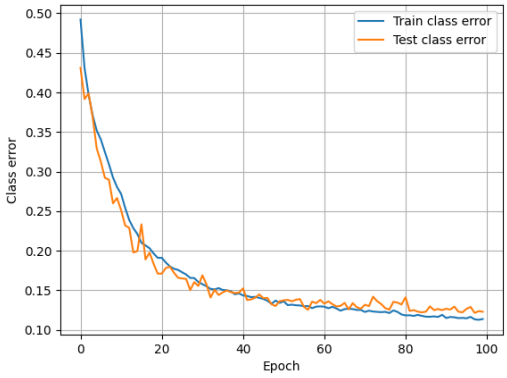


Figure 6: Classification error rates on the training and test datasets

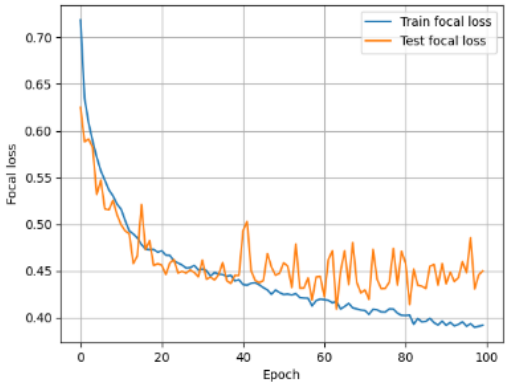


Figure 7: The error rate on the training dataset and the test dataset

In this study, we investigated the effectiveness of the Focal Loss function in enhancing our model's classification performance on small and imbalanced datasets. The results, shown in Figure 7, reveal a notable reduction in error rates on both the training and test sets compared to the traditional Cross-Entropy loss function. Focal Loss focuses on hard-to-classify examples, mitigating the influence of easier ones and thereby improving the model's accuracy and generalization. These findings highlight the critical role of loss function selection

in optimizing machine learning models, especially when facing limited or imbalanced data.

The confusion matrix provides a concise overview, visualizing the accuracy of the model's predictions compared to the actual outcomes. For a model with 20 classes, the confusion matrix will have a size of 20×20, as illustrated in Figure 8. In this matrix: $cm(i, i)$ represents the number of samples from class i that were correctly predicted (True Positives for class i), while $cm(i, j)$ indicates the number of samples from class i that were misclassified as class j (False Positives for class j). This matrix is crucial for evaluating the classification model's performance and identifying its error patterns.

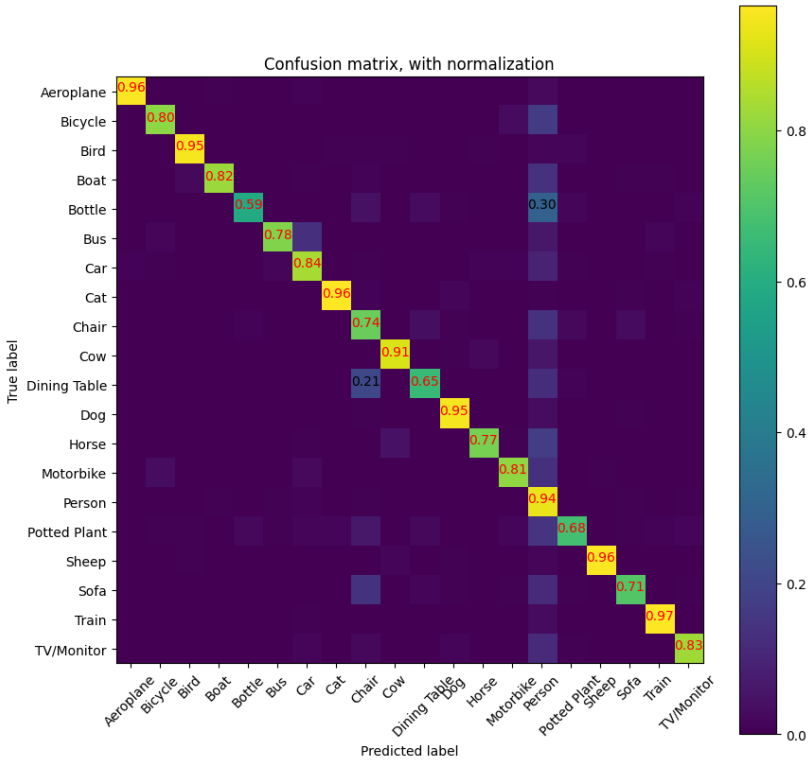


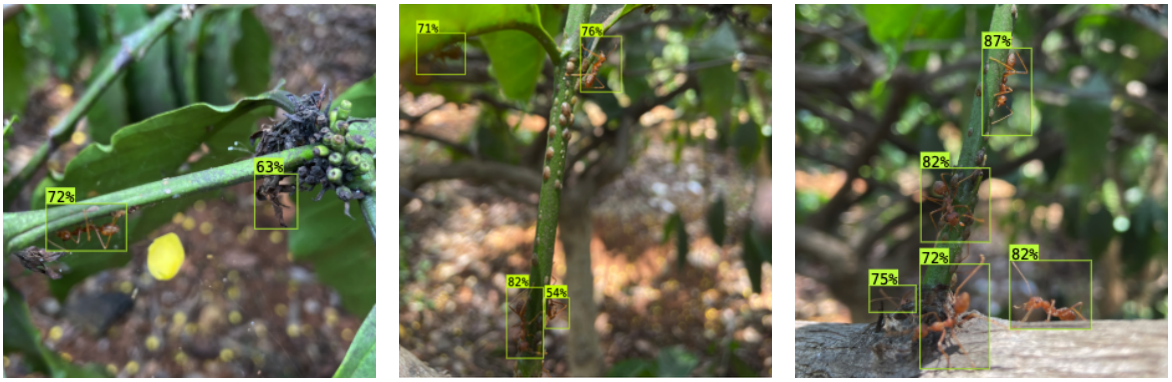
Figure 8: Confusion matrix on the Pascal VOC test dataset

Additionally, the confusion matrix helps us identify classes that the model frequently confuses with each other, allowing us to adjust the training dataset or improve the model features to minimize these errors. For instance, if class A is often confused with class B, we might consider adding more training data for these two classes or enhancing the features to better differentiate them. Moreover, the confusion matrix can be used to calculate various performance metrics such as overall accuracy, per-class accuracy, precision, and recall. These metrics provide a deeper insight into the model's performance and guide subsequent improvement steps.

In our analysis, we observed that the model achieved high precision and recall when classifying objects with ample training data. Conversely, the model's performance diminished when dealing with objects that were underrepresented in the training set, such as the classes Bottle, Dining Table, Potted Plant, and Sofa. This highlights the Transformer-based model's reliance on large datasets for optimal performance.

In Table 3, the comparison of various architectures based on mean Average Precision (mAP@0.5) emphasizes the significant influence of different backbones and methodologies. Particularly noteworthy is the S-DETR model, which incorporates both Linformer and Swin Transformer v2 as backbones. Our adaptations and enhancements to S-DETR have shown substantial improvements in performance compared to traditional methods, highlighting the potential for advancing object detection capabilities. This approach not only enhances existing methodologies but also positions models based on Linformer and Swin Transformer v2 as promising contenders in computer vision. In addition to accuracy, we also examined the practical efficiency of our model. Specifically, the average inference time per image is approximately 181 ms, with maximum GPU memory usage of 1583 MB, which demonstrates the method's suitability for deployment in real-world scenarios with limited computational resources. These results confirm that our approach maintains competitive accuracy while being resource efficient. Overall, the findings in Table 3 highlight the flexibility and performance gains achieved by integrating Linformer and Swin Transformer v2 into the DETR architecture, setting a precedent for future developments in object detection research and exploring new possibilities in global feature integration for enhanced object classification and localization.

To evaluate the effectiveness of our proposed model, we conducted fine-tuning on the Natural Enemy dataset, which consists of 551 annotated images collected from diverse natural environments. The dataset contains species such as dragonflies, ants, butterflies, and other natural pest enemies, characterized by small object sizes, high object density, and complex backgrounds with vegetation and varying illumination. These characteristics make the dataset particularly challenging for detection models, as objects often occupy only a tiny fraction of the image and frequently overlap with each other. After fine-tuning, our model achieved a mAP@0.5 of 50.4%, which represents a significant improvement compared to the baseline DETR model with 43.0%. This result demonstrates that our approach is more effective in handling small-scale objects and complex ecological scenes, thus providing a more robust solution for natural enemy detection tasks. Some inference results are illustrated in Fig. 9.



(a) (b) (c)

Figure 9: Some inference results

Table 3: Comparison of different detection models

Model	Epochs	mAP@0.5(%)	Parameters(M)	GFlops
DETR [9]	100	31.4	41	96
RetinaNet [27]	100	59.2	36	97
FCOS [29]	100	57.5	32	95
ATSS [30]	100	56.2	36	96
VFNet [31]	100	55.1	37	97
S-DETR (Our)	100	**87.0**	214	222

4.4. Ablations study

Importance of the backbone. To validate the importance of selecting a backbone for the model's learning process, we replaced the backbone sequentially with different models, including Vision Transformer [22], Swin Transformer [20, 21], and Vision Permutator [32]. The results of the convergence processes are depicted in Figure 10.

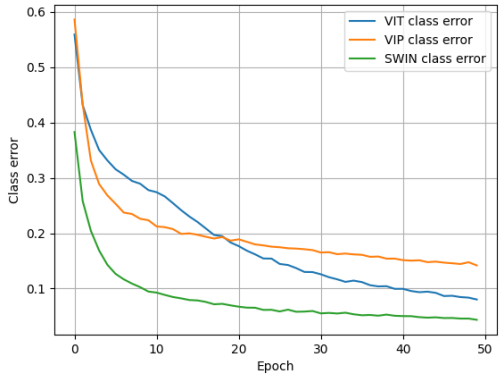


Figure 10: Convergence speed based on different backbones

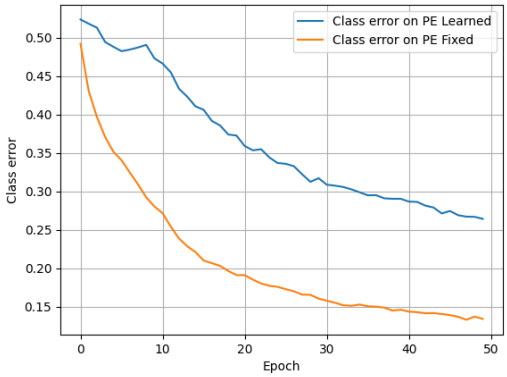


Figure 11: Convergence process based on two types of PE

Importance of positional encodings. There are two types of PE commonly used in model construction: fixed PE and learnable PE. We experimented with both types in our proposed model. The convergence process results are shown in Figure 11. We found that although both types of PE help the model converge, the convergence rate of the model using fixed PE is much faster than that of the model using learned PE.

Table 4: Ablation study on attention mechanisms (mAP@0.5)

Model	Attention Mechanism	mAP@0.5(%)
DETR [9]	Global Self-Attention	31.4
DAB-DETR [33]	Dynamic Anchor Attention	59.4
Conditional-DETR [8]	Conditional Attention	60.7
Deformable-DETR [7]	Multi-Scale Deformable Attention	62.2

Importance of attention mechanisms. We compare different types of attention mechanisms in Transformer-based detectors. Table 4 shows the results. The baseline DETR with standard self-attention reaches only 31.4 mAP@0.5. By incorporating dynamic anchor

queries, DAB-DETR improves the accuracy to 59.4. Conditional-DETR further enhances the performance to 60.7 by leveraging conditional spatial priors. Deformable-DETR achieves 62.2 mAP@0.5 with multi-scale deformable attention, demonstrating the benefits of adaptive feature sampling. Finally, our proposed model achieves 77.6, highlighting the effectiveness of our enhanced attention design.

5. CONCLUSION

In this work, an object detection solution is proposed based on the Transformer model using multiple attention mechanisms, called S-DETR. S-DETR includes two main enhancements: utilizing the Swin-Transformer 2 model as an image feature extractor, and an encoder that processes multi-scale features to improve object detection performance. These improvements enable our proposed model to achieve high efficiency and accurately detect objects in images by combining features at different levels and resolutions. The encoder uses an attention mechanism that focuses only on the important features of the image rather than the entire image, thereby allowing the model to quickly detect objects without consuming as much time as previous models.

REFERENCES

[1] V.-D. Hoang, N. T. Huynh, N. Tran, K. Le, T.-M.-C. Le, A. Selamat, and H. D. Nguyen, "Powering ai-driven car damage identification based on vehide dataset," *Journal of Information and Telecommunication*, pp. 1–19, 2024.

[2] A. M. Founta, D. Chatzakou, N. Kourtellis, J. Blackburn, A. Vakali, and I. Leontiadis, "A unified deep learning architecture for abuse detection," in *Proceedings of the 10th ACM Conference on Web Science*, 2019, pp. 105–114.

[3] T.-C. Pham, A. Doucet, C.-M. Luong, C.-T. Tran, and V.-D. Hoang, "Improving skin-disease classification based on customized loss function combined with balanced mini-batch logic and real-time image augmentation," *IEEE Access*, vol. 8, pp. 150 725–150 737, 2020.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 5998–6008. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[6] T. T. A. Loan, P. T. Anh, L. V. Nam, and H. V. Dung, "An effective deep learning model for recognition of animals and plants," *Journal of Computer Science and Cybernetics*, vol. 38, no. 1, pp. 15–29, March 2022.

[7] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," *arXiv Preprint arXiv:2010.04159*, 2020.

[8] D. Meng, X. Chen, Z. Fan, G. Zeng, H. Li, Y. Yuan, L. Sun, and J. Wang, "Conditional detr for fast training convergence," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 3651–3660.

[9] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European Conference on Computer Vision*. Springer, 2020, pp. 213–229.

[10] D. Nguyen, V.-D. Hoang, and V.-T.-L. Le, "V-detr: Pure transformer for end-to-end object detection," in *Intelligent Information and Database Systems*. Singapore: Springer Nature Singapore, 2024, pp. 120–131.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[12] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 39, no. 06, pp. 1137–1149, June 2017.

[13] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 2016, pp. 21–37.

[15] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.

[16] L. Dong, S. Xu, and B. Xu, "Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5884–5888.

[17] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, "Neural speech synthesis with transformer network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 6706–6713.

[18] M. Topal, A. Bas, and I. van Heerden, "Exploring transformers in natural language generation: Gpt, bert, and xlnet. arxiv 2021," *arXiv Preprint arXiv:2102.08036*.

[19] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," 2014.

[20] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 012–10 022.

[21] Z. Liu, H. Hu, Y. Lin, Z. Yao, Z. Xie, Y. Wei, J. Ning, Y. Cao, Z. Zhang, L. Dong *et al.*, "Swin transformer v2: Scaling up capacity and resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 009–12 019.

[22] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv Preprint arXiv:2010.11929*, 2020.

[23] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 11 976–11 986.

[24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. Ieee, 2009, pp. 248–255.

[25] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," *arXiv Preprint arXiv:2006.04768*, 2020.

[26] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals of Operations Research*, vol. 134, pp. 19–67, 2005.

[27] T.-Y. Ross and G. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2980–2988.

[28] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, pp. 98–136, 2015.

[29] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9627–9636.

[30] S. Zhang, C. Chi, Y. Yao, Z. Lei, and S. Z. Li, "Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9759–9768.

[31] H. Zhang, Y. Wang, F. Dayoub, and N. Sunderhauf, "Varifocalnet: An iou-aware dense object detector," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8514–8523.

[32] Q. Hou, Z. Jiang, L. Yuan, M.-M. Cheng, S. Yan, and J. Feng, "Vision permutator: A permutable mlp-like architecture for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 1328–1334, 2022.

[33] S. Liu, F. Li, H. Zhang, X. Yang, X. Qi, H. Su, J. Zhu, and L. Zhang, "Dab-detr: Dynamic anchor boxes are better queries for detr," *arXiv Preprint arXiv:2201.12329*, 2022.

Received on July 09, 2024
Accepted February on 07, 2026