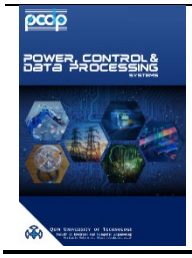
DOI: 10.30511/pcdp.2025.2052926.1018 - ISSN: 3060-8376 - Vol 2(1), 2025



POWER, CONTROL AND DATA PROCESSING SYSTEMS

Available Online at: https://pcdp.qut.ac.ir/

Empirical Evaluation of Well-known Farsi OCR Engines on the IDPL-PFOD Dataset

ARTICLE INFO

Article Type
Original Research

Authors
Fatemeh-sadat Hosseini[1]
Elham Shabaninia*[,2]
Hossein Nezamabadi-pour[3]

[1] Department of Electrical Engineering, Shahid Bahonar University of Kerman, Kerman, Iran, ftmsdt98@gmail.com
[2] Department of Applied mathematics, Graduate University of Advanced Technology, Kerman, Iran, e.shabaninia@kgut.ac.ir
[3] Department of Electrical Engineering, Shahid Bahonar University of Kerman, Kerman, Iran, nezam@uk.ac.ir

**** Correspondence***
Address: Department of Applied mathematics, Graduate University of Advanced Technology, Kerman, Iran
e.shabaninia@kgut.ac.ir

Article History
Received: February 08, 2025
Accepted: March 18, 2025
ePublished: March 30, 2025

ABSTRACT

Optical character recognition (OCR), also referred to as text recognition, extracts text from scanned documents, camera images, etc. OCR has numerous applications in reading forms and cheques; converting archived documents to digital files, reading books and papersetc. An accurate OCR system speeds up these processes by removing time-consuming user tasks. However, OCR is challenging especially in languages such as Farsi due to the intrinsic characteristics of this language and limited resources such as suitable datasets to evaluate the effectiveness and efficiency of proposed methods. IDPL-PFOD is a new synthetic Farsi printed dataset that offers a wide range of variations including different backgrounds, Font types, distortions, blurs, etc. Therefore, in this paper, two OCR engines, Tesseract and EasyOCR are evaluated on the IDPL-PFOD dataset to show the limitations of existing OCR engines for Farsi. Evaluations using standard metrics reveal that Tesseract and EasyOCR respectively achieve an overall accuracy of 84.41% and 73.28% on this dataset. Furthermore, the robustness of these two engines is evaluated against different variations such as textured background, salt & pepper noise, Gaussian blur, and distortions. This paper provides valuable insights to the community by reviewing the current challenges of deep learning methods for Farsi OCR and serving as a foundation for further research and advancements in the future.

Keywords: IDPL-PFOD dataset, Optical Character Recognition (OCR), EasyOCR, Tesseract, OCR engines, Farsi OCR.

1 Introduction

The exponential growth in data generation across sectors like business, healthcare, education, and entertainment has driven the need for data digitization. The adoption of digital devices and platforms has resulted in the creation of vast amounts of digital information, offering cost savings, efficiency benefits, and faster data processing. This has also sparked interest in computer vision and machine learning, particularly in the field of OCR, which focuses on detecting and converting text from printed or handwritten images into computer-friendly formats for processing, storage, editing, and search purposes [1-3].

OCR has a broad range of applications, including invoice data analysis [4], the legal industry [4], banking [4], healthcare [4], captcha reading [5], digital libraries [6], and reading bank checks [7]. It is also extensively utilized in image-based applications such as image searching [8], intelligent inspection [9], industrial automation [10], and robotic routing [11]. As a result, OCR is a well-researched topic in the field of computer vision. While OCR research has been predominantly focused on languages like English, progress has been comparatively slower for languages like Farsi due to the unique characteristics of the language [12]. Here are some key characteristics of Farsi script:

- Farsi is written using a script of 32 characters, and the writing direction is right to left [13].
- There are several pairs of characters in Farsi that have a similar appearance, differing only in the number and placement of dots [14].
- Farsi is also a cursive script, meaning its characters attach in writing [15].
- Farsi characters can have up to four writing styles: beginning, middle, ending, and isolated [16].
- In Farsi, each character has the potential to connect with other characters on one or both sides, depending on their position within a word. This can lead to the formation of "sub-words," which are the separate parts of a word that are written individually [12].
- Different fonts can result in variations in the written style of Farsi characters [17].
- Farsi lacks capital letters [18].
- While numbers are written from left to right, Farsi words, sentences, and dates are written from right to left [19].
- The presence of numerous semicircles, known as "Dandane," in some words poses a challenge for OCR processing [20].
- The incorporation of foreign words in Farsi texts adds complexity to Farsi OCR [20].
- Moreover, the writing style of numbers may differ depending on the writer's literacy level and location [21].
- Farsi and Arabic share many similarities [15], but they have distinct grammatical structures, pronunciation differences, variations in vocabulary, diacritical mark usage, and writing conventions.

Despite the challenges posed by the complex characteristics of Farsi, recent advancements in deep learning methods have significantly improved OCR accuracy [12]. However, the availability of limited datasets in Farsi for training and validation remains a hurdle in further enhancing OCR performance for this language [17]. The IDPL Printed Farsi OCR Dataset (IDPL-PFOD) [17] is a recently introduced synthetic dataset that provides a diverse range of variations, including different backgrounds, font types, distortions, blurs, and more. This dataset aims to facilitate the evaluation of various OCR techniques by offering a comprehensive set of test cases for Farsi OCR.

On the other hand, although there are challenges, there are some well-known OCR engines that support the Farsi language. One such engine is Tesseract [22], an open-source OCR engine developed by Google. Tesseract is known for its accuracy and robustness in converting images containing printed or handwritten text into machine-readable text. It utilizes deep learning techniques and advanced algorithms to analyze image patterns and recognize characters, making it suitable for Farsi OCR tasks. Tesseract can be customized and integrated into various platforms and applications for text recognition purposes.

Moreover, another notable OCR solution is EasyOCR [23], an open-source Python library that simplifies OCR. It supports multiple languages, including Farsi, and offers pre-trained models for text recognition. EasyOCR enables developers to easily integrate OCR capabilities into their applications without extensive coding or training requirements. It supports various image formats and provides options for adjusting OCR settings like confidence threshold and language selection. EasyOCR is widely used for tasks such as document processing, text extraction, and data analysis.

However, despite the availability of these OCR engines, they have not been thoroughly evaluated due to the lack of suitable datasets for testing and comparing different methods. It is essential to examine the weaknesses of previous approaches. To address this problem, the purpose of this paper is to assess two Farsi OCR engines, Tesseract and EasyOCR, using the IDPL-PFOD synthetic dataset. The objective is to identify the unsolved challenges of these engines and provide a benchmark for evaluating Farsi OCR methods. This evaluation will assist Farsi OCR researchers in enhancing the accuracy of these engines and overcoming the existing challenges. Additionally, we aim to investigate the weaknesses of the IDPL-PFOD dataset and enhance it to meet appropriate standards in future work.

The structure of each section is as follows: Section 2: This section provides an overview of OCR models and OCR engines. Section 3: The capabilities and features of the IDPL-PFOD dataset are presented in this section. Section 4: The architecture and features of the OCR engines are discussed in this section. Section 5: This section introduces the metrics that will be utilized for evaluating the accuracy of the OCR engines. Section 6: The experimental results obtained from evaluating the two remarkable OCR engines are presented in

this section. Section 7: In this section, the strengths and weaknesses of both OCR engines and the IDPL-PFOD dataset are discussed, and any unresolved challenges for improvement are identified. Finally, in the concluding part, our findings are summarized, and their significance is highlighted.

2 Related Work

OCR models are the core approach, algorithms, and methodologies within OCR systems that perform text recognition using machine learning techniques [24]. They focus on accurately transcribing text from images. In contrast, OCR engines encompass the entire software ecosystem, integrating OCR models with preprocessing, recognition, and post-processing components. They offer a comprehensive solution for various text recognition tasks, handling image enhancement, layout analysis, and output formatting, making them more versatile tools in practical applications [22].

2.1 OCR models

Following the introduction of Transformers in vision-related tasks [25], optical character recognition (OCR) has incorporated a variety of deep learning techniques to enhance its accuracy. These techniques can be categorized into two groups: Transformer-less methods and Transformer-based methods. Within the Transformer-less category, there are distinctions between CRNN-based and CRNN attention-based methods. Furthermore, the Transformer-based category can be subdivided into Convolutional-less (vanilla) Transformer and Convolutional-based Transformer (CvT).

CRNN-based methods utilize Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) networks, along with their combination, for recognition purposes. CNNs are chosen due to their ability to capture patterns and spatial features from images, applied to character-level and line-of-text recognition [26]. In Farsi OCR, CNN architectures including LeNet [27], AlexNet [28], VGG [29], ResNet [30], and DenseNet [31] have been adapted. RNNs manage sequential data for text recognition. Long Short-Term Memory (LSTM) networks are used in Farsi OCR [32]. while the Gated Recursive Unit (GRU) is preferred for variable-length sequences, as shown in English OCR by [33]. CRNN (Convolutional Recurrent Neural Network) merges CNNs and RNNs into a unified architecture. CNNs extract image features, followed by RNNs for handling sequential data [34]. CRNN excels in text-line recognition. Furthermore, Connectionist Temporal Classification (CTC) manages variable-length sequences without alignment, often paired with CRNN-based architectures for transcribing text [23].

CRNN attention-based methods share similarities with the prior one, substituting the CTC technique with an attention mechanism that guides the model's focus during text generation from input images. Incorporating a trainable feedforward layer with weight vectors, it predicts an image's pixels or sentences' words, potentially enhancing OCR accuracy by refining alignment between visual features and textual content. Ref. [35] used these methods for OCR of the street images.

In [36], the Transformer architecture is employed for scene-text recognition. It is a deep neural network designed for sequences like text or images, comprising an encoder and decoder. The encoder handles the input sequence, while the decoder generates the output sequence. The Transformer's significant advancement is the self-attention mechanism, allowing it to focus on various parts of the input sequence. The encoder produces a sequence of feature vectors, which the decoding network transforms into characters or words.

Convolutional-based Transformer methods combine CNNs and transformers, CNNs extract image features, while transformers capture character context and dependencies. The transformer contains self-attention and feed-forward layers, generating feature vectors. These vectors pass through a fully connected layer to select the highest probability character as recognition for the corresponding image position using a probability distribution. This method is used for historical handwritten text recognition by ref. [37].

2.2 OCR Engines

Numerous OCR engines have emerged in recent times [38]. Here, a selection of them is introduced.

Tesseract [22]: developed from 1985 to 1994 by Hewlett-Packard. The latest version, V5.03.01 can be found on Tesseract GitHub. Tesseract employs a combination of CNNs and RNNs, particularly LSTM networks for character recognition. It involves multiple processing stages including image preprocessing, text line segmentation, feature extraction using CNNs, and sequential processing using LSTMs.

EasyOCR [23]: offered by Jaided AI. The latest version, 1.6.2, was released on September 15, 2022. The developer site provides a demo of this engine. Built on PyTorch, EasyOCR is a combination of CNNs and RNNs, particularly Bidirectional Long Short-Term Memory (Bi-LSTM) networks, which capture past and future contextual information for each position in the feature maps, for character recognition. And CTC network to align between the input image and the sequence without explicit alignment annotation. The architecture involves image preprocessing, feature extraction using CNNs, and post-processing for accurate text recognition.

OCRopus [39]: Based on LSTM networks, Ocropus focuses on layout analysis and text recognition in images. It involves preprocessing, layout analysis, and sequential text recognition using LSTMs.

Kraken [40]: tailored for historical documents and complex text recognition, employs neural networks and advanced methods for fonts, layouts, and languages. It stands as a potent OCR engine adept at historical and challenging document text recognition, supported by neural networks, layout analysis, language variety, and an open-source approach.

GOCR [41]: is an open-source OCR engine, that employs a basic recognition method through template matching. utilizes template matching for basic recognition. While effective for simple text extraction, it may face difficulties when handling more complex documents.
Ocrad [42]: An open-source OCR engine, Ocrad specializes in recognizing machine-printed text through OCR techniques. It combines methods like line segmentation, character recognition, and dictionary-based analysis to achieve accurate results.
Ocular [43]: is an advanced historical OCR system, learning from unknown fonts using images and text. It adeptly handles challenges like noisy documents and supports multilingual content. Ocular excels at learning orthographic variations and transcribes them into both diplomatic (literal) and normalized forms. Combining traditional OCR with deep learning, it employs CNNs for character recognition and a versatile architecture for diverse fonts and languages.
SwiftOCR [44]: stands as a free and open-source OCR engine designed for iOS, macOS, and Apple platforms. It leverages deep learning, particularly CNNs, for rapid text identification within images. Swiftly implemented in Swift, this OCR library employs a neural network for image recognition. Focused on short alphanumeric codes, its training, accuracy, preprocessing, and segmentation enhance adaptability and precision. Moreover, the library's scope has expanded to encompass lowercase characters and connected character segmentation capabilities.
Attention-OCR [45]: is a deep learning-powered OCR model that capitalizes on attention mechanisms to enhance text recognition. It begins by applying a sliding CNN to images, resized to a height of 32 while maintaining aspect ratios. An LSTM is then layered atop the CNN, followed by an attention model serving as a decoder for generating the ultimate results. This approach enables the model to concentrate on pertinent portions of an image, facilitating accurate text identification. By accommodating intricate layouts and varying text sizes, Attention-OCR excels in handling diverse recognition tasks.
RWTH-OCR [46]: from RWTH Aachen University, blends traditional OCR with modern machine learning. It covers preprocessing, character recognition, and language modeling. With an OCR decoder and visual model tools, it's from RWTH Aachen University's Human Language Technology and Pattern Recognition Group. Successful in research, it offers features like OCR decoding, adaptable processing, visual modeling, Gaussian distributions, language modeling, and adaptations.
Simple-OCR-OpenCV [47]: an open-source OCR engine utilizing the OpenCV library, applies image processing techniques including thresholding, contour detection, and character segmentation. It also involves segmentation using preprocessing to identify character regions in images for OCR. Moreover, the project employs supervised learning using the k-Nearest Neighbors (k-NN) algorithm to recognize characters within image segments.
Calamari [48]: is an OCR engine that harnesses advanced deep learning models like LSTM and CNN architectures. It possesses the capability to process various image preprocessing techniques and is proficient in recognizing both printed and handwritten text. Designed for flexibility, Calamari provides a platform for experimentation and exploration in the field of OCR tasks, making it a valuable tool for researchers and practitioners alike.
Doctr [49]: is a specialized Python library designed for document analysis and OCR operations, seamlessly incorporating robust OCR engines such as Tesseract and OCRopus, supplemented by layout analysis utilities. This amalgamation empowers the extraction of textual content and organized data from documents, rendering it applicable across a spectrum of document processing contexts. Rooted in deep learning principles, Doctr employs advanced models like CNNs or Transformers for both feature extraction and sequence-to-sequence modeling, underpinning its capacity for sophisticated OCR endeavors.
PaddleOCR [50]: stands as a comprehensive OCR toolkit built on PaddlePaddle, an open-source deep learning platform. It excels in recognizing and extracting text from images with remarkable accuracy and efficiency. PaddleOCR boasts a wide array of functionalities, including text detection, recognition, layout analysis, and more. Its versatility supports various languages and fonts, making it suitable for diverse document types. The toolkit leverages advanced techniques such as attention mechanisms and transformer-based models to enhance recognition performance.
KerasOCR [51]: employs CNNs for feature extraction and LSTM networks for sequence modeling, often using techniques like CTC loss for text length variations. With its comprehensive training, combined with data preprocessing and character decoding, KerasOCR offers a strong solution for OCR tasks within the Keras deep learning environment.
Older and Abandoned OCR Engines: Several older and abandoned OCR engines persist, bearing titles like ClaraOCR [52], CuneiForm [53], EyeOCR [54] , Kognition [55] , OCRchie [56], ocre [57], Xplab [58] , and hebOCR [59]. The architectures of these older OCR engines vary, but they generally employ traditional methods, like pattern matching, for character recognition and text extraction. Some might also involve preprocessing steps for enhancing image quality.
Of the mentioned OCR engines, Tesseract, EasyOCR, Kraken, and PaddleOCR support the Farsi language. These engines offer different levels of support for Farsi text recognition and extraction. However, in this paper, we focus on Tesseract and EasyOCR engines due to their popularity.

3 IDPL-PFOD Dataset

Introducing a benchmark, which serves as a standard dataset for evaluation, plays a crucial role in the development, testing, and comparison of various OCR methods. It enables researchers in the field to evaluate and compare their methods

fairly. Therefore, the initial step in conducting deep-based OCR research in any language involves the collection of a diverse and comprehensive benchmark [15].

OCR datasets are categorized based on their application into handwritten, printed, and scene-text datasets. Handwritten datasets are generated by photographing or scanning handwritten text images, printed datasets consist of scanned or photographed printed text images, and scene-text datasets contain text images with complex or patterned backgrounds [16, 17], Additionally, datasets can be classified into real and synthetic (artificial) based on their generation method. Real datasets are created by scanning real text images, while synthetic datasets are generated using software that incorporates ready-made texts, different fonts, noises, and backgrounds randomly [12].

The IDPL-PFOD dataset is an example of a synthetic printed Farsi text dataset. It comprises 30,138 images with dimensions of 50 x 700 pixels, stored in ".tif" format. Each image contains a line of text, with approximately 15 words per line. In contrast, other Farsi print synthetic datasets typically contain only single words [60], meaningless words [16], sub-words[61], and pages [62] per image. This unique characteristic of the IDPL-PFOD dataset allows for the examination of OCR methods' performance on real data.

The IDPL-PFOD dataset offers a wide range of variations including:

1. Different backgrounds: The dataset incorporates various background types (see Fig. 1, 2, and 3), such as plain white, noisy backgrounds (Gaussian, Speckle, Salt & Pepper, and Poisson noises), and textured backgrounds (12 different patterns). This allows for the scrutiny of each OCR method's robustness against noise and texture backgrounds, and the assessment of their ability to recognize text in the presence of noisy and textured backgrounds.
2. Different Font sizes and types: The dataset utilizes 11 fonts (Badr, Compset, Lotus, Mitra, Nazanin, Roya, Traffic, Titr, Yagut, Yekan, and Zar), 7 font sizes (ranging from 10 to 16), and 2 font styles (Bold and Normal) of widely used Farsi fonts. This investigation aims to determine whether OCR methods have been trained to recognize widely used Farsi fonts, as Farsi letters may have different shapes depending on the writing font. Additionally, texts may vary in font sizes, so the dataset includes text sizes between 10 and 16 to accommodate training with different sizes. The performance of each OCR method on each font size and style can be assessed accordingly.
3. Different distortions: Images are not always in optimal conditions for recognition, and text may exhibit various distortions (see Fig. 2, 4, and 5). The dataset considers three types of distortions: positive slope, negative slope, and sinewave distortion. This allows for the evaluation of OCR methods' robustness against different types of distortion.
4. Blurring: Blurring is another aspect that hampers recognition in real-world scenarios. To assess OCR methods' robustness against blurring, some images in the dataset are intentionally blurred using Gaussian blur with 0.75 raid (see Fig. 1 and 4).

The IDPL-PFOD dataset's diverse image variations make it unique among the publicly available datasets. (See Table 1). In this dataset, each image's information along with its Ground Truth is stored, facilitating easy separation of images based on the aforementioned aspects. Researchers can easily calculate accuracies for each group and gain a better understanding of their methods' strengths and weaknesses.

Tables 2 and 3 provide a summary of the image features in the IDPL-PFOD dataset. Considering one of the four types of noise and one of the twelve types of texture as a single type, the dataset encompasses 18 different image types. Thus, it covers almost all possible states that real data can exhibit. Table 4 lists these 18 different types.

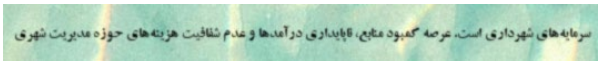


Figure 1: Background: Texture, Font: B Zar Bold, Font size: 13, Distortion: None, Blur: Gaussian.

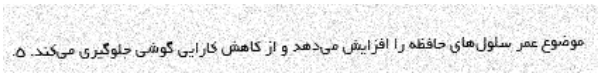


Figure 2: Background: Noisy(gaussian), Font: B Yekan, Font size: 16, Distortion: Sloping (1 degree), Blur: None.

Figure 3: Background: Plain white, Font: B Titr Bold, Font size: 16, Distortion: None, Blur: None.

Figure 4: Background: Texture, Font: B Yagut, Font size: 13, Distortion: Sloping (-1 degree), Blur: Gaussian.

Figure 5: Background: Plain white, Font: B Yekan, Font size: 13, Distortion: Sinewave, Blur: None.

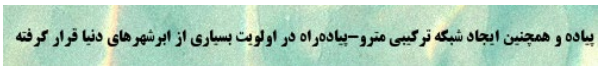


Figure 6: Background: Texture, Font: B Titr Bold, Font size: 14, Distortion: None, Blur: None.

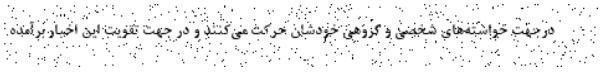


Figure 7: Background: Noisy (S&P), Font: B Nazanin, Font size: 13, Distortion: None, Blur: None.

Table 1: Summary of publicly available Farsi printed datasets.

Name	Samples	Type of content	Year	white	Noisy	Texture
AUT-PFT	10,000	Meaningless words	2015	✓	✓	×
Persian Sub words	19,782	Sub words	2018	✓	×	×
Shotor	120,000	Words	2020	✓	×	×
Arshasb	33,000	Pages	2021	✓	×	×
IDPL-PFOD	30,138	Lines (part of sentences)	2021	✓	✓	✓

Table 2: Summary of the features of IDPL-PFODs' images.

Features	Present	Numbers
Plain white background	50%	15,070
Noisy background	40%	12,056
Texture background	10%	03,012
Total	100%	30,138

Table 3: Summary of the features of IDPL-PFODs' images.

Features	Present	Numbers
None blur, None distortion	90%	27,126
None blur, Sloping distortion	4%	1,210
Gaussian blurring, None distortion	3%	900
Gaussian blurring, One type of distortion	2%	594
None blur, Sinewave distortion	1%	308
Total	100%	30,138

Table 4: 18 different types of IDPL-PFODs' images.

Row	Background			Distortion		Blur
	Plain white	Noisy	Texture	Slope	Sinewave	
01	✓	×	×	×	×	×
02	✓	×	×	✓	×	×
03	✓	×	×	×	✓	×
04	✓	×	×	×	×	✓
05	✓	×	×	✓	×	✓
06	✓	×	×	×	✓	✓
07	×	✓	×	×	×	×
08	×	✓	×	✓	×	×
09	×	✓	×	×	✓	×
10	×	✓	×	×	×	✓
11	×	✓	×	✓	×	✓
12	×	✓	×	×	✓	✓
13	×	×	✓	×	×	×
14	×	×	✓	✓	×	×
15	×	×	✓	×	✓	×
16	×	×	✓	×	×	✓
17	×	×	✓	✓	×	✓
18	×	×	✓	×	✓	✓

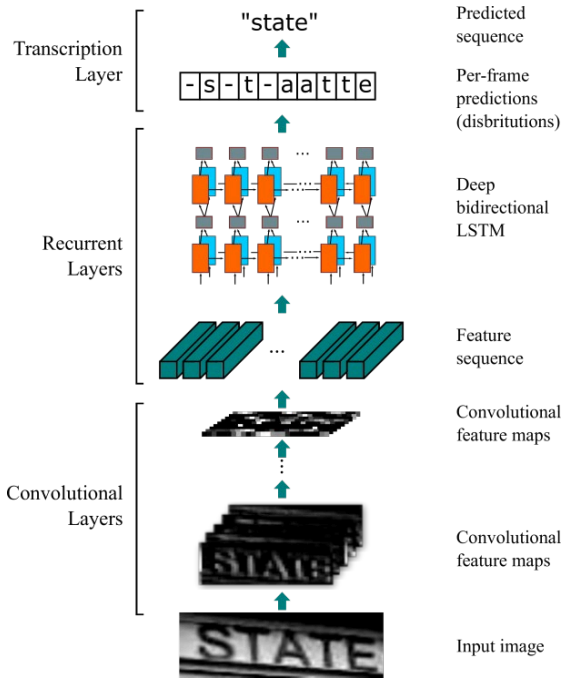


Figure 8: CRNN Network Architecture. [25]

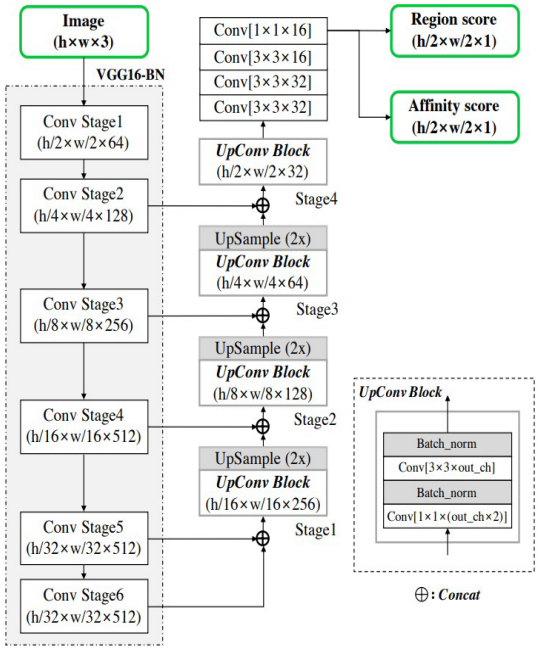


Figure 9: CRAFT Network Architecture. [66]

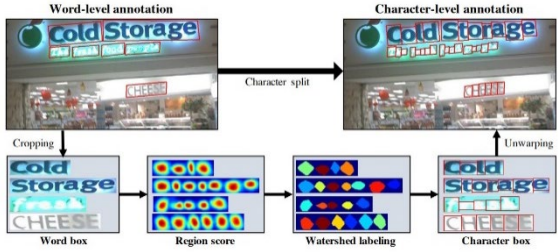


Figure 10: Character-level annotation from word-level annotation. [66]

4 Engines architecture

As mentioned above, we utilize Tesseract and EasyOCR engines for evaluations due to their popularity and widespread use. In this section, the architecture of these engines is reviewed.

4.1 Tesseract

The latest version of Tesseract, Tesseract 5.0, introduced a neural network architecture based on LSTM for text recognition. LSTM is well-suited for sequence data processing, making it particularly effective for recognizing and processing text.

The network architecture in Tesseract 5.0 employs a combination of convolutional and recurrent layers. The convolutional layers are responsible for extracting local image features, while the recurrent layers, specifically LSTM layers, capture the temporal dependencies and long-range context within the text.

The LSTM-based architecture in Tesseract 5.0 has shown significant improvements in accuracy and performance

compared to previous versions. It benefits from the ability to handle longer sequences and capture contextual information effectively, making it more robust in recognizing complex text patterns and challenging fonts.

4.2 EasyOCR

EasyOCR utilizes the CRNN model (Fig. 8) for text recognition and the CRAFT model for text layout detection (Fig. 9). The CRNN architecture comprises three key components: a CNN for extracting features, a recurrent layer for modeling sequences, and a transcription layer for decoding sequences. An input image is subjected to convolutional layers to extract visual features. CNN feature maps are then fed into an iterative layer to model sequential dependencies in the image. A Bi-LSTM layer is used, which includes forward and backward LSTM units. This Bi-LSTM captures past and future contextual information for each position in the feature maps. The recurrent layer outputs a sequence of feature vectors representing the input image. The transcription layer maps these vectors to the corresponding character sequence using a CTC loss function to decode the sequence. CTC enables the network to learn the alignment between the input image and the sequence without explicit alignment annotation [23].

The character region awareness for text detection (CRAFT) model is designed for accurate detection and recognition of text in natural scenes. CRAFT helps in localizing individual characters within the scene, and employs a two-step process: generating a character region score map using a fully convolutional network to assess pixel likelihood for character regions, and refining initial proposals through bounding box regression, to precisely localize text regions in the image [63]. In Figure 10, the process of character segmentation is depicted. Initially, word images undergo cropping. Subsequently, the region score is forecasted, enabling the segregation of character regions through the employment of the watershed algorithm. Finally, bounding boxes are generated, and the coordinates are subsequently transformed to align with the original image [63].

5 Evaluation Metrics

Evaluation metrics can be classified into two primary categories, character level, and word level. Each of these categories encompasses different sub-categories [12].

5.1 Character-Level Metrics

5.1.1 Levenshtein Distance

The Levenshtein distance is a precise measure for assessing the performance of an OCR engine at the character level. This distance metric calculates the disparity between two string sequences by determining the minimum number of single-character (or word) edits (i.e., insertions, deletions, or substitutions) required to transform one sequence into another [12]. Consequently, the greater the dissimilarity between the two text sequences, the higher the number of edits needed, resulting in a larger Levenshtein distance. Conversely, a lower Levenshtein distance indicates better performance of the OCR engine.

Equation (1) provides a means to obtain the Levenshtein distance, while equation (2) defines the Levenshtein Accuracy. It is important to consider three types of errors: insertion, deletion, and substitution. An insertion occurs when an additional character is introduced that was not present in the original text. Conversely, a deletion arises when a character is omitted from the transcript entirely. Lastly, substitution takes place when a character is mistakenly replaced. To illustrate this concept, consider the following example:

OCR Output:	This sentenc contains alll three types of nisrecognized words
Ground Truth:	This sentence contains all three types of misrecognized words

In the word *"sentence"* the character *"e"* is not recognized and must be inserted. In the word *"all", three* instances of the letter *"l"* are incorrectly recognized, with one extra letter that should be deleted. In the word *"misrecognized"* the letter *"n" is* mistakenly recognized and needs to be substituted with *"m"*.

$$Levenshtein\ Distance = \frac{Number\ of\ Insertion+Deletion+substitution}{Total\ length\ of\ Ground\ Truth} \quad (1)$$

$$Levenshtein\ Accuracy = \left(1 - \frac{Number\ of\ Insertion+Deletion+substitution}{Total\ length\ of\ Ground\ Truth}\right) * 100 \quad (2)$$

5.1.2 Character Error Rate (CER)

The calculation of the Character Error Rate (CER) relies on the Levenshtein distance concept. It represents the percentage of characters that require correction, such as insertions, deletions, or substitutions, in the text recognized by the OCR engine. Therefore, a lower CER indicates better performance of the OCR engine [64]. If the OCR output contains more characters than the Ground Truth, the CER will exceed 100%. Overall, the accuracy of the OCR engine, as measured by this metric, can be obtained using equation (3).

$$CER = \left(\frac{Number\ of\ Insertion+Deletion+substitution\ characters}{Total\ number\ of\ characters\ in\ Ground\ Truth}\right) * 100 \quad (3)$$

5.1.3 Character Accuracy

Character Accuracy refers to the proportion of characters accurately recognized by the OCR engine compared to the total number of characters in Ground Truth [65]. Therefore, the Character Accuracy of the OCR engine can be calculated using equation (4).

$$Character\ Accuracy = \left(\frac{Number\ of\ correct\ characters}{Total\ number\ of\ characters\ in\ Ground\ Truth}\right) * 100 \quad (4)$$

5.1.4 Character Precision

The Character Precision metric measures the accuracy of the OCR engine by calculating the ratio of correctly recognized characters to the total number of characters in the Output File. Equation (5) is used to determine the Character Precision of the OCR engine based on this metric.

$$Character\ Precision = \left(\frac{Number\ of\ correct\ characters}{Total\ number\ of\ characters\ in\ Output\ File}\right) * 100 \quad (5)$$

5.2 Word-Level Metrics

5.2.1 Word Error Rate (WER)

The Word Error Rate (WER) represents the proportion of words in the recognized text that require correction, including insertions, deletions, or substitutions, as compared to the Ground Truth [66]. When the OCR engine produces more words than the Ground Truth, the WER exceeds 100%. Equation (6) is used to calculate the accuracy of the OCR engine using this metric.

$$WER = \left(\frac{Number\ of\ Insertion+Deletion+substitution\ words}{Total\ number\ of\ words\ in\ Ground\ Truth}\right) * 100 \quad (6)$$

5.2.2 Word Accuracy

The OCR engine's accuracy can be determined by calculating the Word Accuracy, which is the ratio of the correctly recognized words to the total number of words in the Ground Truth [65]. Equation (7) can be used to obtain the accuracy of the OCR engine using this metric.

$$Word\ Accuracy = \left(\frac{Number\ of\ correct\ words}{Total\ number\ of\ words\ in\ Ground\ Truth}\right) * 100 \quad (7)$$

5.2.3 Word Precision

The OCR engine's accuracy can be determined by using equation (8), which calculates the Word Precision. Word Precision is the ratio of the number of words correctly recognized by the OCR engine to the total number of words in the Output File [65].

$$Word\ Precision = \left(\frac{Number\ of\ correct\ words}{Total\ number\ of\ words\ in\ Output\ File}\right) * 100 \quad (8)$$

In all the evaluation methods mentioned the performance of an OCR engine is determined based on the "distance" between the Ground Truth and the text recognized by the OCR engine. The performance of the OCR engine has an inverse relationship with the CER and WER metrics and Levenshtein distance, and it has a direct relationship with the rest of the metrics. In other words, the lower the CER or WER metrics or Levenshtein distance, the better the performance of the OCR engine, and the higher the other metrics, the better the performance of the OCR engine. Experimental Results

This section presents the empirical results conducted on the IDPL-PFOD dataset using the aforementioned OCR engines. The experiments are conducted on a system with an Intel (R) Core (TM) i7 processor running at 4 GHz, 24 GB RAM, and the Linux operating system (Ubuntu 22.04 distribution). The experiments for the Tesseract engine utilized version 5.0 with the *pytesseract* Python module and the configuration *"--oem 3 --psm 6"*. Similarly, the experiments for the EasyOCR engine utilized version 1.6.2 with the *easyocr* Python module, and the configuration *"text_threshold = 0, link_threshold = 0, detail = 0, blocklist = ';', paragraph=True"*. The code for calculating the accuracy can be found in the IDPL-PFOD GitHub repository.

5.3 Total Accuracy

Figure 11 displays the total accuracy of Tesseract and EasyOCR engines for all metrics. Across all metrics, the Tesseract engine outperformed the EasyOCR engine. The difference in accuracy between the two engines was approximately 8.8% for character-level metrics and approximately 26.96% for word-level metrics. Consequently, when comparing OCR engine performance, both character-level and word-level metrics yield similar results. However, the accuracy difference between the OCR engines is more pronounced when using word-level metrics, indicating that they are more reliable for performance comparison [65].

It is worth noting that even a single character recognition error in a word classifies the entire word as incorrect. The difference in accuracy between the two OCR engines, as measured by the Levenshtein accuracy metric, is approximately the average difference between character-level accuracy and word-level accuracy. Therefore, we focused solely on the Levenshtein metric to identify the key aspects influencing OCR engine accuracy.

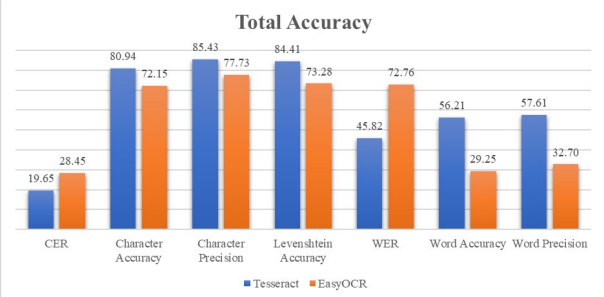


Figure 11: Total Accuracy of Tesseract and EasyOCR engines for all metrics.

5.4 Robustness against different backgrounds

Various aspects can influence the accuracy of OCR, including the background. To evaluate how OCR engines perform on different background types in the IDPL-PFOD dataset, the average accuracy for each background was examined. Figure 12 presents the results for each OCR engine. It is evident that,

except for texture, the Tesseract engine outperforms EasyOCR across all background types. This is expected since EasyOCR is specifically trained for scene-text images. However, when considering only the EasyOCR engine, it performs best on plain white backgrounds. The accuracy remains consistent for backgrounds with Gaussian, Speckle, and Poisson noises, but significantly drops for noisy salt & pepper backgrounds. Consequently, the EasyOCR engine struggles to recognize text on noisy salt & pepper backgrounds. To address this, we recommend developers train this engine using Noisy salt & pepper backgrounds.

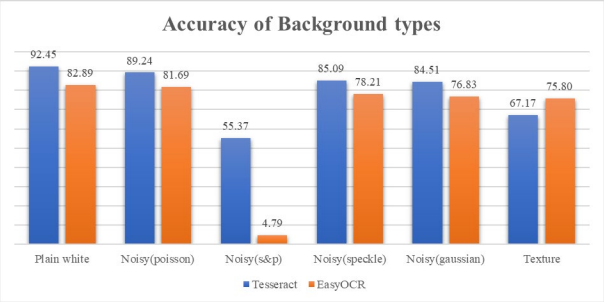


Figure 12: Accuracy of Background Types for Tesseract and EasyOCR engines for all metrics.

On the other hand, it seems that the Tesseract engine was trained on all background types in the IDPL-PFOD dataset, but its accuracy varies for each background. The highest accuracy is achieved on plain white backgrounds, while the lowest accuracy is observed on noisy salt & pepper backgrounds. Furthermore, the Tesseract engine's accuracy for noisy backgrounds, such as Salt & pepper noise and texture, is below 70%. Therefore, training this engine using these two background types (Salt & pepper noise and texture) will provide valuable insights for developers.

5.5 Robustness against different fonts types

The font type of the text is another important aspect that affects the accuracy of OCR. To evaluate the performance of OCR engines on different font types, the IDPL-PFOD dataset was analyzed, and the average accuracy for each font was calculated. The results are presented in Figures 13 and 14.

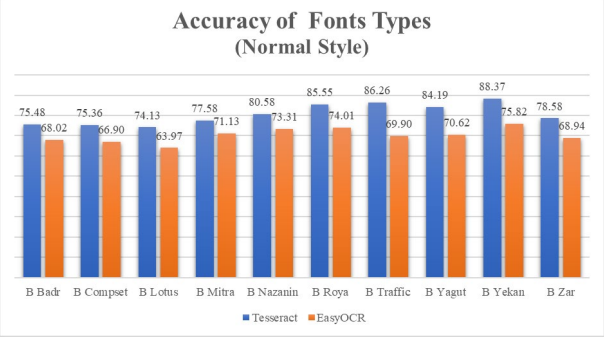


Figure 13: Accuracy of Fonts Types (Normal Style) for Tesseract and EasyOCR engines for all metrics.

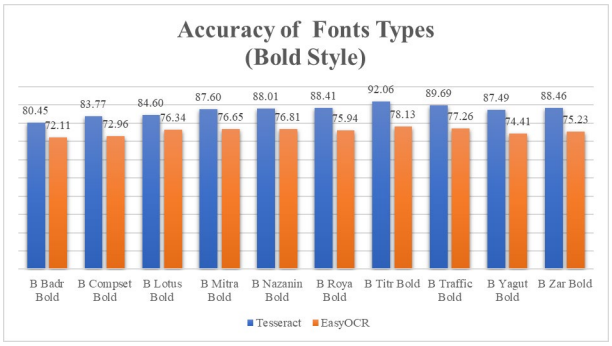


Figure 14: Accuracy of Fonts Types (Bold Style) for Tesseract and EasyOCR engines for all metrics.

Based on the findings, the Tesseract engine outperforms EasyOCR across all font types. Both engines achieve higher accuracy with bold style fonts compared to normal style fonts, indicating that bold style is easier to recognize. Among the fonts in the IDPL-PFOD dataset, the "B Titer Bold" font has the highest accuracy, while the "B Lotus Normal" font has the lowest accuracy.

The Tesseract engine achieves an accuracy above 70% for all fonts, while the EasyOCR engine achieves an accuracy above 60%. This suggests that both engines are trained to recognize all font types in the IDPL-PFOD dataset, but their performance varies for each font.

5.6 Robustness against different font sizes

The accuracy of OCR is affected by various aspects, including the size of the font used in the text. To evaluate how well OCR engines perform at different font sizes, the average accuracy for each font size was calculated and presented the results in Figure 15, comparing different OCR engines. Our findings indicate that Tesseract consistently outperforms EasyOCR across all font sizes. We also observed a clear relationship between accuracy and font size for both engines in the IDPL-PFOD dataset. Generally, larger font sizes result in better text recognition by the OCR engine. However, it is important to note that the maximum allowable font size for official documents is 16px. At this font size, Tesseract achieves an accuracy of 89.92%, while EasyOCR achieves 78%. At 10px, Tesseract hits 73.12% accuracy, while EasyOCR reaches 63.75%.

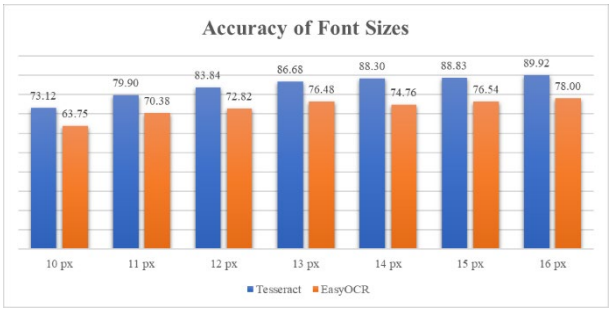


Figure 15: Accuracy of Font Sizes for Tesseract and EasyOCR engines for all metrics.

5.7 Robustness against different distortions

The fourth aspect that affects the accuracy of OCR is the ability of the OCR engine to handle distortions. To evaluate the robustness of OCR engines against different distortions, the average accuracy for each distortion was calculated. The results are shown in Figure 16.

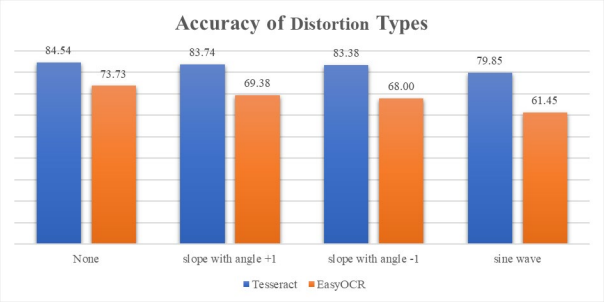


Figure 16: Accuracy of Distortion Types for Tesseract and EasyOCR engines for all metrics.

Based on the findings, the Tesseract engine demonstrates robustness against slope distortion with an angle of ±1. The accuracy of this engine is 84.54% for images without any distortion, and it remains at approximately 83.5% in the presence of slope distortion with an angle of ±1. Therefore, we can conclude that this engine is highly robust against slope distortion with an angle of ±1.

However, Tesseract's robustness against sinewave distortion is not as strong as its robustness against slope distortion with an angle of ±1. The accuracy difference between images without distortion and sinewave distortion in this engine is only 4.7%. Nevertheless, Tesseract still exhibits good robustness against sinewave distortion.

On the other hand, the EasyOCR engine demonstrates good robustness against slope distortion with an angle of ±1. The accuracy difference between images without distortion and slope distortion with an angle of ±1 is approximately 5% in this engine. However, when compared to the Tesseract engine, EasyOCR is slightly less robust against slope distortion with an angle of ±1, indicating that Tesseract performs better in handling this type of distortion.

EasyOCR's robustness against sinewave distortion is acceptable, with an accuracy difference of 12.3% between images without distortion and those with sinewave distortion. However, compared to the Tesseract engine, EasyOCR is 7.6% weaker in terms of robustness against sinewave distortions.

In conclusion, the Tesseract engine generally exhibits greater robustness against distortions compared to the EasyOCR engine.

5.8 Robustness against different blur types

The fifth aspect that significantly affects the accuracy of OCR is the ability of the OCR engine to handle blurriness. To evaluate the robustness of OCR engines against blur, we have divided the average accuracy into two categories: blurred and non-blurred images. Figure 17 illustrates the results for each OCR engine.

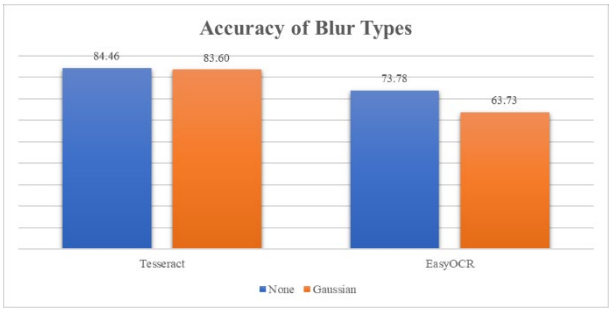


Figure 17: Accuracy of Blur Types for Tesseract and EasyOCR engines for all metrics.

Based on the findings, the Tesseract engine demonstrates high robustness against Gaussian blur with a radius of 0.75. The accuracy difference between images without blur and those with Gaussian blur (radius 0.75) in this engine is less than 1%. On the other hand, the EasyOCR engine exhibits acceptable robustness against blur, with a 10% difference in accuracy between images without blur and those with Gaussian blur (radius 0.75). However, compared to the Tesseract engine, the EasyOCR engine is 9% less robust against Gaussian blur with a radius of 0.75.

In conclusion, the Tesseract engine proves to be more resilient against Gaussian blur with a radius of 0.75 compared to the EasyOCR engine.

6 Discussion and future work

Although the Tesseract and EasyOCR engines are highly regarded OCR systems that support Farsi for many years, there has been no empirical evaluation of these well-knowm OCR engines specifically for Farsi. This is primarily due to the lack of a comprehensive dataset that encompasses all the aspects that contribute to OCR accuracy.

Based on the information provided in section 6, the Tesseract engine outperforms the EasyOCR engine in all aspects except for recognizing images with textured backgrounds. Therefore, one potential area for improving the Tesseract engine is training it to recognize images with textured backgrounds. This engine achieves its highest accuracy of 92.45% on images with a simple white background. However, the accuracy drops to 67.17% for images with textured backgrounds, which is 8.63% lower than EasyOCR. It demonstrates good robustness against Gaussian blur, with only a slight decrease in accuracy from 84.46% to 83.6%. For images with slope distortion (±1 angle) and sinewave distortion, the accuracy decreases to approximately 83.5% and 79.85% respectively. This indicates a high robustness of 99% against slope distortion and 95.31% against sinewave distortion for this engine.

It is important to note that the EasyOCR engine has an accuracy of only 4.79% for images with a noisy salt & pepper background, indicating that it is not specifically trained for this type of image. Furthermore, it achieves modest accuracies of

61.45% and 63.73% for images with sinewave distortion and Gaussian blur with a radius of 0.75, respectively, suggesting room for improvement in these areas. Therefore, optimizing the EasyOCR engine for these specific image types is necessary.

The engine's highest accuracy of 82.89% is observed on images with a simple white background. While it is trained to handle images with textured backgrounds, the accuracy drops to 75.8% for such images. Additionally, when faced with Gaussian blur with a radius of 0.5, the engine's accuracy decreases from 73.78% to 63.73%, indicating a lack of robustness against blur. Similarly, for images with slope distortion (±1 angle) and sine wave distortion, the accuracy decreases to approximately 68.69% and 61.45% respectively. This implies that the engine's accuracy is reduced by 94.96% and 87.72% against these two types of distortion, demonstrating acceptable robustness. Furthermore, the IDPL-PFOD dataset appears to have a limited number of images for texture, noise, and blur distortion. To address this limitation, the IDPL-PFOD dataset needs to be expanded and improved, and the IDPL lab is committed to taking steps in this direction in the future.

Based on the findings presented in Sections 5 and 6, it appears that both engines have been trained using commonly used Farsi fonts. To enhance future work, it is recommended to explore other significant fonts, such as the Nastaliq font. Additionally, since the engines achieved a maximum accuracy of 75.8% for images with textured backgrounds, it is advisable to generate a diverse dataset that includes various texture backgrounds. This dataset can then be used to retrain both engines and address the issue.

The results also indicate the importance of training the engines to improve their performance in recognizing blurred images, sinewave distortion, noisy salt & pepper backgrounds, and texture backgrounds. However, due to the limited number of images in the IDPL-PFOD dataset that pertain to these specific cases, it is suggested that future work focuses on creating an extensive dataset to tackle these challenges. Subsequently, the engines can be trained to improve their performance in recognizing blurred images, sinewave distortion, and texture backgrounds. Furthermore, developing multilingual datasets to assess the engines' performance on documents in multiple languages would be beneficial for potential training purposes.

7 Conclusions

This paper explores the challenges associated with OCR for the Farsi language. Despite these challenges, there are OCR engines available that support Farsi and achieve high accuracy when the image quality and resolution are optimal. However, these engines struggle when presented with real-life images. Therefore, it is necessary to evaluate the efficiency of these engines in real-world scenarios.

Our evaluation focused on two OCR engines, namely Tesseract and EasyOCR, using a Farsi dataset consisting of 18 different image types, along with accurate Ground Truth data. The dataset images were categorized into four groups, encompassing diverse backgrounds, font styles, distortions, and blurriness. This categorization allowed us to assess various aspects of the engines' capabilities. We employed several commonly used evaluation metrics, with a particular emphasis on Levenshtein accuracy, which provides a balanced evaluation at the character and word levels. All metrics were based on measuring the distance between the Ground Truth and the OCR output.

The evaluation results revealed that Tesseract exhibits less resilience than EasyOCR when it comes to recognizing images with textured backgrounds. However, Tesseract outperforms EasyOCR in various other scenarios. Therefore, to enhance Tesseract's performance, it is necessary to train it using textured background images. On the other hand, EasyOCR faces difficulties when recognizing images with salt-and-pepper noise backgrounds. Consequently, comprehensive training from scratch is essential to improve its accuracy in such cases. Additionally, EasyOCR's accuracy declines when dealing with images that exhibit sinewave distortion and Gaussian blur (radius: 0.75). Therefore, additional training specifically targeting these types of images is necessary.

However, the limited number of texture, noise, and blur-distorted images in the IDPL-PFOD dataset is insufficient for training OCR engines to enhance their robustness to background texture, noise, and blur distortions. To overcome this limitation, there is a requirement to enhance and expand the IDPL-PFOD dataset.

Disclosure of Potential Conflicts of Interest

The Authors declare that there is no conflict of interest

Reference

[1] E. A. Santos, "OCR evaluation tools for the 21st century," in Proceedings of the Workshop on Computational Methods for Endangered Languages, 2019, vol. 1.

[2] K. Nikoghosyan, "OCR ENGINE COMPARISON-TESSERACT VS. EASYOCR VS. KERAS-OCR," in СОВРЕМЕННАЯ НАУКА И ТЕХНИЧЕСКИЙ ПРОГРЕСС. НОВАЯ ПРОМЫШЛЕННАЯ РЕВОЛЮЦИЯ В ЗЕРКАЛЕ СОВРЕМЕННОЙ НАУКИ, 2022, pp. 61-67.

[3] R. R. TN and S. Praveen, "Application of OCR to Design an Intelligent PDF Parser for Technical Documentation."

[4] M. Ganis, C. L. Wilson, and J. L. Blue, "Neural network-based systems for handprint OCR applications," IEEE Transactions on Image Processing, vol. 7, no. 8, pp. 1097-1112, 1998.

[5] R. Gossweiler, M. Kamvar, and S. Baluja, "What's up CAPTCHA? A CAPTCHA based on image orientation," in Proceedings of the 18th international conference on World wide web, 2009, pp. 841-850.

[6] J. Barwick, "Building an institutional repository at Loughborough University: some experiences," Program, 2007.

[7] S. Naz, A. I. Umar, S. B. Ahmed, S. H. Shirazi, M. I. Razzak, and I. Siddiqi, "An Ocr system for printed Nasta'liq script: A

segmentation based approach," in 17th IEEE International Multi Topic Conference 2014, 2014: IEEE, pp. 255-259.

[8] S. S. Tsai, H. Chen, D. Chen, G. Schroth, R. Grzeszczuk, and B. Girod, "Mobile visual search on printed documents using text and low bit-rate features," in 2011 18th IEEE International Conference on Image Processing, 2011: IEEE, pp. 2601-2604.

[9] R. Chen, B. C. Desai, and C. Zhou, "CINDI robot: an intelligent Web crawler based on multi-level inspection," in 11th International Database Engineering and Applications Symposium (IDEAS 2007), 2007: IEEE, pp. 93-101.

[10] Y. K. Ham, M. S. Kang, H. K. Chung, R.-H. Park, and G. T. Park, "Recognition of raised characters for automatic classification of rubber tires," Optical Engineering, vol. 34, no. 1, pp. 102-109, 1995.

[11] G. N. DeSouza and A. C. Kak, "Vision for mobile robot navigation: A survey," IEEE transactions on pattern analysis and machine intelligence, vol. 24, no. 2, pp. 237-267, 2002.

[12] S. Kashef, H. Nezamabadi-pour, and E. Shabaninia, "A review on deep learning approaches for optical character recognition with emphasis on Persian, Arabic and Urdu scripts," Journal of Machine Vision and Image Processing, vol. 8, no. 4, pp. 51-85, 2021.

[13] P. J. Haghighi, N. Nobile, C. L. He, and C. Y. Suen, "A new large-scale multi-purpose handwritten Farsi database," in International Conference Image Analysis and Recognition, 2009: Springer, pp. 278-286.

[14] R. Azmi and E. Kabir, "A New Segmentation Technique for Omnifont Farsi Text," Computational Methods in Engineering, vol. 18, no. 2, pp. 1-10, 1999.

[15] R. Safabaksh, A. R. Ghanbarian, and G. Ghiasi, "HaFT: A handwritten Farsi text database," in 2013 8th Iranian Conference on Machine Vision and Image Processing (MVIP), 10-12 Sept. 2013 2013, pp. 89-94, doi: 10.1109/IranianMVIP.2013.6779956.

[16] S. Torabzadeh and R. Safabaksh, "AUT-PFT: A real world printed Farsi text image dataset," in 2015 The International Symposium on Artificial Intelligence and Signal Processing (AISP), 2015: IEEE, pp. 267-272.

[17] F. sadat Hosseini, E. Shabaninia, S. Kashef, and H. Nezamabadi-pour, "IDPL-PFOD: An Image Dataset of Printed Farsi Text for OCR Research," NSURL 2021, p. 1, 2021.

[18] F. K. Jaiem, S. Kanoun, M. Khemakhem, H. E. Abed, and J. Kardoun, "Database for Arabic printed text recognition research," in International Conference on Image Analysis and Processing, 2013: Springer, pp. 251-259.

[19] F. Solimanpour, J. Sadri, and C. Y. Suen, "Standard databases for recognition of handwritten digits, numerical strings, legal amounts, letters and dates in Farsi language," in Tenth International workshop on Frontiers in handwriting recognition, 2006: Suvisoft.

[20] Z. Khosrobeygi, H. Veisi, H. Ahmadi, and H. Shabanian, "A rule-based post-processing approach to improve Persian OCR performance," Scientia Iranica, vol. 27, no. 6, pp. 3019-3033, 2020.

[21] Y. A. Nanehkaran, D. Zhang, S. Salimi, J. Chen, Y. Tian, and N. Al-Nabhan, "Analysis and comparison of machine learning classifiers and deep neural networks techniques for recognition of Farsi handwritten digits," The Journal of Supercomputing, vol. 77, no. 4, pp. 3193-3222, 2021.

[22] R. Smith, "An overview of the Tesseract OCR engine," in Ninth international conference on document analysis and recognition (ICDAR 2007), 2007, vol. 2: IEEE, pp. 629-633.

[23] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," IEEE transactions on pattern analysis and machine intelligence, vol. 39, no. 11, pp. 2298-2304, 2016.

[24] H. Zhang, E. Whittaker, and I. Kitagishi, "Extending TrOCR for Text Localization-Free OCR of Full-Page Scanned Receipt Images," arXiv preprint arXiv:2212.05525, 2022.

[25] E. Shabaninia, H. Nezamabadi-pour, and F. Shafizadegan, "Transformers in Action Recognition: A Review on Temporal Modeling," arXiv preprint arXiv:2302.01921, 2022.

[26] B. Alizadehashraf and S. Roohi, "Persian handwritten character recognition using convolutional neural network," in 2017 10th Iranian Conference on Machine Vision and Image Processing (MVIP), 2017: IEEE, pp. 247-251.

[27] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.

[28] E. Farahbakhsh, E. Kozegar, and M. Soryani, "Improving Persian digit recognition by combining data augmentation and AlexNet," in 2017 10th Iranian Conference on Machine Vision and Image Processing (MVIP), 2017: IEEE, pp. 265-270.

[29] S. Khosravi and A. Chalechale, "Chimp Optimization Algorithm to Optimize a Convolutional Neural Network for Recognizing Persian/Arabic Handwritten Words," Mathematical Problems in Engineering, vol. 2022, 2022.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770-778.

[31] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700-4708.

[32] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures," Neural computation, vol. 31, no. 7, pp. 1235-1270, 2019.

[33] J. Wang and X. Hu, "Gated recurrent convolution neural network for ocr," Advances in Neural Information Processing Systems, vol. 30, 2017.

[34] A. Ranjkesh Rashtehroudi, A. Akoushideh, and A. Shahbahrami, "Persian Scene Text Recognition with Convolutional Recurrent Neural Network," Alireza and Shahbahrami, Asadollah, Persian Scene Text Recognition with Convolutional Recurrent Neural Network, 2022.

[35] Z. Wojna, A. N. Gorban, D.-S. Lee, K. Murphy, Q. Yu, Y. Li, and J. Ibarz, "Attention-based extraction of structured information from street view imagery," in 2017 14th IAPR international conference on document analysis and recognition (ICDAR), 2017, vol. 1: IEEE, pp. 844-850.

[36] R. Atienza, "Vision transformer for fast and efficient scene text recognition," in International Conference on Document Analysis and Recognition, 2021: Springer, pp. 319-334.

[37] K. Barrere, Y. Soullard, A. Lemaitre, and B. Coüasnon, "Transformers for historical handwritten text recognition," in Doctoral Consortium-ICDAR 2021, 2021.

[38] P. Jain, K. Taneja, and H. Taneja, "Which OCR toolset is good and why: A comparative study," Kuwait Journal of Science, vol. 48, no. 2, 2021.

[39] T. M. Breuel, "The OCRopus open source OCR system," in Document recognition and retrieval XV, 2008, vol. 6815: SPIE, pp. 120-134.

[40] D. E. Wood, J. Lu, and B. Langmead, "Improved metagenomic analysis with Kraken 2," Genome biology, vol. 20, pp. 1-13, 2019.
[41] J. Schulenburg, "GOCR: Open-source character recognition, version 0.39," http://jocr. sourceforge. net/index. html, 2004.
[42] GNU, "Ocrad - The GNU OCR," 2022. T. Berg-Kirkpatrick, G. Durrett, and D. Klein, "Unsupervised transcription of historical documents," in Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2013, pp. 207-217.
[43] T. Berg-Kirkpatrick, G. Durrett, and D. Klein, "Unsupervised transcription of historical documents," in Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2013, pp. 207-217.
[44] "SwiftOCR," 2020. [Online]. Available: https://github.com/NMAC427/SwiftOCR.
[45] "Attention-OCR." [Online]. Available: https://github.com/emedvedev/attention-ocr.
[46] D. Rybach, C. Gollan, G. Heigold, B. Hoffmeister, J. Lööf, R. Schlüter, and H. Ney, "The RWTH Aachen University open source speech recognition system," in Tenth Annual Conference of the International Speech Communication Association, 2009.
[47] "Simple-OCR-OpenCV." [Online]. Available: https://github.com/goncalopp/simple-ocr-opencv.
[48] C. Reul et al., "OCR4all—An open-source tool providing a (semi-) automatic OCR workflow for historical printings," Applied Sciences, vol. 9, no. 22, p. 4853, 2019.
[49] H. Feng, Y. Wang, W. Zhou, J. Deng, and H. Li, "Doctr: Document image transformer for geometric unwarping and illumination correction," arXiv preprint arXiv:2110.12942, 2021.
[50] "PaddleOCR." [Online]. Available: https://github.com/PaddlePaddle/PaddleOCR.
[51] M. Troller, "Practical OCR system based on state of art neural networks," Czech Technical University in Prague Dejvice, Czech Republic, 2017.
[52] "ClaraOCR." https://directory.fsf.org/wiki/Clara_OCR (accessed.
[53] "CuneiForm." "EyeOCR." [Online]. Available: https://sourceforge.net/projects/eyeocr/.
[54] "EyeOCR." [Online]. Available: https://sourceforge.net/projects/eyeocr/.
[55] "kognition." [Online]. Available: https://sourceforge.net/projects/kognition/.
[56] "OCRchie." [Online]. Available: https://people.eecs.berkeley.edu/~fateman/kathey/ocrchie.html.
[57] "orce." [Online]. Available: http://lem.eui.upm.es/ocre.html.
[58] "Xplab." [Online]. Available: http://www.pattern-lab.de/.
[59] "hebOCR." [Online]. Available: https://github.com/yaacov/hebocr.
[60] A. A. Asadi. Shotor Dataset. [Online]. Available: https://www.kaggle.com/amir137825/persianocrdataset/version/2
[61] Partdpai.ir. Persian Subwords. [Online]. Available: https://github.com/partdpai/persian_subwords
[62] F. Arefi. Arshasb Persian OCR Dataset. [Online]. Available: https://github.com/persiandataset/Arshasb
[63] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, "Character region awareness for text detection," in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 9365-9374.
[64] M. Li et al., "Trocr: Transformer-based optical character recognition with pre-trained models," in Proceedings of the AAAI Conference on Artificial Intelligence, 2023, vol. 37, no. 11, pp. 13094-13102.
[65] S. Saber, A. Ahmed, and M. Hadhoud, "Robust metrics for evaluating Arabic OCR systems," in International Image Processing, Applications and Systems Conference, 2014: IEEE, pp. 1-6.
[66] F. B. Safir, A. Q. Ohi, M. F. Mridha, M. M. Monowar, and M. A. Hamid, "End-to-end optical character recognition for bengali handwritten words," in 2021 National Computing Colleges Conference (NCCC), 2021: IEEE, pp. 1-7.