International Journal of Combinatorial Optimization Problems and Informatics, 17(2), 2026, 38-54. ISSN: 2007-1558.
https://doi.org/10.61467/2007.1558.2026.v17i2.1266

www.editada.org

Fine tuning Transformers models for converting handwritten scientific texts into LaTeX format

Rodrigo Alvarez-Perez[1], *Ricardo Barrón-Fernandez*[1]
[1] Laboratorio de Inteligencia Artificial, CIC IPN
ralvarezp2025@cic.ipn.mx, rbarron@cic.ipn.mx

Abstract. This study introduces a methodology for optical character recognition (OCR) that leverages transformer-based architectures to enhance the detection and recognition of textual content within images. The approach integrates state-of-the-art models, employing DETR (Detection Transformer) for the generation of bounding boxes corresponding to candidate text sequences, and TrOCR for transcribing the text contained within these regions. Both models were fine-tuned on a proprietary dataset comprising handwritten and digitized notes from mathematics-related subjects, including differential equations, calculus, linear algebra, programming, etc. The dataset predominantly consists of mathematical expressions represented in LaTeX format, thereby allowing the proposed method to effectively address the recognition of complex symbolic content in mathematical texts.

Keywords: OCR, Text detection, Text recognition, DETR, TrOCR, Transformers, LaTeX.

Article Info
Received December 2, 2025
Accepted Jan11 2026

1 Introduction

Optical Character Recognition (OCR) (Chaudhuri et al., 2016) is one of the most widely studied yet challenging tasks in the field of computer vision. It typically involves two subtasks: text detection (i.e., locating textual regions) and character recognition. While these processes are intuitive for humans, they remain difficult for computers, which lack the ability to perceive images holistically. Instead, computational models must rely on processing pixel values and derived features such as color, position, and orientation, which can be mathematically represented as matrices of numerical values.

To address these challenges, a variety of mathematical and statistical algorithms have been proposed, including projection profiling (Javed et al., 2013), feature extraction and classification techniques (Mutlag et al., 2020), and hidden Markov models (Mor et al., 2020). Although effective to some extent, these methods were eventually surpassed by artificial intelligence approaches. Deep learning models such as recurrent neural networks (RNNs) (Mienye et al., 2024) and convolutional neural networks (CNNs) (Li et al., 2022) significantly advanced OCR performance by integrating feature learning into end-to-end frameworks. Despite their success, these architectures were later outperformed by Transformer-based models, first introduced by Google in 2017 (Vaswani et al., 2017). This architecture, through its self-attention mechanism, enables efficient sequence modeling and the extraction of richer contextual features, thereby achieving superior results compared to RNNs and CNNs.

More recently, transformer-based architectures have demonstrated state-of-the-art performance across a variety of vision tasks. For example, the DETR (Carion et al., 2020) model focuses on general object detection, including people, vehicles, facial expressions, traffic signs, and animals, while TrOCR (Li et al., 2021) has been developed specifically for printed and scene text recognition, with applications in domains such as signage and advertisements (Raisi, 2021). A limitation of these models, however, is that most are pretrained predominantly on English-language datasets. As a result, their performance often degrades when applied to text in other languages, leading to inadequate information extraction or unintended translation into the training language.

In this study, we assess the performance of state-of-the-art transformer-based models in specialized OCR tasks. We fine-tuned models from the DETR family on a custom, hand-labeled dataset and compared their performance to a YOLO-family models trained under identical conditions, to evaluate the effectiveness of transformer architectures for detecting words, mathematical expressions, and drawings in digitized handwritten notes. Additionally, we fine-tuned models from the TrOCR family and Nougat-based architectures for handwritten, script, and cursive text recognition, emphasizing the unique challenges associated with these tasks.

2 State of the art

In this study, we conducted a review of contemporary models for text detection and recognition. Based on this assessment, we selected a set of representative and recently proposed object detection architectures, namely DETR, RT-DETR, Deformable-DETR, YOLOv8, and YOLOv11. For the text extraction and recognition component, we employed models from the TrOCR and Nougat families. These models were chosen due to the availability of multiple pre-trained and task-specific variants, as well as their capacity to accommodate broad and diverse vocabularies.

2.1 DETR

The DEtection TRansformer (DETR) introduces a new paradigm in object detection by reformulating the task as a direct set prediction problem. This formulation eliminates the need for traditional handcrafted components such as anchor boxes, region proposals, and non-maximum suppression, thereby simplifying the detection pipeline. DETR combines a convolutional backbone with a Transformer-based encoder–decoder architecture, enabling fully end-to-end training and inference within a unified framework. The model employs a convolutional neural network (CNN), typically ResNet-50 or ResNet-101, to extract a low-resolution feature representation from the input image. These features are subsequently flattened and linearly projected into an embedding space before being processed by the Transformer encoder. The encoder leverages multi-head self-attention mechanisms to capture global contextual relationships (see Figure **1**) among all regions of the image, facilitating non-local reasoning that enhances object differentiation and localization.

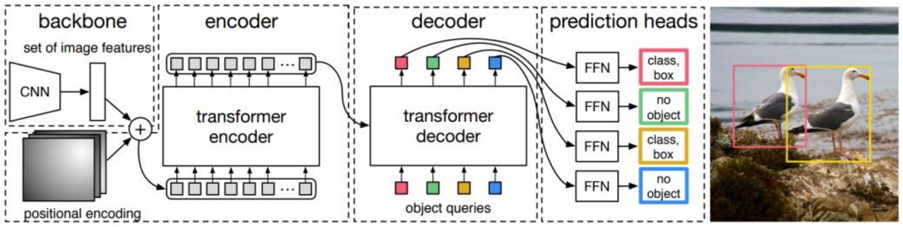


Figure 1. DETR model architecture

The Transformer decoder operates on a fixed set of learned embeddings, known as object queries, each representing a potential object in the image. Through a combination of self-attention and encoder–decoder attention, the decoder iteratively refines these queries by integrating information about inter-object dependencies and global scene context. Each decoder output is then passed through a shared feed-forward prediction network, which estimates the object's class label and corresponding bounding box coordinates, normalized relative to the image dimensions. Predictions that do not correspond to any ground truth instance are assigned to a dedicated "no object" class, ensuring a consistent and comprehensive set of outputs.

2.2 RT-DETR

The Real-Time Detection Transformer (RT-DETR) is a Transformer-based end-to-end object detector capable of real-time performance. Its architecture comprises a ResNet backbone, a hybrid encoder, and a transformer decoder. The backbone extracts hierarchical features, from which the last three stages are passed to the encoder. To overcome the computational bottlenecks of the traditional DETR model architecture, RT-DETR introduces an efficient hybrid encoder that decouples intra-scale and cross-scale processing (see Figure **2**). Specifically, the Attention-based Intra-scale Feature Interaction (AIFI) applies self-attention only to high-level semantic features, while the CNN-based Cross-scale Feature Fusion (CCFF) integrates multi-scale information through convolutional fusion blocks, thereby reducing latency while preserving accuracy. It also introduces an uncertainty-minimal query selection strategy, which improves conventional query initialization methods that relied solely on classification scores. The model simultaneously integrates classification and localization confidence, thereby minimizing epistemic uncertainty and supplying the decoder with higher-quality queries. This leads to improved convergence and enhanced detection accuracy. Subsequently, the decoder iteratively refines these queries to predict class labels and bounding boxes, while auxiliary prediction heads contribute to training stability by regularizing the model's weights.

As reported in the article by the authors, the experimental results underscore the effectiveness of the proposed architecture. On the COCO dataset, RT-DETR-R50 achieved 53.1% AP at 108 FPS, while RT-DETR-R101 reached 54.3% AP at 74 FPS, surpassing YOLOv7 and YOLOv8 models of comparable scale. Furthermore, when compared to DINO-Deformable-DETR-R50 (Zhang et al., 2022), RT-DETR obtained a 2.2% improvement in AP. These findings demonstrate that RT-DETR successfully addresses the limitations of previous DETR variants and establishes a new state-of-the-art in real-time object detection.

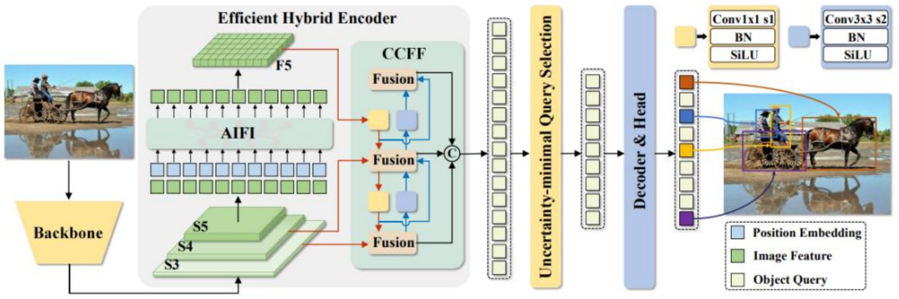


Figure 2. RT-DETR model architecture

2.3 Deformable-DETR

Deformable DETR (Zhu et al., 2021) introduces an improved end-to-end object detection framework that addresses the primary limitations of the original DETR model, namely its slow convergence and reduced accuracy in detecting small objects. The model enhances the Transformer architecture through deformable attention modules, which constrain attention to a sparse set of informative sampling points around reference locations. This design substantially reduces computational complexity, accelerates convergence, and improves detection accuracy while maintaining the Transformer's capability to model global contextual relationships.

The architecture (Figure **3**) preserves the encoder–decoder structure of DETR, employing a convolutional backbone to extract multi-scale feature maps from the input image. These hierarchical features are processed by a deformable Transformer encoder, where each deformable attention module aggregates information from a small number of spatially adaptive sampling points rather than all positions in the feature map. By concentrating on the most relevant regions, the model achieves linear computational complexity with respect to image resolution, allowing efficient processing of high-resolution inputs.

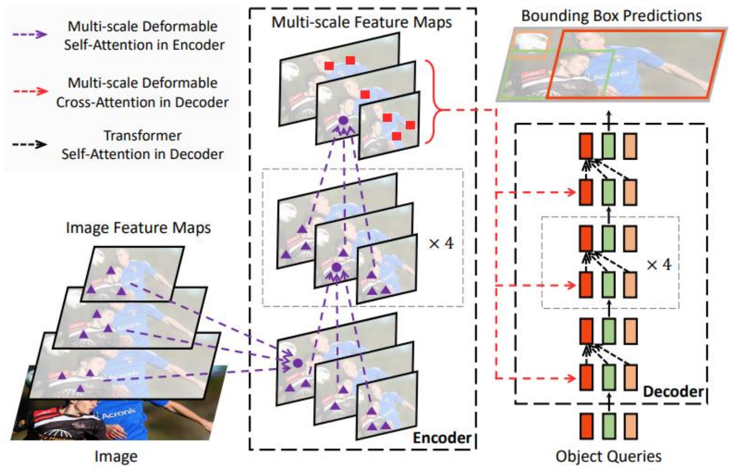

Multi-scale Deformable Self-Attention in Encoder
Multi-scale Deformable Cross-Attention in Decoder
Transformer Self-Attention in Decoder
Multi-scale Feature Maps
Bounding Box Predictions
Image Feature Maps
× 4
× 4
Encoder
Decoder
Image
Object Queries

Figure 3. Deformable DETR model architecture

The encoder generates multi-scale contextual representations that are subsequently refined by the Transformer decoder, which incorporates both self-attention and cross-attention mechanisms. The cross-attention modules utilize deformable attention to integrate image features from the encoder, while the self-attention modules capture dependencies among object queries. Each query corresponds to a potential object, and successive decoder layers progressively refine these representations to predict object categories and bounding box coordinates relative to their reference points.

2.4 YOLOv8

The architecture of YOLOv8 (Yaseen, 2024) adheres to the classical backbone–neck–head structure, incorporating several design optimizations (see Figure **4**) that enhance both efficiency and accuracy. The backbone, built upon an advanced Cross Stage Partial Network (CSPNet), effectively extracts hierarchical features by minimizing computational redundancy and improving gradient propagation. This design enables the model to capture both low-level spatial details and high-level semantic information, which are crucial for precise object detection.

The model integrates a refined combination of the Path Aggregation Network (PANet) and the Feature Pyramid Network (FPN), facilitating efficient multi-scale feature fusion. This configuration strengthens the model's capability to detect objects of varying sizes, particularly small or densely distributed instances. The detection head adopts an anchor-free paradigm, replacing the predefined anchor boxes used in earlier YOLO (Redmon et al., 2015) versions. By directly predicting object centers, dimensions, and classes, this design simplifies training, reduces hyperparameter dependency, and enhances adaptability to diverse object shapes and aspect ratios.

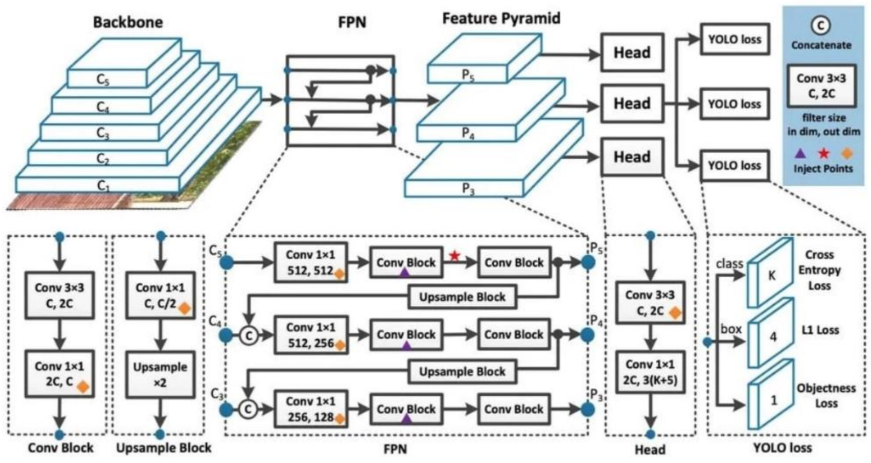


Figure 4. YOLOv8 model architecture

Additional architectural improvements include optimized feature aggregation layers, which enhance computational efficiency and reduce memory consumption, achieving an effective balance between detection speed and accuracy across different hardware configurations.

2.5 YOLOv11

YOLOv11 (Khanam & Hussain, 2024) is the most recent version of the YOLO family of object detectors and introduces architectural refinements that enhance efficiency, accuracy, and versatility across a wide range of computer vision tasks. The model retains the classical backbone–neck–head pipeline but incorporates innovations that distinguish it from its predecessors. In the backbone, feature extraction is improved through the C3k2 block, a lightweight variant of the CSP bottleneck that replaces large kernels with smaller convolutions, thereby reducing computational cost while preserving representational capacity. This stage also integrates the SPPF (Spatial Pyramid Pooling – Fast) module, which accelerates multi-scale feature aggregation, and the C2PSA (Cross Stage Partial with Parallel Spatial Attention) block, which enhances spatial awareness by directing attention toward salient regions, improving the detection of small or occluded objects. The model's architecture is shown in Figure **5**.

Within the neck, multi-scale features are fused using C3k2 blocks, improving computational efficiency while maintaining accuracy, while the inclusion of C2PSA attention further strengthens the model's ability to focus on relevant regions. The head refines predictions through stacked C3k2 modules that adapt flexibly between lightweight and deeper feature extraction, supported by CBS (Convolution–BatchNorm–SiLU) layers that stabilize training. The final detection layers produce bounding boxes, class probabilities, and objectness scores.

These architectural enhancements collectively provide several key improvements. The C3k2 block reduces the number of parameters and accelerates inference, while the C2PSA mechanism improves robustness in complex detection scenarios. Furthermore, YOLOv11 extends beyond conventional object detection to support additional tasks such as instance segmentation, pose estimation, oriented bounding box detection, classification, and tracking, consolidating its role as a versatile and multi-purpose vision model.

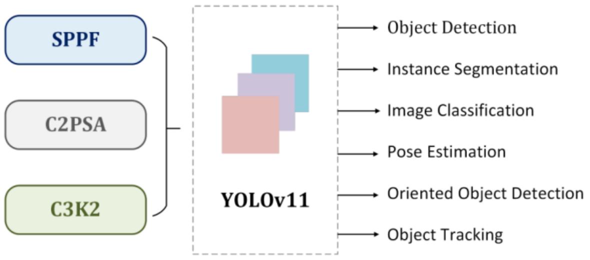


Figure 5. YOLOv11 model architecture

2.6 TrOCR

TrOCR is an end-to-end optical character recognition (OCR) model that departs from conventional convolutional and recurrent architectures by adopting a purely Transformer-based encoder–decoder design. The encoder employs a Vision Transformer that resizes the input image, partitions it into fixed-size patches, and projects them into embedding vectors, which are then processed through self-attention mechanisms without relying on convolutional inductive biases. The decoder is a standard Transformer that autoregressively generates subword tokens, integrating the encoder's visual features with contextual information from previously decoded outputs (see Figure **6**). This design eliminates the need for external language models and character-level decoding, thereby simplifying the transcription pipeline.

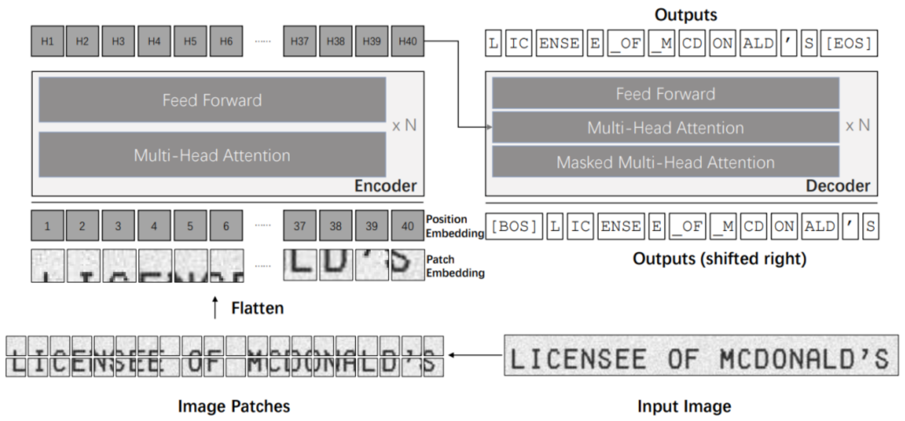


Figure 6. TrOCR architecture

A central innovation of TrOCR is its initialization strategy: the encoder is initialized with pre-trained vision models such as DeiT or BEiT, while the decoder is initialized with pre-trained language models such as RoBERTa or MiniLM. This multimodal initialization leverages large-scale pre-training in both vision and language domains, providing robust priors for visual representation and linguistic modeling. Training was conducted in two stages, beginning with large-scale pre-training on synthetic text-line images, followed by task-specific pre-training for printed, handwritten, or scene text.

2.7 Nougat

Nougat (Blecher et al., 2023) presents a transformer-based architecture developed to convert document images, particularly scientific PDFs, into machine-readable markup text. Built upon the Donut framework (Kim et al., 2022), it employs a fully end-to-end encoder–decoder design that removes the need for external OCR systems or embedded text extraction, enabling direct processing of both digital-born and scanned documents from rasterized images.

The architecture (Figure **7**) consists of two main components: a visual encoder and a text decoder. The encoder is based on the Swin Transformer, a hierarchical vision transformer that divides the input image into non-overlapping windows and applies multi-head self-attention both within and across these regions to capture local and global visual dependencies. This process yields a sequence of latent embeddings representing the document's visual structure. The decoder, adapted from the mBART architecture, autoregressively generates structured markup text by employing self-attention to preserve contextual coherence and cross-attention to align textual predictions with corresponding visual features from the encoder output. The final output consists of tokenized markup sequences encompassing plain text, mathematical expressions, and tabular elements.

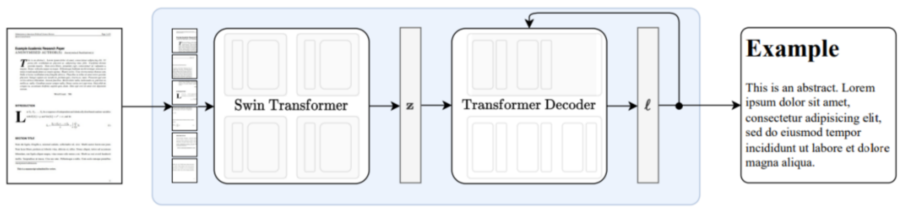

Swin Transformer
z
Transformer Decoder
ℓ
Example
This is an abstract. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Figure 7. Nougat model architecture

Nougat introduces several key advancements over traditional OCR-based systems. By eliminating handcrafted OCR components, it enables end-to-end learning of document semantics, capturing spatial and structural relationships essential for accurately representing mathematical notation.

3 Methodology

3.1 Dataset creation

Given the specific objective of developing a model capable of recognizing handwritten scientific text in Spanish, a custom dataset was manually constructed, as existing public resources available on platforms such as Kaggle, Hugging Face, and Google Datasets proved inadequate for this task. Most existing datasets are restricted to cropped LaTeX equations, which are suitable for optical character recognition (OCR) tasks but insufficient for object detection models, such as DETR, that require precise positional information to localize and classify multiple elements within an image. To overcome these limitations, a new dataset was assembled using digitized handwritten personal notes that include a diverse combination of textual content, LaTeX expressions, and graphical elements such as geometric figures, function plots, and schematic diagrams.

The final dataset comprises **625** images, each containing a variable number of annotated objects, distributed across three categories: LaTeX expressions, text regions, and image or drawing elements. Every element, whether a handwritten word, a mathematical formula, or a graphical shape, was annotated with a bounding box and a corresponding label (Figure **8**). During annotation, spelling errors were corrected, but diacritical marks were intentionally omitted to maintain consistency with standard handwritten inputs. This process yielded a total of **28613** annotated elements across all classes.

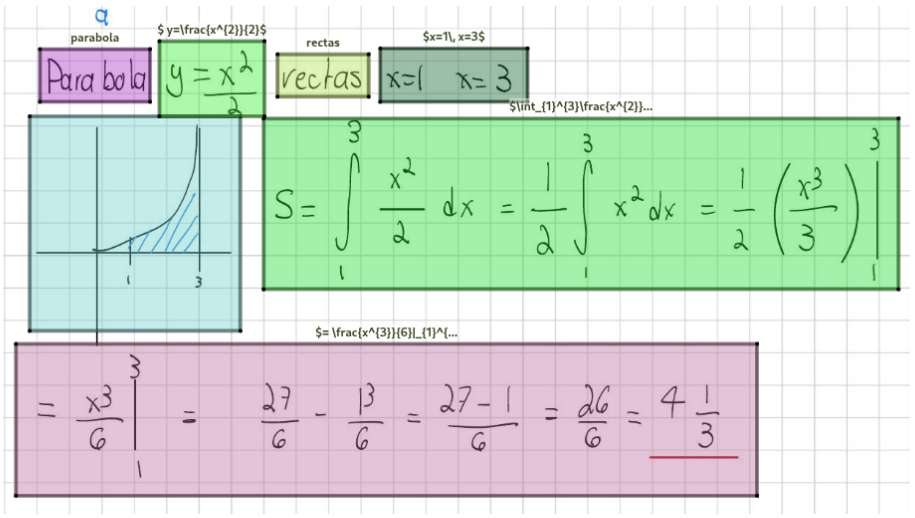
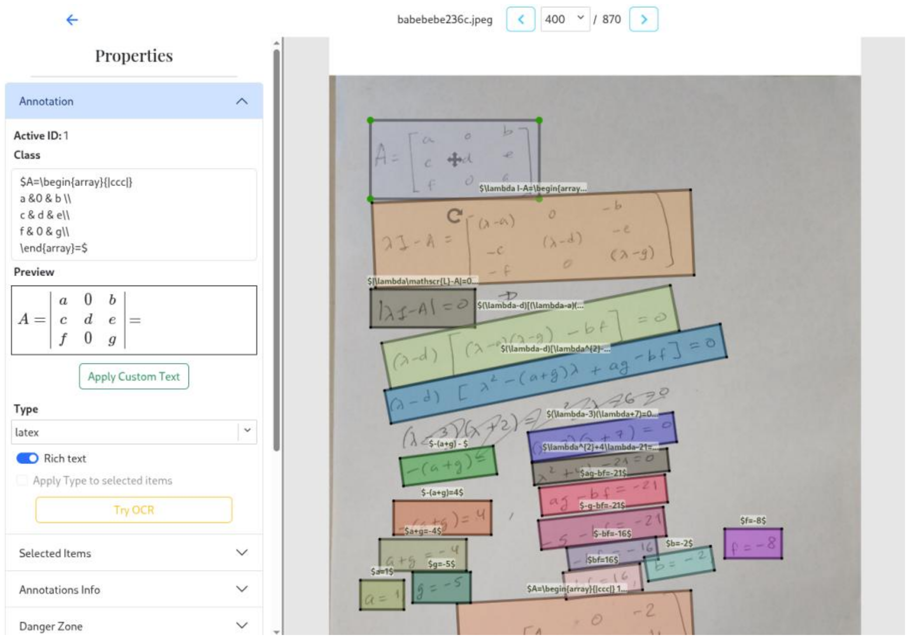


Figure 8. Text (dark pink, yellow), graph (light green) and LaTeX (remaining) annotation examples

As only three labels ("**text**", "**latex**", and "**image**") were defined for the bounding box annotations, an additional step was introduced to assign textual values to cropped elements. Specifically, "text" and "latex" regions were annotated with their corresponding transcriptions or LaTeX code, while "image" elements were left unlabeled, as they were intended exclusively for later reintegration into LaTeX documents during data reconstruction.

To facilitate and streamline the annotation process, a web-based annotation platform (Figure **9**) was developed. This tool enables the creation, visualization, and editing of bounding boxes in real time and integrates a live LaTeX rendering module to verify the accuracy of the annotated code. The platform supports the three defined categories—text, LaTeX, and image—and restricts entries to standard LaTeX syntax, excluding external packages or formatting variations such as font size or alignment, thus ensuring uniformity and reproducibility across annotations.



Active ID: 1

Class

```
$A=\begin{array}{|ccc|}
a &0 & b \\
c & d & e\\
f & 0 & g\\
\end{array}=$
```

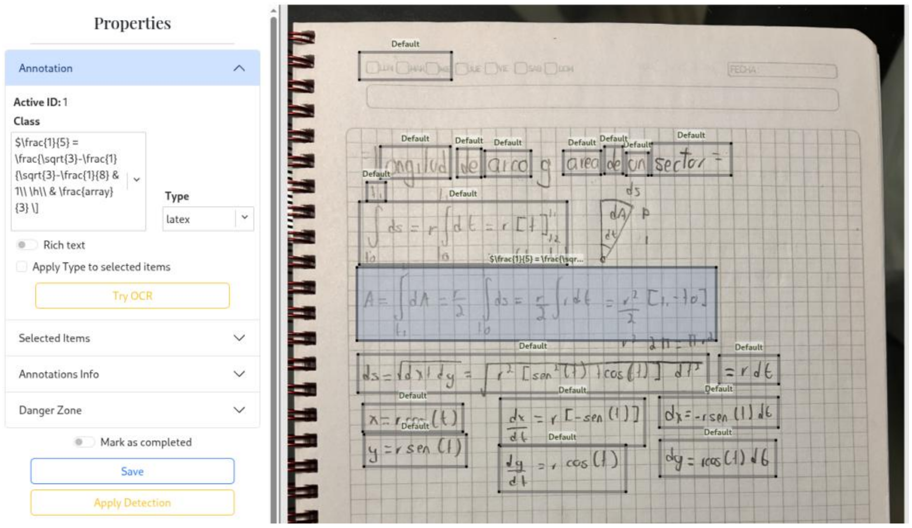
Preview

$$A = \begin{vmatrix} a & 0 & b \\ c & d & e \\ f & 0 & g \end{vmatrix} =$$

Type: latex

Figure 9. Example of the webpage built

To accelerate dataset generation, once enough samples had been labeled, preliminary fine-tuned versions of RT-DETR and TrOCR models were employed to assist in the semi-automatic annotation process. Although these models exhibited difficulties in detecting or transcribing elements in low-resolution or previously unseen data (Figure **10**), their integration significantly reduced the time required to complete the dataset compared to a fully manual workflow.



Active ID: 1
Class
$\frac{1}{5} = \frac{\sqrt{3}-\frac{1}{\sqrt{3}-\frac{1}{8} & 1\\ \h\\ & \frac{array}{3} \]
Type: latex

Figure 10. Results obtained with preliminary finetuned models

After completing the annotation process, the dataset was divided into subsets for object detection and OCR-related tasks. Specifically, **500** images, comprising a total of **26253** bounding boxes, were allocated for training DETR and YOLO-based models, while the remaining **125** images were reserved for validation. For sequence recognition models such as TrOCR and Nougat, the data were further partitioned into **19573** samples for training and **8389** for validation. The resulting dataset composition metrics for each subset are summarized in Table **1**.

Table 1. Datasets metrics

	Object Detection	OCR
Train images	500	19573
Validation images	125	8389
Train annotations	26253	--
Valid annotations	2360	--
Train text object elements	17289	12763
Valid text object elements	592	5434
Train LaTeX object elements	8476	6810
Valid LaTeX object elements	1668	2955
Train image object elements	488	--
Valid image object elements	100	--

3.2 Experimentation phase and data preparation

Preliminary experiments were carried out using a collection of handwritten mathematical notes to evaluate the baseline performance of existing object detection and text recognition models. In the object detection task, both evaluated model families failed to identify relevant elements within the images. This outcome is largely attributed to the nature of their pretraining datasets, which predominantly consist of natural or everyday objects, such as vehicles, animals, and household items, rather than fine-grained components like handwritten characters or mathematical symbols. Consequently, the models were unable to generalize to the structural and spatial characteristics of scientific handwriting.

In terms of text extraction, the TrOCR models produced largely inaccurate textual sequences when applied to handwritten samples in Spanish. Given that their pretraining data primarily comprised English-language text, the models exhibited a strong bias toward English output, frequently generating words and characters inconsistent with the input. Although their performance improved when tested on English samples, persistent errors were observed in the transcription of LaTeX expressions, particularly in symbol recognition and the generation of syntactically valid LaTeX code. Figure **11** shows an example input image for **TrOCR-large-stage-1** and **TrOCR-large-handwritten** for which both models generated an **empty output**.

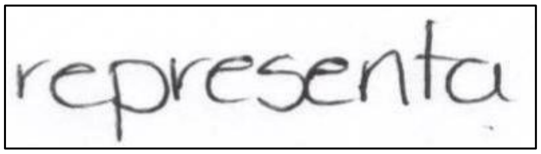


Figure 11. Example image used prior experimentation phase for TrOCR-family models

The Nougat models demonstrated comparatively better performance in the extraction and generation of LaTeX, particularly when using the fine-tuned LaTeX-specific version. Nonetheless, errors remained in the extraction of textual content, primarily due to the model's limited exposure to Spanish-language data. Moreover, Nougat consistently misinterpreted textual elements as mathematical expressions, attempting to encode them as LaTeX formulas or equations. Figure **12** shows an example of LaTeX input image for the base Nougat model. As output, the model generated a partially correct answer.

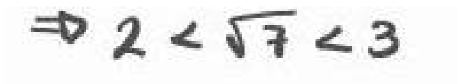


Figure 12. Example image used prior experimentation phase for Nougat-family models

Answer generated: **'<s>\Rightarrow \mathbb{Q} \leq \{\sqrt{ \pi}} \leq ,3</s>'**

When employing the base (vanilla) versions of the models, including the non-fine-tuned TrOCR variants, performance deteriorated significantly. These models were unable to accurately extract or reconstruct any meaningful textual or LaTeX sequences from the dataset, confirming the necessity of task-specific fine-tuning and the inclusion of language-diverse handwritten data for effective recognition in this domain.

Prior to the experimentation phase, the computational environment was configured using Python 3.11 as the primary programming language. An external hard drive was utilized as the main storage medium for the training datasets, model checkpoints, and performance logs. The hardware components of the machine are summarized in Table **2**.

Table 2. Hardware specifications

Hardware	Specification
Motherboard	ASUSTeK PRIME Z370-A
CPU	Intel i7-8700K(12)@4.700GHz
RAM	64 GB – DDR4
Storage	My Passport SSD 1TB
GPU	NVIDIA GeForce RTX 4090 24 GB NVIDIA GeForce RTX 5060 TI 16 GB

To establish the baseline hyperparameters for model fine-tuning, preliminary training sessions were conducted. During these sessions, the initial hyperparameters, primarily those recommended by the original authors of each model, were subjected to minimal adjustments to assess learning dynamics and convergence behavior. This process facilitated the identification of stable baseline configurations, which served as the basis for the subsequent experimental phase.

After defining these baselines, the fine-tuning stage was executed. For the Transformer-based models, a manually implemented training loop was employed, following the conventional deep learning workflow consisting of forward propagation, loss computation, backward propagation, and parameter updates. In contrast, models from the YOLO family were trained using the automated pipeline provided by their developers. Additionally, during the fine-tuning process, two distinct configurations were evaluated: first, models were trained exclusively on the original datasets; subsequently, data augmentation techniques were incorporated to assess their impact on model performance.

3.3 Object detection finetuning

For the fine-tuning of the object detection models, we selected **detr-resnet-101**, **rt-detr-r101vd** and **deformable-detr**, which contain approximately 60 million, 78 million and 40 million parameters, respectively. For the YOLO family, we selected the **YOLOv8-X** and **YOLOv11-X** models, each comprising roughly 57 million parameters. This selection was informed by a preliminary experimentation phase in which a broader set of architectures from the DETR, RT-DETR, and YOLO families, varying in parameter size, capabilities, and pre-training strategies, were evaluated. The chosen models exhibited the most favorable balance of performance and learning capacity for the specific requirements of our task.

During training, the dataset was loaded and divided into mini batches of varying sizes, determined by GPU capacity and the memory footprint of each model. A lazy loading strategy was employed to prevent GPU memory overflow. The **AdamW** optimizer was used due to its demonstrated stability and efficiency in training and fine-tuning deep learning models. Each model retained its native loss formulation: DETR-based models employed a Hungarian-matched multi-task loss combining classification loss with bounding box regression losses (L1 and GIoU), while the YOLO models used a composite loss consisting of classification (cross-entropy) and bounding box regression terms based on IoU/CIoU.

Within the training loop, the mini batches were dynamically retrieved, and images were pre-processed according to the standard pipeline of each architecture. Data augmentation was applied during image loading to avoid excessive GPU overhead. The augmentation procedures included random cropping, image degradation (jitter), blur, horizontal flipping, and adjustments in color and brightness.

Hyperparameter configurations used during fine-tuning, along with the corresponding performance results for each model family are summarized in Tables **3** and **4**.

Table 3. DETR-based models hyperparameters configuration and training results

	DETR	DETR Aug	RT-DETR	RT-DETR Aug	Deformable-DETR	Deformable-DETR Aug
Epochs	300	150	350	350	350	350
Batch size	10	10	10	10	3	5
Model learning rate	5e-5	1e-4	1e-4	1e-4	1e-4	1e-4
Backbone learning rate	5e-4	5e-4	5e-4	5e-4	5e-4	5e-4
Acc. gradient batch size	50	30	30	30	9	15
Training time (hours)	4:15	2:59	2:51	3:20	7:20	6:46
GPU consumption (GB)	19.8	15.6	16.5	18.5	22.4	23.2
Initial loss value	3.17	2.99	36.54	36.40	2.74	2.96
Final loss value	1.52	2.59	4.19	4.68	0.9	0.47

Table 4. YOLO family models hyperparameters configuration and training results

	YOLOv8	YOLOv8 Aug	YOLOv11	YOLOv11 Aug
Epochs	200	200	200	200
Batch size	10	10	10	10
Model learning rate	1e-4	1e-4	1e-4	1e-4
Training time (hours)	1:38	1:00	1:00	1:23
GPU consumption (GB)	8.72	8.83	10.7	10.6
Initial loss value	1.70	1.73	1.70	1.72
Final loss value	0.60	0.60	0.17	0.6

Although the YOLO models were configured to train for a total of 200 epochs, both terminated training at epoch 125. This behavior is explained by the internal early-stopping mechanism, which halts optimization once no further improvement in validation performance is observed over a predefined number of consecutive epochs.

During the training of the DETR-family models, it was observed that both DETR and RT-DETR required approximately 100 epochs before exhibiting clear signs of learning. This behavior is likely attributable to the characteristics of the datasets used during their initial pre-training, primarily containing common objects, and to the relatively small size and high density of the text regions in our task. Nevertheless, once learning stabilized, both models successfully identified text sequences and generated bounding boxes without truncation or positional distortions.

In contrast, the Deformable-DETR model demonstrated rapid convergence within the first 100 epochs, followed by a refinement phase characterized by progressive improvements in feature extraction and bounding box localization.

Although data augmentation techniques were incorporated into the training process, they did not yield substantial improvements in terms of learning rate or convergence speed. However, these techniques did contribute to more robust feature extraction and enhanced stability during training.

3.4 OCR finetuning

For the fine-tuning stage of the OCR models, we selected **TrOCR-Large-Stage-1**, **TrOCR-Large-Handwritten**, **TrOCR-Large-Spanish**, **Nougat-Latex**, and **Nougat-Base**. The TrOCR-based models contain approximately 609 million parameters, while the Nougat-based models comprise roughly 300 million parameters.

Preliminary tests were conducted to determine suitable training hyperparameters. The results indicated that the optimal configurations closely aligned with those originally proposed by the authors of each architecture, with minor adjustments, such as reducing the learning rate and decreasing the number of epochs, to facilitate faster convergence.

The training loop was manually implemented and consisted of loading cropped image samples together with their corresponding labels. As in the object detection setup, data augmentation was applied during image retrieval to minimize memory usage. Preprocessing and tokenization were performed using the tools provided with each model. However, for both model families, it was necessary to manually construct the attention masks and shifted decoder inputs due to unexpected behavior in the default pipeline. Specifically, the tokenizer frequently failed to insert the end-of-sequence token, causing the models to continue generating tokens until the predefined maximum sequence length was reached.

For optimization, cross-entropy loss was employed, given that the models output token-level probability distributions. Parameter updates were performed using the **AdamW** optimizer, which demonstrated stability and efficiency throughout the fine-tuning process. The hyperparameters used for fine-tuning are summarized in Tables **5** and **6**.

Table 5. TrOCR and Nougat models configuration

Hyperparameter	TrOCR-LS-1	TrOCR-LH	TrOCR-LS	Nougat-Base	Nougat-Latex
Epochs	15	15	10	10	10
Batch size	5	5	5	5	5
Model learning rate	4e-5	4e-5	4e-5	4e-5	4e-5
Training time (hours)	4:45	4:28	3:08	1:29	3:49
GPU consumption (GB)	14.6	14.7	9.76	20.2	11.2
Initial loss	1.88	2.20	1.88	1.00	1.27
Final loss	0.37	0.88	0.41	0.08	0.09

Table 6. TrOCR and Nougat models configuration with data augmentation samples

Hyperparameter	TrOCR-LS-1 Aug	TrOCR-LH Aug	TrOCR-LS Aug	Nougat-Base Aug	Nougat-Latex Aug
Epochs	10	10	10	10	10
Batch size	5	5	5	5	5
Model learning rate	4e-5	4e-5	4e-5	4e-5	4e-5
Training time (hours)	1:34	2:55	3:18	1:35	3:56
GPU consumption (GB)	14.71	7.1	20.8	17.9	15.5
Initial loss	1.56	2.17	2.22	1.56	1.28
Final loss	0.18	0.72	0.53	0.18	0.1

During the training process, we observed that the models from both families exhibited similar behavior, independent of the data augmentation strategies applied. In all cases, convergence proceeded slowly but consistently.

To further evaluate the robustness of these models, additional experiments were conducted in which the number of training epochs was increased while the learning rate remained fixed. These experiments revealed that the models became unstable and ultimately unusable after approximately 15 epochs. Beyond this threshold, the models exhibited the characteristic signs of gradient explosion: instead of continuing to decrease smoothly, the loss curves began to rise sharply, indicating divergence.

4 Evaluations and results

The evaluation of the models was carried out by grouping them according to the nature of their respective tasks. For the object detection models, performance was assessed using standard metrics, including Intersection over Union (IoU), Precision, Recall, F1-score, mean Average Precision (mAP), and its variants at different thresholds, namely mAP@50 and mAP@75. These metrics provide a comprehensive view of both the localization accuracy and the overall detection performance of the models.

In the case of optical character recognition (OCR), the task was approached not strictly as a sequence translation problem, but rather as a form of structured "code" generation, particularly relevant for LaTeX-based outputs. Consequently, evaluation was conducted using Word Error Rate (WER) and Character Error Rate (CER), which are widely recognized metrics for assessing the quality of generated text sequences, along with the BLEU and ROUGE metrics.

Additionally, to illustrate the practical performance of the models, qualitative results are presented in the form of images obtained during the inference stage. These include examples of bounding box generation for object detection and LaTeX sequence generation for OCR, thereby providing visual evidence of the outputs alongside the quantitative evaluation.

4.1 Object detection results

The evaluation of the object detection models followed a workflow analogous to that employed during training. The evaluation dataset was first loaded and preprocessed dynamically and on demand, thereby preventing excessive GPU memory usage. Subsequently, the trained model was initialized, and an evaluation cycle was executed in which the predicted outputs—bounding box coordinates and class labels—were compared against the corresponding ground-truth annotations. Performance was then quantified using the metrics previously described. The results of this evaluation are presented in Tables **7** and **8**.

For both DETR and RT-DETR, the fine-tuned models consistently achieved improved performance relative to their original variants (Figure **13**, **14**), despite exhibiting comparatively high loss values. This behavior is largely attributable to the inclusion of auxiliary losses, an inherent feature of DETR architecture. These auxiliary losses, computed at intermediate decoder layers, help stabilize parameter updates during pre-training and fine-tuning, thereby promoting more reliable convergence.

The YOLO models demonstrated strong adaptability to datasets with characteristics substantially different from those used during their pre-training. They also produced bounding boxes with higher spatial precision than those generated by DETR and RT-DETR (Figure **15**), which is advantageous for tasks requiring fine-grained localization. However, during the classification of detected regions, both YOLO models exhibited systematic errors when processing text with irregular character spacing or poorly formed punctuation. In these cases, they frequently produced multiple bounding boxes for a single word and incorrectly classified them as LaTeX elements.

Table 7. DETR models evaluation results

	DETR	DETR Aug	RT-DETR	RT-DETR Aug	Deformable-DETR	Deformable-DETR Aug
IOU	0.4354		0.5398	0.5093	0.4992	0.6434
Precision	0.6398		0.6881	0.6551	0.6888	**0.8020**
Recall	0.6688		0.7767	0.7033	0.7081	0.7947
F1	0.6540		0.7297	0.6988	0.6983	0.7984
mAP	0.1098		0.2719	0.2515	0.1926	0.3616
mAP@50	0.2204		0.3957	0.3150	0.2635	0.5089
mAP@75	0.0950		0.2665	0.2256	0.1513	0.2532

Table 8. YOLO models evaluation results

	YOLOv8	YOLOv8 Aug	YOLOv11	YOLOv11 Aug
IOU	0.4850	0.4576	0.4816	0.4951
Precision	0.5965	0.5650	0.5921	0.6142
Recall	0.7870	0.7951	0.7876	0.7768
F1	0.6786	0.6606	0.6760	0.6860
mAP	0.3044	0.3472	0.2904	0.3551
mAP@50	0.4099	0.4628	0.3933	0.4722
mAP@75	0.3073	0.3586	0.2850	0.3672

The Deformable-DETR models generated bounding boxes with high spatial accuracy and minimal overlap (Figure **16**). Nonetheless, like other transformer-based detectors, they showed limitations when handling text or sequences rotated by up to 10 degrees. Under these conditions, the models tended to produce bounding boxes that were larger than necessary. This limitation stems from the inability of DETR, RT-DETR, and Deformable-DETR to generate orientation-aware bounding boxes aligned with the direction of the text.

Although all models are theoretically capable of generating up to 300 bounding boxes per image, they generally produced fewer than 120 bounding boxes across the three target classes, with the text category being the most frequently assigned label. This constraint presents challenges for images containing more than 200 small or densely packed objects, as the upper limit on bounding box generation restricts the total number of detectable elements. Consequently, several inference experiments resulted in incomplete object detection due to this threshold.

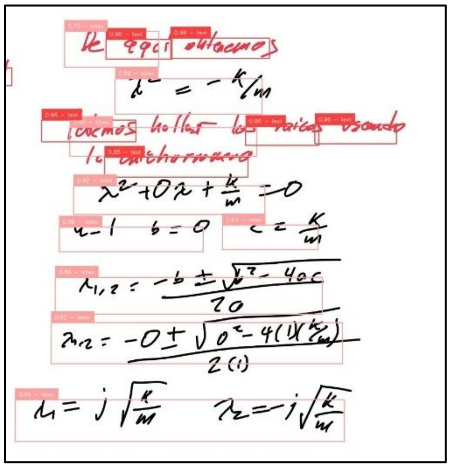


Figure 13. DETR generation output

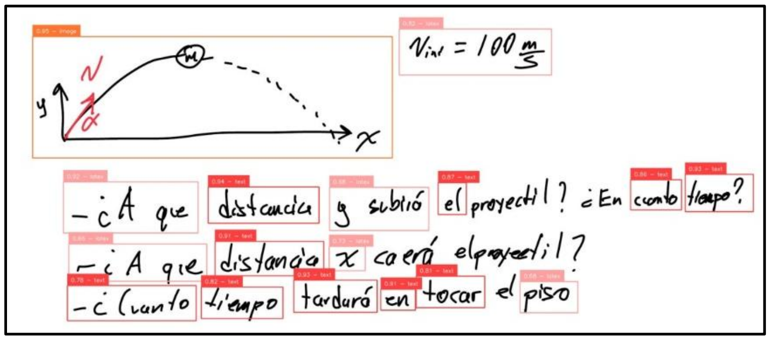


Figure 14. RT-DETR generation example

Finally, when evaluating the models trained with data augmentation techniques, we observed a decline in performance compared to their counterparts trained without augmentation. All models exhibited incorrect bounding box generation (Figure **17**), both in terms of quantity and spatial placement, as well as reduced classification accuracy for elements that were otherwise correctly detected.

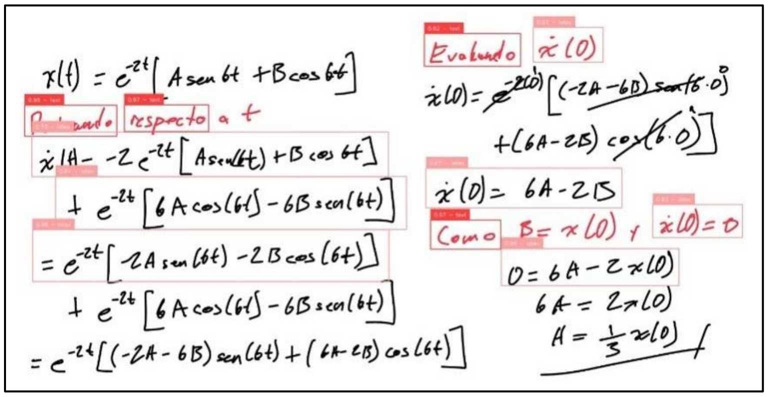


Figure 15. YOLOv11 detection example

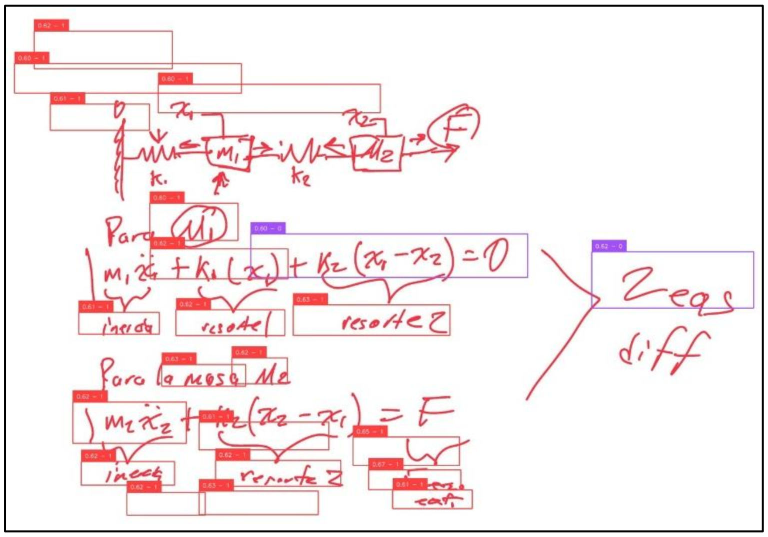


Figure 16. Deformable-DETR detection example

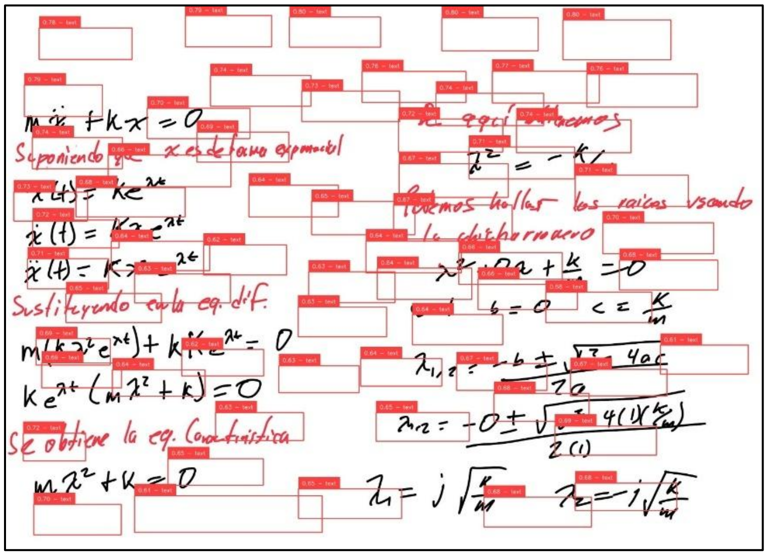


Figure 17. Bounding boxes generated by DETR model trained with augmented data

4.2 OCR results

The evaluation of the TrOCR models followed a workflow analogous to that employed during training. The evaluation dataset was loaded, and both images and their corresponding labels were preprocessed before being provided as input to the model. Unlike in the training phase, the use of an attention mask was not necessary during evaluation, since the model generates output sequences in an autoregressive manner. The results obtained from this process are summarized in Tables **9** and **10**.

Table 9. Evaluation results from OCR models without data augmentation train

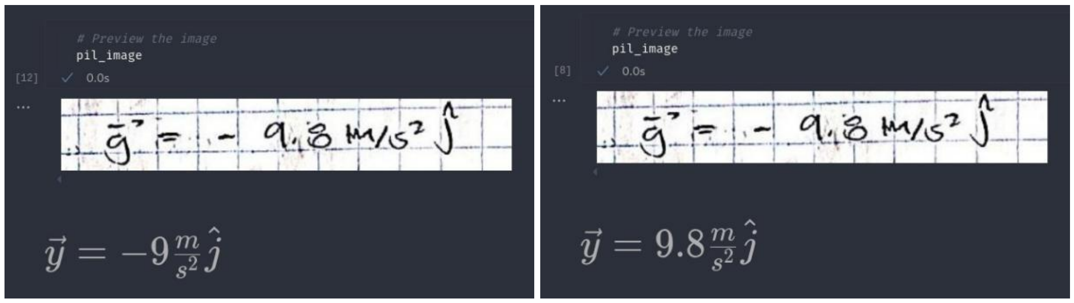
	TrOCR-LS-1	TrOCR-LH	TrOCR-LS	Nougat-Base	Nougat-Latex
WER	76.60	76.36	61.12	57.31	54.95
CER	42.67	103.72	24.42	26.48	27.93
BLEU	46.44	18.20	50.65	65.95	64.63
ROUGE-1	0.74	0.66	0.74	0.75	0.74
ROUGE-L	0.74	0.65	0.73	0.75	0.74

Table 10. Evaluation results from OCR models with data augmentation train

	TrOCR-LS-1 Aug	TrOCR-LH Aug	TrOCR-LS Aug	Nougat-Base Aug	Nougat-Latex Aug
WER	65.98	68.25	59.02	63.70	61.71
CER	37.76	86.11	27.14	33.12	31.13
BLEU	57.95	48.64	50.35	61.44	63.47
ROUGE-1	0.69	0.72	0.75	0.71	0.72
ROUGE-L	0.69	0.71	0.74	0.71	0.72

As shown in Tables **9** and **10**, the models that demonstrated the greatest learning capability, both with and without data augmentation, were the Nougat-based architectures and the TrOCR model previously fine-tuned in Spanish. As expected, the Nougat model pretrained on LaTeX sequences exhibited strong adaptability to the dataset. This behavior is illustrated in Figure **18**, which present examples of LaTeX sequence generation for the TrOCR-Large-Spanish and Nougat-LaTeX models, respectively.

Extensive testing further revealed that models without prior exposure to LaTeX syntax were able to correctly generate the required code for short expressions, such as equalities, function definitions, and fractions. However, these models were unable to produce valid code for more complex structures, such as tabular data (Figure **19**). In contrast, the Nougat-LaTeX model, having been pretrained on LaTeX-rich data, more reliably generated correct table definitions due to its prior knowledge of the underlying structural format.



```
# Preview the image
pil_image
```

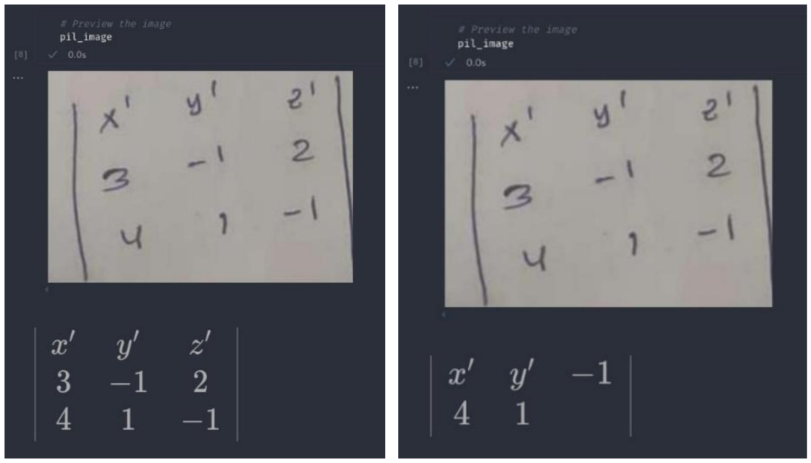
$\vec{y} = -9\frac{m}{s^2}\hat{j}$

```
# Preview the image
pil_image
```

$\vec{y} = 9.8\frac{m}{s^2}\hat{j}$

Figure 18. Example of LaTeX sequences generated by OCR models. a) TrOCR Spanish. b) Nougat-Latex

With respect to general text recognition, all models performed adequately, demonstrating a strong ability to transcribe cursive handwriting. Nonetheless, both the TrOCR-Large-Spanish and TrOCR-Large-Handwritten models exhibited notable limitations when processing longer sequences (exceeding 20 characters). In these cases, the models frequently repeated previously generated segments and failed to correctly produce the end-of-sequence token. Consequently, the output often contained inaccuracies, ranging from partially incorrect transcriptions to completely incoherent sequences.



```
# Preview the image
pil_image
```

$$\begin{vmatrix} x' & y' & z' \\ 3 & -1 & 2 \\ 4 & 1 & -1 \end{vmatrix}$$

```
# Preview the image
pil_image
```

$$\begin{vmatrix} x' & y' & -1 \\ 4 & 1 & \end{vmatrix}$$

Figure 19. Tabular data generated by OCR models. a) Nougat-Latex. b) TrOCR Spanish

5 Conclusions

In conclusion, the findings of this study demonstrate that transformer-based models can extend their knowledge to new domains, provided that the target tasks remain compatible with the objectives of their pre-training. Although vision-focused transformer architectures continue to evolve, they already offer substantial advantages over traditional convolutional approaches, achieving comparable or superior performance, albeit with increased computational and energy demands.

Among the evaluated detection models, RT-DETR and Deformable-DETR exhibited the strongest overall performance, accurately detecting, localizing, and classifying the proposed label categories. The YOLO family of models also showed a high degree of adaptability, despite not being originally designed for the characteristics of this task. However, none of the evaluated models could produce oriented bounding boxes aligned with the direction of handwritten text, which constitutes a significant limitation and leads to a loss of information for sequences containing angled or vertically oriented elements.

With respect to object detection, particularly text, the introduction of data augmentation techniques did not yield improvements in model learning or convergence.

For OCR tasks, the results indicate a clear trend: prior knowledge of the target sequence format and language significantly facilitates convergence and promotes correct model adaptation. This is evident in the Nougat-LaTeX model, which reliably generated LaTeX-formatted output, in contrast to the other models that frequently produced incomplete or inconsistent sequences.

Similarly, using a base model already trained in the target language improves performance by reducing the need for abrupt vocabulary restructuring and by preserving language-specific features such as lexemes and grammatical markers. Although data augmentation did not benefit the object detection models, it did prove slightly advantageous for finer-grained tasks such as OCR, contributing to more stable learning.

The dataset used in these experiments provided an adequate foundation for fine-tuning models for text detection and extraction. Although relatively small, containing 625 images and 28,613 annotated objects, its diversity in language (Spanish), handwriting styles (cursive and printed), and content topics makes it a valuable complement to larger datasets. Despite the use of only three label classes, the LaTeX category contains programming-related structures and isolated annotations that further enrich the dataset. Future work includes exploring architectures capable of generating text-oriented bounding boxes, investigating hybrid detection–recognition models, and developing modifications to existing architectures that improve the precision and granularity of detected and extracted elements. In addition, an integrated approach that unifies detection and sequence generation within a single model may offer a promising direction, provided that such integration does not compromise performance in either task.

Data Availability

The code needed to replicate the experiments, as well as the datasets used, are publicly available in the repository https://github.com/rodrigoalvarez-20/thst2l.

Acknowledgments

We thank the National Polytechnic Institute for its support in carrying out this work through grant IPN: SIP-20253457.

References

Blecher, L., Cucurull, G., Scialom, T., & Stojnic, R. (2023). Nougat: Neural optical understanding for academic documents. arXiv. https://doi.org/10.48550/arXiv.2308.13418

Carion, N., Massa, F., Synnaeve, G., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. arXiv. https://arxiv.org/abs/2005.12872

Chaudhuri, A., Mandaviya, K., Badelia, P., & Ghosh, S. K. (2016). Optical character recognition systems for different languages with soft computing. In *Studies in Fuzziness and Soft Computing*. Springer. https://doi.org/10.1007/978-3-319-50252-6

Javed, M., Nagabhushan, P., & Chaudhuri, B. (2013). Extraction of projection profile, run-histogram and entropy features straight from run-length compressed text-documents. En *Proceedings of the IEEE APSIPA Annual Summit and Conference*. IEEE. https://doi.org/10.1109/ACPR.2013.147

Khanam, R., & Hussain, M. (2024). YOLOv11: An overview of the key architectural enhancements. arXiv. https://arxiv.org/abs/2410.17725

Kim, G., Hong, T., Yim, M., Nam, J., Park, J., Yim, J., Hwang, W., Yun, S., Han, D., & Park, S. (2022). OCR-free document understanding transformer. arXiv. https://arxiv.org/abs/2111.15664

Li, M., Lv, T., Chen, J., Cui, L., Lu, Y., Florencio, D., Zhang, C., Li, Z., & Wei, F. (2021). TrOCR: Transformer-based optical character recognition with pre-trained models. arXiv. https://arxiv.org/abs/2109.10282

Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2022). A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems, 33*(12), 6999–7019. https://doi.org/10.1109/TNNLS.2021.3136258

Mienye, I. D., Swart, T. G., & Obaido, G. (2024). Recurrent neural networks: A comprehensive review of architectures, variants, and applications. *Information, 15*(9), 517. https://doi.org/10.3390/info15090517

Mor, B., Garhwal, S., & Kumar, A. (2020). A systematic review of hidden Markov models and their applications. *Archives of Computational Methods in Engineering, 28*(3), 1429–1448. https://doi.org/10.1007/s11831-020-09422-4

Mutlag, W. K., Ali, S. K., Aydam, Z. M., & Taher, B. H. (2020). Feature extraction methods: A review. *Journal of Physics: Conference Series, 1591*(1), 012028. https://doi.org/10.1088/1742-6596/1591/1/012028

Raisi, Z., Naiel, M. A., Younes, G., Wardell, S., & Zelek, J. S. (2021). Transformer-based text detection in the wild. En *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 3162–3171). IEEE.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). You only look once: Unified, real-time object detection. arXiv. https://arxiv.org/abs/1506.02640

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention is all you need* (arXiv:1706.03762). https://arxiv.org/abs/1706.03762

Yaseen, M. (2024). What is YOLOv8: An in-depth exploration of the internal features of the next-generation object detector. arXiv. https://arxiv.org/abs/2408.15857v1

Zhang, H., Li, F., Liu, S., Zhang, L., Su, H., Zhu, J., Ni, L. M., & Shum, H.-Y. (2022). DINO: DETR with improved de-noising anchor boxes for end-to-end object detection. arXiv. https://arxiv.org/abs/2203.03605

Zhu, X., Su, W., Lu, L., Li, B., Wang, X., & Dai, J. (2021). Deformable DETR: Deformable transformers for end-to-end object detection. arXiv. https://arxiv.org/abs/2010.04159