CSCI 1430 Final Project Report:Text Extraction for Translation

Team name: Keigo Hachisuka, Colby Porter
TA name: Joel Manasseh
Brown University

Abstract

This paper presents an approach to address the challenge of text extraction from images for translation purposes. Leveraging Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) along with Connectionist Temporal Classification (CTC) loss, the proposed model helps in better understanding transcribing handwritten text. Through experimentation with the IAM dataset [5] consisting of handwritten text images, the model demonstrates approximately 67% validation accuracy in text prediction. The model holds the potential for a wide array of applications. There is also a focus on segmenting input images into individual words, such that the images can be passed along to the model. The two mechanisms are the foundation for developing an image text translation pipeline. The findings of this study contribute to the understanding of technology, which closes the gap between handwritten and digital text, paving the way for enhanced communication, accessibility, and efficiency.

1. Introduction

Handwritten text recognition serves multiple use cases, such as digitizing historical documents, enhancing accessibility for visually impaired individuals, and improving data entry processes across various industries. Though current implementations exist, this project seeks to understand the underlying mechanisms of text extraction better, focusing on the challenges posed by the variable nature of handwritten documents. Recognizing text from handwritten sources presents unique difficulties due to the diverse handwriting styles, including cursive and print, as well as each individual's unique styles.

To build a successful model given these issues, the IAM dataset [5], a diverse collection of handwritten samples from over 600 writers, is utilized. This dataset offers a wide range of handwriting styles, making it suitable for training our model. Building upon research done in *CNN-RNN BASED HANDWRITTEN TEXT RECOGNITION* [2], our model architecture integrates Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) with a Connectionist Temporal Classification (CTC) loss function. These components allow the model to extract text from handwritten images effectively, which in turn aids translation into a given language.

By addressing the challenges of the variation in handwriting, this paper aims to study the effectiveness of our proposed model in extracting text from handwritten images. Furthermore, by capturing the underlying patterns of the English language, our model holds the potential to predict words accurately, even with a dataset that does not contain all possible words.

This paper discusses dataset selection, data preprocessing techniques, model architecture, loss function, experimental results, challenges, and next steps. Through this research, the aim is to contribute to the understanding of text extraction from handwritten images. This enhances accessibility and efficiency across various domains, such as document digitization, assistive technology for the visually impaired, and streamlining data entry processes in a wide range of industries.

2. Related Work

The primary resource for this project is the article, *CNN-RNN BASED HANDWRITTEN TEXT RECOGNITION* [2]. This article discusses how the IAM dataset [5] can be leveraged to generate a text extraction model. This paper discusses the general preprocessing and architecture to achieve a 98% accuracy. More research into the CTC loss was required, thus the article *Sequence Modeling With CTC* [3] was used to gain a better understanding. The implementation of the CTC loss function was also unclear, thus an example of audio recognition, *Automatic Speech Recognition using CTC* [1], was studied to get a better understanding of the syntax. The OpenCV package is used for line and word segmentation, and the documentation was accessed to better understand the functionality. Namely, research on Otsu's

algorithm [4] was done to gain an understanding of various approaches to finding line and word boundaries. The Python packages spellchecker and googletrans were installed for auto-correction and translation.

3. Method

Building on the ideas from the reference paper [2], this project is structured into two parts. Initially, the focus is on training the model, which involves preprocessing images and optimizing the model architecture. We then integrate the model into a pipeline for translating input images. The first part consists of preparing the dataset and training a model capable of extracting word text from an image. This involves preprocessing images to standardize their format and developing a model architecture that can effectively learn from the dataset. We leverage the referenced paper to help guide the approach. In the second phase, we utilize the trained model to create a pipeline for translating text from input images. This involves integrating the model into an end-to-end system that can process images and generate accurate translations.

3.1. Model Training

3.1.1 Dataset and Preprocessing

The IAM dataset [5] contains 115320 words and serves as the data for training our model. To ensure data quality, the 96456 images labeled as valid by the dataset were used, while the 18864 images labeled as erroneous were ignored. Given the extensive volume of valid data, manually verifying the erroneous label's accuracy was deemed unnecessary. All unique characters in the dataset were extracted and stored as a string shown in Figure 1. The longest possible string, 21 characters, was also calculated to serve as a vital part of the label encoding.

Preprocessing begins with converting the image to grayscale and resizing to a standardized shape of (32, 128) while preserving the aspect ratio. If necessary, padding or reshaping is performed to maintain consistent dimensions in all images. Additional preprocessing steps involve normalizing pixel values to be between [-1,1] and encoding the labels into numerical representations by mapping each character to its location in the string shown in Figure 1.

"!\"#&'()*+,-./0123456789:;?ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"

Figure 1. String containing all unique characters in the dataset

3.1.2 CNN Layers

CNN layers play an important role in extracting spatial features which are required for effective text recognition from handwritten text images. The CNN component consists of five grouped layers, each comprising of Conv2d, batch normalization, and max-pooling layers, with ReLU activation. Filter sizes progressively increase as follows: [32, 64, 128, 128, 256], and the kernel sizes were 5 for the first two groupings and 3 for the remaining three. These layers produce an output that can be reshaped to (32, 256), as suggested in the paper. This output will be passed along to the RNN layer.

3.1.3 RNN Layers

The Recurrent Neural Network (RNN) component of the model serves its purpose by capturing sequential patterns within words. When paired with the spatial features extracted by the CNN layers, it provides a far more robust model. Two bidirectional LSTM layers, each with 256 units, are used to capture sequential patterns within words. Bidirectional LSTM enhances contextual understanding in both directions, which is beneficial for words, where each letter is dependent on the surrounding characters. The output shape is adjusted to (32, 79) using a Dense layer. The 79 represents the number of unique characters in the dataset, 78 characters, along with an extra "blank" character used for CTC loss calccu-lation. [3]. RNN layers play a crucial role in capturing sequential patterns within handwritten text, complementing the spatial features extracted by the CNN layers. By analyzing the relationships between characters, the RNN contributes to the accurate prediction of text from handwritten text images.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 32, 128, 1)]	0
conv2d (Conv2D)	(None, 32, 128, 32)	832
batch_normalization (BatchNormalization)	(None, 32, 128, 32)	128
max_pooling2d (MaxPooling2D)	(None, 16, 64, 32)	0
conv2d_1 (Conv2D)	(None, 16, 64, 64)	51264
batch_normalization_1 (BatchNormalization)	(None, 16, 64, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 8, 32, 64)	0
conv2d_2 (Conv2D)	(None, 8, 32, 128)	73856
batch_normalization_2 (BatchNormalization)	(None, 8, 32, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 8, 16, 128)	0
conv2d_3 (Conv2D)	(None, 8, 16, 128)	147584
batch_normalization_3 (BatchNormalization)	(None, 8, 16, 128)	512
max_pooling2d_3 (MaxPooling2D)	(None, 8, 8, 128)	0
conv2d_4 (Conv2D)	(None, 8, 8, 256)	295168
batch_normalization_4 (BatchNormalization)	(None, 8, 8, 256)	1024
max_pooling2d_4 (MaxPooling2D)	(None, 8, 4, 256)	0
reshape (Reshape)	(None, 32, 256)	0
bidirectional (Bidirectional)	(None, 32, 512)	1050624
bidirectional_1 (Bidirectional)	(None, 32, 512)	1574912
dense (Dense)	(None, 32, 79)	40527

Total params: 3,237,199
Trainable params: 3,235,983
Non-trainable params: 1,216

Figure 2. The layers of the model

3.1.4 Loss Function: CTC

After comprehensive research, the CTC loss function was selected for its ability to effectively handle the unpredictable nature of word length. Unlike categorical loss functions, CTC loss permits flexibility in input and output lengths, considering all valid label permutations during training. CTC loss allows the model to handle variable-length output

sequences by introducing ”blank” labels and collapsed repeated labels. [3]. The CTC loss function is paired with an Adam optimizer which allows the model to train leading efficient word prediction.

This model's approach leverages both spatial and sequential information, enabling the model to effectively extract text from handwritten images while accommodating the inherent variability in handwriting styles. The CTC loss function is important, as it enables effective training and accurate text recognition from images. By handling variable-length output sequences and exploring all possible alignments between input and output sequences, CTC loss enhances the model's ability to accurately predict text. [3]

3.2. Image Translation Pipeline

Once the model trains, it is added into a pipeline to facilitate the translation of text in images. The pipeline consists of several steps aimed at segmenting the image into lines, then words, extracting text, spell-checking, combining words into a phrase, and finally translating the text. The steps for extracting lines and words are similar, with the only difference being the scale at which the features are extracted.

3.2.1 Line and Word Segmentation

The first step involves determining the boundaries of the line or word. To accomplish this, the image is initially converted into a binary format using Otsu's thresholding algorithm. [4] This algorithm dynamically calculates the threshold value to separate text from the background and any noise. The result can be viewed in Figure 3.

Figure 3. Threshold Image

Next, a structuring element is defined to identify boundaries within the threshold image. This step helps in finding the boundaries of lines and words. However, due to potential variability in image shapes, a generalized formula for the structuring element may not suffice for all scenarios.

Following this, the binary image is dilated to enhance the visibility of key elements, such as words. Dilation expands the boundaries of the text, making them more distinguishable as seen in Figure 4.

Figure 4. Dilated Image

Contour detection is then applied to identify the bounding regions of individual lines or words within the image. This process utilizes the dilated image to extract contours, which represent the boundaries of a given feature. The dilation enhances the key features of the image, which helps better identify such boundaries. Although not explicitly drawn when the image runs through the pipeline, the bounding boxes are shown in Figure 5.

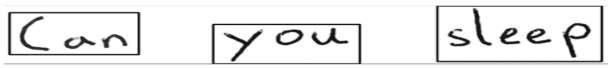

Figure 5. Image with contours visualized

3.2.2 Text Extraction and Spell Checking

Once the contours for the words are identified, the corresponding regions are extracted and passed through the trained model to extract text. However, due to potential inaccuracies in the extracted text, a spell-check package is used to fix any minor issues. Although this is an efficient approach to fixing small errors, there are potential pitfalls which will be discussed later.

3.2.3 Translation and Display

After extracting the words, they are grouped into a single phrase. The extracted text is then translated into the desired language using the Googletrans package. The most common pixel value (which typically is the background color) is used to cover the original text. The translated text is then displayed on the original image, replacing the original text. This process is repeated for all lines found within the image, effectively translating the text.

4. Results

4.1. Model Training

4.1.1 Training Computation

The model underwent training for 100 epochs with a batch size of 500, which took approximately 18 hours. The duration shows the computational complexity of the model, which was further emphasized by the large dataset. Throughout the training, intermediate models were saved based on the best validation loss, training loss, and validation accuracy. The training and validation accuracy/loss are shown in Figure 6.

4.1.2 Training Results

During training, the model's training loss stabilized around 0.15, while the validation loss stabilized around 3.7. This large gap between the two values suggests a degree of

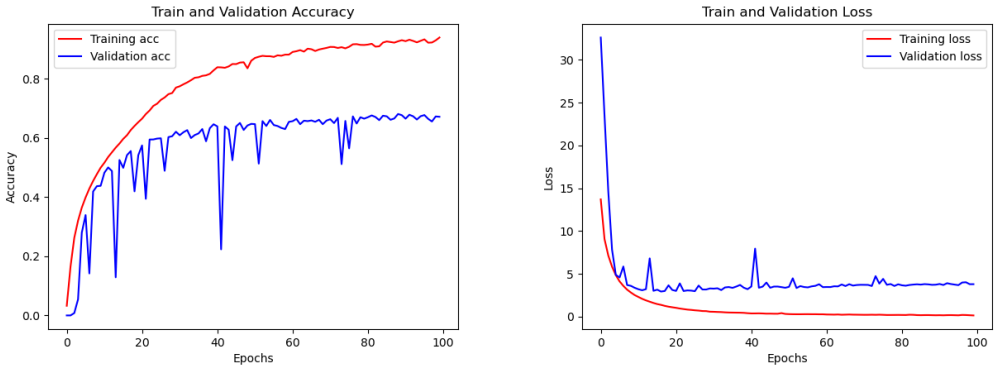

Figure 6. *Left:* Accuracy *Right:* Loss

overfitting. Similarly, the training accuracy plateaued at approximately 93%, while the validation accuracy stabilized at around 67%. Significant drops in validation accuracy during training also indicated overfitting issues. The best loss and accuracy for both training and validation are shown in Table 1. Given these results, the trained model is used to attempt to predict handwritten text in images. The experiments explored later will showcase not only the success of the model, but also its limitations.

Table 1. Training and Validation Metrics

	Best Loss	Best Accuracy
Training	0.13	0.94
Validation	2.92	0.68

4.1.3 Word Segmentation

The line and word segmentation section of the pipeline proved to be successful. Given varying inputs (multi-line, single word), the program can successfully extract the words to be passed along into the model. The high-level steps of the extraction are visualized in Figure 7.

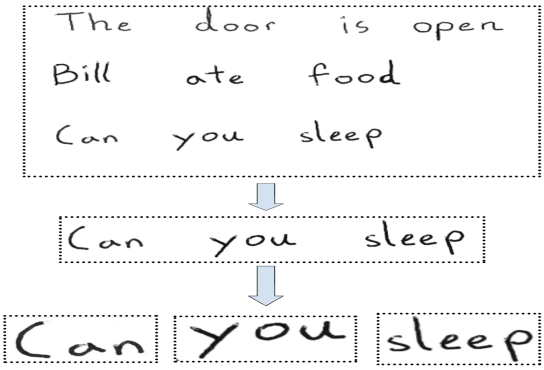

Figure 7. High Level Idea of Word Segmentation

4.1.4 Model Performance

To determine the performance of the model, experiments were run with hand-made text blocks. Various versions of experiments were run to determine the model's performance and limitations.

Multi Line Input: Figure 8 is an example in which multiple lines are inputted to showcase the pipeline's ability to segment words, extract text, and translate. In this experiment, the model can accurately extract the correct word, and thus the phrase is correctly translated.

Figure 8. All lines are correctly extracted and translated

Overfitting: As discussed in 4.1.2 the model's results showed that that the model had likely overfit. This is highlighted when the text "hi" is passed into the model. The model struggles to predict a seemingly simple word as it predicts the word to be "his" as shown in Figure 9.

Upon further investigation, the dataset consisted of a significant number of three-letter words starting with "hi_" and lacked the word "hi". Therefore, the overfit model will expect a third letter causing this issue.

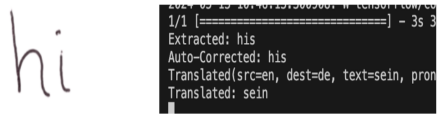

Figure 9. ”hi” being incorrectly labeled as ”his”

Long String Performance The previous experiments focused on shorter-length words however, the model is capable of extracting longer strings with varying success. The model was able to extract ”disturbance” without any issue (Figure 10) but struggled with the word ”pineapple” as it predicted the word to be ”picepple” (Figure 11). This error was minor as the spell checker could fix it, however, there were other cases where the prediction was not able to be fixed by a spell checker.

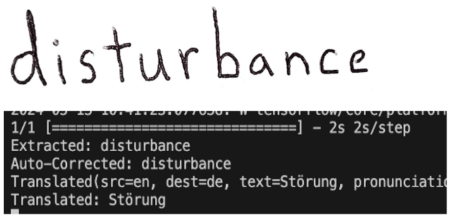

Figure 10. ”disturbance” image text extraction

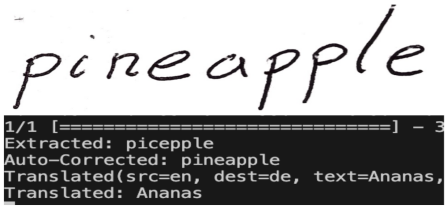

Figure 11. ”pineapple” extracted as ”picepple”

Varying writing style: The last experiment focused on seeing how the same model handles the same word, ”my” in different handwriting styles. The results are showcased in Figure 12.

IMG	Extracted	Autocorrected	Translated (de)
	my	my	mein
	sry	say	sagen
	my	my	mein
	mey	my	mein

Figure 12. ”my” with different handwriting

It is observed that the two lowercase print ”my” images were able to be accurately predicted. The cursive version of ”my” was incorrectly predicted but was close enough to be fixed with a spell checker. When observing the cursive image, the model expecting the letter ”e” is not egregious as there is some semblance of the letter within the image. The model struggled with the all capital ”MY” as it predicted the text to be ”sry” which is not able to be accurately fixed with the spell checker. This is again due to the model being overfit. The lack of words that are all capitalized in the dataset made the model struggle to determine this word. There were successful cases of text prediction, however, the incorrect predictions highlighted the sensitive nature of this model.

As seen from the various experiments, the over-trained nature of the model is highlighted which was expected given the loss and accuracy results from model training. However, there are cases where the model successfully predicts the text in the image. This sensitivity makes the model difficult to use in some scenarios and is an issue to be discussed in the next steps.

4.2. Technical Discussion

4.2.1 Tradeoffs

The significant time required to train the model raises questions about the trade-offs between dataset size, computational resources, and training efficiency. While utilizing a larger and more comprehensive dataset improved the model’s performance, it came at the cost of prolonged training times. Despite the model achieving stability in terms of accuracy and loss, the extended training made it difficult to tweak the model for further improvements. Additionally, the issue of overtraining highlights the need for other approaches to be considered.

The reliance on an external auto-correction package to fix minor inaccuracies in text extraction introduces potential trade-offs in translation accuracy and reliability. While auto-correction effectively handles small deviations in extracted words, it may introduce errors by incorrectly fixing a mistake which will lead to mistranslations. For example, if the word ”sick” is mispredicted to ”sich”, it can be autocorrected to ”such” instead of ”sick”. This will change the context of the phrase and in turn, cause the text to be mistranslated. This issue is concerning when considering the social implications of mistranslated text.

4.2.2 Changes

To address the challenges posed by prolonged training times and overfitting, exploring techniques to optimize the training process is vital for future success. This may involve fine-tuning hyperparameters, implementing early stopping

mechanisms, or utilizing more efficient model architectures. Additionally, leveraging techniques such as transfer learning or data augmentation could enhance model performance while reducing training time.

Given the limitations of auto-correction, using context-aware autocorrection mechanisms is a potential solution. Context-aware autocorrection algorithms could analyze surrounding text to determine the correct word in cases where multiple changes are valid, improving the accuracy of corrections and by default, translations. Although not perfect, these mechanisms can increase the reliability of the pipeline.

5. Challenges

The initial phase of the project encountered significant challenges related to dataset selection and model training. Attempts with datasets such as the IIT 5k-word dataset and a subset of synthetically generated words were ineffective, with models struggling to learn. This led to extensive time and computational resources being used in preprocessing datasets and tweaking model architectures. These issues were exacerbated by the computationally expensive nature of training with these datasets.

Various preprocessing techniques and model layers/parameters were explored in an attempt to improve performance. However, the lengthy training times made it difficult to conduct thorough experimentation to determine optimal parameters. To mitigate this issue, a smaller subset of the IAM dataset was utilized for preliminary experimentation, providing insights into model performance without exhausting computational resources. This is not an ideal scenario as a model performing well on a small dataset does not necessarily transfer over to a larger dataset, however, it was an attempt to find the optimal layers/parameters.

6. Next Steps

Improving the model's generalization capabilities by mitigating overfitting should be prioritized. Exploring various model architectures, regularization techniques, and hyperparameters can help achieve a better balance between model complexity and performance. If this is successful, the extraction of entire phrases from images may be worth exploring. Expanding the model's capabilities to encompass languages beyond English presents an exciting opportunity for future research. Particularly, languages with non-Latin characters pose unique challenges and opportunities for innovation. Adapting the model to handle diverse linguistic structures and character sets can broaden its applicability and impact.

7. Socio-Historical Impact

One of the most significant social impacts lies in the potential for cultural misrepresentation and erasure. Inaccurate text extraction and translation may lead to the distortion or loss of cultural nuances, identities, and histories existing within handwritten texts. This can emphasize stereotypes and misconceptions of certain communities, which harms the efforts toward cultural understanding and appreciation.

Inaccurate translations resulting from incorrect text extraction can also cause linguistic discrimination. Languages spoken by underrepresented communities may be particularly vulnerable to misinterpretation or neglect. This can exaggerate existing inequalities and contribute to the demise of linguistic diversity, cultural heritage, and indigenous knowledge systems.

Misinterpreted texts, especially in legal or official documents, can cause significant harm. Many of these scenarios carry significant legal ramifications and mistakes can lead to large financial losses. In social situations, mistranslations can lead to misunderstandings, conflicts, and increased tensions, hindering meaningful communication and collaboration.

Given these socio-historical implications, stakeholders, especially those who develop such algorithms must carry the ethical responsibilities to mitigate the risks associated with inaccurate text extraction and translation. This requires rigorous validation, quality assurance, and accountability measures throughout the development and deployment of text extraction systems. Collaboration with all stakeholders, including language experts, cultural representatives, and impacted communities, is vital to ensure responsible AI development.

8. Conclusion

To solve the issue of text extraction within images for translation, multiple approaches were considered. However, after some research and thoughtful consideration, combining CNNs/RNNs with CTC loss provided the best solution. Paired with a comprehensive dataset of handwritten texts, a model that can extract text with around 67% accuracy was generated. Although imperfect, this lays the foundation for transcribing handwritten text to a digital format. This system can digitize documents, serve as assistive technology for the visually impaired, or streamline processes in a wide range of industries. The application to this program is endless and can be leveraged in varying contexts. While further refinement and optimization are necessary to improve accuracy

and performance, the development of this text extraction and translation system showcases a significant advancement in digital communication and information accessibility.

References

[1] François Chollet and Keras contributors. Audio-based speech recognition using connectionist temporal classification. https://keras.io/examples/audio/ctc_asr/, 2024. 1

[2] S. Keerthi Venii P. Akshaya G.R. Hemanth, M. Jayasree and R. Saranya. Cnn-rnn based handwritten text recognition. *ICTACT JOURNAL ON SOFT COMPUTING*, 12, 2021. 1, 2

[3] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Sequence modeling with ctc, 2017. Distill, Journal of Machine Learning Research. 1, 2, 3

[4] Satya Mallick. Otsu thresholding with opencv, Year the tutorial was published. LearnOpenCV.com. 2, 3

[5] U. Marti and H. Bunke. The iam-database: An english sentence database for off-line handwriting recognition. *Int'l Journal on Document Analysis and Recognition*, 5, 2002. 1, 2

Appendix

Team contributions

Please describe in one paragraph per team member what each of you contributed to the project.

Keigo Hachisuka I worked on determining the dataset. After determining the dataset I researched the best approach the utilize this dataset to create a successful model. I preprocessed the images, finding the valid images and preprocessing them to be passed along to the model. I developed the model which the data was trained on. After training the model, I create the pipeline of extracting lines, then words from an image to be passed into the model. I worked on developing the best possible approach to group the words to be translate. I created the experiments to better understand the model's performance. I worked on putting together the poster as well as writing the technical sections of the report as well as helping with the socio-historical issues.

Colby Porter I worked on helping find datasets to use as well as finding various API packages for translation and spell check. I also worked on the socio-historical work.