DEGREE PROJECT

Handwritten recognition of English crossed out words using YOLO

Nazrul Islam

Master Programme in Applied AI
2024

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering

LULEÅ UNIVERSITY OF TECHNOLOGY

[This page intentionally left blank]

Abstract

Handwritten text is widely used to record information, whether it's in registries, forms or any sort of document where information is captured. Different text recognition techniques are used widely to read printed text, however reading a crossed-out (striked-through) text is a challenge. This thesis is aimed at building a machine-learning model to be able to read crossed-out handwritten text in English. Similar models are created to read clean handwritten text and then compare the results of different models. In addition, a dataset is also created to train the model. Here, we have used the technique mentioned in [6] to create a training/test dataset. This technique automates the production of a character image training set by determining character locations in a word based on typical character size. We have used the IAM [9] word dataset to train/test the model. We have also used the technique presented in [10] to train a model using You Only Look Once (YOLO) to spot characters in crossed-out words. Here, a deep learning model (YOLOv8 [19]) based on an object detection architecture is used to detect a character from an image file and the detected characters are used to form the word. In this technique, we trained the model to detect each character as an object in an image.

Keywords: Offline handwritting recognition, crossed-out words, character spotting, IAM dataset, YOLOV8, autonomous tagging.

Contents

Acknowledgments

I would like to thank Prof. Elisa H Barney Smith who guided me all the way to complete this thesis project, provided all the necessary guidelines and resources. In addition, I would like to thank Chang Liu, Simon Corbillé and Ludvig Pålsson who helped me at different times.

Luleå, September 2024
Nazrul Islam

Chapter 1

Thesis Introduction

1.1 Background

A handwritten document is a very common way to communicate and store information, even in the current era of a digital world. Court proceedings, application forms in schools, colleges and universities, and medical records are a few examples where handwritten documentation remains prevalent. In addition, there is a large volume of historical documents that needs to be recognized.

Understanding handwritten text from images remains a challenge and is a current subject for research in the domain of digital document analysis as it's used in many applications such as digitizing handwritten manuscripts [7][8], postal automations [14], etc. Hence, development of techniques to recognize such handwritten text is very important. Major advancement has been achieved in recent times. Starting from handwritten numerals and hand-printed characters, promising results with fairly high accuracy have been obtained in recognizing free-form running texts in Roman-based western as well as some oriental scripts [15] [13].

One limitation with other handwriting recognition techniques is that they consider ideal input, i.e., documents containing no writing errors. However, a free-form handwritten manuscript may have misspelled or uncommon words which were struck out by the author during writing. The appropriate word is usually written next to the struck-out one. This is one of the common editing operations done while writing. We think it's even important to read crossed-out text too, for real-time applications like handwritten character recognition, writer identification, digital transcription, forensic applications, and historical document analysis.

1.2 Objective

The primary objective of this thesis is to explore the potential of using an object detection model such as YOLO [19] to recognize handwritten crossed-out texts. The results are

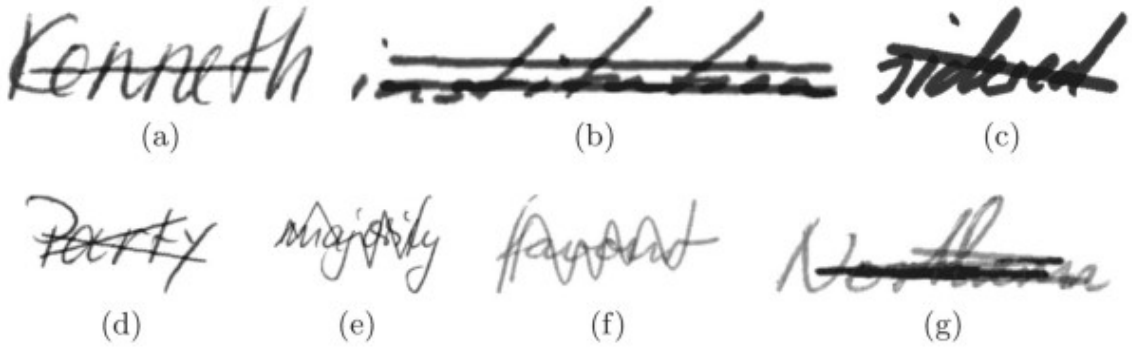


Figure 1.1: Examples of different crossed-out types considered a) Single-line, b) Double-line, c) Diagonal, d) Cross, e) Zig zag, f) Wave, g) Scratch. Figure from [4]

intended to contribute to the understanding of how this problem can be handled in future research work.

One technique to recognize text is through a segmentation free handwriting recognition technique (Character spotting) to spot characters from words and thereby predict the word. The technique is based on an object detection approach which is referred to as character spotting in [6], [5]. We think that this approach when using an object detection model such as YOLO can detect objects even with occlusions, and for texts the lines in crossed-out images are occlusions. In this thesis, we explore how robust the Character Spotting technique is if the words are crossed-out.

We have used the IAM dataset [9], which contains a vast collection of handwritten text. We further processed these images from IAM using techniques in [3] to create crossed-out images. The algorithm creates seven different types of crossed-out images as shown in figure 1.1. We then explore how robust the recognition model is in the presence of each of these types of cross-out. Then we will see how re-training the classifier on the characters containing the cross-out affects the recognition of each.

1.3 Ethical consideration

The dataset used in this thesis is publicly available where data cannot be linked back to any individual contributor. This reduces the privacy concern in using this dataset.

Even though there's a diverse population of 657 contributors who contributed to this dataset to have different variations of handwriting styles, there could be biases similar to any other dataset. It is very important to acknowledge this as it impacts the performance of the recognition system. For example, the system could be less effective for styles that are underrepresented in the dataset.

Chapter 2

Literature Review

2.1 Overview

Handwritten text recognition is the ability of a computer system to read handwritten text from sources such as paper documents, images and other devices. In such text recognition systems, text is recognized solely from digitally acquired image data, usually from a scanner or a camera.

There are different approaches followed by researchers to recognize handwritten texts, a) Segmentation-Based approach and b) Segmentation-Free approach.

Segmentation-based approaches usually go through a two-stage process: first segmenting individual characters from a word image and then applying a classifier to recognize each isolated character. The process of character segmentation is often the most challenging task in this process. Another method of this approach is N-gram modeling, which is used to aid recognition by segmenting words into N adjacent characters, with N usually between 2 (bigrams) or 3 (trigrams). This capitalizes on handwriting being statistically more similar when people write a set of characters than how they write individual characters, since the neighboring characters influence the writing process.

Segmentation-free approaches attempt to recognize a whole word instead of segmenting each character as a small chunks. In this approach, the model is trained to learn a set of vocabulary, hence usually the performance of this type of model decreases with the increasing number of words to identify. LSTM and HMM are considered the best tools for text recognition based on this approach.

Another technique in the category of segmentation-free approaches is based on character spotting, which uses an object detection network to locate and identify each character element in a word image, which is then assembled in a transcription.

While a lot of attention and progress has been made in reading a text, there are only a few solutions related to reading crossed-out text. Arnad et al.[16] introduced a system for detection of stroke-types using a single network architecture based on a Generative Adversial Network (GAN) [2].

Heil et al.[4] has also introduced a method to identify stroke-types and remove the stroke, ultimately providing a clean image. In their paper, the authors have proposed a technique based on Cycle Consistent Generative Adversial Network (CycleGAN)[20] and extended with an attribute-guiding feature to remove cross-out. The clean images could potentially be fed into another model to recognize the characters.

In the same paper, Heil et al.[4] also presented an approach to generate a cross-out dataset based on the IAM database [9]. In addition, they have created a new dataset consisting of genuine samples of cross-out handwritten words and classify those images into one the seven categories as in figure 1.1.

Chauduri et al.[1] also proposed a technique to identify different types of strike-types and deletion of the strokes so as to make clean words that can be fed into any other classifier or OCR module (which will help in automatic transcript generation).

Chapter 3

Character Spotting

3.1 Overview

Character Spotting is an approach defined in [5] which uses an object detection network to spot different character elements in a word image. This approach makes the training process rely on a limited set of possible characters instead of learning a full set of vocabulary.

3.2 Concept

The basic idea behind the Character Spotting method is first to identify and locate individual characters in a word image and then to use the location information to combine the characters to produce a transcription [5]. This concept is explained in figure 3.1.

For example, given the word "Good", the object detection network attempts to find each letter from "A/a" through "Z/z" in the word. For implementations such as YOLO,

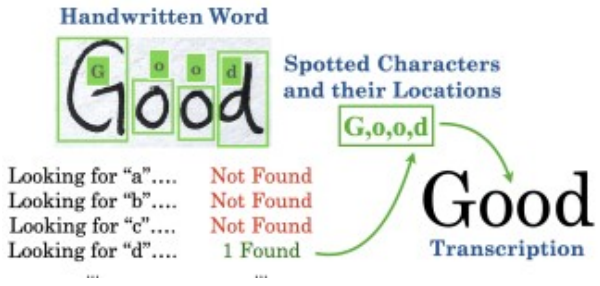


Figure 3.1: Basic concept of character spotting recognition method. An object detection network attempts to find character matches in the word and the transcription is formed with the detected character classes and their relative location information. Figure from [6]

this process is not sequential; the character spotting algorithm runs for all possible characters simultaneously. After it finds all the matches, we use the classes of the characters found (one "G", two "o"s and one "d" in this example) and their relative location information (like the "G" is detected to the left of an "o") to obtain a transcription of the word.

This approach would possibly be a good choice for cross-out text recognition because object detection networks such as YOLO are often able to detect object with overlaps. These type of networks uses techniques such as Non-Maximum Suppression (NMS) [18] during the detection process that can produce fine-tuned boundary box around the localized object. These networks also allow filtering out other possibilities of predicted classes based on Intersection Over Union (IoU) [17] values.

Chapter 4

Datsets used for the experiments

4.1 Overview

We have used the IAM Handwriting Database [9] for training and testing models. This database is structured as:

- 657 writers contributed samples of their handwriting
- 1 539 pages of scanned text
- 5 685 isolated and labeled sentences
- 13 353 isolated and labeled text lines
- 115 320 isolated and labeled words.

The words in this dataset have been extracted from pages of scanned text using an automatic segmentation scheme [21] and were verified manually.

4.2 Auto Tagging

One of the hardest parts of preparing a machine learning method is producing the labeled dataset used in the training process. In this thesis, we have used the auto-tagging technique to create datasets required for training and testing the model.

The concept of Auto-tagging was introduced by Nishatul Majid et al.[6] which automates the production of a character image training set by estimating character locations in a word based on typical character size.

The idea behind autonomous tagging is to produce a training set ready for an object detection network without manually drawing boundary boxes around each character in

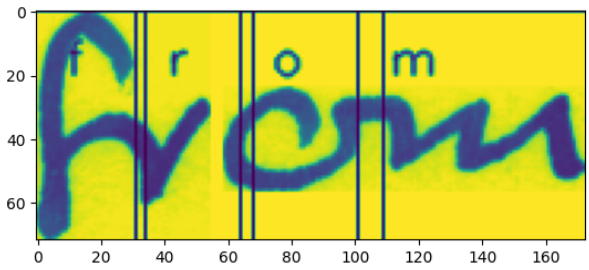


Figure 4.1: The handwritten word "from" is auto-tagged using character classes, and bounding box created. Here, the bounding boxes are created using a estimated size of each character.

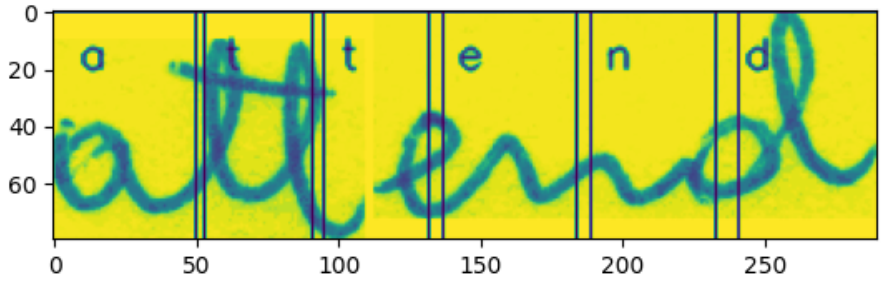


Figure 4.2: In this example, the annotation created by the Autotagging algorithm is not very accurate

each word. Object detection network requires the characters to be defined with bounding boxes and labelled with their class. In figure 4.1, a bounding box is created for each character in the word "from" from a handwritten image. In this technique, every character a-z, A-Z, 0-9 and special characters such as '?', ',', '!', etc. are given a fixed estimated size in pixels. For instance, the width of 'm' is more than 'a' or 'b' which is wider than the letter 'i' or 'l'. Knowing the letters in the word and taking into consideration the possible size of each character, bounding boxes are created.

In the original algorithm by Nishatul Majid et al.[6], the widths were estimated based on measurements from a font library. In the paper, it was shown that the widths could be more approximately estimated for handwritten characters and still produced good recognition results. The heights of the boundary is always set to the height of the word image. In our implementation, we have added 5%-10% buffer to the width depending on the total width of the word image. This means that same letter may have different widths in different images. This often creates a good estimate of the bounding box. Sometimes, it's not very good as in figure 4.2. This is a trade-off against the time taken to create a dataset versus accuracy of the bounding boxes. Had we done the labelling manually, it would have been accurate, but would take huge human effort, while the auto-tag process takes less than an hour for thousands of images. The availability of a larger training set with this method is shown to compensate for the accuracy loss of the inexact position labeling [6].

4.3 Clean Images Dataset

We have picked 38 500 images from the IAM word database to create our dataset. These images were picked at random. We have also removed very small images and images with single character.

We then processed the 38 500 clean images using the auto-tagging algorithm as mentioned in earlier section. As a result, a text file is generated that has the class information along with the co-ordinates of the bounding boxes. This is a typical way to create labels files in YOLO format.

We have then split this dataset into three sub-sets as follows:

- Train set - 25 000 images
- Validation set - 10,000 images
- Test set - 3 500 images

If we look further into the distribution of characters in the training set, we see the distribution of characters as in figure 4.3.

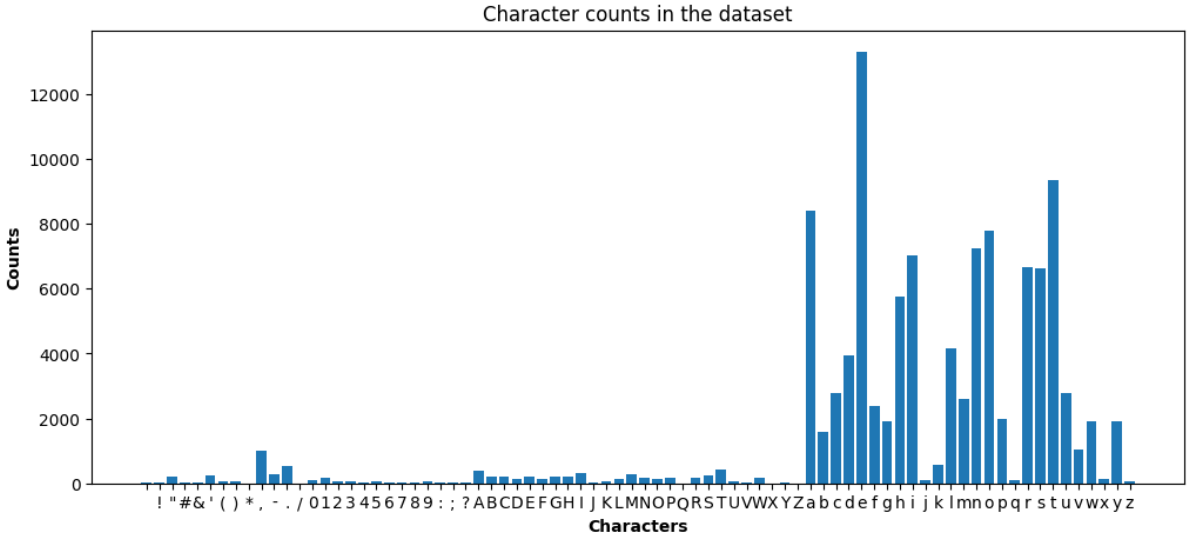


Figure 4.3: Characters counts in the training dataset

4.4 Mixed-Stroke Images Dataset

Using the dataset prepared above, we applied the technique in [3] to create stroked images (crossed-out images). This script creates stroked images where the seven stroke types are as in figure 1.1. The script also creates a balanced dataset such that the number of

images with each stroke types are balanced. Each type of strike is applied to the same set of clean images which allows us to evaluate the cross-out independent of words. In this dataset, the label file remains the same as in the clean images dataset.

For our dataset, table 4.1 shows the number of images by stroke type. Note that there are small differences because the script failed to strike some images.

Table 4.1: Mixed-Stroke Images Dataset

Strike Type	**Training Set**	**Validation Set**	**Test Set**
Singleline	3 557	1 344	504
Doubleline	3 497	1 322	451
Diagonal	3 552	1 352	461
Cross	3 497	1 287	465
Zig zag	3 188	1 212	392
Wave	3 279	1 199	407
Scratch	3 219	1 179	416

4.5 Individual-Stroke Dataset

Further, we created individual datasets, where each dataset contain images with one strike type only. These datasets are prepared by running the script from [3], each iteration creating one type of strike. As a result, we get seven additional datasets, one dataset for each strike type, each containing approximately 24 000 training image, approximately 10 000 validation images, and 3 500 test images.

Chapter 5

Recognition Implementation

5.1 Overview

To implement Character Spotting, we need an object detection network. In this thesis, we have used YOLOv8 [19] as the detection network. In the datasets, the images were auto-tagged with classes which are then fed into the detection network to train the model.

5.2 YOLOv8

YOLOv8 is an object detection algorithm that builds upon the advancements of previous iterations, aiming to further enhance performance and robustness [19]. Leveraging attention mechanisms and dynamic convolution, YOLOv8 introduces improvements specifically tailored for small object detection, the integration of voice recognition techniques enhances the algorithm's capabilities for video-based object detection.

The structure of YOLOv8 is essentially partitioned into two key components: the backbone network and the detection head. The part of the backbone network extracts an assortment of rich features from the input picture at numerous scales. On the other hand, the detection head takes on the task of merging these features and creates different and high-quality forecasts for the bounding boxes.

5.3 Models

To accomodate different scenarios and different hardware environments, YOLOv8 comes with five versions varying in network depths and widths: YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x. From small-scale n-nano to the large scale x-extra large, each version increases the parameter quantity and also resource consumption which also results in improving detection performance. In our experiments, we have used the YOLOv8m network to create the trained model.

In our experiments, we trained the model from scratch, meaning that we didn't use any pre-trained model. All models were trained using the configurations in table 5.1.

Table 5.1: Model Configuration

Configuration	**Value**
No. of epochs	100
Batch size	16
Image size	640
Learning rate	0.00012
Momentum	0.9
Optimizer	AdamW

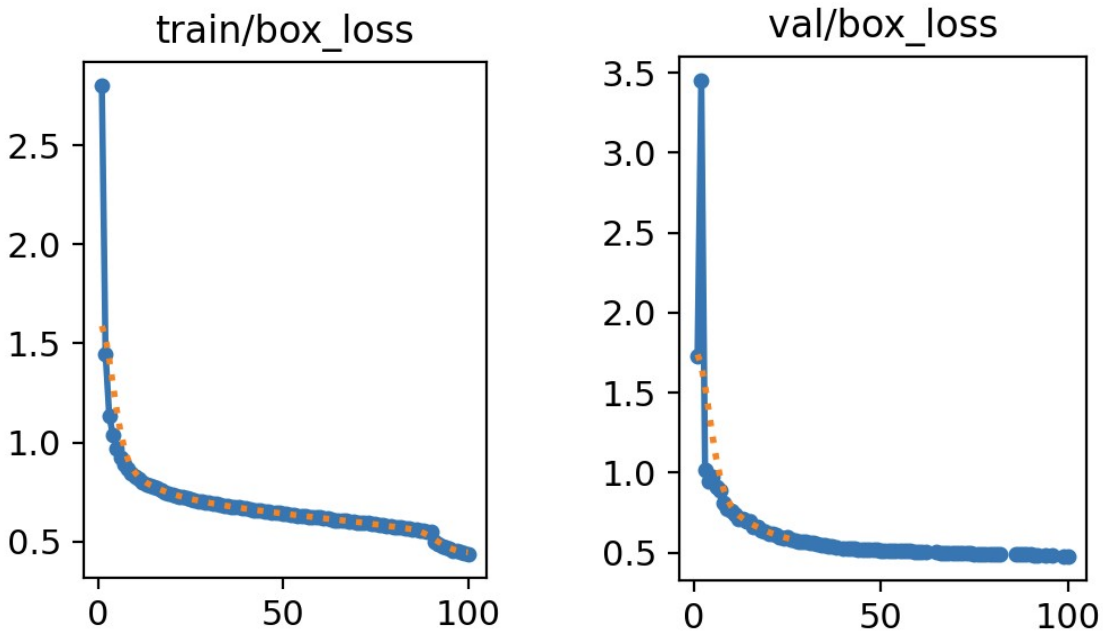
5.4 Experiments

In these experiments, we trained one model for each dataset. This means that we created nine trained models, one trained with clean handwritten images, one trained with dataset with mixed crossed-out types as in 4.1 plus one trained independently with each of the seven strike types.

To monitor the learning activity of the model, we analyzed the learning curves showing box losses. The box loss metric measures the difference between the predicted bounding boxes and the actual bounding boxes of the objects in the training data. A lower box loss means that the model's predicted bounding boxes more closely align with the actual bounding boxes.

In the figure 5.1, we see that model continues to get better and better in each epoch of the training cycle. This curve is from the training of model using the clean images. We have seen similar trend for rest of the training too.

In our experiments, it took around 7 minutes for 1 epoch which means around 11 hours to train one model using 100 epochs when trained in a environment with Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz, NVIDIA GeForce RTX 2080 Ti.



(a) Training Box loss

(b) Validation Box loss

Figure 5.1: Training box loss curve - training with clean images

Chapter 6

Results and Analysis

6.1 Performance Matric

To assess the performanc of the present end-to-end word recognition model, we have used two popularly used performance matrics a) Character Error Rate (CER) [11], and b) Word Error Rate (WER) [12].

6.1.1 Word Error Rate (WER)

Word Error Rate (WER) is a fundamental metric that measures the accuracy of a candidate words by considering three types of errors — substitutions, deletions, and insertions. Word-level errors surface mispredicted words, and it can be useful to visualize common word-level failures to flesh out weaknesses in a model.

Formally, it is defined as the rate of word-level errors in a candidate text.

$$WER = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C}$$

Where,
S is the number of substitutions,
D is the number of deletions,
I is the number of insertions,
C is the number of correct words,
N is the number of words in the reference (N=S+D+C).

6.1.2 Character Error Rate (CER)

Character Error Rate (CER) is another metric that measures the accuracy of a candidate text through substitutions, deletions, and insertions on the individual letters. Unlike word-level errors, character-level errors are useful in surfacing mispronunciations and

erroneous phonemes. CER is defined as the rate of character-level errors in a candidate text.

$$CER = \frac{S + D + I}{N}$$

Where,
S is the number of substitutions,
D is the number of deletions,
I is the number of insertions,
N is the number of characters in the reference texts.

6.2 Results and Evaluation

The object detection model predicts the bounding boxes. Based on the location of each bounding box, we have extracted the transcription for the word which then used to calculate CER and WER.

In our experiments, we crossed all the models with all the testsets, as a result we get 81 test results. Table 6.1 shows the CER and 6.2 shows the WER values.

Table 6.1: Character Error Rates (CER%)

Models	Clean	Mixed	Diagonal	Wave	Zig Zag	Scratch	Cross	Singleline	Doubleline
Clean	19.40	54.61	38.70	54.32	54.30	76.67	57.69	43.94	63.09
Mixed	19.27	17.38	17.80	16.73	14.33	17.16	19.36	17.89	19.06
Diagonal	26.98	35.92	25.94	49.67	42.03	53.77	27.37	25.92	26.69
Wave	23.44	28.77	23.74	18.62	18.11	61.89	28.88	23.03	29.66
Zig Zag	19.42	24.14	18.69	14.41	13.71	58.68	22.59	18.91	22.23
Scratch	30.38	35.77	30.87	55.14	49.02	25.89	30.50	28.58	29.64
Cross	28.52	33.86	26.54	50.88	42.50	38.26	25.78	26.46	27.20
Single	18.55	31.36	20.79	48.87	41.84	49.41	22.60	17.40	18.23
Double	19.31	29.61	19.19	29.61	41.82	40.89	20.65	17.54	17.71

6.3 Analysis

As we obtained nine models in our experiments, and we tested the performance of all models using all the different types of datasets, we got 81 results as in tables 6.1 and 6.2. We have highlighted cells with lower error values towards green and higher ones towards red to get a better overview. The darker the green, the lower the error value and the darker the red, the higher the error value.

Table 6.2: Word Error Rates (WER%)

Models	Clean	Mixed	Diagonal	Wave	Zig Zag	Scratch	Cross	Singleline	Doubleline
Clean	46.60	81.91	73.38	80.45	76.70	92.42	86.48	76.58	88.15
Mixed	46.43	41.60	43.56	37.59	36.99	40.87	43.23	44.84	44.34
Diagonal	53.46	59.82	51.93	68.73	63.43	79.99	53.28	51.56	83.81
Wave	50.67	53.62	50.30	40.46	39.46	83.28	57.75	49.70	58.63
Zig Zag	46.40	48.35	44.71	35.30	34.03	82.93	52.04	45.30	50.24
Scratch	55.83	59.85	58.87	76.51	71.71	47.67	57.22	54.21	55.87
Cross	56.14	57.91	53.28	70.33	63.90	64.94	51.33	52.74	53.28
Single	45.89	58.17	49.79	73.79	68.07	79.24	52.02	43.62	44.34
Double	46.54	55.75	47.32	72.12	67.64	72.73	49.09	43.53	43.74

In tables 6.1 and 6.2, the second row has most of its cells highlighted in green, which means that this model performed better compared to the rest of the models. This model is one that was trained with a dataset that had images of all types of strokes.

Noticeably, all the models that were trained with crossed-out images performed well in reading clean words too. This is due to the fact that YOLO networks are good at detecting objects with occlusions. Here, the cross-outs lines are similar to occlusions.

On the other hand, the first row in the tables 6.1 and 6.2 is more reddish compared to the rest of the rows. This is the model that was trained with a dataset that had images with no cross-outs at all. Here, the model's performance was good in just reading clean words, but not crossed-out words. This is because the model didn't learn anything about any cross-outs.

Compared to the state of the art benchmark, Mondal et. al in [10] obtained WER 29.21% and CER 9.31% for the cleaned images. Compared to our result, it's much better. We think this is because Mondal et al. used a customized dataset that was created manually, hence the ground truth had precise boundary boxes around the characters. They also added supplemented characters to compensate for the low frequency characters as in 4.3. In addition, they carried out 15,000 iterations in their training.

In our training, we have used datasets created automatically using AutoTagging where data labeling isn't accurate (sometimes, the boundary box doesn't accurately encompass the character) and not human verified. We have also used only 100 iterations in our training. From figure 5.1 we see that the models are still learning at 100 epochs, which means more epochs might produce better results.

Chapter 7

Conclusion

7.1 Summary

In this thesis, we have used a different approach to recognize characters from crossed-out word images. In most research works and other OCR models, they first try to delete the stroke and get a cleaner word, and then try to recognize the characters. We have used a technique called Character Spotting that uses an object detection model to recognize the characters. In our experiments, we used YOLO network as the object detection model due to it's versatility in the object detection algorithm space.

One key challenge in our experiment was to obtain a labeled dataset. Most publicly available handwritten datasets do not contain crossed-out words. In this thesis work, we used a technique called AutoTagging to create our own dataset of cross-out words that allowed us to label the dataset in less time, which otherwise might have taken a very long time for a human to label so many images.

We think that the combination of the Character Spotting technique through the use of an object detection such as YOLO model and automatic data labeling technique such as AutoTagging is transformational for handwriting recognition as well as cross-out text. This technique can also be easily extended to adopt and learn other languages.

7.2 Future Work

We think that the dataset aspect in this experiment could be improved. As we have seen, the distribution of characters wasn't balanced, which means the models would learn all the characters well, especially the capital letters and numbers.

Another limitation in our experiments is the number of iterations. We have seen from our training process that the model was still learning at 100 epochs. This suggests that training more epochs might have produced better results.

Further, the YOLO network has many hyper-parameters that could be tuned for better results. More work could be focused in this area too.

References

[1] B. B. Chaudhuri and C. Adak. An approach for detecting and cleaning of struck-out handwritten text. *Pattern Recognition*, 61:282–294, Jan. 2017. doi: 10.1016/j.patcog.2016.07.032.

[2] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath. Generative Adversarial Networks: An Overview. *IEEE Signal Processing Magazine*, 35(1):53–65, Jan. 2018. doi: 10.1109/msp.2017.2765202.

[3] R. Heil. Strikethrough-Generation. `https://zenodo.org/record/4767062`, 2021. doi: 10.5281/ZENODO.4767062.

[4] R. Heil, E. Vats, and A. Hast. *Strikethrough Removal from Handwritten Words Using CycleGANs*, page 572–586. Springer International Publishing, 2021. doi: 10.1007/978-3-030-86337-1_38.

[5] N. Majid and E. H. Barney Smith. Segmentation-Free Bangla Offline Handwriting Recognition using Sequential Detection of Characters and Diacritics with a Faster R-CNN. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, Sept. 2019. doi: 10.1109/icdar.2019.00045.

[6] N. Majid and E. H. Barney Smith. Character spotting and autonomous tagging: offline handwriting recognition for Bangla, Korean and other alphabetic scripts. *International Journal on Document Analysis and Recognition (IJDAR)*, 25(4):245–263, Sept. 2022. doi: 10.1007/s10032-022-00410-x.

[7] S. Majumder, S. Ghosh, S. Malakar, R. Sarkar, and M. Nasipuri. A voting-based technique for word spotting in handwritten document images. *Multimedia Tools and Applications*, 80(8):12411–12434, Jan. 2021. doi: 10.1007/s11042-020-10363-0.

[8] S. Malakar, M. Ghosh, R. Sarkar, and M. Nasipuri. Development of a Two-Stage Segmentation-Based Word Searching Method for Handwritten Document Images. *Journal of Intelligent Systems*, 29(1):719–735, July 2018. doi: 10.1515/jisys-2017-0384.

[9] U.-V. Marti and H. Bunke. The IAM-database: an English sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5(1):39–46, Nov. 2002. doi: 10.1007/s100320200071.

[10] R. Mondal, S. Malakar, E. H. Barney Smith, and R. Sarkar. Handwritten English word recognition using a deep learning based object detection architecture. *Multimedia Tools and Applications*, 81(1):975–1000, Sept. 2021. doi: 10.1007/s11042-021-11425-7.

[11] OECD.AI. Character Error Rate (CER). `https://oecd.ai/en/catalogue/metrics/character-error-rate-cer`, 2022.

[12] OECD.AI. Word Error Rate (WER). `https://oecd.ai/en/catalogue/metrics/word-error-rate-wer`, 2022.

[13] U. Pal, R. Jayadevan, and N. Sharma. Handwriting recognition in Indian regional scripts: a survey of offline techniques. *ACM Transactions on Asian Language Information Processing (TALIP)*, 11(1):1–35, 2012.

[14] U. Pal, R. K. Roy, and F. Kimura. Multi-lingual City Name Recognition for Indian Postal Automation. In *2012 International Conference on Frontiers in Handwriting Recognition*. IEEE, Sept. 2012. doi: 10.1109/icfhr.2012.238.

[15] R. Plamondon and S. Srihari. Online and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):63–84, 2000. doi: 10.1109/34.824821.

[16] A. Poddar, A. Chakraborty, J. Mukhopadhyay, and P. K. Biswas. *Detection and Localisation of Struck-Out-Strokes in Handwritten Manuscripts*, page 98–112. Springer International Publishing, 2021. doi: 10.1007/978-3-030-86159-9_7.

[17] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[18] R. Rothe, M. Guillaumin, and L. Van Gool. Non-maximum suppression for object detection by passing messages between windows. In D. Cremers, I. Reid, H. Saito, and M.-H. Yang, editors, *Computer Vision – ACCV 2014*, pages 290–306, Cham, 2015. Springer International Publishing.

[19] R. Varghese and S. M. Yolov8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness. In *2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*. IEEE, Apr. 2024. doi: 10.1109/adics58448.2024.10533619.

[20] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2223–2232, 2017. doi: 10.48550/ARXIV.1703.10593.

[21] M. Zimmermann and H. Bunke. Automatic segmentation of the IAM off-line database for handwritten English text. In *2002 International Conference on Pattern Recognition*, volume 4, pages 35–39. IEEE, 2002. doi: 10.1109/ICPR.2002.1047394.

Acronyms

CER Character Error Rate. 14, 15

CycleGAN Cycle Consistent Generative Adversial Network. 4

GAN Generative Adversial Network. 3

HMM Hidden Markov Model. 3

IoU Intersection Over Union. 6

LSTM Long Short-Term Memory. 3

NMS Non-Maximum Suppression. 6

OCR Optical Character Recognition. 4, 17

WER Word Error Rate. 14, 15

YOLO You Only Look Once. i, 1, 5, 6, 11, 16, 17