Domain Adaptation for OCR Models

Swathy Anand, and Swetha Mohan
University of Massachusetts, Amherst
300 Massachusetts Ave, Amherst, MA 01003
{swathyanand,swethamohan}@umass.edu

December 4, 2024

1 Abstract

This project aims to improve the performance of Optical Character Recognition (OCR) systems on handwritten text, particularly when faced with unfamiliar handwriting styles. While OCR models excel at recognizing printed text, they often struggle with the variability of handwriting, especially when labeled data is scarce. To address these challenges, we leveraged spell checking and self-supervised learning techniques, including pseudolabeling, to enhance model adaptability. The primary model is a Convolutional Recurrent Neural Network (CRNN), combining CNNs for feature extraction and LSTMs for sequence prediction, trained on the IAM Handwriting Dataset. Despite achieving good performance on the training dataset, the model's accuracy diminished when tested on custom, real-world handwritten data. Through the integration of spell checking and pseudolabeling, we aimed to improve generalization and handle the diversity of handwriting styles more effectively. This work contributes to the advancement of OCR technology, with potential applications in fields such as healthcare, education, and finance, by enabling more accurate and adaptable handwritten text recognition.

2 Introduction

Optical Character Recognition (OCR) systems have excelled in recognizing printed text but often encounter significant challenges with handwritten text, particularly when encountering unfamiliar handwriting styles. Models trained on specific datasets perform well within their domain but struggle to generalize effectively to diverse, real-world handwriting, especially in scenarios where labeled data is scarce or unavailable.

This project addresses these limitations by leveraging spell checking and self-supervised learning techniques to enhance OCR model performance for handwritten text recognition. Recognizing the inherent variability in individual and generational handwriting styles, the project focuses on improving the adaptability of OCR systems to diverse handwriting.

The primary training dataset is the IAM Handwriting Database, a standard benchmark for handwritten text recognition. To assess the model's capacity to generalize, we tested it on a custom dataset of handwritten words in our own handwriting, enabling a clear evaluation of its ability to adapt to novel styles. A Convolutional Recurrent Neural Network (CRNN) was developed for this purpose, combining convolutional layers for feature extraction with bidirectional LSTMs for sequence prediction. The model processes text from images of varying lengths and uses preprocessing techniques like binarization, resizing, and normalization to maintain input consistency. Training was conducted on grayscale image datasets, and performance was evaluated on both the training and test datasets.

To enhance the model's generalizability to unseen handwriting, we integrated spell checking techniques, improving its adaptability. We also looked into meth-

ods of semi-supervised learning, pseudo-labeling and word embedding. Performance was measured using the Character Error Rate (CER) and Word Error Rate (WER), which quantifies the proportion of incorrect characters and words, with lower values indicating a more robust and versatile OCR system.
This research aims to advance OCR technology by enabling more accurate and adaptable recognition of handwritten text, enhancing its applicability across domains such as healthcare, education, and finance.

3 Related Work

The foundation of our project is based on several key studies. "Transformer-based Optical Character Recognition with Pre-trained Models"[1], TrOCR is a Transformer-based OCR model that replaces CNNs with pre-trained image and text Transformers for text recognition, improving performance by leveraging large-scale, unlabeled data without image-specific biases. It achieves state-of-the-art results without the need for complex pre/post-processing steps, making it more efficient and easier to implement. But the weakness of TrOCR lies in its limited adaptability to varied handwriting datasets and occasional inaccuracies with complex texts due to a lack of specific pre-training and post-processing. The project aims to address this by using unsupervised learning methods to better suit diverse handwriting and adding lightweight post-processing to enhance accuracy in challenging recognition conditions.
"AT-ST: Self-Training Adaptation Strategy for OCR in Domains with Limited Transcriptions"[2] improves OCR accuracy in target domains like handwriting by utilizing minimal labeled data and high-confidence machine-generated transcriptions. Key techniques include masking augmentation and confidence-based selection, resulting in notable gains. However, the method may fall short in adapting to diverse handwriting styles or uncommon scripts, potentially affecting real-world performance. This project aims to address this by enhancing model adaptability through domain-specific fine-tuning and incorporating varied handwriting samples or meta-learning, enabling better recognition across different handwriting styles.
"A Survey of Unsupervised Domain Adaptation for Visual Recognition"[3] reviews methods for adapting models to unlabeled target domains but highlights key weaknesses, including over-reliance on pre-trained deep features and poor generalization when training and test data distributions differ. Our OCR project addresses these by customizing feature extraction for handwriting recognition, reducing dependence on static features, and enhancing generalization through tailored pre-training on diverse handwriting samples. This approach aims to ensure more accurate adaptation across varied handwriting styles, even under substantial domain shifts.
The paper "Visual Domain Adaptation: A Survey of Recent Advances"[4] reviews domain adaptation methods for addressing domain shifts in pattern recognition and computer vision. It identifies weaknesses such as limited handling of unstructured data and poor performance when source and target domains are significantly different. Our project addresses these issues by training on diverse handwriting samples to enhance adaptability to various styles and incorporating fine-tuning techniques to bridge distribution gaps, thereby improving performance with minimal labeled data.
Seq-UPS: Sequential Uncertainty-aware Pseudo-label Selection for Semi-Supervised Text Recognition"[5] proposes a robust pseudo-labeling framework leveraging Beam-Search inference and uncertainty-based data selection to enhance performance on unlabelled data. It addresses challenges like noisy labels and hypothesis space complexity using Monte Carlo Dropout for uncertainty estimation. This paper informed our project by highlighting the importance of robust pseudo-labeling and uncertainty measures, guiding our implementation of fine-tuning techniques to improve OCR performance on diverse handwriting styles with limited labeled data.

4 Methods

For the Optical Character Recognition (OCR) model, we developed a Convolutional Recurrent Neural Network (CRNN) model. This architecture combines

Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) with Connectionist Temporal Classification (CTC) loss, making it ideal for sequence-to-sequence tasks like handwriting or speech recognition, where input and output sequences have varying lengths and unknown alignment. The CNN layers extract visual features, while the LSTM layers capture contextual dependencies. The model outputs character probabilities at each timestep, with CTC loss enabling alignment of input-output sequences by introducing a "blank" token. CTC computes all possible alignments and minimizes the negative log-probability of the correct label sequence.
The model is trained using the Adam optimizer, which adjusts learning rates for faster convergence. It combines CNNs for spatial feature extraction, Bi-LSTMs for temporal modeling, and CTC for sequence alignment, making it well-suited for tasks with variable-length sequences.
For training, each word is represented by an image that is pre-processed using OpenCV. The images are resized to a consistent shape of (32, 128, 1), preserving the aspect ratio and padding if necessary. The images are normalized by inverting and scaling pixel values to [0, 1]. The data set is divided into training and validation sets, with every 10th sample used for validation. The images and labels are converted to NumPy arrays, ensuring consistent input during training.
Test dataset collection: The code was designed to extract individual words from test images of different handwritten notes. The process involved reading the input image, converting it to grayscale then using an adaptive binary thresholding technique to highlight the text while minimizing noise. A morphological dilation enlarged the contours of the text, allowing separate letters to merge into coherent words. This helps in accurately detecting word boundaries.

Contours are detected in the dilated image and are sorted from left to right based on their x-coordinates to maintain the correct reading order. Small contours, which are unlikely to represent full words, are filtered out by setting a minimum size threshold for the bounding boxes. Each valid word is then cropped from the original image of paragraphs and saved as a separate image file in a designated output folder.

This approach helps to isolate and save words from handwritten notes, enabling easier manipulation and recognition of the text in subsequent stages of processing.

To further improve the model, we have attempted 2 methods:

- Spell Checker Algorithms
- Retraining with Pseudo-labelling

4.1 Spell Checker Algorithms

The core idea behind integrating Spell Checker Algorithms is to improve the accuracy of character sequence predictions by leveraging a predefined dictionary of English words. The basic approach is to post-process the model's raw predictions by comparing them to existing words in the dictionary and selecting the most likely valid word. This is done by measuring the edit distance or similarity between the OCR model's prediction and the words in the dictionary, replacing predictions that are sufficiently close to a valid word.

Dictionary Comparison: The OCR model's raw predictions are compared against a predefined dictionary of valid English words. This is done by computing similarity measures like Levenshtein distance (edit distance), which quantifies how many single-character edits (insertions, deletions, or substitutions) are required to turn the OCR output into a valid word.

Word Correction: Once a close match is found, the OCR prediction is corrected by replacing the model's output with the valid word from the dictionary. This ensures that the OCR model outputs an actual, meaningful word rather than an incorrect or out-of-vocabulary sequence.
To facilitate this correction process, we tested 2 decoding methods: Greedy Decoding, Beam Search. These methods can be used to refine the final output by ensuring that it matches valid words or improves the likelihood of accurate predictions.

4.1.1 Greedy Decoding

In greedy decoding, the model chooses the most likely character at each timestep independently of others, without considering potential future characters. While efficient, this method may miss global dependencies and optimal character sequences.

4.1.2 Beam Search

Beam search improves upon greedy decoding by maintaining a set of the most likely sequences (called a "beam") rather than just the most probable next character. At each timestep, beam search considers multiple possibilities and keeps the top-k sequences with the highest cumulative probabilities. This approach can yield better results than greedy decoding, especially when the correct sequence has dependencies between characters or words.

4.1.3 Spell Checkers (Sym Spell Checker and PySpellChecker)

After decoding, spell checkers are applied to ensure that the predicted word is valid. Spell checkers work by correcting typographical errors based on a dictionary of correct words and common word mistakes.

- **SymSpell Checker:** It is a fast and memory-efficient spell checking algorithm that builds a dictionary based on word frequencies and stores word corrections in a compressed format. It performs spelling correction by generating all possible edits (insertions, deletions, or substitutions) and comparing them to the dictionary. SymSpell uses a prefix dictionary to speed up lookups and make the correction process more efficient. The key strength of SymSpell is its ability to quickly find near-matches, making it suitable for large-scale OCR correction tasks.

- **PySpellChecker:** Its is a Python library based on a probability model for spell correction. It utilizes a frequency dictionary of words and uses algorithms like n-gram models or edit distance to suggest corrections. Although it is a simpler solution, it still works well for common spelling errors but may not be as fast or scalable as SymSpell.

The main components of the code after training include image preprocessing, prediction, spelling correction, and evaluation metrics (Character Error Rate (CER) and Word Error Rate (WER)).

The preprocessing steps enhance the quality of input images for text recognition by converting them to grayscale, applying Gaussian blur to reduce noise, and using adaptive thresholding for binarization. It also uses morphological operations to enhance text contours before cropping, resizing, and padding the image to fit the model's input dimensions of 32x128 pixels. This results in a clean, consistent image format ready for prediction.

After preprocessing, we load the images and feed them into the trained model. The model predicts the text, which is then decoded using a Connectionist Temporal Classification (CTC) decoder. The predicted text is further refined using the SymSpell algorithm to suggest corrections for spelling errors. The corrected text is displayed alongside the processed image, offering visual feedback on the predictions.

Ground truth labels for the images are provided in a dictionary, which is used to compute the CER and WER. CER measures the number of character-level errors relative to the total number of characters, while WER assesses word-level errors. These metrics are essential for quantifying the effectiveness of the model and spelling correction system. The results, including the corrected texts and evaluation scores.

Based on the CER and WER values, SymSpell with Greedy encoding performed the best, followed by the SymSpell with Beam Search and lastly PySpell checker.

This approach combines optical character recognition (OCR) with spelling correction, making it a robust solution for extracting and refining text from handwritten images, with practical applications in various domains where text accuracy is crucial.

4.2 Pseudo-labelling

Pseudo-labeling is a semi-supervised learning technique that involves using the model's predictions as labels for unlabeled data. The idea is to generate "pseudo-labels" for data that does not have ground-truth labels and use those pseudo-labels as if they were real labels during training. The objective is to

improve the model's performance by leveraging unlabeled data in addition to the labeled data. After training an initial model, we use predictions on the test data as "pseudo-labels" and added them to the training dataset.
We utilize Character Error Rate (CER) and Word Error Rate as key metrics to evaluate the performance of our optical character recognition (OCR) model. CER will provide insights into the percentage of incorrectly predicted characters relative to the total number of characters in the ground truth, helping us identify the model's performance in recognizing text accurately. Word Error Rate (WER) is used to measure the accuracy of predicted text at the word level. It provides insights into how well the model recognizes entire words, not just individual characters, making it a vital complement to Character Error Rate (CER). WER is useful in tasks where recognizing full words correctly is more important than individual character predictions, which is often the case in natural language processing or text recognition systems. By combining these metrics, we can comprehensively analyze the model's strengths and weaknesses, guiding future improvements and refinements.

5 Results

The initial OCR model was trained the model using a dataset of 10,051 grayscale images, each sized 32x128 pixels, over 60 epochs with a batch size of 8, resulting in 1,257 iterations per epoch. Our results indicate that the training accuracy consistently improved over the epochs, demonstrating that the model effectively learned from the training data. However, the validation accuracy exhibited fluctuations, which suggests potential over fitting. The validation loss remained high and did not show signs of improvement, raising concerns about the model's ability to generalize to new data. While the model showed promising results on validation images from the training set, it underperformed on our test dataset.
Training Results - Metrics at epoch 60:
Final Training Loss: 1.1948
Final Training Accuracy: 60.46
Validation Loss: 5.3957
Validation Accuracy: 42.14
We visually compared a few of these predicted sequences with the actual original text to assess the model's performance. Fig 1 shows the OCR model performing very well on the validation dataset.

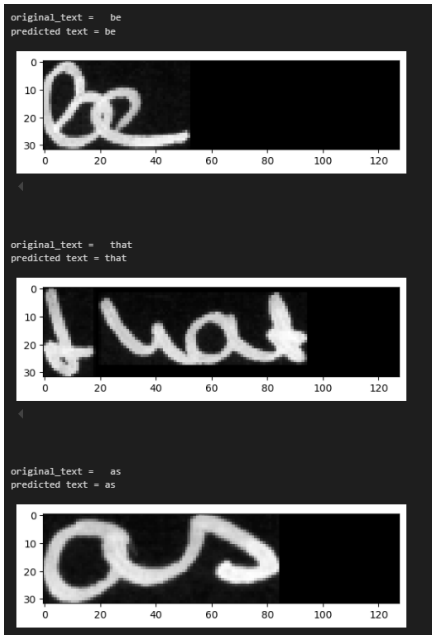

Figure 1: Predicted text on Validation Images

We tested our own handwritten images on the model. After fine tuning the model using word embedding and Spell Checkers, the model's performance was enhanced

Image predictions after correction by SymSpell, the model adapted to a broader range of text styles in the test set. Figure 3 shows some examples of how the model correctly predicts some handwritten images which was earlier being incorrectly predicted.
The Evaluation Metrics used were **Character Error Rate** and **Word Error Rate**. Before fine-tuning the model, the values of CER was 14.291% and WER was 75.862%. After using the Spell Checker algorithm, the values of CER is **18.198%** and WER is **47.932%**.

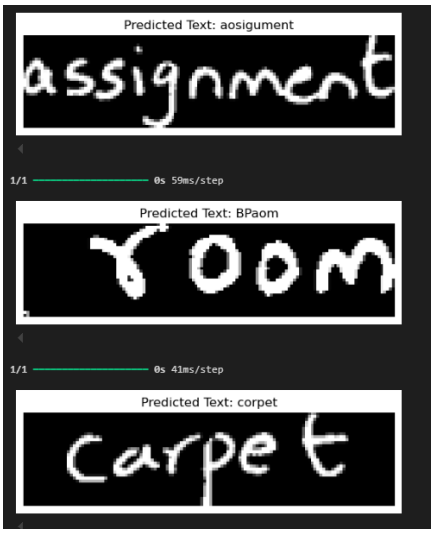

Figure 2: Predicted text on Testing Images

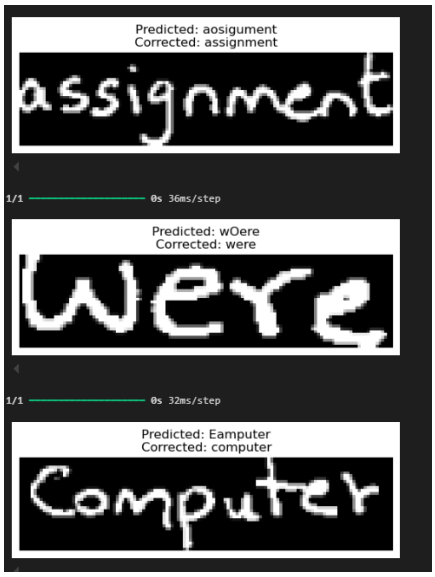

Figure 3: Predicted text on Testing Images after correction

6 Conclusion

The results show some interesting and contrasting insights into the prediction and correction process:
Predicted Text CER (14%): This indicates a moderate rate of character errors in the initial predictions. It suggests that the character-level prediction model is performing reasonably well but still has room for improvement.
Corrected Text CER (21%): After correction, the CER has worsened. This suggests that the correction process using SymSpell introduced errors or over-corrected, resulting in a higher character error rate.
Predicted Text WER (85%): A 100% WER indicates that none of the predicted texts matched the ground truth at the word level, even if some characters were correct.
Corrected Text WER (47%): The correction step significantly improved the word-level error rate. Although the corrected texts introduced more character-level errors, it helped align more words correctly with the ground truth.

Correction Workflow: The SymSpell correction process helped reduce WER significantly but might have overly adjusted correct characters, leading to a higher CER. This trade-off suggests focusing on refining correction rules to balance these metrics.

7 References

1. Minghao Li and Tengchao Lv and Lei Cui and Shaohan Huang and Furu Wei and Zhoujun Li and Ming Zhou.TrOCR: Transformer-Based Optical Character Recognition with Pre-trained Models. 2022. https://arxiv.org/pdf/2109.10282

2. Martin Kiss, Karel Bene˘, and Michal Hradi˘. AT-ST: Self-Training Adaptation Strategy for OCR in Domains with Limited Transcriptions. 2021. https://arxiv.org/pdf/2104.13037

3. Youshan Zhang. A Survey of Unsupervised Domain Adaptation for Visual Recognition. 2021.https://arxiv.org/pdf/2112.06745

4. Vishal M. Patel, Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Visual Domain Adaptation: A Survey of Recent Advances. IEEE Signal Processing Magazine, 32(3):53-69, 2015. https://engineering.jhu.edu/vpatel36/wp-content/uploads/2018/08/SPM_DA_v9.pdf

5. Atman Mishra, A Sharath Ram, Kavyashree C. Handwritten text Recognition Using Convolutional Neural Network. 2023. https://arxiv.org/pdf/2307.05396

6. Gaurav Patel, Jan Allebach, Qiang Qiu. SeqUPS: Sequential Uncertainty-aware Pseudo-label Selection for Semi-Supervised Text Recognition. 2022. https://arxiv.org/pdf/2209.00641v2

7. Md. Abdullah-al-Mamun. Handwritten-Text-Recognition-Tesseract-OCR 2022. https://github.com/bdstar/Handwritten-Text-Recognition-Tesseract-OCR/tree/main?tab=readme-ov-file