# Optimizing OCR Technology for Performance and Efficiency in Edge Networks

**Shaolong Zheng(23320231154461, XY)[2], Wenting Zeng(23020231154162 AI)[1], Yunqian Li(23020231154205, XY)[1], Yilu Sun(23020231154224, XY)[1], Hao Fang(24520230157439, XY)[3]**

[1]Department of Computer Science and Technology
[2]Department of Information and Communication Engineering
[3]School of Medicine

**Abstract**

With the rapid development of Internet of Things (IoT) technology, edge computing, as a distributed computing paradigm, has emerged as a crucial approach for processing real-time data. This study explores the application of Optical Character Recognition (OCR) technology in edge networks, aiming to achieve efficient processing of real-time image text. We first introduce the fundamental principles of OCR technology and optimize it for use in resource-constrained edge devices. By deploying OCR models at edge nodes, we achieve immediate recognition of real-time image text, significantly reducing transmission latency and bandwidth consumption. Furthermore, by employing lightweight algorithms and model pruning techniques in the model training and inference processes, we further enhance performance on edge devices. Experimental results demonstrate that our approach effectively reduces computational resource consumption while maintaining high accuracy, providing robust support for OCR applications in edge network environments.

## Introduction

With the rapid advancement of IoT technology, an increasing amount of real-time data requires processing on edge devices to meet the demands for real-time performance and low latency. The application of OCR technology at the edge provides robust technical support for efficient processing of real-time image text. By deploying OCR technology in edge networks, real-time recognition of image text can be achieved without relying on cloud computing resources(Reiss-Mirzaei, Ghobaei-Arani, and Esmaeili 2023), thereby providing powerful support for various real-time application scenarios, such as intelligent monitoring and automatic recognition.

Edge computing is a distributed computing paradigm (Abdellatif et al. 2023) that brings computing resources and data storage closer to where data is generated, reducing transmission latency and bandwidth consumption (Chi et al. 2023). In edge computing, edge devices play a crucial role and can include various IoT devices such as smartphones, sensors, and cameras (Chai et al. 2023; Cui et al. 2023).

OCR technology is a technique that converts text information from images into editable text format (Kim et al. 2022). Its basic principle involves extracting text regions through image processing algorithms and then using pattern recognition techniques to identify characters in the image as text information. OCR technology has broad application prospects in fields such as image processing and text recognition (Li et al. 2023). However, its application on edge devices faces challenges and requires optimization to adapt to resource-constrained environments.

One of the key functions of edge computing is task offloading (also known as computational offloading), which allows computationally intensive tasks of mobile applications to be offloaded from the user equipment (UE) to an edge computing host at the network edge. In real-world scenarios, OCR technology is actually composed of multiple sub-tasks (Koti, Khare, and Khare 2023), such as text detection, text recognition, etc. (which can actually be divided into even more sub-tasks), and each sub-task has different computational latencies (Liu et al. 2023). When performing task offloading, it is necessary to consider the transmission delay between the terminal and the edge, and choosing appropriate models and offloading methods can effectively reduce the total delay of the system.

## Related work

In recent years, the integration of OCR technology with edge computing has gained significant attention from researchers and practitioners. This section provides an overview of the relevant literature in this field.

### Text Detection

The field of natural scene text detection has rapidly evolved to meet the demands of applications such as optical character recognition, and augmented reality. The early Connected Text Proposal Network (CTPN) (Tian et al. 2016) integrated sliding windows with region-based methods and used Recurrent Neural Networks to connect disjoint text fragments, enhancing text detection across various scales and orientations. However, it had limitations in complex scenarios compared to later methods.TextBoxes++ (Liao, Shi, and Bai 2018), introduced in 2018, employed quadrilateral sliding windows for effectively detecting text in arbitrary orientations and improved adaptability to different text types. EAST (Zhou et al. 2017) struck a balance between speed and accuracy by simplifying the detection process through

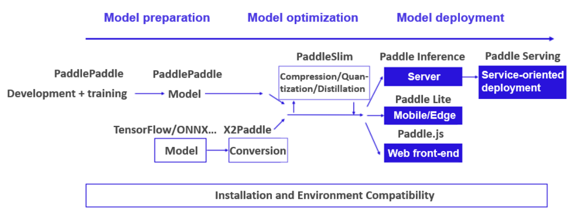

Figure 1: Paddle inference product ecosystem schematic.

direct regression, though it had shortcomings in detecting specific text shapes. PSENet (Wang et al. 2019) used a progressively expanding scale algorithm to enhance the detection of texts with complex shapes.DBNet (Liao et al. 2020) streamlined the detection process with a differentiable binarization module, suitable for various scenarios but still requiring optimization in extreme conditions. CRAFT (Baek et al. 2019) focused on character-level detection, particularly suited for densely populated text scenes. FCENet (Zhu et al. 2021) utilized Fourier contour embeddings to detect complex text shapes without the need for dense annotations. Overall, from early axis-aligned bounding boxes to contemporary arbitrary-shaped text detection, these algorithms each have unique characteristics that collectively advance the field. The choice of an appropriate algorithm should be based on specific applications and requirements.

### Text Recognition

Text recognition technology has seen significant advancements in the OCR field, with numerous studies focusing on enhancing accuracy, speed, and practicality. Semantic Reasoning Networks (SRN) (Yu et al. 2020) achieves precise scene text recognition by integrating multiple modules. VisionLAN (Wang et al. 2021) introduces a visual language modeling network, employing a weak supervision method to generate character-level mask maps. abiNet (Fang et al. 2021) is an autonomous, bidirectional, and iterative model emphasizing explicit language modeling with an iterative correction strategy. VitStr (Atienza 2021), based on the Vision Transformer architecture, balances accuracy, speed, and computational efficiency in scene text recognition. SVTR (Du et al. 2022) proposes a single visual model that achieves competitive accuracy and efficiency without sequence modeling. Together, these studies advance text recognition technology, enhancing OCR systems' capabilities.

### Edge Computing

With the rapid growth of IoT and wireless networks, the number and data volume of edge devices have surged, rendering traditional cloud-based centralized processing models insufficient for efficiently handling edge data (Shi, Pallis, and Xu 2019). In response, edge computing has emerged, deploying computing and storage nodes at the internet's edge. This approach provides fast-response cloud services for mobile computing while ensuring scalability and data privacy in IoT, helping to mitigate brief cloud disruptions (Satyanarayanan 2017). (Du et al. 2020) introduced an ultra-lightweight OCR system, highly suitable for deployment on edge devices. (Du et al. 2021) made improvements to the existing OCR system. PP-OCRv3 (Li et al. 2022), as an OCR foundation framework, is an upgrade of its previous versions, achieving a balance between accuracy and efficiency.

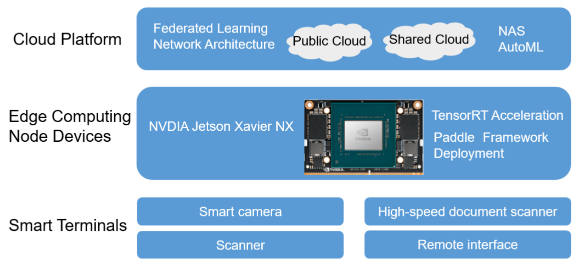

Figure 2: Cloud-Edge collaborative OCR recognition architecture.

## Proposed Solution

### System Design

The technical approach of OCR is outlined from two aspects: architecture and model. Figure 1 shows a schematic of the ecological relationship between Paddle reasoning products. Architecturally, to establish a cloud-edge collaborative structure, PaddlePaddle from Baidu is an open-source deep learning platform developed from industrial practice. It is adaptable for servers, edge devices, and various deployment architectures, encompassing the entire workflow from data annotation and model training to quantization and deployment. It also facilitates the conversion of models from multiple frameworks to the Paddle framework, making it suitable for this scenario with its extensive functionality and support. Paddle is licensed under the Apache License 2.0, an open-source license favorable for commercial use. The only requirement is to include the corresponding License information in the project code, making it suitable for commercial applications. At the model level, OCR operates as a two-stage recognition system, transitioning from text detection to text recognition. The text detection stage uses Differentiable Binarization (DB), while the text recognition stage employs the Convolutional Recurrent Neural Network (CRNN). A text orientation classifier is added between these two modules to manage text recognition in various orientations. The supporting models, PP-OCRv3 and PP-OCRv2, are integrated within the Paddle framework for ease of use.

### System Framework

The OCR system framework is described from three perspectives: cloud, edge, and terminal, as shown in Figure 2.

Cloud Server: Responsible for model training and conversion. With its high computational power and large memory, the cloud server is suitable for extensive data annotation and training of complex models. The Paddle Inference framework is deployed on both server-side and high-performance personal computers. Utilizing the Paddle toolkit, the goal is to minimize model size while maintaining high accuracy, facilitating deployment on edge devices.

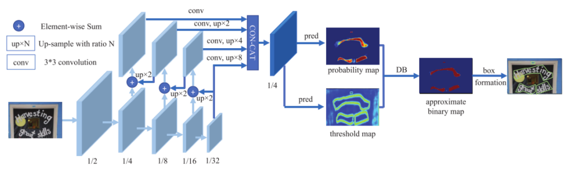
Figure 3: DB (Differentiable Binarization) method architecture.

Edge Device: Handles model deployment and inference. The NVIDIA Jetson series, particularly the Jetson Xavier NX, is ideal for building an edge computing platform due to its small size, low power consumption, and high computational capability. Serving as a mediator between the cloud and terminal, the Paddle Lite framework on the Jetson Xavier NX primarily conducts OCR inference tasks at the edge. The inference process is offline within the NX box, ensuring user privacy without the need for internet data transmission.

Terminal Device: Initiates and receives OCR tasks. Terminals use interfaces to call APIs provided by edge devices, without storing or inferring the recognition model, thus conserving computational and storage resources. They connect to edge devices through interface calls or commands. For OCR recognition, terminals only need to send image samples; the edge device processes these samples and promptly returns the results, completing the inference process.

## OCR Text Detection Algorithm

Differentiable Binarization (DB) algorithm, designed for segmenting scene text, transforms segmentation-generated probability maps into bounding boxes and text regions through a binarization step. Unlike traditional methods using fixed thresholds, DB employs a differentiable operation, integrating binarization into the segmentation network for adaptive thresholding.

The overall DB method is depicted in Figure 3. The DB method processes the input image through the FPN network (with ResNet as the backbone), generating feature maps of varying sizes. These maps are then upsampled by a factor of two and merged to form quarter-sized feature maps. These maps, after Concat operation, create the final feature map F, used to produce probability map P and threshold map T. These maps are processed through a differentiable binarization formula (with k set to 50) to produce a binary map, ultimately generating text detection boxes.

Besides, the DB method possesses differentiable properties, addressing the issue of non-differentiable gradients during training. During gradient backpropagation, the binary cross-entropy loss is given by:

$$\text{CELoss} = -y\log(f(x)) - (1-y)\log(1-f(x)). \quad (1)$$

For the positive sample loss, $l_+$, and negative sample loss, $l_-$, the following expressions are derived:

$$l_+ = -\log\left(\frac{1}{1+e^{-kx}}\right), \quad (2)$$

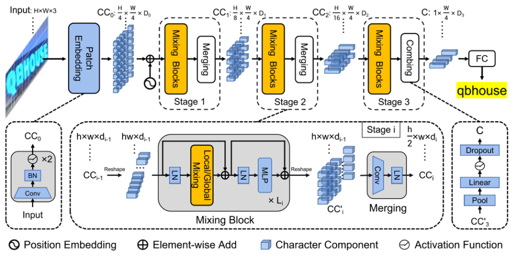
Figure 4: SVTR network structure.

$$l_- = -\log\left(1 - \frac{1}{1+e^{-kx}}\right). \quad (3)$$

## OCR Text Recognition

The SVTR (Segmented Variable Text Recognition) network structure, as depicted in Figure 4, is a specialized, three-stage, decremental network designed for recognizing text characters in images. The network initially takes an image of size $H \times W \times 3$ as input.

In the first stage, the progressive overlapping patch embedding stage, the network uses two consecutive $3 \times 3$ convolutional kernels (with a stride of 2) for patch embedding. This transforms the original image into patches of size $\frac{H}{4} \times \frac{W}{4}$, each with a channel dimension $D_0$, representing ”character components” of the text.

The next stage, the mixing block stage, addresses the need for two types of features in recognition: local component patterns (such as stroke shape features) and inter-character correlations. To capture these, two mixing blocks are utilized: Global Mixing evaluates dependencies between all character components, while Local Mixing assesses correlations within a predefined window among the components.

In the merging stage, the network reshapes the output feature map from the last mixing block into a dimension of $h \times w \times d_i - 1$. It then applies $3 \times 3$ convolutional kernels with different strides on height and width, followed by a layer normalization layer, to produce the final output feature with dimensions $\frac{h}{2} \times w \times d_i$. This reduces the height while maintaining the width.

Finally, in the combining and prediction stage, the network pools the height to 1 and connects a fully connected layer, a non-linear activation layer, and a dropout layer. This compresses the character components into a feature sequence, where each element is a vector of length $D_3$. An $N$-class linear classifier is then used for character recognition, transcribing a sequence of length $\frac{W}{4}$, resulting in a final output dimension of $1 \times \frac{W}{4} \times D_3$. Thus, the SVTR network effectively processes and recognizes text content in images.

## Edge Computing Task Offloading

We define the computational resources at the edge as $f_e$, the computational density (in terms of CPU cycles per bit) as $w_e$, and the size of the task input (in bits) as $t_e$. Thus, the computational delay for offloading an OCR computing task

| **Jetson Xavier NX** | **Specifications** |
|---|---|
| AI Performance | 21 TOPS |
| GPU | 384-core NVIDIA Volta GPU with 48 Tensor Cores |
| CPU | 6-core NVIDIA Carmel ARM v8.2 64-bit CPU, 6MB L2 + 4MB L3 |
| Memory | 8GB 128-bit LPDDR4x, 59.7GB/s |
| Storage | 16GB eMMC 5.1 |
| Power Consumption | 10 Watts — 15 Watts — 20 Watts |
| PCIe | 1 x1 (PCIe 3.0) + 1 x4 (PCIe 4.0), Total 144GT/s |
| CSI Camera | Up to 6 cameras (24 via virtual channels), 14 lanes MIPI CSI-2, D-PHY 1.2 (up to 30Gbps) |
| DL Accelerator | 2x NVIDIA Engines |
| Vision Accelerator | 7-way VLIW Vision Processor |
| Network | 10/100/1000 BASE-T Ethernet |

Table 1: Jetson Xavier NX Performance Parameters

from the terminal to the edge can be represented as:

$$d_c^e = \frac{\omega_e t_e}{f_e}. \tag{4}$$

Task offloading requires data transmission to the edge device through the uplink wireless channel. We consider the Orthogonal Frequency-Division Multiple Access (OFDMA) scheme for task offloading. Let $P_T$ represent the transmission power, $h_e$ denote the signal attenuation coefficient, $l_e$ indicate the distance from the terminal to the edge end, $\alpha$ is the path loss exponent, $N_0$ stands for the noise power, and $I_e^u$ represents interference.

The spectral efficiency of the uplink transmission between the terminal and edge can be expressed as:

$$\eta_e = \log_2 \left(1 + \frac{P_T |h_e|^2 l_e^{-\alpha}}{N_0 + I_e^u}\right). \tag{5}$$

Let $B$ denote the channel bandwidth. Therefore, the wireless transmission delay between the terminal and edge can be represented as:

$$d_e^{ut} = \frac{t_e}{b_e \eta_e}. \tag{6}$$

Next, the task is offloaded, and the result is downloaded back to the terminal. Since the downlink channel for downloading results is unknown, to simplify the system model, we use the average downlink transmission rate as an estimate for the downlink channel. Let $U$ denote the ratio of the result size to the input data size. The wireless transmission delay for downloading results from the edge can be expressed as:

$$d_e^{dt} = \frac{\mu_e t_e}{r_e^d}. \tag{7}$$

Thus, the total system delay can be represented as:

$$D_e(b, f) = d_e^c + d_e^{ut} + d_e^{dt}. \tag{8}$$

## Experiments

### Evaluation Metrics

To comprehensively evaluate the performance of the recognition, we utilizes the Harmonic mean ($H_{\text{mean}}$), also known as the F1-score, on a self-annotated dataset. The $H_{\text{mean}}$ is the harmonic average of precision and recall. Precision is defined as the ratio of correctly predicted positive instances to the total predicted positive instances, while recall is the ratio of correctly predicted positive instances to the total actual positive instances. Additionally, accuracy is introduced as the proportion of all correctly predicted instances (including both positive and negative classes) to the total instances, which is a critical metric for overall performance.

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}, \tag{9}$$

$$H_{\text{mean}} = \frac{2}{\frac{1}{P} + \frac{1}{R}} = \frac{2 \cdot P \cdot R}{P + R}, \tag{10}$$

$$\text{Accuracy} = \frac{TP + TN}{P + N}. \tag{11}$$

### Edge Device

The primary tasks of OCR detection and recognition are performed on edge devices, specifically the Jetson Xavier NX in this project. Its deep learning framework is deployed using Jetpack version 5.0.2. Table 1 details the configuration information of the deep learning libraries under this version, and describes the system performance of the Jetson Xavier NX.

### Model Training

The PP-OCR series models provided by PaddleOCR exhibit good generalizability in common scenarios. In vertical scenarios, if a more optimal model performance is desired, model fine-tuning can be employed to further enhance the accuracy of the PP-OCR series detection and recognition models. In this OCR task, the focus is mainly on the detection and recognition of flat text in invoices. For this specific scenario, the project collects relevant data and conducts further training and fine-tuning based on self-annotated datasets using the PP-OCR pretrained models.

The project utilizes the PP-OCRv3 pretrained model combined with proprietary data for further optimization. For text detection, the DB network pretrained model is used, and for text recognition, the SVTR network pretrained model is employed. Model training is completed on cloud servers, with the training environment configuration detailed in Table 2. During model fine-tuning, incorporating real-world general

| Jetpack | TensorRT | cuDNN | CUDA | OpenCV |
|---|---|---|---|---|
| 5.0.2 | 8.4.1 | 8.4.1 | 11.4.14 | 4.5.4 |

Table 2: Jetpack 5.0.2 Environment Dependencies

scenario data can further enhance model accuracy and generalizability. Over 500 annotated images are used to fine-tune the detection pretrained model, with separate training adjustments for both the detection and recognition models. For the detection model training, batch size is set to 2 and learning rate to 0.00005. For the recognition model training, batch size is set to 32 and learning rate to 0.000025. Other parameters are modified according to specific needs.

### Model Conversion

After training, models undergo quantization and need to be converted into inference models for use in inference tasks. The inference models can be further converted into PaddleLite format models, facilitating deployment on mobile devices. Paddle-Lite is a high-performance, lightweight, flexible, and easily extendable deep learning inference framework. It further optimizes inference models to produce Naive Bayes model suitable for deployment in mobile/IoT environments. It is generally recommended to base the conversion on quantized models, as this allows the model to be stored and inferred in INT8 format, thereby further reducing the model size and improving speed.

The results of the quantized model conversion, including format and size, are illustrated in Table 3.

| **Model** | **Size** |
|---|---|
| Detection inference model | 2.4M |
| Recognition inference model | 5.8M |

Table 3: Quantized Model Conversion Results - Size

| **Model** | $H_{mean}$ |
|---|---|
| pp-ocrv3 Detection Pretrained Model | 52.65% |
| Self-Trained Model | 59.03% |

Table 4: Quantized Model Conversion Results - $H_{mean}$

## Results

### Detection Performance

As demonstrated in Table 4, the self-trained model exhibits a notable improvement in the $H_{mean}$ metric compared to the pre-trained pp-ocrv3 model. Specifically, the Hmean of the self-trained model reached 59.03%, while that of the pre-trained model stood at 52.65%. This quantitative analysis distinctly validates the superior performance of the model trained on a specific dataset. Furthermore, it underscores that tailored optimization for OCR technology on edge devices can significantly enhance its effectiveness in practical applications.

| **Model** | **Accuracy** |
|---|---|
| pp-ocrv3 Recognition Pretrained Model | 82.03% |
| Self-Trained Model | 95.36% |

Table 5: Quantized Model Conversion Results - Accuracy

| **Process** | **Average Duration** |
|---|---|
| Text Detection Computation | 0.088s |
| Text Recognition Computation | 0.219s |
| Uplink Transmission | 0.152s |
| Downlink Transmission | 0.203s |
| **Total Edge Computing Latency** | **0.662s** |
| **End-Terminal Computing Delay** | 1.132s |

Table 6: Comparison of Total Latency for OCR Task in Edge Computing vs. End-Terminal Computing

### Recognition Performance

As shown in Table 5, the self-trained model demonstrated a substantial increase of 13.33% in recognition accuracy over the pre-trained pp-ocrv3 model. This significant improvement in performance indicates that customized training and optimization can greatly enhance the effectiveness of OCR models in practical applications. Such advancements are crucial for scenarios that demand high-precision text recognition, such as automated document processing and real-time text analysis.

### System Latency

Table 6 details average latency at each OCR task stage in edge computing, encompassing text detection, recognition computation, uplink, and downlink transmission. The total edge computing latency is notably 0.662 seconds, significantly surpassing the 1.132 seconds computation delay on end-terminal devices. This efficiency advantage underscores edge computing's effectiveness for OCR tasks, vital for designing responsive real-time OCR systems, particularly in bandwidth-limited or delay-sensitive environments.

## Conclusion

This study aims to efficiently apply OCR technology in edge networks, fully leveraging edge computing as a distributed model in the rapidly advancing landscape of IoT technology. By subdividing the OCR task into multiple sub-tasks and offloading them to edge devices, we effectively optimized for resource-constrained edge devices. This strategy facilitates the deployment of OCR models on edge nodes, enabling immediate recognition of real-time image text. Our approach successfully reduced transmission latency and bandwidth consumption while maintaining high accuracy with a significant decrease in system latency. This research provides valuable support for OCR applications in edge environments and offers insights into the development of IoT and edge computing technologies.

## References

Abdellatif, A. A.; Allahham, M. S.; Khial, N.; Mohamed, A.; Erbad, A.; and Shaban, K. 2023. Reliable Federated Learning for Age Sensitive Mobile Edge Computing Systems. In *ICC 2023 - IEEE International Conference on Communications*, 1622–1627.

Atienza, R. 2021. Vision transformer for fast and efficient scene text recognition. In *International Conference on Document Analysis and Recognition*, 319–334. Springer.

Baek, Y.; Lee, B.; Han, D.; Yun, S.; and Lee, H. 2019. Character region awareness for text detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 9365–9374.

Chai, F.; Zhang, Q.; Yao, H.; Xin, X.; Gao, R.; and Guizani, M. 2023. Joint Multi-Task Offloading and Resource Allocation for Mobile Edge Computing Systems in Satellite IoT. *IEEE Transactions on Vehicular Technology*, 72(6): 7783–7795.

Chi, H. R.; Silva, R.; Santos, D.; Quevedo, J.; Corujo, D.; Abboud, O.; Radwan, A.; Hecker, A.; and Aguiar, R. L. 2023. Multi-Criteria Dynamic Service Migration for Ultra-Large-Scale Edge Computing Networks. *IEEE Transactions on Industrial Informatics*, 19(11): 11115–11127.

Cui, Q.; Zhao, X.; Ni, W.; Hu, Z.; Tao, X.; and Zhang, P. 2023. Multi-Agent Deep Reinforcement Learning-Based Interdependent Computing for Mobile Edge Computing-Assisted Robot Teams. *IEEE Transactions on Vehicular Technology*, 72(5): 6599–6610.

Du, Y.; Chen, Z.; Jia, C.; Yin, X.; Zheng, T.; Li, C.; Du, Y.; and Jiang, Y.-G. 2022. Svtr: Scene text recognition with a single visual model. *arXiv preprint arXiv:2205.00159*.

Du, Y.; Li, C.; Guo, R.; Cui, C.; Liu, W.; Zhou, J.; Lu, B.; Yang, Y.; Liu, Q.; Hu, X.; Yu, D.; and Ma, Y. 2021. PP-OCRv2: Bag of Tricks for Ultra Lightweight OCR System. *arXiv e-prints*, arXiv:2109.03144.

Du, Y.; Li, C.; Guo, R.; Yin, X.; Liu, W.; Zhou, J.; Bai, Y.; Yu, Z.; Yang, Y.; Dang, Q.; and Wang, H. 2020. PP-OCR: A Practical Ultra Lightweight OCR System. *arXiv e-prints*, arXiv:2009.09941.

Fang, S.; Xie, H.; Wang, Y.; Mao, Z.; and Zhang, Y. 2021. Read like humans: Autonomous, bidirectional and iterative language modeling for scene text recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7098–7107.

Kim, G.; Hong, T.; Yim, M.; Nam, J.; Park, J.; Yim, J.; Hwang, W.; Yun, S.; Han, D.; and Park, S. 2022. Ocr-free document understanding transformer. In *European Conference on Computer Vision*, 498–517. Springer.

Koti, A.; Khare, A.; and Khare, P. 2023. An Electronic DIGI White Cane for the Visually-Impaired Personnel. In *International Congress on Information and Communication Technology*, 581–589. Springer.

Li, C.; Liu, W.; Guo, R.; Yin, X.; Jiang, K.; Du, Y.; Du, Y.; Zhu, L.; Lai, B.; Hu, X.; Yu, D.; and Ma, Y. 2022. PP-OCRv3: More Attempts for the Improvement of Ultra Lightweight OCR System. *arXiv e-prints*, arXiv:2206.03001.

Li, M.; Lv, T.; Chen, J.; Cui, L.; Lu, Y.; Florencio, D.; Zhang, C.; Li, Z.; and Wei, F. 2023. Trocr: Transformer-based optical character recognition with pre-trained models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 13094–13102.

Liao, M.; Shi, B.; and Bai, X. 2018. Textboxes++: A single-shot oriented scene text detector. *IEEE transactions on image processing*, 27(8): 3676–3690.

Liao, M.; Wan, Z.; Yao, C.; Chen, K.; and Bai, X. 2020. Real-time scene text detection with differentiable binarization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 11474–11481.

Liu, D.; Sun, F.; Wang, W.; and Dev, K. 2023. Distributed Computation Offloading with Low Latency for Artificial Intelligence in Vehicular Networking. *IEEE Communications Standards Magazine*, 7(1): 74–80.

Reiss-Mirzaei, M.; Ghobaei-Arani, M.; and Esmaeili, L. 2023. A review on the edge caching mechanisms in the mobile edge computing: A social-aware perspective. *Internet of Things*, 100690.

Satyanarayanan, M. 2017. The Emergence of Edge Computing. *Computer*, 50(1): 30–39.

Shi, W.; Pallis, G.; and Xu, Z. 2019. Edge Computing [Scanning the Issue]. *Proceedings of the IEEE*, 107(8): 1474–1481.

Tian, Z.; Huang, W.; He, T.; He, P.; and Qiao, Y. 2016. Detecting Text in Natural Image with Connectionist Text Proposal Network. arXiv:1609.03605.

Wang, W.; Xie, E.; Li, X.; Hou, W.; Lu, T.; Yu, G.; and Shao, S. 2019. Shape Robust Text Detection with Progressive Scale Expansion Network. arXiv:1903.12473.

Wang, Y.; Xie, H.; Fang, S.; Wang, J.; Zhu, S.; and Zhang, Y. 2021. From two to one: A new scene text recognizer with visual language modeling network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 14194–14203.

Yu, D.; Li, X.; Zhang, C.; Liu, T.; Han, J.; Liu, J.; and Ding, E. 2020. Towards accurate scene text recognition with semantic reasoning networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 12113–12122.

Zhou, X.; Yao, C.; Wen, H.; Wang, Y.; Zhou, S.; He, W.; and Liang, J. 2017. EAST: An Efficient and Accurate Scene Text Detector. arXiv:1704.03155.

Zhu, Y.; Chen, J.; Liang, L.; Kuang, Z.; Jin, L.; and Zhang, W. 2021. Fourier contour embedding for arbitrary-shaped text detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 3123–3131.