Journal of Computer Science

Original Research Paper

A Printed Arabic Optical Character Recognition System using Deep Learning

Salah Alghyaline

Department of Computer Science, the World Islamic Sciences and Education University, Amman, Jordan

Article history
Received:14-08-2022
Revised: 28-09-2022
Accepted: 08-10-2022

Email: salah.alghyaline@wise.edu.com

Abstract: Recognizing Arabic script is challenging for many reasons: The Arabic language is cursive and morphologically rich. There is a high similarity between Arabic letters. Moreover, Arabic has many diacritics and dots, and they change the letter's phonetic transcription. This study proposes a Printed Arabic Optical Character Recognition approach (PAOCR) based on the state-of-the-art You Only Look Once (YOLO) object detector. Four techniques were proposed and implemented to design an end-to-end Arabic OCR system. First, the YOLO4 object detector is customized and trained on deep Convolutional Neural Networks (CNNs) to recognize Arabic characters. Second, the overlapped bounding boxes are processed to keep the most accurate box for each character. Third, the Hunspell library checks the word spelling and corrects the wrong ones. Fourth, edit distance is used to compare OCR misspelled words with Hunspell's suggestions and choose the closest correct word. The experimental results in the Arabic Printed Text Image (APTI) dataset showed that the Word Recognition Rate (WRR) of customized YOLO4 Arabic OCR is 66.7%, whereas applying the three proposed techniques with YOLO4 achieved 82.4%. A comparison with two existing OCR systems Tesseract and ABBYY showed that the proposed approach achieved a 95.7% Character Recognition Rate (CRR), whereas Tesseract and ABBYY achieved 92.3 and 84.8%, respectively.

Keywords: Optical Character Recognition, Arabic OCR, Convolutional Neural Network, YOLO4

Introduction

Optical Character Recognition (OCR) applies image processing technology to convert printed and handwritten text images into editable text. The aim is to use the editable text for further processing (Patel *et al.*, 2012). OCR applications includes automatic cheque recognition (Mehta *et al.*, 2010), handwriting form recognition (Fanany, 2017), Automatic License Plate Recognition (ALPR) (Alghyaline, 2020), text reading for blind people (Liambas and Saratzidis, 2016) and digitization of historical documents (Bukhari *et al.*, 2017). Humans can easily recognize printed and written text with high accuracy, and they do that after years of training during school time. However, they could face some problems recognizing the written text because the handwriting is different from one person to another. Therefore, many efforts were made to develop accurate and fast OCR systems. Each language has its specifications and many OCR approaches were proposed for different languages: English OCR (Mehta *et al.*, 2016; Chaudhuri *et al.*, 2017a; Afroge *et al.*, 2016; Zanwar *et al.*, 2020). Japanese OCR (Srihari *et al.*, 1997; Nguyen and Nakagawa, 2016), Chinese OCR (Yin *et al.*, 2019; Li *et al.*, 2016), Indian OCR (Mathew *et al.*, 2016; Chaudhuri *et al.*, 2017b) Arabic OCR systems (Al-Amin and Alsaade, 2017; Nashwan *et al.*, 2017). Languages are classified as cursive and non-cursive and it is more challenging to recognize the cursive alphabet. Also, there are handwriting and printed texts OCR, and it is even more challenging to recognize handwriting texts. Arabic is one of the most difficult languages (Boudelaa *et al.*, 2010; Farghaly and Shaalan, 2009; Lu *et al.*, 1999), it is right-to-left script and the letters are joined together (cursive). Most of the Arabic letters in the word are connected and there is a high similarity between their shapes. Some comparative studies (Alkhateeb *et al.*, 2017) show that the word recognition rate of existing Arabic OCR systems is under 75%. Deep learning features showed superior accuracies in the object recognition field (Girshick, 2015; Redmon *et al.*, 2016; Bochkovskiy *et al.*, 2020). The variation in the character's style, size and font type and the cursive nature of the character makes it more challenging to develop an accurate Arabic OCR system. YOLO4 represents the state-of-the-art object detector (Bochkovskiy *et al.*, 2020). YOLO4 runs in real-time with around 65 FPS speed. Many real-time applications adopt YOLO due to its high accuracy and speed (Alghyaline, 2019; 2020; Jiang *et al.*, 2022).

This study proposes a printed Arabic OCR approach. YOLO4 is customized and trained to recognize Arabic characters. However, YOLO does not yield a reasonable recognition rate because one character may have multiple recognition results. This study improves the YOLO4 OCR recognition accuracy using three techniques: Removing the redundant recognition results by processing the overlapped bounding boxes and keeping the boxes with the highest accuracy. In addition, a spelling checker with edit distance is used for correcting the wrong words. The experimental results on the well-known database called Arabic Printed Text Image (APTI) (Slimane *et al*., 2009) show that using these techniques significantly improves the OCR for the Arabic language. The proposed approach achieved 82.4% Word Recognition Rate (WRR), whereas the YOLO4 approach achieved 66.7% WRR. Furthermore, a comparison with existing commercial and open-source OCR systems shows that the proposed approach achieved higher WRR and Character Recognition Rate (CRR).

Related Works

Most of the developed Optical Character Recognition (OCR) systems are designed for non-cursive scripts like Latin and English text. (Tian *et al*., 2016) developed OCR that recognizes English words from natural images. The approach is implemented based on the Visual Geometry Group (VGG16) model and reached superior results on ICDAR2013 and ICDAR2015 datasets. (Joseph, 2020) proposed OCR approach to recognize Thai characters. Preprocessing operations are used to improve picture quality such as converting the image to grayscale color, binarization, and median filter are used to remove image noise. Canny edge detection is used to segment the characters. The segmented image is represented by 11 × 11 grids to encode local binary features and classified using SVM. (Yang *et al*., 2019) proposed OCR method to recognize Japanese characters. AlexNet and GoogLeNet architectures are trained on documents from Taiwan records to recognize Japanese text. (Zou *et al*., 2019) proposed OCR approach to recognize Chinese characters. The approach is based on CNNs. The proposed CNNs include six conv and six pooling layers. It includes two fully connected layers and dropout layer with a 25%. (Lee *et al*., 2018) trained a model to recognize Korean characters based on AlexNet and GoogLeNet CNNs architectures.

Stolyarenko *et al.* (2011) used scale Invariant Feature Transform (SIFT) descriptor with sliding windows to solve the Arabic OCR problem. (Addakiri and Bahaj, 2012) proposed an online OCR approach for handwritten Arabic characters. The approach converts the input images into binary images; then, the converted images are used to train a neural network to recognize Arabic characters. The networks include input layer, two hidden layers, and output layer. (Yousefi *et al*., 2015) proposed system to recognize handwritten script. The approach starts by normalizing the baseline and the position of letters. Long Short-Term Memory (LSTM) network is used for letter classification. (Radwan *et al*., 2016) proposed OCR for printed text, a neural network with three sliding windows, is used to segment the Arabic characters. Another neural network with two convolutional layers and two max-pooling layers is used for character recognition. (Ahmed *et al*., 2017) proposed CNNs architecture for Arabic OCR. The method converts the image color into grayscale and then changes its dimension to 50 × 50 pixels. The CNNs architecture comprises two convolutional layers, each convolutional layer followed by a max-pooling layer and there is one fully connected layer. (Zayene *et al*., 2017) a Recurrent Neural Networks (RNN) is designed to address character recognition problem. Images from TV videos were used to train and evaluate the approach and the approach does not require any image processing operations. (Radwan *et al*., 2018) proposed offline OCR for printed Arabic text. The method includes three stages; each stage has a shallow neural network to solve a specific sub-task of OCR. The first stage predicts the font size and normalizes it to 18pt. Then the words are segmented into characters. Finally, the segmented characters are passed through a neural network of four layers (two convolutions and two max-pooling) to recognize them. (Doush *et al*., 2018) proposed a post-processing system to improve Arabic OCR accuracy. The post-processing operations include language and error model techniques and google suggestions. (Darwish and Elzoghaly, 2020) proposed approach starts by prepossessing operations; binarization, median filter, morphologic operations, rotation, and resizing. Gray-Level Co-occurrence Matrix (GLCM) extracts the feature, and genetic algorithm is implemented to select the most discriminative features.

Table 1: Summary of some Arabic OCR methods

Reference	Year	Model	Dataset	Type	Accuracy
Yousefi *et al.* (2015)	2015	Long Short-Term Memory (LSTM)	IFN/ENIT	words	87.40%
Elleuch *et al.* (2016)	2016	CNN and SVM	HACDB	Characters	94.47%
Radwan *et al.* (2016)	2016	Multichannel NN	APTI	words	94.80%.
Fasha *et al.* (2020)	2020	deep CNN	APTI	words	76.30%
Butt *et al.* (2021)	2021	CNN-RNN model	Alif dataset	words	85.98%
Rahal *et al.* (2021)	2021	Hidden Markov Models & Sparse Auto-Encoder	IFN/ENIT	words	95.10%
Balaha *et al.* (2021)	2021	deep CNN	HMBD	Characters	92.88%
Jbrail and Tenekeci (2022)	2022	deep CNN	Hijja data set	Characters	99.30%

Finally, Fuzzy K-NN classifier is used to predict class label. (Butt *et al.*, 2021) recognized Arabic script from a video stream. Recurrent Neural Networks (RNNs) and CNNs were to implement the approach. The CNNs architecture was adopted from the VGG model (Saidane and Garcia, 2007).

Elleuch *et al.* (2016) developed an Arabic OCR approach based on CNNs. The CNNs architecture involves two Conv layers, and a sub-sampling layer follows each Conv layer. The last layer of the CNN is fully connected and includes a dropout of fifty percent of the nodes. The extracted features are classified using SVM. (Mustafa and Elbashir, 2020) proposed CNNs architecture for Arabic OCR. The architecture includes three Conv layers and two max-pooling layers. The last layer is a fully-connected layer with a dropout of twenty percent of the nodes. (Alkhateeb *et al.*, 2017) evaluated six well-known Arabic OCR systems using the same criteria and Arabic datasets. The evaluated OCR systems are Tesseract, Abbyy, Sakhr, NovoVerus, Readiris, and Leadtools. The study concluded that ABBYY OCR provides the best recognition accuracy rate among the evaluated systems. (Albashir *et al.*, 2020) proposed a system to translate Arabic to English. Thresholding, noise removal and gray-scale image operation are used in image preprocessing. After that, features are collected using Histograms of Oriented Gradients (HOG) descriptor and second features (corners, mean, connected components, gravity). Finally, classification is implemented using SVM. Table 1 summarizes the characteristics of the proposed Arabic OCR approaches during the last seven years.

Background

Optical Character Recognition

Any OCR system includes three steps, image preprocessing, character segmentation, and character recognition (Abdo *et al.*, 2022). Reprocessing operations could enhance image quality, such as resizing, binarization, smoothing, rotation, and noise removal. However, most of the existing OCR approaches perform resizing and binarization only. Character segmentation includes dividing the original image into sub-images. Each sub-image contains a single character. Character recognition takes the segmented images as input and outputs a label for each image. Finally, it includes extracting the features of the image and classifying them into language characters. There are mainly two kinds of features in the image: Handcrafted and deep-learned features. Deep learning features have shown impressive results during the last few years (Aburass *et al.*, 2022). Figure 1 shows the main steps of the OCR system. Preprocessing includes many operations, such as image resizing, cropping, color conversion, pixel normalization, morphological operations, and skewness correction. Segmentation is used to segment the input word into characters. Some OCRs are segmentation-free and the sliding-window technique detects and recognizes the characters. The recognition stage includes predicting the class label of the segmented character.

Table 2: Arabic Alphabet with similar shapes

Group 1	"ت", "ث", "ب", and "ن"
Group 2	"ظ" and "ط"
Group 3	"ف" and "ق"
Group 4	"ج", "ح", and "خ"
Group 5	"س" and "ش"
Group 6	"ز", "ذ" and "ر"
Group 7	"ه" and "ة"
Group 8	"لا", "لإ" and "لأ"

Table 3: Some variations of the Arabic root Alema (علم)

Transliteration	Arabic word	English meaning
Alema	علم	Knew
Y'alem	يعلم	Know
Ma'lwom	معلوم	known
Taleem	تعليم	Education
Aleem	عليم	Expert
Alolom	العلوم	Sciences
Mutalem	متعلم	Learner
Alem	عالِم	Scientist
Olwom	علماء	Scientists
Ta'leme	تعليمي	Educational
Ma'lomat	معلومات	Information
Aloloom	العلم	The science

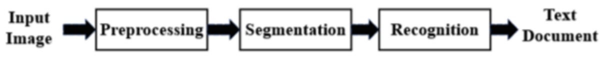
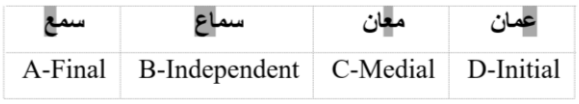
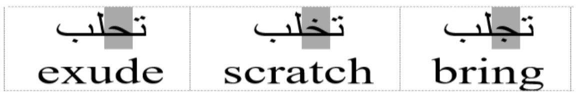


Fig. 1: Main phases of the OCR system



سمع	سماع	معان	عمان
A-Final	B-Independent	C-Medial	D-Initial

Fig. 2: Ayn letter (ع) at different positions in the word



تحلب	تخلب	تجلب
exude	scratch	bring

Fig. 3: Three Arabic words with dots at different locations

Arabic is spoken by more than 400 million people in the middle east and north Africa (The World Bank, 2022). The Holy Quran is written in Arabic and read by Muslims worldwide. Many languages use the Arabic alphabet, such as Urdu, Persian, Kurdish, Punjabi, Baluchi, Pashto Brahui, and Kashmiri. Arabic consists of 28 letters and Hamza (ء) characters. Each letter in Arabic could have up to four shapes according to the letter's position. The four letters' locations are initial, medial, independent, and final, Fig. 2 shows the letter Ayn (ع) (the shaded part), the letter shape is changed according to its location in the word. Figure 3 shows how the dots change the word meaning. Table 2 shows a set of Arabic letters that have similar shapes. The structure of the Arabic language is complex and rich (Boudelaa *et al.*, 2010; Farghaly and Shaalan, 2009). Some variations of Alema (علم) root are shown in Table 3. Arabic also uses diacritics above and under letters, there are eight diacritic markings in Arabic, and they change the word meaning accordingly. Most native speakers of Arabic do not need to write all these diacritics in writing and the reader can pronounce them by understanding the context of the sentence.

Materials and Methods

The main phases of the proposed system are shown in Figure 4 and 5. The YOLO4 object detector is used to train a customized CNN model to recognize Arabic characters and then this model is used with the testing dataset to detect new unknown Arabic characters. YOLO performs pre-processing operations to enhance recognition results, such as Image resizing, cropping, data augmentation, and color conversion. The proposed approach is segmentation-free. YOLO divides the input image into $N \times N$ grids and uses CNNs to predict the class label and the confidence for each grid. First, one character had many detections during the experiments, and some of them were incorrect. Therefore, the overlapping percentage between the bounding boxes and the confidence of each detection is used to filter out incorrect detections.

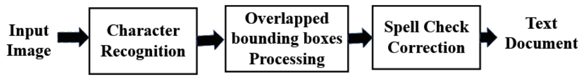


Fig. 4: Main steps of the proposed OCR system

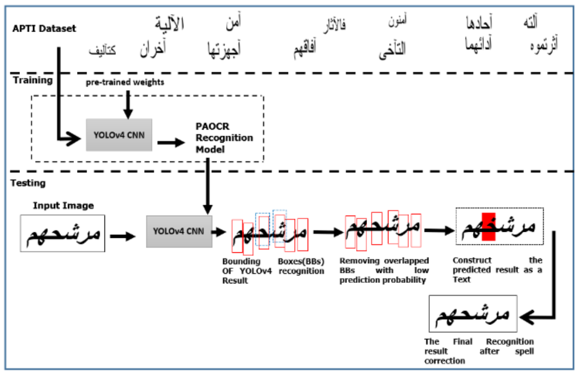

APTI Dataset
Training
pre-trained weights
YOLOv4 CNN
PAOCR Recognition Model
Testing
Input Image
YOLOv4 CNN
Bounding Boxes(BBs) OF YOLOv4 recognition Result
Removing overlapped BBs with low prediction probability
Construct the predicted result as a Text
The Final Recognition result after spell correction

Fig. 5: The Framework of the proposed PAOCR approach

Then the detected characters in different locations are concatenated to construct a word. Finally, the spell of the words is checked. If the word does not exist in the dictionary, it will be replaced by the closest suggestion (using edit distance). The main steps of the proposed Printed Arabic Optical Character Recognition (PAOCP) approach are summarized in Algorithm 1.

Algorithm 1: PAOCP

Input : *D*, a Dataset of printed Arabic Text Images; *overlap_threshold*, Overlap threshold

Output: The corresponding plain text for each image *I* in *D*

```
1. Begin
2. Use D to train a customize YOLO_4 CNN architecture
   to predict printed Arabic text, denote the model as M
   and the customized architecture as CY_CNN
3. For each image I in D do
4.    Use CY_CNN and M to predict the set of characters in
      I, denote the list of bounding boxes as intial_bb and
      the list of confidence as Conf
5.    For each a in initial_bb do
6.       //Use the x, and y coordinates of a to check the
         overlap
         //between a and all items in initial_bb
7.       For each b in initial_bb and a ≠ b do
8.          If overlap_percentage(a,b) > overlap_threshold
9.             If Conf (a) > Conf(b)
10.               Remove b from initial_bb
11.            Else
12.               Remove a from initial_bb
```

```
13.         End If
14.       End If
15.    End for
16. End for
17. Let final_bb be the set of bounding boxes without overlapping
18. final_bb= initial_bb
19. Arrange the characters in final_bb according to their x, and y coordinates to construct the predicted word and denote it as pw.//predicted word
20. Let dic denotes the set of Arabic words in the Hunpsell Dictionary
21. If pw exists in dic
22. Print pw to a text file//the word spelling is correct
23. Else
24. Use Editdistance to find the most similar word to pw in dic and denote it as cw
25. Print cw//correct word
26. End
```

Table 4: Samples of manually labeled Arabic characters using the LabelImg tool

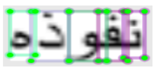
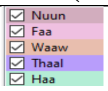
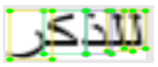
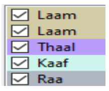
Manually created bounding boxes	Labels (from right to left)	Pronunciation
	 ☑ Nuun ☑ Faa ☑ Waaw ☑ Thaal ☑ Haa	No-fu-tha-hu
	 ☑ Laam ☑ Laam ☑ Thaal ☑ Kaaf ☑ Raa	Lel-thek-ra

Table 5: List of Arabic characters

Arabic letter	Alone	Beginning	Middle	End
Alif	ا			ـا
AlifBroken	ى			ـى
Ayn	ع	عـ	ـعـ	ـع
Baa	ب	بـ	ـبـ	ـب
Daad	ض	ضـ	ـضـ	ـض
Daal	د			ـد
Faa	ف	فـ	ـفـ	ـف
Gaaf	ق	قـ	ـقـ	ـق
Ghayn	غ	غـ	ـغـ	ـغ
Haa	ه	هـ	ـهـ	ـه
Haaa	ح	حـ	ـحـ	ـح
Hamza	ء			
HamzaAboveAlif	أ			ـأ
HamzaAboveAlifBroken	ئ	ئـ	ـئـ	ـئ
HamzaAboveWaaw	ؤ			ـؤ
HamzaUnderAlif	إ			ـإ
Jiim	ج	جـ	ـجـ	ـج
Kaaf	ك	كـ	ـكـ	ـك
Laam	ل	لـ	ـلـ	ـل
LaamAlif	لا			ـلا
LaamHamzaAboveAlif	لأ			ـلأ
LaamHamzaUnderAlif	لإ			ـلإ
LaamTildeAboveAlif	لآ			ـلآ
Miim	م	مـ	ـمـ	ـم
Nuun	ن	نـ	ـنـ	ـن
NuunChadda	نّ	نّـ	ـنّـ	ـنّ
Raa	ر			ـر
Saad	ص	صـ	ـصـ	ـص
Shiin	ش	شـ	ـشـ	ـش
Siin	س	سـ	ـسـ	ـس
Taa	ظ	ظـ	ـظـ	ـظ
Taaa	ت	تـ	ـتـ	ـت
TaaClosed	ة			ـة
Thaa	ث	ثـ	ـثـ	ـث
Thaaa	ط	طـ	ـطـ	ـط
Thaal	ذ			ـذ
TildeAboveAlif	آ			ـآ

Table 5: Continue

Waaw	و			ـو
Xaa	خ	خـ	ـخـ	ـخ
Yaa	ي	يـ	ـيـ	ـي
YaaChadda	يّ	يّـ	ـيّـ	ـيّ
Zaay	ز			ـز

Table 6: Samples of YOLO4 character recognition results

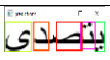
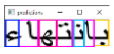
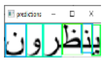
#	YOLO Predicted Bounding boxes	YOLO Predicted label and confidence	Pronunciation
1		Yaa: 99% Taaa: 99% Saad: 43% Daal: 99% AlifBroken: 100%	Ya-ta-sa-da
2		Baa: 96% Alif: 99% Nuun: 98% Taaa: 99% Haa: 99%t Alif: 98% Hamza: 99%	Ben-te-ha'a
3		Yaa: 99% Nuun: 100% Taa: 99% Raa: 100% Waaw: 99% Nuun: 100%	Yan-ta-the-roon

Table 7: Samples of checking the overlapping between the characters

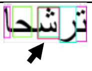
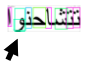
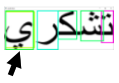
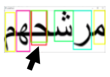
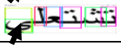
#	YOLO Predicted Bounding boxes	YOLO Predicted labels	Pronunciation	Before overlap checking	After overlap checking
1		Taaa: 99% Raa: 86% SHIIN: 97% Siin: 38% Haaa: 99% Alif: 96%	Ta-ra-sha-haa	ترشحا	ترشسحا
2		Taaa: 99% Taaa: 98% Shiin: 98% Alif: 99% Haaa: 99% Nuun: 100% Waaw: 50% Waaw: 99% Alif: 75%	Ta-ta-sha-ha-no	تتشاحنوا	تتشاحنوا
3		Taaa: 98% Kaaf: 99% Raa: 98% AlifBroken: 31% Yaa: 98%	Tash-ko-re	تشكري	تشكري
4		Miim: 96% Raa: 98% Shiin: 99% Xaa: 65% Haaa: 97% Haa: 99% Miim: 95%	Mo-ra-sha-ha-hum	مرشخحهم	مرشحهم

Table 8: Samples of checking the spelling

#	YOLO Predicted Bounding boxes	Pronunciation	Before Spell Checking	After spell checking
1		Fa-he-sheen	قاحشين	فاحشين
2		Ka-sha-fat	كشقت	كشفت
3		Al-ma-a'a-sh	المعاشن	المعاش
4		Tash-ta'a-lee	تشتعلى	تشتعلي

Arabic Character Recognition

Characters detection and recognition are essential tasks during the OCR process. Arabic characters are objects like other objects. Object recognition accuracy depends on the amount and the type of extracted features from that object. The extracted features from characters are little when compared with other objects like animals and vehicles and this is because characters are small and do not have color features. There is a high similarity between some of the Arabic characters. Consequently, Arabic character recognition is complicated compared with recognizing other objects.

Deep learning-based object detectors such as Faster Region-based Convolutional Neural Networks (Faster R-CNN) (Ren *et al*., 2015), Single Shot MultiBox Detector (SSD) (Liu *et al*., 2016) and YOLO showed remarkable accuracy compared with handcrafted and statistical methods (Sharma *et al*., 2022). However, YOLO 4 achieved a state-of-the-art object detector (Bochkovskiy *et al*., 2020).

It is mentioned that YOLO is the most accurate and fastest object detector. Therefore, a customized YOLO4 CNN is used in the Arabic character recognition phase. The APTI dataset (Slimane *et al*., 2009) (set 1 to set 4) were used to evaluate the proposed technique. 3574 pictures were chosen randomly to train a YOLO4 model. The images are annotated using the LabelImg tool (TzuTa, 2022), as shown in Table 4. The dimensions of Arabic characters are relatively small compared with other objects (dogs, cats, vehicles). Therefore, K-means is running on the training set images to calculate the best sizes for anchor boxes. The number of anchor boxes is set to 9. Figure 6 shows the architecture of the used CNNs during the character recognition stage. The CNN architecture comprises 162 layers as follows 110 Conv layers, 23 Shortcut layers, 21 Rout layers, 3 Maxpooling layers, 2 Up sample layers, and 3 YOLO layers. Two convolution sizes were used: 3 × 3 and 1 × 1. The shortcut layer adds the previous layer's content with the mentioned layer's content. The routing layer concatenates the contents of the mentioned layers. The three YOLO layers detect the characters at three different image scales (52 × 52, 26 × 26, 13 × 13). Each image contains a single Arabic word. The images are resized to 416 × 416 dimensions before entering the first layer of CNN architecture. In Arabic, there are 42 shapes of characters, each character could have up to four shapes as shown in Table 5. There is a high similarity between the same character shapes. Therefore, in this study, the number of character classes is set to 42.

To train a YOLO CNNs model for custom object recognition (Arabic characters), this study followed the YOLO4 recommendation and set the number of filters to 141 before each YOLO layer according to the following equation filters = (classes + 5) × 3. Table 6 shows samples of correctly recognized Arabic characters using the customized YOLO4 object detector.

Overlapped Bounding Boxes Processing

In Arabic, there is no overlapping between the characters of the word. However, during the character recognition stage, YOLO could have multiple bounding box results for each character. Each bounding box has x, y coordinates, confidence, and a class label. If the character has more than one prediction result, the prediction with the highest confidence is selected, and the other is ignored. The *x* and *y* coordinates are used to identify the overlap percentage between the detected bounding boxes for each word. The steps for removing the duplicated predictions are as follows:

1. Suppose that *L* and *C* denote the lists of predicted bounding boxes and confidences, respectively. *Oth* denotes the overlap threshold
2. For each pair of bounding boxes such as *a* and *b* in *L*
    1.2. Suppose that *v* denotes the overlap percentage between *a* and *b*
    2.2. If *v*>*Oth* remove the bounding box with the smallest confidence from *L*
3. Construct the predicted word by concatenating the list of characters in *L*

Table 7 shows sample results from checking the overlapping between characters. For example, In the first row of Table 7, the character Shiin (ش) has two prediction results (bold) Shiin: 97% and Siin: 38%. Therefore, Shiin prediction with 97% confidence is selected and another prediction is ignored (Siin: 38%). After processing the multiple recognitions for the same character and keeping the correct ones, the detected characters are concatenated based on their *x* and *y* coordinates to construct a word.

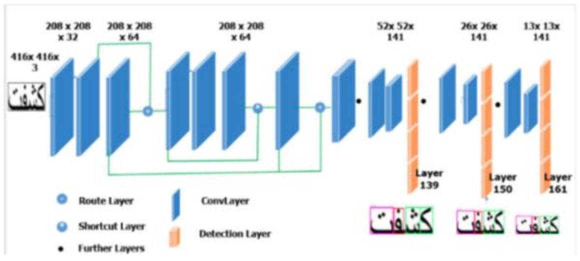


Fig. 6: Architecture of the customized YOLO4 for Arabic character recognition

Spell Check Correction

There is a high similarity between some Arabic letters. For example, in the first row of Table 8, the CNN model falsely predicts the letter "ـف" class as "ق". Therefore, the resulting word after constructing the whole predictions for the word " فاحشين" is " قاحشين". However, " قاحشين" does not exist in the Arabic dictionary and the suggestion "فاحشين "is found in the dictionary and it represents the correct recognition of the word.

Table 8 shows a sample of four corrected words after applying the spell-checking process. The are many open-source spell-checker libraries. One of the most popular is the Hunspell library. Hunspell is a fast and accurate spell checker. It covers 98 languages (Pirinen and Lindén, 2010). It is used by Google Chrome, Opera, Mac OS-X, OpenOffice.org, Thunderbird, LibreOffice, and Mozilla Firefox. The Hunspell database includes two files, the dictionary(*.dic) file, which consists of the list of the words with the applicable flags for each word, and the affix file (*.aff), which contains a list of flags and how these flags affect the word spelling. The C source code from (Hee, 2022) is modified to check the spelling of the Arabic word and find the correct suggestions for the wrong words. The Levenshtein distance (Edit distance) (Levenshtein, 1966) reduces the correction error between the word and its suggestions.

Results and Discussion

C language and Open CV library were used to implement the proposed PAOCR method. The source code of YOLOv4 was downloaded and modified to implement the proposed approach. MATLAB is used to evaluate the system performance by comparing the proposed system output (Text file) with the ground truth result. The experiments were performed in Windows 10 OS, intel (R) Core i5-8600k 3.6 GHz processor, 8GB RAM, and GTX 1080 GPU.

APTI Dataset

The Arabic Printed Text Images (APTI) dataset was created in 2009. The APTI represents one of the largest datasets for recognizing Printed Arabic text with 45313600 text images. Each image includes one Arabic word. The total number of Arabic characters is 250 million words. The dataset consists of six sets of images and the first five sets are publicly available. APTI provided an XML file for each image, including the ground truth annotation for the corresponding image. APTI includes Arabic text with different characteristics, including ten font types, ten sizes, and four styles. In this study, the first four sets were used to train the YOLO CNN model, whereas set 5 is used for evaluation the proposed approach. Arabic Transparent font type and Plain style were used in all experiments. Figure 7 shows sample pictures from the APTI dataset with different font sizes (6pt-24pt).

Commercial and Open-Source Arabic

OCR Software

ABBYY SDK OCR is commercial software. It is based on AI to solve the OCR problem. It supports around 200 different languages. It works with different operating systems, mainly Linux, Windows, and Mac. ABBYY recognizes handwritten and printed Arabic scripts and handles different file formats like TIFF, JPEG, and PDF. Since it is commercial software, the framework of the system is unknown. However, it is mentioned on the software website that a sequence of preprocessing operations is performed to enhance the image quality, such as binarization, rotation, and deskewing. In addition, it applies document analysis to recognize different image components and their formats, such as text fields, tables, diagrams, header and footer in the document. The trial version of the software allows registered users to process up to 1000 pages.

Tesseract (Smith, 2007) is an open-source OCR developed by Hewlett-Packard (HP) Lab between 1985 and 1994. Tesseract was sponsored and developed in the second round by Google between 2006 and 2018. Tesseract supports around 100 languages and works with various OS Linux, Windows, and Mac platforms. The latest Version 5 of Tesseract is used in this comparison. The main stage of the Tesseract includes converting an image into a black-white color image, splitting the text into words using the Fuzzy space, and identifying the Character's layout through the connected component technique. Finally, identify characters labeled using a classifier.

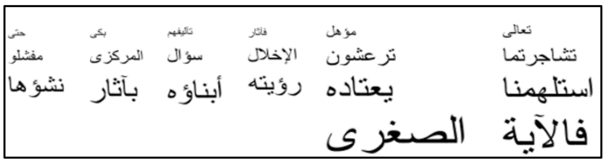

تعالى مؤهل فاثار تاليفهم بكى حتى
تشاجرتما ترعشون الإخلال سؤال المركزى مفشلو
استلهمنا يعتاده رؤيته أبناؤه بآثار نشؤها
فالآية الصغرى

Fig. 7: Samples from APTI dataset

Table 9: The WRR of the proposed approaches

Method	24	18	16
YOLO4	(667/1000)66.7%	(615/1000)61.5%	(595/1000)59.5%
YOLO4 + overlap-check	(799/1000)79.9%	(804/1000)80.4%	(718/1000)71.8%
YOLO 4+ overlap-check + Spell-check + EditDist	(824/1000)82.4%	(814/1000)81.4%	(755/1000)75.5%

Table 10: The CRR of the proposed approaches

Method	24	18	16
YOLO4	(5037/5539)90.90%	(5037/5539)90.90%	(4918/5539)88.80%
YOLO4 + overlap-check	(5313/5539)95.90%	(5299/5539)95.70%	(5109/5539)92.20%
YOLO4 + overlap-check + Spell-check + EditDist	(5300/5539)95.70%	(5275/5539)95.20%	(5088/5539)91.90%

Table 11: WRR comparison with existing OCR systems, Font size: 24, Plain

The proposed approach	Tesseract	ABBYY
(824/1000)82.4%	(710/1000)71%	(748/1000)74.8%

Table 12: CRR comparison with existing OCR systems, font size: 24, Plain

The proposed approach	Tesseract	ABBYY
(5300/ 5539)95.7%	(5112/ 5539)92.3%	(4696/5539)84.8%

Evaluation Metrics

The Word Recognition Rate (WRR) and Character Recognition Rate (CRR) are adopted to measure the accuracy of the proposed PAOCR. In Eq. (1), C denotes the correctly recognized words and A is the number of all tested words. In Eq. (2) $S_C, d_C \, and \, i_C$ denote the minimal number of character substitution, deletion, and insertion operations, to convert the OCR output to the correct text (ground truth). n_C is the total count of characters inside the text. Eq. (3) is used to compute CRR:

$$WRR(\%) = \frac{C}{A} * 100\% \quad (1)$$

$$CER(\%) = \frac{s_c + dc + i_c}{n_c} * 100\% \quad (2)$$

$$CRR(\%) = 100\% - CER \quad (3)$$

Results and Comparison with Other OCR Systems

A set of random 1000 images from set 5 of the APTI datasets is used to evaluate the proposed approach. The total number of the tested characters is 5539 characters. Table 9 shows the WRR of the proposed three techniques to solve the Arabic OCR problem. The three proposed techniques are tested at different font sizes (16 pt,18 t, and 24pt). The first proposed method is a custom YOLO4 model for Arabic character recognition.

In Table 9, the customized YOLO4 character recognition achieved 66.7% WRR for font size 24. The second proposed method is based on YOLO4 and Checking the overlapped bounding boxes for each character and keeping the most accurate bounding box based on the recognition probability. The second proposed technique improves the recognition rate by 13.2% (79.9%). The third approach is based on combining four techniques, YOLO4 for character recognition, processing the overlapped boxes for each recognized character, using Hunspell for spell correction, and finally, editing distance together achieving 82.4% accuracy. Edit distance is used with Hunspell to choose the closest suggestion. It is clear from Table 9 that the accuracy is increased by increasing the font size. The overall WRR of the proposed approach is 75.5% for font size 16pt, whereas it increased to 82.4% for font size 24. Table 10 shows that the proposed method CRR is 95.7%, a 4.8% improvement compared to YOLO4 (90.90% CRR). Diacritics exist in Arabic and Arabs do not use them frequently and still can understand the word's meaning from the context. Therefore Shaddah (ّ), tilde (~), and Hamza (ء) diacritic are ignored in the accuracy evaluation.

A comparison between the proposed approach and two well-known OCR engines shows the efficiency of the proposed Arabic OCR. Table 11 compares the proposed method, Tesseract, and ABBYY. The three systems were tested on 1000 pictures from the APTI dataset. The WRR of the proposed approach is 82.4%, the overall WRR of ABBYY is 74.8% and the Tesseract OCR achieved a 71% WRR. Table 12 shows that the CRR for the proposed approach is 95.7%, whereas Tesseract and ABBYY software reached 92.3% and 84.8% CRR, respectively. In Fig. 8 it is clear that the proposed approach has a higher WRR and CRR compared with ABBYY and Tesseract software.

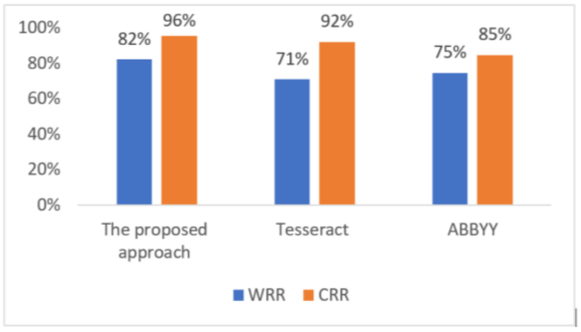


Fig. 8: Performance comparison between the proposed approach and Tesseract and ABBYY

Conclusion

This study proposes an Arabic OCR approach based on three techniques: A custom YOLO4 CNNs model is trained to recognize Arabic characters, and YOLO represents the state-of-the-art object detector. The accuracy of YOLO recognition is improved by removing the incorrect overlapped bounding boxes and keeping the most accurate ones. The Hunspell checker is used with edit distance to correct the word spelling. The proposed approach significantly improves the Arabic OCR accuracy compared with YOLO Arabic OCR. The proposed method has an 82.4% WRR accuracy, whereas the YOLO 4 accuracy is 66.7% (15.7% improvement). The accuracy is improved by increasing the script font size. Comparing the proposed method with two existing OCR software shows that the proposed approach outperformed the ABBYY and the Tesseract software in terms of WRR and CRR by a significant percentage. Future work includes assessing the effect of applying different CNNs architectures to recognize Arabic scripts and using deep learning to develop an accurate Arabic OCR system for handwritten scripts.

Acknowledgment

I thank the editor and the reviewers for the valuable comments and suggestions to enhance and revise the manuscript.

Funding Information

The author received no specific fund from any agency.

Ethics

This article is original and contains unpublished material. The authors confirm that they have read and approved this document and that no ethical issues are involved.

References

Abdo, H. A., Abdu, A., Manza, R. R., & Bawiskar, S. (2022). An approach to the analysis of Arabic text documents into text lines, words and characters. *Indonesian Journal of Electrical Engineering and Computer Science*, 26 (2), 754-763. https://doi.org/10.11591/ijeecs.v26.i2.pp754-763

Aburass, S., Huneiti, A., & Al-Zoubi, M. B. (2022). Classification of Transformed and Geometrically Distorted Images using Convolutional Neural Network. *Journal of Computer Science*, 18 (8), 757.769. https://doi.org/10.3844/jcssp.2022.757.769

Addakiri, K., & Bahaj, M. (2012). Online handwritten Arabic character recognition using artificial neural network. *International Journal of Computer Applications*, 55(13), 42-46. https://doi.org/10.5120/8819-2819

Afroge, S., Ahmed, B., & Mahmud, F. (2016, December). Optical character recognition using back propagation neural network. *In 2016 2nd International Conference on Electrical, Computer & Telecommunication Engineering (ICECTE)* (pp. 1-4). IEEE. https://doi.org/10.1109/ICECTE.2016.7879615

Ahmed, S. B., Naz, S., Razzak, M. I., & Yousaf, R. (2017, April). Deep learning based isolated Arabic scene character recognition. *In 2017 1st International Workshop on Arabic Script Analysis and Recognition (ASAR) (pp. 46-51). IEEE.* https://doi.org/10.1109/asar.2017.8067758

Al-Amin, B., & Alsaade, F. W. (2017). On Arabic character recognition employing hybrid neural network. *International Journal of Advanced Computer Science and Applications*, 8(6), 96-101. https://doi.org/10.14569/ijacsa.2017.080612

Albashir, T. M., Alzoubi, H., & Albatainih, M. (2020). Improving Arabic Instant Machine Translation: The Case of Arabic Triangle of Language. *Journal of Computer Science*, 16(7), 956–965. https://doi.org/10.3844/jcssp.2020.956.965

Alghyaline, S. (2019). A real-time street actions detection. *International Journal of Advanced Computer Science and Applications*, 10(2), 322-329. https://doi.org/10.14569/ijacsa.2019.0100243

Alghyaline, S. (2020). Real-time Jordanian license plate recognition using deep learning. *Journal of King Saud University-Computer and Information Sciences.* 34(6), 2601-2609 https://doi.org/10.1016/j.jksuci.2020.09.018

Alkhateeb, F., Abu Doush, I., & Albsoul, A. (2017). Arabic optical character recognition software: A review. *Pattern Recognition and Image Analysis*, 27(4), 763-776. https://doi.org/10.1134/S105466181704006X

Balaha, H. M., Ali, H. A., Youssef, E. K., Elsayed, A. E., Samak, R. A., Abdelhaleem, M. S., ... & Mohammed, M. M. (2021). Recognizing arabic handwritten characters using deep learning and genetic algorithms. *Multimedia Tools and Applications*, 80(21), 32473-32509. https://doi.org/10.1007/s11042-021-11185-4

Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv*:2004.10934. http://arxiv.org/abs/2004.10934

Boudelaa, S., Pulvermüller, F., Hauk, O., Shtyrov, Y., & Marslen-Wilson, W. (2010). Arabic morphology in the neural language system. *Journal of Cognitive Neuroscience*, 22(5), 998-1010. https://doi.org/10.1162/jocn.2009.21273

Bukhari, S. S., Kadi, A., Jouneh, M. A., Mir, F. M., & Dengel, A. (2017, November). anyocr: An open-source ocr system for historical archives. In 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR) (Vol. 1, pp. 305-310). IEEE. https://doi.org/10.1109/ICDAR.2017.58

Butt, H., Raza, M. R., Ramzan, M. J., Ali, M. J., & Haris, M. (2021). Attention-based CNN-RNN Arabic text recognition from natural scene images. *Forecasting*, 3(3), 520-540. https://doi.org/10.3390/forecast3030033

Chaudhuri, A., Mandaviya, K., Badelia, P., & Ghosh, S. K. (2017a). Optical character recognition systems. In Optical Character Recognition Systems for Different Languages with Soft Computing (pp. 9-41). *Springer, Cham*. https://doi.org/10.1007/978-3-319-50252-6_4

Chaudhuri, A., Mandaviya, K., Badelia, P., & Ghosh, S. K. (2017b). Optical Character Recognition Systems for Hindi Language. *In Studies in Fuzziness and Soft Computing* (Vol. 352, pp. 193–216). https://doi.org/10.1007/978-3-319-50252-6_8

Darwish, S. M., & Elzoghaly, K. O. (2020). An enhanced offline printed Arabic OCR model based on bio-inspired fuzzy classifier. *IEEE Access*, 8, 117770-117781. https://doi.org/10.1109/ACCESS.2020.3004286

Doush, I. A., Alkhateeb, F., & Gharaibeh, A. H. (2018). A novel Arabic OCR post-processing using rule-based and word context techniques. *International Journal on Document Analysis and Recognition (IJDAR)*, 21(1), 77-89. https://doi.org/10.1007/s10032-018-0297-y

Elleuch, M., Maalej, R., & Kherallah, M. (2016). A new design based-SVM of the CNN classifier architecture with dropout for offline Arabic handwritten recognition. *Procedia Computer Science*, 80(2016): 1712-1723. https://doi.org/10.1016/j.procs.2016.05.512

Fanany, M. I. (2017, May). Handwriting recognition on form document using convolutional neural network and support vector machines (CNN-SVM). *In 2017 5th international conference on information and communication technology (ICoIC7)* (pp. 1-6). IEEE. https://doi.org/10.1109/ICoICT.2017.8074699

Farghaly, A., & Shaalan, K. (2009). Arabic natural language processing: Challenges and solutions. *ACM Transactions on Asian Language Information Processing (TALIP)*, 8(4), 1-22. https://doi.org/10.1145/1644879.1644881.http

Fasha, M., Hammo, B., Obeid, N., & Widian, J. (2020). A hybrid deep learning model for arabic text recognition. *arXiv preprint arXiv*:2009.01987. https://doi.org/10.14569/IJACSA.2020.0110816

Girshick, R. (2015). Fast r-cnn. *In Proceedings of the IEEE international conference on computer vision*, (pp. 1440-1448). https://doi.org/10.1109/ICCV.2015.169

Hee, T. K. C. (2022). Spell Checker using Hunspell library. http://www.talkplayfun.com/Spell-Checker/

Jbrail, M. W., & Tenekeci, M. E. (2022). Character Recognition of Arabic Handwritten Characters Using Deep Learning. *Journal of Studies in Science and Engineering*, 2(1), 32-40. https://doi.org/10.53898/josse2022213

Jiang, C., Ren, H., Ye, X., Zhu, J., Zeng, H., Nan, Y., ... & Huo, H. (2022). Object detection from UAV thermal infrared images and videos using YOLO models. International *Journal of Applied Earth Observation and Geoinformation*, 112, 102912. https://doi.org/10.1016/j.jag.2022.102912

Joseph, F. J. J. (2020). Effect of supervised learning methodologies in offline handwritten Thai character recognition. *International Journal of Information Technology*, 12(1), 57-64. https://doi.org/10.1007/s41870-019-00366-y

Lee, S. G., Sung, Y., Kim, Y. G., & Cha, E. Y. (2018). Variations of AlexNet and GoogLeNet to improve Korean character recognition performance. *Journal of Information Processing Systems*, 14(1), 205-217. https://doi.org/10.3745/JIPS.04.0061

Levenshtein, V. I. (1966, February). Binary codes capable of correcting deletions, insertions, and reversals. *In Soviet physics doklady*, 10(8): 707-710.

Li, Q., An, W., Zhou, A., & Ma, L. (2016, August). Recognition of offline handwritten Chinese characters using the Tesseract open source OCR engine. *In 2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)* (Vol. 2, pp. 452-456). IEEE. https://doi.org/10.1109/IHMSC.2016.239

Liambas, C., & Saratzidis, M. (2016, October). Autonomous OCR dictating system for blind people. *In 2016 IEEE Global Humanitarian Technology Conference (GHTC)* (pp. 172-179). IEEE. https://doi.org/10.1109/GHTC.2016.7857276

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd: Single shot multibox detector. In European conference on computer vision (pp. 21-37). *Springer, Cham.* https://doi.org/https://doi.org/10.1007/978-3-319-46448-0_2

Lu, Z. A., Bazzi, I., Kornai, A., Makhoul, J., Natarajan, P. S., & Schwartz, R. (1999, January). Robust language-independent OCR system. *In 27th AIPR Workshop: Advances in Computer-Assisted Recognition* (Vol. 3584, pp. 96-104). *SPIE.* https://doi.org/10.1117/12.339811

Mathew, M., Singh, A. K., & Jawahar, C. V. (2016, April). Multilingual OCR for Indic scripts. *In 2016 12th IAPR Workshop on Document Analysis Systems (DAS)* (pp. 186-191). *IEEE.* https://doi.org/10.1109/DAS.2016.68

Mehta, H., Singla, S., & Mahajan, A. (2016, May). Optical Character Recognition (OCR) system for Roman script & English language using Artificial Neural Network (ANN) classifier. *In 2016 International Conference on Research Advances in Integrated Navigation Systems (RAINS)* (pp. 1-5). *IEEE.* https://doi.org/10.1109/RAINS.2016.7764379

Mehta, M., Sanchati, R., & Marchya, A. (2010). Automatic cheque processing system. *International Journal of Computer and Electrical Engineering*, 2(4), 761. https://doi.org/10.7763/ijcee.2010.v2.224

Mustafa, M. E., & Elbashir, M. K. (2020). A deep learning approach for handwritten Arabic names recognition. *International Journal of Advanced Computer Science and Applications*, 11(1): 678-682. https://doi.org/10.14569/ijacsa.2020.0110183

Nashwan, F. M., Rashwan, M. A., Al-Barhamtoshy, H. M., Abdou, S. M., & Moussa, A. M. (2017). A holistic technique for an Arabic OCR system. *Journal of Imaging*, 4(1), 6. https://doi.org/10.3390/jimaging4010006

Nguyen, K. C., & Nakagawa, M. (2016). Text-line and character segmentation for offline recognition of handwritten japanese text. *IEICE technical report*, 115(517): 53-58.

Patel, C., Patel, A., & Patel, D. (2012). Optical character recognition by open source OCR tool tesseract: A case study. *International Journal of Computer Applications*, 55(10), 50-56. https://doi.org/10.5120/8794-2784

Pirinen, T., & Lindén, K. (2010). Creating and weighting hunspell dictionariesas finite-state automata. *Investigationes Linguisticae*, 21(2010), 1-16. https://doi.org/10.14746/il.2010.21.1

Radwan, M. A., Khalil, M. I., & Abbas, H. M. (2016, September). Predictive segmentation using multichannel neural networks in Arabic OCR system. In IAPR Workshop on Artificial Neural Networks in Pattern Recognition (pp. 233-245). *Springer, Cham.* https://doi.org/10.1007/978-3-319-46182-3_20

Radwan, M. A., Khalil, M. I., & Abbas, H. M. (2018). Neural networks pipeline for offline machine printed Arabic OCR. *Neural Processing Letters*, 48(2), 769-787. https://doi.org/10.1007/s11063-017-9727-y

Rahal, N., Tounsi, M., Hussain, A., & Alimi, A. M. (2021). Deep sparse auto-encoder features learning for arabic text recognition. *IEEE Access*, 9, 18569-18584. https://doi.org/10.1109/ACCESS.2021.3053618

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *In Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788) https://doi.org/10.1109/CVPR.2016.91

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, 28(2015), 1-9. https://doi.org/10.1016/j.biocon.2012.08.014

Saidane, Z., & Garcia, C. (2007, September). Automatic scene text recognition using a convolutional neural network. *In Workshop on Camera-Based Document Analysis and Recognition* (Vol. 1).

Sharma, T., Debaque, B., Duclos, N., Chehri, A., Kinder, B., & Fortier, P. (2022). Deep Learning-Based Object Detection and Scene Perception under Bad Weather Conditions. *Electronics*, 11(4), 563. https://doi.org/10.3390/electronics11040563

Slimane, F., Ingold, R., Kanoun, S., Alimi, A. M., & Hennebert, J. (2009, July). A new arabic printed text image database and evaluation protocols. *In 2009 10th International Conference on Document Analysis and Recognition* (pp. 946-950). IEEE. https://doi.org/10.1109/ICDAR.2009.155

Smith, R. (2007, September). An overview of the Tesseract OCR engine. *In Ninth international conference on document analysis and recognition (ICDAR 2007) (Vol. 2, pp. 629-633). IEEE.* https://doi.org/10.1109/ICDAR.2007.4376991

Srihari, S. N., Hong, T., & Srikantan, G. (1997). Machine-printed Japanese document recognition. *Pattern Recognition*, 30(8), 1301-1313. https://doi.org/10.1016/S0031-3203(96)00168-9

Stolyarenko, A., Dershowitz, N., & Aviv, T. (2011). OCR for Arabic using SIFT descriptors with online failure prediction. *Imaging*, 3(1), 1-10.

The World Bank. (2022). Arab World Population. https://www.worldbank.org/

Tian, Z., Huang, W., He, T., He, P., & Qiao, Y. (2016, October). Detecting text in natural image with connectionist text proposal network. In European conference on computer vision (pp. 56-72). *Springer, Cham*. https://doi.org/10.1007/978-3-319-46484-8_4

TzuTa. (2022). labelImg. https://github.com/tzutalin/labelImg

Yang, Z., Doman, K., Yamada, M., & Mekada, Y. (2019, March). Character recognition of modern Japanese official documents using CNN for imbalanced learning data. *In International Workshop on Advanced Image Technology (IWAIT)* 2019 (Vol. 11049, pp. 25-28). SPIE. https://doi.org/10.1117/12.2521307

Yin, Y., Zhang, W., Hong, S., Yang, J., Xiong, J., & Gui, G. (2019). Deep learning-aided OCR techniques for Chinese uppercase characters in the application of Internet of Things. *IEEE Access*, 7(2019, 47043-47049. https://doi.org/10.1109/ACCESS.2019.2909401

Yousefi, M. R., Soheili, M. R., Breuel, T. M., & Stricker, D. (2015, February). A comparison of 1D and 2D LSTM architectures for the recognition of handwritten Arabic. *In Document Recognition and Retrieval XXII (Vol. 9402, pp. 146-155). SPIE*. https://doi.org/10.1117/12.2075930

Zanwar, S. R., Shinde, U. B., Narote, A. S., & Narote, S. P. (2020). Handwritten English Character Recognition Using Swarm Intelligence and Neural Network. In Intelligent Systems, Technologies and Applications (pp. 93-102). *Springer, Singapore*. https://doi.org/10.1007/978-981-15-3914-5_8

Zayene, O., Amamou, S. E., & BenAmara, N. E. (2017, October). Arabic video text recognition based on multi-dimensional recurrent neural networks. *In 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)* (pp. 725-729). IEEE. https://doi.org/10.1109/AICCSA.2017.126

Zou, J., Zhang, J., & Wang, L. (2019). Handwritten Chinese Character Recognition by Convolutional Neural Network and Similarity Ranking. *arXiv preprint arXiv*:1908.11550. http://arxiv.org/abs/1908.11550