

Text Extraction and Audio Conversion for Assistive Technology Using OCR Models

Quinn Dao and Preethi Raghuraman

December 13, 2024

Abstract

Text detection and recognition from images is a technology with applications ranging from document digitization to assistive technologies for visually impaired individuals. This project uses state-of-the-art Optical Character Recognition (OCR) models to extract text from images and convert it into audio. The system is trained using the MJSynth dataset and the performance of three OCR approaches, Google Tesseract OCR, EasyOCR, and Keras OCR, are compared. These models are evaluated on metrics such as precision, recall, and F1-score. To enhance accessibility, the detected text is converted into audio using a Python text-to-speech (TTS) library, pyttsx3. The system is designed to assist visually impaired individuals by enabling them to "read" text from images through audio feedback.

1 Introduction

Computer vision, a subfield of Artificial Intelligence (AI), enables machines to see, observe, and make sense of digital images, videos, and other visual inputs. Extracting text from images and documents manually can be very tedious and time-consuming. A technology that extracts text from images into editable and searchable text files is known as Optical Character Recognition (OCR) [3]. This technology helps us convert various kinds of input documents into machine-processable data.

Visually impaired individuals face challenges accessing text embedded in everyday images such as signs, product labels, and documents. Current screen readers cannot interpret text within images, limiting access to important information. We propose to develop a machine-learning model that converts text found in images into audio, enabling visually impaired users to hear the content of these images.

This project aims to develop a text-to-audio conversion system using Optical Character Recognition (OCR) and Text-to-Speech (TTS) technologies. The system will assist visually impaired individuals by converting text from images

(e.g., signs, documents) into audible speech, enhancing their ability to access and interact with text in their environment. We will employ machine learning models, integrating OCR for text detection and recognition, followed by TTS for audio output. The primary goal is to deliver a working prototype capable of processing image-based text.

2 Methodology

2.1 Dataset

We used the MJSynth dataset (also known as SynthText), a synthetic dataset designed for training models for text detection and end-to-end text recognition in natural scenes. It consists of 800,000 images, each containing multiple instances of text, and detailed annotations, with bounding boxes for both individual words and characters, ensuring precise localization for training purposes. The background images, sourced from Google Image Search, were carefully chosen from a collection of 8,000 real-world scenes. The text content itself is drawn from the Newsgroup20 dataset, which provides a diverse mix of words, lines, and paragraphs, offering a rich variety of text styles and structures. These features make the chosen data set a versatile and valuable tool for advancing the capabilities of text recognition systems [8].

2.2 OCR Models Used

2.2.1 Google Tesseract OCR

The original Tesseract OCR engine is an open-source optical character recognition engine developed by Hewlett-Packard in 1994 and released in 2005, and Google later took over development [4]. The engine works on images of text where the text regions are already identified and does not include the feature to determine where the text is present on the image. The engine analyses the images to identify the shapes and outlines of the components (which are the shapes in the image). These identified components are grouped into “Blobs” to form letters or characters, which are then arranged into lines and analyzed to see if the text uses fixed spacing or proportional spacing [9].

For text recognition, the legacy Tesseract uses a two-pass recognition process. In the first step, the engine tries to recognize the text word by word. When a word is successfully identified, this word is used for adaptive learning. The adaptive classifier learns from the recognized words and improves its understanding of the text style, font, or character shapes specific to that page. This helps it recognize words more accurately as it moves down the page. After processing the entire page once, the engine goes for a second pass where it re-checks the words it could not recognize during the first pass. In the final phase, Tesseract resolves any remaining issues, for example, the fuzzy spacing between words, and checks for capital letters by analyzing different possibilities for the height of the characters [9].

In 2018, Tesseract 4.0 was released to the public. Tesseract 4.0, uses an LSTM-based OCR engine for line-by-line text recognition and supports language recognition for more than 100 languages [11]. This was the beginning of the use of neural networks. This was a major breakthrough for text detection and text recognition. The LSTM engine executes light image preprocessing such as thresholding, noise removal, and deskewing to improve image quality [12]. Using convolution layers, critical features of the text, such as edges and textures are extracted. Those features are then fed into LSTM layers to predict character sequences. In other words, the LSTM architecture attempts to recognize words from the text image. Post-processing is then executed for spell-checking and context analysis [10].

In 2021, the OCR engine was updated to Tesseract 5.0, the most recent model. The new engine improved recognition performance, enhanced image preprocessing, and optimized for memory consumption [11].

2.2.2 EasyOCR

EasyOCR is implemented using Python and the PyTorch library which supports 80+ languages. It consists of 3 components [13]:

1. Feature extraction - In this step, raw input images are converted to meaningful features which are later used for text recognition. Models such as ResNet (Residual Networks) and VGG (Visual Geometry Group) are used.
2. Sequence labelling - After extracting the features, EasyOCR processes them sequentially using Long Short Term Memory (LSTM) in this step to interpret the content of the text in the image.
3. Decoding - The sequences from the LSTM are decoded into readable text where duplicate characters are removed, ensuring that the final output is meaningful and clear.

We imported the EasyOCR library. Then, we created and initialized the EasyOCR reader object for English ('en') text recognition. The **readtext** method of the reader object reads the image and returns the recognized text. The output is in the form of a tuple consisting of the coordinates of the bounding box, the recognized text and the confidence score [5].

2.2.3 Keras-OCR

Keras-OCR is a deep learning-based OCR library built using Keras and TensorFlow. It is an entire pipeline that can read images with text, determine the location of the text, recognize the words in the text, and generate the final output. It consists of two separate models: one for text detection and one for text recognition [6]. The text detection model is the Character-Region Awareness For Text detection (CRAFT) model. This model was first developed by Clova AI Research of Naver Corporation [1]. Its backbone consists of the VGG-16 convolutional neural network (CNN), a very popular image-recognition CNN [7].

Using a CNN, the model generates region and affinity scores. The region score determines what the localized regions of an image that likely pertains to a character. The affinity score evaluates the likelihood of characters to appear as a single word. These two scores determine the region of text in an image [1].

Once the localized text of an image has been extracted, the text image is fed into the text recognition component of the Keras-OCR engine. A convolutional recurrent neural network (CRNN) model executes text recognition [6]. As described by Qiang et al. [2], CRNNs have the capability to recognize text very well. It generates a temporal memory through LSTM hidden blocks for text segmentation. Hence, the Keras-OCR engine adopted a CRNN model for text recognition.

The Keras-OCR library includes pre-trained models and an end-to-end training pipeline that facilitates the recognition of text in various images. Upon importing the `keras_ocr` library and initializing the pipeline, the `read()` function is employed to load images from specified paths and prepare them for processing. The `pipeline()` function then detects and extracts the text regions, generating the recognized text along with the coordinates of the bounding box as a tuple.

2.3 Conversion of text to Audio using pyttsx3

Text-to-speech technology converts the generated string of text into artificially produced human speech. There are numerous libraries to perform this such as Google Text-to-Speech, Amazon Polly etc. We used `pyttsx3` (Python Text-to-Speech version 3), a text-to-speech conversion Python library, which can work offline. We installed the `pyttsx3` library and initialized the text-to-speech engine. Prediction results from the best performing OCR engine were then fed into the text-to-speech engine to return audio output.

3 Performance Evaluation

The following metrics are used to evaluate the model performance:

1. Precision

Precision is defined as the ratio of True Positives (TP) to the sum of True Positives and False Positives (FP). It is given by the formula:



$$\text{Precision} = \frac{TP}{TP + FP}$$

This metric measures the accuracy of the positive predictions.

2. Recall or True Positive Rate

Recall, also known as the True Positive rate, is the ratio of True Positives to the sum of True Positives and False Negatives (FN). It is calculated as:

$$\text{Recall} = \frac{TP}{TP + FN}$$

			
Legacy Tesseract Engine:	yearbooks	Legacy Tesseract Engine:	transcends.
Tesseract 4.0 LSTM Engine:	yearbooks	Tesseract 4.0 LSTM Engine:	transcend
Tesseract 5.0 LSTM Engine:	yearbooks	Tesseract 5.0 LSTM Engine:	transcen7d-
EasyOCR Engine:	yearbooks	EasyOCR Engine:	transcends
Keras OCR Engine:	yearbooks	Keras OCR Engine:	transcends

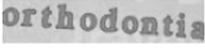

			
Legacy Tesseract Engine:	orthodonti.	Legacy Tesseract Engine:	lovelieb
Tesseract 4.0 LSTM Engine:	orthodontig	Tesseract 4.0 LSTM Engine:	lovelier
Tesseract 5.0 LSTM Engine:	orthodontia	Tesseract 5.0 LSTM Engine:	lovelier
EasyOCR Engine:	orthodontia	EasyOCR Engine:	loveliel
Keras OCR Engine:	orthodontia	Keras OCR Engine:	lovelier

Figure 1: Some output examples from the OCR models

Recall measures the ability of the model to find all the relevant cases (all actual positives).

3. F1 Score

The F1 Score is the harmonic mean of Precision and Recall, providing a balance between them. It is particularly useful when the class distribution is uneven. The F1 Score is calculated using the following formula:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

A higher F1 Score indicates better overall performance, with a perfect score of 1 signifying that the model correctly identifies all relevant positive cases while minimizing false positives.

We compared three OCR models to evaluate their text recognition performance on the MJSynth dataset. The analysis included both traditional and modern OCR models: Tesseract’s legacy and LSTM-based versions, EasyOCR, and Keras OCR.

The legacy Tesseract model exhibited limited performance, achieving an accuracy of 29.8%. This model correctly predicted 596 instances, with a notably high number of incorrect predictions (1404). The character-level precision, recall, and F1-score were 0.63, 0.62, and 0.62 respectively, indicating that this model struggles to accurately recognize characters in the dataset.

The introduction of LSTM architecture in Tesseract brought significant improvements. With an accuracy of 47.7%, this model correctly predicted 954 instances while making 1046 incorrect predictions. Both precision and recall improved to 0.73 and 0.70 respectively, resulting in an F1-score of 0.71. The improvement in accuracy by nearly 18 percentage points illustrates the benefits of integrating

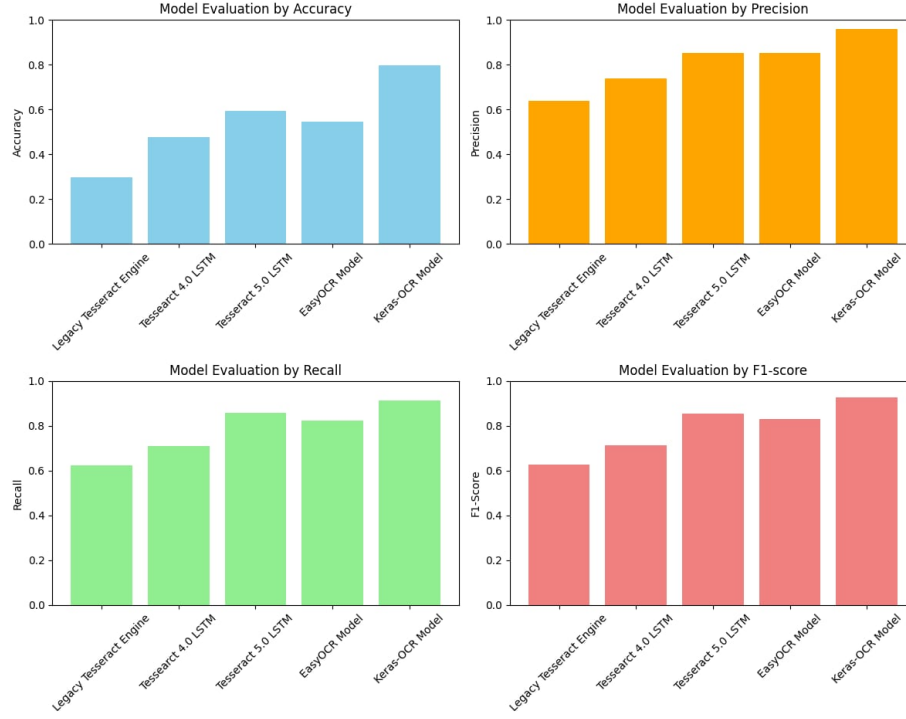


Figure 2: Different evaluation metrics from different models

LSTM technology, which enhanced the model’s ability to discern and learn from the text’s patterns more effectively.

The best performing Tesseract model, utilizing refined LSTM architectures, achieved a slightly higher accuracy of 59.4%, with 1,118 correct predictions and 812 incorrect ones. The precision, recall, and F1-score rose marginally to 0.78, demonstrating a more robust understanding of character sequences and reflecting the value of leveraging sequential learning for text recognition tasks. This represents an additional accuracy gain of approximately 12 percentage points over the older LSTM-based model, showcasing the ongoing improvements in OCR technology.

EasyOCR provided a balanced performance with an accuracy of 54.7%. It correctly predicted 1094 instances while making 906 errors. The character-wise precision being 0.85, recall being 0.82, and F1-score at 0.82, indicate that while EasyOCR performs decently, it slightly lags behind the best LSTM Tesseract model in terms of overall accuracy but excels in precision and recall.

Keras OCR stood out as the best performer in this evaluation, achieving a strong accuracy of 79.6%, with 1593 correct predictions and only 407 errors. Its character-level precision, recall, and F1-score were 0.96, 0.91, and 0.92 respec-

tively, showing it is a very dependable model for text recognition tasks. The neural network-based approach of Keras OCR, along with its ability to learn clear and effective visual patterns, explains its excellent performance. This model surpassed the best LSTM Tesseract model by over 20 percentage points in accuracy, setting a new benchmark for OCR capabilities. The output of the Keras model was used to generate audio of the recognized text using the `pyttsx3` library, enhancing its practical application.

4 Conclusion and Future Work

In the future, we aim to enhance the system’s capabilities by focusing on real-time text recognition and audio conversion, making it more practical for dynamic environments such as reading signs or labels on the go. A key improvement would be training the OCR models on larger and more diverse datasets, such as the COCO-Text dataset. While our attempt to utilize COCO-Text was limited by its size, implementing parallel processing can overcome such challenges, speeding up training and enabling the system to handle larger datasets effectively.

Additionally, we plan to integrate handwriting recognition, expanding the system’s usability for processing notes, forms, and historical documents. This would further enhance accessibility for visually impaired individuals, providing a comprehensive solution for both printed and handwritten text.

Our project demonstrated the potential of combining OCR models and text-to-speech technology to assist visually impaired individuals. By leveraging tools like Keras OCR and EasyOCR, we achieved robust text recognition in diverse conditions. While there is room for improvement, the system lays a foundation for accessible, real-time text-to-audio conversion for future projects.

References

- [1] Clova AI. Craft-pytorch: Character region awareness for text detection, n.d.
- [2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1601.01100*, 2016.
- [3] IBM. Optical character recognition (ocr), 2024.
- [4] Klippa. Tesseract ocr: An overview of features and capabilities, n.d.
- [5] Aditya Mahajan. Easyocr: A comprehensive guide, 2021.
- [6] Fausto Morales. keras-ocr: A library for optical character recognition in keras, n.d.

- [7] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.
- [8] Ray Smith. An overview of the tesseract ocr engine. *arXiv preprint arXiv:1604.06646*, 2016.
- [9] Ray W. Smith. An overview of the tesseract ocr engine. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 2, pages 629–633, 2007.
- [10] Tesseract OCR Team. Modernization efforts of the tesseract ocr engine, 2016.
- [11] Tesseract OCR Team. Tesseract ocr release notes: Version 4.00, n.d.
- [12] Tesseract OCR Team. Tesseract ocr vgs1 specification documentation, n.d.
- [13] D. R. Vedhaviyassh, R. Sudhan, G. Saranya, M. Safa, and D. Arun. Comparative analysis of easyocr and tesseractocr for automatic license plate recognition using deep learning algorithm. In *2022 6th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 966–971, 2022.