Handwriting Detection & Recognition

By

Divya Kumar Karan (561)

Khushal Sharma (570)

Under the Supervision of

Dr Sanjoy Pratihar

submitted to

Indian Institute of Information Technology Kalyani

for the partial fulfilment of the degree of

Bachelor of Technology

in

Computer Science and Engineering/Information Technology

Spring Semester, 2024

Abstract

Handwriting recognition (HWR) is a crucial technology that plays a pivotal role in converting handwritten text into a digital format, facilitating computer processing and analysis.

HWR is widely employed in diverse applications, ranging from check processing and document imaging to mobile computing.

A significant challenge encountered in the domain of HWR is the substantial variation in handwriting styles exhibited by individuals. This variation poses a considerable obstacle for recognition systems, which strive to accurately identify and transcribe handwritten text.

This report is aimed at exploring the potential of traversal, a systematic image analysis technique, to enhance the accuracy of HWR systems. Additionally, it offers an in-depth analysis of the utilization of traversal for handwriting recognition detection and recovery.

Contents

Chapter 1
Introduction

Handwriting recognition, often abbreviated as HWR, is the process of converting handwritten text into a digital format that can be processed by computers. The applications of HWR are widespread and encompass domains such as finance (e.g., check processing), document management (e.g., document imaging and OCR), and mobile computing. The primary objective of HWR systems is to bridge the gap between the analogue and digital worlds, making handwritten text machine-readable and enabling advanced processing, storage, and analysis.

One of the most formidable challenges encountered in the field of HWR is the remarkable diversity in handwriting styles. Handwriting varies significantly from person to person, and even individual variations within a single person's writing can be substantial. This variance makes it particularly challenging for HWR systems to consistently and accurately recognise all instances of handwritten text.

In addressing this challenge, one promising approach is the use of traversal, a systematic technique that involves moving through a handwritten text image in a specific order to identify patterns and characters. Traversal offers a structured methodology for analyzing handwritten text images, enabling the identification of connected components, feature extraction, and character matching.

Chapter 2
Image Pre-Processing

Pre-processing is a sequence of operations, that improves the quality of the image and hence increases the accuracy of the image. For the hand-written character recognition process, the following pre-processing techniques are followed.

a) **Noise-removal:** It is the process of removing noise from the image. This also refers to smoothening the image by reducing the unwanted signals in the image such as removal of creases from photocopy of paper which was once folded passing through some texts which would affect the image/blob analysis for text detection. There exist many algorithms to remove noises in the image. Some of them are the Gaussian filtering method, Min-max filtering method, Median filter etc.

b) **Binarization:** It is a mechanism of converting grey-scale or coloured images to binary images. Binary images contain only 0's and 1's. The pixels in images are partitioned to 0's or 1's based on some constant value. If the pixel value is less than the constant, it is replaced with 0 or else with 1.

c) **Morphological operation:** This is the process of increasing or decreasing the size of an image. This is done mainly because the algorithm would expect a constant image size. To increase the size of an image, we can add pixels to the boundary of the image. To decrease the size of an image, we can remove pixels from the boundary of the image.

d) **Segmentation:** Segmentation in image processing refers to the process of partitioning an image into multiple segments or regions to simplify its representation, making it easier to analyze and understand. The goal of segmentation is to divide an image into meaningful parts based on certain criteria, such as color, intensity, texture, or spatial proximity.

Present techniques for line separation and then word segmentation using a neural network. However, existing word segmentation strategies have certain limitations:

1. Almost all the above methods require binary images. Also, they have been tried only on clean white self-written pages and not manuscripts.

2. Most of the techniques have been developed for machine printed characters and not handwritten words. The difficulty faced in word segmentation is in combining discrete characters into words.

3. Most researchers focus only on word recognition algorithms and considered a database of clean images with well segmented words. Only a few have performed full, handwritten page segmentation.

4. Efficient image binarization is difficult on manuscript images containing noise and shine through.

5. Connected ascenders and descenders have to be separated.

6. Prior character segmentation was required to perform word segmentation and accurate character segmentation in cursive writing is a difficult problem.

Approach 1: Line Segmentation & Word segmentation using scale space/blob analysis

Scale space theory deals with the importance of scale in any physical observation ie. objects and features are relevant only at particular scales. In scale space, starting from an original image, successively smoothed images are generated along the scale dimension. It has been shown by several researchers that the Gaussian uniquely generates the linear scale space of the image when certain conditions are imposed.

Line segmentation allows the ascenders and descenders of consecutive lines to be separated. In the manuscripts it is observed that the lines consist of a series of horizontal components from left to right. Projection profile techniques have been widely used in line and word segmentation for machine printed documents.

In this technique a 1D function of the pixel values is obtained by projecting the binary image onto the horizontal or vertical axis. We use a modified version of the same algorithm extended to gray scale images. Let f(x, y) be the intensity value of a pixel (x,y) in a gray scale image. Then, we define the vertical projection profile as

$$P(y) = \sum_{x=0}^{W} f(x, y)$$

where W is the width of the image. Fig. 1 shows a section of an image in (a) and its projection profile in (b). The distinct local peaks in the profile corresponds to the white space between the lines and distinct local minima corresponds to the text (black ink). Line segmentation, therefore, involves detecting the position of the local maxima. However, the projection profile has a number of false local maxima and minima. The projection function P(y) is therefore, smoothed with a Gaussian (low pass) filter to eliminate false alarms and reduce sensitivity to noise. A smoothed profile is shown in (c). The local maxima is then obtained from the first derivative of the projection function by solving for y such that :

$$\mathbf{P'(y) = P(y)*G_y = 0}$$

The next step after line segmentation is to create a scale space of the line images for blob analysis. The sigma(σ) in Gaussian kernel is equal to $\sqrt{2t}$, where t is the scale parameter & t > 0.

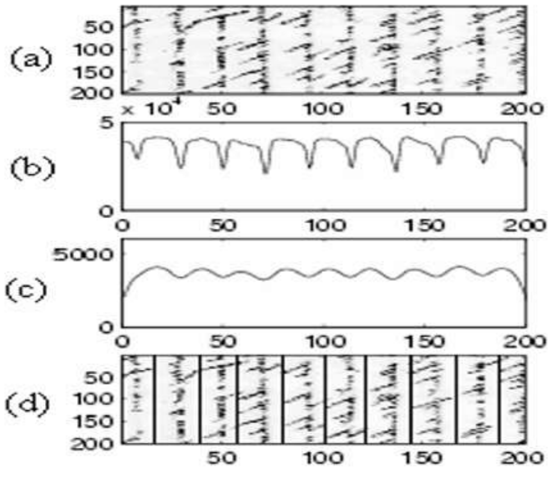


Fig. 1: (a) A section of an image, (b) projection profile, (c) smoothed projection profile (d) line segmented image

Now we examine each line image individually to extract the words. A word image is composed of discrete characters, connected characters or a combination of the two. We would like to merge these sub-units into a single meaningful entity which is a word. This may be achieved by forming a blob-like representation of the image. A blob can be regarded as a connected region in space. The traditional way of forming a blob is to use a Laplacian of a Gaussian (LOG), as the LOG is a popular operator and frequently used in blob detection and a variety of multi-scale image analysis tasks. We have used a differential expression similar to a LOG for creating a multi-scale representation for blob detection. However, our differential expression differs in that we combine second order partial Gaussian derivatives along the two orientations at different scales.

$$G(x, y; \sigma_x, \sigma_y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\left(\frac{x^2}{2\sigma_x^2} + \frac{y^2}{2\sigma_y^2}\right)}$$

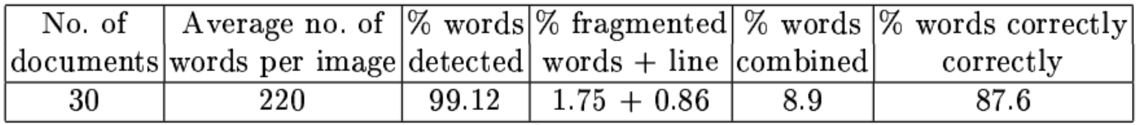
In the scale selection section we will show that the average aspect ratio or the multiplication factor lies between three and five for most of the handwritten documents available to us. Also the response of the anisotropic Gaussian filter (measured as the spatial extent of the blobs formed) is maximum in this range. For the above Gaussian, the second order anisotropic Gaussian differential operator is defined as

$$L(x, y; \sigma_x, \sigma_y) = G_{xx}(x, y; \sigma_x, \sigma_y) + G_{yy}(x, y; \sigma_x, \sigma_y)$$

A scale space representation of the line images is constructed by convolving the image with L. Consider a two dimensional image f(z, y), then the corresponding output image is

$$I(x, y; \sigma_x, \sigma_y) = L(x, y; \sigma_x, \sigma_y) \star f(x, y)$$

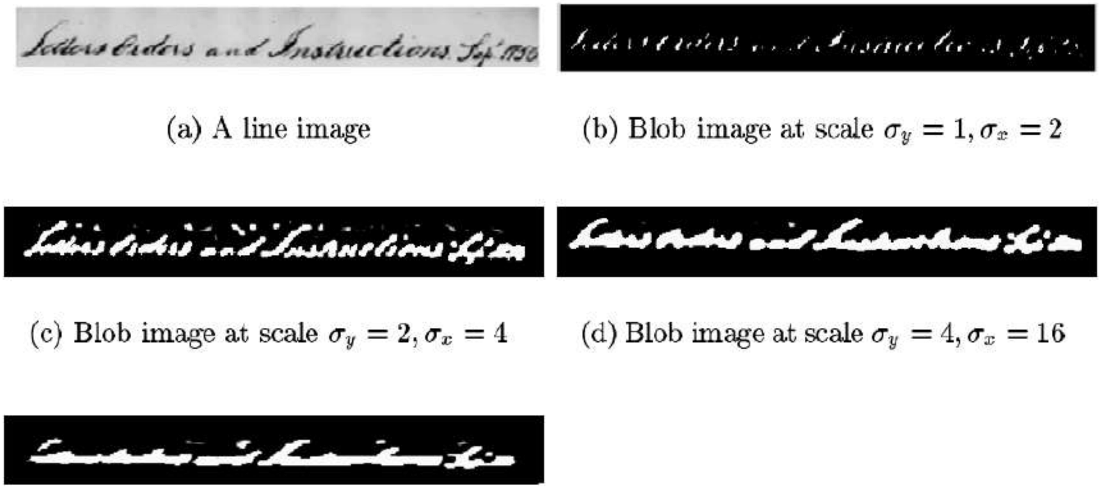
The main features which arise from a scale space representation are blob-like (i.e. connected regions either brighter or darker than the background). The sign of I may then be used to make a classification of the 3-D intensity surface into foreground and background. For example consider the line image in Fig. 2(a). The figures show the blob images I at increasing scale values. Fig. 2(b) shows that at a lower scale the blob image consists of character blobs. As we increase the scale, character blobs give rise to word blobs (Fig. 2(c) and Fig. 2(d)). This is indicative of the phenomenon of merging in blobs. It is seen that for certain scale values the blobs and hence the words are correctly delineated (Fig. 2(d)). A further increase in the scale value may not necessarily cause word blobs to merge together and other phenomenon such as splitting is also observed. These figures show that there exists a scale at which it is possible to delineate most words.



No. of documents	Average no. of words per image	% words detected	% fragmented words + line	% words combined	% words correctly correctly
30	220	99.12	1.75 + 0.86	8.9	87.6

Table 1: Table of segmentation results

A segmentation accuracy ranging from **77-96%** with an average accuracy around **87.6%** was observed. Fig. 4 shows part of a segmented page image with bounding boxes drawn on the extracted words. The method worked well even on faded, noisy images and Table 1 shows the results averaged over a set of 30 images. The first column indicates the average no. of distinct words in a page as seen by a human observer. The second column indicates the % of words detected by the algorithm i.e, words with a bounding box around them, this includes words correctly segmented, fragmented and combined together. The next column indicate the % of words fragmented. Word fragmentation occurs if a character or characters in a word have separate bounding boxes, or if 50 percent or greater of a character in a word is not detected. Line fragmentation occurs due to the dissection of the image into lines. A word is line fragmented if 50 % or greater of a character lies outside the top or bottom edges of the bounding box. The sixth column indicates the words which are combined together. These are multiple words in the same bounding box. The last column gives the percentage of correctly segmented words.



(a) A line image

(b) Blob image at scale $\sigma_y = 1, \sigma_x = 2$

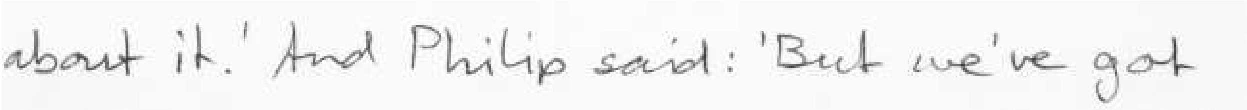
(c) Blob image at scale $\sigma_y = 2, \sigma_x = 4$

(d) Blob image at scale $\sigma_y = 4, \sigma_x = 16$

(e) Blob image at scale $\sigma_y = 6, \sigma_x = 36$

Fig. 2: A line image and the output at different scales

1. **Original image:**

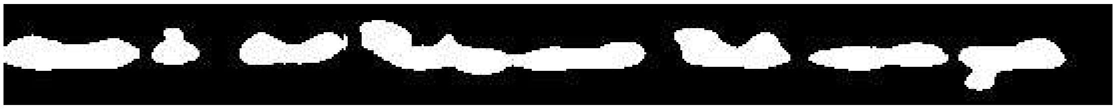

about it.' And Philip said: 'But we've got

⇓

2. **Filtered Image using Gaussian filter**

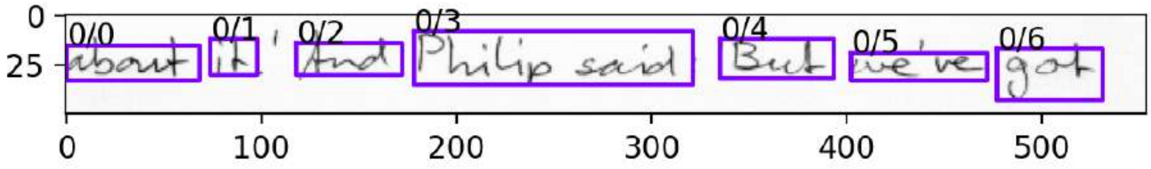
⇓

3. **Threshold image blob analysis:**



⇓

4. **Extracted word from input image:**



Approach 2: Word segmentation using scale space/blob analysis & using clustering algorithm to preserve the order of words (OUR APPROACH)

Here we first perform word segmentation unlike above approach, but using the similar procedure in approach 1 and then using the clustering algorithm preserve the ordering of detected words & thus lines are also preserved.

Clustering Bounding Boxes with DBSCAN algorithm

Clustering bounding boxes corresponding to text regions in an image is an essential task to group related text elements together.

Here, we describe the process of clustering bounding boxes using the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm, with a focus on how Jaccard distances are used to measure the similarity between bounding boxes.

1. Jaccard Distance Calculation:

- Jaccard distances are computed between pairs of bounding boxes using the intersection over union (IoU) metric. This measures the extent of overlap between bounding boxes.

- The Jaccard distance matrix is constructed, where each element represents the dissimilarity between two bounding boxes.

2. DBSCAN Clustering:

- The computed Jaccard distances serve as input to the DBSCAN clustering algorithm.

- Parameters such as ‘max_dist‘ and ‘min_words_per_line‘ are specified to control the clustering behavior.

- DBSCAN iteratively expands clusters based on the Jaccard distances, grouping together bounding boxes that are within the specified distance threshold (‘max_dist‘).

- Bounding boxes that cannot be assigned to any cluster are considered outliers.

Example:

Let's consider an example with four bounding boxes labeled A, B, C, and D, and the following Jaccard distance matrix:

A B C D

A [0.0, 0.1, 0.4, 0.6]

B [0.1, 0.0, 0.3, 0.5]

C [0.4, 0.3, 0.0, 0.2]

D [0.6, 0.5, 0.2, 0.0]

- Applying DBSCAN with **max_dist(epsilon) = 0.3** and **min_words_per_line(minPoints) = 2**, clusters are formed as follows:

- Cluster 1: {A, B, C}

- Outliers: {D}

- Bounding boxes A, B, and C are grouped into Cluster 1, representing a line of text.

- Bounding box D is considered an outlier and does not belong to any cluster.

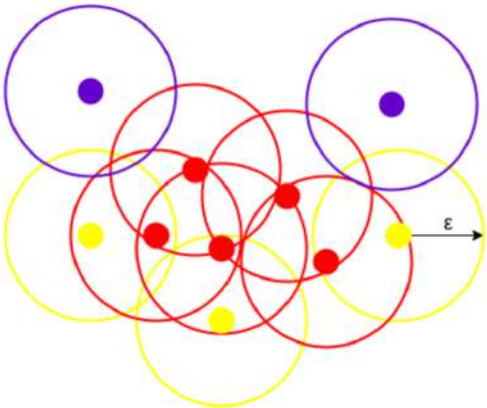


Fig. 3: Cluster of points using DBSCAN

Fig. 3 shows us a cluster created by DBCAN with *minPoints = 3*. Here, we draw a circle of equal radius *epsilon* around every data point. These two parameters help in creating spatial clusters. Here, we see 3 different kinds of points: ***noise points***, ***border points*** & ***cluster points***.

Clustering bounding boxes using DBSCAN with Jaccard distances offers an effective approach to grouping related text elements in an image. By leveraging similarity measures such as Jaccard distances, DBSCAN can effectively identify text regions and provide valuable insights for further text analysis tasks & ordering of words line by line.

Then we simply sort the line & words in every line using the property of sorting based on y-axis coordinate & x-axis coordinate respectively of the bounding boxes.

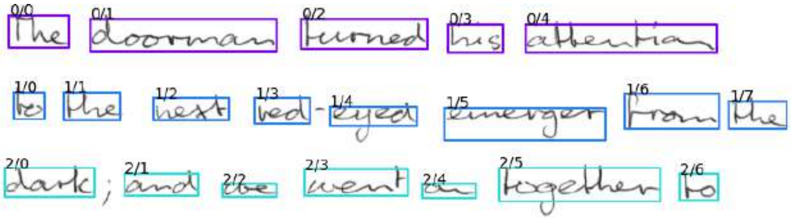


Fig. 4: Word Segmentation of a document

Why did not we use other clustering algorithms such as K-means clustering or Hierarchical clustering instead of DBSCAN?

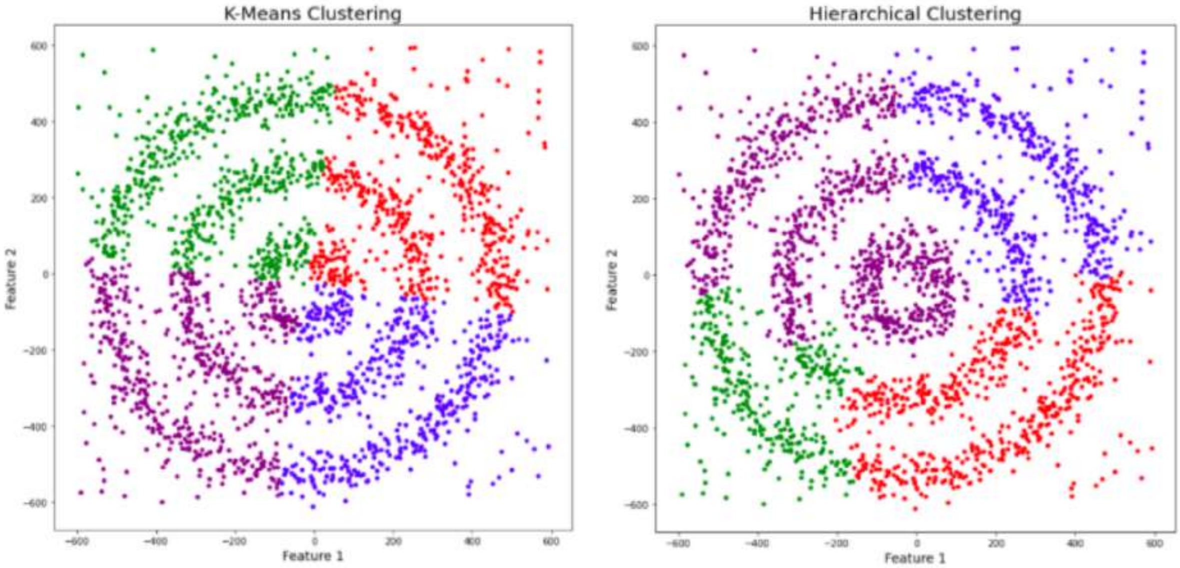


Fig. 5: (a) K-Means Cluster (b) Hierarchical Cluster

As visible in Fig. 5 (a) & (b), Both of them failed to cluster the data points. Also, they were not able to properly detect the noise present in the dataset. Now, let's take a look at the results from **DBSCAN clustering** in Fig. 6.

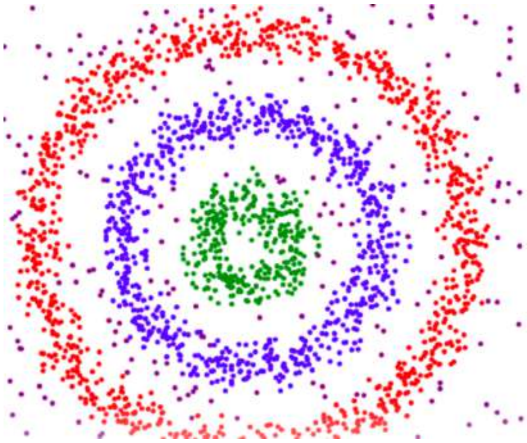


Fig. 6: DBSCAN cluster

Approach 3: Word segmentation using neural network & using clustering algorithm to preserve the order of words

Clustering algorithm to preserve the order of words technique remains same as above.

Word Detector Neural Network with Residual Training

In the field of computer vision, detecting words or text regions in images is a fundamental task with applications in document analysis, OCR (Optical Character Recognition), and scene understanding. Convolutional Neural Networks (CNNs) have shown remarkable performance in object detection tasks, including text detection. One popular technique used to improve the training and performance of CNNs is residual training, which involves the use of residual connections to alleviate the vanishing gradient problem and enable the training of deeper networks.

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}.$$

1. Residual Training:

- Residual training is a technique introduced in the ResNet architecture to address the degradation problem encountered when training very deep neural networks.

- It involves the use of residual blocks, which contain shortcut connections (skip connections) that bypass one or more layers in the network as shown in Fig. 7.

- By using these skip connections, residual networks can learn identity mappings and effectively train deeper networks.

-When there is a dimension increase during shortcut connections, we consider two options: (A) The shortcut still performs identity mapping, with extra zero entries padded for increasing dimensions. This option introduces no extra parameter; (B) The projection shortcut is used to match dimensions (done by 1×1 convolutions). For both options, when the shortcuts go across feature maps of two sizes, they are performed with a stride of 2.

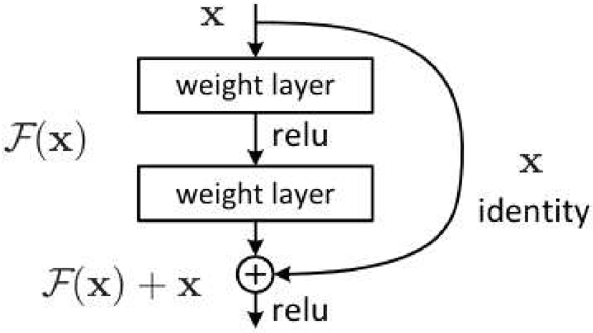

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Fig. 7: Residual Learning: a building block

2. Neural Network Architecture:

- The word detector neural network is based on a CNN architecture, possibly with additional layers such as feature extraction, convolutional layers, and fully connected layers.

- Residual connections are incorporated into the network to enable residual training as represented in Fig. 8.

- The network is trained end-to-end using a large dataset of labeled images containing word-level annotations.

3. Training Process:

- The neural network is trained using backpropagation with gradient descent optimization algorithms such as Adam or SGD (Stochastic Gradient Descent).

- During training, the loss function (e.g., cross-entropy loss) is minimized to learn the parameters of the network.

- Batch normalization and dropout may be used to improve the generalization performance of the network and prevent overfitting.

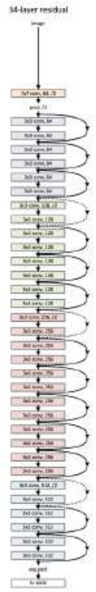


Fig. 8: Deep Residual Neural Network

4. Evaluation Metrics:

- Performance of the word detector neural network is evaluated using standard metrics such as precision, recall, and F1-score.

- Additionally, metrics such as Intersection over Union (IoU) may be used to evaluate the accuracy of bounding box predictions.

The word detector neural network using residual training offers an effective solution for word detection tasks in natural images. By leveraging residual connections to train deeper networks, the model can learn more expressive features and achieve superior performance in detecting words in complex scenes.

COMPARISION STUDY

- **Approach 1 vs. Approach 2 :**

  The main cause of difference in both approaches is the amount of dependency on scale variance.

  In approach 1, scale variance is more due to choosing of scale parameters for both word and line segmentation techniques, which may or may not work in some cases. Also, this approach is prone to detecting false lines, which affects accuracy.

  In approach 2, scale variance is least but not negligible, due to choosing of scale parameter only for word segmentation technique, which has a limited range and DBSCAN algorithm has negligible effect on scale variance. We, thus reduced the **% of fragmented line** in approach 1, i.e. **0.86%** to negligible, i.e. **0.0001%**, thus increasing overall accuracy to **88.5%**. This algorithm gives good results on datasets with large inter-word-distances and small intra-word-distances like IAM.

  So, of course we can rely on approach 2 as it would give the most accurate results & the segmented words can act as input to our model, one by one & thus, we would convert a passage of handwritten words into passage of computerised text.

  However, approach 2 doesn't work for more complex handwritings such as Bentham handwriting(Fig. 9), computer-like handwriting, etc. which would be discussed in further sections.

- **Approach 2 vs. Approach 3 :**

  Between approach 2 & approach 3, approach 3 solves for complex handwritings having low inter-word distance & high intra-word distances and also is scale invariant which approach 2 is unable to work for, but for most cases the above kind of handwritings are sometimes not legible to be read by the humans & thus rarely such handwriting is used. Also, computerised handwritings can be just recognized through OCRs available in market. So, to maintain efficacy of system & reduction of overhead for handling network weights, we would use approach 2 for almost all handwritings. Also, accuracy for both approaches are nearly similar (**approach 2: 88.5% & approach 3: 91.8%**).

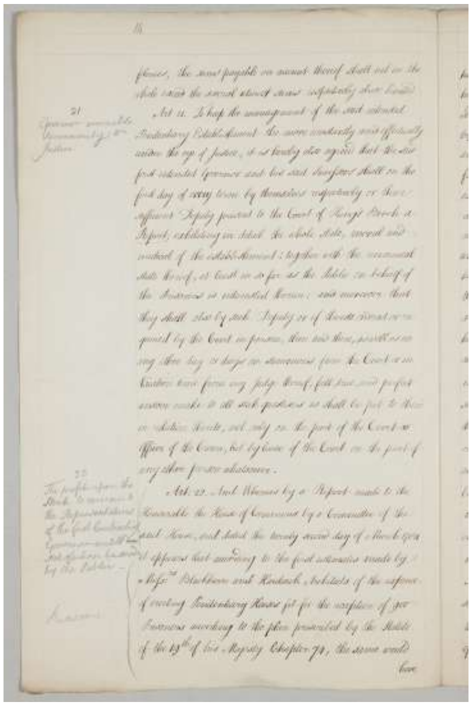


Fig. 9: Bentham Handwritings

Chapter 3
Naive Approach

Stroke traversal is a systematic technique aimed at improving HWR systems' accuracy. The naive approach using stroke traversal is based on the following principles:

1. **Stroke Segmentation:** The first step in this approach is to segment each handwritten character into individual strokes. A stroke is defined as a continuous line or series of lines created by a single movement of the writing instrument (e.g., pen or stylus).

2. **Order and Direction Analysis:** The approach focuses on analyzing the order and direction of strokes used to construct each character. This information is vital because it reflects the unique characteristics of how a particular character is formed.

This is usually done using a **varied version of the Freeman Chaining algorithm.**

3. **Stroke Combination:** After identifying the individual strokes and their order, the approach combines the strokes in the correct sequence to form the recognized character.

4. **Feature Extraction:** Features are extracted from each stroke, considering factors such as stroke length, angle, and curvature. These features contribute to distinguishing one character from another, enhancing the accuracy of recognition.

5. **Matching to Stroke Databases:** Stroke databases can be used to compare segmented strokes with known stroke patterns. This step helps identify and recognize the characters based on the detected strokes.

6. **Character Reconstruction:** Once the individual strokes are matched and recognized, the approach reconstructs the complete handwritten character by assembling the strokes in the correct order.

Advantages:

- Simplicity: The naive stroke traversal approach is straightforward and relatively easy to implement. It doesn't require complex machine learning algorithms, making it accessible for basic HWR applications.

- Stroke-Level Analysis: By focusing on individual strokes, the approach can capture subtle details of how characters are formed, potentially improving accuracy.

- Low Computational Cost: Since it doesn't rely on intensive machine learning models, the approach is computationally efficient.

Limitations:

- Limited Context: This approach may struggle with recognizing characters that rely on contextual information or the relative positions of strokes. For ex: Consider the character "a" in cursive handwriting. The shape of the cursive "a" can vary significantly depending on its context within a word. Recognizing it accurately may be challenging without considering the surrounding letters.

- Overreliance on Stroke Order: In cases where stroke order is not a reliable distinguishing feature (e.g., cursive writing), the naive approach may be less effective. It's ineffective in Offline handwriting recognition. For ex: In some forms of artistic or stylized writing, stroke order may not follow conventional rules. For instance, a calligraphic representation of the letter "S" might have a different stroke order than the standard approach, leading to potential recognition errors.

- Handwriting Variability: While stroke traversal addresses some aspects of handwriting variation, it may not fully account for diverse writing styles. For ex: Take the letter "g" written by different individuals. Some may write it with a single closed loop, while others may have an open loop. Handwriting styles can vary widely, and a rigid stroke traversal approach may struggle to adapt to these variations.

- Not Suitable for All Characters: Certain complex characters or ligatures may not be adequately represented using this approach. For ex: Consider a complex Chinese character with intricate details. The stroke traversal method may not adequately represent the subtleties of the character's components, leading to potential misinterpretation or misclassification.

Applications of the Naive Stroke Traversal Approach

The naive stroke traversal approach is best suited for applications where simplicity and efficiency are prioritized over complex machine learning models. It can find use in:

- Basic HWR systems for limited character sets.
- Educational tools for teaching handwriting.
- Recognition of isolated characters or simple symbols.
- Systems with limited computational resources.

Chapter 4
Advanced Approach

Traversal has several significant applications and advantages in the realm of handwriting recognition:

1. **Connected Component Identification:**

- Description: This method involves systematically identifying connected components within a document image. Connected components are groups of connected pixels in the image that represent individual characters or symbol groups. Recognizing these connected components is a crucial step in breaking down the complex document image into manageable and distinct units.

- Process:

- Image Analysis: The traversal techniques analyze the document image pixel by pixel, starting from a specific point or region of interest.

- Connected Component Detection: As the traversal progresses, it detects clusters of connected pixels. These clusters can represent individual characters or parts of characters.

- **Segmentation:** The connected components are segmented from the rest of the image, effectively isolating them for further analysis.

Thresholding an image is a fundamental image processing technique used to segment an image into regions or objects of interest by separating the foreground (object) from the background. The primary purpose of thresholding is to simplify the image and extract relevant information.

For this, we mostly use Otsu's thresholding method for best results.

- Importance: Identifying connected components is essential as it serves as the foundational step for character recognition. It helps in distinguishing individual characters from one another within the document image.

2. **Feature Extraction:**

- Description: After the connected components are identified, the next step is to extract relevant features from each of these components. Features are characteristics or attributes that can be used to distinguish one character from another.

- Process:

- Feature Selection: Depending on the project's goals and the complexity of the handwriting, various features may be considered. Common features include the number of line segments in a character, the length of line segments, the curvature of line segments, or other properties that differentiate characters effectively.

- Feature Extraction: For each connected component (character), the selected features are computed and extracted. For instance, if the number of line segments is a feature, the system would count the line segments within the character.

- Importance: Feature extraction plays a crucial role in characterizing and distinguishing handwritten symbols. These features are used to train a machine-learning model to recognize characters in the image. By extracting relevant information from each character, the system can make informed decisions during recognition.

3. **Machine Learning Model:**

- Description: The extracted features are used to train a machine learning model, which is designed to recognize characters within the image based on these features. Machine learning, including deep learning techniques, may be employed.

- Process:

- Feature Vector Creation: The extracted features from all connected components in the training dataset are used to create feature vectors. Each vector represents a character and contains its feature values.

- Model Selection: A machine learning model, such as a neural network for deep learning using a Convolutional Neural Network, is chosen based on the project's requirements and complexity.

- Training: The model is trained on the feature vectors, learning the relationships between the extracted features and the corresponding characters.

- Validation and Testing: The trained model is then validated and tested on new data to ensure its accuracy and generalization.

- Importance: The machine learning model is the core component responsible for character recognition. It utilizes the learned patterns from the extracted features to classify and recognize characters in the document image.

4. **Character Reconstruction:**

- Description: Once the individual characters are identified through machine learning, traversal techniques are employed again to systematically reconstruct the handwritten text. This process involves assembling the recognized characters correctly to produce the complete text.

- Process:

- Traversal: Similar to the connected component identification stage, the traversal process navigates through the document image, but this time, it follows the order and direction of the identified characters.

- Character Ordering: The characters are ordered correctly by following the logical progression of the handwriting, for instance, from left to right.

- **Text Assembly:** The recognized characters are assembled together to form the complete handwritten text.

- Importance: Character reconstruction is the final step in converting the segmented and recognized characters back into the original handwritten text. This step ensures the accuracy of the reconstructed text by preserving the correct order and arrangement of the characters.

In summary, these methods, when combined, create a comprehensive system for handwriting recognition detection and recovery using traversal.

It involves the initial identification of connected components, extraction of distinguishing features, training a machine learning model to recognize characters, and ultimately reconstructing the handwritten text accurately.

This approach provides a holistic solution to the challenges posed by recognizing handwritten text in various applications.

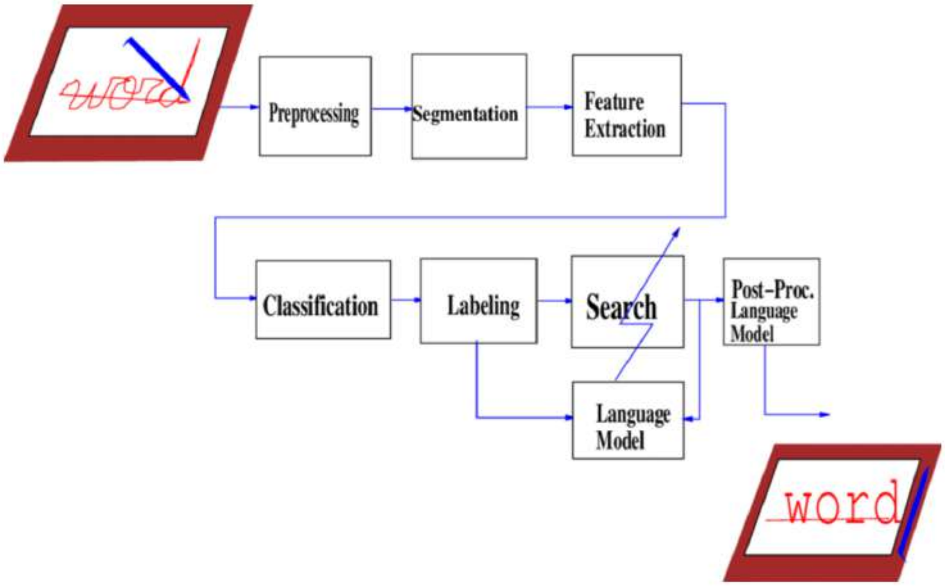


Fig. 10: Steps involved in modelling a handwriting to text recognition system

Chapter 5

Handwritten Text Recognition with Convolutional Recurrent Neural Network (CRNN)

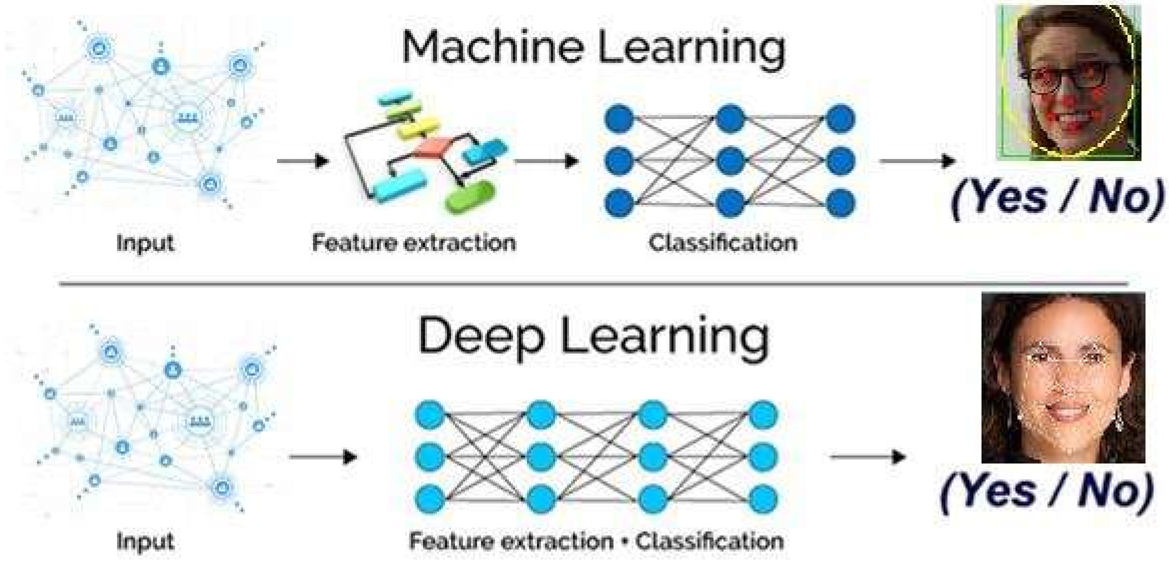


Fig. 11: Differentiating between machine learning & deep learning model

Handwritten text recognition (HTR) is a challenging task in computer vision due to the variability in writing styles and the inherent complexity of human handwriting. Traditional HTR methods typically involve preprocessing steps, such as the segmentation of words or characters, followed by feature extraction and classification.

Convolutional Recurrent Neural Networks (CRNNs) have emerged as a powerful approach for HTR, offering a more robust and end-to-end solution. CRNNs combine the strengths of convolutional neural networks (CNNs) and recurrent neural networks (RNNs) to effectively extract spatial features and capture temporal dependencies in handwritten text.

CRNN Architecture for HTR

The CRNN architecture for HTR consists of three main stages:

1. **Multi-scale feature extraction:** The input handwritten line text image is passed through a series of convolutional layers to extract relevant features at multiple scales. This stage reduces the spatial dimensionality of the input while preserving important information for character recognition(Fig. 12).
2. **Sequence labelling (BLSTM-CTC):** The extracted features from the CNN layers are then fed into a bidirectional long short-term memory (BLSTM) network. BLSTM is a type of RNN that can process sequential data in both forward and backward directions, allowing it to capture long-range dependencies in the handwritten text. The BLSTM output is then passed through a connectionist temporal classification (CTC) loss function, which effectively handles the alignment problem between the input image and the ground truth text(Fig. 13).
3. **Transcription:** The final stage involves decoding the output of the BLSTM-CTC network to obtain the recognized text. The CTC decode algorithm is used to find the most likely sequence of characters from the network's output probabilities(Fig. 14).

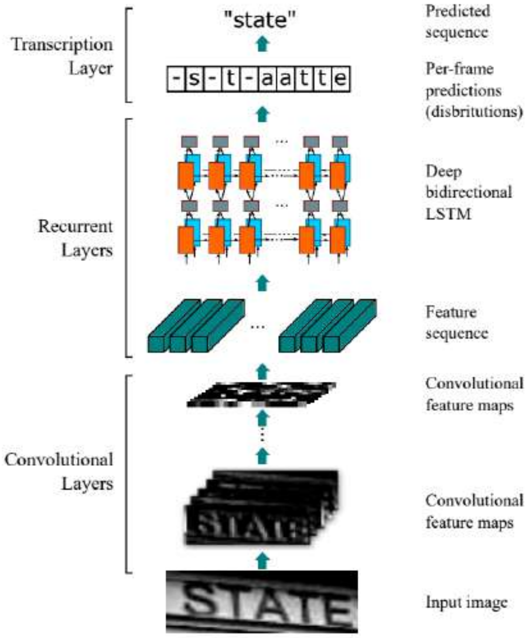


Fig. 12: CRNN architecture

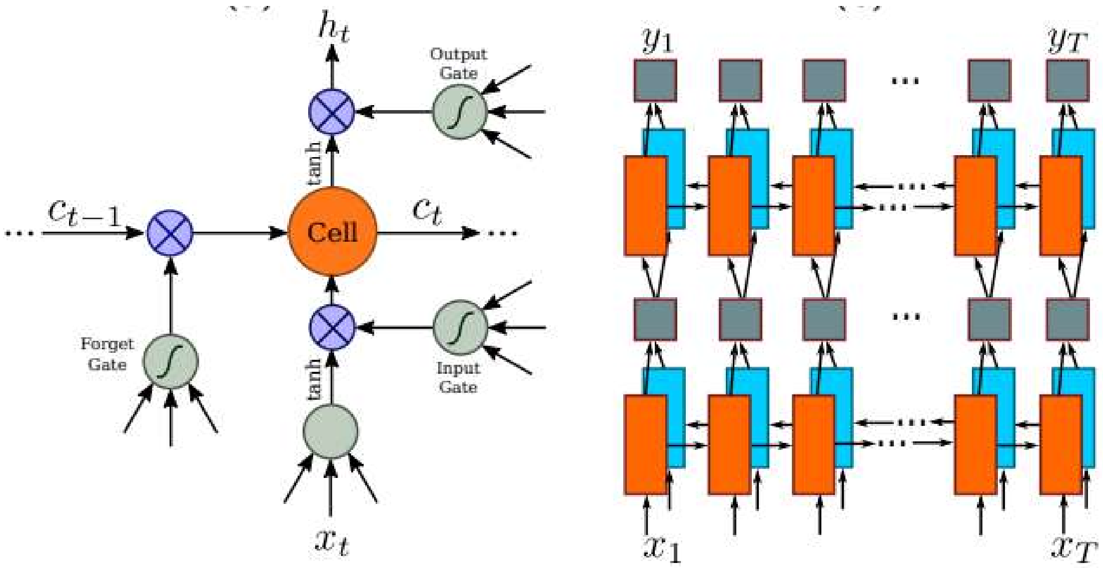
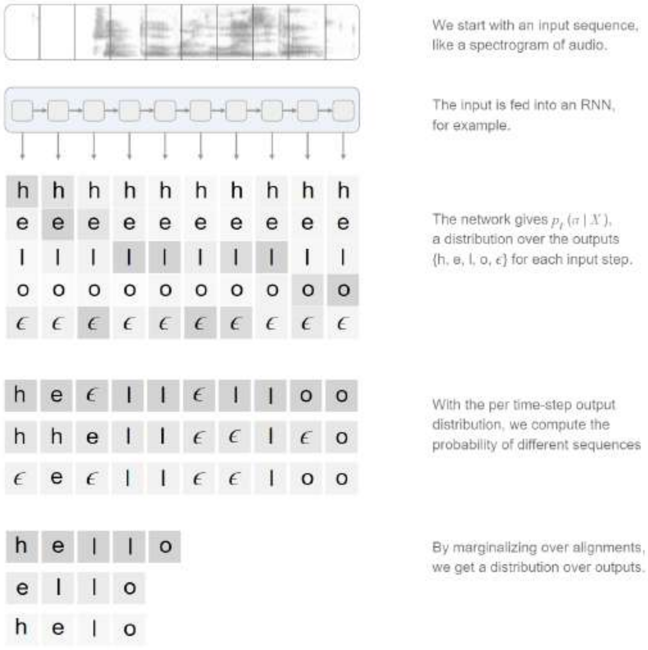


Fig. 13: LSTM cell & network



We start with an input sequence, like a spectrogram of audio.

The input is fed into an RNN, for example.

The network gives $p_t(a \mid X)$, a distribution over the outputs {h, e, l, o, ϵ} for each input step.

h	h	h	h	h	h	h	h	h	h
e	e	e	e	e	e	e	e	e	e
l	l	l	l	l	l	l	l	l	l
o	o	o	o	o	o	o	o	o	o
ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ

With the per time-step output distribution, we compute the probability of different sequences

h e ϵ l l ϵ l l o o

h h e l l ϵ ϵ l ϵ o

ϵ e ϵ l l ϵ ϵ l o o

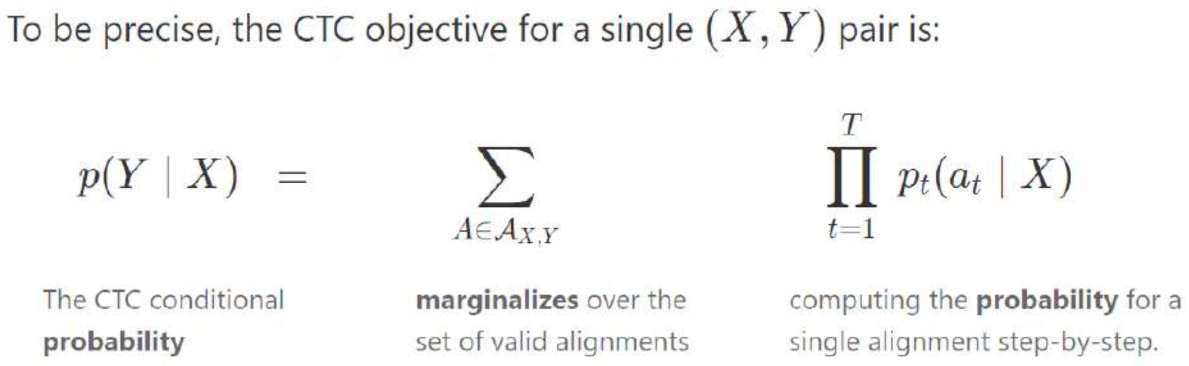
By marginalizing over alignments, we get a distribution over outputs.

h e l l o

e l l o

h e l o

Fig. 14: Process of determining the best possible label through CTC



To be precise, the CTC objective for a single (X, Y) pair is:

$$p(Y \mid X) \quad = \quad \sum_{A \in \mathcal{A}_{X,Y}} \quad \prod_{t=1}^{T} p_t(a_t \mid X)$$

The CTC conditional **probability**

marginalizes over the set of valid alignments

computing the **probability** for a single alignment step-by-step.

Minimizing the negative log of the above probability gives us the CTC Loss & a label with minimum CTC loss is the best prediction/result for a given input.

Benefits of CRNN for HTR

CRNNs offer several advantages for HTR compared to traditional methods:

1. **End-to-end learning:** CRNNs eliminate the need for pre-segmentation of words or characters, making the process more efficient and less prone to errors.
2. **Robustness to noise and variations:** CRNNs are less sensitive to noise and variations in handwriting due to their ability to capture both spatial and temporal information.
3. **Improved accuracy:** CRNNs have demonstrated superior accuracy in HTR tasks compared to traditional methods.

Results

- Here, when we train our model on **10 epochs**, we get the output for a batch test giving an accuracy of approximately **61%**(Fig. 15).

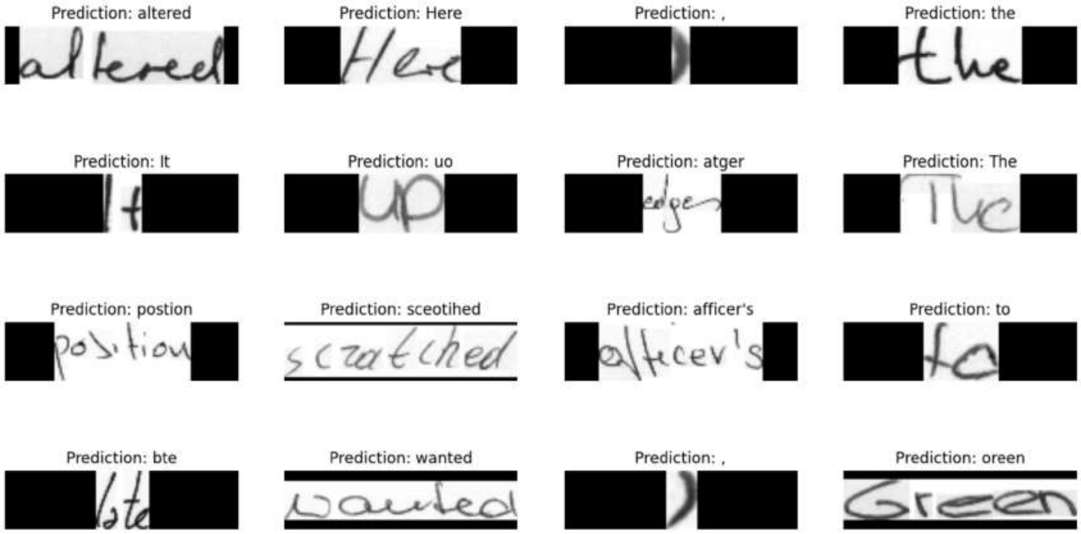


Fig. 15: Word Prediction on 10 epocs

- When we train our model on **60 epochs**, we get the output for a batch test giving an accuracy of approximately **75%**(Fig. 16).

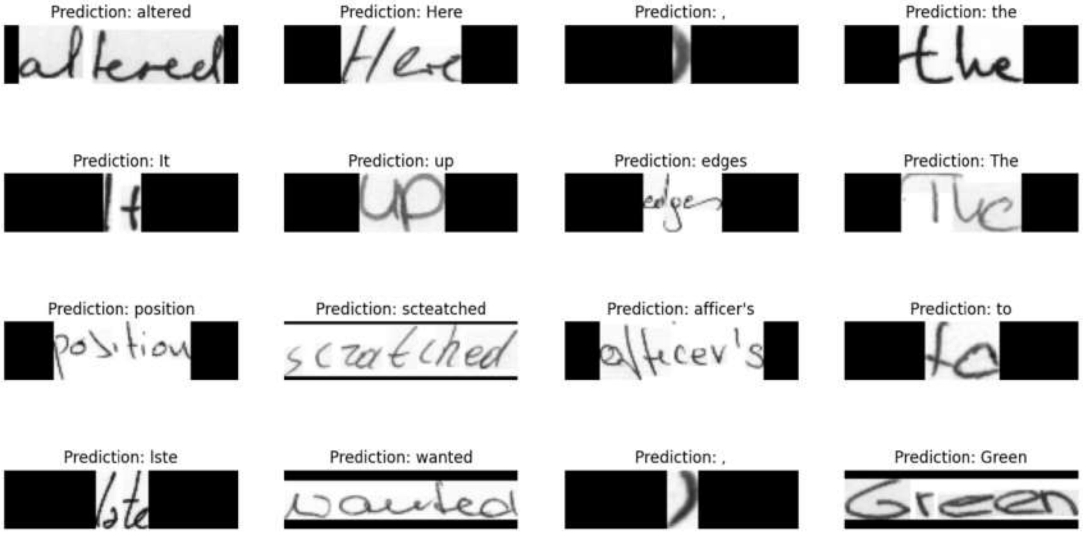


Fig. 16: Word Prediction on 60 epocs

So, we get better results after training the model on at least 50 epochs.

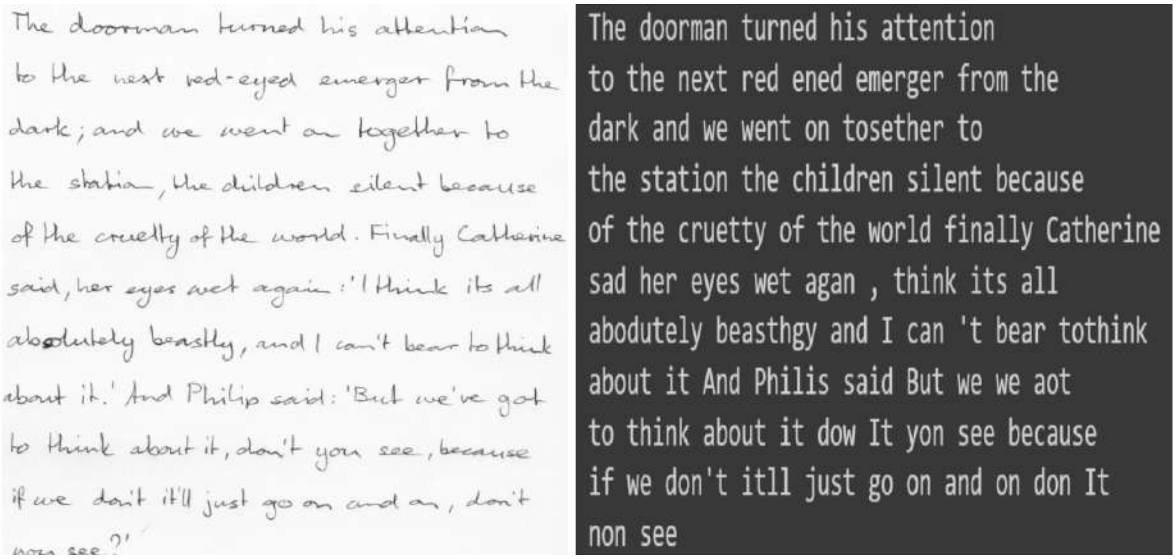
Conclusion

CRNNs have become a promising approach for HTR, offering improved accuracy and robustness compared to traditional methods. Their ability to extract features at multiple scales and capture long-range dependencies makes them well-suited for handling the complexities of handwritten text. As research in CRNNs continues to advance, we can expect further improvements in HTR performance, leading to more accurate and efficient text recognition systems.

Chapter 6

Results & Future Scope

Results:



The doorman turned his attention
to the next red-eyed emerger from the
dark; and we went on together to
the station, the children silent because
of the cruelty of the world. Finally Catherine
said, her eyes wet again: 'I think its all
absolutely beastly, and I can't bear to think
about it.' And Philip said: 'But we've got
to think about it, don't you see, because
if we don't it'll just go on and on, don't
you see?'

```
The doorman turned his attention
to the next red ened emerger from the
dark and we went on tosether to
the station the children silent because
of the cruetty of the world finally Catherine
sad her eyes wet agan , think its all
abodutely beasthgy and I can 't bear tothink
about it And Philis said But we we aot
to think about it dow It yon see because
if we don't itll just go on and on don It
non see
```

Fig. 17: Input Document & Output Results

Future Scope:

There are several avenues for future work in the field of handwriting recognition using traversal:

Multilingual Support: Expanding the system's capabilities to recognize handwritten text in various languages can be done by training models on various languages & simultaneously at the time of classification through CRNN, just feeding the pre processed data into suitable models.

This project serves as a strong foundation for these potential future endeavours in the field of handwriting recognition and offers promising possibilities for continued advancements in the accuracy and applicability of HWR systems.

References

1. Machine Learning for Handwriting Recognition[https://core.ac.uk/download/pdf/327266589.pdf]

2. Handwriting Trajectory Recovery using End-to-End Deep Encoder-Decoder Network [https://arxiv.org/ftp/arxiv/papers/1801/1801.07211.pdf]

3. An Overview of the Tesseract OCR Engine[https://tesseract-ocr.github.io/docs/tesseracticdar2007.pdf]

4. Sequence Modelling Sequence Modeling With CTC[https://distill.pub/2017/ctc/]

5. An Overview of the Tesseract OCR Engine[https://tesseract-ocr.github.io/docs/tesseracticdar2007.pdf]

6. Tesseract Vs Gocr A Comparative Study [https://www.ijrte.org/wp-content/uploads/papers/v2i4/D0788092413.pdf]

7. Scale Space Technique for Word Segmentation in Handwritten Documents [https://ciir.cs.umass.edu/pubfiles/mm-27.pdf]

8. Handwritten Word Detector[https://githubharald.github.io/word_detector.html]

9. Deep Residual Learning for Image Recognition[https://arxiv.org/pdf/1512.03385.pdf]

10. DBSCAN Clustering Algorithm for Machine Learning[https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works]