**Advances in Artificial Intelligence Research (AAIR)**
**ISSN 2757-7422**
**www.dergipark.com/aair/**

**Vol. 4 (No. 1), pp. 10-17, 2024**
**doi: 10.54569/aair.1401234**
**Published online: Aug 30, 2024**

# Efficient and Accurate Date Extraction from Invoices: A Comprehensive Three-Step Methodology Integrating Custom Object Detection, OCR, and Refined Regular Expressions

Mehmet Hilmi Emel [1],*, Murat Terzioğlu [1], Ramazan Özkan [2]

[1] Egaranti Teknoloji A.Ş., Istanbul, Turkey;
[2] Haliç University, Faculty of Engineering, Department of Computer Engineering, Istanbul, Turkey;

**Abstract**

In the realm of contemporary document processing, the challenge of extracting crucial information from diverse invoices necessitates innovative solutions. This article presents a comprehensive three-step methodology to address the complexity of date extraction from invoices. Leveraging LabelStudio, Python, and OpenCV, we constitute a dataset and train a custom object detection model using Ultralytics YOLOv8. Optical Character Recognition (OCR) provides us to convert the image data to string data that is enable to be processed. Regular expressions refine the extracted text, achieving precise date formats. The developed system significantly enhance the time efficiency, marking a noteworthy advancement in date extraction from invoices.

***Keywords:*** *Custom Object Detection; OCR; Invoice Processing; YOLOv8.*

## 1. Introduction

In the dynamic landscape of contemporary document processing, the challenge of extracting crucial information from a multitude of invoices underscores the necessity for innovative and adaptable solutions. Invoices, being pivotal financial documents, exhibit a remarkable diversity in terms of structure, formatting, and content, rendering the creation of a one-size-fits-all algorithm for date extraction a formidable task. Conventional rule-based methodologies, while effective in specific contexts, struggle to accommodate the intricate variations presented by invoices globally. Recognizing this complexity, our research endeavors to overcome these limitations by proposing a comprehensive three-step methodology: Custom Object Detection, OCR, and Regular Expression. To annotate our dataset and train our object detection model, we harness the capabilities of LabelStudio [1], an efficient data labeling tool. This tool, coupled with the versatility of Python and OpenCV libraries, empowers us to meticulously delineate regions of interest within our invoice images. Moreover, to streamline our object detection endeavors, we use the Ultralytics YOLOv8 [2] framework, enhancing the efficiency and accuracy of our model training process. The usage of these technologies allows us to craft a dataset that not only captures the nuances of invoice layouts but also serves as a robust foundation for training an object detection model. Subsequently, we mention the transformative realm of Optical Character Recognition (OCR) using the Paddle OCR [3], converting the visually encoded text within identified regions into machine-readable formats. As we continue to study on this research, we produce regular expressions to refine and pinpoint the date formats within the extracted text. In presenting our comprehensive approach, we not only showcase the technical intricacies but also underscore the practical implications and advancements in the realm of document processing, marking a significant step forward in the quest for efficiency and accuracy in date extraction from diverse invoices.

## 2. Related Works

Satav et al. [4]: Addressing the challenges of invoice management in small enterprises, their web-based application offers a streamlined solution. Recognizing the prevalent use of traditional manual systems, their focus is on accessibility and data extraction. The proposed system employs OpenCV for preprocessing, OCR for data extraction, and RegEx for refining the results. Specifically tailored for small businesses, this application aims to enhance efficiency and accessibility in managing invoice data.

Kosiba et al. [5]: A method for analyzing invoice document structure is proposed to extract essential information. The approach combines textual and graphical processing, examining line and line intersection features, and searching for keywords like item number and total. Valid keyword search regions are identified using connected-component analysis before OCR. Results from keyword search and line analysis are combined to extract relevant data. This analysis will be integrated into a larger invoice interpretation system currently in development.

*Corresponding author
*E-mail address:* mehmethilmi81@gmail.com
Received: 6/Dec/2023; Accepted: 30/Aug/2024.

Sidhwa et al. [6]: This study aims to develop a methodology for extracting data from everyday printed bills and invoices, which can be utilized for tasks like machine learning and statistical analysis. The focus is on extracting key information such as the final bill amount, itinerary, and date from these documents, which contain valuable insights into user preferences. Optical Character Recognition (OCR) technology, specifically the Tesseract OCR engine [7],[8], is employed. Initially, OpenCV is utilized for bill or invoice detection and noise reduction in the image. The Tesseract OCR engine, incorporating Text Segmentation, is then applied to extract written text in various fonts and languages. The methodology demonstrates high accuracy when tested on diverse input images of bills and invoices.

Karsligil et al. [9]: This conference paper introduces a novel texture-based method for identifying text areas in complex document images. The approach utilizes Gabor filter, inspired by the multi-channel filtering approach of the Human Visual System (HVS), to generate an energy map of the document. Text areas, presumed to be rich in high-frequency components, are extracted as connected components through binarization of the energy map employing Otsu's adaptive threshold method. Initial removal of non-text components like pictures and lines is achieved through Gabor filtering. A distinctive aspect of the method involves further elimination of remaining non-text components using character component interval tracing. This two-stage elimination process enhances the accuracy of detecting text areas in various types of digital documents.

Zhang et al. [10]: This paper introduces a method for extracting text from historical Tibetan document images, treating the task as a problem of detecting and locating text areas. The process involves preprocessing the images to address illumination imbalances, tilt, and noise, resulting in a binary image. The regions of interest in historical Tibetan documents are categorized using connected components and grid-based filtering. The remaining grids are utilized to compute vertical and horizontal projections, enabling the detection of the approximate text area location. The final step involves accurate extraction of the text area by correcting the bounding box of the approximate text area. Experimental results on a historical Tibetan document image dataset illustrate the effectiveness of the proposed method.

Dachengwang et al. [11]: Analyzing images of filled-out forms is a significant problem with practical applications in office automation and theoretical challenges for document image analysis. The method described in this work focuses on extracting both typed and handwritten text from scanned and digitized images of completed forms. Without relying on prior knowledge of form structure, the approach decomposes a filled-out form into three fundamental components: boxes, line segments, and the remainder (including handwritten and typed characters, words, and logos). The input binary image undergoes segmentation into small and large connected components. Complex boxes are further broken down into elementary regions using key-point analysis. To separate handwritten and machine-printed text that intersects guide lines and boxes, lines are removed. Characters fragmented by line removal are then reconnected using a character patching method. The effectiveness of the method is demonstrated through experimental results with filled-out forms from various domains, including insurance, banking, tax, retail, and postal sectors.

## 3. Methodology

We proceed through three steps for this purpose: Custom Object Detection, OCR, and RegEx rules. Initially, we employ our custom object detection model to identify the area containing historical information within our proprietary dataset. Subsequently, within the identified region, we employ Optical Character Recognition (OCR) to transform text from a photographic format to a textual format. In the third step, we utilize regular expressions to extract the desired date format from the textual data. The flowchart depicting the methodology employed in this study is illustrated in **Figure 1**.

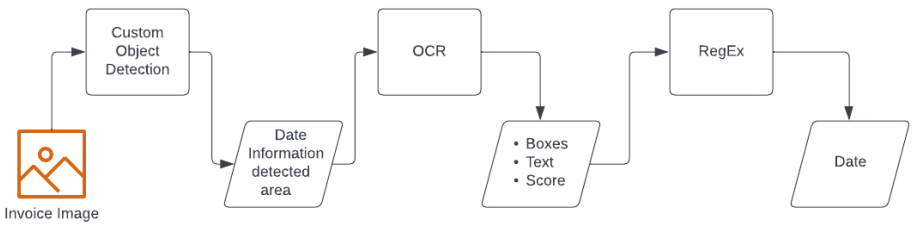


**Figure 1.** *The Flowchart of the methodology*

There were two main reasons for developing our object detection model. The first reason was that when applying OCR technology directly to the entire invoice image, the scanning process of the entire invoice by OCR was time-consuming. Therefore, when turning our developed system into a product, using OCR directly on the entire invoice would make the system inefficient in terms of time. By utilizing our Custom Object Detection model, we could identify a smaller region within the invoice image, specifically containing the date information, and converting the text within that area into string format using OCR proved to be faster. This

approach enhanced the efficiency of the system in terms of processing time. The second reason was evident in **Figure 2**, where the sections containing date information on invoices varied. Developing a rule-based algorithm seemed insufficient in this regard, as it would require examining all invoice images and determining a rule for each invoice example. This approach would be impractical due to the diversity in the location of date information across different invoices.

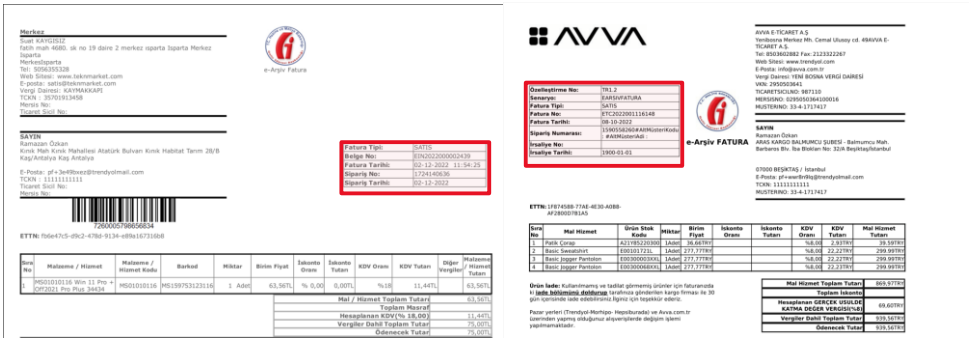

**Figure 2.** *Location of date information in different invoice samples*

### 3.1 Custom Object Detection Model

One of the primary reasons for incorporating object detection into our workflow is the extended length of date information obtained when applying OCR alone. As OCR is required to scan the entire invoice image, locating the date information naturally incurred a longer processing time. By employing object detection to specifically scan the region containing the date information, the response time has been significantly reduced. We conducted the development of the object detection in three stages. The first stage involved preparing the dataset, the second stage focused on model development, and the third stage centered around model evaluation.

#### 3.1.1 Preparing Dataset

To construct our dataset, we initiated the data labeling process employing the Label Studio tool. In the process of dataset creation, we judiciously labeled the region containing date information with the tag "invoice_data." Subsequently, to encompass the details of the individuals acquiring the product in our subsequent endeavors, we introduced the label "personal_data" (The area highlighted within the red rectangle in **Figure 3**). In our pursuit of enhancing the reliability and efficacy of the Object Detection model under development, deliberate efforts were made to diversify our dataset. This involved the augmentation of data variety by engaging in the collection of product acquisition information from diverse users representing various companies. As a result of these initiatives, we amassed a total of 161 invoice examples. The foundational step in our dataset construction involved the utilization of the Label Studio tool, a pivotal instrument for data labeling processes. Specifically, we employed the "invoice_data" (The area highlighted within the blue rectangle in **Figure 3**) label to demarcate the regions within our dataset housing pertinent date information. Subsequent refinement of our project objectives necessitated the incorporation of the "personal_data" label to address the additional dimension of identifying and processing the information pertaining to individuals involved in the product acquisition process.

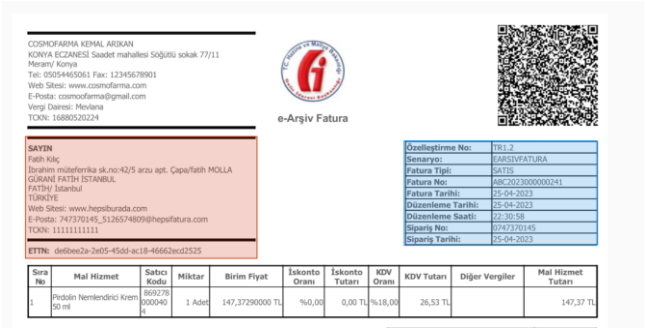

**Figure 3.** *Labelling the invoices*

### 3.1.2 Creating Custom Object Detection

For the construction of our model, we opted to utilize the Ultralytics YOLOv8 model. The Ultralytics YOLOv8 represents a state-of-the-art model within the field of object detection and tracking, instance segmentation, image classification, and pose estimation. This cutting-edge iteration builds upon the achievements of its YOLO predecessors, incorporating novel features and enhancements to augment both performance and adaptability. YOLOv8 is engineered to deliver high-speed processing, precision, and user-friendly operation, rendering it a highly commendable option for diverse applications in computer vision tasks. The model's advancements are poised to contribute significantly to the evolution of methodologies employed in tasks such as object recognition, tracking, and image analysis. The architecture of the YOLOv8 object detection model is illustrated in **Figure 4**.

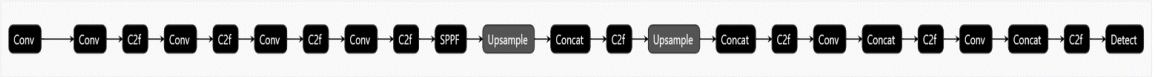


**Figure 4.** *The architecture of the custom object detection model.*

We conducted experiments on this architecture using various optimizers and learning rate values to determine the optimal performance of our model.

### 3.1.3 Evaluating the Object Detection Model

To attain optimal results in our custom object detection model, we conducted a comprehensive study by systematically exploring various epoch values, learning rate parameters, and optimizer configurations. Our objective was to identify the most effective combination of these hyperparameters to enhance the model's performance. Through a series of experiments, we calculated the Mean Average Precision at 50 (mAP50) for each model, aiming to ascertain the impact of different settings on the precision and recall of the object detection framework. This iterative exploration allowed us to fine-tune our model by considering the interplay between epoch duration, learning rate adjustments, and optimizer choices, providing valuable insights into the nuanced dynamics of our custom object detection architecture. Deep learning models heavily rely on optimization algorithms to enhance their training process and achieve optimal performance. In this study, we conducted a detailed examination of various optimization algorithms, including Stochastic Gradient Descent (SGD), Adam, Adamax, AdamW, Nadam, and RAdam, across different learning rates (lr) and epochs. The performance of these algorithms was evaluated based on their impact on the accuracy of a deep learning model. **Table 1** illustrates the performance of the optimization algorithms.

**Table 1.** *The experimental results obtained with different epochs, learning rates, and optimizer values*

| Epoch=10 | SGD | Adam | Adamax | AdamW | NAdam | RAdam |
|---|---|---|---|---|---|---|
| Lr=0.1 | 0.00075 | 0.00963 | 0.00682 | 0.00178 | 0.00021 | 0.00127 |
| Lr=0.01 | 0.93539 | 0.00293 | 0.81918 | 0.0027 | 4.00E-05 | 0.67992 |
| Lr=0.001 | 0.68521 | 0.96554 | 0.98308 | 0.96151 | 0.89302 | 0.90717 |
| Epoch=20 | SGD | Adam | Adamax | AdamW | NAdam | RAdam |
| Lr=0.1 | 0.06638 | 0.00164 | 0.03064 | 0.0087 | 0.00147 | 0.00082 |
| Lr=0.01 | 0.98723 | 0.8449 | 0.97058 | 0.26271 | 0.00212 | 0.97742 |
| Lr=0.001 | 0.93794 | 0.98947 | 0.99483 | 0.99106 | 0.98192 | 0.95569 |
| Epoch=50 | SGD | Adam | Adamax | AdamW | NAdam | RAdam |
| Lr=0.1 | 0.98467 | 0.13369 | 0.40084 | 0.15401 | 0.00322 | 0.9861 |
| Lr=0.01 | 0.995 | 0.98432 | 0.99226 | 0.98604 | 0.70449 | 0.99484 |
| Lr=0.001 | 0.98942 | 0.995 | 0.99492 | 0.995 | 0.99145 | 0.995 |

For the Stochastic Gradient Descent (SGD) algorithm, higher learning rates (e.g., lr=0.1) initially resulted in a significant error rate. Lower learning rates (e.g., lr=0.001) produced a more stable performance, albeit requiring an extended training period. Notably, with increased epochs, SGD demonstrated convergence, indicating improved accuracy, particularly at lower learning rates.

The Adam optimization algorithm, incorporating adaptive momentum, exhibited superior performance at lower learning rates (e.g., lr=0.001), while displaying a potential decrease in accuracy at higher rates (e.g., lr=0.1). Longer training periods (epochs=20 and epochs=50) generally favored Adam, indicating a progressive improvement in accuracy over time. Adamax, a variant of Adam that considers larger gradient

values, demonstrated consistent performance, particularly at lower learning rates (lr=0.001), compared to other algorithms. Across different epochs, Adamax maintained a stable accuracy profile, showcasing its reliability in various training scenarios. AdamW, an extension of Adam with weight decay, consistently outperformed other algorithms, especially at lower learning rates (lr=0.001). As epochs increased, AdamW showcased sustained improvement in accuracy, suggesting its efficacy in capturing complex patterns over extended training periods. Nadam, incorporating Nesterov's momentum into Adam, showcased effective performance, particularly at lower learning rates (e.g., lr=0.001). The convergence of Nadam was evident across epochs, with a steady increase in accuracy. Rectified Adam (RAdam), an adaptive learning rate variant of Adam, proved to be effective, particularly at lower learning rates (e.g., lr=0.001). RAdam demonstrated a consistent and notable increase in accuracy as epochs progressed, underlining its suitability for prolonged training durations. As a result of our extensive investigations, we identified the optimal parameters for our custom object detection model as follows: setting the epoch value to 50, establishing a learning rate of 0.001, and utilizing the Adam optimizer.

When evaluating our model with an epoch value of 50, a learning rate of 0.001, and an Adam optimizer, the train/box_loss graph in **Figure 5** illustrates a decreasing trend in the loss value as the epoch increases. A similar pattern is observed in the val/box_loss graph. Notably, the train/cls_loss graph exhibits a dramatic reduction in cls_loss during the initial epoch, with a subsequent gradual decrease as the epoch progresses. This pattern is mirrored in the val/cls_loss graph. The train/dfl_loss graph shows an initial sharp decline followed by a somewhat fluctuating but overall decreasing trend. A comparable movement is observed in the Val/dlf_loss graph.

Examining the metrics/precision(B) graph in **Figure 5**, despite initial fluctuations in the ratios during the first epochs, precision stabilizes and approaches a value close to 1 in subsequent epochs. The metrics/recall(B), metrics/mAP50(B), and metrics/mAP50-95(B) graphs demonstrate similarities among each other. They all exhibit a sudden increase in the early epochs, followed by a slower ascent in subsequent values. It is worth noting that, despite the occasional fluctuations, the metrics/precision(B) curve attains stability, emphasizing the model's precision as it progresses through epochs. Additionally, the metrics/recall(B), metrics/mAP50(B), and metrics/mAP50-95(B) graphs collectively indicate the model's ability to maintain a consistent performance in terms of recall and mean average precision across varying confidence thresholds.

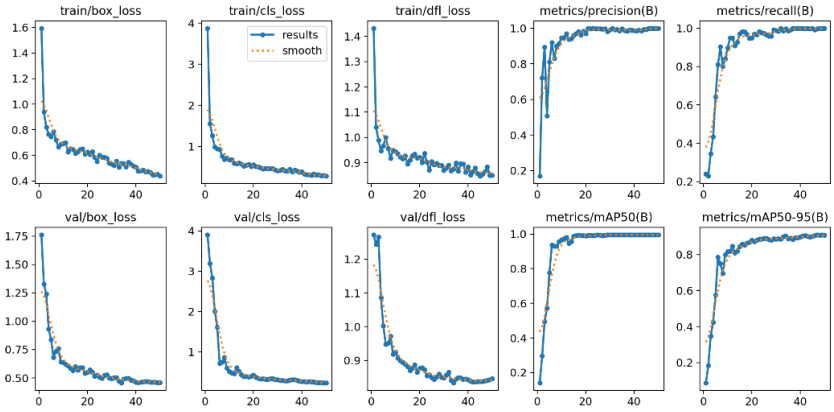


**Figure 5.** *The graphical representation of the values during the training phase of our model.*

When examining the confusion matrix, we observed that during training, our model correctly identified 59 instances of invoice data and predicted 59 instances of personal data accurately, as shown in **Figure 6**. However, we also observed that the model made an error by predicting one instance of background as invoice data, i.e., misclassifying a region without invoice data.

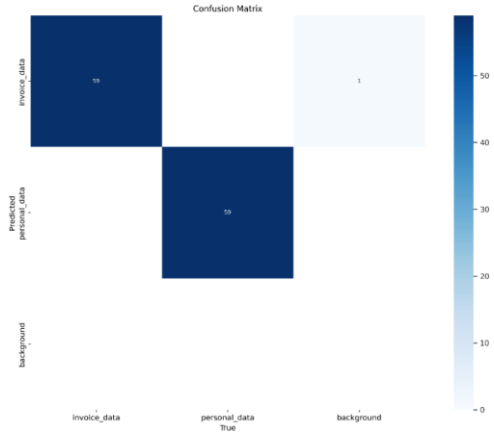


**Figure 6.** *The Confusion Matrix.*

As illustrated in **Figure 7**, when presented with an invoice image that our model had not encountered before, it accurately identified the 'personal_data' region with a confidence rate of 95%. Additionally, it predicted the location of the date information with a confidence rate of 93%.

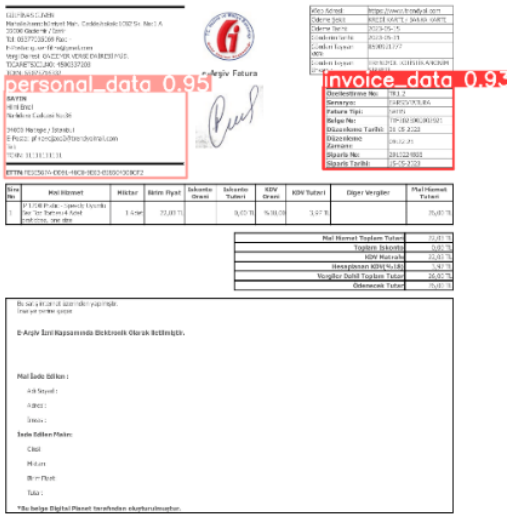


**Figure 7.** *The predictions of our model on the test image.*

### 3.2 OCR

To perform the extraction of dates from invoice images, we needed to convert the data from an invoice image into a string format that we could process. For this purpose, we utilized the Paddle OCR model. When we fed an invoice image into the OCR model, it provided us with the box coordinates of the area containing the text, the text within that area, and the predicted score for that text. The boxes indicating the text regions on the invoice, along with the texts and their corresponding scores, are detailed more comprehensively in **Figure 8**.

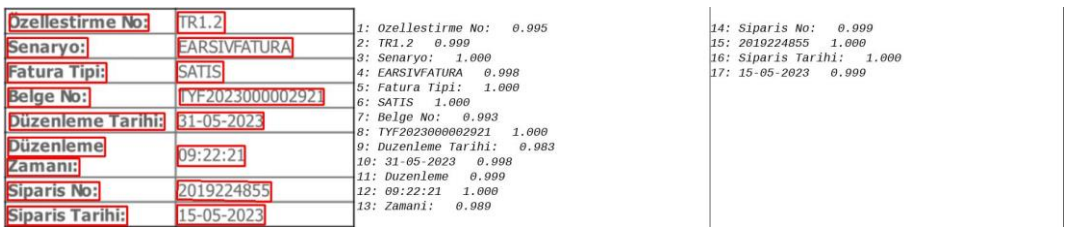


**Figure 8.** *Example of extracting text and its corresponding confidence score from an invoice image*.

### 3.3 Regular Expression

At the regular expression stage, we meticulously crafted regular expressions by considering various possible date formats in invoices. In some invoices, the name or abbreviation of the month is written instead of the numeric representation. To address this, we compiled lists containing both Turkish and English month names, as well as lists encompassing the abbreviations of these months in both languages. After creating the list of Turkish and English months, we proceeded to formulate regular expressions for possible date formats. The regular expression rules we crafted are presented in **Table 2**.

**Table 2.** *Regular expression rules and corresponding date formats.*

| RegEx | Date Format |
|---|---|
| \d{4}\s*-\s*\d{2}\s*-\s*\d{2} | YYYY-MM-DD \| YYYY-DD-MM |
| \d{2}\s*-\s*\d{2}\s*-\s*\d{4} | MM-DD-YYYY \| DD-MM-YYYY |
| \d{2}\s*/\s*\d{2}\s*/\s*\d{4} | MM/DD/YYYY \| DD/MM/YYYY |
| \d{2}\s*\.\s*\d{2}\s*\.\s*\d{4} | MM.DD.YYYY \| DD.MM.YYYY |
| \d{2}\s*(tr_month_regex")\s*\d{4} | 01 Kas 2023 \| 01 Kas 2023 |
| \d{2}\s*(en_month_regex")\s*\d{4} | 01 Jan 2023 \| 01 January 2023 |

After implementing these regex rules, when we tested our system, it efficiently performed the extraction of date information from invoices. Without the assistance of our developed object detection model, applying OCR directly to the entire invoice image took up to 23 seconds to locate the date information. However, in our experiments with the model, the process of finding and processing the date information was optimized and reduced to as fast as 5 seconds. This significant improvement has contributed to making our targeted system more efficient.

## 4. Future Works

In our future endeavors, we will work towards making user information extractable in addition to invoice data. By incorporating different regular expression rules into our model, which can already identify areas containing personal information, we aim to perform extraction processes for user-specific details such as name, surname, address, phone number, and email address from invoices. Additionally, we plan to enhance our model by adding more diverse invoice examples to our dataset. This ongoing improvement process will increase the accuracy of identifying information fields when presented with various invoice samples. Furthermore, we intend to introduce a new label to our dataset to capture information about the products the user has purchased, including details such as product name, quantity, tax rate, and price. We will continue refining our model with these additional labels. Throughout these endeavors, our primary focus will be on maximizing the efficiency of our model in terms of processing time.

## 5. Conclusion

In conclusion, our research introduces a robust methodology for efficient and accurate date extraction from diverse invoices. By combining custom object detection, OCR, and refined regular expressions, we overcome challenges posed by varied invoice structures. The object detection model, trained with a diverse dataset, demonstrates optimal performance with an epoch value of 50, a learning rate of 0.001, and the Adam optimizer. The integration of Paddle OCR and carefully crafted regular expressions enhances the system's precision, achieving remarkable time efficiency. Our approach outperforms traditional methods, providing a valuable contribution to the field of document processing. The interdisciplinary nature of our methodology, combining computer vision, deep learning, and text processing, positions it as a noteworthy advancement in the quest for efficiency and accuracy in date extraction from invoices.

## References

[1] Open Source Data Labeling | Label Studio. (n.d.). Label Studio. https://labelstud.io/

[2] Ultralytics. (n.d.). GitHub - ultralytics/ultralytics: NEW - YOLOv8 in PyTorch > ONNX > OpenVINO > CoreML > TFLite. GitHub. https://github.com/ultralytics/ultralytics

[3] PaddlePaddle. (n.d.). GitHub - PaddlePaddle/PaddleOCR: Awesome multilingual OCR toolkits based on

PaddlePaddle (practical ultra lightweight OCR system, support 80+ languages recognition, provide data annotation and synthesis tools, support training and deployment among server, mobile, embedded and IoT devices). GitHub. https://github.com/PaddlePaddle/PaddleOCR

[4] M. S. Satav, T. Varade, D. Kothavale, S. Thombare and P. Lokhande, "Data Extraction From Invoices Using Computer Vision," 2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS), RUPNAGAR, India, 2020, pp. 316-320, doi: 10.1109/ICIIS51140.2020.9342722.

[5] D. A. Kosiba and R. Kasturi, "Automatic invoice interpretation: invoice structure analysis," Proceedings of 13th International Conference on Pattern Recognition, Vienna, Austria, 1996, pp. 721-725 vol.3, doi: 10.1109/ICPR.1996.547263.

[6] H. Sidhwa, S. Kulshrestha, S. Malhotra and S. Virmani, "Text Extraction from Bills and Invoices," 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), Greater Noida, India, 2018, pp. 564-568, doi: 10.1109/ICACCCN.2018.8748309.

[7] Tesseract-Ocr. (n.d.). GitHub - tesseract-ocr/tesseract: Tesseract Open Source OCR Engine (main repository). GitHub. https://github.com/tesseract-ocr/tesseract

[8] R. Smith, "An Overview of the Tesseract OCR Engine," Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), Curitiba, Brazil, 2007, pp. 629-633, doi: 10.1109/ICDAR.2007.4376991.

[9] Ar, I., Karsligil, M.E. (2007). Text Area Detection in Digital Documents Images Using Textural Features. In: Kropatsch, W.G., Kampel, M., Hanbury, A. (eds) Computer Analysis of Images and Patterns. CAIP 2007. Lecture Notes in Computer Science, vol 4673. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-74272-2_69

[10] Zhang, X., Duan, L., Ma, L., Wu, J. (2017). Text Extraction for Historical Tibetan Document Images Based on Connected Component Analysis and Corner Point Detection. In: Yang, J., et al. Computer Vision. CCCV 2017. Communications in Computer and Information Science, vol 772. Springer, Singapore. https://doi.org/10.1007/978-981-10-7302-1_45

[11] Dachengwang,. (2011). ANALYSIS OF FORM IMAGES. International Journal of Pattern Recognition and Artificial Intelligence. 08. 10.1142/S0218001494000528.