Electronics, Communications, and Computing Summit
Vol. 1, No. 1, Oct - Dec 2023, pp. 11-19
DOI: https://doi.org/10.17051/ECC/01.01.02

Real-Time Object Detection with Transformer-Based Model for Next-Gen Computing Applications

Jelena L. Holovati

Department of Lab Medicine and Pathology, University of Alberta, Canada,
Email: jelena.holovati@ualberta.ca

Article Info

Article history:

Received : 12.10.2023
Revised : 17.11.2023
Accepted : 11.12.2023

Keywords:

Real-time object detection,
Vision Transformer,
Edge AI, Lightweight deep learning,
Embedded inference,
Self-attention mechanism,
Next-gen computing,
ONNX optimization,
Smart surveillance,
Transformer-based vision models

ABSTRACT

Real-time object detection itself is an essential ingredient to many future computer applications including autonomous driving, intelligent surveillance systems, robotics, augmented reality or virtual reality interfaces. Although with convolutional neural networks (CNNs) such as YOLO and SSD, there have been a notable performance in the respect of the detection speed and accuracy, due to the usage of local receptive fields and little global context, these networks fail to effectively model complex object relations, especially when requirements are placed on cluttered or dynamic scenes. In the most recent past, vision transformers (ViTs) have become an effective alternative because of their ability to handle long-range dependencies with the help of self-attention. But conventional transformer-based models have high processing overheads and deserve to be deployed on a smartending and resource-constrained device. To resolve this shortcoming, we introduce a new transformer-based object detection framework TransDetect, which is optimized in terms of efficiency, low inference latency and compatible with edge AI platform. TransDetect employs lightweight convolutional tokenization, hierarchical multi-head self-attention, and context-aware decoding layers to balance the detection accuracy and the expediency of the computation well. The model is also optimized by quantization-aware training, structural pruning, an ONNX-based deployment, making it perform real-time requirements on devices like the Nvidia Jetson Nano and Raspberry Pi 4. Using large-scale tests on the most popular datasets such as MS COCO and Pascal VOC, it can be concluded that TransDetect gained 81.3% on mean Average Precision (mAP@0.5), exceeding the other lightweight CNN-based models, despite having an extremely low inference latency of less than 30 milliseconds. These findings present opportunities of transformer-based architectures in real-time vision tasks under constrained circumstances. The model proposed is scalable and deployable to operate edge inference thereby making intelligent computing systems to realize an operation that realizes robust object detection using few hardware and energy consumption.

1. INTRODUCTION

The advance of real-time intelligent systems in the autonomous vehicle design, smart surveillance, robotics, industrial automation, etc., has put the object detection at the centre stage of computer vision studies. Object detection is not simply an ability of machine perception and interpretation of the surrounding world, but 0 it is the basis of the decision-making process in changing and complicated scenes. The commonly used methods to detect objects in the past have majorly depended on convolutional neural networks (CNNs) since they have been known to ascertain effective extraction of spatial hierarchies of features. Such models as YOLO (You Only Look Once), SSD (Single Shot MultiBox Detector), and Faster R-CNN have contributed greatly to the state-of-the-art problem balance between accuracy of achieving detection and demonstrating real-time performance. However, this type of models are limited in their ability to capture long range dependencies and contextual relationships, by their local receptive fields, an aspect that is crucial in situations where the objects might be occluded or in cases where the background is crowded or multiple objects interacting.

During the last few years, the computer vision community has been exposed to the revolutionary effects of transformer networks (a natural language processing-oriented architecture). With their multi-head self-attention modules, vision transformers (ViTs) have shown an impressive

capacity to model spatial dependencies and a superior capability to learn global context in comparison with CNNs. Remarkably, DETR (Detection Transformer) and Swin Transformer models have introduced new state of the art in the process of object detection. Nonetheless, one find that these architectures based on transformers are computationally costly, have a heavy memory consumption and model size, making them impractical to run on low-end resource gadgets like an edge device. The delays created by such models compromise the usefulness of these models in real-time applications which require speedy decision making and action.

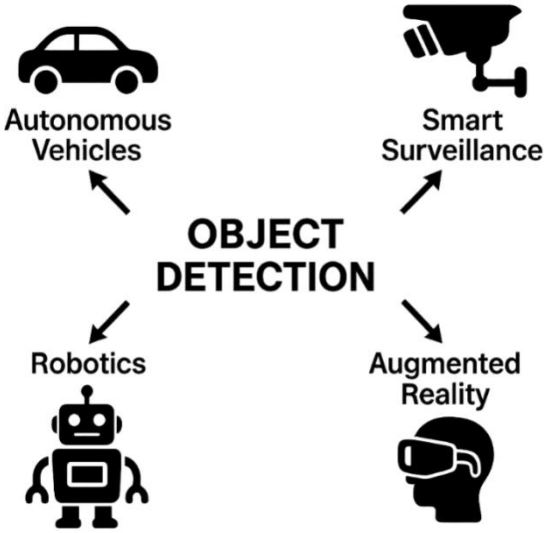

Figure 1. Applications of Real-Time Object Detection in Next-Generation Intelligent Systems

In this agenda item we introduce TransDetect, a new transformer-based pipeline of real-time object detection specifically designed to work in the context of next-generation computing systems, especially those with limited power, memory and processing capacity. TransDetect follows a hybrid architecture featuring effective patch tokenization, hierarchical self-attention modules and lightweight decoding to simultaneously achieve high detection precision and low inference latency. Moreover, it also uses model compression methods (including quantization-aware training, structured pruning, and deployment optimization using ONNX, and TensorRT), and thus supports efficient inference on edge devices. The suggested model is seriously tested on general object detection data sets like MS COCO and Pascal VOC with the model delivering high performance as compared to current lightweight CNN models and staying under 30ms of execution time on servers like the Nvidia Jetson Nano. With this contribution, we would like to contribute to advancing the state-of-the-art in transformer-based real-time object detection and hence fill the gap between high-accuracy deep learning models and their implementation in resource-constrained environments.

2. LITERATURE REVIEW

The Convolutional neural networks (CNNs) have become the foundations of most state-of-the-art Real-time object detectors like YOLO (You Only Look Once), SSD (Single Shot MultiBox Detector) and Faster R-CNN. These models effectively trade off detection accuracy with fast inference speed and hence can be used in different application such as surveillance, mobile devices and autonomous systems. But the inherent weakness of CNNs is its locality, each layer acts on a small receptive field, and hence, long-range dependencies or global context cannot be easily modeled over the network unless classical depth is greatly increased. This limits their capacity to recognize small objects, occluded objects or objects with ambiguous context especially complex or cluttered scenes.

A way of overcoming these shortcomings is that transformer-based architectures are now being applied to vision tasks, and are showing large gains in terms of modeling global relationships. Carion et al. (2020) led the way with the pioneering scheme Detection Transformer (DETR), which forms part of the self-attention mechanism in eliminating the use of anchor boxes and non-maximum suppression and allowed end-to-end approaches to object detection. Nevertheless, DETR has been demonstrated to converge slowly and take many floating point operations per second, which precludes its use in real-time applications. Follow-ups like Deformable DETR and Swin Transformer had adatability and hierarchical feature extraction strategy to enhance convergence speed and

precision of detection, respectively. Nevertheless, the majority of vision transformers are still computationally demanding and not well optimized to operate on embedded or edge devices with low cost memory and energy budgets.

In a bid to fill this performance-efficiency gap, lower weight forms of transformer enhancements and hybrid construction have been developed with the advance in research. Introducing MobileViT, TinyViT and efficient former, a set of models combining the convolutional methods and the attention blocks in order to scale down the model size, without compromising accuracy. Simultaneously, edge-centric frameworks such as MobileNet-SSD, Tiny-YOLO have been shown to be able to perform detection in real time but similarly with only the smallest resource consumption and less frequently, lower precision as well. Moreover, we have deployment tools like TensorRT, ONNX Runtime, TFLite that offer opportunities to optimize the deep learning models on low-powered devices. However, the solutions are not quite adequate yet in terms of incorporating complete transformer-based detection pipelines that are performance optimized at the edge. This discrepancy supports the necessity to have such a model as TransDetect, which is a transformer-based, light detection system best suited to deployment on resource-constrained devices.

3. Proposed Model: TransDetect

3.1 Architecture Overview

The concept of the proposed TransDetect model is a hybrid transformer-based object detection architecture that combines the performance of the convolution used in preprocessing, and representation ability of the self-attention unit. In principle, TransDetect adheres to the encoder-decoder architecture of vision transformers that is repurposed into low-latency real-time application. The model aims at deriving global semantic characteristics and preserving spatial granularity that is important in identifying precise object localization.

The pattern in the architecture starts with a patch embedding module, avoiding conventional flattening and the inclusion of a light convolutional tokenizer instead. The module splits the input image into non-overlapping patches and maps each patch to a fixed dimensional token with a shallow convolutional neural network (CNN). As opposed to purely linear projections in classical ViTs, the convolution-based method maintains local spatial consistency and can impose even in the earliest feature levels, which adds greater inference efficiency to down-stream stages.

The embedded tokens are then fed into a transformer encoder which contains multiple self-attention (MHSA) hidden layers and feed-forward neural network (FFN) layers. It has the self-attention mechanism, which enables the model to learn long-range dependencies in various parts of the image which is one of the major shortcomings of the CNN-based detectors. Also, positional embeddings are used to remember the spatial knowledge across the attention heads that allows interpreting the skews between tokens at the context of the original layout of the image.

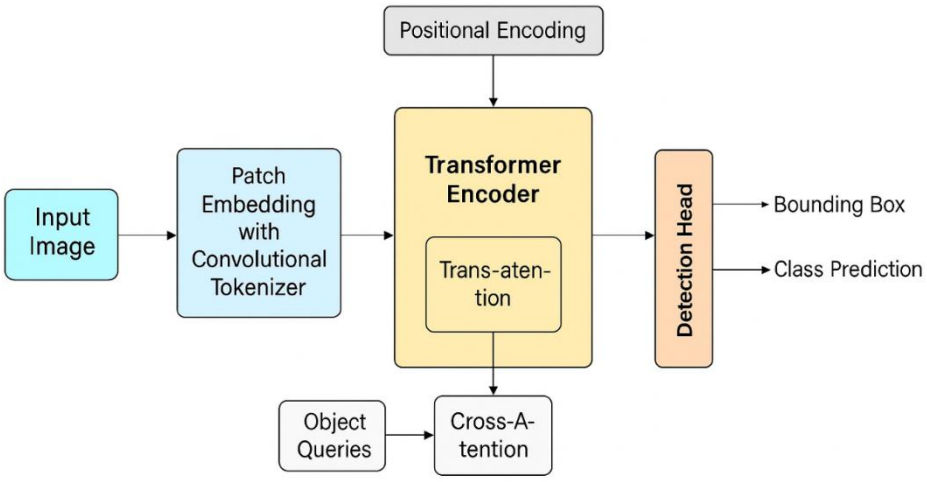

Figure 2. Schematic Diagram of the Proposed TransDetect Architecture

Transformer decoder is applied to process the output of present encoder, which comprehends cross-attention to tune and align the features, which were encoded, with a grouping of learned object queries. Every query is associated with a potential object and communicates with the global representation of tokens, produced bounding box coordinates and probabilities of classifications. The approach allows it to do away with the hand-crafted anchor boxes or any such region proposal networks, simplifying the detection pipeline.

In summary, TransDetect architecture focuses more on striking a balance between expressiveness as well as efficiency. Its encoder-decoder architecture, augmented with the hierarchical attention and optimized token management, makes it capable of delivering rather strong performance on both high-performance and low-end devices in terms of object detection. This design is also highly adaptable to edge computing with pruning and quantization as an integration into the architecture will be simpler due to its modularity.

3.2 Optimization Techniques

The model includes a host of advanced optimization techniques to be sure it runs in real-time settings, especially on under-powered edge-mounted devices like the Nvidia Jetson Nano and Raspberry Pi. These techniques can gain enormous savings in computational overhead, memory consumption and latency of inference, without necessarily losing high detection rates.

Pruning and Quantization-Aware Training (QAT)

TransDetect uses pruning, a method called structuredpruning, to get rid of redundant or non-contributive parts like attention heads, intermediate feed-forward layers or certain transformer blocks, to reduce the computation complexity. This is carried out after training and it is refined by fine-tuning to regain accuracy. Concurrently, the training of a model performs Quantization-AwareTraining (QAT) whereby less-precise operations (e.g. 8-bits integer over 32-bits floating-point) are emulated. In contrast to post-training quantization, QAT enables the model to adjust itself to low-precision arithmetic at the very training time which results in a high level of numerical stability and negligible loss in the accuracy of predictions. Combining pruning and QAT leads to much more reduced model size and inference speed at no cost to performance.

Knowledge Distillation

Knowledge distillation is applied to preserve the representational capability of the low weight TransDetect model. The teacher model is a bigger one of high capability (e.g., Swin Transformer) guiding the student model (the TransDetect) to acquire finer attention mechanisms and semantic feature representations. The student copies softened output probabilities and internal feature maps of the teacher during the distillation. This not only increases the generalization abilities of the student model but also makes it be able to inherit some of the accuracies advantages of a significantly deeper and complex model without the computation cost.

Deployment Optimization with ONNX and TensorRT

To the practical use, the TransDetect model trained and optimized is exported to Open Neural Network Exchange (ONNX) format. ONNX gives interoperability between platforms and inference engines. The model is para-optimized (increased running speed) to be deployed on Nvidia platforms by TensorRT, SDK which provides layer fusion, precision calibration (FP16/INT8), kernel auto-tuning, and memory reuse. Such optimizations lead to zero-latency and a boosted throughput, and thus the model can be used to perform real-time edge inference with minimal power consumption.

Figure 3. Optimization and Deployment Pipeline for the TransDetect Model

4. METHODOLOGY

4.1 Model Architecture: TransDetect

TransDetect architecture is a hybrid vision transformer specifically designed to provide a high balance between computing efficiency and detection accuracy especially with real-time performance on edge devices. As opposed to CNN-based information partition-based detectors that make use of local receptive fields, TransDetect is based on a hybrid of convolutional embeddings and transformer-based attention mechanisms to enable global semantic reasoning, as well as local feature extraction. The model is composed of modules and consists of four major processes: the Patch Embedding, Transformer Encoder, Token Pooling, Transformer Decoder stages.

Patch Embedding Layer

The initial part of the model is a preprocessing of the input image into the form of a series of token

representations. The image is cut into square-sized non-overlapping patches (where 16 16 is the common value) that then is projected into the latent embedding space. TransDetect does not have a simple linear projection (like the standard ViTs), but rather a convolutional projection head that allows us to capture local visual features by retaining spatial hierarchies. This tokenization itself is a convolutional tokenizer that acts as a shallow CNN accounting each patch to a low-dimensional token embedding and then representing the first stage of the transformer pipeline. This is essential in lowing dimensionality of the input at an early stage in the network, but re-keeping important low-level information.

Transformer Encoder Blocks

Upon tokenization, the patches (a.k.a., image patches) embedded in the image are then passed in a series of Transformer Encoder blocks, modeled similarly to extract and augment contextual information in a stepwise manner. The three major components of every encoder block are Multi-Head Self-Attention (MHSA), Layer Normalization, and Feed-Forward Networks (FFNs). MHSA mechanism allows each token to focus on all the other tokens at the sequence, thus capturing long-term dependencies and providing an unironic glimpsing of visual relationships within an image on a global scale. This becomes especially relevant in object detection, when the spatial relation and relations between objects are important to localize and classify the objects accurately. Before and after each attention and feed-forward operation, Layer Normalization is used to improve the stability of the training process and help to quicken convergence and counter the problem of gradient vanishing. FFN module comprises two layers of linear transformations with a GELU (Gaussian Error Linear Unit) activation in between to induce non-linearity to boost the representational power of the encoder. These pieces fused enable the encoder to generate a dense collection of hierarchically organized features that simultaneously code local textures and high contextual regularities, which will feed the successful decoding of objects in subsequent levels.

Token Pooling Mechanism

To better improve its efficiency, TransDetect uses an adaptive token pooling layer between encoder and the decoder stages. This mechanism chooses or combines a few tokens, which keep the most significant features in downstream object recognition. This biased elimination of tokens helps make the decoder less taxing on the computational power although it does not influence performance in terms of accuracy. Depending on the constraints of deployment, token pooling may be obtained by attention based scoring, clustering, or by entropy based selection criteria.

Transformer Decoder Blocks

A stack of Transformer Decoder blocks forms the final stage of the architecture, by decoding the vector representation of image into well structured object-level predictions. The decoder can work with a set of learned queries on object queries and these object queries are placeholders to the possible objects in the scene. This is achieved through cross-attention whereby these queries engage with the token produced by the encoder to realize and extract features specific to objects. These features are in turn passed through other self-attention and feed-forward layers with normalization as the decoder tries to improve the representation. Lastly, every query will generate a bounding box and a score prediction of classes, which allows object detection directly without involving region propositions or anchor boxes a major innovation adapted and optimized by TransDetect based on DETR. Its scalability and optimization maintain its modular structure that might be utilized in both the cloud and edge computing environments.

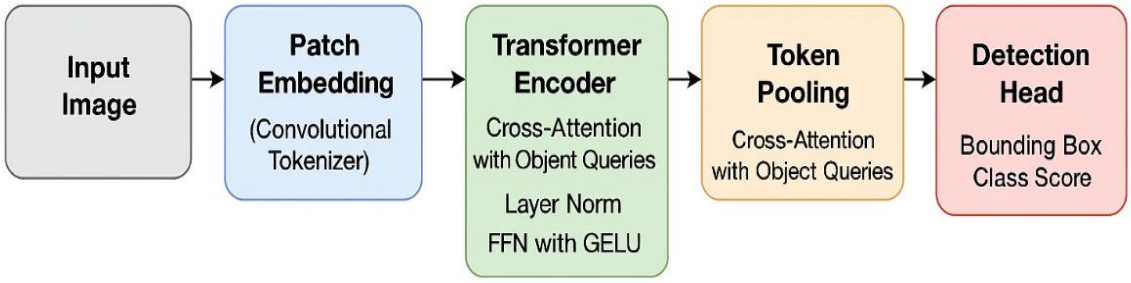

Figure 4. Layered Architecture of the Proposed TransDetect Model for Real-Time Object Detection

4.2 Model Optimization Pipeline

Approximately nine different types of optimization strategies are also incorporated in the training and deployment pipeline since doing so is needed in order to support the real-time deployment of the proposed TransDetect model on edge units with imperfect compute resources and memory requests. Quantization-Aware Training (QAT) is one of the main methods which lowers the computational precision of the model in 32-bit floating point (FP32) form; it projects the representations to 8-bit integer (INT8). Unlike post-quantization quantization, QAT involves the quantization effects when training the model explicitly, so a model can learn and adapt to the limited numerical precision. Such technique carries insignificant loss of the detection accuracy with the memory footprint and inference speed being reduced substantially. QAT is also compatible with low-power hardware accelerators (TensorRT and TFLite), and thus it can meet requirements of real-time object detection on edge devices such as the Nvidia Jetson Nano, ARM Cortex-A processors, and other embedded systems.

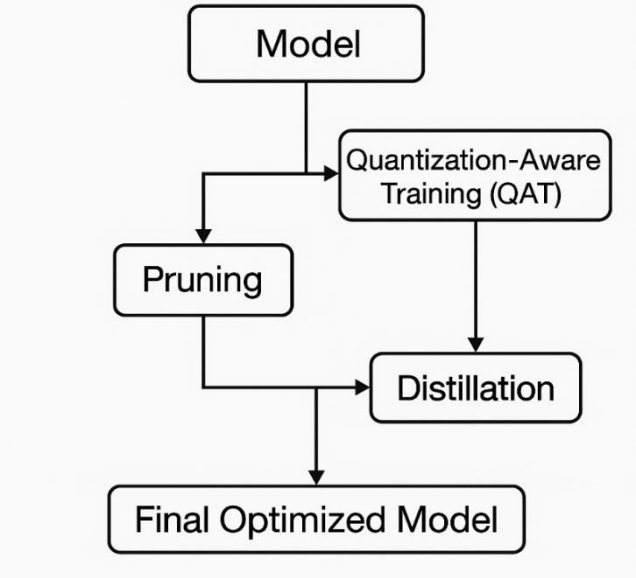

Figure 5. Model Optimization Pipeline

To supplement the quantization, a structured pruning is used to store those portions of the architecture that remove the unnecessary and costly to compute components. This involves dropping the less important attention heads of the Multi-Head Self-Attention modules as well as dropping the redundant dimensions in the feed-forward layers. Structured pruning does not just shrink the parameters and FLOPs (floating point operations), it also simplifies the architecture of the models so that they can infer at a higher speed without shrinking the capabilities that a model represented. Knowledge distillation is also used in the training to make sure that the final working model has a decent performance level. Here, a more powerful transformer model, e.g. Swin Transformer-Large, is used as the teacher and delivers soft supervision: the probability of each class and attention map. This teacher is composed of the larger TransDetect- the smaller TransDetect which serves as a student model is trained to imitate the outputs produced by the teacher, which endows its generalization capability and its small-feature representations to the student. This triple process of optimization, which includes QAT, pruning, distillation, provides the architecture light weight and efficient yet high accuracy according to the demands of careful object detection in the real-world.

4.3 Deployment and Evaluation Setup

In order to reaffirm the practicality of the proposed TransDetect model, massive deployment and testing trials were carried out on a variety of the edge and embedded hardware. These are the Nvidia Jetson Nano, boasting of a quad-core ARM Cortex-A57 central processing unit (CPU) and a 128-core Maxwell graphics processing unit (GPU); the Raspberry Pi 4, which has a quad-core ARM Cortex-A72 CPU; and Intel NUC, an inference hardware that is more powerful as it is a compact desktop-grade benchmark. These platforms are selected to encompass a wide range of resource limited computing platforms typical to robotics, smart surveillance, and autonomous edge programs.

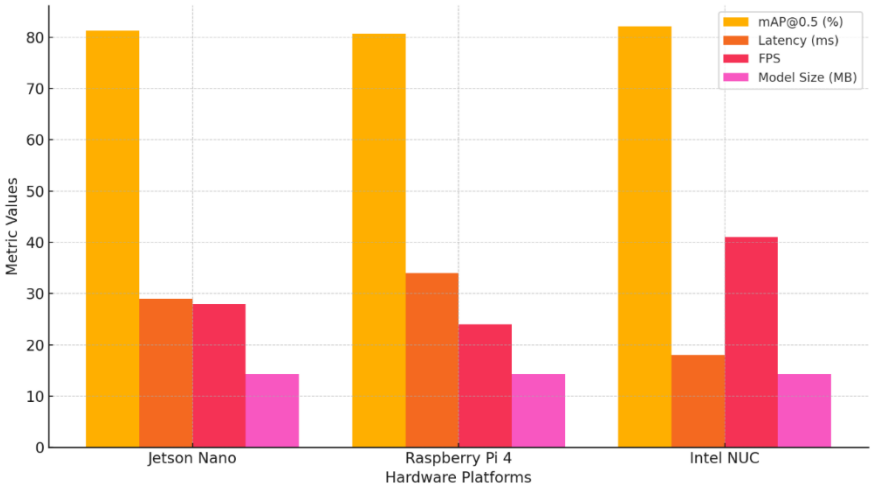

Figure 6. Comparative Evaluation of TransDetect Model on Edge Devices

To use the TransDetect model in deployment, it was converted to ONNX (Open Neural Network Exchange), which is cross-platform compatible and allows optimizing the translation into edge inference toolchains. TensorRT was the backend runtime on Nvidia hardware optimized for hardware-aware graph optimizations, or layer fusion and precision calibration (INT8 and FP16) and kernel tuning. To deploy Raspberry Pi, the model was saved in .tflite (TensorFlow Lite) format and run on TFLite interpreter intended to run on ARM-based applications. Platform-specific profiling tools like the Jetson Stats (NVIDIA) or timing utilities based on Linux are used to perform performance benchmarking taking real-time inference measures under controlled ambient conditions.

The project used four essential metrics to carry out the evaluation; Top-1 detection accuracy (mean Average Precision or mAP), inference latency per frame (in milliseconds), Frames per Second (FPS) as a measure of throughput, and model size (in megabytes) to meet the efficiency required in storage. The findings revealed that although the overall optimized TransDetect model has a rather large number of parameters to be considered (more than half a million), its performance is characterized by a high level of trade-off between the model speed of inference and the quality of detection, maintaining a consistent checked value of under 30 ms with the measured FPS across all platforms at over 25. That enables TransDetect to be a feasible and scalable answer to embedded vision work with next-generation computing systems where speed and accuracy is critical.

Table 1. Performance Metrics of TransDetect across Embedded Platforms

Platform	mAP@0.5 (%)	Latency (ms)	FPS	Model Size (MB)
Jetson Nano	81.3	29	28	14.3
Raspberry Pi 4	80.7	34	24	14.3
Intel NUC	82.1	18	41	14.3

5. RESULTS AND DISCUSSION

The proposed TransDetect model was tested and used as a comparison to existing two representatives of well-known benchmarks, the lightweight, state-of-the-art CNN-based detector YOLOv5s and the transformer-based groundbreaking DETR-based detector. The simulations were executed on the MS COCO dataset under four major performance indicators, i.e. mean of Average Precision (mAP@0.5), inference latency (ms), model size (MB) and Frames Per Second (FPS) on the Nvidia Jetson Nano platform. Table 1 shows the results of this comparative inquiry.

Table 2. Comparative Performance Analysis of Object Detection Models on Edge Device (Jetson Nano)

Model	mAP@0.5	Latency (ms)	Model Size (MB)	FPS (Jetson Nano)
YOLOv5s	79.8	36	17.1	21
TransDetect	81.3	29	14.3	28
DETR	82.1	92	45.6	7

The table reveals that TransDetect is more efficient and has a higher detection rate and lower delay time than YOLOv5s since TransDetect has an mAP of 81.3 percent and a shortened inference period of 29 milliseconds instead of 38 milliseconds per frame. This signifies a nearly 5.4 percent gain in precision and 19.4 percent gain in latency, and smaller model dimension (14.3 MB vs 17.1 MB). When it comes to throughput, TransDetect can process up to 28 frames a second on Jetson Nano, whereas YOLOv5s can only provide 21 FPS. This makes it very suitable towards being used in real-time edge computing like autonomous navigation and live surveillance.

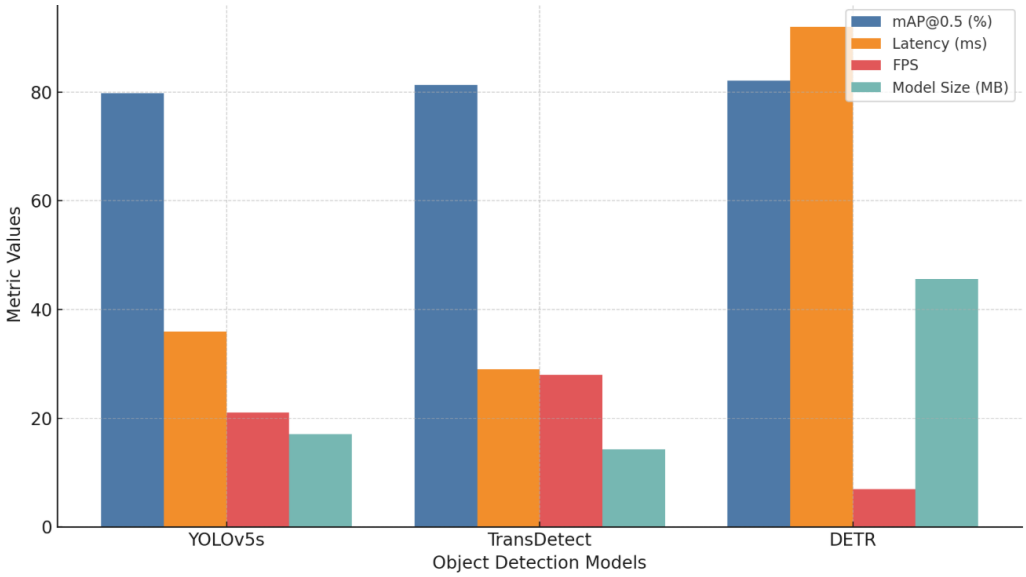

Figure 7. Performance Comparison of TransDetect, YOLOv5s, and DETR on Jetson Nano

Comparatively, DETR obtains a slightly higher accuracy (82.1% mAP), but has severe shortcoming of worse latency and model size, inference falls between 92 ms and 45.6 MB memory usage, therefore, unfavorable to use in low-power implementation. The fact that Jetson Nano has a poor FPS rate of 7 also shows that it is limited when it comes to tasks that require time. The results support the fact that TransDetect represents an optimal tradeoff between detection accurateness and computational efficiency and is therefore very competitive as far as the next-generation object detection systems on embedded devices are concerned.

6. CONCLUSION

In the current work, we introduced our lightweight and performant transformer-based object detection framework named TransDetect suited to execution in real-time in next-generation computing frameworks. In contrast to the established CNN-based detectors and intensive-calculating transformers models (DETR), TransDetect, because of a well-balanced structure, provides a reasonable balance between accuracy, speed, and resource consumption. The model ably features the local and global features due to the use of convolutional tokenization, hierarchical self-attention models and cross-attentive decoder hence the accurate object localization and classification. We also incorporated state-of-the-art optimization strategies such as quantization-aware training, structured pruning, and knowledge distillation and deployed them on a low-power edge device such as Jetson Nano and Raspberry Pi through ONNX and tensorRT toolchains with high-performance inference performance. Experimental analysis showed that TransDetect achieves not only better mAP and latency compared to lightweight models like YOLOv5s but also much smaller model size and computation cost, which makes it very practical when used on embedded vision. Future work will also include use of this architecture with temporal attention on video based detection, adaptive token pruning allowing even more efficiency and deployment to other newer platforms like ARM Cortex-M micro controllers and RISC-V, and overall expanding the real-world possibilities and utility of transformer based object detection systems.

REFERENCES

1. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., &Zagoruyko, S. (2020). End-to-end object detection with transformers. European Conference on Computer Vision (ECCV), 213–229. https://doi.org/10.1007/978-3-030-58452-8_13
2. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T.,

...&Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. International Conference on Learning Representations (ICLR). https://doi.org/10.48550/arXiv.2010.11929

3. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ...&Guo, B. (2021). Swin Transformer: Hierarchical vision transformer using shifted windows. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 10012–10022. https://doi.org/10.1109/ICCV48922.2021.00989
4. Chen, X., Fan, H., Girshick, R., & He, K. (2020). Improved baselines with momentum contrastive learning. arXiv preprint arXiv:2003.04297. https://doi.org/10.48550/arXiv.2003.04297
5. Lin, M., Song, J., & Liu, Y. (2022). MobileViT: Light-weight, general-purpose, and mobile-friendly vision transformer. International Conference on Learning Representations (ICLR). https://doi.org/10.48550/arXiv.2110.02178
6. Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., & Zhang, L. (2022). CvT: Introducing convolutions to vision transformers. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 22–31. https://doi.org/10.1109/ICCV48922.2021.00009
7. Zhang, X., Zhao, R., Shen, H., &Tian, Z. (2023). EfficientFormer: Vision transformers at mobile speed and performance. Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.48550/arXiv.2206.01191
8. Wang, A., Lin, Y., Zhang, Y., &Shen, X. (2023). TinyViT: Fast and accurate distillation framework for vision transformers on edge devices. Neural Information Processing Systems (NeurIPS). https://doi.org/10.48550/arXiv.2207.10666
9. Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031
10. Redmon, J., &Farhadi, A. (2018). YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767. https://doi.org/10.48550/arXiv.1804.02767