algorithms

MDPI

Article

Efficient Expiration Date Recognition in Food Packages for Mobile Applications

Hao Peng, Juan Bayon, Joaquin Recas * and Maria Guijarro

Facultad de Informática C/ Profesor José García Santesmases, Universidad Complutense de Madrid, 9 Ciudad Universitaria, 28040 Madrid, Spain; penghao@ucm.es (H.P.); jbayon01@ucm.es (J.B.); mguijarro@ucm.es (M.G.)
* Correspondence: recas@ucm.es

Abstract: The manuscript introduces an innovative framework for expiration date recognition aimed at improving accessibility for visually impaired individuals. The study underscores the pivotal role of convolutional neural networks (CNNs) in addressing complex challenges, such as variations in typography and image degradation. The system attained an F1-score of 0.9303 for the detection task and an accuracy of 97.06% for the recognition model, with a total inference time of 63 milliseconds on a single GeForce GTX 1080 GPU. A comparative analysis of quantized models—FP32, FP16, and INT8—emphasizes the trade-offs in inference speed, energy efficiency, and accuracy on mobile devices. The experimental results indicate that the FP16 model operating in CPU mode achieves an optimal equilibrium between precision and energy consumption, underscoring its suitability for resource-constrained environments.

Keywords: expiration date recognition; convolutional neural network; vision impairment

Academic Editor: Stefano Mariani

Received: 9 April 2025
Revised: 9 May 2025
Accepted: 10 May 2025
Published: 15 May 2025

Citation: Peng, H.; Bayon, J.; Recas, J.; Guijarro, M. Efficient Expiration Date Recognition in Food Packages for Mobile Applications. *Algorithms* **2025**, *18*, 286. https://doi.org/10.3390/a18050286

1. Introduction

Expiration date recognition is an important technology for visually impaired individuals, contributing positively to their daily routines. Globally, there are millions of people with visual impairments [1] who face challenges in identifying critical information on product packaging, such as expiration dates. This limitation can lead to the consumption of expired products, posing health risks and reducing their quality of life. Europe, with its diverse population, has made significant strides in developing advanced technologies. However, the accessibility of expiration date recognition for visually impaired people continues to hold importance. The integration of this technology into everyday tools and devices is vital for enhancing the independence and safety of these individuals. As the continent continues to progress in various sectors, it is important to prioritize the development and implementation of accessible technologies, such as expiration date recognition, to assist visually impaired individuals.

Researchers and developers are increasingly dedicating their efforts to creating fast, accurate, and reliable methods for expiration date detection. These innovations not only improve the lives of visually impaired individuals but also contribute to the broader goal of inclusive technology that benefits everyone. In the computer vision field, notably, the convolutional neural network (CNN) yields promising results in practical applications: Ref. [2] developed a deep learning framework with 97.74% accuracy for recognizing expiration dates on product packages. The framework includes three networks: a date detection network, a day–month–year (DMY) detection network, and a recognition network, and it handles 13 date formats and challenging conditions like varying fonts and blurring

using a publicly available dataset called ExpDate. Ref. [3] proposed a novel system to help visually impaired individuals recognize expiration dates on beverage packages. The system captures an image of the product taken by the user, processes it using a Spatial Transformer Network (STN) to handle skewed, misaligned, or partially missing date images, and then utilizes a dedicated convolutional neural network (CNN) for final recognition. The system achieved high recognition rates of 99.42% for images without spaces and 98.44% for images with spaces. Ref. [4] proposed an enhanced method for optical character recognition (OCR) to detect and recognize expiry dates on food packages. This method uses an improved DBNet combined with a Convolutional Block Attention Module (CBAM) to better extract character features in complex contexts. The improved model shows high accuracy and robustness, achieving a character detection accuracy of 97.9% and a recognition accuracy of 97.8%. The system has been successfully deployed on the NVIDIA Jetson Nano, enhancing its practical applicability in the food packaging industry.

While the algorithms mentioned above yield excellent detection results, they are characterized by large model sizes, numerous parameters, and significant computational requirements. Consequently, many studies have shifted their focus towards developing lightweight target-detection algorithms. Ref. [5] proposed an advanced method for detecting Litchi leaf diseases and pests. This method employs an improved fully convolutional one-stage object detection network (FCOS) named FCOS-FL, which incorporates G-GhostNet-3.2 as the backbone to create a lightweight model. The model enhances feature extraction using the Central Moment Pooling Attention (CMPA) mechanism, improving the center sampling and central loss by utilizing real target dimensions. FCOS-FL demonstrates high detection accuracy, with specific accuracy rates of 93.2% for *Mayetiola* sp. and 92% for Litchi algal spot. The system is designed for deployment on embedded devices, offering practical applications in agriculture. Ref. [6] presented an enhanced coral bleaching detection model named FCOS_EfficientNET. The model uses EfficientNet as its backbone and integrates the Bidirectional Feature Pyramid Network (BiFPN) for better feature extraction. FCOS_EfficientNET demonstrates high accuracy and real-time performance, with FCOS_EfficientNETb3 achieving a mean average precision (mAP) of 48.5% on the MS COCO dataset [7] and 81.5% accuracy on a custom coral bleaching detection dataset. The study demonstrates the potential of this model for real-time and large-scale monitoring tasks in assessing coral reef health. Ref. [8] proposed a compact and highly accurate real-time edge–AI detector for monitoring chicken health. The detector leverages an improved FCOS-Lite model with MobileNet as the backbone, incorporating a gradient weighting loss function and Complete Intersection over Union (CIoU) loss function to enhance classification and localization accuracy. Knowledge distillation is used to transfer critical information from a larger teacher model to the FCOS-Lite detector, maintaining its compactness without compromising performance. The experimental results demonstrate that the proposed detector achieves a mean average precision (mAP) of 95.1% and an F1-score of 94.2%, operating efficiently at over 20 FPS on a resource-constrained edge–AI-enabled device. The innovative approach ensures low power consumption and minimal bandwidth costs, making it suitable for practical applications in automated poultry health monitoring.

The above discussions have proved that the detection models could be deployed on edge devices, whereas the recognition models are not deployed yet. Ref. [9] presented a novel approach for scene text recognition using a single visual model, named SVTR. This method dispenses with sequential modeling and decomposes image text into small patches called character components. The model then employs hierarchical stages of component-level mixing, merging, and combining to recognize characters through a simple linear prediction. Ref. [10] presented a novel approach called PARSeq for scene text recognition (STR). The method leverages Permutation Language Modeling (PLM) to train

an ensemble of autoregressive (AR) models with shared weights. PARSeq unifies context-free non-AR and context-aware AR inference, offering state-of-the-art (SOTA) results on various STR benchmarks with an accuracy of 96.0% when trained on real data. Ref. [11] proposed SwinTextSpotter, an end-to-end scene text spotting framework leveraging a Transformer-based architecture. The framework introduces Recognition Conversion (RC) to integrate detection and recognition features, enhancing performance through joint optimization. SwinTextSpotter eliminates the need for character-level annotations or rectification modules and achieves state-of-the-art results across six benchmarks, including multi-oriented, arbitrarily shaped, and multilingual datasets. The discussed recognition models show excellent results in scene text recognition; however, the model size is huge when deploying on computation-constrained devices.

In this study, we propose that our optical character recognition (OCR) system operates in two phases: text detection and text recognition. The text detection network employs modified fully convolutional one-stage object detection (FCOS) [12] to locate characters on packages, enclosing them in rectangular boxes. The character recognition network uses a convolutional recurrent neural network (CRNN) [13] trained with CTC loss to recognize characters within these boxes. To address date time format, a date time parser is applied after the date is recognized in the original date format. The experimental results demonstrate that our method achieves high accuracy and fast inference. We propose using this OCR system to tackle the challenge of recognizing expiration dates in daily scenarios. The system, combined with edge devices, is portable, easy to deploy in daily life, and energy efficient.

The main contributions of our work are as follows:

1. We propose a lightweight backbone network for detection models, which addresses the challenges presented regarding real-time performance on computationally constrained devices, thereby significantly enhancing the model's performance on edge devices.
2. We design a date time parser to parse up to 17 different date formats for a better understanding of the dates.
3. We implement a quantized model on edge devices to reduce hardware power consumption, shrink the model size, and enhance portability.

This study is structured as follows: Section 2 presents the network composition, detector, recognizer, and parser; Section 3 details the datasets, model training, model quantization, and deployment; and Section 4 provides the results of the proposed work.

2. Methodology

The comprehensive workflow is illustrated in Figure 1. Initially, the detector identifies the date segment from the source image. This segment is then transmitted to the recognizer for date identification. Following this, the date parser translates the date into a comprehensible format—specifically, year, month, and day. Ultimately, the system provides the simplified date.

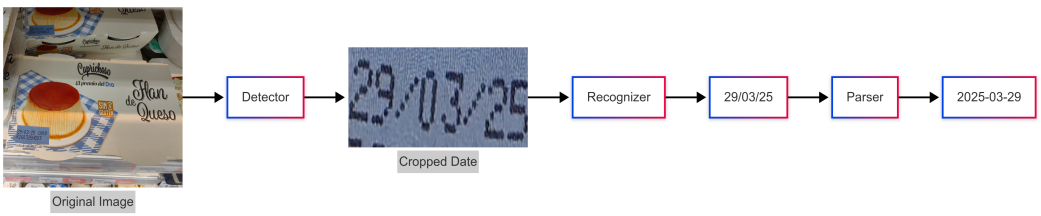

Figure 1. The overall architecture to recognize the expiration date.

2.1. Date Detector

There are commonly two object detection methods in the current deep learning field: the anchor-based object detection model and anchor-free object detection [14]. FCOS (fully convolutional one-stage object detection) [12] is adopted in this work. FCOS is an anchor-free object detection model designed to simplify the detection process while maintaining high accuracy. The basic structure of the model is shown in Figure 2.

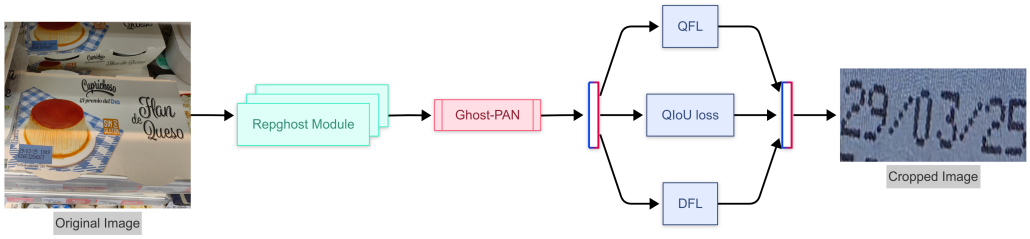

Figure 2. The architecture of the detection model.

In this study, the backbone network is tasked with deriving features from the input image, utilizing the Repghost module [15] as the foundation. Compared to the Ghost module [16], this choice lowers computational demands on hardware while delivering enhanced accuracy with fewer parameters and similar latency, which is advantageous for mobile devices. Unlike the Ghost module, the RepGhost module eliminates the use of the inefficient concatenation operator, which considerably reduces inference time. Additionally, the process of information fusion is implicitly handled by the add operator rather than delegating it to other convolutional layers. Ghost-PAN [17] substitutes the traditional Feature Pyramid Networks (FPNs) due to the substantial computational load of typical FPNs. The network head produces three outputs: quality focal loss (QFL), distribution focal loss (DFL), and GIoU loss. QFL and DFL are based on generalized focal loss [18]. QFL is designed to refine both the classification and localization aspects of object detection, enhancing its effectiveness in dense object detection tasks. QFL, in particular, targets improving classification by emphasizing hard-to-classify samples. The QFL function is defined as (1)

$$QFL(\sigma) = -|y - \sigma|^{\beta}[(1-y)log(1-\sigma) + ylog\sigma] \tag{1}$$

where σ is the predicted classification probability, y is the annotated label, and β is the weight.

GIoU ranges from -1 to 1, where 1 means perfect overlap, 0 means no overlap, and negative values indicate that the predicted box is outside the ground truth box. GIoU function is shown in (2)

$$GIoU = \frac{|A \cap B|}{|A \cup B|} - \frac{|A \cup B| - A \cap B}{|A \cup B|} \tag{2}$$

where A and B are areas of the predicted and ground truth bounding boxes, respectively. $|A \cup B|$ stands for the area of the union of the predicted and ground truth bounding boxes, while $|A \cap B|$ is the area of the intersection of the predicted bounding box A and the ground truth bounding box B.

DFL enhances the bounding box regression by encouraging the model to predict more distributed and qualified bounding boxes,

$$DFL(s_i, s_{i+1}) = -[(y_{i+1} - y)log(s_i) + (y - y_i)log(s_{i+1})] \tag{3}$$

where y_i and y_{i+1} are the two closest intervals of the true label y, and s_i and s_{i+1} are the corresponding model outputs.

2.2. Date Recognizer

For the purpose of date recognition, an adapted convolutional recurrent neural network (CRNN) [13] is employed, known for its effective application in recognizing handwritten text. This is because it captures spatial features of characters and the sequential flow of text efficiently. Initially, the pretrained model undergoes fine-tuning with a dataset that comprises a variety of characters, including uppercase, lowercase, and diverse symbols commonly found in dates.

Figure 3 depicts the structure of the CRNN model.

(1) Convolution layer extracts spatial features from the input using convolution operations;
(2) MaxPooling layer reduces the spatial dimensions of the feature maps generated by the convolutional layer, which makes the model computationally efficient and reduces the risk of overfitting;
(3) Reshape layer flattens the 2D spatial data into a 1D vector to prepare for the RNN layer;
(4) Bidirectional LSTM captures temporal or sequential patterns in the data.

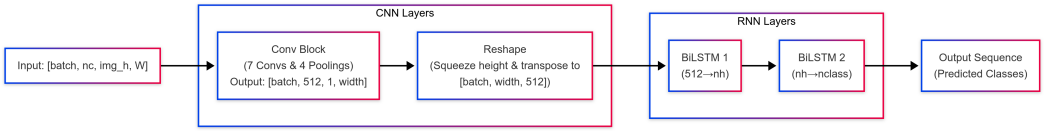

Figure 3. The CRNN model structure.

2.3. Date Time Parser

The date time parser can assist individuals with vision impairments by helping them to comprehend the date quickly. It breaks down the date into its components—day, month, and year—and converts these into a clearer y–m–d format. This conversion is vital for precise comprehension. At present, the algorithm accommodates the date format detailed in Table 1, supporting 17 distinct formats. In these formats, *Y* signifies the year as a 4-digit number; *y* represents the year with two digits; *m* and d indicate the month and day as two-digit numbers, respectively; and *b* is the month's abbreviation. Algorithm 1 illustrates the parser's operation: it initially inputs the date string to ascertain possible date interpretations, then measures each interpretation's proximity to the current date, ultimately selecting the closest match.

Algorithm 1 Date Time Parser Algorithm

Input: *date_str* (string), *formats* (list of string formats)
Output: *closest_date* (the closest date to the current date)
Initialize: *current_date* ← **current date**, *parsed_dates* ← **empty list**, *closest_date*
for each *format* **in** *formats* **do**
  attempt to parse *date_str* **using** *format*
  if parsing is successful then
    parsed_date ← **datetime.strptime**(*date_str*, *format*)
    add *parsed_date* **to** *parsed_dates*
  end if
end for
differences ← **empty list**
for each *date* **in** *parsed_dates* **do**
  distance ← $|date - current\_date|$
  add (*distance*, *date*) **to** *differences*
end for
closest_date ← *date* **with the minimum** *distance* **from** *differences*
return *closest_date*

Table 1. Different date formats supported to parse.

Number	Date Format	Sample
1.	Y,m,d	2024,02,13
2.	d,m,Y	13,02,2024
3.	m,d,Y	02,13,2024
4.	b,d,Y	FEB,13,2024
5.	d,b,Y	13,FEB,2024
6,	Y,b,d	2024,FEB,13
7.	m,Y	02,2024
8.	Y,m	2024,02
9.	b,Y	FEB,2024
10.	Y,b	2024,FEB
11.	y,m,d	24,02,13
12.	d,m,y	13,02,24
13.	d,b,y	13,FEB,24
14.	b,d,y	FEB,13,24
15.	y,b,d	24,FEB,13
16.	m,d	02,13
17.	d,m	13,02

3. Datasets and Implementation Details

3.1. Datasets

The detection model dataset is used from the *ExpDate* of Seker et al. [2]. It consists of two datasets, which are a dataset **Products-Real** that contains 1767 real-world expiration date images from food, beverage, and medicine packages and a dataset **Products-Syn** that has more than 12 thousand synthetic images that were used to train the network. For the recognition model, the dataset consists of self-labeled expiration date images, which contain 510 samples [19].

3.2. Model Training

The proposed models are trained with the PyTorch CUDA 11.8 framework. They are trained on a single NVIDIA GeForce GTX 1080 GPU with 8 GB of memory. Table 2 shows the parameters of the detection model and the recognition model, respectively.

Table 2. The parameters of the detector and recognizer, respectively.

Model	Detector	Recognizer
input size	[800,1280]	[32,128]
optimizer	AdamW	Adam
learning rate	0.001	0.0007
batch size	6	384
epoch	20	20

Detection Model: AdamW optimizer is chosen because it can achieve faster convergence compared to the standard Adam optimizer by decoupling weight decay from the gradient update and making it more resilient to variations in the training data. The input size is 800 in width and 1280 in height, and the initial learning rate is 0.001 with warm-up changes. The batch size is 6 per GPU and a total of 20 training epochs. Data augmentation is applied to make more training samples; the augmentation techniques include scale, stretch, translate, random rotation no more than 5°, random brightness, random saturation, random contrast, and normalized with mean [103.53, 116.28, 123.675] and variance [0.017429, 0.017507, 0.017125] in BGR order.

Recognition Model: the optimizer is Adam; the input size is resized to 32 width and 128 height. The initial learning rate is 7×10^{-4}; the batch size per GPU is 384 with a total of 20 training epochs. The data augmentation used includes Gaussian blur, motion blur, random rotation no more than 30°, shear, translate, and normalized with mean [127.5, 127.5, 127.5] and variance [1/127.5, 1/127.5, 1/127.5] in RGB order.

3.3. Android Deployment

Neural networks have significantly pushed forward the limits in various applications; however, they frequently require considerable computational resources. To incorporate modern networks into edge devices that have stringent power and compute constraints, it is crucial to minimize the power consumption and latency of neural network inference. Quantizing neural networks stands out as a highly effective method for achieving these reductions, although it may introduce additional noise that could result in decreased accuracy [20].

NCNN is a high-performance neural network inference computing framework optimized for mobile platforms [21]. This research employed the NCNN framework to quantify the model and construct a real-time expiration date identification and character recognition application utilizing the quantized model. Figure 4 illustrates the quantization procedure.

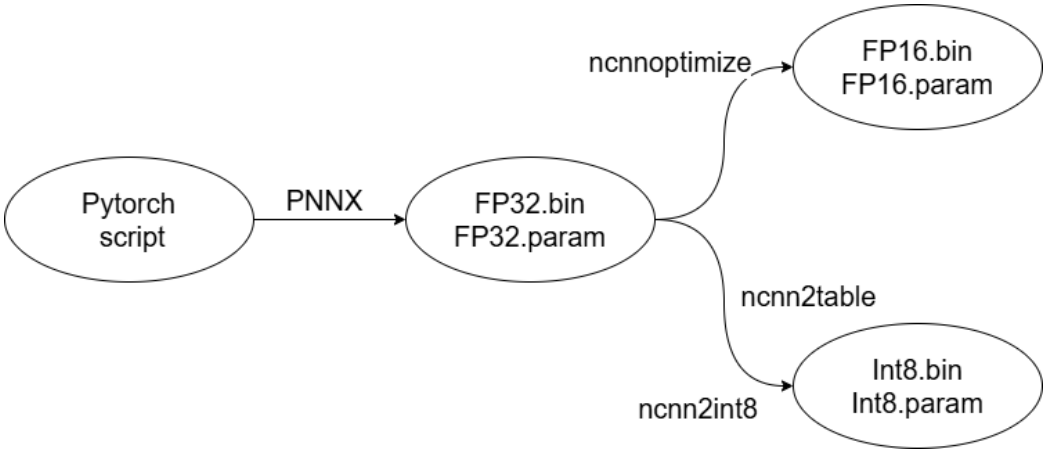

Figure 4. Flow chart of NCNN quantization.

The Android implementation is verified on an Oppo Find X5 running with Android version 13 with processor Qualcomm Snapdragon 888, a Realme GT Neo2 running Android 13 with Snapdragon 870, and Redmi 10 Pro running Android 13 with Snapdragon 732G. The smartphones have an application that was developed based on Java Development Kit 21.0.3, NCNN 20241226-android-vulkan, and Android Studio 2024.2.2 Patch 1.

The expiration date real-time recognition application created in this study is composed of three main components (Figure 5): the video acquisition module, the processing module, and the result display module. The video acquisition module uses the smartphone's camera to capture food packages in real time; the processing module inputs these video streams into the detection model frame by frame while keeping the resolution size of the image fixed at 480×640, and the detected area is cropped out; then, it preprocesses the cropped image and sends it to the recognition model; after the recognition receives the correct date, the date time parser tries to transform the date format into a unified format; the result display module then shows the predicted bounding boxes and the parsed characters in real time, providing the non-maximum suppression (NMS) threshold bar and the confidence threshold bar to adjust in the expiration date recognition app developed in this research. As shown in Figure 6, the app's overall interface features an image with working unit CPU or GPU of the smartphone, the size of the image, the inference time, frames per second

(FPS), and the average FPS. Users can adjust the NMS threshold bar and the confidence threshold bar to change how the detected area would be shown on the interface.

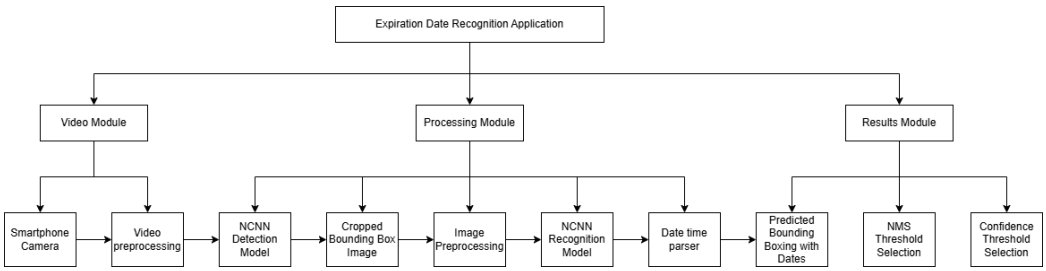

Figure 5. Expiration date recognition application flow chart.

Figure 6. Application interface.

4. Results

This part provides an assessment of each network separately, as well as an analysis of the overall effectiveness of the proposed work and the Android application.

4.1. Detector Results

In this evaluation, a confusion matrix serves as the primary metric. This matrix is an effective means of assessing a classification model's performance, including its application in detection scenarios. It supplies a detailed comparison of the model's predictions versus the true labels, enabling the identification of areas where the model may falter. This particular study focuses on a classification task involving a single class, the expiration date.

To illustrate the impact of model conversion and quantization, the configurations remain unchanged. Although the IoU metric might not ensure that all the digits are encapsulated within the detected region—since it can be considered a true positive even when missing some digits, potentially leading to varied date interpretation—it serves as a useful parameter for assessing the performance of different models.

The same datasets are used by this work and [2] regarding the evaluation of the *Products-Real* dataset. It is worth noting that the date label is only selected since the other labels would be redundant. The numerical results are taken from [2] for comparing the differences in our proposed work with theirs. From Table 3, the precision, recall, and F1-score are 0.9850, 0.8814, and 0.9303, respectively, in this work; by contrast, the metrics are 0.9758, 0.9905, and 0.9831, respectively, in the previous work. While our model excels in precision, it sacrifices recall, which results in an overall F1-score that is lower than the previous work's. The precision, recall, and F1-score are calculated as follows,

1. Precision: $(\frac{TP}{TP+FP})$;
2. Recall (Sensitivity): $(\frac{TP}{TP+FN})$;
3. F1-Score: $(2 \times \frac{Precision \times Recall}{Precision+Recall})$.

Table 3. Combined performance metrics of detection and recognition models.

Model	Precision	Recall	F1-Score	Rec Acc	Det Size	Rec Size
Torch Script	0.9850	0.8814	0.9303	97.06%	23.63 MB	31.96 MB
FP32	0.9809	0.8756	0.9253	97.06%	2.60 MB	31.80 MB
FP16	0.9809	0.8756	0.9253	97.06%	1.32 MB	15.90 MB
INT8	0.7281	0.9780	0.8347	44.12%	0.75 MB	8.54 MB

The tests were conducted in the same environment on a single GeForce GTX 1080 GPU. The detection model of [2] took 239 ms per image on average, while the proposed detection model can infer with 53 ms per image on average. The Torch Script model offers the best overall performance in terms of precision and F1-score but is also the largest in size for the PC platform. The FP32 and FP16 models have similar performance with slightly reduced precision and F1-score but are significantly smaller in size for edge devices. The INT8 model has a significant drop in precision and F1-score.

4.2. Recognizer Results

After fine-tuning the model, the accuracy can reach 97.06%. To build statistical confidence, k-fold cross validation is performed. Due to the small amount of real data, k is selected as 5. The validation accuracy interval is [94.35%, 99.76%]. The fine-tuning accuracy, 97.06%, falls in the interval. The accuracy is calculated by comparing each output character and the ground truth character, which consists of letters and symbols. The recognition inference in this work is 10 ms per image, while, in the work of [2], it is 92 ms per image; both are tested in the same environment on a single NVIDIA GTX 1080 GPU.

Table 3 shows that the Torch Script, FP32, and FP16 models all have a similar high accuracy of 97.06%, indicating that they perform almost identically in terms of prediction accuracy, and the INT8 model has a significantly lower accuracy of 44.12%, which suggests a substantial loss in performance. As for the model size, the FP32 and Torch Script models have similar sizes, both around 31.8 MB, and the FP16 model, using half-precision floating points, has a significantly smaller size of 15.9 MB while maintaining the same accuracy, which is beneficial for scenarios where storage or memory efficiency is crucial without compromising on performance. While the INT8 model is the smallest, at 8.54 MB, this shows that it has been heavily quantized. However, this quantization comes at the cost of accuracy.

4.3. The Overall Performance

Figure 7 illustrates the functionality of the proposed method on local product packages. It shows that the area indicating the expiration date can be detected, with the date subsequently recognized and converted into a standardized format. Dates lacking a specific day are automatically adjusted to the start of the month; for instance, in the first row of Figure 7, 01 2025 is translated to 2025-01-01. The second row in Figure 7 depicts that dates including the full year are accurately processed. The third row demonstrates that dates using a two-digit format for the year can also be correctly interpreted, as seen with 30.03.25, which is processed as 2025-03-30.

Figure 7. Recognition results of expiration dates after date time parsing on PC.

4.4. Android Deployment

Figure 8 shows the different quantized detection and recognition models run 100 times to obtain the mean inference time and the standard deviation.

Observations across phone models indicate that the Oppo phone generally performs well across all the tests, with lower inference times on both the CPU and GPU. Notably, the GPU times are consistently better than the CPU times, particularly for detection. The Realme phone is slightly slower than Oppo in most tests, especially for GPU detection times. However, its CPU performance is fairly competitive in recognition. The Redmi phone is significantly slower than the other phones across all the models, especially in GPU detection times. The CPU detection and recognition are also notably higher. The processors of the three phones can be ranked in descending order of performance as Oppo, Realme, and Redmi. Consequently, on the whole, the Oppo phone delivered the best performance.

Regarding analysis by quantized models (FP32, FP16, and INT8), the FP32 detection model on both the CPU and GPU shows relatively consistent performance but falls behind the FP16 model. The FP32 recognition times are moderate and fairly stable. The FP16 model typically exhibits faster inference times compared to FP32, particularly for recognition. This is evident in both the CPU and GPU tests. The FP16 detection on the GPU remains steady

across devices. The INT8 model shows mixed results, with faster recognition times on the CPU compared to FP32 and FP16. The INT8 detection on the GPU varies; Oppo handles INT8 models better than Realme and Redmi. Transforming a model from FP32 to INT8 could lead to a reduction in precision because of the less complex numerical representation. This might necessitate additional processing steps like dequantization or reshaping of data, especially in devices that do not natively accommodate INT8. Consequently, the INT8 model might perform slower compared to the FP32 model.

When comparing mobile CPU and GPU performance, GPUs generally expedite detection tasks through parallel processing, although some models defy this trend, with CPUs outperforming GPUs in these scenarios. In recognition tasks, the performance differences are more nuanced, with GPUs sometimes only marginally slower than CPUs. The data transfer overhead between the two further affects the outcomes—especially for tasks involving small datasets or simple computations—often giving the CPU an edge.

When converting a network from full precision to INT8, certain layers tend to be more sensitive than others. Recurrent layers, due to their internal gating mechanisms and the wide dynamic range of activations that occur during sequence processing, recurrent layers, such as the Bidirectional LSTM units in our CRNN, are particularly prone to quantization error. The quantization noise can disrupt the delicate balance between the gating operations (input, forget, and output gates), leading to a significant drop in performance [22]. Moreover, our testing results demonstrated that the quantization and dequantization operations are introduced after INT8 quantization as the inference is even slower. Experimenting on quantization-aware training (QAT) would also yield slow inference. Overall, the benefits of pursuing advanced methods might not justify the additional complexity.

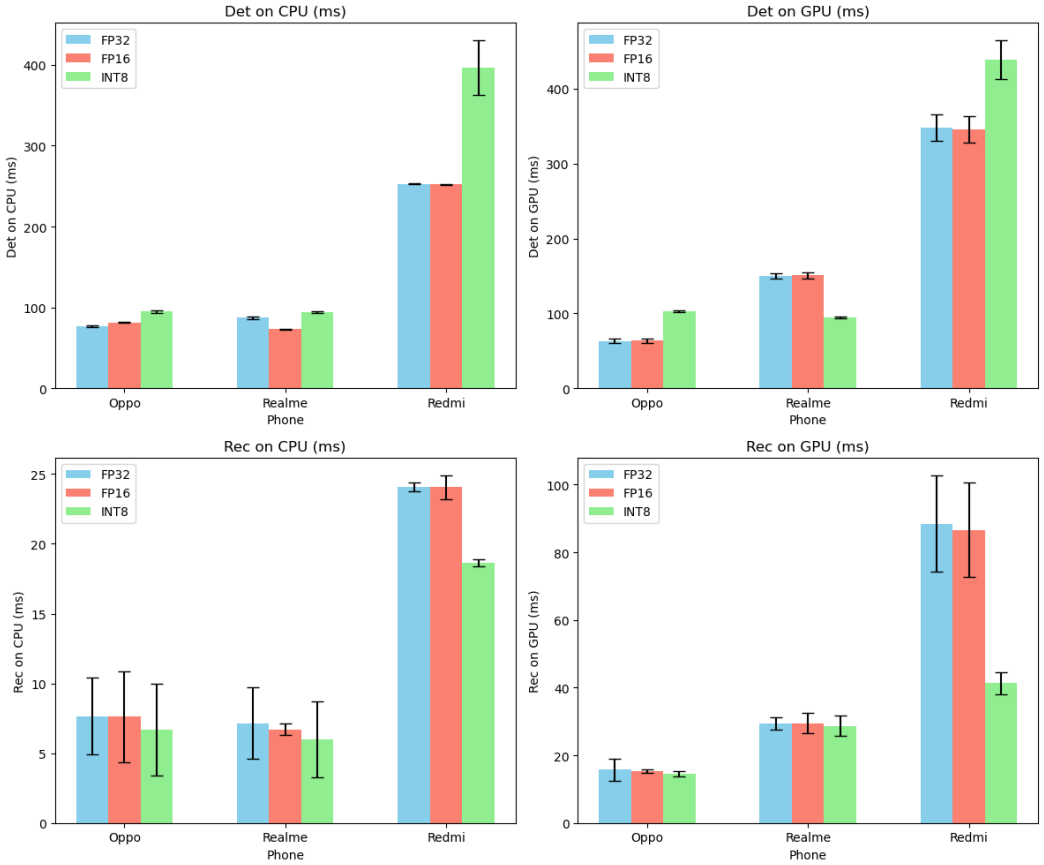

Figure 8. Detection and recognition of quantized models' inference time on CPUs and GPUs with different phones.

As Figure 9 shows, the energy consumption of the application on the Oppo phone reflects its previous performance. The phone is set in flight mode, and the brightness is at maximum while the application is working without any other applications in the background, and the remaining battery energy is recorded every 1 min starting from 100% battery energy, continuously working for 2 h. While the controlled conditions might represent a worst-case or idealized scenario, they serve as an essential baseline for comparing the relative energy efficiency improvements across different model configurations and quantization strategies. Future work will extend these evaluations to more realistic usage conditions; however, establishing a baseline under standardized conditions is a critical step for isolating the performance characteristics of the methods under investigation. The FP32, FP16, and INT8 models are tested in CPU and GPU modes, respectively. In CPU mode, the FP16 model and INT8 model almost consume the same energy, while there is a huge energy drop in GPU mode compared to the same model. Additionally, the INT8 model suffers a significant drop in accuracy after quantization. To take into account the battery of the phone and the accuracy of the model, it is recommendable to use the FP16 model in CPU mode.

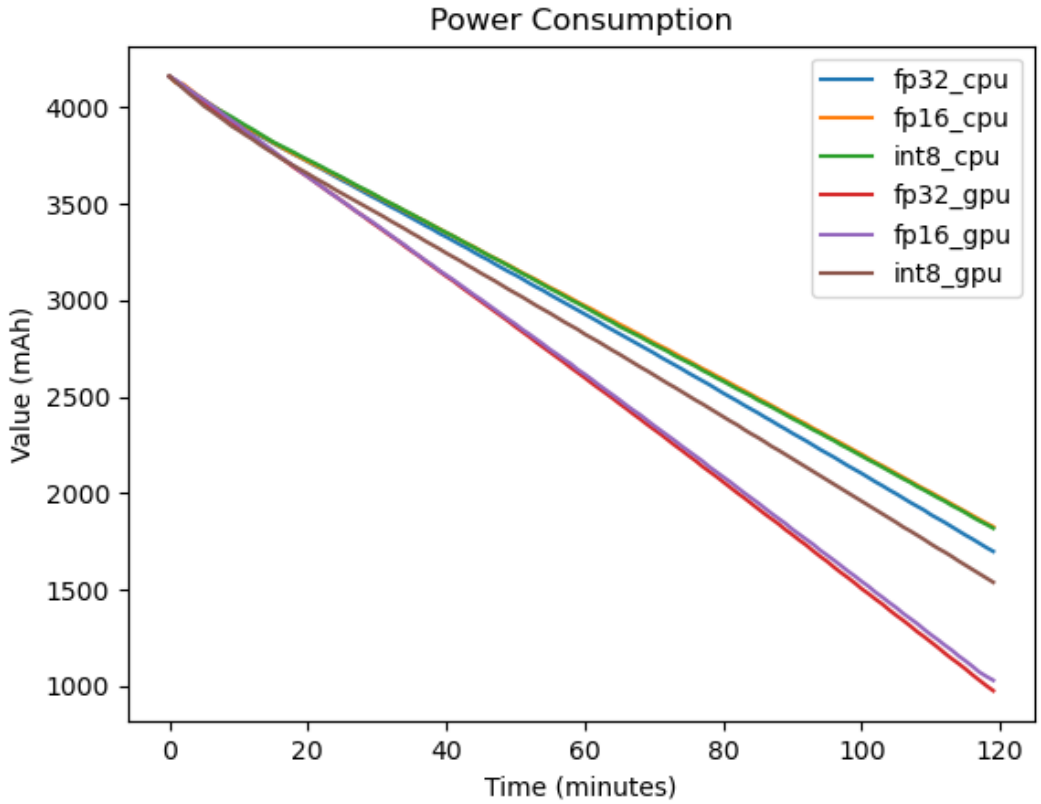

Figure 9. Power consumption on the Android phone.

Figure 10 shows how the proposed work recognizes the expiration dates in real time. All the images from Figure 10 are screenshots from the Redmi phone; the average FPS working in CPU mode is around 2. It shows feasibility on computationally constrained devices and a huge advantage in terms of power consumption.

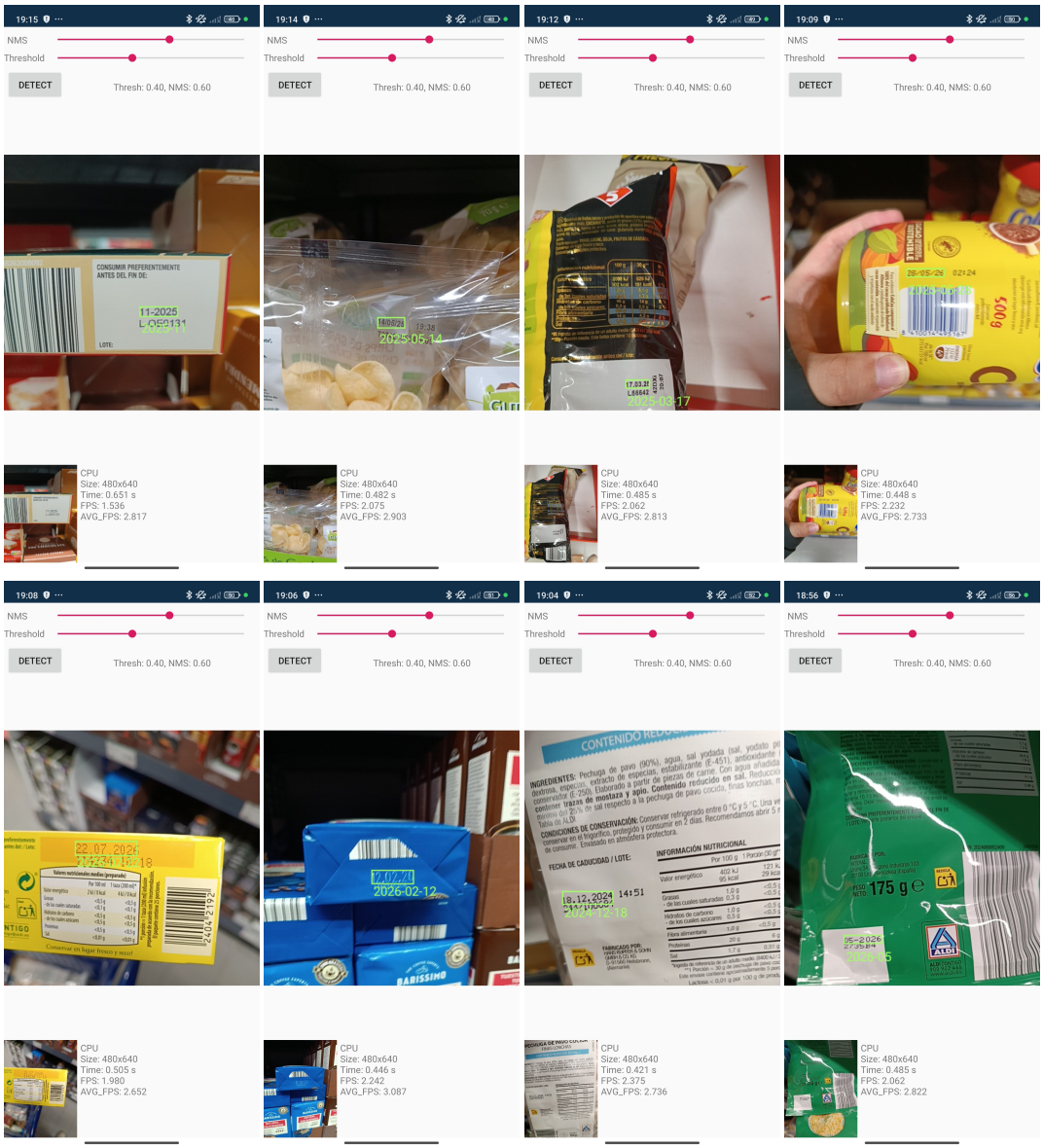

Figure 10. Recognition results of expiration dates after date time parsing on mobile phone.

5. Conclusions

In conclusion, this work has demonstrated significant advancements in the recognition and interpretation of expiration dates on product packages, emphasizing the importance of model optimization for edge devices. Among the tested models, the FP16 configuration strikes an ideal balance between performance, accuracy, and energy efficiency, making it particularly suited for deployment in computationally constrained environments. Furthermore, the proposed method's ability to process a wide variety of date formats and adapt to computational limitations showcases its practicality and versatility. We are particularly grateful to ONCE (Organización Nacional de Ciegos Españoles), the Spanish National Organization of the Blind, for suggesting this concept to us. We intend to carry out evaluations to guarantee the fundamental accessibility of our solution.

Comparing prior research, this study achieves much faster inference and proposes an Android application, emphasizing accessibility for visually impaired users. Exploring alternative quantization methods and expanding the dataset for diverse environments are essential steps for improving robustness and generalizability. Future research could explore rotating box detection while maintaining computational efficiency and accuracy. Here is

our open-source model: https://github.com/AnanasPizzaMigliore/ExpRec (accessed on 1 April 2025).

Author Contributions: Conceptualization, H.P.; methodology, H.P.; software development, H.P.; validation, H.P. and J.B.; formal analysis, H.P. and J.R.; investigation, H.P.; resources, H.P.; data curation, H.P.; writing—original draft preparation, H.P.; writing—review and editing, H.P., J.B., J.R. and M.G.; visualization, H.P.; supervision, J.R. and M.G.; project administration, J.R. and M.G.; funding acquisition, J.R. and M.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the Ministry of Science and Innovation (Grant nos. PID2021-125596OB-I00 and PLEC2022-009261) and the UCM-ONCE-Tiflotechnology Chair.

Data Availability Statement: The data are available at https://felizang.github.io/expdate/ and https://github.com/AnanasPizzaMigliore/510-Date (accessed on 1 April 2025).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. World Health Organization. Blindness and Visual Impairment. 10 August 2023. Available online: https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment (accessed on 26 March 2025).
2. Seker, A.C.; Ahn, S.C. A generalized framework for recognition of expiration dates on product packages using fully convolutional networks. *Expert Syst. Appl.* **2022**, *203*, 117310. [CrossRef]
3. Takeuchi, Y.; Suzuki, K. Expiration date recognition system using spatial transformer network for visually impaired. In Proceedings of the International Conference on Computers Helping People with Special Needs, Linz, Austria, 8–12 July 2024; pp. 517–524.
4. Zheng, J.; Li, J.; Ding, Z.; Kong, L.; Chen, Q. Recognition of expiry data on food packages based on improved DBNet. *Connect. Sci.* **2023**, *35*, 1–16. [CrossRef]
5. Xie, J.; Zhang, X.; Liu, Z.; Liao, F.; Wang, W.; Li, J. Detection of litchi leaf diseases and insect pests based on improved FCOS. *Agronomy* **2023**, *13*, 1314. [CrossRef]
6. Xin, G.; Xie, H.; Kang, S.; Chen, Y.; Jiang, Y. Improved research on coral bleaching detection model based on FCOS model. *Mar. Environ. Res.* **2024**, *200*, 106644. [CrossRef] [PubMed]
7. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In Proceedings of the Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, 6–12 September 2014; Proceedings, Part v 13; pp. 740–755.
8. Tong, Q.; Wang, J.; Yang, W.; Wu, S.; Zhang, W.; Sun, C.; Xu, K. Edge AI-enabled chicken health detection based on enhanced FCOS-Lite and knowledge distillation. *Comput. Electron. Agric.* **2024**, *226*, 109432. [CrossRef]
9. Du, Y.; Chen, Z.; Jia, C.; Yin, X.; Zheng, T.; Li, C.; Du, Y.; Jiang, Y.G. Svtr: Scene text recognition with a single visual model. *arXiv* **2022**, arXiv:2205.00159.
10. Bautista, D.; Atienza, R. Scene text recognition with permuted autoregressive sequence models. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23–27 October 2022; pp. 178–196.
11. Huang, M.; Liu, Y.; Peng, Z.; Liu, C.; Lin, D.; Zhu, S.; Yuan, N.; Ding, K.; Jin, L. Swintextspotter: Scene text spotting via better synergy between text detection and text recognition. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 4593–4603.
12. Tian, Z.; Shen, C.; Chen, H.; He, T. FCOS: A simple and strong anchor-free object detector. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *44*, 1922–1933. [CrossRef] [PubMed]
13. Shi, B.; Bai, X.; Yao, C. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *39*, 2298–2304. [CrossRef] [PubMed]
14. Liu, S.; Zhou, H.; Li, C.; Wang, S. Analysis of anchor-based and anchor-free object detection methods based on deep learning. In Proceedings of the 2020 IEEE International Conference on Mechatronics and Automation (ICMA), Beijing, China, 13–16 October 2020; pp. 1058–1065.
15. Chen, C.; Guo, Z.; Zeng, H.; Xiong, P.; Dong, J. Repghost: A hardware-efficient ghost module via re-parameterization. *arXiv* **2022**, arXiv:2211.06088.
16. Han, K.; Wang, Y.; Tian, Q.; Guo, J.; Xu, C.; Xu, C. Ghostnet: More features from cheap operations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 1580–1589.

17. RangiLyu. NanoDet-Plus: Super Fast and High Accuracy Lightweight Anchor-Free Object Detection Model. 2021. Available online: https://github.com/RangiLyu/nanodet (accessed on 5 April 2025).
18. Li, X.; Wang, W.; Wu, L.; Chen, S.; Hu, X.; Li, J.; Tang, J.; Yang, J. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 21002–21012.
19. AnanasPizzaMigliore. 510-Date. 2025. Available online: https://github.com/AnanasPizzaMigliore/510-Date (accessed on 8 April 2025).
20. Nagel, M.; Fournarakis, M.; Amjad, R.A.; Bondarenko, Y.; Van Baalen, M.; Blankevoort, T. A white paper on neural network quantization. *arXiv* **2021**, arXiv:2106.08295.
21. Tencent. ncnn: A High-Performance Neural Network Inference Framework. 2025. Available online: https://github.com/Tencent/ncnn (accessed on 2 April 2025).
22. Alom, M.Z.; Moody, A.T.; Maruyama, N.; Van Essen, B.C.; Taha, T.M. Effective quantization approaches for recurrent neural networks. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.