|  |  |
| --- | --- |
| Internship Project Title | Automate extraction of handwritten text from an image |
| Name of the Company | TCS iON |
| Name of the Industry Mentor | Mr. Debashis Roy |
| Name of the Institute | Vishwakarma University, Pune |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Start Date | End Date | Total Effort (hrs.) | Project Environment | Tools used |
| 15/02/2025 | 02/03/2025 | 60 | Jupyter Notebook (Python 3) | Google Colab, Python3,Matplotlib,Tensorflow,etc. |
| Milestone | 1 | Milestone: | Explored OCR tools, collected and preprocessed handwritten images, implemented text extraction | |

**TABLE OF CONTENT**

* Acknowledgements
* Objective
* Introduction / Description of Internship
* Internship Activities
* Approach / Methodology
* Assumptions
* Exceptions / Exclusions
* Charts, Table, Diagrams
* Algorithms
* Challenges & Opportunities
* Risk Vs Reward
* Reflections on the Internship
* Recommendations
* Outcome / Conclusion
* Enhancement Scope

## **ACKNOWLEDGEMENT**

I would like to express my deepest gratitude to my industry mentor, Mr. Debashis Roy, from TCS-iON, and my academic mentor, Prof. Shriprada Chaturbhuj from Vishwakarma University, for their valuable guidance and support throughout the internship period. Their continuous encouragement and expert advice were crucial to the completion of this project.

I also extend sincere thanks to TCS-iON and Vishwakarma University for offering this remarkable opportunity. The internship allowed me to gain practical experience in deploying machine learning solutions, which significantly contributed to my professional growth.

**OBJECTIVE**

The main objective of this internship project was to create an automated solution for recognizing handwritten text from images using advanced OCR techniques powered by deep learning. The system aimed to extract meaningful text content from scanned handwritten documents using tools like EasyOCR and a custom-built CRNN model.

Key goals included:

* Implementing preprocessing steps for better text visibility
* Detecting handwritten regions using OCR tools
* Recognizing characters using deep learning models
* Presenting recognized text in readable format

This solution has applications in digital archiving, document automation, accessibility tools, and educational technology.

## **INTRODUCTION/DESCRIPTION OF THE INTERNSHIP**

This internship was carried out under the TCS iON RIO125 program. The goal was to gain real-world experience in building an Optical Character Recongnition (OCR) system using Python. The system used EasyOCR to perform initial detection and recognition, supported by a deeper Convolutional Recurrent Neural Network (CRNN) architecture for improved accuracy.

Google Colab served as the primary development platform because of its GPU acceleration and collaborative capabilities. Through this project, I was exposed to computer vision fundamentals, deep learning concepts, and the challenges associated with handwritten text extraction.

## **INTERNSHIP ACTIVITIES**

Key activities during the first 15 days included:

* Dataset preparation and image preprocessing.
* Text detection using EasyOCR and bounding box extraction.
* Cropping detected words from original images.
* Sorting and naming cropped images for correct text sequence.
* Building a deep learning model using TensorFlow and Keras.
* Training and testing the model for recognizing handwritten characters.
* Implementing a complete inference pipeline to output recognized text.

## **APPROACH/METHODOLOGY**

The following methodology was applied:

1. **Image Preprocessing**:
   * Convert colored images to grayscale to simplify the input.
   * Resize images while preserving aspect ratio to fit the model input size.
   * Apply padding to maintain uniform input shape.

**2.Text Detection:**

* + EasyOCR detects text regions using a deep learning-based detector.
  + Bounding boxes are drawn around each detected word or text line.

**3. Cropping and Sorting:**

* + Detected text regions are cropped from the original image.
  + Cropped segments are sorted from top to bottom and left to right to preserve reading order.

1. **Data Labeling:**
   1. Assign ground truth labels to each cropped image for supervised learning.
2. **Model Architecture:**
   1. CNN layers extract spatial features from the image.
   2. Bi-directional LSTM layers capture sequential dependencies in the text.
   3. Dense layer outputs character probabilities.
3. **Loss Function:**
   1. CTC (Connectionist Temporal Classification) loss is used to train the model without the need for character-level alignment.
4. **Model Training and Evaluation:**
   1. Model is trained on the labeled dataset with validation accuracy and loss plotted.
   2. Evaluation involves decoding predicted sequences and comparing with ground truth.
5. **Text Decoding and Output Generation:**
   1. TensorFlow’s StringLookup layer maps numeric predictions back to characters.
   2. Final output is reconstructed text displayed in logical reading order.

## **ASSUMPTIONS**

* Handwritten text is well-aligned and not cursive or overlapping.
* Input images are of a certain resolution and quality (e.g., PNG, JPG, BMP).
* Text regions are non-rotated and in horizontal format.
* Training data represents all characters expected in real-world testing.

**EXCEPTIONS/EXCLUSIONS**

* System does not process multilingual content (English-only model)
* Skewed, low-light, or heavily noisy images were excluded
* Word spacing and character-level bounding boxes not optimized
* Real-time webcam-based recognition not implemented

## **ALGORITHMS**

**Step-by-Step Algorithm:**

This section explains the algorithm used for recognizing handwritten text in a simple and detailed manner:

**Step 1: Image Acquisition**:

* + The first step is to gather images that contain handwritten text. These images can be uploaded from the local system or imported directly from cloud storage platforms such as Google Drive.This input acts as the raw material for the entire OCR pipeline.

**Step 2: Image Preprocessing**:

To improve the performance of the OCR system, basic preprocessing techniques are applied. This typically includes:

* + Images are first converted to **grayscale**. This removes color and simplifies the image for further processing.
  + The grayscale image is resized so that all images have the same dimensions. This helps the model work with consistent input.
  + Noise in the image is removed using techniques like **Gaussian blur or median filtering**.

This step helps clean the image and makes the text more visible.

**Step 3: OCR Engine Initialization**:

* + EasyOCR, a tool based on deep learning, is initialized. It comes with pre-trained models that are already capable of recognizing English characters.

**Step 4: Text Detection and Recognition**:

The algorithm then processes the preprocessed image using the OCR engine. It detects regions in the image that contain text and attempts to recognize the characters within those regions. For each text region, the OCR engine outputs:

* The **bounding box** around the text
* The **recognized string**
* A **confidence score**, indicating the accuracy of recognition

This step forms the core of the system, turning visual patterns into textual information.

**Step 5: Confidence Filtering and Post-Processing**:

* + After recognition, the algorithm filters out low-confidence results to improve output quality. Only text predictions above a certain confidence threshold are retained. This helps eliminate misrecognized text or irrelevant noise.
  + In post-processing, the extracted text may be cleaned to remove any special symbols or formatting issues. The final recognized text is then prepared for display or storage.

**Step 6: Text Sequence Generation**:

* + The recognized text is converted from numbers (model output) into readable characters using TensorFlow's StringLookup.
  + These characters are then combined to form words and full sentences.

**Step 7: Display Output**:

The final recognized text is shown to the user. It can be printed on the screen or saved into a file for future use.

## **CHALLENGES & OPPORTUNITY**

**Challenges:**

1. **Low Accuracy in Complex Handwriting:**
   * Extracting handwritten content from images with cursive, skewed, or cluttered text proved difficult. Pre-trained models like EasyOCR showed inconsistencies in such scenarios.
2. **Image Quality Variations:**
   * Images had inconsistent lighting, background noise, and resolution, which affected the recognition pipeline. This required manual preprocessing adjustments like grayscale conversion, denoising, and resizing.
3. **OCR Misinterpretation:**
   * Characters such as '1' and 'l' or 'O' and '0' were often confused. Such OCR errors were difficult to avoid without deeper model customization.

**Opportunity:**

1. **Real-time Application Potential:**
   * The project can be scaled to scan exam sheets, digitize handwritten forms, or even aid visually impaired users by converting notes to speech.
2. **Scope for Integration:**
   * Integration with Flask or Streamlit can lead to a fully interactive web-based handwritten text reader.
3. **Multilingual Expansion:**
   * EasyOCR supports multiple languages; this opens the door to process regional or foreign scripts with future tweaks.

## **RISKS vs REWARDS**

**Risks Identified:**

* **OCR Inaccuracies:** Risk of misclassification could lead to incorrect interpretations, especially in critical use-cases like medical forms or official documentation.
* **Image Sensitivity:** OCR's dependency on clean inputs means a small flaw in image capture can lead to huge drops in accuracy.
* **Scalability Risk:** Without custom models and batch processing, large-scale deployment may be inefficient.

**Rewards/Benefits:**

* **Automated Workflow:** Manual transcription efforts are greatly reduced.
* **Baseline Framework Ready:** The project sets the foundation for future integrations like mobile scanners or cloud-based solutions.
* **Learning Curve:** Exposure to OCR, Python libraries, and image handling improved overall technical competency and problem-solving skills.

## **REFLECTION ON THE INTERNSHIP**

This internship helped me move beyond just learning from books. I got a chance to work on a real project and apply what I had learned in the classroom to solve real-world problems using OCR and machine learning.

**Personal Growth:**

* I worked with real images and data, which helped me understand the challenges in reading handwritten text.
* My confidence in using Python and building computer vision models improved a lot.
* I also became comfortable using tools like Google Colab to run and test my code online.

**Learning Highlights:**

* I learned how tools like EasyOCR work and how they can recognize text in images.
* I understood how important it is to clean and prepare images before giving them to the model.
* I saw that open-source tools, when used correctly, can give really good results.

## **RECOMMENDATIONS**

1. **Introduce Custom Model Training (Advanced Phase):**
   * Use libraries like PyTorch with CNN-RNN hybrid architectures for better accuracy on complex handwriting.
2. **Incorporate Multilingual Support:**
   * Expand to regional languages using EasyOCR’s multilingual capabilities.
3. **Web UI Deployment:**
   * Integrate with a basic frontend using Streamlit or Flask for non-developers to test OCR features.
4. **Add Preprocessing Automation:**
   * Implement auto contrast, skew correction, and noise removal to standardize image quality.
5. **Enable Batch Uploading:**
   * Support multiple image uploads for faster testing and report generation.

## **OUTCOME/CONCLUSIONS**

The internship resulted in a functional Python-based OCR solution that extracts handwritten English text from images. While limited in language support and real-time capabilities, it served as a strong prototype for future improvements. The project provided insights into text recognition systems and hands-on exposure to OCR challenges

.

## **ENHANCEMENT SCOPE**

**1. Multilingual Handwriting Recognition**

* Currently, the system supports only English characters. A potential enhancement involves incorporating **multilingual capabilities** (e.g., Hindi, Marathi, Tamil, etc.) using EasyOCR’s multi-language support or integrating Google’s Vision API.
* This would expand the usability of the tool in diverse linguistic contexts, especially in multilingual countries like India.

**2. Bulk Image Processing**

* At present, the system processes one image at a time. An enhancement would include the ability to **upload and process multiple images** in a batch.
* This would reduce manual effort and improve productivity, especially for datasets or scanned documents.

**3. Custom OCR Model Training**

* The current implementation uses EasyOCR’s pre-trained models. Training a **custom OCR model** on a curated dataset of handwriting samples can significantly improve accuracy, especially for unusual or highly cursive writing styles.
* This can be done using frameworks like TensorFlow, PyTorch, or Tesseract with custom datasets.

**4. Real-Time Image Capture & Processing**

* Integrating **real-time camera input** can allow the system to instantly extract handwritten content from live video feeds or photos.
* This enhancement is useful in applications like classroom board scanning or form digitization on the go.