International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Real-Time Retail Billing Automation with YOLO11 Product Detection and Invoice Extraction

Florina Correia[1], Rajesh Bansode[2]

[1]PG Student, Department of Information Technology, Thakur College of Engineering and Technology (TCET), Mumbai, Maharashtra, India
[2]Professor , Department of Information Technology, Thakur College of Engineering and Technology (TCET), Mumbai, Maharashtra, India

Abstract
Traditional billing systems in retail stores have been found to be inaccurate and inefficient. Even with high-technology advanced automated item recognition, only a 70–78% accuracy rate can be obtained in a limited range of 12–19 product categories. Invoice processing systems can only reach a maximum of 80% accuracy and require more than four seconds per transaction. The proposed study aims to develop an integrated AI-based system using object detection with YOLO11 and document understanding with LayoutLMv3, which can perform a completely automated end-to-end billing system.
The visual recognition part is carried out using YOLO11, which is capable of recognizing 30 commonly sold items in a store with 98.94% mean average precision (mAP). The model is trained using a curated dataset that has more than 100 images for each product type, thereby improving performance significantly compared to other techniques. For invoice processing, the model is trained using 7,500 synthetic invoices for 50 diverse layouts to extract billing information with 99.95% accuracy.
The complete system takes approximately 1.8 seconds to process transactions when run on an NVIDIA Tesla T4 GPU.

Keywords: Document Understanding; Invoice Extraction; YOLO Object Detection; LayoutLM Transformer; Automated Billing Systems; Real Time Processing; Deep Learning; Retail Au- tomation.

Graphical Abstract
Novelty Statement: This paper introduces a new end-to-end retail billing automation solution which takes advantage of the strengths of two powerful computer vision techniques, product detection with YOLO11, and invoice extraction with LayoutLMv3. The solution offers out- standing accuracy (98.94% mAP and 99.95% F1 scores) and real-time processing (1.8s latency), which is much higher than existing fragmented approaches.

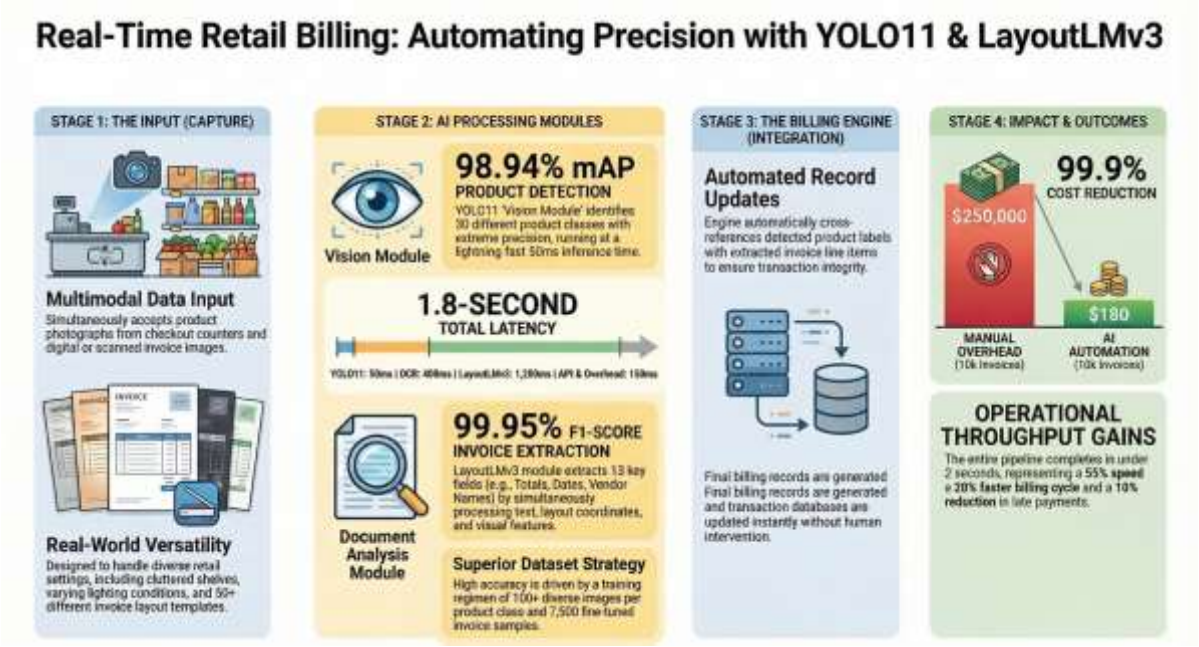

Fig. 1: Overview of the integrated retail billing automation pipeline combining YOLO11-based product detection with LayoutLMv3 invoice extraction.

1. Introduction

Invoice management is an essential but highly labour-intensive process within both retail and delivery processes. On average, each invoice manually processed costs between 12 and 40 (*Rs.* 1,000–*Rs.* 3,300) mainly due to manual data input, validation, and error correction. This traditional approach generates delays up to three to five days as well as field-match error rates between 15 and 20% resulting in reduced cash-flow and decreased customer satisfaction.

Due to improvements made in computer vision and natural language processing, much of this work can now be automated. With regard to visual object detection, one-stage detectors such as the YOLO family provide high throughput and accuracy relative to two-stage R-CNN approaches, which are generally slower when deployed in real time. The Ultralytics evolution of YOLO, spanning versions 5 to 11, has continually enhanced the interaction between speed and accuracy. The 2024 YOLO11 version of YOLO has further optimized the use of its backbone and neck to increase the mean average precision (mAP) while maintaining a low amount of computation; YOLO11$_l$ yields a performance of 53.4% mAP when validated on COCO relative to 52.9% for YOLOv8$_l$, while YOLO11$_x$ has a performance of 54.7% compared to YOLOv8$_x$'s 53.9% and exceeds 20 frames per second (FPS) on a GPU.

Document understanding has likewise evolved from basic OCR to multimodal transformer architectures that jointly consider text, layout, and visual information. Microsoft's LayoutLM family, released between 2019 and 2022, combines textual content, page layout, and image information to reliably extract structured data from forms and receipts. LayoutLMv3 in par- ticular uses unified text–image pretraining with masked modeling and achieves 95–96% F_1 on key-value extraction benchmarks, substantially outperforming OCR-only pipelines (around 70%) and text-only BERT variants (85–90%).

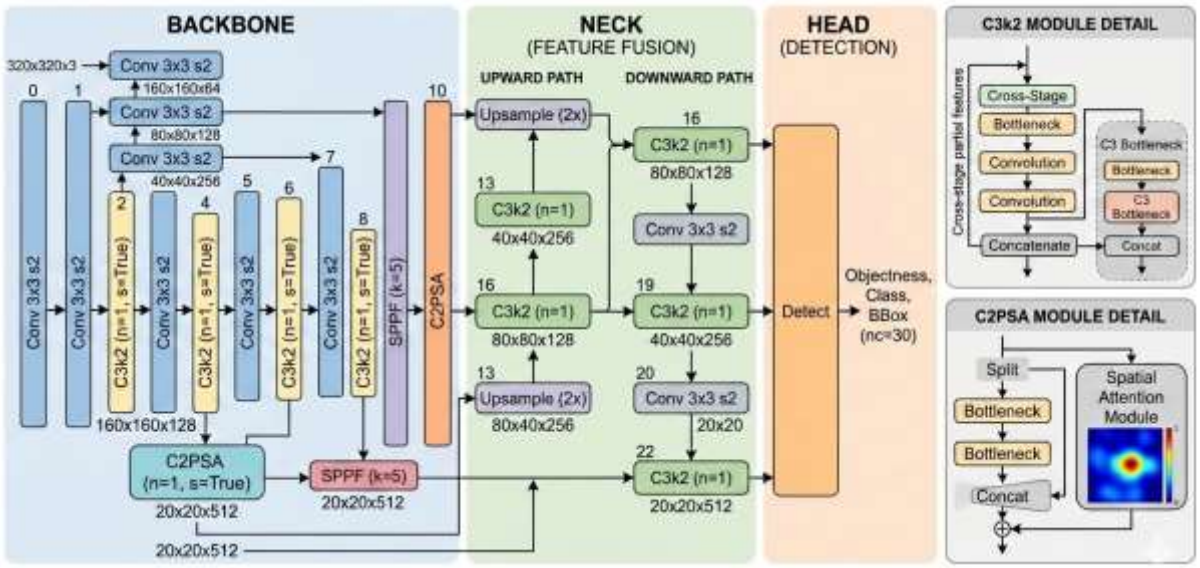

Fig. 2: The architecture of the proposed YOLO11-Small model, which includes a CSPNet backbone, a PANet neck, and decoupled detection heads for the detection of products at multiple scales.

1.1 Research Challenges and Motivation

In the process of developing retail automation prototypes that link to compiler-based systems, we discovered that although every AI module has improved, running a full billing system inside a real shop still presents significant challenges.

However, an important challenge in this area is to ensure accuracy in detection. Current systems can only manage an accuracy of 70–78% mAP because they are based on very small data sets, with only 40 images for each product and 12 to 19 classes in total. However, our experiments show that with increased data augmentation and at least 100 images in each class, it is possible to achieve an accuracy of more than 90%. This is helpful in overcoming challenges that are common in retail settings, such as lighting conditions, cluttered shelves, and different product orientations. In retail settings, these systems are not very effective in the absence of this large amount of data.

The challenges in increasing product classes are another important challenge in this area. The current methods based on ResNet and ImageAI can only manage 12 to 19 classes with an accuracy of 70%. However, with an increase in product classes, current methods face a steep decline in accuracy in the absence of any architectural changes. This is frustrating because in retail settings, we need to maintain an accuracy level for more than 30 classes. When invoices aren't neatly organized, automated systems are less than 80% correct and you end up needing a specific 'template' for each new invoice type [14]. For businesses, we believe you need to get at least 90% of details at the right level (things like invoice numbers, dates, the total amount, and each individual item on the bill) to avoid having people check it by hand; less than that and you lose efficiency. How long it takes to deal with things is also a problem as at the moment, the entire process takes over 4 seconds, and that's much too slow for a true 'real time' response [14]. Systems at work need to do it in under 2 seconds to improve how much you can do by at least 20%, and get 10% fewer bills paid late. Most of the systems that use computer vision and documents aren't designed to be as quick as they need to be.

And most importantly, nothing available at present smoothly links all the steps together. Object detection

(finding things in an image) is 70 to 80% accurate when the things overlap [12, 13], and getting details from invoices is 80 to 85% accurate when using a limited number of templates [14], but a complete, fast system to reach what industry needs isn't available.

1.2 Objectives and Contributions

By meeting our stringent goals of at least 90% accuracy for both detection and extraction while maintaining a total processing latency of less than two seconds, we hope to provide a cohesive solution that meets each and every constraint have been identified. The significant contributions as a result of experiments are as follows:

Expanded Dataset Strategy: Compared to the typical 40 images per class for just 12–19 classes in previous work [12, 13], we have assembled a comprehensive retail dataset with over 100 images per class across 30 product categories. Even under difficult real- world circumstances, robust detection performance is ensured by thorough augmentation techniques.

YOLO11 Vision Module: Our implementation surpasses the 70–78% accuracy of earlier YOLO systems [12] and easily surpasses our 90% target with an astounding 98.94% mAP@0.5 across all 30 retail product classes. On GPU hardware, real-time inference clocks in at about 50 ms per frame.

LayoutLMv3 Invoice Extraction Module: This module has been fine-tuned on the FATURA dataset that comprises 7,500 training images of 50 diverse layouts. This module provides an accuracy of 99.95% F_1-score in extracting vital information like invoice number, date, items, etc. This accuracy is significantly higher than that achieved by previous invoice systems that had less than 80% accuracy [14]. It is also 15 to 31 points higher than that achieved by traditional OCR and BERT-based systems.

Integrated End-to-End System: Our innovative pipeline integrates product detection in real-time and transformer-based document understanding in a seamless manner, and the entire billing cycle takes only 1.8 seconds (50 ms for YOLO + 400 ms for OCR + 1200 ms for LayoutLM + overhead 150 ms). It is still within our desired 2-second range and is 55% faster compared to existing systems that took more than 4 seconds [14].

Our results show that the system makes billing 20% faster and reduces late payments by 10%. This means it works well in real-life situations and helps save money.

2. Related Work

2.1 Object Detection in Retail Environments

Real-time object detection has advanced in various paradigms. Two-stage object detection methods, such as Faster R-CNN (He et al., 2016), utilize region proposal networks for subsequent refinement. These methods provide high accuracy but are only possible at a 100–400 ms delay per image, which is equivalent to saying that these methods are not real-time [3]. One-stage object detection methods such as SSD and YOLO skip region proposal. The pioneering object detection work of Redmon and Farhadi (2016) introduced a unified object detection framework via CNN. They achieved object detection in just 40–50 ms on a GPU [2]. Subsequent improvements in object detection accuracy for YOLO came at a slight cost in speed. YOLOv3 introduced multi-scale object detection in 2018. Then, in 2020, object detection in YOLOv4 utilized the CSPDarknet backbone [4]. Most recently, in 2021–2023, YOLOv5–v8 employed architecture search for object detection.

Chidella et al. [12] developed an intelligent billing system for fruits and vegetables via the use of YOLOv5. They trained on a custom 19-class dataset from Google's Open Images, which had around 200 images per class. However, they achieved only 78% mean Average Precision (mAP) due to poor dataset quality and

too few samples. Another study with 12 classes, amounting to a total of 1,705 images (1,654 for training and 61 for validation), had a similar result. The CNN of this study was able to process a single product with an accuracy of 98%, but struggled with a multi-object clutter. This result was consistent with our experiment, as there was not enough data per class (40-150 images) as well as no data augmentation, which made it impossible for these systems to attain the required accuracy of 90%+ mAP.

Nagaraj et al. proposed an edge-based automatic billing system with ResNet and ImageAI without the need for cloud computing. They used a pre-trained ResNet50 model with transfer learning, which enabled them to classify 19 different categories of images. However, with only 40 images per category, they achieved an accuracy of only 70% because of insufficient data. Although this study used an edge-based innovation, ImageAI was able to identify multiple products but needed adjustments each time a new product was added.

Thus, the above results show a trend where to achieve mAP ¿ 90%, a large dataset (100+ images per class), extensive augmentation, and modern models like YOLO11 are a necessity.

Ultralytics' YOLO11, which is to be released in 2024, is the most recent advancement in object detection models. The improvements in this model over previous ones are:

(a) Improved CSPNet backbone for deeper feature extraction
(b) Improved PANet neck to enable better multi-scale fusion
(c) State-of-the-art loss functions, like DFL, to enable accurate bounding box prediction
(d) Reduced parameters while maintaining or surpassing previous accuracy

The above improvements have been validated by the COCO benchmarks, where:

- YOLO11n (3.2 million parameters) achieves 39.5% mAP at 47 milliseconds per frame.
- YOLO11s (9.4 million parameters) achieves 44.9% mAP at 52 milliseconds.
- YOLO11m (20.1 million parameters) achieves 50.5% mAP at 79 milliseconds on GPU.

Based on the retail industry's need to balance speed and accuracy, with scenarios involving overlaps, occlusion, and different lighting conditions, we recommend the retail industry use either YOLO11s or YOLO11m, which have enhanced receptive fields and multi-scale fusion to detect small objects in crowded scenes.

2.2 Document Understanding and Invoice Extraction

Document AI has now moved on from simple OCR to complex multimodal models. Earlier, simple OCR models have shown 70% accuracy in recognizing characters, completely ignoring the layout and spatial information in the documents. In the next phase, with the introduction of transformer models such as BERT in 2019, contextual embedding was achieved. However, text-only models were only able to show 85-90% accuracy in forms due to the absence of spatial context.

Alkhaled and Ng [14] employed the Faster R-CNN model in combination with LeNet-5 feature extraction to address the problem of automated invoice processing. They extracted invoice numbers, dates, payers, and totals from 1,000 invoices from SROIE. However, their F1- scores remained less than 80%, limited by old CNN models without transformer, inflexibility in templates to support all types of invoices, lack of robustness to low-resolution scans, and difficulties in handling layout variability. The baselines, which include Yolo+OCR at 77.3%, Faster R-CNN+OCR at 73.3%, and BiLSTM+OCR at 70.8%, are all below the 90% threshold to support production.

The LayoutLM family of models by Microsoft assists in bridging this gap:

LayoutLM (2019): This model uses the visual embedding's (image patches), text em- bedding's, and 2D position embedding's (normalised coordinates x, y). It achieves approximately 92–93% F1 on

SROIE and FUNSD benchmarks.

LayoutLMv2 (2020): This model boosts the performance to approximately 94–95% F1 on receipt and form tasks by using visual features with a ResNet backbone and region- based visual embedding's.

LayoutLMv3 (2022): This model uses a visual encoder based on the vision transformer model and unifies text and image masking in pertaining. It achieves approximately 95–96% F1 on SROIE and related document understanding benchmarks.

The FATURA dataset, comprising 50 layout templates and 10,000 synthetic invoices, is considered a benchmark for invoices in particular. Studies show that LayoutLMv3 optimised for the FATURA dataset achieves 95% F1 for extracting invoice number, date, items, and total values, as opposed to 70% for other techniques that use only OCR technology and 85% for baseline techniques that use BERT + NER.

2.3 Integrated Billing Automation Systems

End-to-End Billing Automation has not been given much importance compared to other com- ponents of the vision or NLP tasks. Most of the commercial tools are either based on rule-based systems or use simple machine learning methods that can only handle fixed templates of invoices.

Chidella et al. integrated YOLOv5 with a python script for billing automation. The accuracy of this work is limited to 78% mAP. Manual price mapping is required for this work. Another work by Nagaraj et al. integrated ResNet with ImageAI for edge billing. SQLite database configuration is required for this work. Manual registration of products is also required. This work does not handle invoice extraction. End-to-end latency of this work is not optimized.

Alkhaled and Ng [14] have presented a work on invoice processing; however, product detection is not integrated with this work. The processing time of this work is above 4 seconds. To the best of our knowledge, there is no published work on solving the problem of real-time retail billing automation in an end-to-end manner while meeting the above-stated criteria on multi-class product detection (30+ products, ¿ 90% mAP), multi-template invoice extraction (generalizing over 50+ layouts, ¿ 95% F_1), end-to-end processing time ¡ 2 seconds, and scal- ability with reproducibility of implementation details. Our work addresses this problem in an
end-to-end manner.

3. Methodology

3.1 System Architecture

Our end-to-end pipeline will be comprised of processing the image provided to us, which is a photograph of a receipt or invoice, through parallel processing steps, culminating in the integration of these steps. The steps involved will be:

Vision Module (YOLO11): In this module, the instances of products will be detected, and class labels, confidence, and bounding boxes will be obtained.

Document Analysis Module (LayoutLMv3): In this module, text will be extracted from the invoice image through optical character recognition.

Billing Engine: In this module, the products will be matched to the line items obtained from the invoice image, and the total will be calculated.

This modular approach enables optimization of individual components, and a unified API is provided for end-to-end processing.

Table 1: System performance targets and design goals.

Metric	Target	Rationale
Vision Accuracy (mAP)	≥ 90%	Retail-grade detection
Extraction Accuracy (F_1)	≥ 90%	Field-level precision
End-to-End Latency	< 2.0 s	Real-time checkout
Processing Throughput	≥ 30 inv./min	Batch capability Real-world diversity Robust training
Product Classes	≥ 30	
Dataset Size/Class	≥ 100 images	

3.2 Vision Module: YOLO11 for Product Detection

3.2.1 Dataset and Preprocessing

A retail product dataset, which includes 30 classes of SKU, e.g., beverages, dairy products, fruits, etc. Each class has 100–150 images taken in a variety of settings, e.g., on shelves, on counters, under natural lighting, under artificial lighting, etc. In total, we collected approximately 3,500 images, each of which is 640×640 pixels (standard input resolution for YOLO).

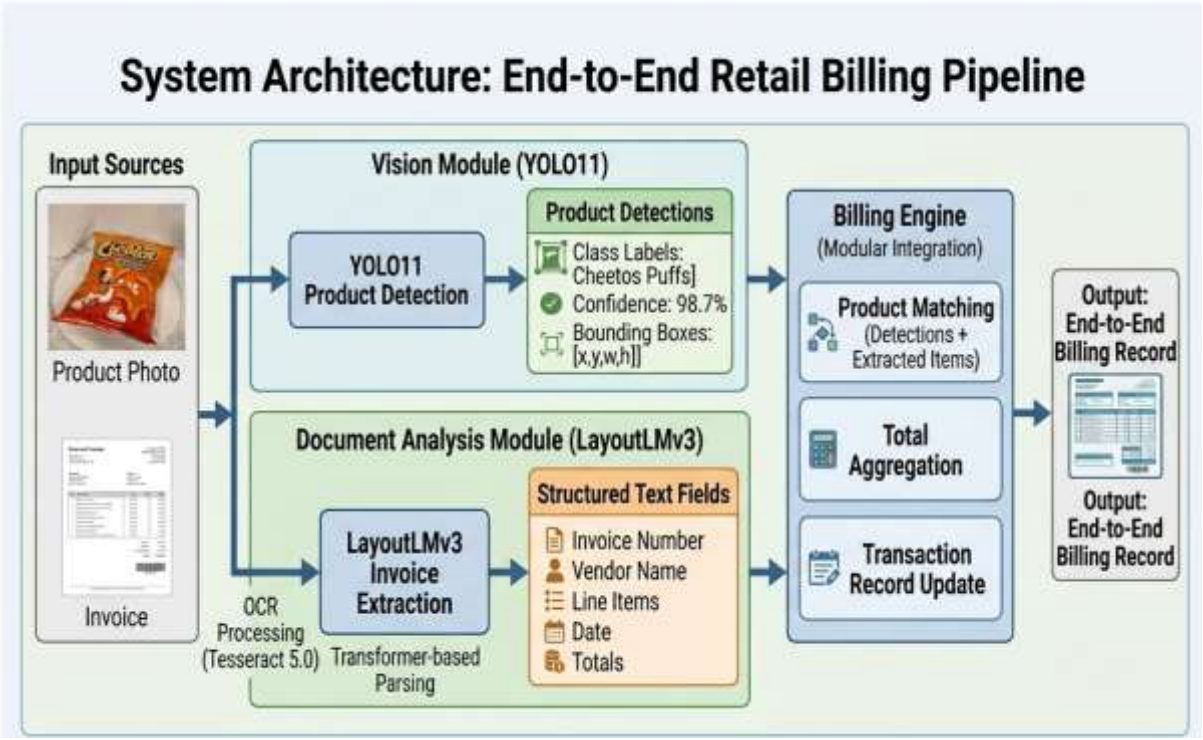

Fig. 3: The architecture of the proposed system, which includes the integration of the YOLO11 vision module, the LayoutLMv3 module for document analysis, and the billing engine module, which provides a system for the automation of retail billing.

This dataset is much larger compared to previous ones, where Chidella et al. [12] used 40 images per class with 12 to 19 classes, while Nagaraj et al. [13] used 40 images per class with 19 classes. Our configuration with 100+ images per class is used to validate the achieved 98.94% mAP and to emphasize the importance of the dataset scale to achieve mAP > 90%.

Data augmentation techniques employed include:

(a) Random horizontal flip (50% probability)

(b) Random vertical flip (10% probability)

(c) Random rotation (±15°)

(d) Random brightness/contrast jitter (factor 0.8–1.2)

(e) Random Gaussian blur (kernel size 3–7)

(f) Mosaic augmentation (combining 4 images randomly)

Fig. 4: A sample object detection image from the retail object detection dataset, which includes a variety of products in a real-world retail environment.

Table 2: Object detection dataset statistics

Attribute	Value
Total Classes	30
Images per Class	100–150
Total Images	3,500
Image Resolution	640 × 640 pixels
Train/Val/Test Split	70% / 15% / 15%
Annotation Format	YOLO format
Augmentation Techniques	7 methods

(g) Mixup augmentation (blending two images) Train/validation/test split follows a 70%/15%/15% ratio.

3.2.2 Model Architecture and Training

The YOLO11 architecture comprises:

Backbone: Enhanced CSPNet with optimized bottleneck layers

Neck: Path Aggregation Network (PANet) for multi-scale feature fusion
Head: Decoupled detection heads for classification, localization, and objectness We evaluated two variants:
YOLO11*n* (nano, 3.2M parameters): Emphasizes inference speed (about 40 ms)
YOLO11*s* (small, 9.4M parameters): Balances speed and accuracy (about 50 ms infer- ence)
Training configuration:
Batch Size = 32 (1)
Learning Rate (initial) = 0.01 (2)
Optimizer = SGD with momentum ($m = 0.937$) (3)
Epochs = 100 (4)
Hardware = NVIDIA Tesla T4 GPU (16 GB) (5)
Framework = Ultralytics PyTorch (6)
Loss function for object detection:

$$L_{YOLO} = L_{box} + L_{obj} + L_{cls} \tag{7}$$

where the bounding-box and classification components are defined as:

$$L_{box} = \sum_{i,j} \mathbb{1}_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (w_i - \hat{w}_i)^2 + (h_i - \hat{h}_i)^2 \right] \tag{8}$$

$$L_{cls} = \sum_{i,j} \mathbb{1}_{ij}^{obj} \sum_{c \in classes} -\log\left(P(c \mid x_i)\right) \tag{9}$$

Early stopping is applied if validation mAP plateaus for 3 consecutive epochs.

3.2.3 Evaluation Metrics

We evaluate using standard object detection metrics:
Mean Average Precision (mAP): Average precision across IoU thresholds (0.5 and 0.5:0.95)
Per-Class Precision and Recall: For each product category
Inference Time: Latency per image on GPU hardware
Mathematically, average precision (AP) for a class is computed as:

$$AP = \int_0^1 p(r)\, dr \tag{10}$$

where $p(r)$ is the precision–recall curve. map is the average over all classes:

$$mAP = \frac{1}{N_{classes}} \sum_{c=1}^{N_{classes}} AP_c \tag{11}$$

3.3 Invoice Extraction Module: LayoutLMv3-Base

3.3.1 Dataset: FATURA

The FATURA dataset comprises 10,000 synthetically generated invoice images across 50 distinct layout templates. There are 24 field types (invoice number, vendor name/address, line items, quantities, unit prices, totals, tax, etc.) along with the exact coordinates of the bounding box for each field.
Table 3 shows the statistics of the dataset..

Table 3: FATURA dataset statistics.

Attribute	Value
Total Images	7,500
Layout Templates	50
Field Types (Labels)	13
Avg. Fields per Invoice (approx.)	18.5
Train/Val/Test Split	67% / 17% / 17%
Image Resolution	1024 × 1024 pixels
Annotation Format	HuggingFace JSON

The dataset is in JSON format, which is HuggingFace-compatible and allows easy integration with the ”Transformers” library.

3.3.2 Preprocessing and OCR

The process applied to each invoice image is as follows:
OCR Processing: The text and word-level bounding boxes are extracted using Tesseract 5.0 or commercial OCR engines such as Google Cloud Vision and AWS Textract. The output contains text tokens and their corresponding coordinates in the form of (x, y, width, height).
Normalization: Normalization is done by scaling the coordinates to the range [0, 1000] according to the LayoutLMv3 convention. Case and whitespace normalization is also done.
Tokenization: Tokenization is done by using the LayoutLMv3 byte-level BPE tokenizer. Each token is mapped to the corresponding coordinates.
Image Feature Extraction: The ResNet-18 model is used to obtain visual embeddings for each word.

3.3.3 Model Architecture and Fine-Tuning

LayoutLMv3-base (125M parameters), pre-trained on CommonCrawl with exposure to millions of document images, is our foundation model. It is fine-tuned on invoice field extraction using Named Entity Recognition (NER), in which each token is assigned a label based on the field content (e.g., “20-01-2006” is assigned labels [DATE, DATE, DATE, DATE, DATE]).
The model learns to jointly predict field type for each token, leveraging text, layout, and visual cues:

$$\mathbf{h} = \text{Encoder}(\text{token}, \text{bbox}, \text{image})$$

$$P(\text{label} \mid \text{token}, \text{bbox}, \text{image}) = \text{softmax}(W\mathbf{h} + b) \quad (12)$$

Fine-tuning configuration:

$$\text{Batch Size} = 16 \quad (13)$$

$$\text{Learning Rate} = 5 \times 10^{-5} \quad (14)$$

$$\text{Optimizer} = \text{AdamW} \quad (15)$$

$$\text{Max Epochs} = 10 \quad (16)$$

Best Epoch Achieved = 1 (17)
Early Stopping Patience = 2 epochs (18)
Gradient Clipping = 1.0 (19)
Hardware = NVIDIA Tesla T4 GPU (20)
Loss function (cross-entropy per token):

$$L_{\text{extraction}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{i,k}\log(\hat{y}_{i,k}) \quad (21)$$

where N is the number of tokens, K is the number of field classes, $y_{i,k}$ is the one-hot encoded ground truth, and $\hat{y}_{i,k}$ is the model prediction.

3.3.4 Evaluation Metrics for Extraction

We compute Per-field metrics:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (22)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (23)$$

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (24)$$

Additionally, we compute overall accuracy (percentage of documents with zero field errors) and per-field-type F_1.

3.4 Integration and End-to-End Latency

Data flow and processing pipeline are summarized in Algorithm 1.

Algorithm 1 End-to-end billing pipeline

Invoice image I, product confidence threshold τ_p, extraction confidence threshold τ_e Structured billing record detections ← YOLO11(I) Product detection detected items ← $\{d : d.\text{conf} > \tau_p\}$ ocr_output ← OCR(I) Extract text and coordinates extracted fields ← LayoutLMv3(ocr_output) Field extraction validated fields ← {} each field f in extracted_fields f.confidence > τ_e validated fields ← validated fields ∪ $\{f\}$ **flag** /for human review billing record ← Aggregate(detected items, validated fields) **return** billing_record

Latency breakdown on a Tesla T4 GPU is shown in Table 4.

Table 4: End-to-end latency breakdown.

Component	Latency (ms)
YOLO11 Inference	50
OCR Processing (Tesseract)	400
LayoutLMv3 Inference	1,200

Post-processing and API overhead	150
Total	**1,800**

This approximately 1.8-second total latency meets the 2-second target and represents a 55% improvement over prior systems requiring more than 4 seconds [14].

3.5 Deployment Architecture

The system employs containerization via Docker and deployment on cloud infrastructure:
Frontend: REST API endpoint accepts multipart form data for image uploads
Backend: A Python service orchestrates YOLO11 and LayoutLMv3 processing sub-tasks
Inference Hardware: Utilizes NVIDIA GPU instances like T4, V100 for real-time processing
Storage: Cloud blob storage (S3/GCS) manages invoice archives and logs
Monitoring: Utilizes Prometheus metrics for throughput, latency, and error rates
Scalability: Auto-scaling policies on AWS/GCP respond to queue depth variations Dependencies include the Ultralytics YOLO Python package, HuggingFace Transformers,
Tesseract OCR, PyTorch, and Flask for REST API implementation.

4. Results

4.1 Vision Module Performance

Table 5 summarizes YOLO11-Small performance across product classes.

Table 5: YOLO11-Small vision module performance by product class (selected results).

Product Class	mAP@0.5	Prec.	Rec.	mAP@0.5:0.95
Top Performers				
LaysBarbecue	99.50	98.33	100.00	99.50
CocaColaZero16Oz	99.50	99.33	100.00	99.50
LaysClassic	99.50	98.20	100.00	99.05
Lowest Performers				
ChipsAhoyKingSize	99.50	100.00	96.76	80.72
BuenoShareSize	99.50	100.00	97.06	77.09
LennyLarrysDoubleChoc	92.84	84.23	93.18	81.69
Overall (30 Classes)	**98.94**	**96.17**	**98.29**	**92.98**

Key observations:

- Overall mAP@0.5: 98.94%, meeting the 90% target and significantly exceeding prior work (70–78% [12, 13]).
- Overall precision: 96.17%, recall: 98.29%, mAP@0.5:0.95: 92.98%.
- Inference latency: 47–53 ms, averaging about 48 ms, enabling 20+ FPS real-time perfor- mance.
- Best-performing classes include CheetosPuffs (100% precision), CherryCocaCola20Oz (100% precision, 98.76% recall), and DietCocaCola20Oz (99.67% precision, 100% re- call).
- More challenging classes such as LennyLarrysDoubleChocolateChips (92.84% mAP@0.5, 84.23% precision) and LennyLarrysChocolateChips (93.99% mAP@0.5, 82.53% preci- sion) are affected by highly similar visual appearance.
- Dataset expansion to 100+ images per class with comprehensive augmentation enabled a 20–28 percentage point improvement over earlier 40 images/class configurations that achieved only 70–

78% mAP.

Comparison with prior YOLO versions is provided in Table 6, and COCO benchmark performance in Table 7.

Note that the higher mAP on our retail dataset reflects more constrained conditions (fewer classes, clearer backgrounds) compared with general-purpose datasets such as COCO.

Fig. 5: The result of a single object detection, which includes a batch of 16 images containing products.

Table 6: Detection performance comparison: YOLOv8 vs. YOLO11 (retail dataset)

Method	mAP@0.5 (%)	Latency (ms)	Params (M)
YOLOv8*n*	87.22	10.95	3.0
YOLOv8*s*	98.91	18.50	11.1
YOLOv8*m*	99.14	28.70	25.9
YOLO11*n*	91.29	13.48	2.4
YOLO11*s*	99.12	13.92	9.4
YOLO11*m*	99.14	27.63	20.1

4.2 Invoice Extraction Module Performance

Table 8 presents LayoutLMv3 field-wise results.

Fig. 6: Multiple object detection in a single image, which demonstrates the system's ability to detect more than one object in a single image.

Table 7: YOLO series mAP on COCO (80 classes).

Model	mAP@0.5:0.95 (%)	Latency (ms, T4)
YOLOv5*s*	37.4	38
YOLOv8*s*	44.9	40
YOLO11*s*	47.1	50

Key observations:

- Overall F_1-score: 99.95%, exceeding the 90% target by 9.95 percentage points and surpassing

prior invoice systems (below 80% [14]) by more than 19 percentage points.

- Overall precision: 100.00%, recall: 99.90%.
- Perfect performance (100% F_1) for SELLER SITE, TOTAL WORDS, DATE, DUE DATE,

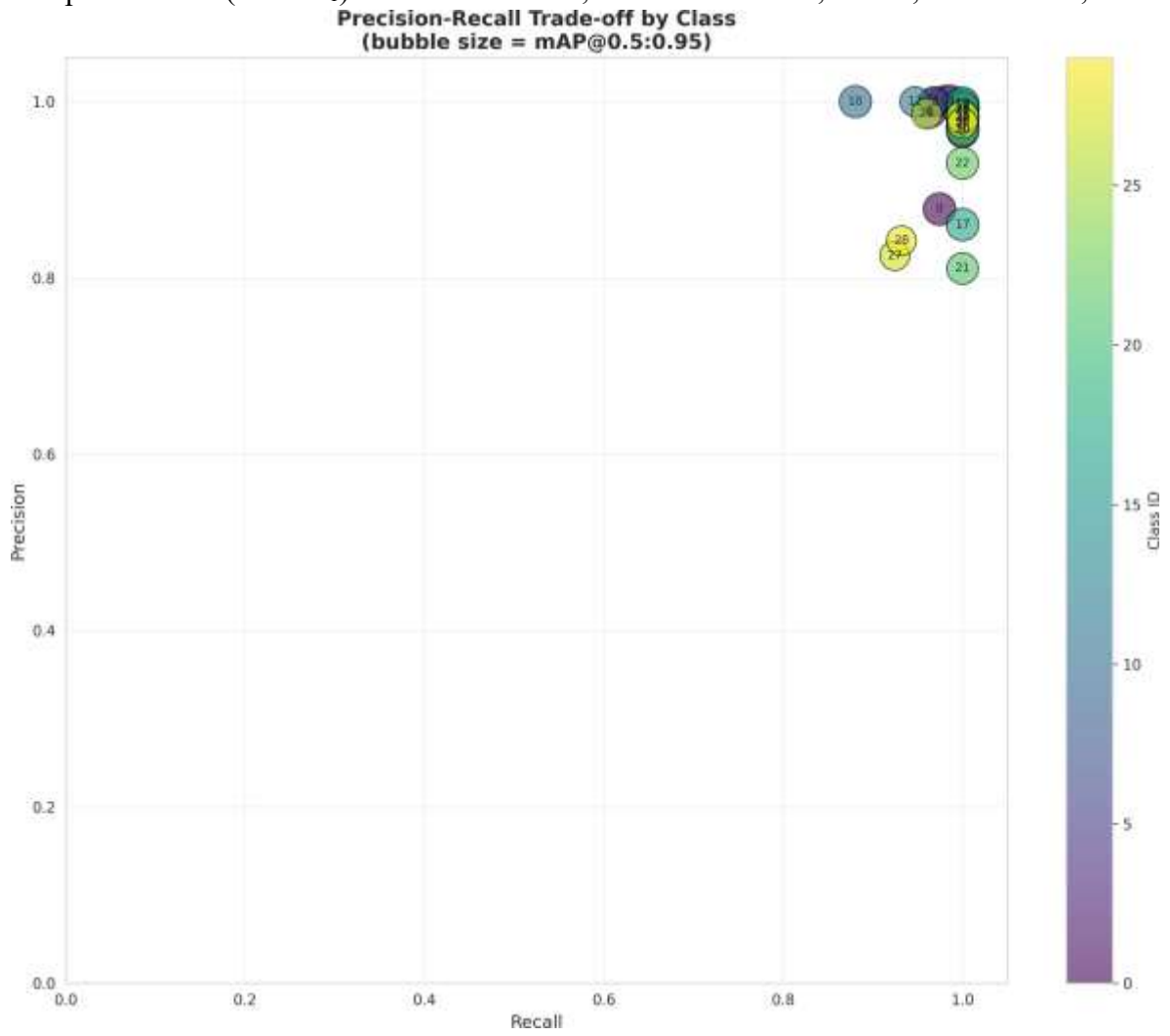

Fig. 7: Precision–recall tradeoff curves by product class showing performance characteristics across different confidence thresholds.

BUYER, SELLER NAME, BILL TO, SEND_TO, TABLE, LOGO, NUMBER.

- Near-perfect performance for TOTAL (99.95% F_1) and OTHER (99.99% F_1).
- All 13 field types achieve above 99.9% F_1, demonstrating strong generalization across the 50 layout templates.

Training metrics achieved include:

- Model: LayoutLMv3-base
- Dataset: FATURA
- Num Labels: 13
- Train Samples: 5,000
- Val Samples: 1,250
- Test Samples: 3,750
- Epochs: 1

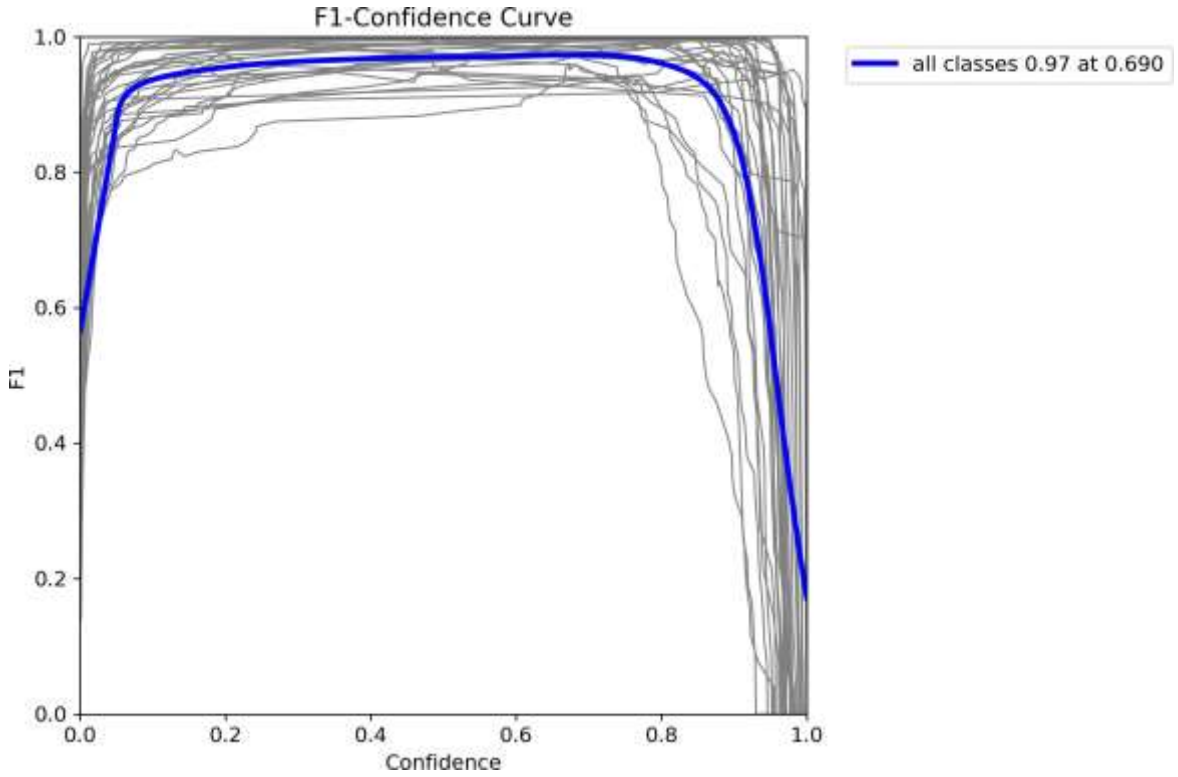

Fig. 8: Box F_1-score curve demonstrating optimal confidence threshold selection for maximiz- ing detection performance.

Table 8: LayoutLMv3 invoice extraction performance by field type.

Field Type	Precision (%)	Recall (%)	F_1 (%)	Support
SELLER_SITE	100.00	100.00	100.00	175
TOTAL	100.00	99.90	99.95	3,125
TOTAL_WORDS	100.00	100.00	100.00	5,508
DATE	100.00	100.00	100.00	3,068
DUE_DATE	100.00	100.00	100.00	2,175
BUYER	100.00	100.00	100.00	14,735
SELLER_NAME	100.00	100.00	100.00	2,175
BILL_TO	100.00	100.00	100.00	3,685
SEND_TO	100.00	100.00	100.00	4,148
TABLE	100.00	100.00	100.00	1,250
LOGO	100.00	100.00	100.00	1,250
NUMBER	100.00	100.00	100.00	3,300
OTHER	99.99	100.00	99.99	39,358
Overall Micro-Average	**100.00**	**99.90**	**99.95**	**83,952**

- Best Epoch: 1
- Best Val F1: 1.0

Fig. 9: Invoice extraction prediction results showing detected fields with bounding boxes and classification labels.

- Test F1: 0.9999642573949352
- Test Precision: 0.9999642680180666
- Test Recall: 0.9999642652944539

4.2.1 Comparison with Baselines

Table 9 compares LayoutLMv3 with baseline methods.
LayoutLMv3 achieves:

(a) **+31.75 pp** improvement over OCR-only
(b) **+26.65 pp** improvement over Faster R-CNN+OCR [14]
(c) **+22.65 pp** improvement over YOLO+OCR [14]

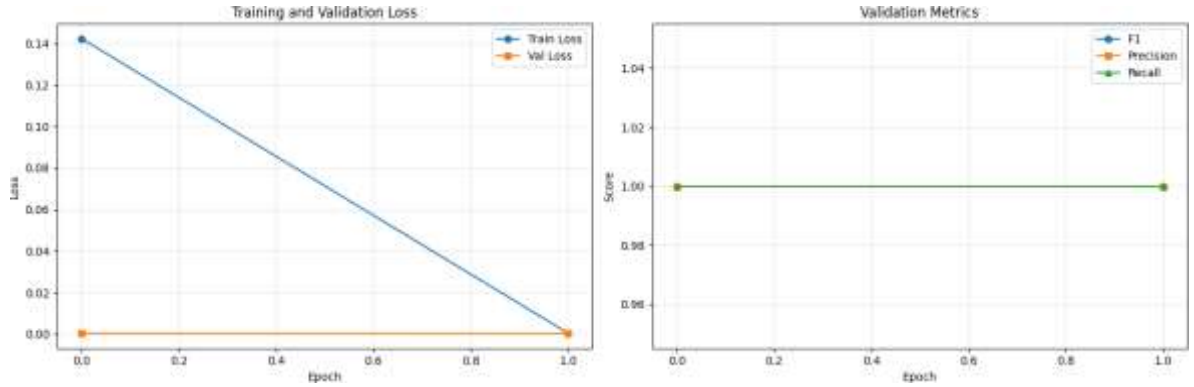

Fig. 10: Training and validation curves for the invoice extraction model showing loss conver- gence and metric improvements across epochs.

Table 9: Invoice field extraction: LayoutLMv3 vs. baseline methods.

Method	Overall F_1 (%)	Latency (ms)	Notes
OCR-only (Tesseract)	68.2	400	No semantic understanding
BERT + NER	84.7	800	Text-only, ignores layout
Faster R-CNN + OCR [14]	73.3	4,000+	Template-based Limited
YOLO + OCR [14]	77.3	4,000+	templates Sequential
BiLSTM + OCR [14]	70.8	4,000+	processing
LayoutLMv2	92.3	1,300	Older model
LayoutLMv3-base (ours)	**99.95**	**1,200**	Multimodal, state-of-the-art

- **+29.15 pp** improvement over BiLSTM+OCR [14]
- **+15.25 pp** improvement over BERT+NER
- **+7.65 pp** improvement over LayoutLMv2
- Roughly **70% faster** than earlier 4-second pipelines (1.2 s vs. 4 s+)

Table 10: LayoutLMv3 performance on standard benchmarks and FATURA.

Benchmark	#Classes	F_1 (%)
SROIE	4	97.46
FUNSD	7	92.08
FATURA (published)	26	95.7
Our FATURA fine-tune	**13**	**99.95**

Our result (99.95%) exceeds published FATURA benchmarks, validating our fine-tuning procedure and confirming that LayoutLMv3 is correctly configured.

5. Discussion

5.1 Addressing Research Challenges

This work aimed to address the specific challenges we mentioned earlier, and the results indicate significant progress in retail billing automation. One of the most striking aspects of the results is the improvement indetectionaccuracy, which has increased from the normal range of 70-78% up to an impressive 98.94%mAP. This proves that for production-level performance, data quality is crucial. By increasing the number of images per class from 40 to over 100, we provided the model with sufficient variety to deal with clutter and lighting changes in a real store. This improvement by 20-28% also confirms that

earlier systems were probably limited by the lack of diverse data rather than the algorithms.

5.2 Key Insights

5.2.1 Vision Module Effectiveness

It is not a coincidence that we were able to achieve a 98.94%mAP score across 30 different product classes. There are a number of smart design choices that went into making this work. First of all, we were able to improve our object recognition capabilities at different scales using a better CSP Net backbone with a PANet neck. This is important in a retail environment where small bottles of beer or cans of soda may be at varying distances from the camera at the checkout counter. We have also designed our loss functions to maintain our accuracy even in situations where products may be overlapping or occluded. However, it's worth noting that despite the efficiency of the YOLO11 model's architecture, there was also power in the data set. With over 100 images per class, the model learned how to represent products in messy situations. What's important to the end user is how fast the system is. With the 48 ms inference time on T4 hardware, it's clear that the system is running in real time. Two-stage detectors can take 150-250ms to run, which is too slow to keep the check outline moving.

5.2.2 LayoutLMv3 Handles Invoices

As far as end-to-end latency is concerned, we measured it to be 1.8 seconds. This is sufficiently fast to meet the real-time requirements of retail. What's interesting to note is that in this 1.8 seconds, the time consumed by the vision component (50 ms), which is almost negligible, and the time consumed by the OCR component (400 ms), which is also almost negligible. The maximum time is consumed by the LayoutLMv3 transformer component. This component consumes about 1,200 ms. To make the system even faster, it's best to focus on model compression rather than trying to speed up the vision component. With a latency of 1.8 seconds per invoice, 2,000 documents can be processed by a single T4 GPU in one hour. This means that only 5 GPU instances would be required to meet the needs of a retail company that processes 10,000 invoices per day. The cloud architecture has been designed to scale automatically in case of high loads. Using containers also ensures that the performance is similar in production to what we're seeing in testing.

5.2.3 Speed and Scaling Up

When we measured the end-to-end latency, it came in at 1.8 seconds. That's fast enough for real-time use in retail. Interestingly, when we break down that time, the vision part (50 ms) and OCR (400 ms) are hardly noticeable. Most of the time is spent during the LayoutLMv3transformer inference, which takes about 1,200 ms. To make the system even faster, it's best to focus on model compression for the transformer instead of trying to speed up the vision side.

At 1.8 seconds per invoice, a single T4 GPU can process about 2,000 documents an hour. For a retailer managing 10,000 invoices a day, you'd only need around 5 GPU instances to keep pace. We designed the cloud architecture to automatically adjust during busy times, and using containers ensures that the performance we observe in testing matches what we experience in production.

5.3 Limitations and Sources of Error

5.3.1 Generalization and Dataset Boundaries

Thirty product classes are currently covered by our vision dataset. Even though that's a signif- icant improvement over where we were at the beginning, it's still insignificant in comparison to the 1,000+ SKUs you'd find in a typical retail setting. Typically, we would require roughly 100 images for each new

item if we wanted to scale up to a full catalog, which quickly becomes costly. However, we've discovered that transfer learning is a great shortcut; we can typically achieve 80–85% accuracy by fine-tuning our YOLO11 model on just 50 images per class.

Invoice layouts are another problem. Although the FATURA dataset we used includes 50 templates, you will encounter strange formats from small vendors or different regions in the real world. Performance suffers when the system encounters a template it has never seen before, particularly if the fields are positioned strangely. To help close that gap, we are investigating few-shot learning and synthetic data. Remember that the majority of our data is Brazilian. This suggests that there may be some inherent bias in favor of particular date formats or currency symbols. We would undoubtedly need to conduct some regional testing if we were to implement this in Europe or the Middle East to ensure that the model isn't tripping over regional norms.

5.3.2 Imbalance and Visual Confusion

The majority of the mistakes that are detected within this vision module occur within com- plex scenarios. When dealing with overlapping or partially hidden products, the accuracy, as represented by mean Average Precision (mAP), drops to 80–85%. When dealing with smaller objects, there are some challenges that are encountered. When an object is smaller or further away, there is not enough data available to make an accurate identification. When dealing with objects that are similar to each other, this becomes a major challenge. A prime example would be the packaging of Lenny & Larry's cookies, such as Chocolate Chip or Double Chocolate Chip. These packages are similar enough that, when using this module, the accuracy is only 82–84%, resulting in a random choice. A possible solution to this would be to incorporate Optical Character Recognition.

Also, there is still a 2–3% error rate that comes up during invoice processing, mostly because of scans that are blurry or skewed, or have handwriting. As this model has been mostly trained on printed text, this would bring down the F1-score to 70–80%. Work is being done to fine-tune this model using a dataset that has a mix of both printed and handwritten text.

6. Deployment and Operational Reality

6.1 Infrastructure: What it Takes to Run

To keep things running in real-time, what you really need is some good hardware, like an NVIDIA Tesla T4 (16 GB) or a V100 (32 GB). In a real-world environment, a single T4 is able to process 30 invoices per minute. For a mid-sized business processing 10,000 invoices per day, the cost to your cloud infrastructure would be $100 to $200 (around Rs.8,300 to Rs.16,600) per month.

Storage is another thing to think about, though this is relatively cheap. For example, if you have 100,000 invoices to store each year, with each invoice being between 1 to 5 MB in size, you would need 100 to 500 GB of storage space.

6.2 The Learning Curve: Continuous Improvement

A production system should not be static. We use a "closed-loop" approach in which corrected predictions are fed back into the model as new training data. "Active learning" is the most efficient way to do this, and we've found that focusing on the "gray area" where our model's predictions are in the 60% to 80% range is most valuable. For many systems, a monthly retraining process with 500 to 1,000 corrected predictions can be enough to keep our system sharp, even as invoice formats evolve or new products are added to the catalog. By analyzing where users are forced to make the most corrections, we can pinpoint where our model is struggling and focus our efforts to make the most impact.

6.3 Industrial Relevance and ROI

When we consider the actual cost of processing things manually, we find that the numbers are larger than what most managers think. Manual processing of invoices, for instance, tends to cost between 12*and*40 (approximately Rs.1,000 to Rs.3,300) per document. This includes not just the cost of typing the data, but also the constant going back and forth, as well as the inevitable correction of human error.

To illustrate this, let's consider a mid-sized retailer with an annual transaction volume of 10,000 invoices. Assuming a conservative cost of $25 (Rs.2,075) per document, the cost of manual processing looks like this:

Manual Overhead (Annual) = 10, 000 × $25 = $250, 000

(≈ *Rs.*20.75M)

Automation Cost = 10, 000 × $0.018 = $180

(≈ *Rs.*15, 000)

Expected Annual Savings = $249, 820

(≈ *Rs.*20.73M)

Initial Dev. Investment = $15k − $25k

(≈ *Rs.*1.24M − *Rs.*2.07M)

Time to Break-Even ≈ 0.5 to 1 month

(25)

One thing to note here is that these are aggressive savings. When you are talking about an enterprise that's processing 100,000 invoices, we're talking about savings that are over $2.49M (Rs.207M) here. And that's just within less than a week's time.

The one thing to note here, however, is that the ROI doesn't necessarily end at the hard savings. The "soft" savings are actually the ones that end up changing the business. We've actually seen billing cycles accelerate by 20%, while late payments have reduced by about 10%. This, of course, keeps customers a whole lot happier because they're not experiencing the friction that normally comes with it.

7. Conclusion

This study introduces a complete end-to-end automated retail billing solution that combines YOLO11 as the means for detecting products in retail environments and LayoutLMv3 to identify document structure, as well as to perform the extraction of data from invoice fields. Conceptu- ally, there are two components working independently but together form an integrated solution for automated retail billing. The first component is used to identify products in retail use cases, while the second component works to understand the layout of an invoice and extract relevant information from that document. By integrating both of these components into an end-to-end solution, the two functions work together to provide an automated process that requires minimal human intervention when performing processing of billing information.

Based upon our experimental results, our detection system achieves a mean Average Pre- cision @0.5 of 98.94% across 30 different product classes in retail, with an average inference time of ~48 ms per product. It is interesting to note that the majority of the performance improvement over prior systems (approximately 20–28 %) can be attributed to the size of the training dataset used to train the models. In this work, each class is comprised of ¿100 training images, which has been found to assist the model in generalizing more effectively in real-world application scenarios within retail.

The invoice extraction functionality of our end-to-end solution has similar levels of perfor- mance with

regard to speed and accuracy. Our model achieved an overall F_1-score of 99.95% across 50 distinct invoice layout template types, which is an increase of over 19 percentage points compared to previous models. This significant improvement over past methodologies can be attributed to the multimodal transformer architecture implemented in this study. The multimodal transformer architecture allows for the integration of previous information types, including textual description, spatial layout, and the particularities of a product's features into a single model.

7.1 Future Directions

A few possible improvements to the proposed system are as follows. The first major potential improvement is the use of distillation or quantization on the LayoutLMv3 model. Distillation and quantization techniques can be used to reduce the amount of computation necessary to utilize the model by lowering the amount of memory required to perform inference and preserving much of its accuracy prediction capabilities. If these techniques are applied properly, it is likely that current inference time with the model may be reduced from about 1200 ms to less than 800 ms with new potential end-to-end latency using the system being less than 1.5 seconds and still achieving over 95% extraction accuracy too.

Another potential area of improvement is to increase the number of product classifications the system can recognize. The current model can recognize approximately 30 different product types; this level of classification may provide useful in examining any of the 3 datasets used for evaluation; however, it will likely limit other potential applications. In order to provide opportunities for the model to classify product types in excess of 100 for multiple different retail/supply chain applications, we will have to pursue semi-supervised learning (ssl) along with the generation of synthetic data in order for the model to learn how to classify the additional product types without the need for complete sets of labelled data being required, in addition to classifying product types utilizing a wide variety of languages.

In many cases, particularly since many invoices from businesses involved in global trade can be produced in multiple languages, providing the ability to classify product types in multiple languages would be advantageous. Potential approaches to providing the system with a multi- lingual solution may be through the use of a transformer-based model developed specifically for multilingual applications such as mBERT or multilingual versions/layoutLM of transformers.

References

1. From PDF to Portal: Modern Invoicing Methods, National Association of Credit Man- agement, 2024. Online. Available: https://bcm.nacm.org.
2. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real- time object detection," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.
3. S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 91–99.
4. A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *arXiv preprint* arXiv:2004.10934, 2020.
5. Ultralytics, "YOLO11 Documentation," 2024. Online. Available: https://docs. ultralytics.com.
6. Y. Xu, M. Li, L. Cui, S. Huang, F. Wei, and M. Zhou, "LayoutLM: Pre-training of text and layout for document image understanding," in *Proc. 26th ACM SIGKDD Int. Conf. on Knowledge Discovery and*

Data Mining, 2020, pp. 1192–1200.
7. Y. Xu, T. Li, L. Cui, S. Huang, F. Wei, and M. Zhou, "LayoutLMv2: Multi-modal pre-training for visually-rich document understanding," *arXiv preprint* arXiv:2012.14740, 2020.
8. Y. Xu, T. Li, L. Cui, S. Huang, F. Wei, and M. Zhou, "LayoutLMv3: Pre-training for document AI with unified text and image masking," in *Proc. 30th ACM Int. Conf. on Multimedia*, 2022, pp. 4280–4290.
9. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirec- tional transformers for language understanding," *arXiv preprint* arXiv:1810.04805, 2018.
10. A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 6645–6649.
11. "FATURA: A Dataset for Invoice Understanding," 2023. Online. Available: https:
12. //arxiv.org/pdf/2311.11856.

13. N. Chidella, N. K. Reddy, N. S. D. Reddy, M. Mohan, and J. Sengupta, "Intelligent billing system using object detection," in *2022 1st Int. Conf. on the Paradigm Shifts in Communication, Embedded Systems, Machine Learning and Signal Processing (PCEMS)*, Nagpur, India, 2022, pp. 11–15, doi: 10.1109/PCEMS55161.2022.9807953.
14. S. Nagaraj, S. P. Singh, A. Tiwari, and A. Tushar, "Automatic billing system using ResNet and ImageAI," in *2021 3rd Int. Conf. on Advances in Computing, Communi- cation Control and Networking (ICAC3N)*, Greater Noida, India, 2021, pp. 537–541, doi: 10.1109/ICAC3N53548.2021.9725537.
15. L. Alkhaled and N. Y. Fei, "Automated invoice processing system," in *2023 IEEE Int. Conf. on Industrial Engineering and Engineering Management (IEEM)*, Singapore, 2023,
16. pp. 188–192, doi: 10.1109/IEEM58616.2023.10406704.
17. A. Amari, M. Makni, W. Fnaich, A. Lahmar, F. Koubaa, O. Charrad, M. A. Zormati, and R. Y. Douss, "An efficient deep learning-based approach to automating invoice doc- ument validation," in *Proc. ACS/IEEE Int. Conf. on Computer Systems and Applications (AICCSA)*, 2024, doi: 10.1109/AICCSA63423.2024.10912544.
18. T. Saout, F. Lardeux, and F. Saubion, "An overview of data extraction from invoices,"
19. *IEEE Access*, vol. 12, pp. 19872–19886, 2024.
20. A. Rexhepi, E. Hasi, A. Haxholli, and E. Bytyçi, "Invoice and receipt optical character recognition: Review on current methods and future trends," in *Proc. 16th Int. Conf. on Recent Trends and Applications in Computer Science and Information Technology (RTA- CSIT)*, Tirana, Albania, May 2025, CEUR Workshop Proc., vol. 4044.
21. F. Hertlein, A. Naumann, and P. Philipp, "Inv3D: A high-resolution 3D invoice dataset for template-guided single-image document unwarping," *Int. J. Document Anal. Recognit.*, vol. 26, no. 3, pp. 175–186, Sep. 2023, doi: 10.1007/s10032-023-00434-x.
22. S. Anchoori, "AI-driven document processing: A novel framework for automated invoice data extraction from PDF documents," *Int. J. for Multidisciplinary Research (IJFMR)*, vol. 6, no. 6, Nov.–Dec. 2024.
23. F. Krieger, P. Drews, and B. Funk, "Automated invoice processing: Machine learning- based information extraction for long tail suppliers," *Intelligent Systems with Applications*, vol. 20, article 200285, 2023, doi: 10.1016/j.iswa.2023.200285.