

Whiteboard Text Recognition and Note Saving System

Hari Chandana B.¹, Avula Adithya Babu², Boya Udayasree², Tamidepati Venkata Vindya Sai²,
Kottam Jagadesh² and Ke Eswar Goud²

¹Department of ECE, Mohan Babu University erstwhile Sree Vidyanikethan Engineering College, Tirupati, Andra Pradesh, India

²Department of ECE, Sree Vidyanikethan Engineering College, Tirupati, Andra Pradesh, India

Keywords: OpenCV, Tesseract OCR (Optical Character Recognition).

Abstract: The use of modern technologies makes Whiteboard Text Recognition and Note Saving System to effectively take handwritten notepad of whiteboard text to identify and save. It works by taking a picture of a white board using a phone or camera, and uploading it for processing. The detected image is then improved using edge detection, contrast, and noise removal algorithms through OpenCV— to better isolate portions of readable text. With the contour detection and bounding box techniques, we are able to distinguish which RAW data inside an image contains handwritten textual data that is then detected and segmented using OpenCV algorithms. The extracted image segments are then passed into a TensorFlow based (OCR) model that accurately predicts the handwritten characters. Postprocessing with formatting adjustments used to make the recognized text more readable and accurate. The processed text is then saved in an ordered fashion into the database, making it easily accessible for taking notes and retrieval. The user-friendly dashboard allows for viewing, editing, and sharing of the digital notes, facilitating collaboration and sharing in both educational and professional settings. The app converts handwriting from a whiteboard into a digital format that aid knowledge sharing and collaboration. OpenCV is used for image preprocessing and TensorFlow for recognizing words to achieve it.

1 INTRODUCTION

It is an experience data has evolved; the way we collect, organize and share information” has changed with the rapid development of digital tech. Conventional whiteboards have not gone out of style in the manual writing of notes in education and collaborative spaces. Yet, it has downsides: risk of losing important notes, inefficiency in sharing info and organizations, etc. To overcome these challenges, we present a novel system of whiteboard text recognition and note-saving, using optical character recognition (OCR) technology, to effectively convert handwritten notes to a digital format.

With cross-industry demands for efficient information management in professional and academic settings, the process of quickly digitizing and storing whiteboard notes can recognize significant productivity trade-offs. As hybrid models of working, learning and collaborating become the new normal for education and workplaces across the world, this technology brings instant access to stored

content, while enabling instant sharing and archiving of key information. Our solution uses computer vision to crop the text from a live lecture video with an incredibly low false positive rate. We achieve this with high recognition accuracy and adaptability to different handwriting styles using modern OCR techniques. The note creation and uploading system has an intuitive user interface that can be easily utilized by a wide variety of users.

This study is about the design, implementation, performance and real-life use of the system. Depending on the need, this means we can have a more analysis-oriented experience on the success of our system, e.g. we can say that our means to this end, a more systematic and organized approach towards note taking, is the achieved fit.

2 LITERATURE SURVEY

This paper, titled "Handwritten Text Recognition using Deep Learning with TensorFlow" written by Sri

Yugandhar Manchala and students of Aditya Institute of Technology and Management, describes the implementation of the TensorFlow package for neural networks (CNN and RNN) in combination with CTC layers in order to create a reliable handwritten OCR system. This model trained on the IAM Handwriting Database exceeds 90.3% accuracy while efficiently accommodating the diversity of handwritten input and working best on low-noise images. We use a CNN-RNN architecture for the feature extraction and sequence learning modules, supplemented by image preprocessing, data loading, model management and system integration Python modules. As an additional help in the outputted text produced by the text on the system, at the centre lies the python spell-checker that processes the recognised text. It is scalable to increased performance when larger datasets are introduced, and can be adapted to full-line text recognition.

Nikitha A, Geetha J Dr, and JayaLakshmi D.S Dr describe a handwritten text recognition (HTR) system using deep learning techniques based on their work "Handwritten Text Recognition using Deep Learning" in their paper from Ramaiah Institute of Technology. It employs 2D-CNN and Long Short-Term Memory (LSTM) networks (concretely 2D-LSTM) to boost the accuracy of recognition at the word level. Trained on the IAM Handwriting Dataset, the model achieves 8.2% Character Error Rate (CER) and 27.5% Word Error Rate (WER) with a accuracy of 94% HTR combines OCR to convert handwritten text to digital text and the design leverages pre-processing techniques including noise reduction, segmentation, and binarization so that the text has optimal image quality. This configuration enhances document durability, storage efficiency, and text accessibility useful properties in the context of archiving and retrieval. Its flexible architecture allows for future scaling up and cross-language capabilities, making it a wonderful candidate for automated text digitisation

3 PROPOSED METHOD

3.1 System Overview

Figure 1 illustrates the architecture. Video Input Module: The video input module receives frames from a live camera stream or an input video file. It acts as a gateway which feeds raw video data for processing. The frames are extracted in a sequence so that the flow is uniform for analysis. By providing a

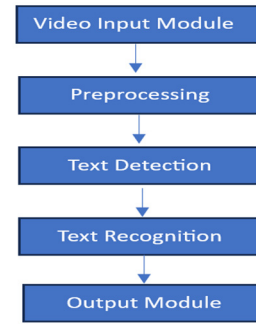


Figure 1: System Architecture.

continuous flow of visual data to the system, this module paves the way for the next few stages.

Preprocessing: The sharpness of shown Video Frames, again enhances detection and recognition accuracy. To make computations faster, frames are converted to grey scale which discards unnecessary information (color). Gaussian or median noise reduction filters will clean the frames of distortion. Text regions are enhanced using contrast enhancement which further increases the accuracy of the characters.

Text Detection: The process of finding areas in every frame that contain text, which helps us find regions of interest. The text sections are fragmented via edge-detection, bounding box, and neural net techniques. This act of focusing on certain locations leaves behind unnecessary data in the background. This improvement expedites and enhances the subsequent phase of text recognition.

Text Recognition: Text recognition utilizes Optical Character Recognition (OCR) to recognize the localized text areas determined in the previous step. The system goes through each region and turns any recognised characters into machine readable text. This stage transforms the pixel data into a defined text structure, human-readable and eligible for further processing to the next stages.

Output element: Once the text has been detected, the output module displays or saves it for practical consumption. It also generates editable text files such as for real-time applications. txt or. csv files respectively, or provide a real-time view. This step enables the completion of pipelines and allows to perform data analysis, documentation, workflow integration, etc.

3.2 Video Input Module

The video input module reads frames from a live video feed, which is the primary source of data for the text recognition system. This module provides a

continuous stream from a connected camera (if available) or from a video file if it is provided through the boot pack. The module keeps track of parameters such as frame rate, resolution and video format to achieve optimal frame quality and mitigate processing delays. By normalizing these inputs, the Video Input Module helps standardize each frame for the subsequent stages of the pipeline, making processing uniform and efficient in real-time applications.

3.3 Preprocessing

The pre-processing level processes each video frame to enhance the precision of text detection and recognition.

Grayscale Conversion: The process of grayscale conversion simplifies each video frame to a single channel that contains intensity values and removes all colour components. To do this, the algorithm enhances and removes noise from the output image, making it easy to process the image for text detection and also less expensive computationally. All grayscale retains is the hue and saturation of text relative to the background the stuff necessary for mattering text. Without colour there is a reduced possibility of you making a mistake or your eye straying to something else in the same frame. It simplifies the process in a way that provides better detection accuracy by driving the system to pay attention to the structural elements of a text.

Denoising: Denoising is one of the basic preprocessing step which users do to remove unwanted components or visual noise from a video frame. Image smoothing is achieved by methods like Gaussian and median filters which reduce high-frequency noise and does not hide important image details such as edges. Gaussian filters use a weighted average of the surrounding pixel values across a region, producing a smoothening effect that eliminates any pixel noise without distorting text. Median filters on the other hand are designed to remove outlier types of noise such as speckle noise and salt-and-pepper noise by taking a median value of each pixel with its neighbours. As a result, the performance of text recognition algorithms is improved in the denoising process, leaving the text sharp and clear. Figure 2 and 3 gives the information of noise in the image and noise eliminated image.

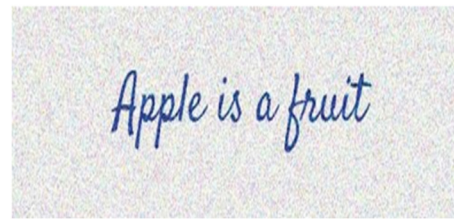


Figure 2: Noise in the Image.



Figure 3: Noise Being Eliminated from the Input Image.

3.4 Text Detection

After pre-processing, the image is passed to: Tesseract OCR to read the text, at which point the content analysis begins. Now, for each individual character in the image we will use Tesseract's `image_to_boxes` method. It calculates bounding boxes around each character to identify areas that contain text using each identified character's coordinates (x, y, width, and height). These bounding boxes in figure 4 make it easier to verify and debug the detection process by visually showing which regions of the image are likely to contain the text. Figure 4 is the input text image.



Figure 4: Input Image Having Text.



Figure 5: Text Detection from Input Image.

3.5 Text Recognition

Text recognition is the stage of recovering the text from an image after the text detection phase. At this step, Optical Character Recognition (OCR) methods like Tesseract OCR are applied to identify and transform the detected text regions to machine-readable text. Once the text has been recognized and boxed, the next step is to find out what characters, words or phrases are inside those boxes. These

algorithms analyse the pixel patterns of the image. Once the text is detected and recognized, the spell-checking algorithms help it figure out if there are any errors or if any words have been misidentified. This is a crucial step for guaranteeing the accuracy of the extracted text. There are multiple techniques to achieve this such as dictionary-based techniques, context-aware models, machine learning-based techniques, etc. The document is converted into a structured text-based as shown in figure 7 editable format. This text can then be processed or leveraged in applications such as document digitization, data extraction, and organization. Figure 6 is the hand-written image.



Figure 6: Input Handwritten Text.

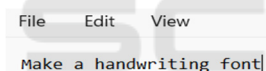


Figure 7: Output Recognized Text.

3.6 Output Module

In a text recognition system, a text file as shown in figure 9 is generated after the output module transforms raw text returned from OCR to some structured text format which is editable, organized, and ready for post-processing. It transforms the text into a standardized format by correcting errors, formatting it for maximum clarity, and saving it in widely supported formats like .txt or .csv. The structure of the output allows it to be interacted with easily by data analysis, indexing, and automated workflow technologies so the text may be searched and retrieved simply. Incorporating metadata and allowing for spell corrections make for better quality and more useful text for applications like document digitization, research, data mining, and legal record-

keeping. As a last note, this phase makes sure the detected text is dynamic not static. Figure 8 is the input of text image for recognition.



Figure 8: Input Image for Text Recognition.

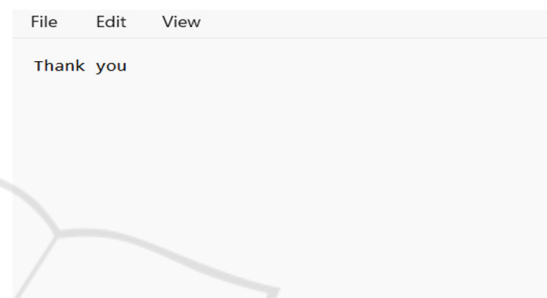


Figure 9: Output Recognized Text Saved in Notepad.

4 RESULTS AND DISCUSSION

So, the final goal of this project is to get text from live lectures recording using OpenCV and Tesseract OCR. Starting with the OpenCV recording of individual frames from the live video stream, it follows a sequence of preprocessing steps to enhance visibility of the text and to maximize the accuracy of text recognition. These processes include grayscale conversion that reduces image complexity by removing color information and noise reduction methods such as Gaussian blur that remove any visual distortion and pollution from the background. After preprocessing, the frame is passed to the text recognition stage, where recognition and extraction of the text in the image happens using Tesseract OCR. The recognized text is subsequently machine-learning-auto-corrected to fix any recognition issues such as misinterpreted/similar to real character or word. All these techniques together allow the system to extract the clean, correct text from a live lecture video no matter what handwriting style, size, or orientation it is.

5 CONCLUSIONS

Finally, this paper introduces a very efficient method for text identification and recognition, which uses newly developed image processing algorithms and OCR tools like Tesseract. The process starts with input image pre-processing to enhance the quality. Text detection is then performed to determine the areas that contain text, and finally, text recognition extracts the text using optical character recognition methods. The presented approach precisely identifies and extracts text from different picture sources which can be purposeful in document digitalization, data extraction, and automated content analysis.

REFERENCES

- A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in ICML, 2006.
- A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in NIPS, 2009.
- A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," PAMI, vol. 31, no. 5, pp. 855–868, 2009.
- D. Keysers, T. Deselaers, H. A. Rowley, L. Wang, and V. Carbune, "Multi-language online handwriting recognition," PAMI, vol. 39, no. 6, pp. 1180–1194, 2017.
- D. Castro, B. L. D. Bezerra, and M. Valena, "Boosting the deep multidimensional long-short-term memory network for handwritten recognition systems," in ICFHR, 2018.
- Gideon Maillette de Buy Wenniger; Lambert Schomaker; Andy Way, "No Padding Please: Efficient Neural Handwriting Recognition", in ICDAR 2019.
- Herleen Kour; Naveen Kumar Gondhi, "Machine Learning approaches for Nastaliq style Urdu handwritten recognition: A survey" in ICACCS, 2020.
- J. Puigcerver, "Are multidimensional recurrent layers really necessary for handwritten text recognition?" in ICDAR, 2017.
- J. Walker, Y. Fujii, and A. C. Popat, "A web-based ocr service for documents," in DAS, Apr 2018, pp. 21–22.
- M. Kozielski, P. Doetsch, and H. Ney, "Improvements in rwth's system for off-line handwriting recognition," in ICDAR, 2013.
- Nikitha A, Geetha J, and JayaLakshmi D.S, "Handwritten Text Recognition using Deep Learning," in 2020 5th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT-2020), Bangalore, India, Nov. 2020, pp. 388–392, doi: 10.1109/RTEICT49044.2020.9315679
- P. Doetsch, M. Kozielski, and H. Ney, "Fast and robust training of recurrent neural networks for offline handwriting recognition," in ICFHR, 2014.
- P. Voigtlaender, P. Doetsch, S. Wiesler, R. Schlter, and H. Ney, "Sequence-discriminative training of recurrent neural networks," in ICASSP, 2015
- P. Voigtlaender, P. Doetsch, and H. Ney, "Handwriting recognition with large multidimensional long short-term memory recurrent neural networks," in ICFHR, 2016.
- Rohan Vaidya; Darshan Trivedi; Sagar Satra; Prof. Mrunalini Pimpale, "Handwritten Character Recognition Using Deep-Learning", in ICICCT, 2018.
- S. Ghosh and A. Joshi, "Text entry in indian languages on mobile: User perspectives," in India HCI, 2014.
- S. Y. Manchala, J. Kinthali, K. Kotha, K. S. Kumar, and J. Jayalaxmi, "Handwritten Text Recognition using Deep Learning with TensorFlow," International Journal of Engineering Research & Technology (IJERT), vol. 9, no. 5, pp. 594–600, May 2020.
- Shahbaz Hassan, Ayesha Irfan, Ali Mirza, Imran Siddiqi, "Cursive Handwritten Text Recognition using Bi-Directional LSTMs: A case study on Urdu Handwriting", in Deep-ML, 2019.
- T. Bluche and R. Messina, "Gated convolutional recurrent neural networks for multilingual handwriting recognition," in ICDAR, vol. 01, 2017.
- V. Pham, T. Bluche, C. Kermorvant, and J. Louradour, "Dropout improves recurrent neural networks for handwriting recognition," in ICFHR, 2014.
- V. Carbune, P. Gonnet, T. Deselaers, H. A. Rowley, A. Daryin, M. Calvo, L.-L. Wang, D. Keysers, S. Feuz, and P. Gervais, "Fast multi-language lstm-based online handwriting recognition," ArXiv, 2019.