URDU Handwritten Optical Character Recognition Development

Hiba Shahid
School of Science and Engineering
Habib University
Karachi, Pakistan
hs08036@st.habib.edu.pk

Amnah Qureshi
School of Science and Engineering
Habib University
Karachi, Pakistan
aq08004@st.habib.edu.pk

Roozain Qazalbash
School of Science and Engineering
Habib University
Karachi, Pakistan
rq08445@st.habib.edu.pk

Mohammad Farzam Ghouri
School of Science and Engineering
Habib University
Karachi, Pakistan
mg08228@st.habib.edu.pk

Abdul Samad
School of Science and Engineering
Habib University
Karachi, Pakistan
abdul.samad@sse.habib.edu.pk

Sandesh Kumar
School of Science and Engineering
Habib University
Karachi, Pakistan
sandesh.kumar@sse.habib.edu.pk

***Abstract*—Developing an Urdu Handwritten OCR using Deep Learning Models. Motivation: Urdu, being a low-resource language, lacks tools for converting handwritten text into digital format. This project aims to create an efficient Urdu Handwritten OCR that can accurately transform handwritten text into digital form. Such a tool is essential for processing information, enabling it to be utilized for various professional, educational and other learning purposes. This effort is driven by the need to empower Urdu and support language accessibility in the digital era.**

***Index Terms*—language identification, audio classification, deep learning, multilingual speech, Pakistani languages, signal processing**

I. Introduction

The Urdu OCR (Optical Character Recognition) technology traces its roots back to as early as the 1920's. Since the advent of neural networks and advancements in computer vision techniques, it has become one of the most actively and extensively researched areas in deep learning. However, while significant work has been done on script recognition systems of both cursive and non-cursive scripts, research on Urdu text is relatively recent. Despite Urdu's rich cultural heritage and approximately 170 millions speakers worldwide, development of efficient OCR systems for it have been insufficient. This setback can be attributed to the complex nature of Urdu font (Nastalique) and the lack of resources, two of which we address in our research through the use of comprehensive datasets for model training and the implementation of efficient neural network models.

II. Research Question

The research question we are addressing is "Transforming Handwritten Urdu Text into Digital Text using Deep Learning." Our research statement can be summarized as, "Given a snippet of handwritten Urdu text, convert it into its corresponding digital representation." In this study, we will explore various deep learning architectures, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), to develop an effective Optical Character Recognition (OCR) system. The input to the model will consist of images of handwritten Urdu text, while the output will be a machine-readable digital text format. This research aims to create a robust solution that captures the unique characteristics of handwritten Urdu, including the cursive nature and contextual forms of its letters. The primary motivation behind this research is the growing need for digitization of Urdu literature and documents, which are often handwritten. As more people seek to access these texts in a digital format, the challenges proposed by the variations in human handwriting, image noise, and script complexity must be addressed. Current OCR systems for Urdu are limited and often rely on printed text images, which may not perform well with diverse handwriting styles. By applying deep learning methodologies, this research seeks to improve the accuracy of recognizing handwritten Urdu, enabling applications in education, archiving, and processing of low resource languages like URDU.

III. Literature Review

A. Methodology used in the papers

TABLE I
Models Used in Reviewed Papers

Year	OCR Language	DL Model
2023 [1]	Urdu	CNNs, Transformers
2022 [2]	Arabic	CNNs, LSTMs
2019 [3]	Urdu	CRNN
2018 [4]	Urdu	GBLSTMs
2018 [5]	Urdu	MLPs, LRs
2017 [6]	Arabic	HMM

Developing OCR systems for Urdu handwriting poses unique challenges due to the language's cursive nature, complex ligatures, and the significant impact of diacritics, where a single dot can alter the meaning of a word. Hidden Markov

Models (HMMs), often used in Arabic OCR, complement deep learning models when explicit sequential modeling is needed but face limitations with Urdu's variability and contextual dependencies due to their independence assumptions and fixed hidden states [1][5]. While Long Short-Term Memory (LSTM) networks and Recurrent Neural Networks (RNNs) better capture long-range dependencies, Urdu's higher number of ligatures and intricate character connections still present obstacles [2] [5] . This complexity has led to OCR models focusing on isolated, non-cursive Urdu text, underscoring the need for novel solutions tailored to the language's linguistic intricacies [5].

OCR pipelines generally include structured stages: image acquisition via cameras or scanners, preprocessing for binarization, noise removal, and uniform resizing, followed by segmentation of paragraphs into lines, words, and characters [1][4]. Feature extraction maps segmented data onto binary grids for classification, where words and characters are recognized based on their features. Urdu OCR systems use either holistic segmentation to directly recognize words or analytical segmentation to further break them into characters [4]. The cursive script's complexity, combined with extensive ligatures outside the standard Urdu alphabet, makes text recognition particularly demanding. Despite advancements, Urdu OCR remains underexplored, highlighting the need for further research to address its intricate challenges [2].

The paper [1] presents a robust architecture for Urdu text recognition by leveraging a Convolutional Neural Network (CNN) encoder, a transformer-based encoder-decoder framework, and innovative data augmentation techniques. The CNN encoder effectively reduces the spatial dimensions of text images, transforming the feature maps into embeddings suitable for the transformer encoder, with configurations optimized for Urdu text recognition. The transformer encoder-decoder uses trainable positional embeddings and a combination of CTC loss and cross-entropy loss, ensuring effective learning through backpropagation. This architecture also incorporates a pre-trained vanilla transformer decoder trained on the extensive 'Urdu News Corpus 1M Dataset,' further enhancing its language modeling capabilities. To improve generalization, the authors employ the Tiling and Corruption (TACo) technique for data augmentation, introducing variability through tiled image manipulation and corrupted replacements. The synergy of these components leads to improved recognition performance on smaller datasets and complex Urdu text structures.

IV. Data Set

We initially collected around 25,000 images of handwritten urdu text. After cleaning and pre-processing the data we were left with 18,270 images as our final input to the model. Since data is the most essential part for designing an Urdu handwritten OCR, we made sure that the data was of good quality and adequate for this purpose. Acquiring this type of dataset was a major challenge as there was a lack of such data available online, especially for Urdu handwritten images available with their corresponding labels.

The images we collected consist of varying styles and scripts of Urdu. From the total input data, 80% of our data was split for training the model, 10% was for validation and 10% for testing. The image dimensions were kept consistent across all models with a height of 64 pixels and width 512 pixels.

The main sources through which we collected our dataset were *kaggle.com* and handwritten by ourselves.

A. Acquisition

The primary source for our dataset collection was *Kaggle*. It was a challenge to find the data on *Kaggle* as much of the dataset that was published either had only images of handwritten text or only text files. Furthermore, some of the dataset that we collected was in spreadsheet format, that we utilized by converting it into respective text files for labels. Additionally, we used the datasets that had digital text but no images by creating the handwritten images for them by ourselves. We also asked family members and friends to write a couple of sentences on paper and send them to us as images every day. Those images were then cropped and used as sentences for input.
Following are the links to our dataset which we took from Kaggle:

- PUCIT-OHUL: PUCIT Handwritten Urdu Lines Dataset (kaggle.com)
- Urdu OCR data (kaggle.com)
- 80 clean handwritten Urdu OCR lines (kaggle.com)
- UNHD Dataset (kaggle.com)
- Nust-UHWR Dataset (kaggle.com)

These images only contain sentences in a single line format. The datasets that we have obtained are written in different styles of characters and punctuations. Most of the kinds of styles and scripts available in the Urdu language were added to the dataset. Since Urdu letters revolve a lot around dots, it is a common practice to write the dots conjoined. For example, when two dots are written together, it is often represented as a dash. Keeping this in mind, data has been collected for such writing practices as well. We made sure to choose data that is of excellent quality, which means blurry pictures and smudged handwriting was avoided. The output data is in the form of text files which contain digital text corresponding to each image.

B. Example of Our Data

Our dataset is divided into two subcategories, where every input value corresponds to an output value. The x input represents handwritten Urdu text images taken from a smartphone or any other camera source, while the corresponding y value is a text file holding the Urdu digital text. The (x, y) pair of our data can be represented as follows:

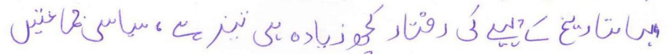
ہماری تاریخ کے پہیے کی رفتار کچھ زیادہ ہی تیز ہے ، سیاسی جماعتیں

Figure 1: Sample Input Image

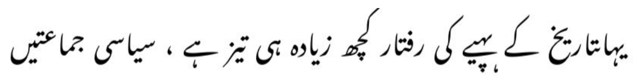
یہاں تاریخ کے پہیے کی رفتار کچھ زیادہ ہی تیز ہے ، سیاسی جماعتیں

Figure 2: Sample Output Text

V. Data Image Preprocessing

The images used in our model are handwritten Urdu images. Since the model is trained using CNN, which relies on pattern recognition, image preprocessing is necessary to convert any type of image into a suitable format for training. An example of a preprocessed image is shown below:

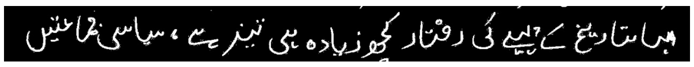
یہاں تاریخ کے پہیے کی رفتار کچھ زیادہ ہی تیز ہے ، سیاسی جماعتیں

Figure 3: Post processed Image

VI. Methodology

Through the review of different research works, we noticed the pattern of usage of CNNs as the foundational model. This is due to its feature extraction property, which allows the model to extract features from the handwritten images, which can then be mapped for acquiring respective digital text.

A. Dataset Processing

For processing our Dataset we dealt with the Labels and the Images separately. Firstly, we processed our images using the following key components in our approach:

1) **Directional Histogram Calculation:** The code calculates a directional histogram (horizontal or vertical) to capture the intensity distribution of the image, helping identify potential areas of interest for segmentation.
2) **Smoothing the Histogram:** A smoothing function is applied to the histogram to reduce noise and make the peak detection more accurate.
3) **Thresholding:** Intensity thresholding is used to convert the image into a binary format, separating text from the background for easier processing.
4) **Peak Detection:** The smoothed histogram is analyzed to detect peaks, which indicate potential line boundaries in the image.
5) **Run-Length Encoding (RLE):** Run-length encoding is used to identify stable segments in the histogram, helping to filter out small noise artifacts.
6) **Optimal Thresholding:** The threshold for detecting line cuts is adjusted to ensure the correct number of cuts, matching the detected peaks.

For label preprocessing and dataset preparation, the approach involved two main steps:

1) **Label Cleaning:** We cleaned the labels by removing any unwanted characters (e.g., BOM marker) and stripping unnecessary spaces from the beginning and end.
2) **Label Vectorization:** The cleaned labels were then converted into a numerical representation suitable for model training. This was done by splitting the labels into individual characters, converting each character to its corresponding index.

We prepared the dataset by combining the processed images and their corresponding vectorized labels. The resulting datasets, consisting of image paths and labels, were created separately for training, validation, and testing.

B. Models Used: Architecture

As mentioned in section III, additional layers such as bi-directional LSTM and transformer layers significantly enhance performance. Therefore, we incorporated these layers to achieve improved outcomes.

1) **CNN:** As one of the most widely used foundational models, we experimented with a CNN to generate digital text by extracting features from preprocessed input images. The model comprised three convolutional layers, each with a kernel size of 3 and padding of 1. The ReLU activation function was used after each convolution layer to introduce non-linearity, while batch normalization and max pooling were applied between layers to stabilize training and enhance feature extraction. This architecture effectively extracted feature maps from the input images.
2) **CNN-BLSTM:** This model comprised three convolutional layers, each with a kernel size of 3, same padding, and ReLU activation. Batch normalization and max pooling were applied between layers to stabilize training and enhance feature extraction. The extracted features were then passed to a bi-directional LSTM (BLSTM) layer, which leveraged bidirectional context for sequential understanding. Finally, the output from the BLSTM layer was sent to a fully connected (dense) layer for classification. This combination allowed the model to generate sentences as the output, effectively translating the handwritten text into digital form.
3) **CNN-Transformer:** The CNN component of this model consisted of five convolutional layers, each with a kernel size of 3 and padding of 1, except for one layer with a kernel size of 4 and valid padding. Batch normalization and max pooling were utilized to enhance feature extraction. The extracted features, along with their labels, were then sent to the transformer model. The encoder processed the CNN-extracted features, while the decoder generated the final digital text output. The transformer architecture ensured coherence in the predicted text by leveraging its attention mechanism to learn word patterns effectively.
   The model utilized the following:
   - **roBERTa Encoder:** A pre-trained encoder from the Hugging Face platform, fine-tuned on our dataset for improved feature encoding.
   - **GPT-2 Decoder:** A pre-trained decoder that works alongside the encoder to enhance predictions of digital text, ensuring better coherence and accuracy.

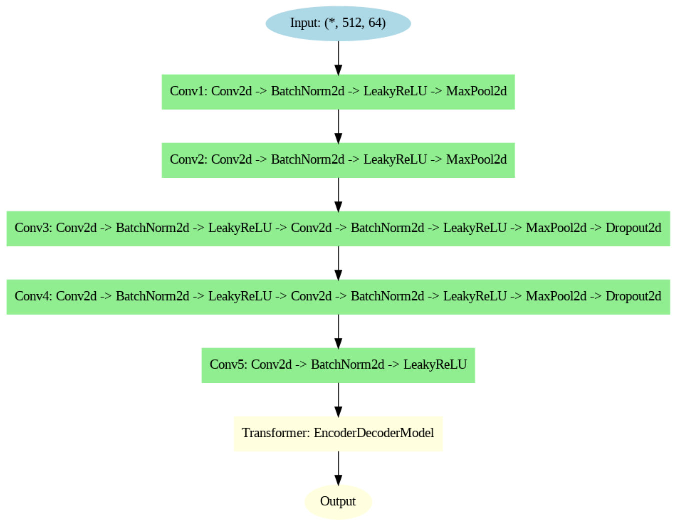
Input: (*, 512, 64)
Conv1: Conv2d -> BatchNorm2d -> LeakyReLU -> MaxPool2d
Conv2: Conv2d -> BatchNorm2d -> LeakyReLU -> MaxPool2d
Conv3: Conv2d -> BatchNorm2d -> LeakyReLU -> Conv2d -> BatchNorm2d -> LeakyReLU -> MaxPool2d -> Dropout2d
Conv4: Conv2d -> BatchNorm2d -> LeakyReLU -> Conv2d -> BatchNorm2d -> LeakyReLU -> MaxPool2d -> Dropout2d
Conv5: Conv2d -> BatchNorm2d -> LeakyReLU
Transformer: EncoderDecoderModel
Output

Fig. 1. Neural Network Flowchart

VII. Evaluation Metrics

The performance of the models were evaluated using the following metrics:

1) **Character Error Rate (CER):** CER measures the percentage of character-level errors (insertions, deletions, substitutions) between the predicted text and the ground truth. A lower CER indicates better model accuracy.

$$\text{CER} = \frac{S + D + I}{N}$$

Where:
S = Number of substitutions
D = Number of deletions
I = Number of insertions
N = Total number of characters in the ground truth

2) **Loss Functions:**
   - **CTC Loss:** Used for CNN and LSTM models, it helps align input sequences with variable-length outputs effectively.
   - **Sparse Categorical Cross-Entropy Loss:** Applied to the Transformer-ROBERTA model, it optimizes classification over vocabulary tokens without requiring one-hot encoding.

VIII. Experiments and Results

As defined in the previous section, we trained three models for our research problem. All of these models were trained with different hyper-parameters and evaluated using the aforementioned metrics to arrive at the best model.

We started with a basic CNN model and the architecture for this model has been described earlier. The training parameters for this model are mentioned in table II

TABLE II
CNN

Hyper-parameters	Value
Epochs	50
Batch size	32
Learning rate	0.0001
Optimizer	Adam
Loss function	CTC layer
Early stopping patience	5

After training our model on CNN, we figured that the training and testing accuracy was not as good as we expected. The training accuracy was 20.29% and testing accuracy was 19.45%. The results shown below in figure 2 clearly describes the accuracy for test data using this model.

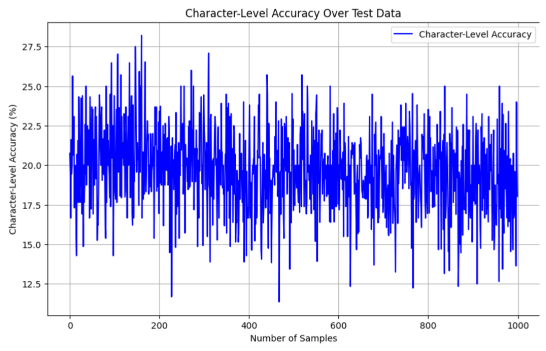

Fig. 2. Testing Accuracy for CNNs

The next model is based on the CNN-LSTM model utilized in the paper [2], we built upon the model adding more CNN layers and altering the parameters mentioned in III

TABLE III
CNN + BLSTM

Hyper-parameters	Value
Epochs	50
Batch size	64
Learning rate	0.001
Optimizer	Exponential decay
Loss function	CTC layer
Early stopping patience	3

For this model as well we did not get a very good result either. The character training CER for this model was 2.28 and for testing was around 6. The loss results as shown in figure 3 were also unsatisfactory, primarily because LSTMs are challenging to train and require significant memory. This leads to high GPU RAM usage, and due to time and memory limitations, we were unable to fine-tune the model as expected.

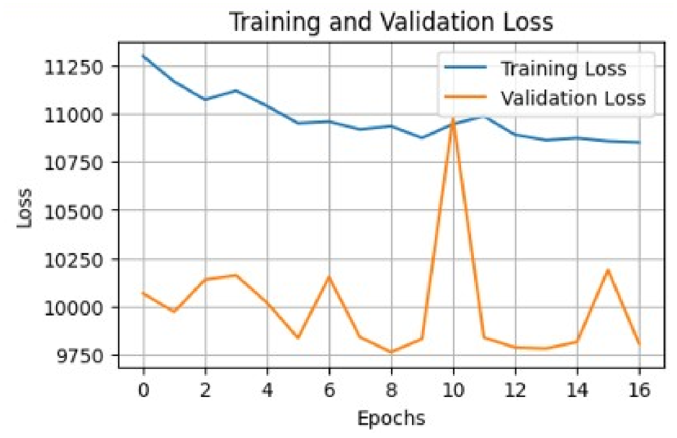

Fig. 3. Loss per epoch for CNN-LSTM

The next model was inspired from the paper [1], where we used CNN combined with the Transformer Encoder-Decoder.The parameters used are defined in the table IV

TABLE IV
CNN + TRANSFORMER

Hyper-parameters	Value
Epochs	50+30
Batch size	16
Learning rate	0.001
Optimizer	Adam
Loss function	default
Early stopping patience	3

For this model we got significantly better results. This model got a training accuracy of 80.30% and testing CER as 0.8185. The train and validation loss for this model can be observed from the graph below 4.

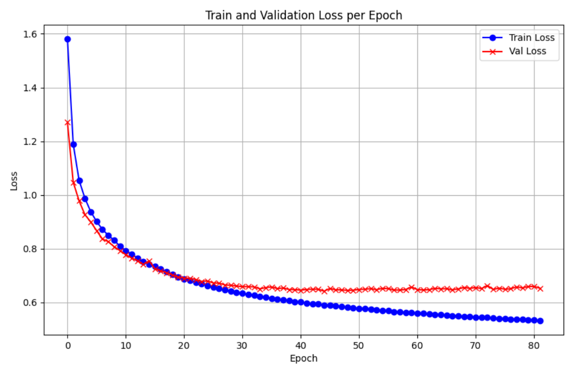

Fig. 4. Train loss for CNN+Transformer

Following is the graph representing CER observed during the testing phase of this model in the figure 5.

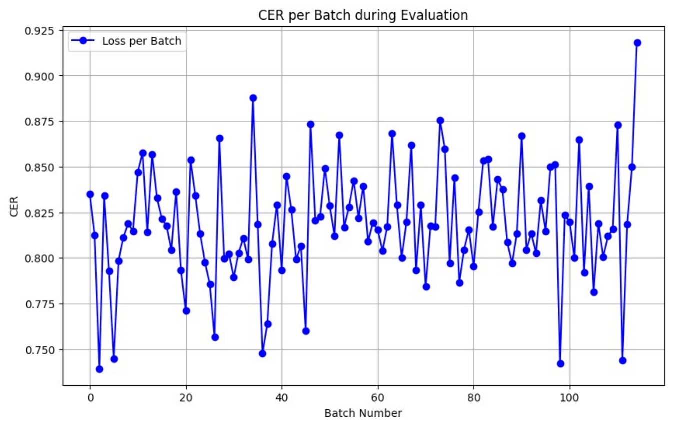

Fig. 5. CER for CNN+Transformer

The loss per batch of this model observed during the testing phase can be seen from the graph below in figure 6.

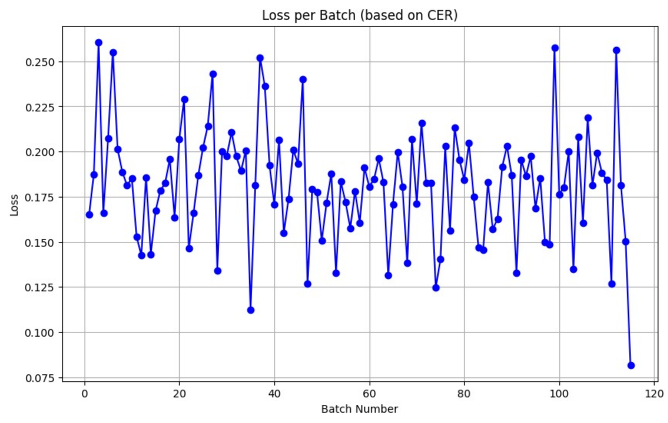

Fig. 6. Loss per batch for CNN+Transformer

The CNN+Transformer model worked out as the best model for us as it was giving us the maximum accuracy with minimum loss and high learning rate. It proved to be learning well through the validation tests done. During the testing phase, the input image was sent to the CNN which then extracted meaningful features from them in the form of tensors. 125 such features were extracted per image with a depth of 256 each. These feature maps were fed into the Transformer, where the encoder captured contextual relationships between features, and the decoder generated the predicted sentences. This combination allowed the model to effectively learn patterns and produce accurate results during testing. The results for our model framework comapred to the previous [1], is mentioned in the table V

TABLE V
COMPARISON WITH PREVIOUS WORK

Results	**Previous Model [1]**	**Proposed Model**
Handwritten Dataset	8,000	18,270
Training CER	0.02	0.81
Validation CER	0.06	0.32
Testing CER	0.06	0.80

IX. Challenges and Limitations

Creating an OCR system for Urdu handwritten text presents several challenges and limitations due to the unique characteristics of the script, data availability, and computational complexity. Urdu is written in a cursive script, where letters within a word are connected, making segmentation difficult. Additionally, many words in Urdu are formed by joining letters and the shape of letters changes depending on their position in a word. The shape of the letter depends if the letter is in the initial part, middle, final, or isolated. This requires the model to recognize contextual variations of letters. Furthermore, the available datasets often suffer from repetitive words, causing models to memorize these patterns rather than generalize to new examples.

One of the major challenges faced during the project was the limited computational resources. Running this model on CPU for training was very slow, which significantly delayed the process and we did not have enough memory space to run our model which resulted in limitations in fine tuning our model.

X. Future Works

In future, several improvements can be explored to enhance the performance of our Urdu Handwritten OCR model. One potential improvement is optimizing the CNN+Transformer model by experimenting with alternative transformer architectures like Vision Transformers which may yield better results in capturing the nuances of Urdu script. Exploring advanced CNN models like Resnet and DenseNet may further improve the accuracy of handwritten image text recognition. We can also further improve our model by expanding the dataset with more diverse sources which would improve generalization across different handwriting styles and scripts of Urdu. A vast dataset with more variety of words will help the model learn on a broad spectrum instead of memorizing a few frequently appearing words. To further improve our model, we can also train our model to recognize longer text sequences such as paragraphs without significant loss in accuracy.

XI. Conclusion

While the CNN-Transformer model showed promising results, achieving complete accuracy in Urdu handwritten OCR remains challenging. The architecture requires further refinement to address the unique complexities of Urdu, particularly its diverse diacritics and intricate handwriting styles. These factors demand meticulous preprocessing and tailored training strategies for improved performance in neural network models.

References

[1] Maqsood, A., Riaz, N., Ul-Hasan, A., Shafait, F, "A Unified Architecture for Urdu Printed and Handwritten Text Recognition," *Document Analysis and Recognition - ICDAR 2023: 17th International Conference, Proceedings*, part. IV, 2023.

[2] N. Alrobah and S. Albahli, "Arabic handwritten recognition using deep learning: A survey," *Arabian Journal for Science and Engineering*, vol. 47, no. 9, pp. 9943–9963, 2022.

[3] N. ul Sehr Zia et al., "A Convolutional Recursive Deep Architecture for Unconstrained Urdu Handwriting Recognition," *Neural Computing and Applications*, Sep 2021.

[4] K. A. S. R. Rizvi et al., "Optical Character Recognition System for Nastalique Urdu-like Script Languages Using Supervised Learning," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 33, no. 10, p. 1953004, 2018.

[5] N. H. Khan and A. Adnan, "Urdu Optical Character Recognition Systems: Present Contributions and Future Directions," *IEEE Access*, vol. 6, pp. 46019–46046, 2018.

[6] M. Alghamdi and W. Teahan, "Experimental Evaluation of Arabic OCR Systems," *PSU Research Review*, vol. 1, pp. 229–241, 2017.

[7] L. Alzubaidi et al., "Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions," *Journal of Big Data*, vol. 8, no. 1, pp. 1–74, 2021.