DOI https://doi.org/10.30525/978-9934-26-597-6-26

TEXT RECOGNITION IN CHALLENGING CONDITIONS USING NEURAL NETWORK-BASED COMPUTER VISION ALGORITHMS

Khotunov Vladyslav*, Lutsenko Maksym, Marchenko Stanislav
Cherkasy State Business-College, Ukraine
**Corresponding author's e-mail: vkhotunov@gmail.com, sv.marchenko1989@gmail.com*

Abstract

This paper presents the research and development of a neural network-based text recognition system tailored for varying image quality and lighting conditions. The study investigates the architecture, training process, and implementation of convolutional and recurrent neural networks (CRNNs) for recognizing individual characters in raster images. Using a combination of CRAFT for region detection and a custom classifier network for symbol recognition, the system demonstrates 83.87% accuracy on synthetic datasets. The methodology includes advanced preprocessing, regularization techniques, optimization strategies, and integration with AimStack for training visualization. Results suggest high applicability for educational use and potential integration in broader AI-driven document processing systems.

Keywords: neural network, computer vision, text recognition, CRAFT, CRNN, OCR, image processing

Traditional OCR engines often rely on deterministic algorithms that fail under dynamic environmental conditions such as non-uniform lighting, perspective distortions, and diverse font styles. The modern shift towards data-driven approaches, particularly deep learning, allows systems to learn hierarchical features from raw image data, making them resilient to such variability. By using annotated data, these systems learn to generalize better to unseen conditions, which is crucial in practical deployment scenarios like mobile scanning apps, document digitization pipelines, and accessibility tools for the visually impaired.

Moreover, the increasing availability of pretrained models and open datasets has lowered the entry barrier for developing custom OCR systems. However, there remains a need for lightweight, modular, and customizable architectures that educational institutions and small-scale developers can use without requiring extensive computational resources. The presented system

directly addresses this gap by providing both a robust recognition pipeline and a pedagogically-oriented design.

Recent advancements in neural networks have significantly transformed the capabilities of computer vision. Optical Character Recognition (OCR) systems now extend beyond scanned black-and-white documents to include dynamic, real-world environments. However, accurate recognition remains a challenge when images suffer from poor lighting, blur, or low resolution. The aim of this work is to develop a modular, extensible system that not only detects but also classifies text from complex visual contexts using deep learning techniques.

To further validate the practical utility of the developed system, a series of real-world images were collected using smartphone cameras under various conditions–natural daylight, indoor lighting, low-light scenarios, and with background clutter. These images were used to evaluate system robustness and the ability to generalize beyond the synthetic dataset. The system maintained an average accuracy of 78–81% on these untrained images, which is promising given the challenging conditions and absence of fine-tuning.

Additionally, an ablation study was conducted to quantify the contribution of each architectural component. Removing dropout, normalization, or specific convolutional layers resulted in consistent accuracy drops of 5–12%, underscoring the importance of the chosen architecture. Visual inspection of feature maps and activation layers using tools such as TensorBoard further confirmed that meaningful character representations were learned at intermediate network stages.

To improve user experience, a graphical user interface (GUI) prototype was developed using Tkinter and PyQt, enabling real-time image selection, detection preview, and export of recognized text. This version can be used as an educational demonstration tool in machine learning courses or for small business applications like receipt scanning, ID processing, and printed form digitization. Localization support was also partially implemented, preparing the system for multilingual expansion in future iterations.

To enhance the generalization capacity of the recognition model, extensive experimentation was conducted with various convolutional layer depths, kernel sizes, and activation functions. Among them, ReLU and LeakyReLU demonstrated superior performance, particularly when combined with batch normalization layers. Dropout layers with a rate of 0.3–0.5 were inserted between dense layers to prevent overfitting, especially during later stages of training. The system also incorporated cyclic learning rate scheduling, which helped prevent local minima during convergence.

Evaluation of the character classification model included precision, recall, and F1-score metrics for each character class. Misclassified instances were visualized, and confusion matrices were analyzed. It was found that similar glyphs such as 'O' and '0', or 'I' and 'l', were frequent sources of errors. To mitigate this, specific augmentation techniques were applied–rotational jitter, affine transformations, and controlled noise injection to improve robustness. Data augmentation not only increased dataset size virtually but also enabled learning of spatial invariance.

The CRAFT model was fine-tuned to detect characters individually, rather than word-level bounding boxes. It allowed for integration with the classifier component, resulting in a seamless end-to-end system. Intermediate outputs from CRAFT, such as character heatmaps and affinity scores, were visualized for debugging and used for adjusting the confidence threshold. This flexible thresholding mechanism permitted trade-offs between recall and precision based on specific application needs.

An additional module was implemented to evaluate word-level recognition by aligning character predictions using Levenshtein distance. This helped in scoring the overall quality of text output from the system. While no language model was integrated in this prototype, its modular architecture enables future incorporation of spelling correction or n-gram language models for higher-level semantic correction. The processing pipeline can also be integrated into web or mobile applications via REST API endpoints built with FastAPI.

The system architecture consists of two primary components: a region detection model based on CRAFT, and a classification model for recognizing symbols. The CRAFT model extracts bounding boxes around potential text elements, while the classifier processes cropped image segments to determine character labels.

The development involved:

– Analysis of classical OCR methods vs. deep learning approaches (Tesseract, EAST, FOTS);

– Comparative study of CRNNs, CNNs, and RNNs for character-level accuracy;

– Implementation of dataset generation tools (TextRecognitionDataGenerator);

– Use of PyTorch and OpenCV for model development and image processing;

– Console-based interface with configurable modes (train, test, inference).

The training dataset includes 6200 labeled characters (upper/lowercase Latin letters and digits), generated synthetically with controlled augmentation.

The character recognition network features multiple convolution-pooling layers followed by dense layers with dropout and batch normalization. Optimization involved adaptive learning rate reduction, weight decay, and momentum strategies.

Model training and evaluation were tracked using AimStack. Training took place over several epochs, with validation performed on a separate dataset. Key issues, such as the similarity of characters like "m" and "T", were addressed through geometric augmentation. CRAFT's pretrained weights on ICDAR datasets were used, ensuring compatibility with real-world text variations. During inference, results were returned in editable text format, with optional image overlays for visual validation.

The trained system provides the following features:

– Multi-mode operation: training, evaluation, live prediction;

– 83.87% recognition accuracy in noisy test environments;

– Modular code architecture suitable for educational use; – Adjustable preprocessing pipeline for generalization;

– Graphical visualization of detected regions and training metrics.

The system was developed using Agile methodology, enabling iterative improvements in response to observed limitations. Educational focus informed the software design, emphasizing clarity, usability, and adaptability. Python and VSCode served as the main development environment, with Git version control and Docker support for reproducibility.

Conclusions

The architecture and implementation of this OCR system demonstrate a balance between model complexity and practical usability. Through rigorous evaluation and iterative design, the software showcases how advanced AI methods can be distilled into accessible tools. In particular, the layered modularity of the system–separating detection, classification, and evaluation–provides flexibility for learners and developers alike.

Looking ahead, expanding the dataset to include non-Latin characters, handwritten text, and multilingual scripts could significantly broaden the system’s utility. Incorporating temporal models, such as attention-based transformers, may allow for recognition of cursive and flowing text, while also reducing the reliance on precise character segmentation. Ultimately, the project's value lies not only in its current performance but in its potential as a foundation for educational and applied research in OCR.

The developed software solution successfully integrates deep learning methods for symbol detection and classification in raster images. The approach proves effective for text recognition under diverse conditions. Although accuracy remains below state-of-the-art multilingual OCR engines, the modular design and focused dataset strategy make this system well-suited for instructional settings and further R&D. Future improvements may involve

integrating CRNN with sequence prediction for full word-level detection, adding multilingual datasets, and implementing lightweight deployment on mobile or embedded devices.

References

1. IBM. What is Computer Vision? https://www.ibm.com/topics/computer-vision

2. IBM. What is Machine Learning? https://www.ibm.com/topics/machine-learning

3. Ravil I. Mukhamediev, Yelena Popova, Viktors Gopejenko (2022) Review of Artificial Intelligence and Machine Learning Technologies: Classification, Restrictions, Opportunities and Challenges Mathematics 2022, 10, 2552. https://doi.org/10.3390/math10152552