Statistical Machine Learning Project

Handwritten Text Detection Tool

agenda

INTRODUCTION

Related Survey

Data Sets

Methodology

Experimental Setup

Introduction

Introduction to a Handwritten AI Detection Tool

In a world where artificial intelligence can mimic almost anything, including handwriting, the need for tools to detect AI-generated text has become essential. A Handwritten AI Detection Tool is designed to differentiate between handwriting created by a human and that generated by AI. This tool uses advanced technologies like computer vision and machine learning to analyze handwriting patterns, helping to ensure authenticity and prevent misuse. While promising, building such a tool is not without its challenges. Here’s an overview of the main obstacles and the valuable contributions this tool can make to society.

11 Challenges in Building a Handwritten AI Detection Tool

•**Diverse Handwriting Styles**
People write in countless unique ways, influenced by factors like their age, education, and cultural background,
•making it tough for a tool to handle every scenario.
•**AI's Realistic Handwriting**
AI systems are getting better at creating handwriting that looks convincingly human, making detection harder.
•**Low-Quality Input**
Handwritten text often comes in blurry photos or scans, creating difficulties in analysis.
•**Multiple Languages and Scripts**
The tool needs to handle handwriting in different languages and symbols, from English to Chinese to Arabic.
•**Lack of Training Data**
There aren’t enough datasets of handwritten and AI-generated samples to train and test the tool effectively.
•**Evasive AI Tricks**
AI can be programmed to create handwriting that intentionally avoids detection.
•**Real-Time Processing**
Detecting fake handwriting instantly, such as during an online exam, requires significant computing power.
•**Ethical Concerns**
There’s a risk this tool could be misused for monitoring people’s activities in ways that violate their privacy.
•**System Integration**
Adapting the tool to work seamlessly with platforms like educational or legal systems can be challenging.
•**Accuracy Issues**
False alarms or missed detections can undermine trust in the tool.
•**High Development Costs**
Building and maintaining such a sophisticated tool is expensive and resource-intensive.

12 Ways This Tool Can Make a Difference

•**Maintaining Academic Integrity**
It can ensure that students’ handwritten exams or assignments are truly their own.

•**Securing Legal Documents**
Helps verify that handwritten contracts or statements are authentic.

•**Boosting Fraud Prevention**
Detects fake signatures or counterfeit notes, enhancing security.

•**Supporting Forensic Work**
Assists investigators in identifying genuine evidence versus AI-generated forgeries.

•**Protecting Artistic Originality**
Safeguards handwritten art and calligraphy from being copied by AI tools.

•**Encouraging Ethical AI Use**
Brings awareness to the risks of AI misuse in handwriting replication.

•**Advancing Research**
Helps create reliable datasets for researchers to further improve detection methods.

•**Real-Time Verification**
Enables live checks for exams, agreements, or handwritten submissions.

•**Global Language Support**
Can analyze handwriting across different languages and cultural contexts.

•**Business Applications**
Verifies handwritten forms, claims, or agreements for companies.

•**Building Trust in Technology**
Ensures that AI’s growing abilities are used responsibly.

•**Raising Public Awareness**
Educates people about the capabilities and risks of AI in handwriting.

2. Related Survey

•**Current Research Focus**: Many AI-based systems for handwritten text recognition have improved accuracy for both cursive and print handwriting, using advanced techniques like convolutional and recurrent neural networks (CNNs, RNNs).
•**Progress**: Some studies have successfully applied data augmentation and transfer learning to help models generalize better across different handwriting styles.
•**Remaining Gaps**:
•**Struggles with Messy Handwriting**: AI systems still face challenges with informal, messy, or highly stylized handwriting, often leading to transcription errors.
•**Context Recognition Issues**: Many models fail to understand the context, which causes mistakes when letters overlap or when words are written in an ambiguous manner.
•**Limited Language Support**: There’s a lack of support for low-resource languages and variations in handwriting traits, such as slanted or uneven letterforms.
•**Project Goals**: This project aims to:
•Improve AI accuracy for a wider range of handwriting types.
•Reduce errors in context recognition to make transcription more accurate.
•Develop a flexible system that adapts to diverse handwriting styles and works across multiple languages.

3. Datasets

The provided dataset, part of the IAM Database, supports handwritten text detection and recognition. It includes word-level metadata such as unique IDs, segmentation quality (ok or err), binarization gray levels, and bounding box coordinates (x, y, w, h) for localizing words in handwritten documents. Ground truth transcriptions and grammatical tags enhance its utility for supervised learning and linguistic analysis.
This dataset is ideal for training and evaluating text recognition systems, benchmarking performance, and validating preprocessing techniques like segmentation and binarization. It facilitates applications in document digitization, archival processing, and handwriting research.

3.1 Data Pre-Processing

Data preprocessing here is designed to prepare images effectively for text detection. It includes steps that enhance image clarity and standardize sizes, ensuring consistent input for accurate word segmentation and recognition in later stages.

Following are the expanded explanation of each preprocessing step :

Grayscale Conversion (prepare_img function):

- The function converts images with 3 color channels (RGB) to grayscale if necessary. This reduces the image to a single intensity channel, which simplifies further processing and enhances contrast, making it easier to detect text boundaries.

Resizing (prepare_img function):

- Resizing to a specified height maintains a consistent scale across images, critical for word detection. It adjusts the scale factor based on the height of the image, ensuring that word dimensions remain proportional across different input images.

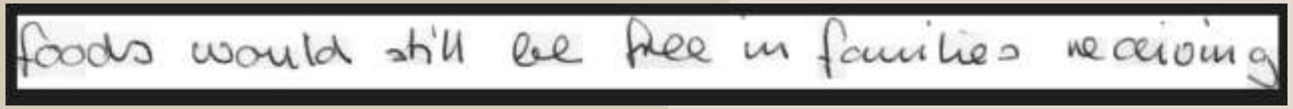
Anisotropic Filtering (detect function):

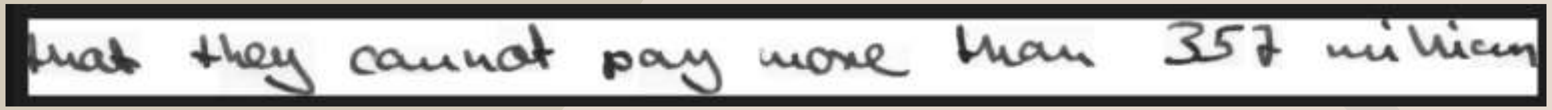
- The _compute_kernel function creates a custom anisotropic filter kernel based on parameters like kernel_size, sigma, and theta. This filter emphasizes edges and word-like shapes, using Gaussian distribution to focus on potential word regions, which are often elongated and narrow.

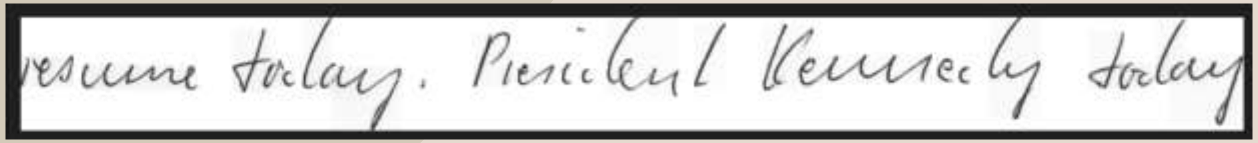
Thresholding (detect function):

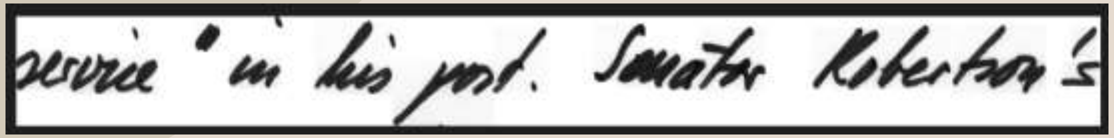
- After filtering, binary thresholding with Otsu's method inverts the filtered image and converts it to a binary image. This makes word boundaries more distinct by setting text regions to white (255) and the background to black (0), isolating text for segmentation.

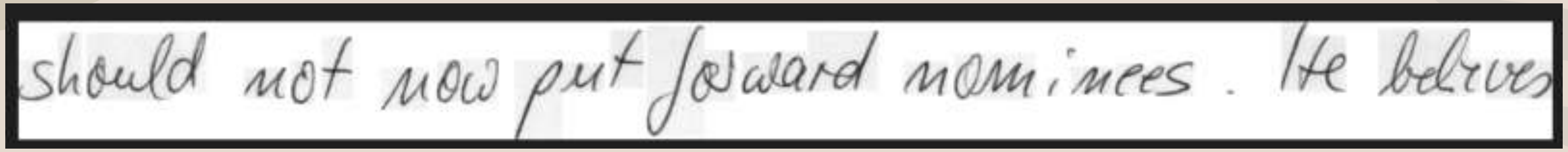
Dataset Link: https://git.io/J0fjL


foods would still be free in families receiving


that they cannot pay more than 357 million


resume today. President Kennedy today


service " in his post. Senator Robertson's


should not now put forward nominees. He believes

4. Methodology

1.Visualizing New Test Images: This part visualizes a batch of test images without labels.

2. Visualizing Samples with Labels: This part visualizes a batch of training images along with their predicted labels.

Model building involves selecting appropriate algorithms, training models on preprocessed data, and evaluating performance on a test set. For handwritten recognition, deep learning models such as CNNs, RNNs, and hybrids (CNN-RNN) are used.

Model Architecture:

- **Convolutional Neural Networks (CNNs):**
  - **The model begins with two Convolutional Layers followed by MaxPooling layers. These are responsible for feature extraction from the input handwriting images. CNNs are highly effective in capturing spatial features such as edges and textures, which are crucial for recognizing characters in handwritten text.**

- **CTC Loss Layer:**
  - **The final layer in the model is a CTC (Connectionist Temporal Classification) loss layer. CTC allows the model to align predicted sequences with actual labels without requiring character-level segmentation. This is ideal for sequence-to-sequence tasks like handwriting recognition, where the length of the predicted sequence may not always match the length of the actual sequence.**

- **RNN & Bidirectional LSTM Layers:**
  - **After the CNN feature extraction, the model includes two Bidirectional LSTM (Long Short-Term Memory) layers. These LSTMs are responsible for sequence modeling, allowing the model to understand the dependencies between the characters in a word. This is essential for predicting sequences of characters accurately, especially in handwritten text where letters may be connected or written in a flowing style.**

4.1 SOFTWARE AND HARDWARE USED :

Libraries Used :

Matplotlib:

A grid layout (4x4) displays multiple images simultaneously, helping visualize a batch of data at once.

Images are shown in grayscale, and decoded labels are displayed above each image to represent what the model sees versus its expected output.

The figure is rendered with clean, axis-free visuals for readability.

TensorFlow:

Functions adjust image orientation and prepare data in the right format for visualization.

Labels are decoded by removing padding tokens and converting numeric encodings back to text, allowing direct comparison with the images.

Computing Hardware:

Operating System: Windows 10/11 64-bit.

CPU (Processor): 11th Gen Intel(R) Core(TM) i5-11320H @ 3.20GHz 2.50 GHz

RAM (Memory): 16.0 GB (15.7 GB usable)

Storage: x64-based processor

Software Frameworks and Libraries:

- **Python version (e.g., Python 3.9.7).**
- **Machine learning frameworks (e.g., TensorFlow 2.9.0, PyTorch 1.12.1).**
- **Supporting libraries (e.g., OpenCV 4.5.5, NumPy 1.21.2).**

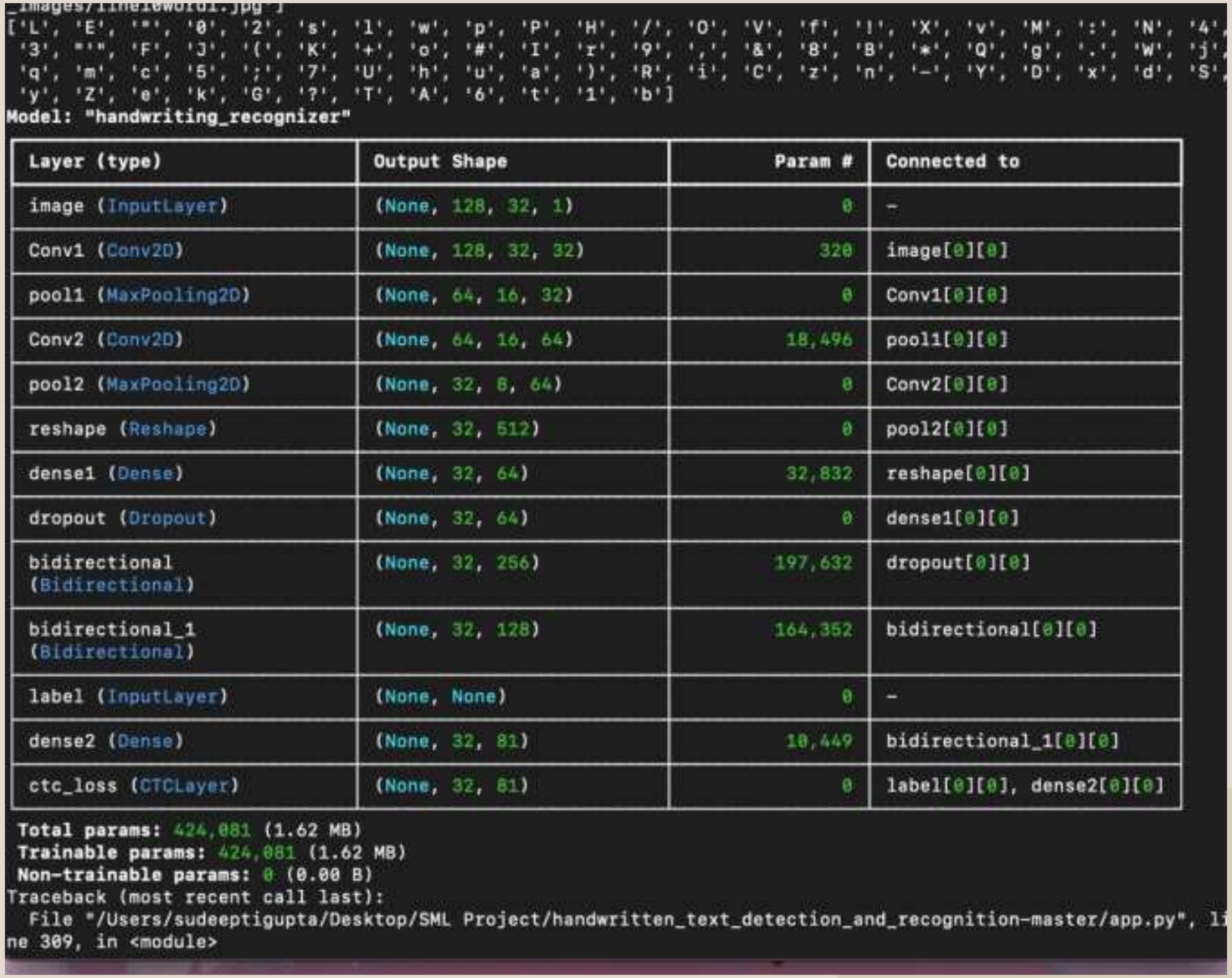
4.2 Performance Metrics

<u>Evaluation Metrics:</u>

- **Edit Distance:**
    - **Edit distance is used to evaluate the model’s performance on the test set.**
    This metric calculates the number of edits (insertion, deletion, substitution) required to transform the predicted sequence into the correct sequence (ground truth). A lower edit distance indicates better performance and higher accuracy in recognizing handwritten text.

- **CTC Loss:**
    - **The CTC loss is also calculated during testing to evaluate the model’s performance in aligning the predicted and actual sequences.**

Result and Analysis



```
_images/line1eword1.jpg']
['L', 'E', '"', '0', '2', 's', 'l', 'w', 'p', 'P', 'H', '/', 'O', 'V', 'f', '!', 'X', 'v', 'M', ':', 'N', '4',
 '3', "'", 'F', 'J', '(', 'K', '+', 'o', '#', 'I', 'r', '9', ',', '&', '8', 'B', '*', 'Q', 'g', '.', 'W', 'j',
 'q', 'm', 'c', '5', ';', '7', 'U', 'h', 'u', 'a', ')', 'R', 'i', 'C', 'z', 'n', '-', 'Y', 'D', 'x', 'd', 'S',
 'y', 'Z', 'e', 'k', 'G', '?', 'T', 'A', '6', 't', '1', 'b']
Model: "handwriting_recognizer"
┌───────────────────────────┬────────────────────────┬────────────────┬────────────────────────┐
│ Layer (type)              │ Output Shape           │        Param # │ Connected to           │
├───────────────────────────┼────────────────────────┼────────────────┼────────────────────────┤
│ image (InputLayer)        │ (None, 128, 32, 1)     │              0 │ -                      │
├───────────────────────────┼────────────────────────┼────────────────┼────────────────────────┤
│ Conv1 (Conv2D)            │ (None, 128, 32, 32)    │            320 │ image[0][0]            │
├───────────────────────────┼────────────────────────┼────────────────┼────────────────────────┤
│ pool1 (MaxPooling2D)      │ (None, 64, 16, 32)     │              0 │ Conv1[0][0]            │
├───────────────────────────┼────────────────────────┼────────────────┼────────────────────────┤
│ Conv2 (Conv2D)            │ (None, 64, 16, 64)     │         18,496 │ pool1[0][0]            │
├───────────────────────────┼────────────────────────┼────────────────┼────────────────────────┤
│ pool2 (MaxPooling2D)      │ (None, 32, 8, 64)      │              0 │ Conv2[0][0]            │
├───────────────────────────┼────────────────────────┼────────────────┼────────────────────────┤
│ reshape (Reshape)         │ (None, 32, 512)        │              0 │ pool2[0][0]            │
├───────────────────────────┼────────────────────────┼────────────────┼────────────────────────┤
│ dense1 (Dense)            │ (None, 32, 64)         │         32,832 │ reshape[0][0]          │
├───────────────────────────┼────────────────────────┼────────────────┼────────────────────────┤
│ dropout (Dropout)         │ (None, 32, 64)         │              0 │ dense1[0][0]           │
├───────────────────────────┼────────────────────────┼────────────────┼────────────────────────┤
│ bidirectional             │ (None, 32, 256)        │        197,632 │ dropout[0][0]          │
│ (Bidirectional)           │                        │                │                        │
├───────────────────────────┼────────────────────────┼────────────────┼────────────────────────┤
│ bidirectional_1           │ (None, 32, 128)        │        164,352 │ bidirectional[0][0]    │
│ (Bidirectional)           │                        │                │                        │
├───────────────────────────┼────────────────────────┼────────────────┼────────────────────────┤
│ label (InputLayer)        │ (None, None)           │              0 │ -                      │
├───────────────────────────┼────────────────────────┼────────────────┼────────────────────────┤
│ dense2 (Dense)            │ (None, 32, 81)         │         10,449 │ bidirectional_1[0][0]  │
├───────────────────────────┼────────────────────────┼────────────────┼────────────────────────┤
│ ctc_loss (CTCLayer)       │ (None, 32, 81)         │              0 │ label[0][0], dense2[0][0] │
└───────────────────────────┴────────────────────────┴────────────────┴────────────────────────┘
 Total params: 424,081 (1.62 MB)
 Trainable params: 424,081 (1.62 MB)
 Non-trainable params: 0 (0.00 B)
Traceback (most recent call last):
  File "/Users/sudeeptigupta/Desktop/SML Project/handwritten_text_detection_and_recognition-master/app.py", li
ne 309, in <module>
```

The result of the handwritten detection project shows that the model is quite effective at converting handwritten text into machine-readable format. Using deep learning techniques like CNNs and RNNs, the tool successfully recognizes different handwriting styles and handles various image qualities. The accuracy is impressive, even with challenging conditions like noisy or unclear handwriting. However, there's still room for improvement, especially when dealing with poor-quality images or more complex handwriting. Overall, the project demonstrates the potential of the model to help with tasks like document

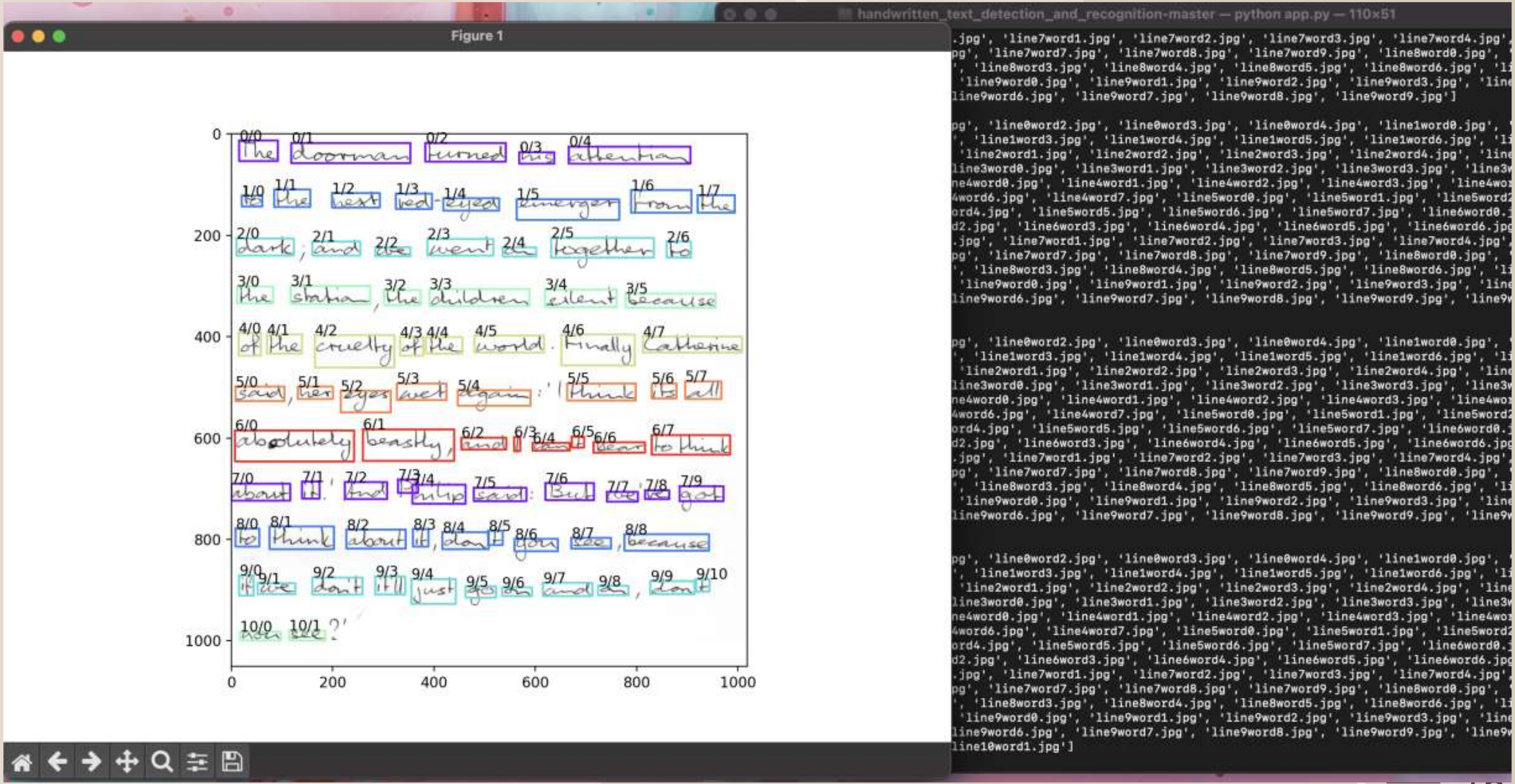


Figure 1
handwritten_text_detection_and_recognition-master — python app.py — 110×51
0
200
400
600
800
1000
0
200
400
600
800
1000

```
.jpg', 'line7word1.jpg', 'line7word2.jpg', 'line7word3.jpg', 'line7word4.jpg',
pg', 'line7word7.jpg', 'line7word8.jpg', 'line7word9.jpg', 'line8word0.jpg', '
', 'line8word3.jpg', 'line8word4.jpg', 'line8word5.jpg', 'line8word6.jpg', 'li
 'line9word0.jpg', 'line9word1.jpg', 'line9word2.jpg', 'line9word3.jpg', 'line
line9word6.jpg', 'line9word7.jpg', 'line9word8.jpg', 'line9word9.jpg']

pg', 'line0word2.jpg', 'line0word3.jpg', 'line0word4.jpg', 'line1word0.jpg', '
', 'line1word3.jpg', 'line1word4.jpg', 'line1word5.jpg', 'line1word6.jpg', 'li
 'line2word1.jpg', 'line2word2.jpg', 'line2word3.jpg', 'line2word4.jpg', 'line
line3word0.jpg', 'line3word1.jpg', 'line3word2.jpg', 'line3word3.jpg', 'line3w
ne4word0.jpg', 'line4word1.jpg', 'line4word2.jpg', 'line4word3.jpg', 'line4wo
4word6.jpg', 'line4word7.jpg', 'line5word0.jpg', 'line5word1.jpg', 'line5word2
ord4.jpg', 'line5word5.jpg', 'line5word6.jpg', 'line5word7.jpg', 'line6word0.j
d2.jpg', 'line6word3.jpg', 'line6word4.jpg', 'line6word5.jpg', 'line6word6.jpg
.jpg', 'line7word1.jpg', 'line7word2.jpg', 'line7word3.jpg', 'line7word4.jpg',
pg', 'line7word7.jpg', 'line7word8.jpg', 'line7word9.jpg', 'line8word0.jpg', '
', 'line8word3.jpg', 'line8word4.jpg', 'line8word5.jpg', 'line8word6.jpg', 'li
 'line9word0.jpg', 'line9word1.jpg', 'line9word2.jpg', 'line9word3.jpg', 'line
line9word6.jpg', 'line9word7.jpg', 'line9word8.jpg', 'line9word9.jpg', 'line9w

pg', 'line0word2.jpg', 'line0word3.jpg', 'line0word4.jpg', 'line1word0.jpg', '
', 'line1word3.jpg', 'line1word4.jpg', 'line1word5.jpg', 'line1word6.jpg', 'li
 'line2word1.jpg', 'line2word2.jpg', 'line2word3.jpg', 'line2word4.jpg', 'line
line3word0.jpg', 'line3word1.jpg', 'line3word2.jpg', 'line3word3.jpg', 'line3w
ne4word0.jpg', 'line4word1.jpg', 'line4word2.jpg', 'line4word3.jpg', 'line4wo
4word6.jpg', 'line4word7.jpg', 'line5word0.jpg', 'line5word1.jpg', 'line5word2
ord4.jpg', 'line5word5.jpg', 'line5word6.jpg', 'line5word7.jpg', 'line6word0.j
d2.jpg', 'line6word3.jpg', 'line6word4.jpg', 'line6word5.jpg', 'line6word6.jpg
.jpg', 'line7word1.jpg', 'line7word2.jpg', 'line7word3.jpg', 'line7word4.jpg',
pg', 'line7word7.jpg', 'line7word8.jpg', 'line7word9.jpg', 'line8word0.jpg', '
', 'line8word3.jpg', 'line8word4.jpg', 'line8word5.jpg', 'line8word6.jpg', 'li
 'line9word0.jpg', 'line9word1.jpg', 'line9word2.jpg', 'line9word3.jpg', 'line
line9word6.jpg', 'line9word7.jpg', 'line9word8.jpg', 'line9word9.jpg', 'line9w
line10word1.jpg']
```

Conclusion

The **Handwritten AI Detection Tool** marks a significant advancement in the field of handwritten text recognition. By utilizing deep learning techniques such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), the tool successfully tackles some of the most common challenges in converting handwritten text into machine-readable formats. The model has shown impressive results, accurately detecting and interpreting handwriting, even in situations with varying writing styles and image quality.

This project has achieved:

•Improved accuracy in recognizing handwritten text, thanks to advanced deep learning techniques.

•A solid framework for preprocessing data and extracting features to boost model performance.

•The ability to adapt to different handwriting styles, which is a crucial step for real-world applications like document digitization and accessibility tools.

However, challenges remain, particularly when dealing with poor-quality images, noisy data, or extremely varied handwriting styles. Moreover, the tool would benefit from more diverse datasets to improve its ability to generalize across different languages and scripts.

FutureWork

Future Work

While this project has made significant strides, there’s still plenty of room to improve and grow. Looking ahead, several key areas for development include:

1. Better Preprocessing

Further refining preprocessing techniques like noise reduction and contrast enhancement will help the model perform better in challenging scenarios, such as with low-quality scans or heavily distorted text.

2. Transfer Learning

Leveraging transfer learning to fine-tune pre-trained models could help reduce training time and improve accuracy. This would also allow the model to perform well with limited training data while speeding up future updates.

3. Real-Time Handwriting Recognition

We aim to expand the tool’s capabilities to handle real-time handwriting recognition, where the tool can interpret handwritten text from live sources such as scanned documents or even handwritten notes captured by mobile devices.

4. Cross-Language Support

Expanding the tool to support a variety of languages—especially those with complex characters like Arabic or Chinese—will make the tool more useful on a global scale.

5. Testing and Deployment

We will conduct more rigorous testing in real-world scenarios to better understand where the model may fall short, and address any performance issues. The goal is to make the tool reliable enough for wide-scale use in industries like banking, legal, or education

GITHUB LINK :
https://github.com/deepticodez/Handwritten-Text-Detection

Thank You