Revista de Informática Teórica e Aplicada - RITA - ISSN 2175-2745
Vol. 33, Num. 1 (2026) 41-49

RESEARCH ARTICLE

Transformers or Not: A Comparative Study on Defect Inspection in Printed Circuit Boards

Transformers or Not: Um Estudo Comparativo na Inspeção de Defeitos em Placas de Circuito Impresso

Pedro Luiz Henriques Benedetti[1]*, Marcelo Ricardo Stemmer[1]

Abstract: A Printed Circuit Board (PCB) is a fundamental component in the manufacturing process of electronic devices. When talking about the process of fabrication of these devices, the quality control stage represents, in general, the biggest cost in production. This stage is usually realized by humans, which are prone to fail, as well as being slower and less consistent when compared to computers. This study presents the utlization of six distincts Artificial Intelligence archtiectures trained for PCB defect detection. The results of the experiments highlighted key differences in training time, accuracy, and computational complexity across various models. YOGA-s achieved a good balance with the second-best mAP 50 accuracy (83.2%) and a moderate training time of 50 minutes. YOLOv11n, despite its shorter training time (40 minutes), outperformed YOGA-s in mAP 50-95 (42.7%). DETR demonstrated the highest accuracy (92.4% mAP 50), but at the cost of significantly longer training times and higher complexity. LW-DETR, while efficient, showed limited performance improvements as its scale increased. YOLOv11n emerged as the most cost-effective option for PCB inspection tasks.
Keywords: Automated Optical Inspection — Artificial Intelligence — Printed Circuit Board

Resumo: Uma placa de circuito impresso (PCI) é um componente fundamental na construção de dispositivos eletrônicos. Quando falamos sobre o processo de manufatura destes componentes, a etapa do controle de qualidade representa, em geral, o maior custo na produção. Esta etapa é geralmente realizada por humanos, que são propensos a falhas, além de serem mais lentos e menos consistentes quando comparados a computadores. O presente estudo apresenta a utilização de seis arquiteturas distintas de Inteligência Artificial treinadas visando a detecção de problemas em PCIs. Os resultados dos experimentos destacaram as principais diferenças no tempo de treinamento, na precisão e na complexidade computacional dos vários modelos. O YOGA-s obteve um bom equilíbrio com a segunda melhor precisão do mAP 50 (83,2%) e um tempo de treinamento moderado de 50 minutos. O YOLOv11n, apesar de seu tempo de treinamento mais curto (40 minutos), superou o YOGA-s no mAP 50-95 (42,7%). O DETR demonstrou a maior precisão (92,4% mAP 50), mas ao custo de tempos de treinamento significativamente mais longos e maior complexidade. O LW-DETR, embora eficiente, apresentou melhorias limitadas de desempenho à medida que sua escala aumentava. O YOLOv11n se mostrou a opção com melhor custo-benefício para tarefas de inspeção de PCI.
Palavras-Chave: Inspeção Ótica Automatizada — Inteligência Artificial — Placa de Circuito Impresso

[1] *Departamento de Automação e Sistemas, Universidade Federal de Santa Catarina (UFSC), Brasil*
***Corresponding author**: bndttipedro@gmail.com
DOI: http://dx.doi.org/10.22456/2175-2745.145580 • **Received:** 22/02/2025 • **Accepted:** 10/12/2025

1. Introduction

A printed circuit board (PCB) is a fundamental component in the construction of electronics. When it comes to the manufacturing of these components, quality control generally represents the largest portion of production costs. This is because the quality of PCB production has a significant impact on the performance of various electronic products. In addition to the high cost, quality control work is mostly performed by humans, who are prone to errors and are also slower and less consistent compared to automated computational systems [1].

Research shows that human operators present effectiveness between 80% and 90% when performing visual inspection. However, after around the first half hour, the accuracy of the human operator decreases significantly for only one defect inspection [2].

In the article ”Automated inspection of printed circuit

boards through machine vision" [1], the authors adopt a computer vision approach using a digital image subtraction-elimination system to detect defects in PCBs, with image subtraction techniques being the most widely used methods for years in the field of PCB visual analysis [3].

However, with the increase in computational power and the popularization of machine learning and deep learning techniques, these technologies have become increasingly relevant in the aforementioned context, and various neural network architectures focused on computer vision applications have been developed. One of the candidates for the experiments is the DETR architecture, which has proven to achieve similar effectiveness to other architectures while using less computational power [4].

This work presents a comparative analysis of four different architectures: YOGA, YOLOv11, DETR, and LW-DETR, applied to the problem of defect detection in PCBs. The experiments were conducted with the aim of evaluating the effectiveness of these architectures using metrics such as mAP 50, mAP 50-95, computational complexity (GFLOPS), number of parameters, and training time.

2. Methods

This section describes the approach adopted to train and evaluate different computer vision architectures in the context of defect detection in printed circuit boards (PCBs). The main objective is to conduct a comparative analysis of these architectures, taking into account the required computational power, training time, and the final accuracy of the models.

The methodology is divided into three main stages: data preparation, model training, and comparative evaluation of the architectures. Data preparation involves the collection and preprocessing of the image dataset HRIPCB, which will be used to train the different architectures. Careful selection of the data is crucial to ensure that the models learn effectively and generalize well to new examples.

The computer vision architectures that will be analyzed include:

1. YOGA;
2. YOLOv11;
3. DETR;
4. LW-DETR;

Descriptions of the architectures, including their details and a visual representation, are available in sections 2.2 and 2.3.

After training, each model was evaluated based on performance metrics, including accuracy, training time, and computational resource utilization. The results will be presented in graphs and tables to facilitate comparative analysis. At the end of the work, a recommendation will be made based on the conclusions drawn during the analysis.

2.1 Computer Configuration

The computer used for the training sessions was configured with an Intel Core i9 9900k processor, 64 GB of RAM, an Nvidia GeForce RTX 3090 graphics card with 24 GB of VRAM, and the Ubuntu 20.04.4 LTS operating system.

2.2 Detection Transformer-Based Architectures

The DETR architecture [4] was created with the goal of simplifying the traditional object detection pipeline. The authors eliminate the need for manually designed components, such as non-maximum suppression and anchor generation, by using transformers. By implementing self-attention and a bipartite matching-based loss scheme, DETR makes unique and parallel predictions, reducing redundancies and maintaining simplicity. The architecture was chosen because its source code is readily available on GitHub and because it is the most basic Vision Transformer architecture for object detection, making it a candidate for comparisons with derived architectures.

The DETR architecture consists of three main components: a CNN backbone to extract compact feature representations, a transformer encoder-decoder for sequential processing, and a feed-forward network (FFN) to generate the final detection predictions. The backbone converts images into activation maps, while the encoder reduces dimensions and applies self-attention, using fixed positional encodings. The decoder uses learned object queries to predict objects in parallel, generating bounding box coordinates and classes through a shared FFN. The architecture simplifies the traditional pipeline, eliminating the dependency on manual components and enabling global predictions with the full context of the image. Auxiliary losses are applied to improve training performance and ensure accurate object prediction. A graphical representation of the architecture can be seen in Figure 1.

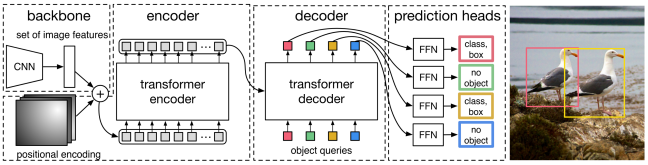


Figure 1. DETR Architecture [4].

LW-DETR [5] is an architecture designed to be lightweight and focuses on real-time object detection, outperforming convolutional network-based methods, such as the YOLO series, on benchmarks like COCO. The architecture combines a simplified ViT encoder, a convolutional projector, and a shallow DETR-based decoder, utilizing recent techniques to improve training, such as IoU-based loss, deformable cross-attention, and encoder-decoder pre-training. For inference efficiency, it adopts alternating window and global attentions, reducing computational complexity. The architecture demonstrated superior performance compared to existing real-time detectors, achieving an mAP of 58.3% in its largest model. The architecture was chosen because its source code is readily available

on GitHub and because it focuses on real-time detection.

2.3 Non-Transformer-Based Architectures

To establish a contrast with Transformer-based architectures, a search was conducted for architectures that do not involve the use of transformers.

The YOGA architecture [6] is an architecture that aims to achieve competitive accuracy with a focus on real-time detection while remaining lightweight enough to be implemented on low-cost field devices, such as the Nvidia Jetson Nano 2GB. The authors claim a reduction of up to 34% in model size while still maintaining competitive accuracy. The architecture was chosen for its focus on low-cost field device applications and real-time detection, as well as for the ease of access to its source code on GitHub.

The design of YOGA consists of a backbone called CSPGhostNet (cross-stage partial GhostNet), a neck named AFF-PANet (attention feature fusion-based path aggregation network), and a head based on YOLO. A graphical representation of the architecture can be seen in Figure 2.

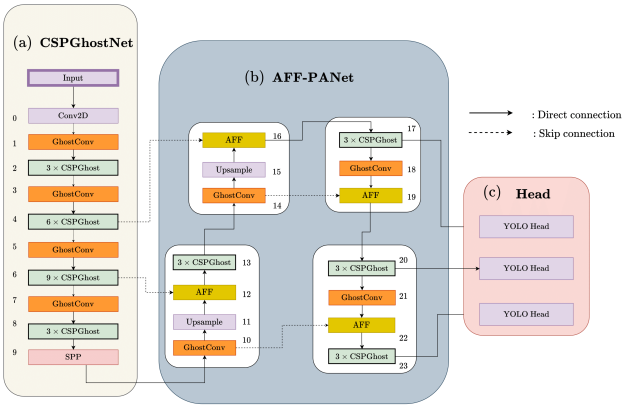


Figure 2. YOGA architecture [6].

As a result, the authors achieved better performance with the YOGA-n (nano) model when compared to YOLOv5n and similar performance with fewer parameters when comparing the Small, Medium, and Large models. The model also proved to be promising for low-cost devices, with the YOGA-n model performing inference on one image every 0.57 seconds.

YOLOv11 is the latest version of the popular YOLO architecture. YOLOv11 introduces several innovations, such as the C3k2 block (Cross Stage Partial with kernel size 2) and the SPPF (Spatial Pyramid Pooling - Fast) and C2PSA (Convolutional block with Parallel Spatial Attention) components [7]. The architecture was chosen for its easy accessibility through a Python library and for being recent, having been released in October 2024.

YOLOv11 retains the same SPPF from recent versions. However, with the addition of C2PSA, the spatial attention of the architecture is improved. As a result, YOLOv11 can better focus on specific areas of interest, significantly enhancing object detection across various sizes and positions.

The C3k2 block was introduced to replace the previous C2f in the neck of the architecture. This block was designed to be faster and more efficient, improving overall performance. It is also implemented in multiple instances in the head of the architecture, where it is employed to process and refine feature maps.

The YOLOv11 model presents significant improvements in both inference speed and accuracy compared to previous architectures such as YOLOv5 and YOLOv10, as shown in benchmark analyses on the COCO dataset. The YOLOv11x model achieves approximately 54.5% mAP50-95 with a latency of 13ms, the highest among all YOLO iterations. The YOLOv11s model performs exceptionally well in low-latency ranges, reaching about 47% mAP50-95 between 2-6ms, making it ideal for real-time applications. Additionally, YOLOv11 utilizes computational resources more efficiently than previous generations. Figure 3 presents a comparison of the YOLO generations.

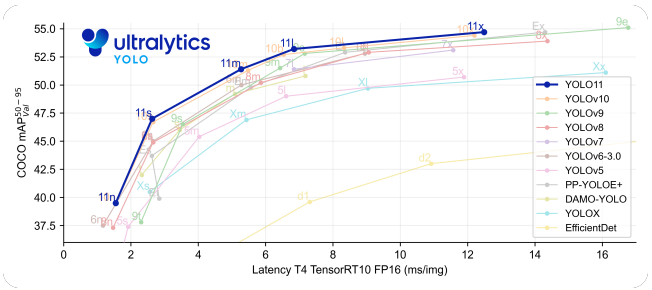


Figure 3. Comparison chart of YOLO architectures [8].

2.4 Dataset

The chosen dataset, HRIPCB, provides 1,386 synthesized images of PCBs without inserted components, containing different types of defects such as mousebites, spur, among others. The dataset was developed to support tasks of defect detection, classification, and registration, making it particularly useful for training the models mentioned in this work. The proposed approach, called RBCNN, employs a reference-based inspection method complemented by an end-to-end convolutional neural network (CNN) for defect classification, demonstrating superior performance compared to conventional methods that rely on pixel-by-pixel processing. The choice of the HRIPCB dataset is justified by its high-quality and detailed images, which are essential for effectively training deep learning models such as convolutional neural networks and vision transformers. These models require diverse and precise data to achieve robust performance in complex computer vision tasks.

The relevance of the HRIPCB dataset is highlighted, for instance, in its use in the paper ”PCB Defect Detection based on Generative Adversarial Network” [9]. In this study, the author presents an approach for detecting PCB defects using Generative Adversarial Networks (GANs) with an enhanced super-resolution architecture called EESRGAN (Edge-Enhanced Super-Resolution GAN), originally applied in the field of

remote sensing. The proposed methodology involves improving PCB images to enable super-resolution detection. This, combined with different preprocessing models, allows for the precise identification of six common defect types in a PCB dataset. The conducted experiments demonstrate the effectiveness of the model, emphasizing that preprocessing through sliding window cropping yields the highest F-measure values, reflecting high accuracy in defect identification.

The study explores three preprocessing methods—scaling, rotation-based cropping, and sliding window cropping—applied to the HRIPCB dataset. The results indicate that combining sliding window cropping with EESRGAN provides the best detection accuracy, with F-measure values exceeding 85%, particularly for missing hole and short-circuit defects, whose F-measures approach 95%. Additionally, this approach reduces the number of training iterations, resulting in shorter training times, making this technique appealing for real-world PCB defect detection applications. An example of one of the dataset images can be seen in Figure 4.

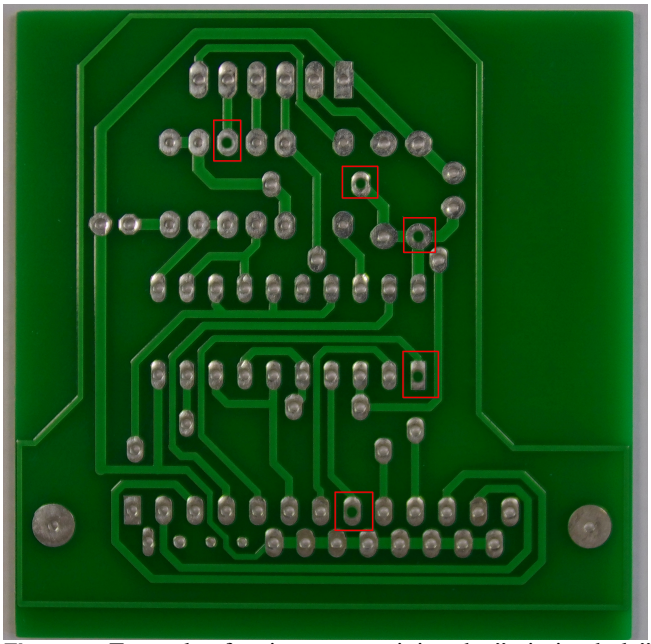


Figure 4. Example of an image containing the ”missing hole” defect [10].

2.5 Evaluation Metrics

To assess the performance of the analyzed architectures, five parameters were selected: Training time (in hours, minutes, and seconds), mAP, number of parameters, computational complexity (measured in GFLOPS), and Frames Per Second (FPS).

Training time represents the total time required to train the architecture. A shorter training time can be beneficial for architecture selection, as a longer training time implies increased energy and infrastructure costs, for example. With higher energy consumption, the carbon footprint also increases, making a shorter training time a relevant criterion for sustainability.

mAP (mean Average Precision) is a metric used to evaluate the accuracy of object detection models in AI, such as in neural networks. For each class, precision (how many detected objects were correct) and recall (how many actual objects were detected) are calculated, and the relationship between precision and recall is plotted for different confidence thresholds. AP (Average Precision) is the area under the PR curve for a single class. Therefore, mAP is the average of the APs across all classes. For mAP 50 and mAP 50-95, an Intersection over Union (IoU) calculation is performed. For mAP 50, the minimum IoU is 50%, meaning there must be a 50% overlap between the predicted bounding box and the ground truth. For mAP 50-95, the minimum IoU is 95%, making it a more stringent metric.

The number of parameters refers to the quantity of adjustable weights in the model. Smaller models (i.e., with fewer parameters) are more efficient and can run on devices with limited memory (such as embedded systems or mobile devices). On the other hand, larger models may be more accurate but consume more resources and are slower. Computational complexity, measured in GFLOPS, represents the number of mathematical operations (in billions—Giga) the model performs per inference. The higher this value, the more operations are required, making the model heavier and more computationally demanding.

FPS, or frames per second, measures how many images the model processes per second on specific hardware. A high FPS (30 or greater) is ideal for real-time applications (video, robotics, automation), while a low FPS (10 or lower) may be impractical for tasks requiring quick responses.

These metrics are important for measuring the trade-off between accuracy and efficiency: A model may have a high mAP, but if it is slow (low FPS) or heavy (high GFLOPS), it may be impractical in real-world scenarios. Depending on the application, these metrics may have varying degrees of impact on the decision-making process. In a real-time computer vision system (e.g., conveyor belts in a factory), high FPS and low GFLOPS should be prioritized. In more static environments, such as medical image analysis, FPS may be less critical, but a higher mAP is required. It is also worth considering that high computational costs—associated with models with many parameters and high GFLOPS—will demand more expensive GPUs, increasing implementation costs.

3. Results and Discussion

In this section, the results obtained from the experiments conducted in the context of this work are presented and analyzed. The results include performance metrics, comparisons between the different proposed methods, and discussions on their impacts on the decisions that supported the study’s conclusions.

The experiments were conducted to evaluate the effectiveness of four computer vision architectures in the PCB inspection problem, using metrics such as accuracy, training time, and model complexity, expressed in terms of the number of parameters and the number of floating-point operations per second (FLOPS).

Each architecture was analyzed in its respective section, and finally, a comparative analysis of the results was conducted.

3.1 YOGA

The YOGA architecture presented varied results across its different models, reflecting a balance between training time, computational complexity, and accuracy. The YOGA-s model stood out as the most balanced option, with a training time of 50 minutes and 13 seconds, achieving the best accuracy in mAP 50 (83.2%) and an mAP 50-95 of 36.2%. These results demonstrate a good trade-off between performance and efficiency, making it an interesting choice for scenarios where computational complexity is moderate.

In contrast, the YOGA-n model had the shortest training time (32 minutes and 34 seconds) and the lowest computational complexity, with only 1.83 million parameters and 4.4 GFLOPS, but obtained a lower mAP 50-95 accuracy (33.0%). The YOGA-m model, while consuming more training time (1 hour and 19 minutes), achieved the highest mAP 50-95 score (37.0%), being recommended for applications where an increase in accuracy justifies the higher computational cost. In contrast, the YOGA-l model showed the worst accuracy performance (mAP 50 of 66.3% and mAP 50-95 of 30.9%), despite having the longest training time and highest complexity, proving to be non-competitive compared to the others. The previously mentioned data can be observed in Table 1.

3.2 YOLOv11

The experiments with the YOLOv11 architecture were conducted following the methodological protocol established in the previous chapter, aiming to evaluate the performance of the architecture in the PCB inspection problem. These results can be observed in Table 2.

Table 2 represents the training results on the HRIPCB dataset for various YOLOv11 model sizes. The initial experiments with 100 epochs showed unsatisfactory accuracy. Therefore, the number of epochs was increased in the training sessions. We can observe that, in general, the model's performance (mAP 50 and mAP 50-95) improves significantly with an increase in the number of epochs, although this results in a considerable increase in training time. For example, the YOLO11n model, which has the shortest training time, achieved 69.2% mAP 50 and 32.7% mAP 50-95 in 100 epochs, whereas with 500 epochs, it reached 85.8% and 42.7%, respectively, with a total training time of 40 minutes and 14 seconds.

When comparing models of different sizes, we notice an advantage toward smaller models, such as YOLO11n and YOLO11s, which achieved the best performances in terms of mAP 50 and mAP 50-95 when trained for 500 epochs, with 85.8% and 85.7%, and 42.7% and 43.6%, respectively. On the other hand, larger models such as YOLO11l and YOLO11x showed lower performance results even after 500 epochs. This may indicate that smaller models are more suitable for PCB scenarios, whereas larger models, despite being more robust, may not fully leverage the increase in the number of epochs relative to task complexity and the dataset used, tending to overfit.

We can also observe the training process evolution graphs for the "n" model in Figures 5 and 6. The other graphs are available in the GitHub repository [1].

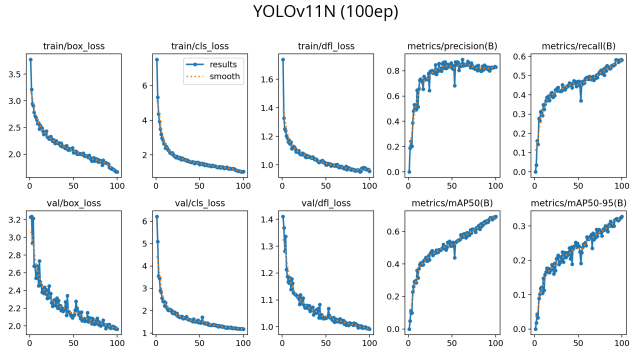


Figure 5. Training process evolution of YOLO11n (100 epochs).

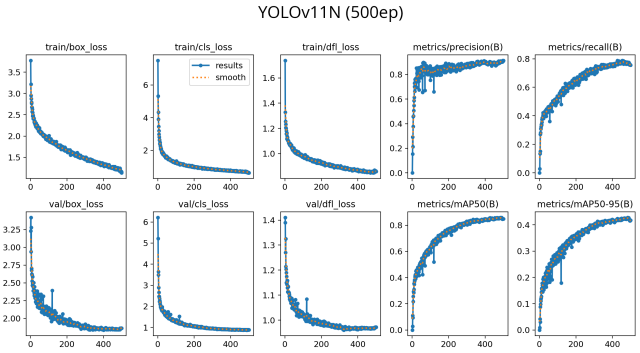


Figure 6. Training process evolution of YOLO11n (500 epochs).

Based on the presented data, the YOLO11n model trained for 500 epochs emerges as a strong candidate for the PCB inspection task considering its achieved performance. With a training time of 40 minutes and 12 seconds, a complexity of 6.3 GFLOPS, and only 1.1% less accuracy in mAP 50-95, the "n" (nano) model proves to be a strong contender for PCB inspection tasks, especially in applications that require a good cost-benefit ratio between accuracy and computational complexity of the model.

[1] https://github.com/46-neko/master-thesis

Model	Training Time	mAP 50	mAP 50-95	Params (M)	GFLOPS	FPS
YOGA-n	32m 34s	76.6	33.0	1.83	4.4	103.10
YOGA-s	50m 13s	83.2	36.2	7.33	15.6	99.01
YOGA-m	01h 19m 26s	82.2	37.0	15.94	33.2	81.96
YOGA-l	02h 11m 06s	66.3	30.9	33.09	69.8	68.96

Table 1. Training results table for the YOGA architecture.

Model	Epochs	Training Time	mAP 50	mAP 50-95	Params (M)	GFLOPS	FPS
YOLO11n	100	08m 04s	69.2	32.7	2.5	6.3	171.03
YOLO11n	500	40m 14s	85.8	42.7	2.5	6.3	177.03
YOLO11s	100	12m 15s	62.8	30.6	9.4	21.3	173.37
YOLO11s	500	59m 55s	85.7	43.6	9.4	21.3	171.65
YOLO11m	100	22m 46s	63.8	30.9	20.0	67.7	135.05
YOLO11m	500	01h 54m 33s	81.6	41.6	20.0	67.7	130.83
YOLO11l	100	31m 40s	64.0	31.9	25.28	86.6	84.89
YOLO11l	500	02h 27m 16s	80.1	41.4	25.28	86.6	84.20
YOLO11x	100	54m 39s	64.5	31.5	56.8	194.4	67.20
YOLO11x	500	04h 12m 13s	78.9	40.6	56.8	194.4	66.91

Table 2. Table of training results for the YOLOv11 architecture.

3.3 DETR

The DETR architecture was, among all the architectures in the experiment, the one that presented the longest training time. The shortest time was 4 hours and 42 minutes, and the longest was 13 hours and 7 minutes. However, DETR proved to be very efficient in the task of defect detection in PCBs, achieving the best result among all architectures, with 92.4% in mAP 50 and 44.6% in mAP 50-95.

The models with the r101 backbone were the ones that achieved the best results on the HRIPCB dataset. However, they were also the models with the longest training times and the highest computational complexity.

Table 3 shows the results obtained with different variations of the ResNet architecture, considering training time, performance (mAP 50 and mAP 50-95), and model configurations. The ResNet 101 model, without DC5 (dilated C5 stage), stood out by achieving a mAP 50 of 92.2% and a mAP 50-95 of 41.6% in a training time of 5 hours and 46 minutes with a complexity of approximately 107 GFLOPS, demonstrating a balance between performance and computational cost. On the contrary, the ResNet 50 version showed lower performance, with a mAP 50 of 90.6% and a mAP 50-95 of 39.2%, but with a shorter training time, finishing in 4 hours and 42 minutes and a relatively lower computational complexity, with just under 60 GFLOPS. With a difference of 1.6% in mAP 50 and 2.4% in mAP 50-95, while presenting a difference of just over 47 GFLOPS, the ResNet 50 backbone architecture proves to be a strong candidate for PCB defect detection, especially in low computational power environments.

By incorporating the DC5 modification, the models showed significant changes. ResNet 101 + DC5 achieved the best performance among all configurations, with a mAP 50 of 92.4% and a mAP 50-95 of 44.6%, but with a much higher training time, exceeding 13 hours, and a computational complexity above 160 GFLOPS. On the other hand, ResNet 50 + DC5 showed a drop in mAP 50 to 89.2%, but a slight improvement in mAP 50-95 (39.9%) compared to the version without DC5, at the cost of more than doubling the training time and nearly doubling the complexity. These results suggest that using DC5 may be advantageous in scenarios where accuracy is prioritized, although it demands higher computational power.

3.4 LW-DETR

The performance analysis of different variations of the LW-DETR architecture is crucial to evaluate both the efficiency and effectiveness of the model in detection tasks. In this section, we present the training results, considering accuracy metrics such as mAP 50 and mAP 50-95, the time required for training, and the computational complexity of the models, including the number of floating-point operations (FLOPS). Table 4 summarizes the main findings, which will be detailed below.

The results demonstrate the varied performance of the LW-DETR architecture configurations in terms of training time, accuracy, and computational complexity. The LW-DETR Medium model achieved the best balance between computational cost and performance, reaching a mAP 50 of 60.4% and a mAP 50-95 of 24.0%. This performance was achieved with a training time of 2 hours and 40 minutes and a complexity of 42.79 GFLOPS, proving efficient both in accuracy and computational resource consumption.

Yet, the LW-DETR Tiny model, despite having the shortest training time (1 hour and 17 minutes) and low computational complexity (11.24 GFLOPS), showed inferior performance, with a mAP 50 of 47.3% and a mAP 50-95 of 19.6%. This configuration may be suitable for scenarios where computational resources are limited, and accuracy is not the main priority.

Model	Training Time	mAP 50	mAP 50-95	Params (M)	GFLOPS	FPS
DETR ResNet 50	04h 42m 43s	90.6	39.2	41.3	59.56	41.12
DETR ResNet 101	05h 46m 20s	92.2	41.6	60.2	106.9	32.26
DETR ResNet 50 + DC5	09h 11m 08s	89.2	39.9	41.3	113.11	32.82
DETR ResNet 101 + DC5	13h 07m 02s	92.4	44.6	60.2	160.46	26.23

Table 3. Table of training results for the DETR architecture.

Interestingly, the LW-DETR X-Large model, with the highest number of parameters (118.03 million) and the highest computational complexity (174.20 GFLOPS), did not show performance proportional to its scale. Despite consuming 4 hours and 39 minutes for training, the mAP 50 was 51.7% and the mAP 50-95 was 21.9%, falling behind the LW-DETR Medium. This result reinforces that increasing scale and complexity does not always guarantee significant improvements in accuracy, highlighting the importance of considering the cost-benefit of each model.

3.5 Comparative Analysis

After the experiments, the models with the best cost-benefit ratio in terms of training time, accuracy, number of parameters, and computational complexity were selected. This selection is best visualized in Table 5.

The YOGA-s architecture demonstrated a reduced training time, with only 50 minutes and 13 seconds, and achieved the second-best mAP 50 accuracy, reaching 83.2%, reflecting a good balance between precision and efficiency. However, YOGA-s has more than double the complexity compared to YOLOv11n and obtained an inferior mAP 50-95 score compared to other models, which limits its competitiveness in terms of broader accuracy.

Despite this, YOGA-s remains competitive when compared to transformer-based models due to its low computational complexity and short training time. However, in direct comparison with YOLOv11n, which achieves higher accuracy (mAP 50-95 of 42.7) and lower computational complexity, YOGA-s appears as a second option. Still, the YOGA-s architecture stands out as a balanced choice for scenarios that demand a trade-off between accuracy and computational efficiency.

The YOLOv11 models stood out for their high accuracy and low training time, while also maintaining low computational complexity. Although YOLOv11 did not achieve the best mAP 50 score, its mAP 50-95 score surpassed the others and it was by far the model with the lowest computational complexity. This combination of high performance and efficiency makes YOLOv11 an excellent choice for PCB defect inspection, especially in edge environments, such as embedded devices or systems with limited computational resources. In Figure 7, we can se some images used for validation with their respective labels (Figure 7a), and the inference from the YOLO11n trained for 500 epochs are shown in Figure 7b.

The DETR architecture demonstrated great robustness in its detection capabilities. Despite having the longest training time among the evaluated architectures, just under 5 hours, it achieved the best mAP 50 performance, surpassing YOLOv11 by 4.8%. However, it is also the most complex and heavy architecture, with 41.3 million parameters and almost 60 GFLOPS. In scenarios where computational power is not a limitation, DETR stands out as the best choice due to its high accuracy.

Alternatively, LW-DETR, although designed for applications with low computational resources, proved to be less efficient compared to the other architectures. With a mAP 50 just above 60% and a mAP 50-95 below 24%, LW-DETR performed below expectations. Furthermore, its complexity, higher than that of more accurate models, undermines its appeal, offering no clear advantages within the context of the experiment.

Among the models analyzed, YOLOv11n proved to be, in general, the most suitable option for the task of PCB inspection, considering a combination of factors such as accuracy, computational complexity, and training time. With the smallest number of parameters (2.5M) and GFLOPS (6.3), along with a short training time of 40 minutes and 14 seconds, the model achieved the highest mAP 50-95 score (42.7) while still maintaining a competitive mAP 50 score (85.8), demonstrating efficiency while keeping size and complexity reduced. This superior performance, combined with low computational complexity, positions YOLOv11n as the best choice for embedded device applications or scenarios with limited computational resources.

On the other hand, for scenarios where the priority is to maximize accuracy and computational complexity is not a restrictive factor, the DETR model with a ResNet 50 backbone stands out. With the highest mAP 50 (90.6) and competitive mAP 50-95 performance (39.2), it offers the best detection capability among the evaluated models, although with a significant cost in terms of training time and computational resources. Thus, the final recommendation depends on the application context: YOLOv11n is ideal for edge scenarios with low computational power, while DETR is more suitable for environments with greater computational power available and a greater focus on accuracy.

3.6 Conclusion

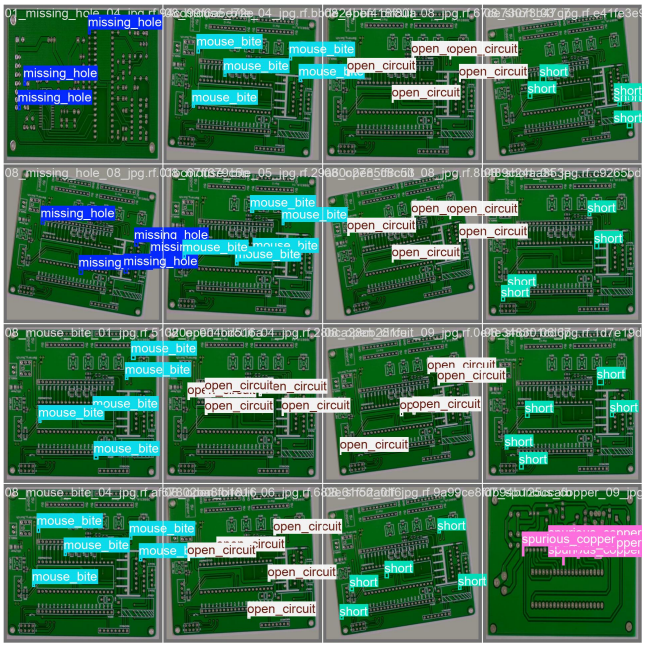
The primary objective of this work was to conduct a comparative study of different computer vision architectures for PCB inspection using the HRIPCB dataset. During development, the performance of the YOGA, YOLOv11, DETR, and LW-DETR architectures was evaluated across various

Model	Training Time	mAP 50	mAP 50-95	Params (M)	GFLOPS	FPS
LW-DETR Tiny	01h 17m 01s	47.3	19.6	1.2	11.24	110.78
LW-DETR Small	02h 26m 24s	44.4	17.6	14.55	16.65	89.08
LW-DETR Medium	02h 40m 44s	60.4	24.0	28.23	42.79	70.08
LW-DETR Large	03h 29m 30s	48.4	18.6	46.82	71.55	55.28
LW-DETR X-Large	04h 39m 33s	51.7	21.9	118.03	174.20	31.98

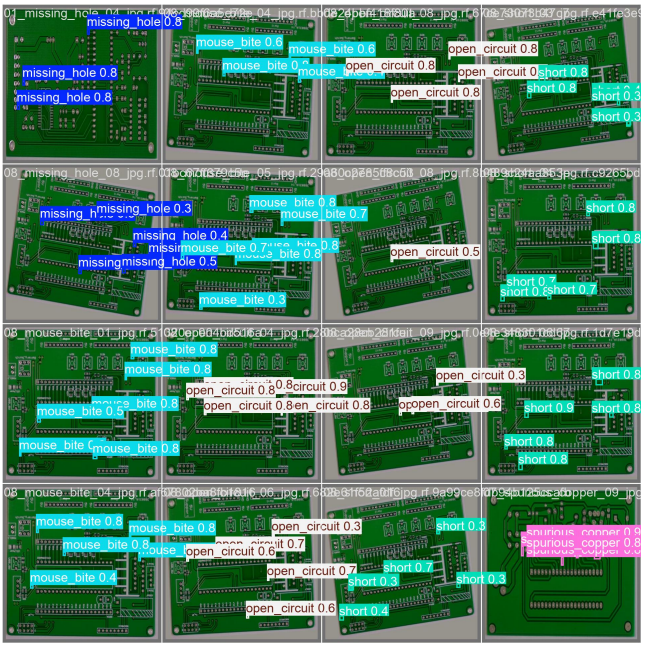
Table 4. Table of training results for the LW-DETR architecture.

Model	Training Time	mAP 50	mAP 50-95	Params (M)	GFLOPS	FPS
YOGA-s	50m 13s	83.2	36.2	7.33	15.6	99.01
YOLO11n (500 eps)	40m 14s	85.8	42.7	2.5	6.3	177.03
DETR ResNet 50	04h 42m 43s	90.6	39.2	41.3	59.56	41.12
LW-DETR Medium	02h 40m 44s	60.4	24.0	28.23	42.79	70.08

Table 5. Table presenting the selected models for the comparative analysis.



(a) Validation images used for prediction with the YOLO11n (500 eps) model with their respective labels.



(b) Example of inference from the YOLO11n (500 eps).

Figure 7. Comparison of labeled validation images and model predictions from YOLO11n trained for 500 epochs.

aspects, including total training time, computational resource consumption, and defect detection accuracy.

The results demonstrated that the evaluated architectures exhibit complementary characteristics, with some being more suitable for scenarios requiring high accuracy, while others stand out for computational efficiency. For instance, the YOLOv11 architecture achieved strong accuracy values, particularly in mAP 50-95, while maintaining low computational cost, high inference speed, and extremely short training time. On the other hand, the base DETR architecture proved advantageous in terms of robustness, delivering the best results when using the ResNet 101 backbone combined with DC5. These findings highlight the importance of considering the specific application context when selecting a computer vision architecture.

The YOGA architecture also achieved strong results in terms of training time, with mAP 50 exceeding 80% and mAP 50-95 surpassing 36%, along with a high inference speed of nearly 100 FPS, while maintaining low complexity and a small model size (7.33 million parameters and 15.6 GFLOPS). However, no clear advantages over YOLOv11 were observed. Meanwhile, LW-DETR emerged as the weakest candidate for PCB inspection, delivering the worst results in both mAP 50 and mAP 50-95, and ranking as the second most complex and resource-intensive architecture—trailing only DETR—despite its decent FPS performance.

A significant contribution of this work is addressing gaps identified in the literature regarding the application of modern models (such as YOLOv11, released in October 2024) in automated industrial inspections.

However, some limitations were identified. Training was conducted using limited computational resources, which restricted the ability to explore more complex architecture configurations or conduct experiments with even larger datasets.

In conclusion, this study represents a significant advancement in the application of artificial intelligence for PCB inspection, offering contributions to both academia and industry. The results not only demonstrate the feasibility of the proposed solutions but also provide valuable insights for selecting architectures in practical applications.

3.7 Future Works

Based on the results obtained, some directions for future work are suggested. Additional research could explore the use of model fusion techniques, aiming to combine the advantages of different architectures. For example, we could experiment with replacing the YOLOv5 head in the YOGA architecture with a head based on YOLOv11, the most recent iteration (as of the publication of this work) of the YOLO architecture. Additionally, the development of data augmentation methods specifically tailored for PCB defects could further enhance model performance.

It is also recommended to investigate the application of the studied architectures in other industrial contexts, broadening the impact of this work. Furthermore, the analysis focused solely on quantitative results, and future studies could explore qualitative aspects of defect detection as well.

Other datasets could also be used in experiments, along with testing the application in low computational power environments, such as embedded systems, which are widely used in the industry.

Author contributions

Pedro Luiz Henriques Benedetti: Training, dataset and architectures preparation and debugging, analysis of the results, reporting results; Marcelo Ricardo Stemmer: Academic advising, supervision.

References

[1] WU, W.-Y.; WANG, M.-J. J.; LIU, C.-M. Automated inspection of printed circuit boards through machine vision. *Computers in Industry*, v. 28, n. 2, p. 103–111, 1996. ISSN 0166-3615. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/0166361595000631⟩.

[2] SILVA, C. et al. The visual inspection of solder balls in semiconductor encapsulation. In: . [S.l.: s.n.], 2022. p. 750–757.

[3] JIN, J. et al. Defect detection of printed circuit boards using efficientdet. In: *2021 IEEE 6th International Conference on Signal and Image Processing (ICSIP)*. [S.l.: s.n.], 2021. p. 287–293.

[4] CARION, N. et al. End-to-end object detection with transformers. In: SPRINGER. *European conference on computer vision*. [S.l.], 2020. p. 213–229.

[5] CHEN, Q. et al. Lw-detr: A transformer replacement to yolo for real-time detection. *arXiv preprint arXiv:2406.03459*, 2024.

[6] SUNKARA, R.; LUO, T. YOGA: Deep object detection in the wild with lightweight feature learning and multiscale attention. *Pattern Recognition*, Elsevier, v. 139, p. 109451, 2023.

[7] KHANAM, R.; HUSSAIN, M. *YOLOv11: An Overview of the Key Architectural Enhancements*. 2024. Disponível em: ⟨https://arxiv.org/abs/2410.17725⟩.

[8] YT7589. *YOLOv11*. [S.l.]: GitHub, 2024. ⟨https://github.com/yt7589/yolov11⟩.

[9] YOU, S. Pcb defect detection based on generative adversarial network. In: *2022 2nd International Conference on Consumer Electronics and Computer Engineering (ICCECE)*. [S.l.: s.n.], 2022. p. 557–560.

[10] HUANG, W. et al. Hripcb: a challenging dataset for pcb defects detection and classification. *The Journal of Engineering*, v. 2020, 05 2020.