|  |  |
| --- | --- |
| 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'm truly grateful for the unwavering support and guidance extended to me throughout my project, RIO-125: Automate extraction of handwritten text from an image. I want to express my heartfelt appreciation to my industry mentor, Mr. Debashis Roy from TCS-iON, and my academic mentor, Prof. Shriprada Chaturbhuj from Vishwakarma University. Their constant motivation played a pivotal role in my journey.

Additionally, I extend my sincere thanks to TCS-iON and Vishwakarma University for granting me this invaluable opportunity, which has enriched my understanding of the industry landscape. I want to emphasize that I completed the project independently, without any external assistance.

## OBJECTIVE

The primary objective of this internship was to automate the extraction of handwritten text from images using EasyOCR and TensorFlow-based deep learning models. The goal was to implement a pipeline that detects and recognizes handwritten words in images, enabling efficient and accurate data extraction for various applications such as digitizing historical documents, automated form processing, and enhancing accessibility tools.

## 

## INTRODUCTION/DESCRIPTION OF THE INTERNSHIP

This internship was conducted under the TCS iON RIO125 program and involved developing a complete solution for recognizing handwritten text using deep learning. The project utilized EasyOCR for Optical Character Recognition (OCR) and a custom Convolutional Recurrent Neural Network (CRNN) model for sequence learning. Google Colab was used as the development environment due to its GPU acceleration and ease of collaboration. The internship provided a hands-on opportunity to understand and implement computer vision, OCR, and deep learning techniques.

## 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

Steps:

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

* The solution does not handle multi-language scripts or mixed-script content.
* Skewed, noisy, or low-resolution images were not considered during training.
* The pipeline does not perform character-level bounding or word spacing corrections.
* Does not support real-time webcam-based OCR.

## ALGORITHMS

**Step 1: Image Input Acquisition**

The first stage of the algorithm begins with acquiring the input image, which contains handwritten text. These images can be uploaded manually by the user or accessed from a predefined location, 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:

* **Grayscale Conversion**: Simplifies the image by removing color data, making text detection easier.
* **Noise Removal**: Eliminates background noise or irregularities.
* **Resizing**: Ensures the text is neither too small nor too large for detection.
* **Contrast Adjustment**: Enhances the difference between text and background.

This step ensures that the image is optimized for the OCR model to read more accurately.

**Step 3: OCR Engine Initialization**

A deep learning-based OCR model is initialized. In this case, EasyOCR is used, which loads a pre-trained neural network specifically designed to recognize characters from various scripts. The model supports multiple languages, though only English was used for this project phase.

**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: Output Generation**

The final stage involves presenting the extracted handwritten text to the user. This could be printed on the console.

## CHALLENGES & OPPORTUNITY

The different challenges and opportunities that were associated are as follows:

**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

“The internship enabled me to think beyond theory and explore real-world use of OCR.”

* **Personal Growth:**
  + I gained hands-on experience in implementing computer vision models using Python.
  + It gave me confidence in using platforms like Google Colab for real-time code execution and visual feedback.
* **Learning Highlights:**
  + Understood how OCR tools work under the hood.
  + Realized the importance of image preprocessing in improving model performance.
  + Discovered how existing open-source models can be utilized efficiently within a limited timeline.

## 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 interim phase successfully created a Python-based OCR solution using EasyOCR to extract handwritten text from images on Google Colab. While it was limited to English characters and individual image inputs The 15-day internship proved highly beneficial as it provided real-world insights into OCR implementation challenges and solutions.

## 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.