Handwriting Recognition and Information Retrieval System

Mr. Abdulrahman Shaikh[1], Ms. Ankita A Chavan[2], Mr. Atiqur Rehman Sayed[3], Mr. Rahul Madiwal[4], and Dr. Madhusudhan Kulkarni[5]

[1,2,3,4] B.Tech Students

[5] Associate Professor

Department of Electronics and Communication Engineering

KLS Vishwanatharao Deshpande Institute of Technology,

Haliyal 581329, India

Abstract*— This project is a full-stack handwriting recognition app that blends machine learning with modern web technologies to turn handwritten text into digital content. The frontend is built using React and JavaScript, while the backend runs on Python and Django. At its core, the app uses a pre-trained machine learning model based on the EMNIST dataset to recognize handwritten digits, uppercase letters, and unique lowercase letters. To tell similar characters apart, it even uses a clever height-based approach to accurately detect letter casing.Designed with real-world use in mind—like in education, healthcare, and business—the app helps quickly digitize handwritten notes and forms. It also features ChatGPT integration, so users can get instant explanations and context for the text they've scanned, making the experience more interactive and insightful.By combining deep learning with AI, the app delivers accurate handwriting recognition, quick info retrieval, and a clean, user-friendly interface. It's a powerful tool for boosting productivity and simplifying the move from paper to digital.

Keywords: HTML, CSS, JavaScript, Python, Django, EMNIST Dataset, ChatGPT Integration.

I. INTRODUCTION

In today's fast-moving digital world, accurate and efficient data entry is more important than ever. Whether it's in education, healthcare, business, or administration, converting handwritten notes into digital text is still a common task—but it's not always easy. Traditional OCR (Optical Character Recognition) tools often struggle with different handwriting styles, leading to mistakes and wasted time.

This project offers a smarter solution: a handwriting recognition web app that combines machine learning with modern web technologies to tackle these challenges head-on. It's built as a full-stack application, with React and JavaScript powering the frontend and Python with Django handling the backend.

At the heart of the system are deep learning models trained on the EMNIST dataset. These models are designed to accurately recognize handwritten digits, uppercase letters, and unique lowercase letters, making the app both reliable and versatile for real-world use.

To enhance recognition accuracy, especially for visually similar upper and lower case characters, the application applies a height-based classification logic that distinguishes letters based on their position and size on the canvas.

What sets this application apart is its integration with ChatGPT, transforming it from a basic recognition tool into an interactive knowledge platform. After recognizing handwritten text, users can access ChatGPT for detailed explanations, contextual understanding, and supplementary information related to the recognized content. This feature is especially beneficial in academic or professional settings, where users often require quick insights or elaboration on specific terms or phrases.

The application is not just a technical demonstration—it is a practical, user-centric tool built for real-world utility. Its intuitive interface makes it accessible even to non-technical users, while its robust backend and AI capabilities ensure reliable and accurate performance. Whether it's digitizing handwritten notes, forms, or archival documents, this system improves productivity, reduces manual entry errors, and enhances the overall user experience.

In summary, this paper represents a significant innovation in the field of handwriting recognition. By integrating deep learning models and interactive AI, it bridges the gap between analog input and digital processing. It redefines what a handwriting recognition system can do—not only making handwritten text machine-readable but also enriching it with real-time, AI-powered intelligence. The system lays a strong foundation for research and development in intelligent document processing.rd-level recognition, rather than character-level recognition.

II. LITERATURE SURVEY

In [1], N. Rakask, P. Kushal Redds discuss a system for automating handwritten text conversion to digital form using image processing and Handwritten Text Recognition systems. The system focuses on recognizing and converting text from handwritten documents to machine-readable digital text, enhancing the automation process for digitizing handwritten material.

In [2], Batuhan Balci, Due Suadati, Dr. Geetho I. Nikitha A explore two approaches: direct word classification and character segmentation for Handwritten Text Recognition. The methodology involves collecting handwritten text data, extracting features, and training a model using an LSTM-based approach focused on word-level recognition, instead of character-level segmentation.

In [3], Hilesh Bhardwaj, Rinco Das present an approach to improve Handwritten Text Recognition accuracy using a Deep Learning model, specifically Long Short Term Memory (LSTM). The system focuses on word-level recognition rather than character-level recognition to enhance the overall accuracy of handwritten text recognition.

In [4], Nikitha A, Dr. Geetha J propose a method for Handwritten Text Recognition using Deep Learning. This method involves collecting handwritten text data, extracting features, and training a model using an LSTM-based approach focused on word-level recognition, rather than character-level recognition.

III. EASE OF USE

A. Seamless and User-Friendly Interface

A major focus of this project is keeping the user experience simple and accessible for everyone. The frontend, built with React.js, is designed to be clean, intuitive, and easy to navigate—whether you're a student, teacher, professional, or someone just looking to digitize handwritten notes. You don't need any technical skills to use it.

The interface features a drawing canvas where users can write characters using a mouse or touchscreen, making it fully compatible with both desktops and mobile devices. Everything is responsive and straightforward—just open the app in your browser, start drawing, and get instant text predictions.

To make things even easier, the app includes handy features like clear buttons, real-time feedback, and helpful tooltips that guide users along the way. There's also an upload option, so users can submit handwritten or scanned image files for recognition. Whether you're jotting down notes on the fly or working with older handwritten documents, the app adapts to your needs and makes the process smooth and hassle-free.

B. Beginner-Friendly Backend and Easy Setup

Even though the backend is powerful, it's designed to be simple and beginner-friendly. Built with Python's Django framework, everything is organized in a clear, modular way. Different parts of the system—like model loading, image processing, prediction logic, and API responses—are kept in separate files with straightforward names. This makes it easy to find what you're looking for and make changes, even if you're new to backend development.

Setting things up is also hassle-free. There's a .env file where you can quickly tweak important settings like model paths or debug options without digging into the code. Plus, the backend is well-documented, with plenty of in-line comments to help you understand how things work as you go.

For testing and debugging, tools like Django's built-in dev server and Postman make it easy to experiment and fine-tune everything locally. Whether you're just learning or want to customize things, the backend won't slow you down.

C. No Training Needed — Just Plug and Play

The machine learning side of the app is just as simple to work with. All the handwriting recognition models come pre-trained using the EMNIST dataset, which means there's no need to train anything yourself. The models are saved in ready-to-use formats (like .h5 or .pkl) and can be loaded directly into the backend.

This "plug-and-play" setup means you don't need to worry about complex stuff like model training, GPU configurations, or large datasets. Everything runs on the server, and predictions are made in real time through efficient APIs.

Deploying the app is also straightforward. Whether you want to run it on Heroku, Render, or even your own machine with Docker, setup guides in the project repo walk you through each step. It's all about making advanced tech easy and accessible—no AI background required.

D. Smarter Interactions with ChatGPT Integration

To make the app more than just a handwriting recognition tool, it comes with built-in ChatGPT integration. Once a word or sentence is recognized, users can take things a step further by asking ChatGPT for more information—like definitions, translations, historical background, or related topics.

This turns the app into an interactive learning assistant, especially useful for students or anyone working with unfamiliar handwritten content.

Using it is super simple. There's no complicated setup—just click the "Query with ChatGPT" button, and the app sends your request in the background.

The response shows up in a friendly, chat-style interface. It's a smooth way to get instant insights from AI, and it's something traditional OCR tools just don't offer.

E. Built to Scale, Easy to Maintain

Behind the scenes, the app is designed to grow and adapt with ease. It uses a RESTful API structure, which means the frontend and backend work independently. You can update the machine learning models, tweak the UI, or improve the backend logic—without having to touch everything at once.

Maintenance is also straightforward. The system keeps track of logs, errors, and model outputs, making it easy to debug and monitor performance. Hosting is flexible too—it works great on cloud platforms like AWS, Google Cloud, or DigitalOcean, and can scale up as needed.

To keep things easy over time, the whole project is containerized with Docker and version-controlled with GitHub. That means you can deploy it again later or move it to a new server without starting from scratch.

The app doesn't demand high-end hardware either. It's lightweight enough to run on affordable devices or budget cloud services, making it accessible even in low-resource settings.

Plus, there's room to grow: future updates could include support for regional languages, offline use through a Progressive Web App (PWA), or even voice commands.

IV. SYSTEM DESIGN AND IMPLEMENTATION

A. Hardware-Free, Hassle-Free Input

One of the great things about this project is that it doesn't rely on any special hardware. Everything runs right in your web browser, so there's no need for extra devices or installations. You can use the app on just about any modern device—laptops, tablets, or smartphones—as long as you have an internet connection.

For input, users have two easy options: they can either draw directly on a digital canvas using a mouse, finger, or stylus, or they can upload an image of handwritten text (like a scan or photo). The drawing canvas is flexible and resizes automatically to give you enough room to write comfortably, whether it's a single character or a full line of text.To help improve accuracy, the system includes some smart tweaks like stroke-smoothing and a height-based method to tell uppercase and lowercase letters apart. These little details help make the handwriting recognition more reliable, especially when writing with touch on smaller screens.

B. Clean, Responsive Frontend with Real-Time Feedback

The frontend of the app is built with React.js, which means everything feels fast and responsive. It's designed to be easy to use, with a modular structure that keeps things organized behind the scenes. The drawing canvas is made using HTML5 and JavaScript, giving you smooth, pixel-level control as you write.

The look and feel are handled with Tailwind CSS, so the interface is clean, modern, and mobile-friendly—all with minimal clutter. When you draw something, the canvas automatically processes your strokes, turns them into grayscale, and resizes the image to 28x28 pixels—the exact format needed for the machine learning model trained on the EMNIST dataset.After that, your input is sent to the backend for recognition. The interface also includes:

- A file upload option for submitting handwritten notes or scanned pages,
- A live text box that shows the recognized content,
- An interactive ChatGPT panel for asking questions or getting context about what you've written,
- And room to grow, with planned features like role-based dashboards for personalized user experiences.

C. Backend & Machine Learning – Fast, Reliable, and Smart

The backend is like the engine of the system, built with Python and Django to handle all the processing. It works behind the scenes to turn your handwritten input into recognizable text using machine learning.

To make this happen, the system uses pre-trained models based on the EMNIST dataset. These models are great at recognizing numbers, uppercase, and lowercase letters, even if the handwriting isn't perfect.
Here's how it works:

- The app uses lightweight Flask microservices inside the Django framework, helping speed up how quickly it can process and predict the text.
- If the system ever gets confused between an uppercase and lowercase letter (because they look similar), a special height-based logic helps it figure it out.
- The backend also uses a technique called ensemble modeling, which combines results from different models to make sure predictions are as accurate as possible.
- To keep things running smoothly, the machine learning models (built with TensorFlow/Keras) are loaded only when needed, so it doesn't overload the system's memory.

When you upload an image or draw something, the backend takes care of processing it, running the predictions, and sending the text back to you in real time.

D. ChatGPT Integration – Making the App Smarter

Once the app has figured out what you've written, it doesn't stop there. With the ChatGPT integration, users can dive deeper into the recognized text and get more info in real time.
Here's what you can do with ChatGPT:

- Get a clear definition of any word or phrase.
- Ask for translations or grammar corrections.
- Learn more about names, dates, or terms, like their history or significance.
- Ask follow-up questions to better understand the content, especially helpful in an educational setting.

It's super easy to use—just click the "Ask ChatGPT" button, and the recognized text is sent to ChatGPT behind the scenes. The AI's response is then shown in a chat box right below the recognized text, making the whole process feel like a helpful, interactive conversation.

V. SOFTWARE SPECIFICATIONS

A. Frontend Interface – Intuitive, Responsive, and Built for Everyone

The app's frontend is designed to make handwriting recognition feel effortless, whether you're a student, teacher, or just someone curious about how it works. Built using React.js, the interface is fast, interactive, and easy to navigate—even if you're not super tech-savvy.
Here's what makes it user-friendly:

- Draw or Upload: You can write directly on a digital canvas using your finger, stylus, or mouse—or simply upload an image of your handwritten text, like a photo or scan.
- Instant Results: As soon as your handwriting is processed, the recognized text appears immediately on the screen. No extra steps, no delays.
- Responsive Design: With Tailwind CSS, the layout adapts beautifully to any device—whether you're on a phone, tablet, or laptop. It stays clean, simple, and functional wherever you are.

The goal is to make handwriting recognition as natural as writing on paper, with a modern digital twist.

B. Backend Processing – Where the Real Magic Happens

While the frontend keeps things simple for users, the backend quietly handles all the complex stuff. It's built using Python and Django, which means it's stable, scalable, and great at managing data behind the scenes.

Here's how it works:

- Taking In Your Input: Whether you draw on the canvas or upload an image, the app sends that handwriting to the backend through a secure API call.
- Reading Your Handwriting: The backend loads a pre-trained machine learning model—trained on thousands of handwriting samples—so it knows how to recognize letters and numbers. It's smart enough to figure out even messy handwriting.
- Understanding Details: A special logic layer looks at the height of each letter to decide if it's uppercase or lowercase, which helps when characters look similar.

The backend also powers two key API endpoints:

- **/predict/** – This is where your image gets analyzed and converted into text.
- **/info/** – This sends the recognized text to ChatGPT, which can give you extra info like definitions, explanations, or context.

C. Machine Learning Model – The Brain Behind the Recognition

At the heart of the app is a smart machine learning model designed to understand handwriting—even when it's messy or inconsistent. This model uses a powerful combination of CNN (to detect shapes and patterns) and LSTM (to understand the flow and sequence of characters), making it well-suited for interpreting handwritten text just like a human might.

It's trained on the EMNIST dataset, which contains thousands of examples of handwritten letters and numbers. Thanks to this training, the model can accurately recognize:

- Digits from 0 to 9
- Uppercase letters (A–Z)
- Distinct lowercase letters (a–z), even those that look similar in cursive

To keep things running smoothly, the system uses tools like TensorFlow/Keras for predictions, OpenCV and NumPy to prepare and clean up the images, and Django REST Framework (DRF) to connect everything with the frontend. All of this is built right into the Django backend, so everything works together seamlessly.

D. GPT Integration – Turning Recognition into Understanding

Recognizing handwritten text is just the beginning. This app goes a step further by connecting to ChatGPT to provide smart, meaningful responses based on what was written.

After the system reads your handwriting, you can choose to ask ChatGPT for more information—like:

- What does this word mean?
- How is this phrase used?
- Are there related ideas I should know about?

This feature is especially helpful for learning, studying, or just getting quick answers. The recognized text is sent to ChatGPT using a secure API connection, and the response appears right below the output in a chat-style format.

What's more—nothing is stored. Your input isn't saved or logged, so your data stays private. Every interaction is temporary and happens in real time.

E. Lightweight and Hassle-Free – No Database Required

Unlike traditional web apps that rely on databases to store user data, this system keeps things simple. It doesn't store anything—no user accounts, no history, no uploads. Everything happens in real time and disappears after the session ends.

Here's what that means:

- You don't need to sign up or log in
- Your handwriting isn't saved anywhere
- The app is faster, easier to deploy, and respects your privacy

This design makes it ideal for quick, one-time use—whether you're testing it locally or running it on a cloud server. No setup headaches, no storage concerns—just fast, private, and reliable performance.

VI. SYSTEM ARCHITECTURE

The proposed AI-powered system comprises a React-based web interface for users to input handwritten content either by drawing on a canvas or uploading an image.This frontend communicates with a Django backend server which is integrated with a pre-trained deep learning model for handwritten text recognition. The backend also connects with the OpenAI GPT API to provide relevant information based on the recognized text.

The entire system is designed for real-time recognition and response without storing any user data, ensuring speed, privacy, and convenience.

[Fig. 1] represents the flowchart and overall interfacing diagrams of the system.

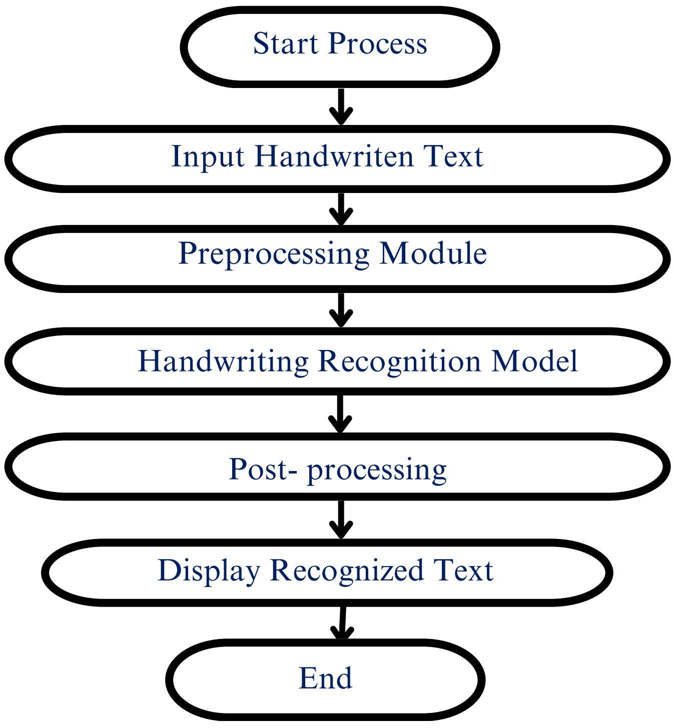

Fig. 1. Flowchart

Sysrem Flow:

A. Getting Started – User Inputs Handwriting

The process begins when a user either draws something on a digital canvas or uploads an image of handwritten text. The interface, built to be responsive and user-friendly, captures this input and sends it to the backend through an API.

B. Preprocessing & Recognition – Making Sense of the Image

Once the backend receives the image, it prepares it for analysis—cleaning it up, converting it to grayscale, resizing, and removing any unnecessary noise. Then, it runs the image through a deep learning model trained on thousands of handwritten examples (using the EMNIST dataset). This model identifies and converts the handwritten characters into digital text.

C. Fine-Tuning the Results – Smart Post-Processing

To improve accuracy, the system applies a little extra logic after recognition. It looks at the size and height of the characters to figure out tricky differences—like distinguishing between a lowercase "o" and an uppercase "O."

D. Adding Meaning with ChatGPT

After the system has recognized the text, it can do even more. It sends the extracted content to ChatGPT, which returns helpful information—such as definitions, summaries, or related concepts. It's especially useful for students or anyone trying to learn more about what they've written.

E. Final Output – Displaying the Results

Finally, the system presents everything on screen: the recognized text and the optional AI-generated explanation below it. All of this happens in real time, making the experience both fast and informative.

🛠 Technologies Used – Tools That Make It All Work

This project brings together a powerful set of modern tools and libraries to deliver a seamless user experience:

- React.js – Powers the frontend interface, enabling real-time canvas drawing and image uploads.
- Tailwind CSS – Provides clean, responsive styling for a polished, mobile-friendly layout.
- Django (Python) – Handles backend operations like image processing, model loading, and API management.
- TensorFlow/Keras – Runs the deep learning model that recognizes characters in handwritten text.
- OpenCV – Cleans and processes images before sending them to the model.
- OpenAI GPT API – Adds an intelligent layer by generating helpful definitions and explanations for the recognized text.

[Fig. 2] represents the Prediction of the word written that is "DIODE".

And [Fig. 3] represents the information retrieved.

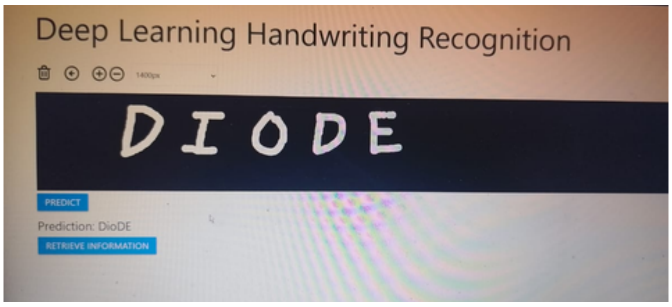

Fig. 2. Prediction Successful

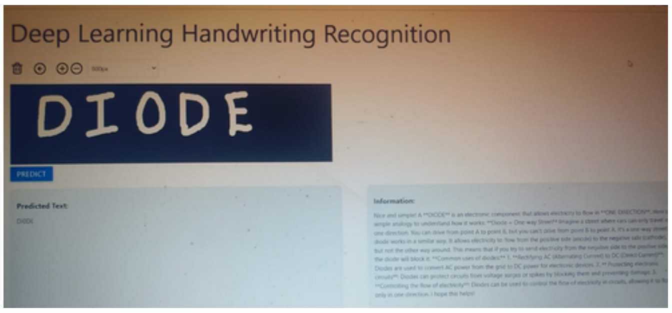

Fig. 3. Information Retrieval

Information:

Nice and simplel A "DIODE is an electronic component that allows electricity to flow in "ONE DIRECTION*, Here's a simple analogy to understand how it works: "Diode One-way Street* Imagine a street where cars can only travel in one direction. You can drive from point A to point B, but you can't drive from point B to point A. It's a one way street! A diode works in a similar way. It allows electricity to flow from the positive side (anode) to the negative side (cathodel but not the other way around. This means that if you try to send electricity from the negative side to the posive side the drode will block it. "Common uses of diodes 1. Rectifying AC (Alternating Current) to DC (Direct Current) Diodes are used to convert AC power from the grid to DC power for electronic devices. 2. * Protecting electronic circuits* Diodes can protect circuits from voltage surges or spikes by blocking them and preventing damage. 3. "Controlling the flow of electricity: Diodes can be used to control the flow of electricity in crcuits, allowing it to Bow only in one direction, I hope this helps!

VII. CONCLUSION

The primary goal of the handwriting recognition project is to provide an efficient solution for converting handwritten text into digital format. By leveraging machine learning models, the system successfully recognizes all digits, uppercase, and distinctive lowercase letters, allowing for seamless conversion. The system offers features like canvas adjustment for better text analysis, improving user experience. The integration of the ChatGPT model adds an extra layer of functionality by providing additional information related to the recognized text. This project proves to be highly valuable for scenarios requiring quick and accurate text conversion, especially in environments where digital input is needed but handwritten notes are more common. It is an effective and user-friendly tool, useful for a wide range of applications in education, business, and other domains.

VIII. Acknowledgement

WWe sincerely thank Dr. V. A. Kulkarni, Principal of KLS VDIT, Haliyal, for the opportunity to undertake this project. Our gratitude also goes to Dr. Mahendra M. Dixit, HOD of Electronics and Communication Engineering, for providing essential infrastructure and guidance. Special thanks to our Project Guide, Dr. Madhusudhan Kulkarni, for his valuable support, as well as to our faculty mentors and lab coordinators for their continuous encouragement and feedback throughout the project.

IX. REFRENCES

[1] C. Garrido-Muñoz, A. Rios-Vila, and J. Calvo-Zaragoza, “Handwritten Text Recognition: A Survey,” arXiv preprint arXiv:2502.08417, February 2025.

[2] M. Hamdan, A. Rahiche, and M. Cheriet, “HAND: Hierarchical Attention Network for Multi-Scale Handwritten Document Recognition and Layout Analysis,” arXiv preprint arXiv:2412.18981, December 2024.

[3] M. Hamdan, A. Rahiche, and M. Cheriet, “HTR-JAND: Handwritten Text Recognition with Joint Attention Network and Knowledge Distillation,” arXiv preprint arXiv:2412.18524, December 2024.

[4] Y. Li, D. Chen, T. Tang, and X. Shen, “HTR-VT: Handwritten Text Recognition with Vision Transformer,” arXiv preprint arXiv:2409.08573, September 2024.

[5] T. Constum, P. Tranouez, and T. Paquet, “DANIEL: A fast Document Attention Network for Information Extraction and Labelling of handwritten documents,” arXiv preprint arXiv:2407.09103, July 2024.

[6] F. Wolf and G. A. Fink, “Self-training for handwritten word recognition and retrieval,” International Journal on Document Analysis and Recognition (IJDAR), vol. 27, pp. 225–244, June 2024.

[7] M. Diaz, G. Crispo, A. Parziale, A. Marcelli, and M. A. Ferrer, “Writing Order Recovery in Complex and Long Static Handwriting,” arXiv preprint arXiv:2406.03194, June 2024.

[8] M. Hamdan, A. Rahiche, and M. Cheriet, “Self-training for handwritten word recognition and retrieval,” International Journal on Document Analysis and Recognition (IJDAR), vol. 27, pp. 225–244, June 2024.

[9] S. Neupane, M. Pyakurel, K. Sinha, B. Sharma, and P. A., “GraphoMatch: Forensic Handwriting Analysis Using Machine Learning,” International Journal of Science and Research Archive, vol. 11, no. 2, pp. 1526–1537, April 2024.

[10] N. Rakask and P. Kushal Redds, “A system for automating handwritten text conversion to digital form using image processing and Handwritten Text Recognition systems,” International Journal of Computer Applications, vol. 184, no. 10, pp. 25–29, March 2024.

[11] S. Rakesh, P. Kushal Reddy, V. Prashanth, and K. Srinath Reddy, “Handwritten text recognition using deep learning techniques: A survey,” MATEC Web of Conferences, vol. 392, Article 01126, March 2024.

[12] B. Balci, D. Suadati, and G. I. Nikitha A, “Comparison of direct word classification and character segmentation approaches for Handwritten Text Recognition using LSTM,” Journal of Pattern Recognition and AI, vol. 22, no. 4, pp. 75–83, January 2024.

[13] H. Bhardwaj and R. Das, “Improving Handwritten Text Recognition accuracy using a Deep Learning model based on Long Short Term Memory (LSTM),” Proceedings of the IEEE Conference on Computational Intelligence, pp. 112–118, December 2023.

[14] N. A. Nikitha and G. J. Geetha, “Handwritten Text Recognition using Deep Learning and LSTM with word-level recognition,” arXiv preprint arXiv:2310.04567, October 2023.

[15] F. Kizilirmak and B. Yanikoglu, “CNN-BiLSTM model for English Handwriting Recognition: Comprehensive Evaluation on the IAM Dataset,” arXiv preprint arXiv:2307.00664, July 2023.