Sindhi Ligature Recognition in Printed Text

Zeeshan Ali, Safdar Abbas Khan

School of Electrical Engineering & Computer Science, NUST, Islamabad, Pakistan

E-mail: `zpanhwar.mscs19seecs@seecs.edu.pk, safdar.abbas@seecs.edu.pk`

Nov 2024

Abstract. The advent of Deep Learning in Computer Vision has resulted in advancements in many domains, encompassing a diverse set of fields. Object Character Recognition has played a vital role in the modern age of Artificial Intelligence. This is a challenging task, difficult to implement, and computationally expensive. Sindhi is a literature-rich language that is spoken by millions of people worldwide. Most Sindhi literature uses an extended Perso-Arabic script. To the best of our knowledge, no benchmark dataset has yet been published. Consequently, no state-of-the-art Sindhi OCR model has been developed. This study attempts to fill these research gaps by making the following contributions. It collected a set of 22,597 ligatures found in the Sindhi literature. It synthesizes a benchmark dataset for Sindhi ligature recognition in printed texts. The dataset was font-diverse and comprised 256 unique fonts. Finally, it presents a baseline model that achieves 91.85% test accuracy on the dataset. Our baseline can be used to build the complete pipeline of a font-invariant Sindhi OCR.

1. Introduction

Sindhi is an Indo-Aryan language that is rooted in the lower Indus River valley [1]. It is closer to the old Prakrit than to the Marathi, Hindi, Panjabi, and Bangali. It has an exuberance of preserved grammatical forms [2]. There are many writing systems for the Sindhi language. Perso-Arabic Script accounts for the majority of Sindhi literature [3]. The script is widely used in Pakistan. On the other hand, Devanagari Script is mainly used in India. Historically, the Khudabadi (Hatvanki) script was also used, but it became rare after 1947. The Persian alphabet is a modification of the Arabic alphabet, with four additional letters. It became the basis of the Sindhi alphabet with two digraphs and 18 new letters. The Sindhi alphabet has 52 letters, twice the number of letters in English, covering a wide variety of sounds.

Object Character Recognition (OCR) is the process of extracting text from an image. Text can be typed or written. The input can be a scanned Portable Document

Format (PDF), a camera image, or an image-only PDF, but the output must be a sequence of ASCII, or Unicode characters. An OCR is required for electronic computers to understand text. An image is a matrix of pixels with intrinsic patterns that are directly unrecognizable by a computer. Text is defined and represented based on ASCII or Unicode, which are specially designed for computers. OCR eliminates the manual labor required to obtain text from images. This enables text editing to produce an updated version of the text. Intelligent Character Recognition (ICR) uses Artificial Intelligence (AI) methods, especially Deep Learning [4]. OCR technology gained popularity in the 1990s with the digitization of historical newspapers. Currently, OCR solutions offer high accuracy and automate complex document-processing workflows. Manual retyping is the only option for digital formatting, but OCR has made it easier and more accurate. OCR services, such as Google Cloud Vision OCR, are widely accessible and enable the scanning and storage of documents on smartphones.

OCR plays a crucial role in digitizing patient records and medical documents [5]. It can convert printed or handwritten medical records, lab reports, and prescription notes into machine-readable text, making it easier to store, search for, and analyze patient information. This improves healthcare providers' efficiency in accessing and managing patient data. OCR can automate the process of reading and extracting information from prescription forms. It helps in accurately capturing medication details, dosages, and patient instructions, reducing the chances of errors and improving prescription management. OCR is widely used to digitize printed documents such as books, articles, invoices, forms, and receipts. By converting text into a machine-readable format, OCR enables efficient storage, retrieval, and search of documents. Advanced OCR techniques can recognize and convert handwritten text into digital format. This is useful in applications, such as digital note-taking, digitizing historical documents, and processing handwritten forms. Self-driving cars use OCR to extract text from road signs [6].

With the advancement in AI methods, especially Deep Learning based Computer Vision (CV), the preprocessing, segmentation, or contour extraction, along with the postprocessing stages, have all been replaced by end-to-end Deep Learning (DL) models. The OCR system involves two stages: text detection and text recognition. The first stage is done mostly at the character level, at the word level, or sometimes at the ligature level. OCR methods based on DL have recently shown state-of-the-art results; however, these results are not evenly distributed across all languages. Most research has focused on English, and a great deal of work has also been done on languages such as Chinese. Arabic script is hard to recognize as its characters are often interlinked with each other in words, unlike Latin based languages such as English, in which each letter is separated in most of its font styles in printed documents. The Online tool Tesseract OCR [7] supports Arabic script; however, it performs poorly on it.

Sindhi uses an extended Perso-Arabic script with an alphabet of size 52, whereas the Arabic alphabet contains 28 letters. The presence of ligatures in the Sindhi script makes it challenging for an OCR to correctly recognize the text. A ligature is a part of a word that has all its letters interlinked. Therefore, an OCR for Arabic or Perso-Arabic scripts, especially the extended Perso-Arabic script of Sindhi, must have text recognition at the ligature level rather than at the character or word level. The Sindhi script has high inter-character similarities, as many characters differ only by I'jam, the number, or the position of the dots. It has high glyph variability, that is, intra-character dissimilarity, as many characters change their form based on their position (start, middle, or end) in a ligature. It also faces the problem of inter-font variability, as different fonts also create intra-character dissimilarity in terms of the alignment of dots and the structure of Rasm strokes. The Arabic script and its extended scripts heavily involve Harakat, which makes the recognition of characters, ligatures, or words difficult. Therefore, designing an OCR system for Sindhi is even more challenging and can have a direct impact on its parent scripts, Persian and Arabic, as well as on its sister scripts, such as Urdu, Pashto, Siraiki, and Kashmiri.

The ability to recognize objects is crucial for human survival; therefore, human vision is remarkably good. Similarly, Object Recognition has been one of the most fundamental tasks in Computer Vision, or more generally, in Machine Vision. With the introduction of ImageNet [8, 9] as a benchmark dataset for Object Categorization in images, more commonly known as Image Classification [10] and Object Localization [11], there has been a significant advancement in the tasks. Leaving behind the human level perception, it has achieved state-of-the-art (SOTA) results in most classical and modern Computer Vision tasks such as Object Detection [12, 13], Semantic- [14, 15], Instance- [16, 17], or collectively, Panoptic-Segmentation [18], Visual Object Tracking [19], Image Captioning [20, 21], and Video Categorization [22]. Object Character Recognition (OCR) has significantly evolved in recent years. Several benchmark datasets have been developed for this purpose. DocBank [23] includes 500k document pages, and FUNSD [24] is a collection of 199 real scanned forms, TextOCR [25], 24902 train 3232 test images, and IAM Handwriting [26] is created by 657 writers, and has 13,353 images of handwritten lines of text, all covering a wide variety of scenarios. Several state-of-the-art models were designed for each benchmark. However, these advancements have mostly been made in English and other latin-based languages. Many benchmark datasets have also been designed for Chinese OCR recently, such as Chinese Text in the Wild [27], ChineseLP [28], and MSDA [29]. OCR systems for Arabic have also advanced [30, 31]. Tesseract OCR [7] supports Arabic but performs poorly on it. Research on Urdu OCR has produced good benchmark datasets. Word-level text categorization from images has been proposed [32]. Qaida [33] is a large scale font independent Urdu Text Recognition system based on ligature level text categorization in images. Urdu Nastaliq Handwritten Dataset (UNHD) [34] is an end-to-end Urdu OCR benchmark dataset that is used for handwriting recognition. It contains the hand writings of 500 writers, focusing on the most occurring ligatures in Urdu language text. Little to no effort has been made to develop a Sindhi OCR benchmark dataset. No state-of-the-art Sindhi OCR model has been developed. Therefore, there is great room for the development of OCR models for Document Understanding, Handwriting Recognition, and Sindhi texts in the wild. Our research focused on printed text in documents.

An OCR system involves two broad stages: text detection and text recognition. Text detection is mostly performed using Object Detection methods, which can be extended to recognize detected text at the character or word level. This research mainly attempts to solve the Text Recognition task at the ligature level for the Sindhi OCR. It is difficult to detect characters separately from the text in the extended Perso-Arabic script of the Sindhi writing system because its characters are often interlinked with each other in ligatures, unlike latin-based languages such as English. The abundance of unique words in a language, especially Sindhi, makes the number of categories too large and makes text categorization difficult at the word level. Therefore, the only viable option is Ligature Recognition (LR). However, no benchmark dataset exists for Sindhi Ligature Recognition. Therefore, this study attempts to synthetically develop a Sindhi Ligature Recognition dataset.

The success of OCR systems in English has inspired researchers in the past decade to develop OCR models in other languages, such as Chinese. Arabic language has not been taken into consideration as much. The Perso-Arabic languages face the same dilemma, and the Sindhi language falls into the same category. Efforts have been made to develop the Sindhi OCR [35, 36]. Sindhi Language Authority [37] developed the online Sindhi OCR [38]. It is a beta version based on a Deep Learning model claimed to have 90-95% accuracy. It was designed to work on images with a suggested minimum of 300 Dots Per Inch (DPI) for the best results. It only supports three fonts: MB Lateefi, Awami, and Adabi, and more fonts are set to be supported in a newer version. AMBILE Hamiz Ali Sindhi OCR [39] has been developed in 2020 by AMBILE [40]. It supports 62 Sindhi fonts and a minimum of 300 DPI for the best results. It is also strictly dependent on the horizontal alignment of text in the input images, and the model itself is not rotation invariant. More than 280 fonts have been developed, including formal and stylish font styles, which need to be covered for a more generalized and sophisticated font-independence OCR. The lack of availability of such public benchmarks has been a bottleneck to advancements in the Sindhi OCR. Hence, there is no state-of-the-art Sindhi OCR model available on the Internet.

This research is an extension to [41] and attempts to tackle the two bottlenecks in the research on developing OCR for the Sindhi language, as mentioned in the previous section. It introduced a synthetically generated large-scale font diverse Sindhi Ligature Recognition dataset (SLRSet). It also provides a baseline convolutional neural network model for Sindhi Ligature Recognition (SLRNet). The model was trained on SLRSet, achieving a test accuracy of 91.85%.

A complete pipeline was designed to synthesize the ligature-recognition dataset. The pipeline is language independent; hence, it can be used to develop similar datasets for any script, especially the Perso-Arabic ones. There have been no attempts to collect ligatures that encompass Sindhi literature. Here, we present a large collection of 22,597 unique ligatures. This set of ligatures is utilized in 256 different fonts to generate ligature-level gray images of size 80×80. The resultant dataset comprised a collection of 5,784,832 images labeled with 22,597 classes. The dataset is font-diverse and can leverage models to attain font invariability for the Sindhi OCR in printed documents, forms, sheets, and other texts.

A convolutional neural network (CNN) was built to serve as a baseline that can be used in device models for Sindhi OCR. The model is font-invariant, that is, it recognizes not just the 256 fonts, but also the ligatures in any future fonts. The model trained on SLRSet exhibited a state-of-the-art test accuracy of 91.85%. It also promises to recognize the ligatures provided in unseen fonts. Hence, any model for the Sindhi OCR, when trained via transfer learning on our baseline, will also be font-invariant and have state-of-the-art performance in terms of accuracy.

2. Methodology

The synthesis of images with Sindhi ligatures is the first objective of this research, as benchmark datasets play a key role in developing state-of-the-art models. The steps performed to synthesize the dataset of Sindhi ligatures are as follows. A set of ligatures was collected from multiple texts. An exhaustive list of Sindhi fonts that can be found online was collected. A few fonts were unrealistic or too informal; hence, they were *outcasted*; that is, they were discarded. Images were synthesized such that each image contained a single ligature. The data were split into training and test sets based on font styles with a 75:25 ratio. A deep learning neural network model was trained for Ligature Recognition on different variations of data. The trained models were tested, and the results were evaluated using quantitative measures *precision*, *recall*, *accuracy*, and $F_1 - score$.

2.1. Data Synthesis

Urdu already has an extensive set of more than 18 thousand ligatures that Qaida [33] is based on. However, Sindhi did not have this luxury. No research has been conducted on the Sindhi ligatures. Therefore, to synthesize a large-scale Ligature Recognition dataset of annotated images, a large set of ligatures was extracted from multiple sources of text. To incorporate a large number of fonts, 284 fonts could be found online, and 256 of them were found to be suitable for consideration. Finally, the images were generated such that each image contained a single ligature.

2.1.1. Corpora Mining Although there has been some research work done on the Sindhi NLP † and some Sindhi corpora have been developed [42, 43], however they lack the diversity that the Sindhi language has in terms of its vocabulary size. Therefore, certain corpora were mined. Not all the cases were considered. There were four that were sufficiently large, formal, and free from spelling errors; Shah Jo Risalo, Sachal Jo Kalam, Quran Jo Tarjumo, and Digital South Asia Library. Only these four were used for further cleaning.

2.1.2. Text Cleaning The Arabic Unicodes range from 0x600 to 0x6FF. All the characters in the text that fell outside the range were dubbed as foreign and were replaced with a space character before further preprocessing. The replacement of foreign characters with a space character, and not a void character, was performed to avoid any unwanted merger of the two ligatures. All Harakat or other markers that were present over the alphabet in the text were removed. Some Arabic alphabets can also be formed using the base alphabet, followed by Maddah, or by a Hamza above or below. To avoid duplication in the vocabulary of words or the set of ligatures, the pairs of unicodes are replaced with their counter singleton way of achieving the same. The EXTENSION (0x640) is a special Unicode that is used in the Arabic script and, by extension, Sindhi to prolong words for the beauty and alignment of the text. These extensions have been removed from the texts. The English punctuation was removed along with the rest of the foreign characters. However, Arabic punctuation was removed, that is, replaced with spaces, separately. Finally, all the Unicode characters, except the main Sindhi characters, were removed.

2.1.3. Words Extraction Shah Jo Risalo is considered to be one of the best novel works in the Sindhi language. Its content was taken from the GitHub repository of Amar Fayaz Buriro ‡. A set of 48,444 clean words was extracted from the text. The

† `https://sindhinlp.com/`
‡ `https://github.com/amarfayazburiro/shah-sachal-sindhi-language/blob/master/Risalo(without-airab).txt`

poetry of Sachal Sarmast in Sindhi is another novel set of literature in the Sindhi language. Its content was taken from the same GitHub repository of Amar Fayaz Buriro §. After cleaning, 74,732 unique words were unioned with the previous set of words. The translation of the Quran by Hazrat Taj Mahmood Amroti is recognized internationally and accepted worldwide as one of the earliest and best translations of the Quran. The translation text was taken from the website of Abdul Majid Bhurgri ‖ and can also be accessed from my personal repository ¶. In total, 123,176 unique words were extracted from the text after cleaning. The Digital South Asia Library is a digital resource collection that provides materials for reference and research on South Asia [+]. Parmanand Mewaram authored a Sindhi-English dictionary published in Hyderabad, Sindh by the Sindh Juvenile Co-operative Society in 1910. In addition, the Center for Language Engineering (CLE) offers another Sindhi-English dictionary. This version was created through a project overseen by Dr. Sarmad Hussain, who led the CLE at the Al-Khwarizmi Institute of Computer Science, a division of the University of Engineering & Technology Lahore in Pakistan. The project was a collaborative effort involving Dr. Jennifer Cole from the University of Illinois Urbana-Champaign, and received funding from the South Asia Language Resource Center at the University of Chicago. These dictionaries are available online [*]. The **Parmanand Mewaram** ♯ dictionary was mined and cleaned. In total, 636,049 unique words were identified. The Sindhi Lughata by **Nabi Bakhshu Khanu Balocu** †† was also mined and cleaned. A total of 6,341,972 words were identified. Collectively, there were 24,024,262 words found in the four corpora discussed above. However, this was a collection of many repeated words. The frequencies of each word were calculated, and a set of unique words was formed that contained 104,145 words. The maximum size of a word was 14, that is, it contained many characters.

2.1.4. Ligature Collection Observing the ending characters of ligatures, it was established that the ligatures always end at some particular alphabet who does not get attached to the next character, hence forming a ligature whenever they appear in a word. We call these lig-breakers because they break a ligature into many. Based on the lig-breakers, space was added after each occurrence of lig-breakers. The text was split based on space, and a set of 22,597 ligatures were found.

§ https://github.com/amarfayazburiro/shah-sachal-sindhi-language/blob/master/Corpus.txt
‖ https://bhurgri.com/
¶ https://github.com/zeeshanalipanhwar/quran-jo-tarjumo
[+] https://dsal.uchicago.edu/about.html
[*] https://dsal.uchicago.edu/dictionaries/list.html#sindhi
♯ https://dsal.uchicago.edu/dictionaries/mewaram/
††https://dsal.uchicago.edu/dictionaries/baloch/

2.1.5. Fonts Acquisition The Sindhi OCR system does not support the diversity of fonts in the Sindhi language. Generic OCR systems must be font invariant. Therefore, a quest was initiated to find as many fonts as possible online. A total of 264 fonts were found online and used for writing Sindhi. These fonts are used to create the ligature dataset, making it font-diverse so that the models trained on the data are font-invariant. In the following paragraphs, an overview of the acquired fonts supporting AUS U0600 is presented.

PakType Pakistani Typography † are sets of Unicode based open source OpenType fonts in which the Latin script is based on the Universalis ADF Std Regular, as shown in Table 2. SindhiFonts ‡ are collections of fonts as shown in Table 3. SindhSalamatFonts § is the largest collection of fonts from 12 different designers as shown in the Table 1.

Table 1. Sindh Salamat Fonts

	Designer	**Number of Fonts**
1	Abdul Majeed Bhurgiri	3
2	Abdul Sattar Zarar	20
3	Magero Aziz	9
4	Ahmed Soomro	2
5	Shabeer Kumbhar	13
6	Sanaullah Saram	154
7	Lajpat Ra Meghwar	2
8	Irfan Shikarpuri	8
9	Mansoor Pirzado	1
10	Sindh Salamat Team	6
11	Masoom Sahil	2
12	Charan	1
	Total	**221**

† https://paktype.sourceforge.net/index.html
‡ https://sindhifonts.com/sd
§ https://font.sindhsalamat.com/

Table 2. PakType Pakistani Typography

Font	**Glyphs Designer**
PakType Ajrak	Ayaz Gul Soomro
PakType Naqsh	Lateef Sagar Shaikh
PakType Naskh Basic	KACSTQurn
PakType Naskh Basic Wide	KACSTQurn
PakType Tehreer	Mohammad Hanif

Table 3. SindhiFonts

Collection	**Size**
MB-Sarem-MA-Kareem-Khilji-2-fonts	2
Sarem-BD-2-Fonts	2
Sarem-M-Azam-Fonts-Pack	2
Top-20-Sindhi-Fonts-of-2021	20
Sarem-All-158-Fonts-Old-Versions	158
Sarem-Hafsa	2
Sarem-Sindh-Online-School-Pack-With-Harkat-Ok	4
Total	**190**

2.1.6. Fonts Outcasting Some of the acquired fonts were unrealistic or too informal to consider, which we termed as the *outcast* . Only the following 8 fonts were outcast; 'Mangrio Aziz Herodep', 'MB Sarem of Roz-e-Dhanni 2', 'Mangrio Aziz Hindi', 'MB Sarem of Roz-e-Dhanni 3', 'MB Sarem Dokri', 'MB Sarem of Roz-e-Dhanni', 'MB Sarem Hala', and 'MB Sarem Sindhrri'. Figure 1 shows samples of the outcast fonts. The remaining 256 fonts were considered for the Data Synthesis. The fonts covered in the process have a diverse range of glyphs and follow a wide range of styles.

2.1.7. Image Synthesis Choosing the right font size depends on factors like readability, medium, audience, and design preferences. While no definitive standard exists, commonly followed guidelines for Latin script suggest a font size of 10-12 points for body text [44]. Headings and subheadings typically use larger font sizes, ranging from 14 to 24 points, to provide visual hierarchy [44]. The display of large text elements employs font sizes greater than 24 points [44]. Considering that accessibility is crucial, guidelines such as WCAG recommend minimum font sizes and contrast ratios [45]. These suggestions are not rigid rules, and can be adjusted based on specific project requirements and target audiences. Prioritizing overall readability and user experience is essential for determining appropriate font sizes. Arabic fonts are typically larger than Latin fonts because Arabic letters are

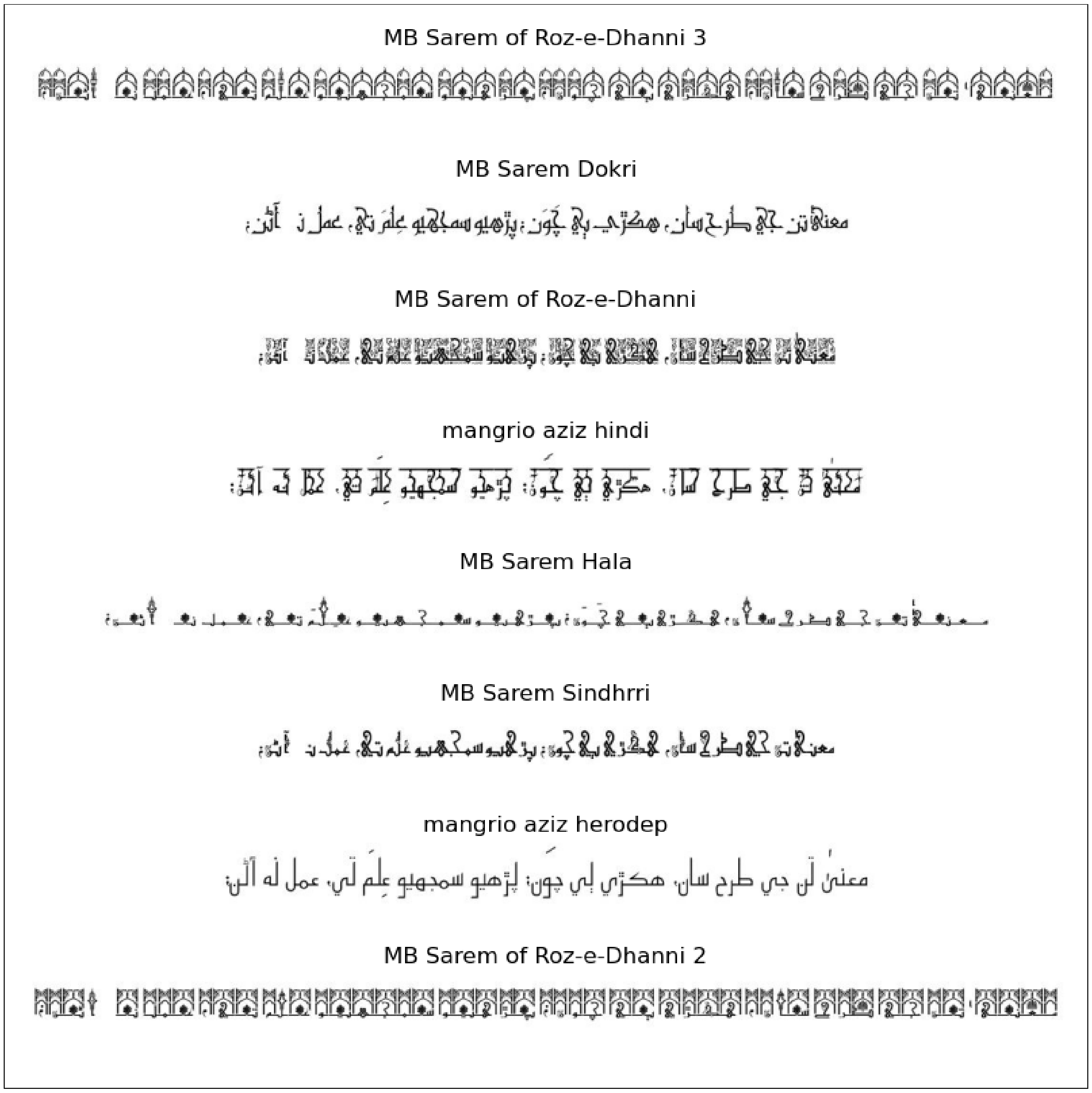
MB Sarem of Roz-e-Dhanni 3
MB Sarem Dokri
MB Sarem of Roz-e-Dhanni
mangrio aziz hindi
MB Sarem Hala
MB Sarem Sindhrri
mangrio aziz herodep
MB Sarem of Roz-e-Dhanni 2

Figure 1. Sample images with text in outcast fonts

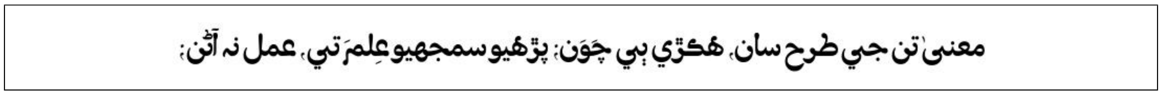
معنیٰ تن جي طرح سان، ھڪڙي ٻي ڇوَن؛ پڙھيو سمجھيو عِلمَ تي، عمل نه آڻن؛

Figure 2. Sample image with text in a clear font: *Sarem Hafsa Regular*

more intricate. Research suggests that for printed materials, font sizes between 12 and 16 points are suitable for Arabic body text, whereas on digital platforms such as websites or mobile apps, font sizes ranging from 14 to 18 pixels work well. These guidelines consider the complexity of Arabic script and the need for clear spacing between the characters. Larger font sizes help make Arabic ligatures and diacritical marks more legible, particularly at smaller sizes. In our case, we chose font sizes ranging from 12 pixels (= 9 points) to 40 pixels (= 30 points) to incorporate both the small length ligatures with sufficiently large font sizes and the larger length ligatures with sufficiently small sizes in images of size 80×80. The ligatures inside the images were restricted by a 8 pixels wide boundary so that they did not cross the boundaries of the image. Each of the 22,597 ligatures was written in 256 fonts. These constitute a database of $22,597 \times 256 = 5,784,832$ gray images labeled with ligatures as image-level labels. The ligatures were kept center-aligned in images with a black font color and white background. Figure 4 shows a ligature of 256 fonts. Figure 3 shows a few ligatures. No image augmentation was performed to increase the dataset size for ease of storage. However, several augmentations can be applied, such as affine and protective transformations, Gaussian blur, noise, and image scaling. These choices are often based on the applications and nature of the images for which a model must be trained. The following mini datasets were also prepared: (i) SLRSet-55, (ii) SLRSet-1119, (iii) SLRSet-7031, (iv) SLRSet-16472, (v) SLRSet-22597 ligatures of lengths 1, 2, 3, 4, and 9, respectively; and (vi) SLRSet-5000, of the top 5,000 most frequent ligatures.

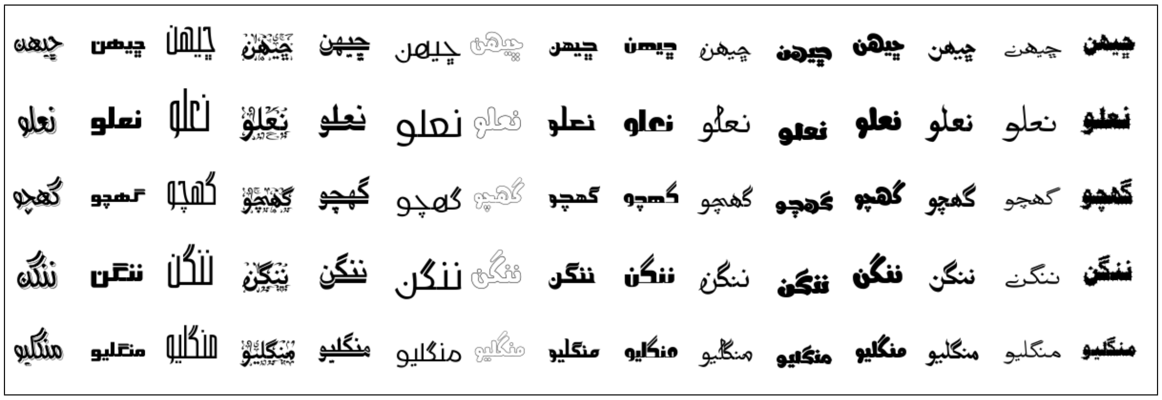

Figure 3. Images of Five Random Sindhi Ligatures in 14 Random Fonts

2.1.8. Data Splitting A good benchmark dataset must have a good split between *train* and *test* sets. The *train* set must be large enough for models to achieve generalization without falling into the case of overfitting. Test set samples must represent the population. This must have a distribution similar to that of the *train*

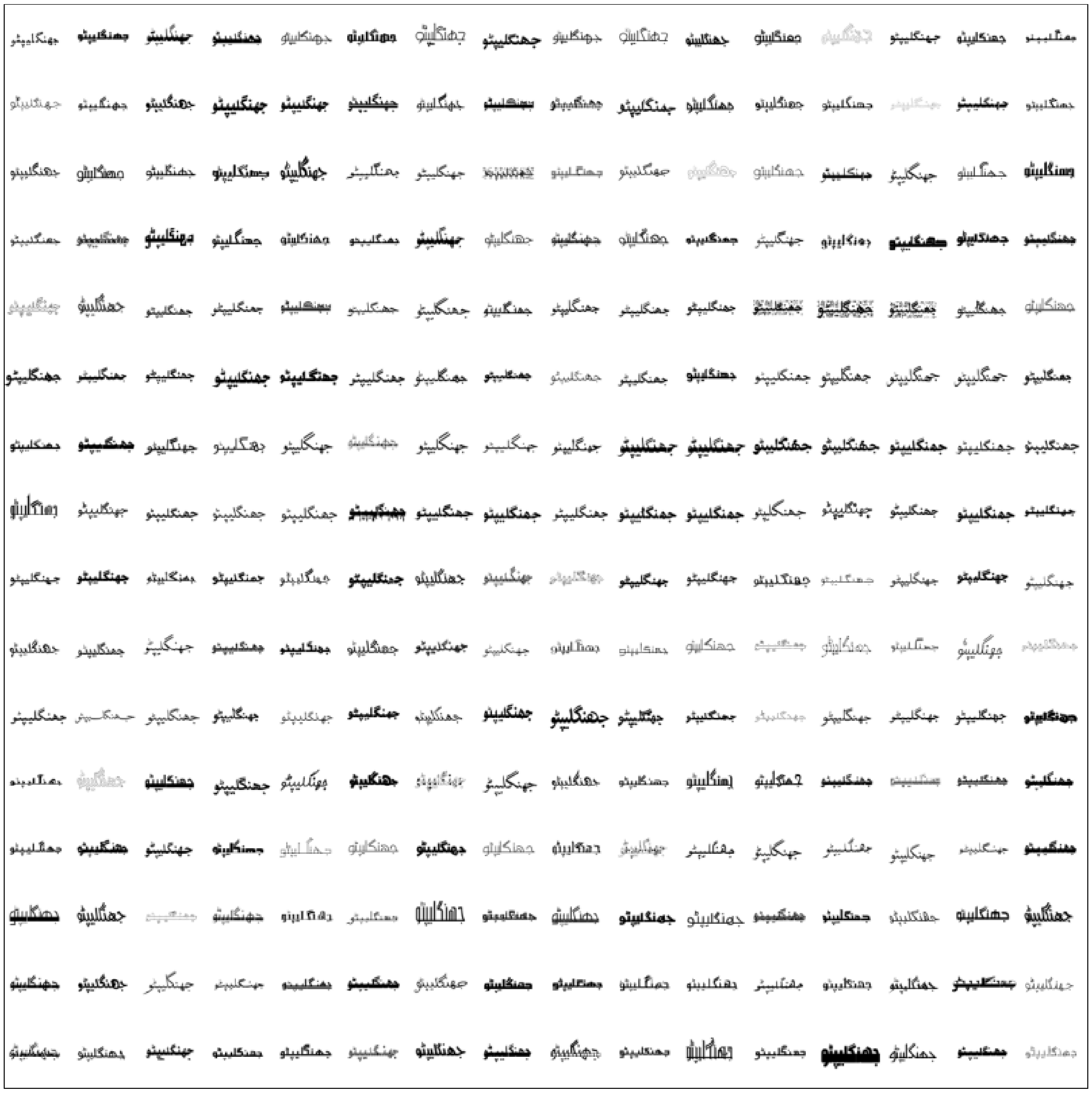

Figure 4. A Ligature in 256 Fonts

set. Therefore, obtaining an appropriate split is crucial. There are several methods of data splitting [46] such as Simple random sampling (SRS), trial-and-error methods, systematic sampling, and convenience sampling. SRS is the most commonly used technique for data splitting because of its simplicity and efficiency [46]. The samples were selected uniformly to form the *test* and *train* sets. For example, for the *test* set $D_{test} : \rho(x \in D_{test}) = \frac{n_{test}}{n_{total}}$, where $n_{test} = |D_{test}|$ and $n_{total} = |D_{total}|$. In other words, the probability of picking all samples from the dataset for the test set is the same. Hence, the resulting split had no bias. The ratio for data splitting plays a vital role in achieving SOTA models. It is generally recommended to keep the test set size small, and ratios of 90:10, 80:20, and 70:30 are common. For our dataset and its mini-versions, we chose a 75:25 split ratio with respect to font styles. This implies that 192 (= 75%) and 64 (= 25%) samples per ligature were taken for the *train* and *test* sets, respectively.

2.2. SLRNet: Sindhi Ligature Recognition Network

The residual block-based network ResNet [47], dubbed SLRNet, was chosen as the base model. It is a powerful deep learning architecture that excels in CV tasks. This overcomes the problem of vanishing gradients, thereby enabling the training of deep networks. The residual connections ensure faster convergence and maintain accuracy, even with increased depth. ResNet is parameter-efficient, focusing on learning the differences between input and output. It supports transfer learning, where pre-trained models can be fine-tuned for specific tasks, saving time and resources. Variants, such as ResNet-50, ResNet-101, and ResNet-152, offer flexibility for different needs.

2.2.1. SLRNet-18: Base Model Text instances, characters, ligatures, or words usually occupy little space. Therefore, for the Sindhi ligatures, the suitable size of images was found to be 80×80, which is larger than that of EMNIST [48] (28×28) and smaller than the input size 224×224 for most models trained on ImageNet-1k [9]. With images of this size, the deeper models would overfit the SLRSet and would not generalize. Therefore, ResNet-18, dubbed SLRNet-18, was considered for training on the SLRSet. SLRNet-18 starts with a 7×7 convolutional layer that is applied directly to the input. The result was forwarded as input to a pair of residual blocks, each having two convolutional layers with a kernel size 3×3. Next, three blocks are used, as follows: they have a pair of convolutional layers without a skip connection and a residual block of two convolutional layers. Next, average pooling is performed, which is followed by a linear dense layer, that is, a fully connected (fc) layer, of $22,597$ neurons based on the number of categories in the main model, which is trained to recognize all of the ligatures.

2.2.2. Loss Function Cross-entropy loss is a standard loss function that is used in Text Recognition [33]. It is a measure of dissimilarity between the predicted probability distribution and the true distribution. This is defined as follows.

$$\text{CEL} = -\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(p_{ij}) \qquad (1)$$

where N is the number of samples, C is the number of categories, y_{ij} is the ground-truth label (1 if sample i belongs to category j and 0 otherwise), and p_{ij} is the probability of sample i predicted to belong to category j. In the OCR, the predicted distribution is usually the output of the SoftMax layer of the model, representing the probabilities of the classes. SLRNet-18 uses a SoftMax layer at the end of the FC layer. The SoftMax function takes a vector of logits, which are the unnormalized scores or outputs of the model, and transforms them into a probability distribution over classes. It essentially 'squashes' the logits into values between 0 and 1 that sum up to 1, representing the probabilities of each class.

For a given sample X_i and category Y_j, the SoftMax function is defined as follows.

$$p_{ij} = \frac{e^{z_{ij}}}{\sum_{k=1}^{C} e^{z_{ik}}} \tag{2}$$

where z_{ij} represents the logit (score) associated with sample i and category j. The numerator $e^{z_{ij}}$ exponentiates the logit, whereas the denominator $\sum_{k=1}^{C} e^{z_{ik}}$ computes the sum of the exponentiated logits over all classes for sample i. This ensures that the predicted probabilities p_{ij} are non-negative and sum to 1, making them suitable for representing category probabilities. Using the SoftMax function to compute the predicted probabilities, the cross-entropy loss measures the dissimilarity between the predicted probabilities and the true distributions represented by the labels y_{ij}. The loss penalizes the model when it assigns low probabilities to the correct categories and high probabilities to the incorrect categories, thereby driving the model to optimize its predictions towards the true labels.

3. Results

This section starts by describing the data-generator class and how it works. It covers the setting of different models before training or testing them.

3.1. Data Augmentations

The images are originally stored as gray of size 80×80 to save space for easy storage and retrieval of the datasets. Our data loader class transforms the gray images to RGB images simply by replicating the gray channel as the Red, Green, and Blue channels of the images. Different affine transformations were randomly selected and applied to each image after loading. Table 4 provides a brief overview of the transformations.

Table 4. Affine Transformations for Data Augmentation

Transformation	Parameters	Explanation
Rotation	min=5, max=10	An angle of rotation is chosen randomly in range [min, max].
Translation	a=b=0.1	Translation w.r.t. x-, and y-axis is applied with ratios in ranges [-a, a], and [-b, b].
Scale	min=1.0, max=1.1	The scaling factor is randomly sampled from the range [a, b].
Shear	min=0, max=5	The shear angle with respect to x-axis is chosen randomly in range [min, max].

The above transformations are applied to the images randomly modifying them. This retains the size of the database. However, to achieve better generalization, the transformations can be used to generate more samples and multiply the actual dataset size by four or more by applying a sequence of the above transformations and other new ones.

3.2. Performance Measures

The OCR models use different performance measures to evaluate the quality of Text Recognition, based on the level at which their detection models operate. Common performance measures are the Character Error Rate (CER) and Word Error Rate (WER), which are defined as the number of characters or words recognized correctly over the total number of words or characters, respectively. In our case, Text Recognition was performed at the ligature level. Therefore, *accuracy* is used as the primary measure of the performance of our models. Ligature Error Rate was simply $1 - LigatureRecognitionAccuracy$. The performance evaluation measures used to access our models are explained in terms of four concepts as follows: True Positive (TP): Imagine a detective correctly identifying a criminal, ensuring justice is served; True Negative (TN): Picture a security system accurately recognizing an authorized person and allowing them access; False Positive (FP): Visualize a smoke detector triggering an alarm due to burnt toast, causing unnecessary panic; False Negative (FN): Think of a medical test failing to detect a disease, leading to a missed diagnosis and delayed treatment.

3.2.1. Precision Precision is level of correctness of the predictions made. The precision of a model in predicting a positive class is defined as TP out of the samples that are predicted as positive, i.e. the TP plus the FP. Equation 3 shows its formula.

$$precision = \frac{TP}{TP + FP} \tag{3}$$

3.2.2. Recall Recall is the ability of a model to recognise the samples of ground truth correctly. It is the ratio of TP versus the total number of samples of that class present in the ground truth, i.e. TP plus FN. Equation 4 shows its formula.

$$recall = \frac{TP}{TP + FN} \tag{4}$$

3.2.3. Accuracy Accuracy is the measure of true predictions with respect to the total number of predictions made. Formally, it is defined as shown in Equation 5.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

3.2.4. F_1- Score $F_1 - Score$ is the harmonic mean of precision and recall. It gives a score which is balanced over precision and recall.

$$F_1 = \frac{\frac{1}{precision} + \frac{1}{recall}}{2} = \frac{\frac{precision+recall}{precision\times recall}}{2} = \frac{2(precision \times recall)}{precision + recall} \tag{6}$$

or,

$$F_1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{7}$$

To determine the F_1-scores of a multi-class dataset, a method called one-vs-all is employed to calculate separate scores for each class within the dataset. The harmonic mean is applied to the precision and recall values specific to each class. Subsequently, the overall F_1-score is computed utilizing various averaging techniques, such as follows.

$$F_1 = \sum_{c=1}^{C} \frac{F_1^c}{C} \tag{8}$$

where C is the number of categories, and F_1^c is the F_1 score for the class c.

3.3. SLRNet-18 Training on SLRSet

The SLRNet-18 model was trained on different subsets of the SLRSet as well as the complete SLRSet separately using the same configurations as ResNet-18 [47]. The kernel size of the first convolutional layer was 7×7 and that of the rest was 3×3. A Rectified Linear Unit (ReLU) [49] was used as an activation function for the convolutional layers. Maximum pooling (MaxPool) is performed only after the last convolutional layer before passing its results to the fully connected (FC) layer. The models were initialized with the weights of ResNet-18, pretrained on ImageNet-1k.

3.3.1. SLRSet-55 The SLRSet-55 is a set of ligatures of length 1. The Google Colab Free version provides a Tesla T4 GPU, which is used for training SLRNet-18 on SLRSet-55. Table 5 lists the hyperparameters configured for the training. The resultant best model achieved 93.84% accuracy, 93.66% precision, 93.84% recall, and 93.59% F_1-Score on its test set.

Table 5. Hyper-parameters of SLRNet-18 trained on SLRSet-55

Hyper-parameter	**Value**
Batch Size	256
Learning Rate	0.002
Epochs	100
Number of workers	2

3.3.2. SLRSet-1119 The SLRNet-18 is trained using the Tesla T4 GPU on SLRSet-1119. The Table 6 shows the list of its hyper-parameters. It achieved 88.90% test-accuracy.

Table 6. Hyper-parameters of SLRNet-18 trained on SLRSet-1119

Batch Size	1024	**Learning Rate**	0.002	**Epochs**	4

3.3.3. SLRSet-5000 SLRSet-5000 is the set of the top $5,000$ most frequent ligatures based on the corpora from which they are extracted. The training of SLRNet-18 for this dataset was performed on an NVIDIA Tesla V100 SXM2 16 GB GPU. Table 7 lists the hyperparameters that are used to tune the model for the training. The resultant best model exhibited 90% test accuracy. This can be further improved by training for longer periods of time.

Table 7. Hyper-parameters of SLRNet-18 trained on SLRSet-5000

Hyper-parameter	**Value**
Batch Size	2048
Learning Rate	0.002
Epochs	5
Number of workers	2

3.3.4. SLRSet-22597 The NVIDIA A100 SXM4 40 GB GPU system is used on the Google Colab Pro+ ‖ platform for the training of the SLRNet-18 on the SLRSet-22597. This dataset is a super-set of all ligature datasets. Table 8 shows the hyperparameter settings for training the model on the complete dataset. The resulting model achieved 91.85% accuracy on the test set. See Table 9 for the performance measures and values achieved by the model on the test set.

‖ https://colab.research.google.com/signup

Table 8. Hyper-parameters of SLRNet-18 trained on SLRSet-22597

Hyper-parameter	**Value**
Batch Size	4096
Learning Rate	0.002
Epochs	20
Number of workers	12

Table 9. Performance of SLRNet-18 trained on SLRSet-22597

Precision	92.55%	**Recall**	91.85%	**Accuracy**	91.85%	F_1**-Score**	91.95%

4. Conclusion

This study presents a complete pipeline for Data Synthesis. Using this, a Sindhi Ligature Recognition benchmark dataset (SLRSet-22597) is synthesized. In addition, several mini versions of the dataset were prepared. SLRSet-22597 is a large-scale font-diverse Sindhi Ligature Recognition dataset that can be used as a benchmark dataset to devise Sindhi OCR systems. The dataset covers the largest ever collection of 256 fonts and 22,597 ligatures. Several font sizes were chosen based on ligature lengths covering a wide range (9–30 pt) of font sizes. A baseline convolutional neural network was also designed for Sindhi Ligature Recognition (SLRNet-18). The model showed state-of-the-art accuracy of 91.85% and F_1-score of 91.95%. It is translation invariant, scale invariant, and rotation invariant. It can recognize unseen font styles (64 fonts reserved for testing only), which shows that it is font-invariant. Our model works best for font sizes of at least 12 points, which is the ideal font size for Sindhi writings. However, it can be enhanced to enable the recognition of even smaller fonts by providing data that contain font sizes in the range of 9-11 points as well.

Our model can be used as a baseline to assess the performance of future methods and as a base for building a complete Sindhi OCR system. The models can be fine-tuned over fonts of smaller sizes, and newer font styles in the future may also be incorporated to make the models more font scale and styles invariant. Newly introduced ligatures may be added to the dataset and the models be fine-tuned accordingly for it to be able to recognise the newly introduced ligatures.

4.1. Data availability statement

The data and the models that are trained on the data as produced by this research shall be made available at `www.github.com/zeeshanalipanhwar/qaido` after this work has been published.

References

[1] Jennifer S. Cole. “Sindhi”. In: *Encyclopedia of Language & Linguistics*. Elsevier Ltd, 2006, pp. 384–387. DOI: 10.1016/B0-08-044854-2/05229-6.

[2] Ernest Trumpp. *Grammar of the Sindhi Language Compared with the Sanskrit-Prakrit and the Cognate Indian Vernaculars*. London: Trubner and Co., 1872, p. 1.

[3] Motilal Jotwani. “SINDHI: A RICH FARE”. In: *Indian Literature* 13.4 (1970), pp. 85–92.

[4] Thomas M. Breuel et al. “High-Performance OCR for Printed English and Fraktur Using LSTM Networks”. In: *2013 12th International Conference on Document Analysis and Recognition*. 2013, pp. 683–687. DOI: 10.1109/ICDAR.2013.140.

[5] Daniela Gifu. “AI-backed OCR in Healthcare”. In: *Procedia Computer Science* 207 (2022). Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 26th International Conference KES2022, pp. 1134–1143. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2022.09.169. URL: https://www.sciencedirect.com/science/article/pii/S1877050922010511.

[6] Sangeeth Reddy et al. “RoadText-1K: Text Detection & Recognition Dataset for Driving Videos”. In: *CoRR* abs/2005.09496 (2020). arXiv: 2005.09496. URL: https://arxiv.org/abs/2005.09496.

[7] R. Smith. “An Overview of the Tesseract OCR Engine”. In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2*. Vol. 2. 2007.

[8] J. Deng et al. “Construction and Analysis of a Large Scale Image Ontology”. In: Vision Sciences Society. 2009.

[9] J. Deng et al. “ImageNet: A Large-Scale Hierarchical Image Database”. In: *CVPR09*. 2009.

[10] Xiangning Chen et al. “Symbolic Discovery of Optimization Algorithms”. In: *ArXiv* abs/2302.06675 (2023).

[11] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[12] Wenhai Wang et al. “InternImage: Exploring Large-Scale Vision Foundation Models with Deformable Convolutions”. In: *arXiv preprint arXiv:2211.05778* (2022).

[13] Tianhe Ren et al. “A Strong and Reproducible Object Detector with Only Public Datasets”. In: *ArXiv* abs/2304.13027 (2023).

[14] Liang-Chieh Chen et al. “Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation”. In: *European Conference on Computer Vision*. 2018.

[15] Peng Wang et al. “ONE-PEACE: Exploring one general Representation Model toward unlimited modalities”. In: *arXiv preprint arXiv:* (2023).

[16] Tan N. N. Doan et al. “SONNET: A Self-Guided Ordinal Regression Neural Network for Segmentation and Classification of Nuclei in Large-Scale Multi-Tissue Histology Images”. In: *IEEE Journal of Biomedical and Health Informatics* 26 (2022), pp. 3218–3228.

[17] Yuxin Fang et al. “EVA: Exploring the Limits of Masked Visual Representation Learning at Scale”. In: *ArXiv* abs/2211.07636 (2022).

[18] Feng Li et al. “Mask DINO: Towards A Unified Transformer-based Framework for Object Detection and Segmentation”. In: *ArXiv* abs/2206.02777 (2022).

[19] Bao Xin Chen and John K. Tsotsos. “Fast Visual Object Tracking with Rotated Bounding Boxes”. In: *ArXiv* abs/1907.03892 (2019).

[20] Chenliang Li et al. “mPLUG: Effective and Efficient Vision-Language Learning by Cross-modal Skip-connections”. In: *ArXiv* abs/2205.12005 (2022).

[21] Jevgenij Gamper and Nasir M. Rajpoot. “Multiple Instance Captioning: Learning Representations from Histopathology Textbooks and Articles”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 16544–16554.

[22] Feng Mao et al. “Hierarchical Video Frame Sequence Representation with Deep Convolutional Graph Network”. In: *ArXiv* abs/1906.00377 (2018).

[23] Minghao Li et al. *DocBank: A Benchmark Dataset for Document Layout Analysis*. 2020. arXiv: `2006.01038 [cs.CL]`.

[24] Guillaume Jaume, Hazim Kemal Ekenel, and Jean-Philippe Thiran. “FUNSD: A Dataset for Form Understanding in Noisy Scanned Documents”. In: *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)* 2 (2019), pp. 1–6.

[25] Amanpreet Singh et al. “TextOCR: Towards large-scale end-to-end reasoning for arbitrary-shaped scene text”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 8798–8808.

[26] Urs-Viktor Marti and Horst Bunke. “The IAM-database: an English sentence database for offline handwriting recognition”. In: *International Journal on Document Analysis and Recognition* 5 (2002), pp. 39–46.

[27] Tailing Yuan. "Chinese Text in the Wild Dataset". In: 2018.

[28] Wen-gang Zhou et al. "Principal Visual Word Discovery for Automatic License Plate Detection". In: *IEEE Transactions on Image Processing* 21 (2012), pp. 4269–4279.

[29] Shuhao Qiu, Chuang Zhu, and Wenli Zhou. "Meta Self-Learning for Multi-Source Domain Adaptation: A Benchmark". In: *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)* (2021), pp. 1592–1601.

[30] Saad Mohamed Darwish and Khaled Osama Elzoghaly. "An Enhanced Offline Printed Arabic OCR Model Based on Bio-Inspired Fuzzy Classifier". In: *IEEE Access* 8 (2020), pp. 117770–117781.

[31] Ahmed Hussain Aliwy and Basheer Al-Sadawi. "Corpus-based technique for improving Arabic OCR system". In: *Indonesian Journal of Electrical Engineering and Computer Science* 21 (2021), pp. 233–241.

[32] Asghar Ali Chandio et al. "Cursive-Text: A Comprehensive Dataset for End-to-End Urdu Text Recognition in Natural Scene Images". In: *Data in Brief* 31 (2020), p. 105749. ISSN: 2352-3409. DOI: https://doi.org/10.1016/j.dib.2020.105749. URL: https://www.sciencedirect.com/science/article/pii/S2352340920306430.

[33] Atique ur Rehman and Sibt ul Hussain. "Large Scale Font Independent Urdu Text Recognition System". In: *ArXiv* abs/2005.06752 (2020).

[34] Saad Bin Ahmed et al. "Handwritten Urdu character recognition using one-dimensional BLSTM classifier". In: *Neural Computing and Applications* 31 (2017), pp. 1143–1151.

[35] Nisar Ahmed Memon, Fatima Abbasi, and Shehnila Zardari. "Glyph Identification and Character Recognition for Sindhi OCR". In: *Mehran University Research Journal of Engineering and Technology* 36 (2017), pp. 933–940.

[36] Nasreen Nizamani et al. "Optical Recognition of Isolated Machine Printed Sindhi Characters using Fourier Descriptors". In: *International Journal of Advanced Computer Science and Applications* (2019).

[37] Sindhila - Sindhi Language Authority. *Sindhila*. https://sl.sindhila.org/. Accessed: May 21, 2023, 11:20. 1999.

[38] Sindhi OCR. *Sindhi OCR*. http://www.sindhiocr.com/. Accessed: May 21, 2023, 11:20. 2017.

[39] *Ambile Hamiz Ali Sindhi OCR*. https://ambile.pk/ambile-hamiz-ali-sindhi-ocr.html. Accessed: May 21, 2023, 11:20. 2020.

[40] *Ambile*. https://ambile.pk/. Accessed: May 21, 2023, 11:20. 2019.

[41] Zeeshan Ali et al. “A Large-Scale Font-Diverse Sindhi Ligature Recognition System”. In: *2023 International Conference on Frontiers of Information Technology (FIT)*. 2023, pp. 132–137. DOI: 10.1109/FIT60620.2023.00033.

[42] Mazhar Ali Dootio and Asim Imdad Wagan. “Unicode-8 based linguistics data set of annotated Sindhi text”. In: *Data in Brief* 19 (2018), pp. 1504–1514.

[43] Mazhar Ali Dootio and Asim Imdad Wagan. “Development of Sindhi text corpus”. In: *J. King Saud Univ. Comput. Inf. Sci.* 33 (2019), pp. 468–475.

[44] Matthew Butterick. *Typography for Lawyers Essential Tools for Polished & Persuasive Documents*. Jones McClure Publishing, 2020.

[45] *Web Content Accessibility Guidelines (WCAG)*. https://www.w3.org/WAI/standards-guidelines/wcag/. Accessed on Jun 6, 2023.

[46] Zuzana Reitermanová. “Data Splitting”. In: 2010.

[47] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 770–778.

[48] Gregory Cohen et al. “EMNIST: an extension of MNIST to handwritten letters”. In: *arXiv preprint arXiv:1702.05373* (2017).

[49] Abien Fred Agarap. “Deep Learning using Rectified Linear Units (ReLU)”. In: *ArXiv* abs/1803.08375 (2018).