International Journal on Document Analysis and Recognition (IJDAR) (2023) 26:273–287
https://doi.org/10.1007/s10032-023-00436-9

SPECIAL ISSUE PAPER

Printed Ottoman text recognition using synthetic data and data augmentation

Esma F. Bilgin Tasdemir[1]

Received: 15 November 2022 / Revised: 8 February 2023 / Accepted: 16 April 2023 / Published online: 24 May 2023

Abstract
The Ottoman script, which was in use for over five centuries, is an Arabic alphabet-based writing system. It became obsolete after the change of alphabet in Turkey. There are plenty of Ottoman documents, overwhelmingly printed in Naskh style. This work presents a DL-based character recognition system for the printed Ottoman script. We first generate a synthetic text image dataset from a text corpus and then augment it using some image processing methods. We develop a hybrid convolutional neural network-bidirectional long short-term memory recognizer and train it with the original and the augmented datasets. Finally, we apply a transfer learning procedure for adapting the system to real image data. The proposed system obtains **0.11** CER on synthetic data and **0.16** CER on real data comprising of line images from a printed historical Ottoman book.

Keywords Ottoman text recognition · Optical character recognition · Deep learning · Turkish

1 Introduction

Optical character recognition (OCR) is the computer technology for converting text images into machine-encoded text. Using a suitable OCR method, images of handwritten, typed and printed texts can be converted to a computer-readable format, making these documents editable and searchable. OCR studies date back to the 1950 s [36] with a primary focus on recognizing printed text. The recognition accuracy of modern OCR systems is increasing with the advances in computer vision, natural language processing (NLP) and deep learning (DL) [37]. Nevertheless, the recognition problem is still far from being solved for documents with poor image quality, especially for specific scripts, fonts and handwriting.

Most of the earlier studies were targeting OCR of Latin-based scripts. Other writing systems started to receive attention in the last few decades. In parallel with this trend, there is a recent increase in the number of studies on OCR of Arabic and Arabic alphabet-based scripts like Urdu, Persian and Pashto [4, 39, 60, 69]. Such writing systems share the common feature of cursiveness and contextual character shape variance which introduce extra challenges for OCR tasks and directly affect the recognition performance.

The Ottoman script is based on the Arabic alphabet as well. It was a writing system of the Turkish language for several centuries until it was replaced with the modern Turkish script, which is based on the Latin alphabet, in 1928. Today, the Ottoman script is readable to only a limited number of scholars and enthusiasts. However, there is a huge number of printed and handwritten Ottoman documents preserved in archives, libraries, museums and private collections in different countries. For instance, there are approximately 25, 500 records for Ottoman Turkish books that are published between 1729 and 1928 in one of the standard catalogs of Turkish literature [41]. The total number of manuscripts and printed materials other than books is much higher than this figure.

On the other hand, the recent upsurge of digitization of Ottoman documents resulted in many institutions' opening up their collections in the form of digital images. However, the content of the documents is not readily accessible since these documents are not converted to editable and searchable digital text.

OCR and text recognition technologies can be a solution to this problem in the form of automated and semi-automated conversion systems. Existing OCR systems that are designed for the Arabic script do not provide a viable solution for the Ottoman script because it has extra characters that do not exist

✉ Esma F. Bilgin Tasdemir
esmabilgin.tasdemir@medeniyet.edu.tr

[1] Department of Information and Document Management, Istanbul Medeniyet University, 34704 Istanbul, Turkey

in the Arabic alphabet. Furthermore, differences between languages of training data can have a negative effect on recognition output since models can learn character dependencies from the training samples, incorporating an inherent character-level language model into the optical recognition. A better solution would be a recognition system explicitly designed for the Ottoman script. However, developing a modern recognition system requires a sufficiently large training dataset to achieve state-of-the-art recognition performance.

Unfortunately, there is no publicly available dataset for the Ottoman script. Generating such a dataset requires collecting a sufficient amount of image data and then labeling it manually which is an error-prone and time-consuming task. A recent solution to the data scarcity problem is using synthetic data for training recognition systems [34, 35, 60]. Generating data that mimics real data is an efficient way of creating abundant training data for supervised systems. Synthetic and real data are usually used together within a transfer learning scheme.

This study presents a DL-based character recognition system for the Ottoman script. In this work, we first synthesize data by rendering images from a corpus of Ottoman text to obtain a sufficiently large training dataset. Then, we augment this dataset by exploiting some image processing methods to increase its size. We train a neural model with the augmented data, and finally, we use the transfer learning (TL) technique to fine-tune the system with real data. Also, we evaluate the option of using some standard Arabic datasets from the literature to recognize the Ottoman script and vice versa. Our contributions are as follows; (i) we generate a corpus of Ottoman digital text from the web, (ii) we construct a synthetic machine-printed Ottoman line image dataset using the corpus, (iii) we increase the size of the dataset by augmenting original data using some image processing techniques, (iv) we develop a recognition system for machine-printed Ottoman script (v) we adapt the system to recognize characters in line images extracted from real books and (vi) we report on the usability of datasets with a language other than that of the evaluation data.

The rest of the article is organized as follows; Sect. 2 introduces the Naskh style and the challenges of the Ottoman script are explained in Sect. 3. In Sect. 4, a summary of the related work is presented. Section 5 contains information about the proposed solution including the details on the dataset and methodology used. The experiments and results are described in Sect. 6. Section 7 contains remarks on the experimental results and a conclusion and future work is presented in Sect. 8.

2 Naskh style

Naskh is one of the most popular calligraphic styles for Arabic alphabet-based scripts. It was preferred in producing handwritten documents and manuscripts where legibility is essential. Naskh has been widely used in printed documents and books as well. There are many digital fonts based on the conventional Naskh style.

Fig. 1 The Ottoman characters with positional shapes. Letters that do not join to the left have no separate medial and initial forms. In that case, the final form is used in the medial position, and the isolated form is used in the initial position

Fig. 2 Multiple glyphs for three Arabic symbols with the DecoType Naskh font, all shown with red color. On the left; glyphs for isolated ر, in the middle; glyphs for medial ه and on the right; glyphs for medial م. Best viewed in color

Naskh has a large predefined set of character shapes. There are several reasons for that abundance of character shapes. The Arabic characters take different forms according to their position in a word (i.e., the initial, medial, final and isolated forms) as depicted in Fig. 1. In addition to the contextual forms, stylistic forms of Naskh provide alternative character shapes (see Fig. 2). Moreover, multiple characters can be represented together by a ligature (see Fig. 5) which further increase the size of the glyph set. Also, the Naskh style provides shapes for 'Harakat,' which are tiny diacritic marks used to show some unrepresented vowels, especially in manuscripts.

All of the Naskh fonts are designed based on the Naskh calligraphic style. However, some of them do not include all alternative forms of a character. Also, they differ in the rendering of the shapes; some of them closely reflect the conventional Naskh style (e.g., DecoType Naskh variations), while others apply some simplifications (e.g., Microsoft Traditional Arabic).

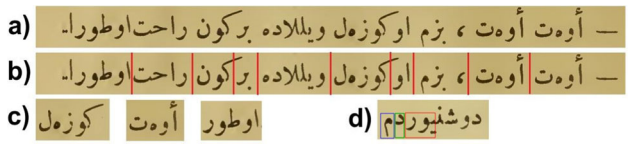

Fig. 3 The spacing problem of the Ottoman script: **a** a line from a printed book, **b** word boundaries marked with red bars in the line, **c** words from the line which have within-word spaces due to non-joiner characters, **d** suffixes ending with non-joiner characters; suffixes are shown in boxes each marked with a different color

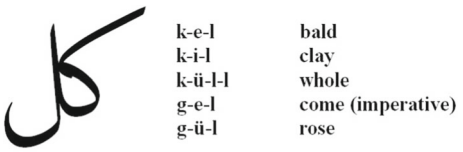

Fig. 4 Five different words share the same shape. The absent vowels can be seen in the modern Turkish transcriptions given in the middle column. The rightmost column contains the English translations. Here, one Ottoman character maps onto multiple modern Turkish characters

Naskh had widespread use in handwritten Ottoman documents. Similarly, most printed historical Ottoman documents have the Naskh style as the primary font type, while other calligraphic styles can be found in headers and title pages. Following that tradition, much of the contemporary works, both printed and digital, favor Naskh.

3 The Ottoman script

The Ottoman Script is a cursive writing system where the characters are written right-to-left by connecting to each other. Being a derivative of the Arabic alphabet, the Ottoman alphabet has 34 characters in total; 28 letters of the Arabic alphabet with the addition of five extra symbols to represent some Turkish sounds. Figure 1 shows contextual shapes of the Ottoman characters. Some characters share the same body shape and differ in only dots.

Some of the difficulties in recognizing the Ottoman script arise from some features of the Arabic script, such as cursiveness and contextual character forms. There are several additional challenges specific to the Ottoman script. One such challenge stems from non-joiner characters which break the cursiveness and create within-word spaces.

Modern Latin script-based writing systems use spaces that usually denote word boundaries. However, the Ottoman script has two types of space; between the words as word separators and within the words due to the non-joiner characters. Within-word spaces divide words into connected groups of characters, i.e., sub-words or ligatures. Sub-words may be meaningful individually, or they may be meaningful only with other sub-words. To further complicate the issue, between- and within-word spaces are usually used inconsistently in many of the printed real texts, as shown in Fig. 3.

Self-meaningful sub-words and within-word spaces can be confusing when determining word boundaries. Although the within-word spaces problem is common for writing systems using the Arabic alphabet, the Ottoman script is afflicted more severely because of the agglutinative nature of Turkish. New words are generated by adding one or more suffixes to the end of word roots in Turkish. In the Ottoman Script, some of the most frequent suffixes end with non-joiner characters, so there are more within-word spaces than in the Arabic script (see Fig. 3d).

Another challenge stems from the complex and sometimes inconsistent orthography of the Ottoman writing system. Although this one does not directly affect the character recognition task itself, we include it for the sake of completeness. In Ottoman script, some vowels are represented explicitly with a character, while others are unrepresented but deduced from the meaning of the word within the context. For example, in Fig. 4, several words are written exactly the same with two consonant characters, while they differ in the unrepresented vowel in the middle position. Furthermore, there is no standard orthography and the spelling of words can change with the writer and the period of the document. That results in the same word having different spellings.

4 Related work

This section presents a brief overview of works on the recognition of printed texts relevant to this work. For the sake of comprehensiveness, we include research on Arabic alphabet-based scripts related to our work. Since our work is focused on the Naskh font type, we report results for recognition of Naskh and Naskh-like fonts only. The section is organized into two subsections: (i) recognition of the Arabic script and (ii) recognition of other Arabic alphabet-based scripts

4.1 Recognition of the Arabic script

The literature on the recognition of printed and handwritten Arabic documents is much richer than that on the Ottoman script [4, 5, 8, 43, 48]. Due to the large set of contextual shapes of characters and the cursive nature of the Arabic script, Arabic text recognition systems have lower recognition accuracy than those designed for the recognition of Latin alphabet-based scripts. HMMs are preferred overwhelmingly in printed Arabic text recognition [2, 30, 38] while deep learning-based approaches are getting popular for Arabic handwriting recognition tasks [5, 7]. Also, some HMM-NNs hybrid models are proposed in the literature [30, 50].

The cursive nature of the Arabic script makes it hard to detect character boundaries in word shapes. Explicit segmen-

tation of images into words, part of words, characters and part of characters before the recognition phase is a solution to this problem. Alternatively, sequence learners based on specific techniques like HMMs and RNNs can perform implicit segmentation and recognition simultaneously without any need for explicit segmentation. An HMM system is used to recognize text images of different scripts, including Arabic in a script-independent OCR system in [38]. To overcome image noise, pixel percentile features, which reflect the rate of blackness with respect to image height, are devised. The system, which is trained with 192 pages randomly selected from the DARPA Arabic Machine Print (DAMP), obtained 0.03 CER on 102 pages of the same collection.

Another HMM-based recognition system is proposed to recognize printed Arabic offline text in [29]. Some statistical features are extracted by using the sliding window technique, where each window is divided into a predefined number of cells. Using a proprietary dataset of 1500 training lines and 1000 test lines, 0.14 WER is reported for the Naskh font type.

The system introduced in [38] is improved further by utilizing context-dependent HMMs, and parts of Arabic words (PAW) language models in [46]. Context dependency of the HMMs is provided with position-independent models where a separate set of Gaussians is estimated for each state of all HMMs associated with a particular character. Discriminative training is preferred over the Maximum Likelihood (ML) estimation for HMM parameters estimation. In a two-pass recognition scheme, the system achieves 0.096 WER on 60 images from the DARPA Arabic Machine Print (DAMP) by utilizing a PAW trigram language model.

Another HMM-based recognition system is proposed by Al-Muhtaseb et al. [6]. They use a novel hierarchical sliding window technique with overlapping and non-overlapping windows yielding 16 features. A dataset of line images is created by using multiple fonts. 2500 training samples and 266 test samples are generated from the same source text for each font type. Using seven states for each of the symbols from a 128-element codebook, they obtained over 0.02 CER for Naskh font.

A recognition system is proposed for low resolution images of words synthetically generated using three complex fonts in [62]. Two sets of hand-crafted features as features common to all fonts and features specifically designed for each font, are used in an HMM system. Utilizing a subset of the Arabic Printed Text Image (APTI) database, they propose a two-step scheme with font identification and recognition parts. The standard division of APTI is adopted as 75,557 word images for training and 18,868 word images for testing. They report 0.025 CER for 24 pt. Naskh font. A similar work proposes a bootstrapping method for character boundary determination by HMM state number optimization in small-size Arabic printed text recognition task in [27]. The performance of this work is measured as 0.02 WER and 0.003 CER on 6 pt. Arabic Transparent font subset of APTI dataset.

Printed text recognition systems can be designed to recognize multiple fonts and multiple font sizes. Font identification before the recognition phase is a viable strategy when designing multi-font recognition systems. For example, in [2], the authors use an adaptive sliding window approach in an HMM-based multi-font printed Arabic text recognition system. In a two-step approach, a given line image is first associated with the closest known font, which is then recognized by the HMM system that was trained for that particular font. In the case of unidentified fonts, the HMM associated with the most similar font is adapted to the new data by supervised adaptation technique if there are a sufficient amount of labeled data of unseen font or unsupervised adaptation otherwise. They evaluate the system on five selected fonts from the APTI dataset and obtain a CER of 0.036 on Traditional Arabic font which resembles the Naskh font. Additionally, they conduct experiments on the P-KHATT dataset they created by scanning images of printed Arabic text lines. They report 0.03 CER on 1424 test samples written in Naskh font.

NN-based approaches are also popular for the recognition of text from images. A multi-dimensional long short-term memory (MDLSTM)-CTC network is used for recognition of word samples from the APTI dataset in [52]. Raw image data as patches of pixels is fed to the network, and 0.006 WER and 0.003 CER are obtained on Naskh samples. Another deep network is presented in [30]. The authors propose an adaptive window positioning for HMM and deep neural network (DNN)-HMM hybrid models where a conventional sampling window is repositioned to align its center such that it overlaps with the center of gravity of the sampling area. They report mean CERs of 0.015 and 0.002 in recognition of Traditional Arabic and Arabic Transparent subsets of the APTI dataset, respectively, when the font type and size are known. As for the DNN-HMM model, they report a mean CER of 0.03 from tests with multiple sizes of Arabic Transparent subset of APTI.

A NN pipeline with three networks is presented in [49]. A document image is segmented into lines and sub-words by using conventional projection-based methods. A NN is designed to detect the font size of an input word for size normalization. A three-window input containing a frame and its left and right neighbors is used for capturing the context. Then a CNN is employed to segment words or sub-words into characters. Finally, another CNN is used to recognize characters from features output by the segmentation network. The reported mean WER and CER of the system are measured as 0.056 and 0.014, respectively, on Arabic Transparent subset of the APTI dataset.

In [50], authors use a hybrid network, combining a bag-of-feature (BoF) framework for feature extraction based on a deep sparse auto-encoder (SAE), and hidden Markov mod-

els (HMMs) for sequence recognition. They replace the k-means clustering method with an SAE for generating visual vocabulary (codebook) of extracted local features in a BoF representation. Using a sequence of BoF vectors extracted by the sliding window technique, they train an HMM system for open vocabulary, multi-size, multi-font and multi-resolution Arabic printed texts. They report a 0.024 CER on Naskh test samples of the P-KHATT dataset and 0.005 CER on Arabic Transparent test samples of the APTI dataset.

Another deep learning-based system is proposed in [57]. Using explicit character segmentation, the authors employ a CNN system to recognize multi-font, multi-size text line images from the APTID-MF dataset [26]. The average CER is increased from 0.05 to 0.078 by augmenting the data with affine transformations.

Arabic script is used as a writing system for languages other than Arabic as well. For example, Urdu and Farsi use alphabets modified from the Arabic alphabet. There is a recent increase in the number of works on recognition of both handwritten and machine-printed text recognition in such languages [23, 39, 53, 63, 66, 69]. In [45], an SVM classifier is used to recognize sub-words represented with features extracted by an Auto Encoder. The system is trained and tested on a proprietary, single-font, multi-size synthetic dataset and obtained 0.076 CER on B Nazanin font type. Rahmati et al. [51] presents a similar system for Farsi, where a CNN-LSTM network is trained with synthetic data. They report 0.007 CER for a proprietary, single-font, multi-size synthetic dataset generated with B Nazanin font type.

4.2 Recognition of other Arabic alphabet-based scripts

There is quite a limited number of studies dealing with Ottoman documents in the literature. Most of them focus on the retrieval of Ottoman documents using traditional computer vision techniques to map word shapes [9, 10, 12, 17, 56]. Some others constitute the early works on Ottoman character recognition using conventional machine learning algorithms with quite limited proprietary datasets [31, 32, 42, 68]. In a recent work, a CNN-based segmentation method is used for spotting numerals on handwritten Ottoman documents [13]. Spotted numerals are then classified by a system which is trained on Arabic handwritten digit datasets, employing the transfer learning method.

Recently, DL methods are employed for recognition of Ottoman documents in several works. An LSTM network is trained with hand-crafted features extracted from 169K handwritten word images in [11]. The performance of the proposed system is measured as 0.124 CER and 0.221 WER on a test set of images containing 1,000 unique words. Dolek and Kurt [15] presents a DL-based solution for printed Ottoman text line recognition. Using a Variable-size Graph Specification Language (VGSL) prototype by Tesseract-OCR engine, a CNN-LSTM network is trained with both synthetic and real data. The system performance is measured using the Ratcliff/Obershelp string-matching algorithm, which calculates the similarity of two strings as the number of matching characters divided by the total number of characters in the two strings. They report a character recognition rate of 88.86% and a word recognition rate of 64% on 21 pages of real data. Their test set recognition output is publicly available. We calculate the system's performance as 0.148 using the conventional CER measure based on the Levenshtein distance.

5 Proposed solution

We present a solution for the recognition of Ottoman printed text with limited real data. Our approach is based on generating synthetic data from an Ottoman text corpus, augmenting that data and applying transfer learning on the system trained with the augmented data to recognize real data.

The synthetic data generation includes collecting an Ottoman text corpus and its normalization, then segmenting text into lines with appropriate lengths and finally rendering images of the lines. It should be noted that the number of images generated is determined by the size of the source text corpus, which is relatively small in our case. An ad-hoc data augmentation procedure is used to increase the dataset size. Then, the synthetic data is used for training a CNN-BLSTM-CTC network. The network is fine-tuned later with a limited amount of real data containing line images extracted from a printed Ottoman book.

We explain the details regarding each of these steps in the rest of this section.

5.1 The need for an Ottoman dataset

A considerable amount of data is needed for developing an OCR system using DL techniques. The generation of such a collection requires acquiring samples first and then tagging them with ground-truth labels. This is an error-prone, labor-intensive and time-consuming task if done manually. A recent alternative to manual ground truth generation is using synthetically generated text images of which labels are readily available.

There are several rendered text image datasets for Arabic [34, 60] and Arabic script-based languages as Urdu [53], Farsi [25], Pashto [66] and Sindhi [23] generated at different levels of granularity (i.e., character, ligature, word or line). Although the Ottoman script is based on the Arabic alphabet, several issues make existing Arabic script datasets less usable for building a recognition system.

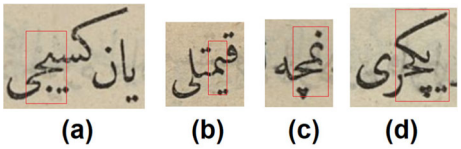

Fig. 5 Examples of ligatures from Ottoman books printed in the 19th century using a Naskh typeface; **a** يجى , **b** يم, **c** نمچ, **d** يكچ

First of all, the five additional letters in the Ottoman script which do not exist in the Arabic alphabet (see Sect. 3) would not be recognized correctly by an OCR system trained with an Arabic dataset.

Character occurrence frequencies vary between languages. Similarly, statistics of character co-occurrences are different for different languages. The Ottoman and Arabic scripts are dissimilar in that sense. The occurrence frequency of a specific character can directly impact recognition accuracy based on how complex and various the shape of that character is. For example, the occurrence of a ligature is directly related to the occurrence of the characters constituting that ligature in a certain order, which is a language-specific feature. The existence of complex and various shapes can make the recognition task harder due to the difficulty of decomposing character groups and confusion with other letterforms (see Fig. 5). Actually, ligature forms in Arabic can be a major source of recognition errors [61].

Finally, spaces occurring within words directly impact the word shapes (see Sect. 3). They divide the word into parts referred to as a part-of-Arabic-word (PAW). The number of PAWs per word and the number of characters per PAW are higher in the Ottoman script than in the Arabic script. For instance, the average number of PAWs per word and the average number of characters per PAW for the Arabic script are reported as 2.33 and 2.03 in [1], respectively. Using a corpus of approximately 150 K words, we found the average number of PAWs per word as 3.14 and the average number of characters per PAW as 4.18 for the Ottoman script, which indicates the Ottoman script has more within-word spaces and longer connected character groups. Longer PAWs require more rigorous effort for segmentation. Also, the frequency of occurrence for non-joining characters is higher. That leads to initial forms of characters occurring more frequently. Table 1 gives some statistics regarding the datasets.

It is known that RNNs and their variants capture character dependencies and incorporate a character-level language model to the recognition process [35, 54]. Thus, the optical model is somewhat affected by the language of training examples. The language of the training data and the test data should be the same for better recognition performance. Considering the differences mentioned above, it is necessary to have a dataset containing samples of Ottoman text images for building a recognition system specifically for the Ottoman script.

Table 1 Some statistics for the OPTI, APTI and P-KHATT Dataset

	OPTI	APTI	P-KHATT
Num. of words	133 K	113.3 K	106.9 K
Num. of unique words	40.3 K	113.3 K	32.7 K
Num. of characters	756 K	640 K	601 K
Num. of PAWs	308.4 K	227.2 K	205 K
Num. of unique PAWs	10.9 K	30.1 K	18.2 K
Avg. word length	4.8	2.4	2.4
Avg. PAW length	1.9	2.4	2.4
Avg. PAW/words	3.14	1.6	1.6

Unfortunately, there is neither a real image dataset nor a synthetically generated image dataset that is publicly available. One of this work's contributions is generating the synthetic Ottoman script line image dataset and making it accessible to researchers.

5.2 Synthetic data generation

A synthetic document image dataset can be generated by several methods. Cropping words or characters from already existing text images and composing them to words and lines [35], blending text images with different backgrounds and noise [14], rendering images of a given text [28, 34, 37] and image-to-image translation [44] are some of the common approaches each with different applicability in different problem domains. In this work, we use image rendering methods to generate synthetic data of which the ground truth is readily available. In our case, the ground truths are derived from a large text collection.

5.2.1 Source text collection

In the text image recognition domain, a synthetic data is an image of a symbol or a sequence of symbols with or without a background. The symbols come from a predefined set like digits or a specific alphabet. If the image contains a word, it usually comes from a lexicon. When the text is a word sequence, it is usually a sentence or an excerpt from a sentence to represent the use of words in relation with the context better. Such sentences are generally derived from a large text corpus which is sometimes referred to as the source text.

In order to obtain representative word sequences as lines for line image generation, we need an Ottoman text corpus. Within this study, such a text corpus is generated from online resources that are publicly available. The corpus comprises Ottoman texts covering various domains such as history, literature, law, geography and religion. Some of the texts are

نەدر	\u0646 + \u06d5 \u062f \u0631 Arabic Letter Noon + Arabic Letter Ae + Arabic Letter Dal +Arabic Letter Reh
نەدر	\u0646 + \u0647 + \u200c + \u062f + \u0631 Arabic Letter Noon + Arabic Letter Heh + Zero-Width-Non-Joiner + Arabic Letter Dal +Arabic Letter Reh
نە در	\u0646 + \u0647 + \u2009 + \u062f + \u0631 Arabic Letter Noon + Arabic Letter Heh + Thin-space + Arabic Letter Dal +Arabic Letter Reh
نەدر	\u0646 + \u0647 + \u200a + \u062f + \u0631 Arabic Letter Noon + Arabic Letter Heh + Hair-space + Arabic Letter Dal +Arabic Letter Reh

Fig. 6 Four different Unicode character sequences are used to render the same word in the source text

typed from Ottoman documents created before 1928 while the others are written by mostly unprofessional contemporary writers. Due to the lack of a standard orthography for the Ottoman Turkish, there is a diversity of spellings observed in the corpus. Also, we built another collection containing mostly literature texts dated before 1928, typed by a human typist. These two collections are merged to develop a corpus containing 133K words and 40.3 K unique words.

5.2.2 Source text normalization

We observe that writers and content creators use different Unicode symbol sequences to write the same word in different texts, as exemplified in Fig. 6. In order to normalize the source text to remove symbol variance, we first generate a list of unique symbols. Then we map alternative symbols used for a character to its general Arabic Unicode code. We end up with a list of 72 symbols, including 41 letters, 19 punctuation symbols, 10 Arabic digits, space and Zero-Width-Non-Joiner (ZWNJ) character, which forces its previous character to take its final form. ZWNJ usually follows ه and ی. It introduces a break to the cursiveness of words. It should be noted that the letter set includes some extra symbols like composite characters, which have a single Unicode value, such as "Arabic Letter Waw With Hamza Above" ؤ and "Arabic Letter Alef with Madda Above" آ. No diacritic mark is included in the symbol set.

The text collected from websites is organized in running word sequences rather than lines and paragraphs. Similarly, there are paragraphs of running sentences in the typed collection. Therefore, we segment them into lines based on the distribution of the number of words per line derived from 2528 real printed book lines. After the line splitting process, we obtain 14, 800 lines with a minimum line length of 3 characters and a maximum line length of 99.

5.2.3 Image generation

After the corpus is normalized and split into lines, we render images from text lines in grayscale. DecoType Naskh®TrueType Font closely resembles the Naskh font used in Ottoman

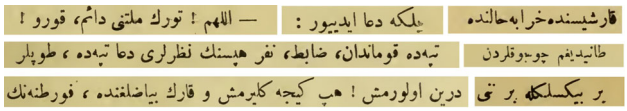

Fig. 7 Examples from the real image dataset, some of which are afflicted with imperfections like broken characters, touching characters, and faded or excessive ink

printed materials. Therefore we use that font in rendering text with 24 pt. font size on a solid color background using the Python PIL library. We use the regular version of DecoType Naskh font, which does not include more complex ligatures covered in other versions such as Swashes and Variant. For each line image, we generate the ground truth as Unicode values of the characters in that text line. In that scheme, different positional forms of a letter are represented with the same Unicode value.

The resulting dataset, Ottoman Printed Text Image (OPTI), is the first synthetic Ottoman text image dataset in the literature, to the best of our knowledge. There are 14, 800 grayscale line images in the OPTI dataset. It should be noted that the size of the line image dataset, which is comparable to sizes of similar datasets, is determined by the size of the corpus we generated.

5.3 Data augmentation

When there is not enough data available, data augmentation is a common technique to increase dataset size artificially [23, 34, 53, 60, 66]. Several image augmentation methods are used in the previous studies [3, 47, 53, 63, 69].

We design a pipeline of image manipulation methods to mimic various distortions observed in printed historical Ottoman documents, which can be seen in Fig. 7. Specifically, we add blotches of various sizes to break the characters as in historical documents. We also use erosion, dilation and opening to mimic the ink-related imperfections in real images. The pipeline contains scaling and rotation around the center as well. We apply all of the augmentations probabilistically with randomized parameters.

The input of the augmentation pipeline is a 72-dpi grayscale image. It is scaled by a percentage ranging from 80 to 120 with a probability of 0.7. A rotation by an angle between ± 3 degrees is applied with a 0.8 probability. Blotches of size 3×3 pixels are probabilistically added to the image. The number of blotches is decided as image width divided by image height. Then the image is binarized before the morphological operations. A kernel is chosen randomly from the set of kernels $s = \{(1, 1), (1, 2), (2, 1), (2, 2), (1, 3), (3, 1)\}$. Either one of the morphological operations is applied using the kernel, or the image is kept unchanged probabilistically. The final image is binary with a resolution of 72 dpi. The height of the image

is not the same as the input image if it is changed by one or more of the augmentation operations. Height normalization is carried out as a preprocessing step (see Sect. 6).

5.4 Recognition of the Ottoman characters

Convolutional neural networks (CNNs) have been in use for a while as an automatic feature extraction and recognition method in image recognition and pattern classification tasks [33]. Their robustness to shape variations, distortions and translations and their ability to work on raw pixels without preprocessing made CNNs very preferable.

CNNs that do not have trainable layers are used in combination with other network types. In that case, CNNs learn visual features which are fed to a trainable network such as recurrent neural networks (RNNs) or support vector machines (SVMs) [24, 47]. Due to their ability to capture temporal dependencies, RNNs and their variants like long short-term memory (LSTM), bidirectional-LSTMs, multi-dimensional LSTMs (MDLSTMs) and gated recurrent units (GRUs) have been effectively used for sequence labeling tasks [20–22, 58, 63, 65]. Their hybrid models with CNNs are shown to be quite successful in studies as well [16, 39, 59]. Particularly, CNN-MDLSTM hybrid systems have been quite successful and popular in sequence labeling tasks in recent years. One disadvantage of MDLSTM networks is their complexity which makes them computationally costly. On the other hand, it is shown that a system with one-dimensional LSTMs can achieve better or equivalent results at a lower computational cost in some handwriting recognition tasks [47]. Following these successful applications, we propose a CNN-BLSTM hybrid network with a connectionist temporal classification (CTC) layer. Similar networks are used for isolated word recognition in [59] and handwritten line recognition in [47].

Our method consists of the following steps. First, grayscale line images are resized with padding to the maximum width and height of the samples in the dataset. The ground truth for each image is its transcription with a sequence of characters from the defined alphabet, which are the symbols occurring in the dataset. Next, the input samples are fed onto convolutional layers to extract features. Then, BLSTM layers process the features column-wise in left-to-right and right-to-left directions. The output of BLSTM layers are concatenated depth-wise to generate a block which is then mapped to an output label represented as a sequence of characters by the CTC algorithm [19]. In this fashion, the system outputs a probability distribution over a variable-length output sequence without explicit segmentation of the input sequence. The network configuration and other details can be found in Sect. 6. Once we pre-train the recognition system using the synthetic data, we apply TL by training the trained system with real data containing line images cropped from historical documents. Details of the pre-training and TL training are explained in Sect. 6 as well.

Table 2 Network configuration summary. When a parameter is not applicable for a layer its cell is shown as "–"

Layer	Filters	Kernel	Stride	Padding	Nodes	Probability
Input	–	–	–	–	–	–
Convolution	32	5*5	1*1	1*1	–	–
MaxPooling	–	2*2	2*2	–	–	–
Convolution	64	5*5	1*1	1*1	–	–
MaxPooling	–	2*2	2*2	–	–	–
Convolution	128	3*3	1*1	1*1	–	–
Batch Norm	–	–	–	–	– –	–
Convolution	128	3*3	1*1	1*1	–	–
MaxPooling	–	2*2	2*1	–	–	–
Dropout	–	–	–	–	–	0.2
Convolution	256	3*3	1*1	1*1	–	–
Batch Norm	–	–	–	–	– –	
Dropout	–	–	–	–	–	0.2
BLSTM	–	–	–	–	128	–
Dropout	–	–	–	–	–	0.5
BLSTM	–	–	–	–	128	–
Dropout	–	–	–	–	–	0.5
Dense	–	–	–	–	72	–
Output	–	–	–	–	–	–

6 Experiments

We design a set of experiments, each with a specific goal towards building an Ottoman script recognition system. This section presents the experimental setup, the experiments and their results in detail.

6.1 Experimental setup

Our CNN-BLSTM network configuration, which is decided empirically, is given in Table 2. We use 5 blocks of convolutional layers, where three of them use a max pooling layer. We apply two batch normalization layers after convolutional layers at the 3rd and 5th blocks. A dropout layer is added to the last two blocks with a probability equal to 0.2. There are 3 Maximum Pooling layers at the first, second, and fourth blocks with vertical and horizontal strides of (2, 2), (2, 2) and (2, 1), respectively. The activation functions of the convolutional blocks are Rectifier Linear Units (ReLU). The convolutional blocks are followed by two recurrent blocks, each formed by a BLSTM layer of 128 hidden units and a dropout layer with a probability of 0.5. Finally, a CTC layer maps the output from recurrent blocks to one of the 70 sym-

bols; 69 symbols are the ones appearing in the dataset and one blank symbol.

Python and TensorFlow libraries are used for the implementation of the CNN-BLSTM-CTC network. We run the experiments on a NVIDIA GeForce RTX 2060 graphical processing unit (GPU) card. The network is trained to minimize the CTC loss function. We use the RMSprop optimization algorithm with a learning rate of 0.0003 and mini-batches of size 32 during pre-training. The network weights are initialized using the Glorot uniform technique [18]. In all experiments, training is stopped when the loss on the validation set does not improve after 20 epochs.

We evaluate the performance of our system by calculating character error rate (CER) percentage using the edit distance which is calculated as the minimum number of edits with substitution, insertion and deletion of characters from the reference string to the output, normalized by the number of reference characters. CER calculation is formularized below, where $Insertion$, $Deletion$ and $Substitution$ refer to the minimum number of insertion, deletion and substitution operations required for the transformation.

$$\text{CER} = \frac{\text{Insertion} + \text{Deletion} + \text{Substitution}}{\text{Total number of characters}} * 100 \quad (1)$$

6.2 Datasets

We use four datasets to implement and evaluate our proposed solution; (i) The Ottoman Printed Text Image (OPTI) dataset, which contains synthesized Ottoman text line images, (ii) The Arabic Printed Text Image (APTI), which contains synthesized Arabic word images and (iii) The Printed-KHATT (P-KHATT) which contains images of printed and scanned Arabic text lines and (iv) a collection of line images extracted from real printed books.

6.2.1 Ottoman printed text image (OPTI)

The dataset OPTI is a collection of 14, 800 rendered line images containing Ottoman words. It contains images of text lines rendered with Naskh font with 24 pt. font size. It is synthetically generated from a corpus of Ottoman text. The data synthesis process is explained in detail in Sect. 5.2. The ground truth of each line is created as Unicode values of the characters of that text line. Different positional forms of a letter are represented with the same Unicode value. Table 1 presents some statistics regarding the OPTI.

6.2.2 Arabic printed text image (APTI)

APTI (Arabic Printed Text Image) [60] is a multi-font, multi-size and multi-style Arabic text images dataset. It is synthetically generated using a lexicon of 113,284 words, 10 Arabic fonts, 10 font sizes and 4 font styles. The database contains 45,313,600 single-word images. The annotation of ground truths includes the number of characters, the number of sub-words, the sequence of characters, the size, the style and the font used to generate each image. The dataset is split into six sets for each font, size, and style combination. Five of the six sets are publicly available, whereas the sixth set is employed in competitions to evaluate submitted OCR systems. Each set contains different word images but the character distributions are almost the same for each set. We use 24-pt, plain, DecoType Naskh samples from the APTI dataset in this work.

6.2.3 P-KHATT

The Printed-KHATT database contains text line images printed with eight different fonts. It has a standard division splitting the data into three non-overlapping sets as train, validation and test sets. Images are captured by scanning printed documents. There are 9310 lines and 106,945 words in the dataset. Approximately 70% of the samples are in the training set. Test and validation sets each have approximately 15% of the samples. We use the 24 pt Naskh samples printed with KFGQPC Uthman Taha Naskh font in the P-KHATT.

6.2.4 Ottoman real line images

It is a small dataset of 1200 line images obtained from a printed Ottoman book[1] published in 1922. The book pages are segmented into lines semi-automatically. The images are in RGB format. The ground truth of each line is created manually as Unicode values of the characters of that line. There are some broken characters due to imperfections in printing and other noise, but most of the lines are relatively clean. Figure 7 shows some images from the real image dataset.

6.3 Experiments

We run four sets of experiments, each with a different goal. We first validate the network architecture using the APTI, OPTI and PKHATT datasets. Then we show to what extent the augmentations improve the recognition performance. After that, we use TL to fine-tune trained models for recognizing real data. Finally, we study the impact of the language of the training dataset on recognition performance. In all experiments, grayscale images are first height normalized to 64 pixels by keeping the aspect ratio and then padded with the background color to have a width matching the length of the longest sample of the dataset it belongs.

[1] The book is publicly accessible at https://archive.org/details/dagackankurt0001ad.

Table 3 Results from the baseline systems

Train	Test	CER
APTI	APTI	0.009
P-KHATT	P-KHATT	0.32
OPTI	OPTI	0.23
P-KHATT_split	P-KHATT_split	0.18
P-KHATT_split	P-KHATT	0.51
P-KHATT	P-KHATT_split	0.23
P-KHATT-syth_split	P-KHATT-syth	0.24
OPTI_split	OPTI_split	0.13
OPTI_split	OPTI	0.20
OPTI	OPTI_split	0.17

6.3.1 Evaluation of the network

We experiment with the APTI, OPTI and P-KHATT datasets to validate the network model. For this purpose, we first train a system with the APTI dataset using 72.5K samples from sets 1–4 of the 24 pt. plain Naskh division of APTI. We use 10% of the data for validation and the rest for training. That system is tested on set 5 of the same division yielding a CER of 0.009, as shown in Table 3.

Secondly, we train a baseline system using the P-KHATT dataset with the standard dataset split; 6472 samples are used for training samples, and 1414 samples are used for validation. The result of this experiment is given in Table 3. The P-KHATT baseline system has a CER of 0.32 on 1424 test samples.

The OPTI dataset has 14,800 samples. After splitting the dataset into a training set of 11, 300 samples, a test set of 2000 samples and a validation set of 1500 samples randomly, we train the OPTI baseline system and measure its performance as 0.23 CER.

We see that the model makes more recognition errors as the sequences get longer after examining the recognition output of the baseline P-KHATT and OPTI systems. Actually, there are some very long samples with transcriptions containing more than 100 characters, while nearly half of the transcriptions have 70 or more characters. So, we automatically split P-KHATT samples wider than a threshold value into halves.

Using a vertical projection histogram, the splitting algorithm first finds spaces between words larger than 5 pixels. The space nearest to the middle of the image is declared to be the splitting region. The image is divided into two at that region. It is assumed that each half contains the same number of words after the division. Hence, the transcription of the original line is divided into two based on that assumption. The error of that procedure is measured as less than 5% on the P-KHATT dataset. After splitting the lines of the training and validation sets, the sizes of the sets become 11,414 and 2463, respectively.

We train a model with the split lines and test it on both the original P-KHATT test samples and the split test samples. It obtains a CER of 0.18 on the split samples and a CER of 0.51 on the original images. Also, we measure the performance of the previous model, which is trained with the original P-KHATT lines on the split test set as 0.23 CER, as can be seen in Table 3.

The recognition problem of long sequences is observed for the OPTI dataset samples as well. We follow a more direct procedure on OPTI samples and split line transcriptions longer than a certain threshold in each subset of the OPTI then re-render the split lines. We split each of the train, test and validation samples separately. In the end, we obtain a split version of the OPTI with 16,944 training samples, 2240 validation samples and 3012 test samples. Results from the system trained with the split and original versions of the OPTI are presented in Table 3. The system trained with the original version obtains 0.23 CER on the original test samples and 0.17 CER on the split ones. The model which is trained with the split OPTI performs better; it has 0.13 CER on split test set and 0.20 CER on the original one.

The P-KHATT dataset consists of real images obtained by scanning printed text lines. On the other hand, the APTI and OPTI datasets both contain synthetic images. To have a more solid ground of comparison between these three datasets, we create a synthetic version of the P-KHATT dataset by rendering the *split* line transcriptions following the same procedure of the OPTI dataset generation. Then, we train another network with the P-KHATT synthetic data and obtain 0.24 CER on the P-KHATT synthetic test samples. Results of all experiments are given in Table 3 (Fig. 8).

6.3.2 Evaluation of the data augmentation procedure

We conduct a series of experiments to measure the effect of data augmentation on the recognition performances of the baseline systems. We augment all datasets by applying the image processing techniques explained in Sect. 5.3. Data is augmented offline such that each time the augmentation procedure is applied on the original data, an augmented version is generated and saved per sample. Original data is not included in the corresponding augmented dataset. Hence for a dataset of size S, $S \times n$ samples are generated after augmenting it n times. We observe that augmenting data more than 3–4 times does not improve recognition performance significantly. Hence, we set $n = 3$. Some examples of augmented images can be seen in Fig. 9.

The augmentation procedure is applied for the APTI, and *split versions* of the OPTI and P-KHATT datasets, since the latter two yield better recognition results than their original versions. Each model trained with an augmented dataset is

evaluated on an unaugmented test set of its corresponding dataset. Results of these experiments are presented in Table 4.

6.3.3 Evaluation of the proposed system on the real data

After we validate the network architecture and the augmentation procedure in the previous sets of experiments, we test models trained with the original and augmented versions of the datasets on the *real data* (see Sect. 6.2). The real data is composed of 1200 line images from a printed book. We randomly split the dataset into *non-overlapping* train, validation and test sets with 800, 200 and 200 images, respectively.

In the testing phase, we apply the line splitting technique on the real images by splitting a test line into two using the splitting method described in Sect. 6.3.1. The system recognizes each half separately and then merges outputs to obtain that line's recognition output. The baseline OPTI, P-KHATT and synthetic P-KHATT systems obtain 0.60, 0.55 and 0.65 CER values, respectively, on the real image data. Systems trained with augmented datasets perform significantly better compared to the baseline systems. The system trained with the augmented OPTI data has a 0.35 CER which means 36% reduction in CER. Augmented versions of the P-KHATT and the synthetic P-KHATT get 0.50 CER and 0.55 CER, respectively.

Finally, we apply transfer learning (TL) by the fashion of a fine-tuning approach. TL is a technique for exploiting information obtained during a learning process to improve another learner from a different but related domain. TL is preferred, especially in cases where training data is limited. TL has been in use for various problems like text sentiment classification, image classification, and text recognition [40, 67]. Once a network is trained with data from a related domain, TL can be applied by freezing some of the layers where knowledge is accumulated as weights. Freezing prevents the update of the weights. Other layers which will continue learning can be reset or keep learned weights before the network is trained with new data from the target domain. Another method is to fine-tune the system with the new data. Here, the weights of all layers are updated according to the information learned from the new data. When the domains are similar, fine-tuning becomes a preferable method.

In our case, we empirically found out that the fine-tuning method improves performance more than freezing some of the CNN and BLSTM layers which is justifiable since the source and target domains are quite similar. Additionally, we use the augmentation scheme defined previously on the fine-tuning data to measure its effect on the real data. Table 5 summarizes the results from experiments with the real data. The systems trained with augmented versions of the OPTI, P-KHATT and synthetic P-KHATT get 0.24, 0.44 and 0.45 CERs, respectively. Using an augmented version of the real data for model fine-tuning brings at least a 30% decrease in CERs for all models. The system trained with the augmented P-KHATT data benefits from fine-tuning with augmented real data as the CER decreases by almost 50%.

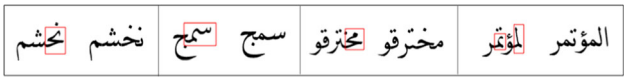

Fig. 8 Examples of ligatures existing in the APTI dataset but are absent from the OPTI dataset. For each pair, the right word is an OPTI word image, while the left is the same word written with a ligature in the red box. All samples are generated with DecoType Naskh®TrueType Font

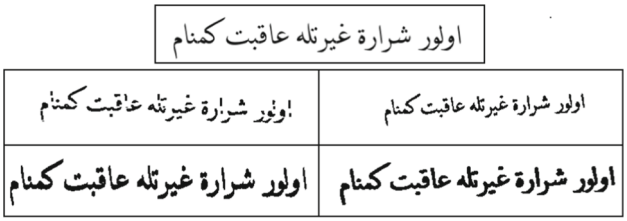

Fig. 9 An OPTI sample (the top box) generated with DecoType Naskh®TrueType Font and several of its augmented versions

Table 4 Results from the data augmentation experiments

Train	Test	CER
Aug.-APTI	APTI	0.02
Aug.-P-KHATT_split	P-KHATT_split	0.07
Aug.-P-KHATT-syth	P-KHATT-syth	0.08
Aug.-OPTI_split	OPTI_split	0.11

6.3.4 Effect of the language of the training data on the recognition accuracy

Finally, we explore the impact of language on recognition accuracy when the training and testing data contain texts from different languages. Here, we use only the models trained with synthetic data for a more direct comparison.

It has been shown that sequence learners like RNNs and their variants capture character dependencies and incorporate a character-level language model to the recognition process [35, 55, 64]. Based on this, we examine the usability of datasets of different languages for recognition of one another by testing a recognizer on datasets other than the one it is trained with.

Due to the different versions of Decotype Naskh font employed in the rendering process of the APTI and OPTI, some of the shapes and ligatures used in the APTI are absent from the OPTI.

Figure 8 shows examples of missing shapes and their counterparts in the OPTI. For a fair comparison, we need to eliminate APTI samples containing such absent forms when testing the OPTI-trained systems. However, the detection of such samples is a manual and time-consuming task. So, we

Table 5 Results from the fine tuning experiments

Train	Fine-tune	Test	CER
OPTI	–	Real data	0.60
PKHATT	–	Real data	0.55
PKHATT_syn	–	Real data	0.65
Aug.-OPTI	–	Real data	**0.38**
Aug.-PKHATT	–	Real data	0.50
Aug.-PKHATT_syn	–	Real data	0.55
Aug.-OPTI	Real data	Real data	0.24
Aug.-OPTI	Aug.-real data	Real data	**0.16**
Aug.-PKHATT_syn	Real data	Real data	0.44
Aug.-PKHATT_syn	Aug.-real data	Real data	0.32
Aug.-PKHATT	Real data	Real data	0.45
Aug.-PKHATT	Aug.-real data	Real data	0.23

Results showing the effect of fine-tuning on recognition accuracy. The best results obtained with and without fine-tuning are given in bold

Table 6 Results from the cross-language experiments

Train	Test	CER
OPTI	APTI	0.29
OPTI	APTI-reduced	0.23
Aug.-OPTI	APTI	**0.24**
Aug.-OPTI	APTI-reduced	**0.18**
OPTI	P-KHATT-syth	0.26
Aug.-OPTI	P-KHATT-syth	**0.23**
P-KHATT-syth	OPTI	0.45
Aug.-P-KHATT-syth	OPTI	0.30

Performance of the system trained with augmented OPTI on APTI, reduced APTI and P-KHATT-syth datsets are emphasized in bold

confine ourselves to a reduced APTI version where samples containing only four absent forms are removed. These forms, which can be detected automatically, are listed as stretched (i.e., kasheeda) versions of ن and ى and two diacritics; madda over characters other than initial ا and shadda. Testing with the reduced version, which contains 12K samples, aims to show the effect of the absence of ligatures and shapes on recognition accuracy. We test the OPTI-trained models on two versions of the set5 of the APTI dataset; the original version of the set5 of APTI and the reduced version of APTI. Testing an APTI-trained model on the OPTI dataset is not practical since the network trained with single words of the APTI, which are relatively shorter sequences, is incapable of recognizing longer sequences of the OPTI lines.

Results of these experiments are presented in Table 6. According to the results, the system trained with the OPTI obtains a CER of 0.29 on the original APTI test set and 0.23 CER on the reduced version. The augmented OPTI-trained model performs better with a CER of 0.24 on the original APTI test set and 0.18 CER on the reduced version.

When the OPTI and augmented OPTI-trained models are evaluated on the synthetic P-KHATT, their CER values are measured as 0.26 and 0.23, respectively. On the other hand, systems trained with the synthetic P-KHATT and its augmented version obtain 0.45 CER and 0.30 CER, respectively.

7 Discussion

We can make several observations using the results from the four sets of experiments. Firstly, the proposed solution provides a viable solution to the printed Arabic alphabet-based scripts written with the Naskh font, as can be seen from Tables 3, 4 and 5. The line-splitting strategy is quite important within the proposed scheme because it provides a solution to the problem of remembering longer sequences. It is a known fact that RNN and its variants have a limit to the number of time steps from which they can learn dependencies. Here, the splitting strategy helps keep the sample sizes by these limits. Both the P-KHATT and OPTI systems benefit from line splitting; CER of the baseline OPTI system decreases from 0.23 to 0.13, and CER of the baseline P-KHATT system decreases from 0.32 to 0.18. Another observation is that the splitting strategy is useful even if it is used only in evaluation. For example, CER of the P-KHATT baseline system, which is trained with the original lines, decreases by 28% if the test samples are split. In the case of the OPTI baseline system, a 26% CER decrease is observed.

For all of the datasets except the APTI dataset, the augmentation procedure brings a significant improvement over the baseline systems. The baseline APTI system already achieves a CER of less than 1%. Augmenting the data leads to noise for the APTI dataset. The P-KHATT system benefits from the augmentation most, with a decrease of CER from 0.18 to 0.07. Similarly, the synthetic P-KHATT system also significantly improves performance with a decrease of CER from 0.24 to 0.08. However, the CER of the OPTI system decreases by only 2 points. An explanation for this difference may be found in the numbers of unique PAWs the OPTI and P-KHATT datasets have. Although the numbers of words and characters of these two datasets are similar, the number of unique PAWS in the P-KHATT is almost double the number of unique PAWS in the OPTI (see Table 1). That can be interpreted as the number of unique shapes in the P-KHATT are higher than that of the OPTI. Augmenting the OPTI data brings less variability than augmenting P-KHATT data since the number of unique shapes represented in the original training set is comparatively smaller.

The systems trained with augmented datasets are more successful in recognizing the real data. The augmented OPTI system has a better performance than the P-KHATT and the

synthetic P-KHATT systems, as seen in Table 5. Interestingly, the system trained with the original P-KHATT dataset performs better with a CER of 0.55 compared to the synthetic P-KHATT and OPTI systems. This can be attributed to the fact that both the P-KHATT and the real data are *real images* captured from some printed documents and not rendered images. Similarly, the observation that the augmented version of P-KHATT performs better than the augmented synthetic P-KHATT strengthens that claim.

Fine-tuning with real data brings considerable improvement for all of the systems. Using an augmented version of the real data for model fine-tuning brings at least a 30% decrease in CERs for all models. The system trained with the augmented P-KHATT data benefits from fine-tuning with augmented real data as the CER decreases by almost 50%. This is an expected outcome; since the amount of real data is really small and augmentations increase the dataset size artificially. The best performance is obtained as 0.16 CER by the system trained with the augmented OPTI dataset and fine-tuned with augmented real data.

Finally, in Table 6, we observe very high CER values when any of the systems trained on one dataset is evaluated on another dataset, even though they are generated with a quite similar procedure. All of the OPTI-trained systems have a CER more than 20% on both APTI and reduced APTI test sets, except the augmented OPTI system, which obtains 18% CER on the reduced set. Part of the problem is due to the missing ligature forms, as indicated by the decrease in CER when the reduced APTI samples are used. Removal of the samples containing ligatures absent in the OPTI dataset from the test set results in an average decrease of 0.6 points for all systems. It should be noted that in the reduced APTI test samples, only four of the absent forms are removed. The other reason for the low recognition accuracy on APTI test samples in Table 6 may be some contextual character shapes not occurring frequently enough in the OPTI. There are over 30 K unique PAWs in the APTI collection, while OPTI datasets have only 10.9 K PAWs. The occurrence of a certain PAW is directly related to word formation characteristics which are language-specific. If a character combination is rare or invalid in a language, then the contextual shapes used in the combination may be underrepresented or absent from dataset of that language. Considering the huge difference between the number of unique PAWs, it is possible that some APTI contextual character shapes rarely occur in the OPTI dataset. That possibility requires further investigation, which is out of the scope of this work.

The augmented OPTI system obtains 0.23 CER on the synthetic P-KHATT system, while the augmented synthetic P-KHATT system has a CER of 0.30 on the OPTI dataset in Table 6. Considering the performance of these two systems on their datasets in Table 4, we can say that language of the training data has a direct effect on recognition accuracy since the main difference is the language of the rendered texts. It is possible to fine-tune or re-train a system using new data containing texts of another language. However, the results of the P-KHATT and synthetic P-KHATT experiments with real Ottoman data in Table 5 show that the performance will be quite sub-optimal.

8 Conclusion

In this work, we investigated the challenges in recognition of the Ottoman script and proposed a solution using synthetic data and a data augmentation method. Specifically, (i) we prepared a corpus from digital texts and generated a synthetic machine-printed Ottoman line image dataset using the corpus, (ii) we augmented the synthetic data using an image processing pipeline which is designed for Ottoman documents, (iii) we used an effective line image splitting method for increasing recognition accuracy, (iv) we developed and evaluated a recognition system for machine-printed Ottoman script with Naskh font and (v) we reported on the usability of two Arabic datasets for recognizing Ottoman documents.

Our results on two standard datasets (i.e., APTI and P-KHATT) and real Ottoman documents are on par with the results from the literature. We plan to work on larger real image datasets to design a multi-font recognition system in the future.

Author Contributions All authors contributed equally.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. AbdelRaouf, A., Higgins, C.A., Pridmore, T.P., Khalil, M.I.: Building a multi-modal Arabic corpus (MMAC). Int. J. Doc. Anal. Recogn. **13**(4), 285–302 (2010)
2. Ahmad, I., Mahmoud, S.A., Fink, G.A.: Open-vocabulary recognition of machine-printed Arabic text using hidden Markov models. Pattern Recogn. **51**, 97–111 (2016)
3. Ahmad, R., Naz, S., Afzal, M.Z., Rashid, S.F., Liwicki, M.: A deep learning based Arabic script recognition system: benchmark on KHAT. Int. Arab J. Inf. Technol. **17**(3), 299–305 (2020)
4. Al-Badr, B., Mahmoud, S.A.: Survey and bibliography of Arabic optical text recognition. Signal Process. **41**(1), 49–77 (1995)
5. Al-Helali, B.M., Mahmoud, S.A.: Arabic online handwriting recognition (AOHR): a survey. ACM Comput. Surv. **50**(3), 33:1-33:35 (2017)
6. Al-Muhtaseb, H.A., Mahmoud, S.A., Qahwaji, R.: Recognition of off-line printed Arabic text using hidden Markov models. Signal Process. **88**(12), 2902–2912 (2008)
7. Alrobah, N.A., Albahli, S.: Arabic handwritten recognition using deep learning: a survey. Arab. J. Sci. Eng. **47**, 9943–9963 (2022)
8. Al-Salman, A., Alyahya, H.: Arabic online handwriting recognition: a survey. In: Hamdan, H., Boubiche, D.E., Klett, F. (eds.)

Proceedings of the 1st International Conference on Internet of Things and Machine Learning, IML 2017, Liverpool, United Kingdom, October 17–18, 2017, pp. 51:1–51:4. ACM (2017)
9. Ataer, E., Duygulu, P.: Matching ottoman words: an image retrieval approach to historical document indexing. In: Proceedings of the 6th ACM International Conference on Image and Video Retrieval, pp. 341–347 (2007)
10. Ataer, E., Duygulu, P.: Retrieval of ottoman documents. In: Wang, J.Z., Boujemaa, N., Chen, Y. (eds.) Proceedings of the 8th ACM SIGMM International Workshop on Multimedia Information Retrieval, MIR 2006, October 26–27, 2006, Santa Barbara, CA, USA, pp. 155–162. ACM (2006)
11. Aydemir, M.S., Aydin, B., Kaya, H., Karliaga, I., Demir, C.: Tübıtak Turkish–Ottoman handwritten recognition system. In: 2014 22nd Signal Processing and Communications Applications Conference (SIU), Trabzon, Turkey, April 23–25, 2014, pp. 1918–1921. IEEE (2014)
12. Can, E.F., Duygulu, P., Can, F., Kalpakli, M.: Redif extraction in handwritten ottoman literary texts. In: 2010 20th International Conference on Pattern Recognition, pp. 1941–1944 (2010)
13. Can, Y.S., Kabadayı, M.E.: Computerized counting of individuals in ottoman population registers with deep learning. In: Bai, X., Karatzas, D., Lopresti, D. (eds.) Document Analysis Systems, pp. 277–290. Springer, Cham (2020)
14. Capobianco, S., Marinai, S.: Docemul: A toolkit to generate structured historical documents. In: 14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, November 9–15, 2017, pp. 1186–1191. IEEE (2017)
15. Dolek, I., Kurt, A.: A deep learning model for ottoman OCR. Concurr. Comput.: Pract. Exp. **34**(20), e6937 (2022)
16. Dutta, K., Krishnan, P., Mathew, M., Jawahar, C.V.: Improving CNN-RNN hybrid networks for handwriting recognition. In: 16th International Conference on Frontiers in Handwriting Recognition, ICFHR 2018, Niagara Falls, NY, USA, August 5–8, 2018, pp. 80–85. IEEE Computer Society (2018)
17. Duygulu, P., Arifoglu, D., Kalpakli, M.: Cross-document word matching for segmentation and retrieval of ottoman divans. Pattern Anal. Appl. **19**(3), 647–663 (2016)
18. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 9, pp. 249–256. JMLR Workshop and Conference Proceedings (2010)
19. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 369–376 (2006)
20. Graves, A., Schmidhuber, J.: Offline handwriting recognition with multidimensional recurrent neural networks. In: Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, pp. 545–552 (2008)
21. Graves, A.: Supervised Sequence Labelling with Recurrent Neural Networks, Studies in Computational Intelligence, vol. 385. Springer (2012)
22. Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., Schmidhuber, J.: A novel connectionist system for unconstrained handwriting recognition. IEEE Trans. Pattern Anal. Mach. Intell. **31**(5), 855–868 (2009)
23. Hakro, D.N., Talib, A.Z.: Printed text image database for Sindhi OCR. ACM Trans. Asian Low Resour. Lang. Inf. Process. **15**(4), 21:1-21:18 (2016)
24. Hamdi, Y., Boubaker, H., Dhieb, T., Elbaati, A., Alimi, A.M.: Hybrid DBLSTM-SVM based beta-elliptic-CNN models for online Arabic characters recognition. In: 2019 International Conference on Document Analysis and Recognition, pp. 545–550 (2019)
25. Hosseini, F.s., Kashef, S., Shabaninia, E., Nezamabadi-pour, H.: Idpl-pfod: an image dataset of printed Farsi text for OCR research. In: Proceedings of The Second International Workshop on NLP Solutions for Under Resourced Languages (NSURL 2021) co-located with ICNLSP 2021, pp. 22–31. Association for Computational Linguistics, Trento, Italy (2021)
26. Jaiem, F.K., Kanoun, S., Khemakhem, M., Abed, H.E., Kardoun, J.: Database for Arabic printed text recognition research. In: Petrosino, A. (ed.) Image Analysis and Processing—ICIAP 2013—17th International Conference, Naples, Italy, September 9–13, 2013. Proceedings, Part I. Lecture Notes in Computer Science, vol. 8156, pp. 251–259. Springer (2013)
27. Jiang, Z., Ding, X., Peng, L., Liu, C.: Modified bootstrap approach with state number optimization for hidden Markov model estimation in small-size printed Arabic text line recognition. In: Perner, P. (ed.) Machine Learning and Data Mining in Pattern Recognition—10th International Conference, MLDM 2014, St. Petersburg, Russia, July 21–24, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8556, pp. 437–441. Springer (2014)
28. Journet, N., Visani, M., Mansencal, B., Kieu, V.C., Billy, A.: Doccreator: a new software for creating synthetic ground-truthed document images. J. Imaging **3**(4), 62 (2017)
29. Khorsheed, M.S.: Offline recognition of omnifont Arabic text using the HMM toolkit (HTK). Pattern Recogn. Lett. **28**(12), 1563–1571 (2007)
30. Khoury, I., Giménez, A., Juan, A., Andrés-Ferrer, J.: Window repositioning for printed Arabic recognition. Pattern Recogn. Lett. **51**, 86–93 (2015)
31. Kilic, N., Gorgel, P., Ucan, O.N., Kala, A.: Multifont Ottoman character recognition using support vector machine. In: 2008 3rd International Symposium on Communications, Control and Signal Processing, pp. 328–333 (2008)
32. Kurt, Z., Turkmen, H., Karsligil, E.: Linear discriminant analysis in Ottoman alphabet character recognition. Appl. Therm. Eng. **28**, 601–607 (2009)
33. Li, Z., Liu, F., Yang, W., Peng, S., Zhou, J.: A survey of convolutional neural networks: analysis, applications, and prospects. IEEE Trans. Neural Networks Learn. Syst. **33**(12), 6999–7019 (2022)
34. Märgner, V., Pechwitz, M.: Synthetic data for Arabic OCR system development. In: 6th International Conference on Document Analysis and Recognition, pp. 1159–1163. IEEE Computer Society (2001)
35. Martínek, J., Lenc, L., Král, P.: Building an efficient OCR system for historical documents with little training data. Neural Comput. Appl. **32**(23), 17209–17227 (2020)
36. Mori, S., Suen, C.Y., Yamamoto, K.: Historical review of OCR research and development. Proc. IEEE **80**(7), 1029–1058 (1992)
37. Namysl, M., Konya, I.: Efficient, lexicon-free OCR using deep learning. In: 2019 International Conference on Document Analysis and Recognition, ICDAR, pp. 295–301. IEEE (2019)
38. Natarajan, P., Lu, Z., Schwartz, R.M., Bazzi, I., Makhoul, J.: Multilingual machine printed OCR. Int. J. Pattern Recogn. Artif. Intell. **15**(1), 43–63 (2001)
39. Naz, S., Umar, A.I., Ahmad, R., Siddiqi, I., Ahmed, S.B., Razzak, M.I., Shafait, F.: Urdu Nastaliq recognition using convolutional-recursive deep learning. Neurocomputing **243**, 80–87 (2017)
40. Niu, S., Liu, Y., Wang, J., Song, H.: A decade survey of transfer learning (2010–2020). IEEE Trans. Artif. Intell. **1**(2), 151–166 (2020)
41. Özege, M.S.: Eski Harflerle Basılmış Türkçe Eserler Kataloğu. Fatih Yayınevi Matbaası, İstanbul (1982)
42. Ozturk, A., Gunes, S., Ozbay, Y.: Multifont ottoman character recognition. In: ICECS 2000. 7th IEEE International Conference on Electronics, Circuits and Systems, vol. 2, pp. 945–949 (2000)

43. Parvez, M.T., Mahmoud, S.A.: Offline Arabic handwritten text recognition: a survey. ACM Comput. Surv. **45**(2), 23:1-23:35 (2013)
44. Pondenkandath, V., Alberti, M., Diatta, M., Ingold, R., Liwicki, M.: Historical document synthesis with generative adversarial networks. In: 2nd International Workshop on Machine Learning, WML@ICDAR 2019, Sydney, Australia, September 22–25, 2019, pp. 146–151. IEEE (2019)
45. PourReza, M., Derakhshan, R., Fayyazi, H., Sabokrou, M.: Subword based Persian OCR using auto-encoder features and cascade classifier. In: 9th International Symposium on Telecommunications, IST 2018, Tehran, Iran, December 17–19, 2018, pp. 481–485. IEEE (2018)
46. Prasad, R., Saleem, S., Kamali, M., Meermeier, R., Natarajan, P.: Improvements in hidden Markov model based Arabic OCR. In: 19th International Conference on Pattern Recognition (ICPR 2008), December 8–11, 2008, Tampa, Florida, USA, pp. 1–4. IEEE Computer Society (2008)
47. Puigcerver, J.: Are multidimensional recurrent layers really necessary for handwritten text recognition? In: 14th IAPR International Conference on Document Analysis and Recognition, pp. 67–72. IEEE (2017)
48. Qaroush, A., Awad, A., Modallal, M., Ziq, M.: Segmentation-based, omnifont printed Arabic character recognition without font identification. J. King Saud Univ. Comput. Inf. Sci. **34**(6 Part A), 3025–3039 (2022)
49. Radwan, M.A., Khalil, M.I., Abbas, H.M.: Neural networks pipeline for offline machine printed Arabic OCR. Neural Process. Lett. **48**(2), 769–787 (2018)
50. Rahal, N., Tounsi, M., Hussain, A., Alimi, A.M.: Deep sparse auto-encoder features learning for Arabic text recognition. IEEE Access **9**, 18569–18584 (2021)
51. Rahmati, M., Fateh, M., Rezvani, M., Tajary, A., Abolghasemi, V.: Printed Persian OCR system using deep learning. IET Image Process. **14**(15), 3920–3931 (2020). https://doi.org/10.1049/iet-ipr.2019.0728
52. Rashid, S.F., Schambach, M., Rottland, J., von der Nüll, S.: Low resolution Arabic recognition with multidimensional recurrent neural networks. In: Govindaraju, V., Natarajan, P., Chaudhury, S., Lopresti, D.P., Setlur, S., Cao, H. (eds.) Proceedings of the 4th International Workshop on Multilingual OCR, MOCR@ICDAR 2013, Washington, DC, USA, August 24, 2013, pp. 6:1–6:5. ACM (2013)
53. Sabbour, N., Shafait, F.: A segmentation-free approach to arabic and urdu OCR. In: Document Recognition and Retrieval XX, part of the IS&T-SPIE Electronic Imaging Symposium. SPIE Proceedings, vol. 8658, p. 86580N. SPIE (2013)
54. Sabir, E., Rawls, S., Natarajan, P.: Implicit language model in LSTM for OCR. In: 6th International Workshop on Multilingual OCR, 14th IAPR International Conference on Document Analysis and Recognition, MOCR@ICDAR 2017, Kyoto, Japan, November 9–15, 2017, pp. 27–31. IEEE (2017)
55. Sabir, E., Rawls, S., Natarajan, P.: Implicit language model in LSTM for OCR. In: 6th International Workshop on Multilingual OCR, 14th IAPR International Conference on Document Analysis and Recognition, MOCR@ICDAR 2017, Kyoto, Japan, November 9–15, 2017, pp. 27–31. IEEE (2017)
56. Saykol, E., Sinop, A.K., Gudukbay, U., Ulusoy, O., Cetin, A.E.: Content-based retrieval of historical ottoman documents stored as textual images. IEEE Trans. Image Process. **13**(3), 314–325 (2004)
57. Qaroush, A., Awad, A., Modallal, M., Ziq, M.: Segmentation-based, omnifont printed Arabic character recognition without font identification. J. King Saud Univer.—Comput. Inf. Sci. **34**(6, Part A), 3025–3039 (2022)
58. Shewalkar, A., Nyavanandi, D., Ludwig, S.A.: Performance evaluation of deep neural networks applied to speech recognition: RNN, LSTM and GRU. J. Artif. Intell. Soft Comput. Res. **9**(4), 235–245 (2019)
59. Shi, B., Bai, X., Yao, C.: An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE Trans. Pattern Anal. Mach. Intell. **39**(11), 2298–2304 (2017)
60. Slimane, F., Ingold, R., Kanoun, S., Alimi, A.M., Hennebert, J.: A new arabic printed text image database and evaluation protocols. In: 10th International Conference on Document Analysis and Recognition, pp. 946–950. IEEE Computer Society (2009)
61. Slimane, F., Zayene, O., Kanoun, S., Alimi, A.M., Hennebert, J., Ingold, R.: New features for complex arabic fonts in cascading recognition system. In: Proceedings of the 21st International Conference on Pattern Recognition, ICPR 2012, Tsukuba, Japan, November 11–15, 2012, pp. 738–741. IEEE Computer Society (2012)
62. Slimane, F., Zayene, O., Kanoun, S., Alimi, A.M., Hennebert, J., Ingold, R.: New features for complex Arabic fonts in cascading recognition system. In: Proceedings of the 21st International Conference on Pattern Recognition, pp. 738–741. IEEE Computer Society (2012)
63. Ul-Hasan, A., Ahmed, S.B., Rashid, S.F., Shafait, F., Breuel, T.M.: Offline printed Urdu Nastaleeq script recognition with bidirectional LSTM networks. In: 12th International Conference on Document Analysis and Recognition, pp. 1061–1065. IEEE Computer Society (2013)
64. Ul-Hasan, A., Breuel, T.M.: Can we build language-independent OCR using LSTM networks? In: Govindaraju, V., Natarajan, P., Chaudhury, S., Lopresti, D.P., Setlur, S., Cao, H. (eds.) Proceedings of the 4th International Workshop on Multilingual OCR, MOCR@ICDAR 2013, Washington, DC, USA, August 24, 2013, pp. 9:1–9:5. ACM (2013)
65. Voigtlaender, P., Doetsch, P., Ney, H.: Handwriting recognition with large multidimensional long short-term memory recurrent neural networks. In: 15th International Conference on Frontiers in Handwriting Recognition, pp. 228–233. IEEE Computer Society (2016)
66. Wahab, M., Amin, H., Ahmed, F.: Shape analysis of Pashto script and creation of image database for OCR. In: 2009 International Conference on Emerging Technologies, pp. 287–290 (2009)
67. Weiss, K.R., Khoshgoftaar, T.M., Wang, D.: A survey of transfer learning. J. Big Data **3**, 9 (2016)
68. Yalniz, I.Z., Altingovde, I.S., Güdükbay, U., Ulusoy, Ö.: Integrated segmentation and recognition of connected ottoman script. Opt. Eng. **48**, 117205 (2009)
69. Zahoor, S., Naz, S., Khan, N.H., Razzak, M.I.: Deep optical character recognition: a case of Pashto language. J. Electron. Imaging **29**(02), 023002 (2020)